TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Heleriin Ots 155421 IAPB

TTÜ SATELLIIDIPROGRAMMI MAAJAAMADE PROJEKTI TARKVARALAHENDUSE LOOMINE

Bakalaureusetöö

Juhendaja: Evelin Halling
MSc

Tallinn 2018

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: Heleriin Ots

21.05.2018

Annotatsioon

Käesoleva töö põhieesmärgiks on andmebaasitäienduse, API, veebilehe ning töölauarakenduse loomine TTÜ Satelliidiprogrammi maajaamade projekti jaoks.

Maajaamade projekti eesmärgiks on tutvustada Eesti põhi- ning keskkooliõpilastele satelliiditehnoloogiaid. Kooliõpilastele korraldatakse töötubasid, kus ehitatakse antenn ning raadio, millega saavad õpilased püüda satelliitide poolt saadetavaid teateid. Maajaamade projekti raames peavad koolid saama saata teistele projektis osalevatele koolidele sõnumeid TTÜ satelliidi vahendusel. Lisaks sellele peavad õpilased saama püüda TTÜ satelliidi poolt saadetavaid laulutükke. Projekti raames toimuvate tegevuste ning osalevate koolide haldamiseks on vaja veebirakendust ning iseehitatud antenniga püütud sõnumite kuvamiseks on vaja töölauarakendust.

Töö käigus valmivad TTÜ satelliidi missiooni juhtimise süsteemi andmebaasi täiendus, kus hakatakse hoidma maajaamade projektiga seotud informatsiooni ning API maajaamade projektiga seotud informatsiooni haldamiseks. Lisaks sellele luuakse töö käigus veebileht ning töölauarakendus. Veebilehel kuvatakse projektiga seotud informatsiooni ning statistikat. Projektis osalevad koolid saavad saata sõnumeid ning hallata oma sõnumeid ja püütud laulutükke. Projekti korraldajad saavad registreerida uusi koole ning hallata projektis osalevaid koole. Töölauarakendus võimaldab näha omaehitatud antenniga püütud sõnumeid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 36 leheküljel, 8 peatükki, 18 joonist, 0 tabelit.

Abstract

The Establishment of a Software Implementation for the Ground Station Project of the TTU Satellite Program

The main objective of this bachelor thesis is to create a database supplement, an API, a website and a desktop application for the ground station project of the TTU Satellite program.

The aim of the ground station project is to introduce satellite technologies to Estonian middle and secondary school students. Workshops will be held as part of the project, in which students can build their own antenna and radio, with which they can receive messages sent by satellites. The schools must be able to send messages to other schools participating in the ground station project via the TTU satellite. In addition, students must be able to catch song pieces sent by the TTU satellite. In order to administrate the activities and schools participating in the project, a web application is needed and in order to display the messages caught by self-built antennas a desktop application is required.

As part of the bachelor thesis a supplement for the TTU satellite mission control database will be created, where the information regarding the ground station project will be held and an API for administrating the information of the ground station project. Furthermore, a website and a desktop application will be created as part of this bachelor thesis. The website will display information and statistics about the project. The schools participating in the project will be able to send messages and manage their messages and the song pieces they have caught. The administrators of the project will be able to register new schools to the system and administrate the schools participating in the project. The desktop application enables the students to see the messages caught by their self-built antenna.

The thesis is in Estonian and contains 36 pages of text, 8 chapters, 18 figures, 0 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*. Rakendusliides, programmiliides, API-liides.
FXML	XML-il põhinev Oracle korporatsiooni poolt JavaFX rakenduste defineerimiseks välja töötatud märgistuskeel.
HTTP	*HyperText Transfer Protocol*. Hüperteksti edastusprotokoll.
L3	Satelliidi kommunikatsiooni jaoks kasutatav suhtlusprotokolli kiht.
Material Design	Google'i poolt arendatud disainivool.
URI	*Uniform Resource Identifier*. Ühtne ressursi-indikaator.

Sisukord

Jooniste loetelu

1 Sissejuhatus

TTÜ Satelliidiprogrammi raames korraldatakse maajaamade projekti, mille eesmärgiks on tutvustada Eesti põhi- ning keskkooliõpilastele satelliiditehnoloogiaid. Kooliõpilastele korraldatakse töötubasid, kus ehitatakse antenn ning raadio, millega saavad õpilased püüda satelliitide poolt saadetavaid teateid. Teadete lugemiseks on vaja töölauarakendust, mis on seotud raadio ning antenniga.

Maajaamade projektis osalevad koolid peavad saama saata teistele projektis osalevatele koolidele TTÜ satelliidi vahendusel sõnumeid. Lisaks sellele korraldatakse projekti raames Kosmoselaulu püüdmise võistlust. TTÜ satelliit hakkab tükkide kaupa edastama laulu „Ta lendab mesipuu poole“ ning projektis osalevad koolid saavad võimaluse oma antenniga kinni püüda kõik edastatavad laulutükid. Projekti raames toimuvate tegevuste, osalevate koolide ning maajaamade projektiga seotud statistika haldamiseks on vaja veebirakendust.

Lahenduseks pakub autor välja veebirakenduse ning töölauarakenduse loomise. Veebirakendus võimaldab koolidel saata sõnumeid ning saadetud ja kätte saadud sõnumeid hallata. Veebirakendus aitaks koolidel hallata ka kinni püütud kosmoselaulu tükke. Veebirakendus võimaldaks maajaamade projekti korraldajatel hallata osalevaid koole ning koguda statistikat. Lisaks sellele on veebirakendusel TTÜ tudengisatelliiti ning maajaamade projekti tutvustav funktsionaalsus ning külastajatele kuvatakse statistikat projekti kohta. Töölauarakendus võimaldab koolidel näha omaehitatud antenniga püütud sõnumeid.

Käesoleva töö raames seab autor endale neli peamist eesmärki:

- luua andmebaasitäiendus TTÜ satelliidi missiooni juhtimise andmebaasile, milles hakatakse hoidma maajaamade projekti jaoks olulist infot;
- luua API, mis võimaldab hallata koolide, projekti raames toimuvate tegevuste ning projekti statistikaga seotud infot;

- luua eelnevas punktis mainitud API-t kasutav veebileht projektist huvitatud isikute, projektis osalevate koolide ning projekti korraldajate jaoks;
- luua töölauarakenduse kasutajaliides ning teises punktis mainitud API-ga suhtlev osa ning ühendada see eraldiseisva antenni ja raadioga suhtleva tarkvaraga.

2 Analüüs

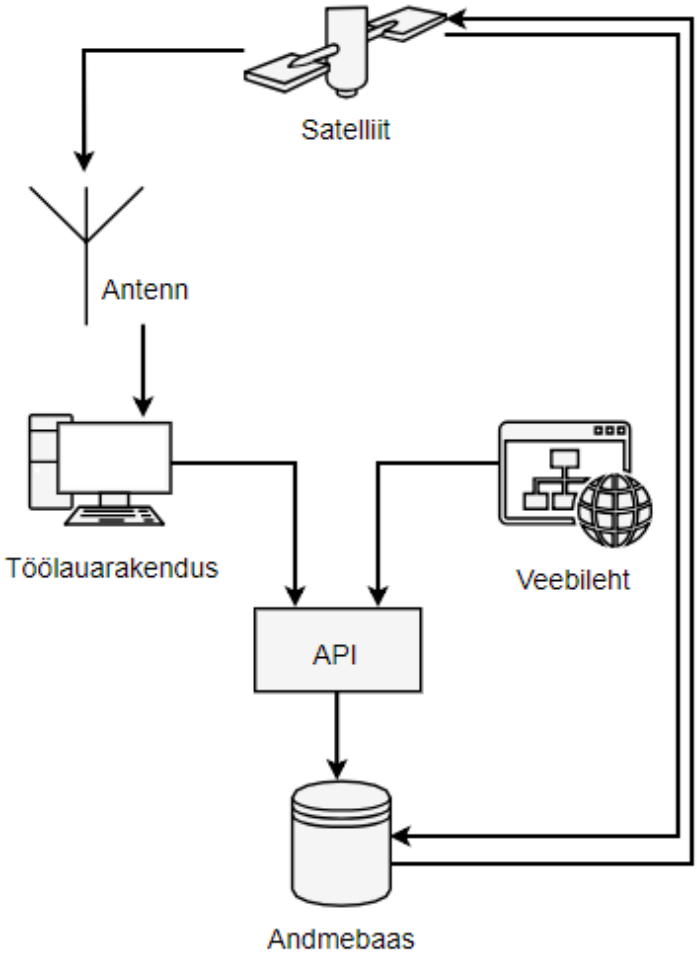
Käesolevas peatükis annab autor ülevaate TTÜ Satelliidiprogrammi maajaamade projektist ning kirjeldab projekti jaoks loodavate rakenduste funktsionaalseid ning mittefunktsionaalseid nõudeid.

2.1 Maajaamade projekt

TTÜ Satelliidiprogrammi maajaamade projekti eesmärgiks on tutvustada Eesti kooliõpilastele satelliiditehnoloogiaid ning anda õpilastele võimalus ise satelliidi poolt saadetavaid sõnumeid püüda. Projekti raames peavad koolid saama saata sõnumeid üle satelliidi teistele koolidele, lisaks sellele saavad koolid enda ehitatud väikeste antennide abil püüda teiste koolide poolt neile saadetud sõnumeid ning kosmoses olevate satelliitide edastatavaid signaale. Teiseks oluliseks funktsionaalsuseks on kosmoselaulu saatmine ning püüdmine: Eesti Vabariigi 100. sünnipäeva auks edastab satelliit kosmoselaulu "Ta lendab mesipuu poole". Laul on jagatud tükkideks ning seda saadetakse osade kaupa alla. Projektis osalevad koolid peavad saama kosmoselaulu tükke oma antennidega kinni püüda, eesmärgiks on kokku koguda kõik laulutükid, et saada kätte terve laul. Laulutükkide püüdmine on võistlus projektis osalevate koolide vahel.

Maajaamade projekti jaoks tuleb täiendada satelliidi missiooni juhtimise rakenduse jaoks loodud andmebaasi, luua uus API (*Application Programming Interface*) ning uus veebileht satelliidiprogrammi avaliku veebi jaoks, lisaks sellele tuleb luua ka töölauarakendus, mida koolid saavad kasutada koos endaehitatud antenniga sõnumite, kosmoselaulu tükkide ning satelliitide poolt edastatavate teadete püüdmiseks. Loodava süsteemi arhitektuuri kirjeldab Joonis 1.

Süsteemi arhitektuuri joonisel on näha andmebaas, millele tuleb maajaamade projekti raames lisada täiendus. Andmebaasiga suhtleb loodav API ning kaudselt ka satelliit. API käest pärivad andmeid loodav veebileht ning töölauarakendus. Töölauarakendus suhtleb ka õpilaste poolt ehitatud antenni ning raadioga, mis suudab püüda kinni satelliidi poolt saadetavat infot.



Joonis 1. Süsteemi arhitektuur.

2.2 Funktsionaalsed nõuded

Funktsionaalsed nõuded jagunevad käesoleva töö puhul neljaks – nõudeid vaadeldakse veebilehte kasutava administraatori, veebilehte kasutava projektiga liitunud kooli, töölauarakendust kasutava projektiga liitunud kooli ning veebilehega tutvuva isiku seisukohast.

Kõik isikud sh veebilehega tutvuvad isikud saavad teostada veebilehel järgnevaid toiminguid:

- avalehel oleva statistika ning informatsiooniga tutvumine;
- informatsiooni lehel oleva täiendava informatsiooniga tutvumine;

- statistika lehel oleva statistikaga tutvumine;
- sisselogimise lehel oleva sisselogimisvormi nägemine.

Veebilehte kasutav projektiga liitunud kool saab teha järgnevaid toiminguid:

- sisselogimislehel oma kooli kasutajaga veebilehele sisse logimine;
- veebilehelt välja logimine;
- sõnumite lehel sõnumi lisamine sh sõnumi teksti sisestamine ning teiste projektis olevate koolide hulgast adressaadi valimine;
- sõnumite lehel kätte saadud ning saadetud sõnumite nimekirja nägemine ning sõnumitega seotud informatsiooni nägemine;
- töölauarakenduse allalaadimine;
- kosmoselaulu lehel kinni püütud kosmoselaulu tükkide nägemine.

Veebilehte kasutav administraator saab teostada järgnevaid toiminguid:

- sisselogimise lehel oma kasutajaga veebilehele sisse logimine;
- veebilehelt välja logimine;
- registreerimise lehel uute koolide ning administraatorite registreerimine, sisestades kasutajanime, parooli ning valides kooli nime ja rolli;
- sõnumite lehel sõnumite lisamine ning kõigi saadetud sõnumite nimekirja ning info nägemine;
- koolide lehel osalevate koolide nimekirja nägemine;
- kosmoselaulu lehel saadetud laulutükkide info nägemine ning koolide püütud laulutükkide statistika nägemine.

Töölauarakendust kasutav projektiga liitunud kool saab teostada järgnevaid toiminguid:

- oma kooli kasutajaga töölauarakendusse sisse logimine;

- töölauarakendusest välja logimine;
- antenniga kinni püütud satelliitide poolt saadetud teadete, sh TTÜ satelliidi poolt edastatud kosmoselaulu tükkide ja sõnumite, nimekirja nägemine;
- kinni püütud sõnumite saatja, saatmisaja ning sisu nägemine;
- kinni püütud kosmoselaulu tükkide nägemine.

2.3 Mittefunktsionaalsed nõuded

Mittefunktsionaalne nõue, mis kehtib kõikidele realiseeritavatele komponentidele on inglisekeelne lähtekood.

Veebilehe mittefunktsionaalseteks nõueteks on rakenduse töötamine brauserite Google Chrome ning Mozilla Firefox uuemate versioonidega ning eestikeelne kasutajaliides.

Töölauarakenduse mittefunktsionaalseteks nõueteks on rakenduse töötamine Windows, Linux ja macOS operatsioonisüsteemidel ning eestikeelne kasutajaliides.

2.4 Rakenduste realiseerimiseks kasutatavad tehnoloogiad

Rakenduste realiseerimiseks kasutatavate tehnoloogiate valikut selgitatakse täpsemalt järgnevates peatükkides, antud jaotise eesmärk on täpsustada süsteemi arhitektuuri ning anda lühike ülevaade kasutatavatest tehnoloogiatest.

Maajaamade projekti jaoks realiseeritavate komponentide loetelu, koos neis kasutatavate tehnoloogiatega:

- Andmebaas[1] – PostgreSQL 9.5[2] ja Liquibase[3]

[1] https://gitlab.cs.ttu.ee/tut-satellite/mcs

[2] https://www.postgresql.org/docs/9.5/static/

[3] https://www.liquibase.org/

- API[1] – järgib REST põhimõtteid, Java 8[2], Spring Boot[3] ja Hibernate[4]
- Veebileht – Vue.js[5] raamistik, Vuetify.js[6] raamistik, EcmaScript 6[7] ja Webpack[8]
- Töölauarakendus[9] – Java 8, JavaFX[10]

Süsteemi arhitektuuri täpsustab Joonis 2.

[1] https://gitlab.cs.ttu.ee/heleot/public-web-project

[2] https://docs.oracle.com/javase/8/docs/

[3] https://projects.spring.io/spring-boot/

[4] http://hibernate.org/

[5] https://vuejs.org/

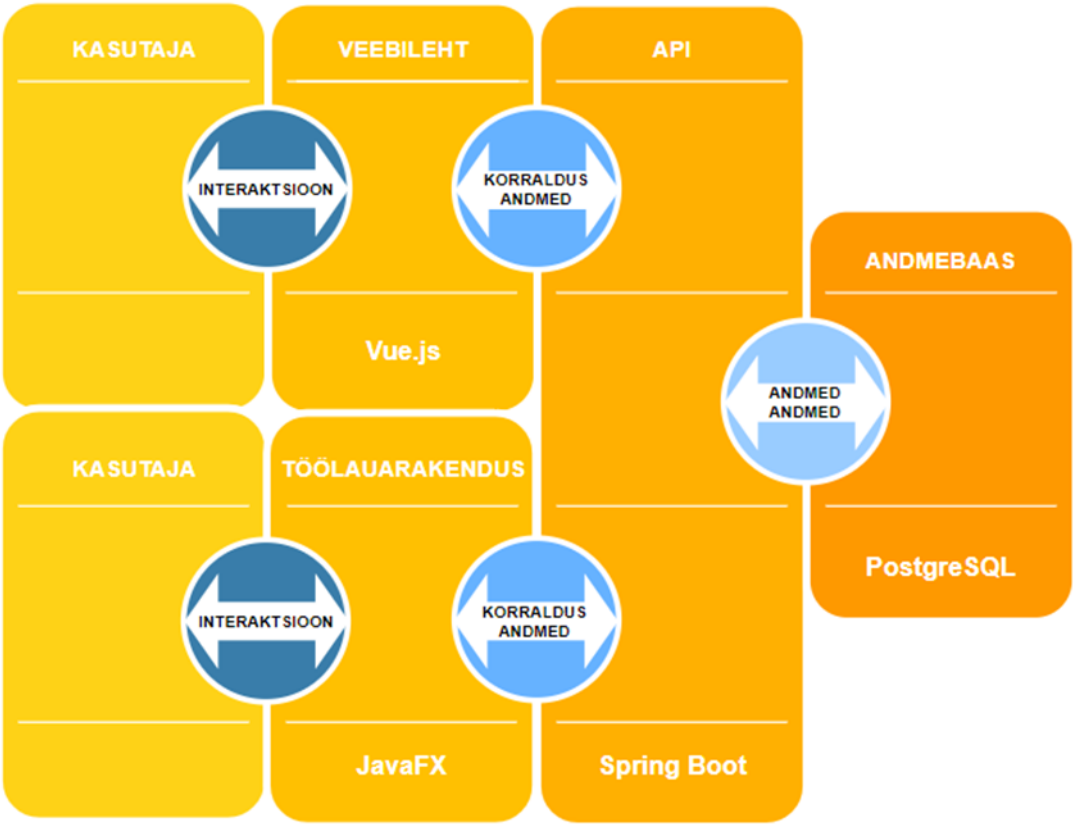
[6] https://vuetifyjs.com/en/

[7] http://es6-features.org

[8] https://webpack.js.org/

[9] https://gitlab.cs.ttu.ee/heleot/ground-station-desktop

[10] https://docs.oracle.com/javase/8/javase-clienttechnologies.htm



Joonis 2. Süsteemi arhitektuur koos kasutatavate tehnoloogiatega.

3 Andmebaasi täiendus

Käesolevas peatükis kirjeldatakse maajaamade projekti jaoks loodud andmebaasitäienduse planeerimist ning realiseerimist. Uus andmebaasiskeem lisati täienduseks TTÜ Satelliidiprogrammi missiooni juhtimise andmebaasile.

3.1 Andmebaasiskeem

Üheks juhendaja poolt andmebaasile seatud nõudmiseks oli olemasolevasse andmebaasi uue andmebaasiskeemi lisamine. Erinevalt traditsioonilisest ning aegunud praktikast, kus rakenduse erinevate osade jaoks luuakse eraldiseisvad andmebaasid, milles kõik objektid paigutatakse *public* andmebaasiskeemi, on mitme andmebaasiskeemiga andmebaasi loomisel mitmeid eeliseid.

Esiteks on võimalik ühe andmebaasiühenduse raames pääseda ligi erinevates andmebaasiskeemides olevatele objektidele ning seega ei pea rakendus looma mitut andmebaasiühendust, mis muudaks andmebaasidega ühenduvate rakenduste konfigureerimise keerukamaks. Teiseks ei pea ühele serverile paigaldama mitut andmebaasi, sest piisab ainult ühe andmebaasi paigaldamisest. Kolmandaks on objektid üksteisest eraldatud, sest asuvad erinevates nimeruumides. Lisaks eelnevalt mainitud eelistele pakub mitme andmebaasiskeemi kasutamine turvalisemat lahendust. Andmebaasiskeemid võimaldavad lihtsalt ja mugavalt piirata andmetele ligipääsemisega seotud õigusi ning seega suureneb andmebaasis hoitavate isikuandmete ning missiooni kontrolliks kasutatavate satelliidiga seotud andmete turvalisus. Näiteks on mitme andmebaasiskeemiga andmebaas kaitstum SQL injektsiooni ning andmete kogemata kustutamise või ülekirjutamise osas [3], [7].

Andmebaasile otsustas autor lisada uue skeemi nimega *groundstation* (maajaam), kuna uue skeemi peamiseks eesmärgiks on maajaamade projektiga seotud andmete hoiustamine.

3.2 Maajaamade projekti registrite detailanalüüs

Järgnevalt kirjeldab autor maajaamade projekti jaoks olulisi andmeid ning kontseptuaalset andmemudelit, mis koosneb olemisuhte diagrammidest ning nendel

olevatest olemitüüpidest ja atribuutidest. Maajaamade projekti jaoks on olulised kolme tüüpi andmed – kasutajate ehk koolide ja administraatoritega seotud andmed, sõnumite andmed ning kosmoselauluga seotud andmed. Seetõttu on andmebaasis neli registrit – koolide register, sõnumite register, laulutükkide register ning ka klassifikaatorite register. Neist viimane on klassifikaatoritabelitele, näiteks sõnumi olekute tabeli jaoks.

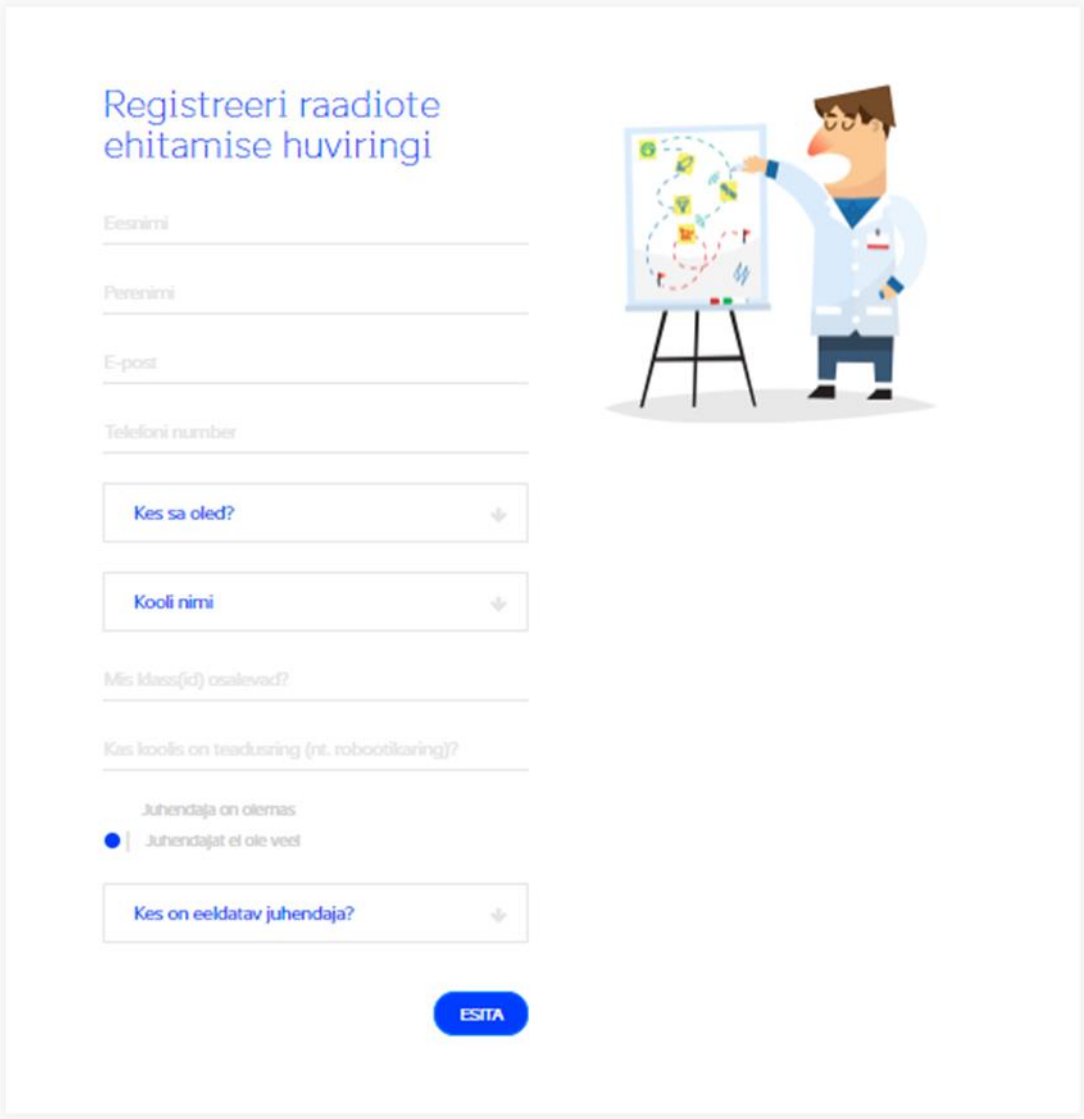
Olemisuhte diagrammidel on registrisse kuuluvad põhiobjektid tähistatud sinisega, registrisse kuuluvad mittepõhiobjektid on tähistatud kollasega ning rohelisega on tähistatud registrisse kuuluvad objektid, mida on vaja süsteemi toimimise tagamiseks (vt Joonis 4, Joonis 5, Joonis 6, Joonis 7).

3.2.1 Koolide register

Koolid registreerib süsteemi administraator. Selleks, et maajaamade projektiga liituda, peavad projektist huvitatud koolide esindajad täitma registreerimisvormi, mis asub „Kosmosevabariik 100“[1] veebilehel (vt Joonis 3). Sisestatud andmete põhjal luuakse igale registreerunud koolile kasutajanimi ning parool. Antud projekti jaoks on vaja andmebaasis hoida registreerunud koolide nimesid, neile määratud paroole ja kasutajanimesid ning ka koolide kontakt e-posti aadresse.

Kuna süsteemis on peale registreerunud koolide vaja luua ka kasutajad administraatoritele, kellel on samuti vaja kasutajanime ning parooli, tuleks segaduse vältimiseks ning süsteemi lihtsuse säilitamiseks hoida administraatorkasutajate andmeid samas tabelis koolide andmetega. Administraatorkasutajatel on samad andmed, mis registreerunud koolidel, kasutajanime ning parooli saavad nad endale ise määrata, kooli nimeks on Tallinna Tehnikaülikool ning kontakt e-postiaadressiks on konkreetse administraatori e-posti aadress.

[1] https://kosmos100.ee



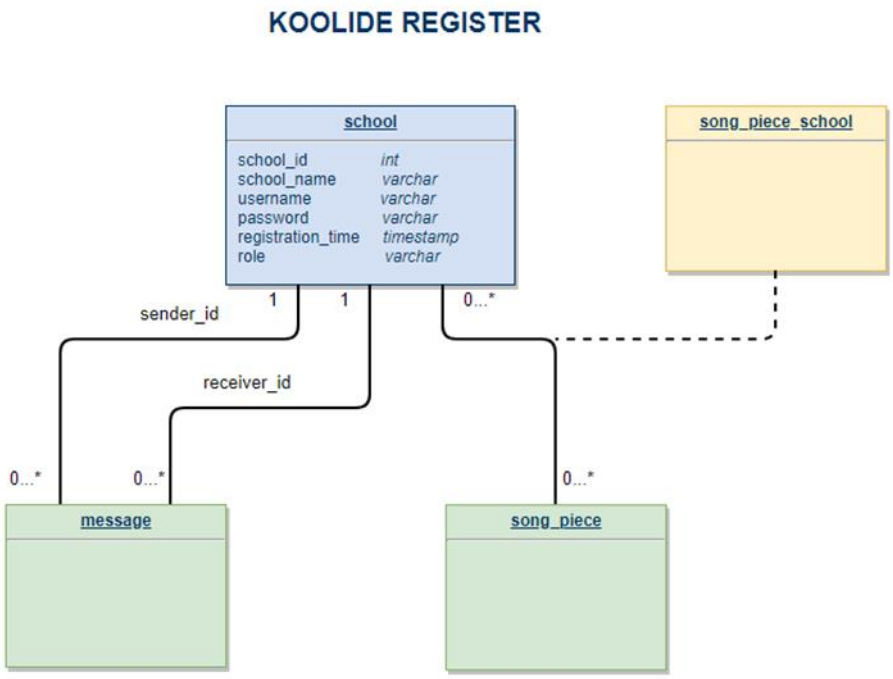
Joonis 3. Registreerimisvorm „Kosmosevabariik 100“ veebilehel.

Selleks, et eristada koole administraatoritest tuleb igale kasutajale määrata roll. Rollideks on administraator ning kool. Rollide alusel on võimalik piirata kasutajate ligipääsu erinevatele ressurssidele. Rollid ei saa kasutaja eluea jooksul muutuda, kuna koolide kasutajad on loodud terve kooli jaoks ning administraatorkasutajad on loodud ühe konkreetse administraatori jaoks. Seetõttu otsustas autor rollist klassifikaatorit mitte teha.

Koolide registri olemisuhte diagrammil (vt Joonis 4) on näha tabel *school* (kool), milles on väljad eelnevalt kirjeldatud andmete jaoks – *username* (kasutajanimi), *school_name* (kooli nimi), *password* (parool) ning *role* (roll). Lisaks sellele on tabelis ka andmebaasisüsteemi poolt koolile genereeritav identifikaator *school_id*, mida saab kasutada kasutajate tuvastamiseks, andmetüübiks valisin *int*, kuna süsteemi kasutajaid ei

saa olla nii palju, et tekiks vajadus kasutada *bigint*'i. Eestis on praegusel hetkel 293 põhikooli ning 164 gümnaasiumit, seega oleks maksimaalne kasutajate hulk süsteemis koos administraatoritega 500 ringis [1]. Tabelis *school* on ka väli kasutaja registreerimisaja jaoks.

Tabeliga *school* on seotud ka tabelid *messages* (sõnumid) ja *song_piece* (laulutükk), kuna koolid saavad sõnumeid saata ja püüda ning kosmoselaulu tükke püüda. Iga sõnumiga on seotud kaks kooli - saatja ning adressaat. Koolide identifikaatorid on sõnumite tabeliga seotud välisvõtmete abil. Igale kosmoselaulu tükile saab vastata mitu kooli, mis on selle laulutüki kinni püüdnud, ja iga kool võib kinni püüda mitu laulutükki, seega tuleb teha andmebaasi vahetabel, sest tegemist on mitu-mitu suhtega.



Joonis 4. Koolide registri olemisuhte diagramm.

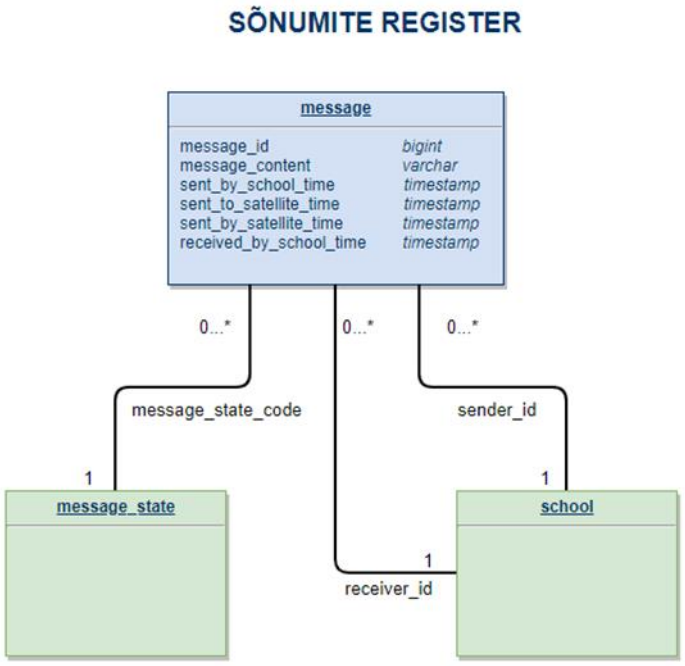
3.2.2 Sõnumite register

Nagu eelnevalt mainitud tuleb iga sõnumi kohta teada saatja kooli ning adressaatkooli. Lisaks sellele on sõnumi juures oluline sisu, mis ei tohi olla liiga pikk. Sõnumi pikkusele seab piirangu L3 *frame*'i mahtuvus. L3 on satelliidi kommunikatsiooni jaoks kasutatav suhtlusprotokolli kiht. Sõnum tuleb mahutada ühte L3 *frame*'i, kuna satelliidile saab saata andmeid piiratud koguses ning lühikeste ajaperioodide vältel. Koolidevahelised sõnumid on madalama prioriteediga kui satelliidile saadetav uus tarkvara ning juhtimiskäsklused.

Ühte L3 *frame*'i mahub kuni 1960 bitti ehk 245 baiti andmeid, seega tuleb sõnumi pikkusele seada piirang [4]. Sõnumi maksimaalseks pikkuseks valis autor 140 tähemärki, põhinedes populaarse sotsiaalmeediarakenduse Twitter esialgsel sõnumipikkusel, mis tõestab, et ka 140 tähemärgi pikkuste sõnumitega saab kõik olulise edastatud [6]. Lisaks sellele on sõnumi pikkus harjumuspärane inimestele, kes on Twitterit kasutanud. Eeldusel, et kasutatava tähemärgi suuruseks on 8 bitti on ühe sõnumi maksimaalseks pikkuseks 140 baiti, mis jätab pisut lisaruumi juhuks, kui koolid saadavad sõnumeid, mis sisaldavad sümboleid, mis on suuremad kui 8 bitti.

Sõnumite puhul tuleb säilitada ka infot sõnumi oleku kohta. Sõnumil saab olla neli olekut – kool on saatnud sõnumi, sõnum on saadetud satelliidile, satelliit on sõnumi edastanud ning adressaatkool on sõnumi kätte saanud, kas sõnumi ise antenniga kinni püüdes või siis hiljem veebilehel sõnumit nähes. Sõnumi oleku järgi saab kindlaks teha, millised sõnumid on veel satelliidile saatmata. Kõigi olekumuutuste kohta on vaja ka säilitada kellaaega, et koolidel oleks huvitavam sõnumi teekonda jälgida. Lisaks sellele on satelliidile sõnumi saatmise kellaaja järgi võimalik ennustada, millal satelliit sõnumi edastab.

Sõnumi olek on klassifikaator, igale olekule saab vastata mitu sõnumit vt (Joonis 5). Info sõnumi erinevate saatmisaegade kohta salvestatakse tabelisse *message*, sest tegemist on konkreetset sõnumit iseloomustavate aegadega. Igal sõnumil on ka andmebaasisüsteemi poolt genereeritud unikaalne identifikaator *message_id*. Sõnumiga on välisvõtme abil seotud saatjakooli ning adressaatkooli unikaalsed identifikaatorid.

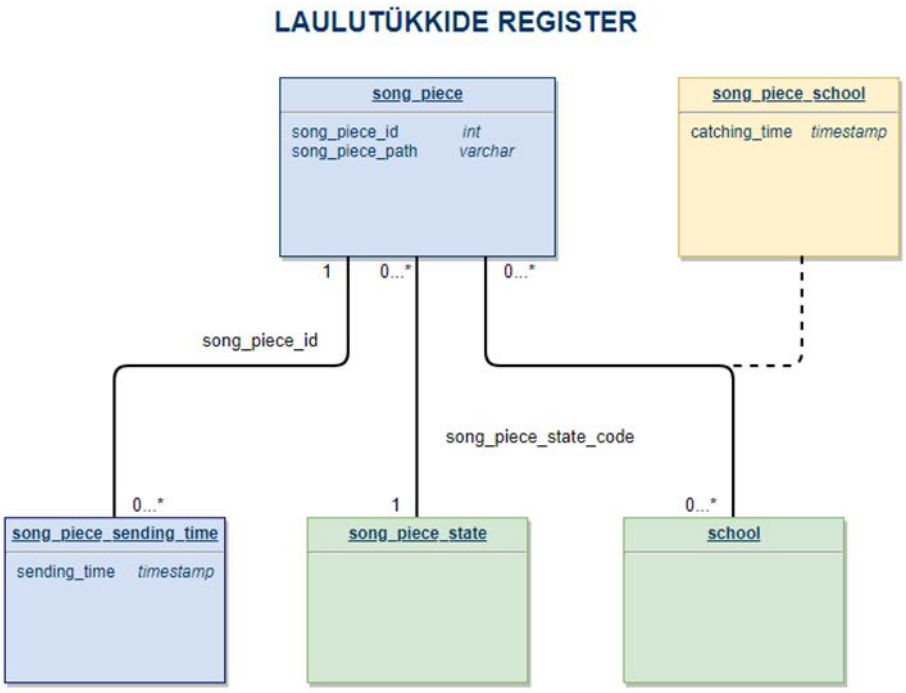


Joonis 5. Sõnumite registri olemisuhte diagramm.

3.2.3 Laulutükkide register

Iga kosmoselaulu tüki kohta on vaja teada tema järjekorranumbrit, mis on ka laulutüki identifikaatoriks. Laulutüki identifikaatori võib genereerida andmebaasisüsteem, kui laulutükkide info andmebaasi korrektses järjekorras sisestada. Laulutükkide arv ning sisu lepitakse enne projekti algust kokku, seega saab kõik tükid korraga andmebaasi sisestada. Laulutükkide arv ei tohi olla liiga suur, kuna eesmärgiks on siiski see, et koolid, kes kosmoselaulu püüdmise võistlusest osa võtavad, suudavad terve laulu kokku saada. Praeguseks hetkeks pole täpne laulutükkide arv veel selgunud.

Autor otsustas, et laulutüki sisuks oleva MP3 faili asemel hakatakse andmebaasis hoidma viidet laulutüki asukohale failisüsteemis. Kuna kosmoselaulu tükkide püüdmine on koolidevaheline võistlus, tuleb säilitada ka infot selle kohta, millised koolid on konkreetse laulutüki kinni püüdnud ning millal nad laulutüki kinni püüdsid. Selleks tuleb luua vahetabel *song_piece_school*, kus salvestatakse seos kooli ning laulutüki vahel, lisaks sellele salvestatakse tabelisse ka laulutüki kinnipüüdmise aeg (vt Joonis 6).



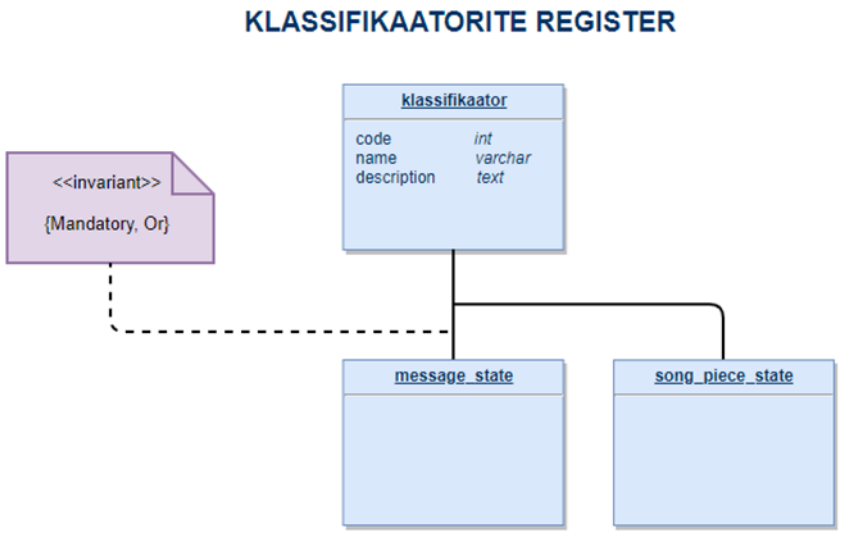
Joonis 6. Laulutükkide registri olemisuhte diagramm.

Maajaamade projektiga liituvad koolid järk-järgult, seega tuleb üht laulutükki satelliidil saata mitu korda, mistõttu on vaja teada, millal satelliit on konkreetset laulutükki edastanud. Laulutüki saatmisaegade salvestamiseks tuleb teha tabel *song_piece_sending_time*, milles on laulutüki unikaalne identifikaator seotud laulutüki saatmisajaga.

Oluline on ka laulutüki olek, vaja on informatsiooni selle kohta, kas laulutükki pole kunagi saadetud või on teda vähemalt kord juba satelliidi poolt edastatud. Selleks tuleb luua klassifikaatoritabel *song_piece_state*, kus on kaks olekut – laulutükk on saadetud ning laulutükki ei ole saadetud.

3.2.4 Klassifikaatorite register

Klassifikaatorite registris on kaks tabelit, sõnumite olekute tabel *message_state* ning laulutükkide olekute tabel *song_piece_state* (vt Joonis 7). Igal olekul on oma kood, nimetus ning kirjeldus. Nagu juba eelnevalt mainitud saab sõnum olla kas kooli poolt saadetud, satelliidile saadetud, satelliidi poolt edastatud või adressaatkooli poolt kätte saadud. Kosmoselaulu tükk võib olla kas edastatud või mitteedastatud.



Joonis 7. Klassifikaatorite registri olemisuhte diagramm.

3.3 Andmebaasi täienduse realiseerimine

Käesolevas alapeatükis annab autor ülevaate andmebaasi realiseerimisest.

3.3.1 Olemasolev andmebaas

Andmebaasitäienduse realiseerimine algas olemasoleva andmebaasiga tutvumisest. Satelliidiprogrammi missiooni juhtimise tarkvara jaoks loodud andmebaas on loodud PostgreSQL 9.5 andmebaasisüsteemi abil. Andmebaasi migratsioonide haldamiseks on projektis kasutusele võetud Liquibase teek. Sellest tulenevalt jätkas autor antud tehnoloogiate kasutamisega ning lisas olemasolevasse andmebaasi uue andmebaasiskeemi.

3.3.2 Tabelite lisamine andmebaasiskeemi

Tabeleid andmebaasi lisades alustas autor teistest tabelitest mitte sõltuvatest tabelitest, ehk klassifikaatoritabelitest ning koolide tabelist. Seejärel lisas autor sõnumite ning laulutükkide tabelid ning lõpus vahetabelid ning lisainfo tabelid. Viimase sammuna lisas autor andmebaasi ka esialgsed klassifikaatorite väärtused.

Tabeleid luues pidas autor oluliseks kasutada ära PostgreSQL-i poolt pakutavaid võimalusi, et vältida üleliigset tööd API loomisel. Seega on kõikides tabelites olemas primaarvõtmed. Vahetabelitesse lisas autor kahest väljast koosnevad primaarvõtmed (vt Joonis 8). Lisaks sellele pidas autor tabelite loomisel oluliseks sobivate piirangute seadmist veergudele. Juhul kui väärtus tohib esineda ainult ühe korra, on veerud unikaalse määratlusega või primaarvõtmed. Autor lisas kohustuslikele väljadele juurde ka *NOT NULL* märgise. Võimaluse korral kasutas autor ka vaikimisi väärtusi, näiteks oli sellest kasu uue sõnumi loomisel, kui sai sõnumile automaatselt määrata sobiva klassifikaatori väärtuse. Vaikimisi väärtused tulid kasuks ka ajaliste väärtuste korral, kus sai uue rea lisamisel kaasa anda ka näiteks uue koolikasutaja loomise aja (vt Joonis 9), sõnumi kooli poolt saatmise aja või laulutüki kinni püüdmise aja (vt Joonis 8). Vaikimisi väärtust sai kasutada ka rolli korral koolide tabelis, sest suurem osa lisatavatest kasutajatest on maajaamade projektiga liitunud koolid (vt Joonis 9).

```
CREATE TABLE groundstation.song_piece_school
(
school_id     INT
references groundstation.school(school_id) NOT NULL,
song_piece_id INT
references groundstation.song_piece(song_piece_id) NOT NULL,
catching_time TIMESTAMP
WITHOUT TIME ZONE NOT NULL DEFAULT NOW(),
PRIMARY KEY (school_id, song_piece_id)
);
```

Joonis 8. Koolide ja laulutükkide vaheliste seoste tabel.

```
CREATE TABLE groundstation.school
(
school_id         SERIAL NOT NULL PRIMARY KEY,
username          VARCHAR(50) NOT NULL UNIQUE,
school_name       VARCHAR(50) NOT NULL,
password          VARCHAR(60) NOT NULL,
registration_time TIMESTAMP
WITHOUT TIME ZONE NOT NULL DEFAULT NOW(),
role              VARCHAR(50) NOT NULL DEFAULT 'SCHOOL'
);
```

Joonis 9. Koolide tabel.

4 API

Antud peatükis kirjeldab autor maajaamade projekti ning TTÜ Satelliidiprogrammi avaliku veebi jaoks loodud API planeerimist ning realiseerimist.

4.1 API planeerimine

Tööd alustades oli autoril olemasoleva informatsiooni põhjal arusaam, et maajaamade projekti jaoks on vaja täiendada olemasolevat satelliidiprogrammi missiooni juhtimise tarkvara API-t. Sellest tulenevalt alustas autor API täienduse planeerimist missiooni juhtimistarkvara projekti uurimisest. Missiooni juhtimise API on loodud kasutades Spring Booti, Mavenit[1], Hibernate'i ning JHipsterit[2]. Autor tutvus projekti ülesehitusega ning lõi eraldiseisvad Java paketid uute kontrollerite ning teenusklasside jaoks. Lisaks sellele lõi autor ka esialgsed kontrollerklassid koolide, laulutükkide, statistika ning sõnumitega seotud teenuste jaoks.

Töö käigus lisainformatsiooni otsides ning juhendajale küsimusi esitades selgus aga, et tegelikult peavad maajaamade projekti API ning veebilehe arendused olema mitte koos satelliidiprogrammi siseveebi jaoks loodud missiooni juhtimise projektiga, vaid hoopis koos avaliku veebi projektiga. Avaliku veebi projekti polnud aga keegi veel loonud, seega sai autor ülesandeks luua uus projekt avaliku veebi jaoks, mille komponentideks pidi olema nii API kui ka klientrakendus ehk veebileht. Sellest tulenevalt suurenes ka planeeritud töö maht ning autor pidi maajaamade projekti jaoks vajalike komponentide loomise ulatuses tegema kärpeid. Autor võttis eesmärgiks töö käigus valmis saada võimalikult suure osa esialgselt planeeritud funktsionaalsustest, kuid keskendus ennekõike andmebaasi ning API loomisele, seejärel veebilehe loomisele ning lõpuks töölauarakenduse arendamisele.

API arendamiseks kasutatavaid tehnoloogiaid valides seadis autor eesmärgiks kasutada nii palju raamistikke ja teeke kui vaja ning nii vähe kui võimalik, et säilitada koodi

[1] https://maven.apache.org/

[2] https://www.jhipster.tech/

loetavus ning vältida projekti ülekoormamist vähekasutatud sõltuvustega. Seetõttu otsustas autor kasutada Spring Booti, Mavenit ning Hibernate'i.

Teenuste planeerimisel kasutas autor nimekirja funktsionaalsetest nõuetest ning mõtles läbi, millised teenused tuleb API-is luua, et veebilehel ning töölauarakenduses saaks ettenähtud tegevusi teha. Autor otsustas loodavad teenused jagada nelja kategooriasse – koolide, sõnumite, laulutükkide ning statistikaga seotud teenused. Eraldiseisvateks teenusteks on sisselogimise ning väljalogimise teenused.

4.2 API realiseerimine

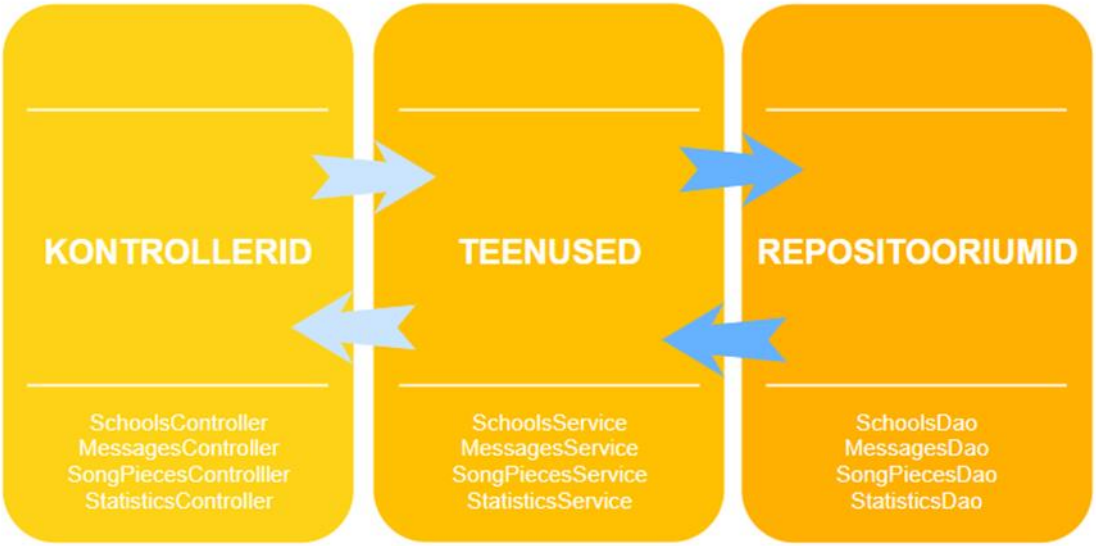
API realiseerimine algas uue Spring Boot projekti loomisega. Kõigepealt seadis autor paika projekti struktuuri. Loogiliselt on teenuste realiseerimine jagatud kolme kihti – kontrollerid, teenused ning repositooriumid. API kihte selgitab Joonis 10.

Esimene kiht on kontrollerite kiht, milles asuvad kontrollerklassid kõigi teenusekategooriate jaoks (vt Joonis 10). Kontrollerklasside peamiseks eesmärgiks on REST teenuste API otspunktide implementeerimine. Autor järgis teenuste loomisel REST API-de nimetamise reegleid. Teenused on nimetatud järjepidevalt ning loetavuse parandamiseks kasutas autor sidekriipse ning väiketähti. Lisaks sellele kasutas autor kaldkriipse, et väljendada ressursside vahelisi hierarhilisi suhteid [5]. Kõik teenused tagastavad HTTP (*HyperText Transfer Protocol*) staatuskoodi. Juhul kui päring õnnestus, tagastatakse staatuskood 200, kui päring tagastab tühja järjendi või objekti, tagastab teenus staatuskoodi 204. Autentimise ebaõnnestumise korral tagastavad kõik autentimist nõudvad teenused staatuskoodi 401.

Teiseks kihiks on teenuste kiht, kus asuvad teenuseklassid (vt Joonis 10). Teenuste kihi peamiseks ülesandeks on äriloogika implementeerimine, näiteks moodustatakse teenuskihis statistika päringu vastuseks olev objekt ning kontrollitakse API-le tehtud POST päringuga saadetud väärtuste valiidsust ning olemasolu.

Kolmandaks ning viimaseks kihiks on repositooriumite kiht. Andmebaasiga suhtleb loodud API Hibernate'i abil. Kõigi API jaoks oluliste tabelite kohta lõi autor *entity* (olemi) klassid, mille abil Hibernate saab küsida tabelitest vajalikku informatsiooni. Olemiklassid lõi autor nt koolide, sõnumite ning laulutükkide tabelite jaoks. Suhtlus

andmebaasiga toimub repositooriumiklassides (vt Joonis 10), nendes klassides olevaid meetodeid kutsuvad välja teenusekihi klassid.



Joonis 10. API kihtide struktuur.

Kõige keerukamaks osaks uue API loomisel osutus autori meelest Spring Security[1] konfigureerimine. Autor otsustas luua sisselogimise ning väljalogimise kasutades Spring Securityt. Lisaks sellele lõi autor raamistiku abil ka rollipõhise autoriseerimise. Autor määras ainult sisselogitud kasutajatele kätte saadavad API otspunktid autentimiskontrolliga otspunktide hulka. Konfiguratsiooni loomisel osutus keerukaks see, et paljud dokumentatsioonis ning juhendites väljapakutud lahendused ei toiminud tegelikkuses nii, nagu oli ette nähtud, seega nõudis konfiguratsiooni paikaseadmine rohket koodi silumist, katsetamist ning internetis olevate materjalidega tutvumist. Lisaks sellele pidi autor arendama paljudele Spring Security liidestele oma lahendused. Autentimise lõi autor kasutades sessiooni identifikaatorit. Enne uue kooli andmebaasi lisamist arvutatakse kasutaja paroolile bcrypt algoritmi kasutades räsi. Spring Security abil loodud sisselogimise teenus loob kasutajale uue sessiooni, kui päringuga kaasa antud parooli räsi on sama, mis andmebaasis olev räsi. Edaspidi kontrollib Spring Security päringutega kaasa antavat sessiooni ID-d.

[1] https://projects.spring.io/spring-security/

Realiseeritud teenused on koos HTTP meetodite, URI-de (*Uniform Resource Identifier*) ning seletustega esitatud töö lisades kontrollerklasside kaupa – koolidega seotud teenused (vt Lisa 1), sõnumitega seotud teenused (vt Lisa 2), laulutükkidega seotud teenused (vt Lisa 3) ning statistikaga seotud teenused (vt Lisa 4).

Kõikide ühiktestidega testitavate meetodite kohta kirjutas autor ka testid kasutades JUnit 4[1] testimisraamistikku. Peamised ühiktestidega testitavad meetodid asusid koos äriloogikaga teenuste kihis, lisaks sellele kirjutas autor teste ka sobivate HTTP staatuskoodide valimiseks kirjutatud meetoditele ning API jaoks loodud utiliitklassides olevatele meetoditele. Ühiktestide kirjutamisel pööras autor tähelepanu sellele, et testitavate meetodite funktsionaalsused oleksid täielikult testidega kaetud. Seega testis autor kõiki meetodeid nii korrektsete kui ka väärate sisenditega. Integratsiooniteste otsustas autor antud töö raames mitte kirjutada, sest API teenused võivad veel muutuda, kuna maajaamade projektiga seotud funktsionaalsused võivad tulevikus muutuda. Seetõttu katsetas autor käsitsi läbi kõik API otspunktid ning testis lisaks korrektsetele sisendparameetritele ka väärasid sisendeid.

[1] https://junit.org/junit4/

5 Veebileht

Käesolevas peatükis käsitleb autor projekti veebilehe planeerimist ning realiseerimist.

5.1 Veebilehe planeerimine

Sarnaselt API planeerimisele oli autori esialgseks arusaamaks, et seoses maajaamade projektiga täiendatakse satelliidi missiooni kontrolli süsteemi jaoks loodud veebilehte.

Esialgse plaani kohaselt oleks missiooni juhtimise veebilehele saanud sisse logida lisaks satelliidi missiooni kontrolli administraatoritele ka koolid oma kasutajatega. Neile oleks kuvatud maajaamade projektiga seotud vaated, mille nägemiseks neil oleksid olnud õigused. Sisse logimata veebilehe külastajatele oleksid nähtavad olnud satelliidi ning maajaamade projektiga seotud informatsiooni ning statistikaga vaated. Pärast sisselogimist oleksid koolide kasutajad näinud sõnumite vaadet, kus koolid oleksid saanud uusi sõnumeid lisada, kätte saadud ja saadetud sõnumite nimekirja vaadata, ning kosmoselaulu vaadet, kus koolid oleksid saanud kinni püütud kosmoselaulu tükke kuulata.

Missiooni kontrolli veebileht on loodud kasutades JHipsterit ning Angular[1] raamistikku, lisaks sellele on projektis kasutusel ka Bootstrapi[2] teek ning Sass[3]. Seega alustas autor veebilehe täienduse realiseerimist olemasoleva projekti ülesehitust ning seal kasutusel olevaid raamistikke uurides. Lisaks sellele realiseeris autor ka näidislehti, et mõista paremini veebilehe toimimist.

Pärast seda, kui autor oli juba alustanud olemasoleva veebilehe täiendamisega, selgus, et tegelikult tuleks lisada maajaamade projektiga seotud funktsionaalsused mitte missiooni kontrolli veebilehele, vaid hoopis TTÜ Satelliidiprogrammi avaliku veebi veebilehele, mis peab olema missiooni kontrolli veebilehest sõltumatu. Avaliku veebi jaoks polnud

[1] https://angular.io/

[2] https://getbootstrap.com/

[3] https://sass-lang.com/

aga projekti loodud, seega sai autori ülesandeks luua uus projekt ning valida selle jaoks sobivad raamistikud.

Autor otsustas luua uue veebliehe Vue.js raamistikku kasutades. Alguses olid kaalukausil nii Angular kui ka Vue.js raamistik, kuna autoril oli eelnev kogemus mõlema raamistiku kasutamisega. Vue.js osutus aga autori meeles paremaks valikuks. Tegemist on lihtsakoelise raamistikuga ning kuna autor soovis vältida projekti kohmakust, mida oli märganud missiooni juhtimise projekti uurides, tundus Vue.js parem valik kui Angular. Lisaks sellele on Vue.js sobilik väiksematele ning keskmise suurusega projektidele, mida arendab üks inimene või väike tiim, seevastu on Angular ideaalne suurte projektide jaoks. Autor uskus, et projekti käigus tuleb õppida kasutama ka raamistike keerukamaid funktsionaalsusi, seega on plussiks ka Vue.js raamistiku kerge õpitavus [2], [8] .

Vue.js raamistikuga koos otsustas autor kasutada ka Vuetify.js raamistikku, mis võimaldab kasutada *Material Design*'i komponente koos Vue.js'iga. *Material Design* on Google'i poolt arendatud disainivool. Vuetify.js raamistik võimaldab luua professionaalse väljanägemisega kasutajaliideseid, kuid ei piira rakenduse disainielementide isikupärastamist CSS-i abil.

Projektis olevate sõltuvuste ning ressursside haldamiseks otsustas autor kasutada Webpacki.

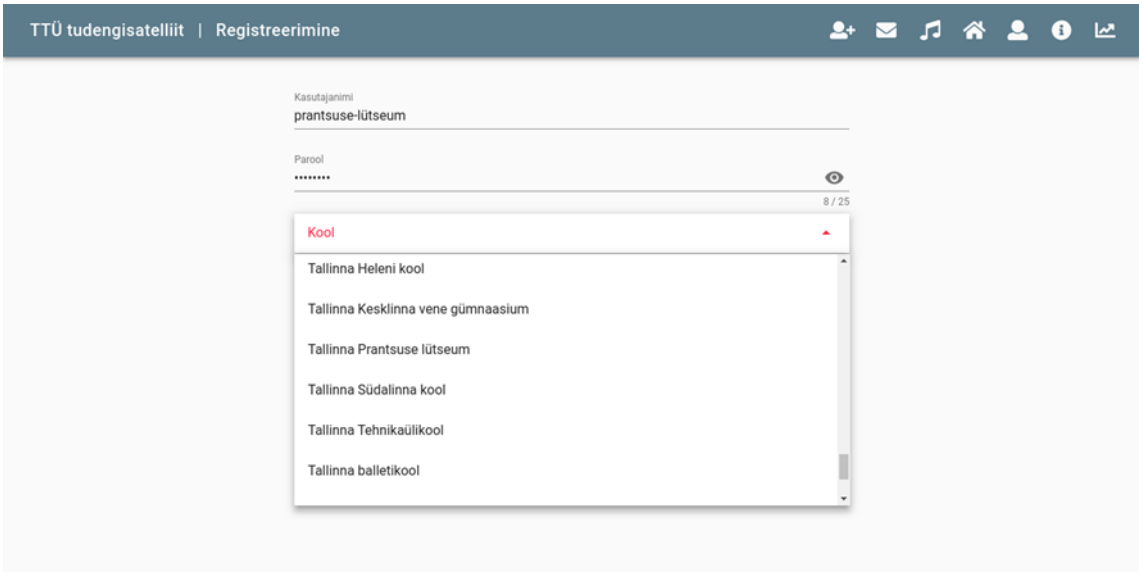
Veebilehe kasutajaliides põhineb esimeses peatükis loetletud funktsionaalsetel nõuetel. Sisselogimata kasutajad saavad navigeerida nelja vaate vahel, milleks on avaleht ning informatsiooni, sisselogimise ja statistika vaated. Sisselogitud kool saab lisaks eelmainitud vaadetele näha ka sõnumite ning kosmoselaulu vaateid. Administraatorkasutajad näevad veebilehel ka vaadet uute kasutajate registreerimiseks. Kasutajad saavad erinevate vaadete vahel navigeerida, kasutades menüüd. Rakendust planeerides kaalus autor nn burgermenüü kasutamist, kuid autori meelest polnud menüüelementide arv piisav, et seda õigustada. Seetõttu otsustas autor lehe ülaosas asuva navigatsiooniriba kasuks, millel kuvatakse erinevaid menüüelemente ikoonidena.

5.2 Veebilehe realiseerimine

Autor alustas veebilehe realiseerimist uue projekti loomisega. Kõigepealt lõi autor vaated kõigi vajalike lehekülgede jaoks – avalehe, sisselogimise, informatsiooni, statistika,

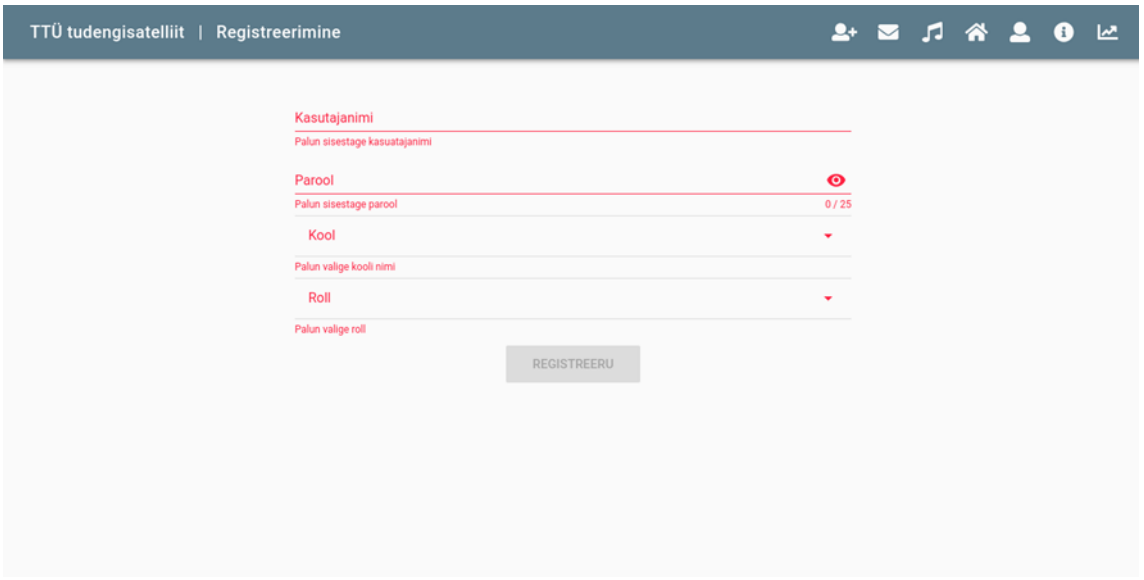
registreerumise, sõnumite ning kosmoselaulu lehed. Seejärel seadis autor paika lehtede ühtse väljanägemise, lisades iga lehe ülaossa navigatsiooniriba koos menüüelementidega. Järgmiseks oluliseks sammuks oli lehtede vahelise navigatsiooni loomine. Lehtede vahel saab navigeerida, kasutades lehe ülaosas asuval navigatsiooniribal asuvaid ikoone. Pärast kasutajaliidese üldiste elementide paika seadmist asus autor lehtedele sisu looma.

Registreerimise lehele lisas autor väljad kasutajanime ning parooli lisamiseks (vt Joonis 11, Joonis 12). Parool peab olema vähemalt kaheksa tähemärgi pikkune ning seda sisestades on võimalik näha, mitu tähte on juba olemas. Lisaks sellele on võimalik parooli nii peita kui ka tavatekstina näha. Seejärel saab kasutaja valida rippmenüüst sobiva kooli. Nimekirjas on kõik Eesti põhi- ning keskkoolid, lisaks sellele on nimekirjas ka Tallinna Tehnikaülikool administraatorkasutajate jaoks. Pärast kooli valimist saab administraator valida teisest rippmenüüst uuele kasutajale sobiva rolli. Juhul kui vorm pole korralikult täidetud, näiteks kui väljad on tühjaks jäetud, parool on liiga lühike või pole valitud kooli või rolli, kuvab leht veateateid ning ei luba vormi esitada (vt Joonis 12). Kui kõik vajalikud andmed on sisestatud, saab vajutada nuppu „Registreeru“. Seejärel genereeritakse paroolile räsiväärtus kasutades SHA-256 algoritmi ning saadetakse axiost[1] kasutades POST päring API-le. Päringu õnnestumise ning ebaõnnestumise kohta antakse kasutajale tagasisidet ka vastavate teadetega veebilehel.



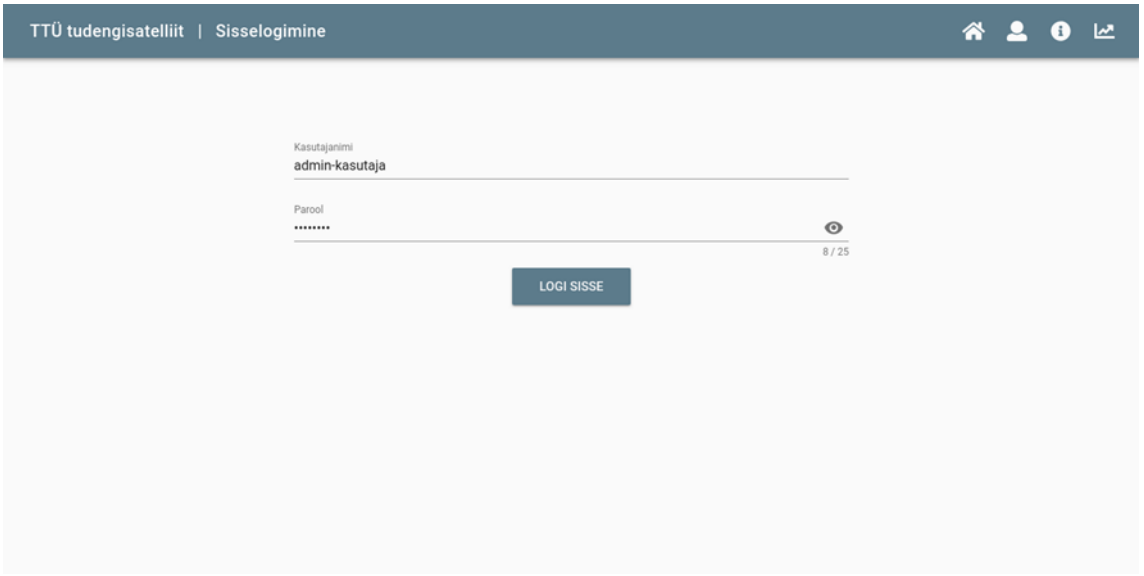
Joonis 11. Registreerumise lehekülg.

[1] https://github.com/axios/axios



Joonis 12. Registreerumise lehekülg veateadetega.

Sisselogimise leht (vt Joonis 13) sarnaneb registreerimise vaatega. Lehel on kaks välja, neist üks kasutajanimele ning teine paroolile. Kui kasutaja sisestab väärtused õigesti, muutub sisselogimise nupp aktiivseks, seda vajutades tehakse päring API pihta. Päringu õnnestumisel suunatakse kasutaja sõnumite leheküljele, ebaõnnestumise korral kuvatakse kasutajale veateadet. Valede sisendite korral kuvatakse taaskord veateateid.



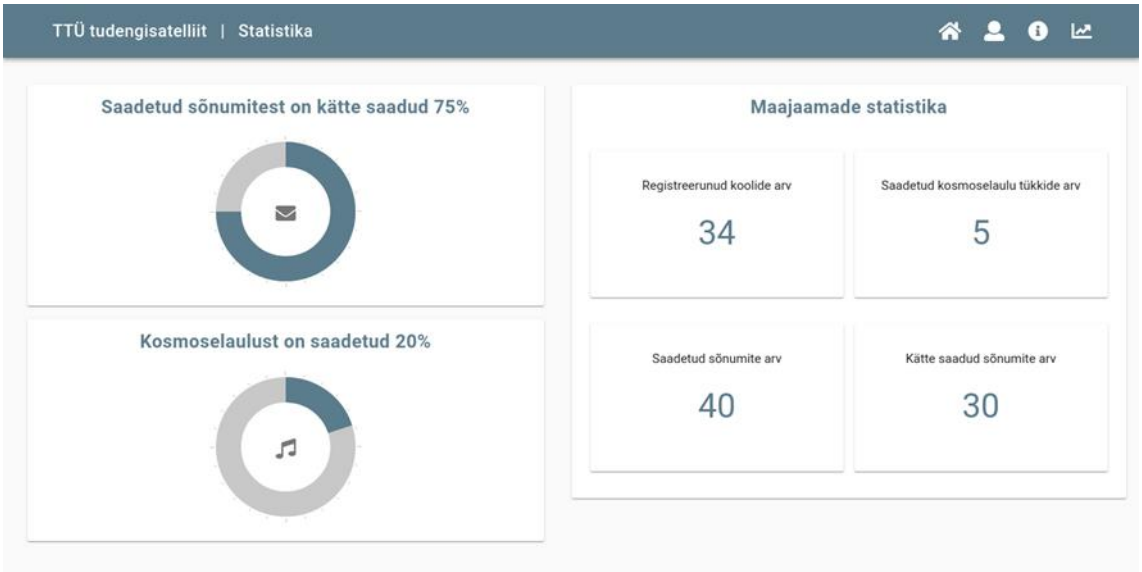
Joonis 13. Sisselogimise lehekülg.

Informatsiooni vaate (vt Joonis 14) jaoks lõi autor kaks erinevat sektsiooni. Informatsiooni vaate vasakul küljel on informatiivne tekst TTÜ Satelliidiprogrammi kohta, millele autor kujundas Vuetify.js komponente ning CSS-i kasutades küljenduse. Lehe paremas servas asuvad infokastid. Infokaste võib tulevikus lehele panna näiteks lähiajal toimuvate ürituste reklaamimiseks või olulisele informatsioonile viitamiseks. Iga infokast koosneb pildist, pealkirjast, lühikesest kirjeldavast tekstist ning nupust, millele vajutades juhatatakse kasutaja informatsiooniga seotud veebilehele, mis võib olla nii rakenduse sisene leht kui ka viide mõnele teisele veebilehele.

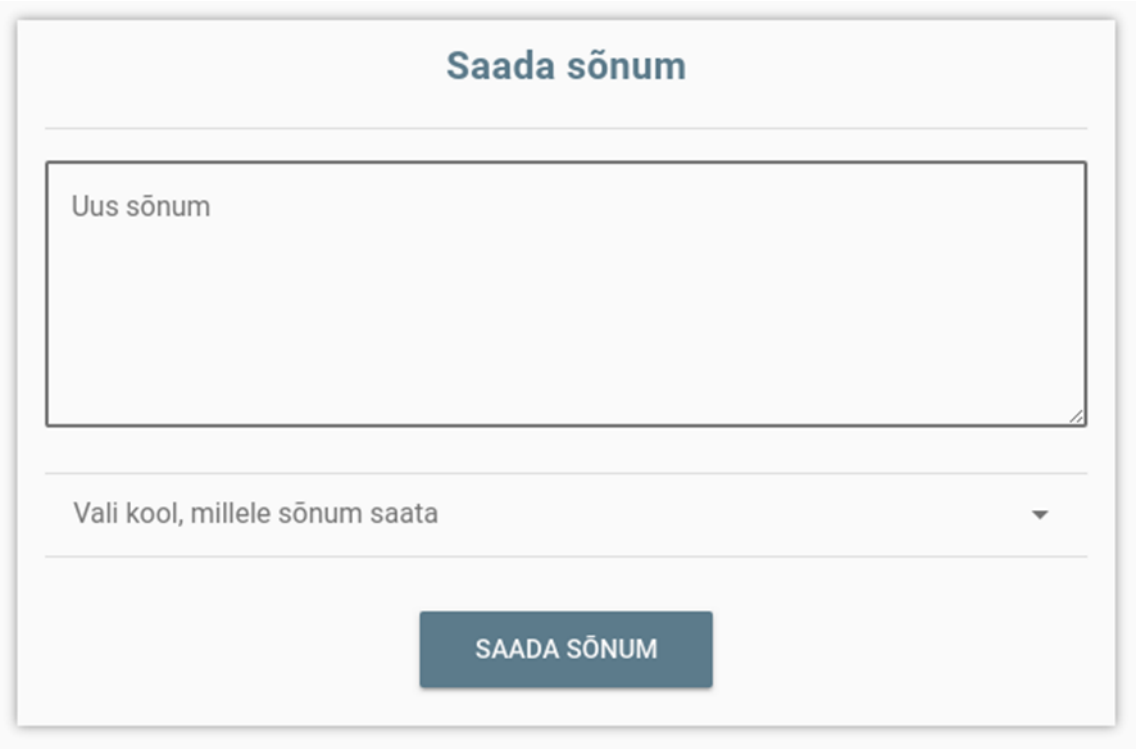
Joonis 14. Informatsiooni lehekülg.

Statistika leheküljel (vt Joonis 15) kuvatava statistika jaoks vajaliku informatsiooni saab tehes päringu API pihta. Hetkel tagastab teenus maajaamade projektiga liitunud koolide arvu, saadetud kosmoselaulu tükkide arvu, kõigi saadetud sõnumite arvu ning kõigi kätte saadud sõnumite arvu. Seda infot kuvatakse lehe paremal küljel. Lehe vaskaul küljel kuvatakse kaht graafikut: neist esimene näitab, mitu protsenti kõikidest saadetud sõnumitest on koolide poolt kätte saadud, ning teine graafik näitab, mitu protsenti kosmoselaulust on saadetud.

Sõnumite leheküljel (vt Joonis 16) asub sõnumi saatmise vorm, kuhu kasutaja saab kirjutada teisele koolile saadetava sõnumi ning valida rippmenüüst maajaamade projektiga liitunud koolide hulgast endale meeldiva adressaadi. Kui sõnum on valmis, võib kasutaja vajutada nuppu sõnumi API-ile saatmiseks.



Joonis 15. Statistika lehekülg.



Joonis 16. Sõnumite saatmise vorm.

Veebilehe mittefunktsionaalsed nõuded olid eestikeelne kasutajaliides ning veebilehe toimimine Google Chrome'i ning Mozilla Firefoxi uusimate versioonidega. Kuna rakenduse kasutajaliides on eestikeelne ning veebileht toimib mittefunktsionaalsetes nõuetes välja toodud brauseritel, võib lugeda mittefunktsionaalsed nõuded täidetuks.

Veebilehe funktsionaalsetest nõuetest ei saanud kõik täidetud, kuna API loomiseks läks autoril planeeritust kauem aega. Avalehel pole hetkel informatsiooni ning statistikat, kuna kahjuks pole praeguseks hetkeks veel täpselt selgunud, millist statistikat ning infot peaks avalehel kuvama. Statistika ning informatsiooni vaated said seevastu aga töö käigus implementeeritud. Samuti on võimalik kasutajatel lehele sisse logida ning administraatoritel lisada süsteemi uusi kasutajaid. Sõnumite lehel on olemas ka vorm uute sõnumite API-le saatmiseks. Töö käigus jäid osad veebilehe vaated viimistlemata ning puudu on ka sõnumite vaates saadetud ning kätte saadud sõnumite loetelu. Kosmoselaulu lehel ei ole hetkel veel nimekirja kooli poolt kätte saadud laulutükkidest. Veebilehel ei ole ka võimalust töölauarakenduse allalaadimiseks.

6 Töölauarakendus

Käesolevas peatükis kirjeldab autor maajaamade projekti töölauarakenduse planeerimist ning realiseerimist.

6.1 Töölauarakenduse planeerimine

Töölauarakenduse jaoks raamistikke valides kaalus autor mitmeid erinevaid võimalusi. Kuna autor oli eelnevalt töölauarakendusi teinud ainult JavaFX'i kasutades, oli autor huvitatud uute raamistike proovimisest, mis aitaksid vältida JavaFX'i kasutamisega seotud probleeme – vähest paindlikkust ja kena ning kasutajasõbraliku liidese loomise keerukust. Esimeseks valikuks osutus Electron[1] raamistik. Electroni abil loodud töölauarakendusi saab kasutada nii Windows, Linux kui ka macOS operatsioonisüsteemides. Electron raamistik kasutab JavaScripti, HTML-i ning CSS-i, mis muudab koodi kirjutamise mugavamaks ning võimaldab luua paremaid kasutajaliideseid kui JavaFX [9].

Töölauarakenduse üks olulisemaid funktsionaalsusi on kooliõpilaste poolt ehitatud antenni ning signaale vastu võtva SDR-raadioga[2] suhtlemine. Raadio ja antenniga suhtlemiseks on eraldiseisev rakendus, mis tuleb töölauarakendusega integreerida. Kuna integreeritav rakendus kirjutatakse Javas, oli töölauarakenduse arendamiseks valitava platvormi juures äärmiselt oluline lihtne ühilduvus Javas kirjutatud koodiga. Electron raamistik seda kahjuks ei võimalda. Seega pidi autor otsustama, kas kirjutada eraldi API, mis suhtleks raadio ning antenniga seotud rakendusega ning mida saaks kasutada Electroni abil loodud töölauarakendus või siis kasutada JavaFX, mis võimaldab Java koodi kasutada. Enne lõpliku otsuse tegemist katsetas autor erinevaid töölauarakenduse loomise lahendusi, mis võimaldasid kasutada Javas kirjutatud koodi, kuid autor leidis, et eksperimentaalsed lahendused, mis toimivad ainult ühe või kahe näidisprojekti jaoks, pole jätkusuutlikud. Autor otsustas, et antud olukorras on mõistlikum kasutada JavaFX'i, kuna ei pea kirjutama veel üht API-t.

[1] https://electronjs.org/

[2] http://www.sdr-radio.com/

6.2 Töölauarakenduse realiseerimine

Töölauarakenduse realiseerimist alustas autor JavaFX projekti loomisega. Autor kasutas ka Mavenit, mis võimaldab lihtsalt kokku ehitada jar-faili, mida saab käivitada erinevates operatsioonisüsteemides.

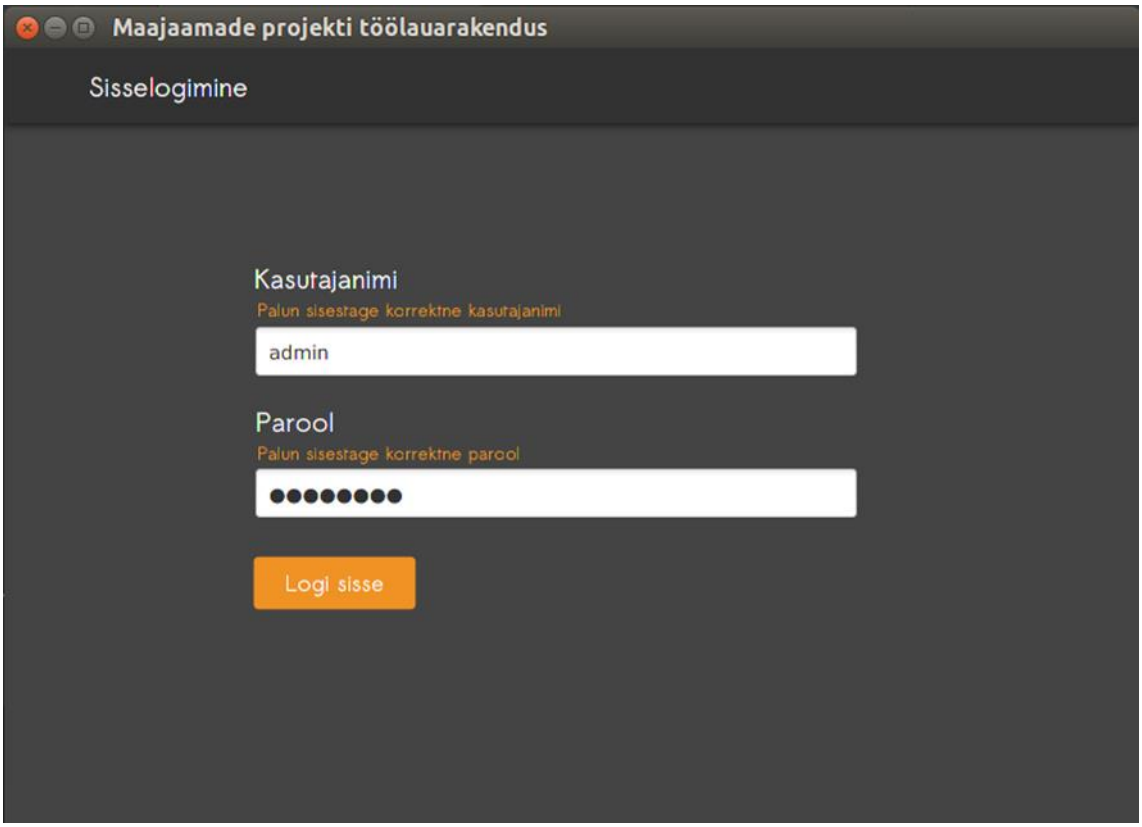
Seejärel otsustas autor luua kasutajaliidese disaini ning selle realiseerida. JavaFX ei paku mugavaid lahendusi kena ning kasutajasõbraliku liidese loomiseks, seega pidi autor katsetama, millised olemasolevatest elementidest ning fontidest jätavad kõige professionaalsema mulje. Lisaks sellele on keerukas leida sobivat elementide paigutust. Üheks võimaluseks on kogu liides genereerida, kirjutades Java koodi. Selle eeliseks on võimalus koodi mitte korrata, kirjutades meetodeid, mida saab taaskasutada erinevate liidese lehtede loomisel. Antud variandi miinuseks on aga keerukus visualiseerida valmivat liidest. Selleks, et liideses toimunud muutusi näha, tuleb kas projekt uuesti ehitada ning käivitada või luua uus jar-fail, mida saab käivitada. Autor leidis, et antud variant muudab liidese loomise tülikaks. Seetõttu otsustas autor luua eraldiseisvad FXML failid iga liidese lehekülje jaoks ning kasutada SceneBuilderit[1] liidese loomiseks. SceneBuilder võimaldab näha, milline loodav liides hakkab tegelikult välja nägema. SceneBuilderi miinusteks on selle rakenduse aeglus, kohmakus ning ebaintuitiivne kasutajaliides, kuid autor leidis, et antud olukorras olid rakenduse eelised olulisemad kui selle puudujäägid.

Töölauarakenduse loomise käigus ei saanud autor kahjuks võimalust integreerida loodavat töölauarakendust raadio ning antenniga suhtleva tarkvaraga, kuna tarkvara pole praeguseks hetkeks veel valminud. Sellest tulenevalt ei teadnud autor, kui palju ning mis kujul informatsiooni raadio ning antenniga suhtlev rakendus satelliitidelt saab, seega ei suutnud autor planeerida töölauarakenduse sõnumite vaadet ning luua funktsionaalsusi, mis võimaldaksid antenniga kinni püütud sõnumite ning kosmoselaulu tükkide kohta infot API-le saata. Autor otsustas, et antud olukorras on mõistlik keskenduda süsteemi teiste komponentide loomisele ning jätta töölauarakenduse realiseerimine tahaplaanile.

Autor realiseeris siiski töölauarakenduse sisselogimise vaate (vt Joonis 17) ning esialgse sõnumite vaate (vt Joonis 18). Sisselogimise vaatel on väljad kasutajanime ning parooli

[1] http://gluonhq.com/products/scene-builder/

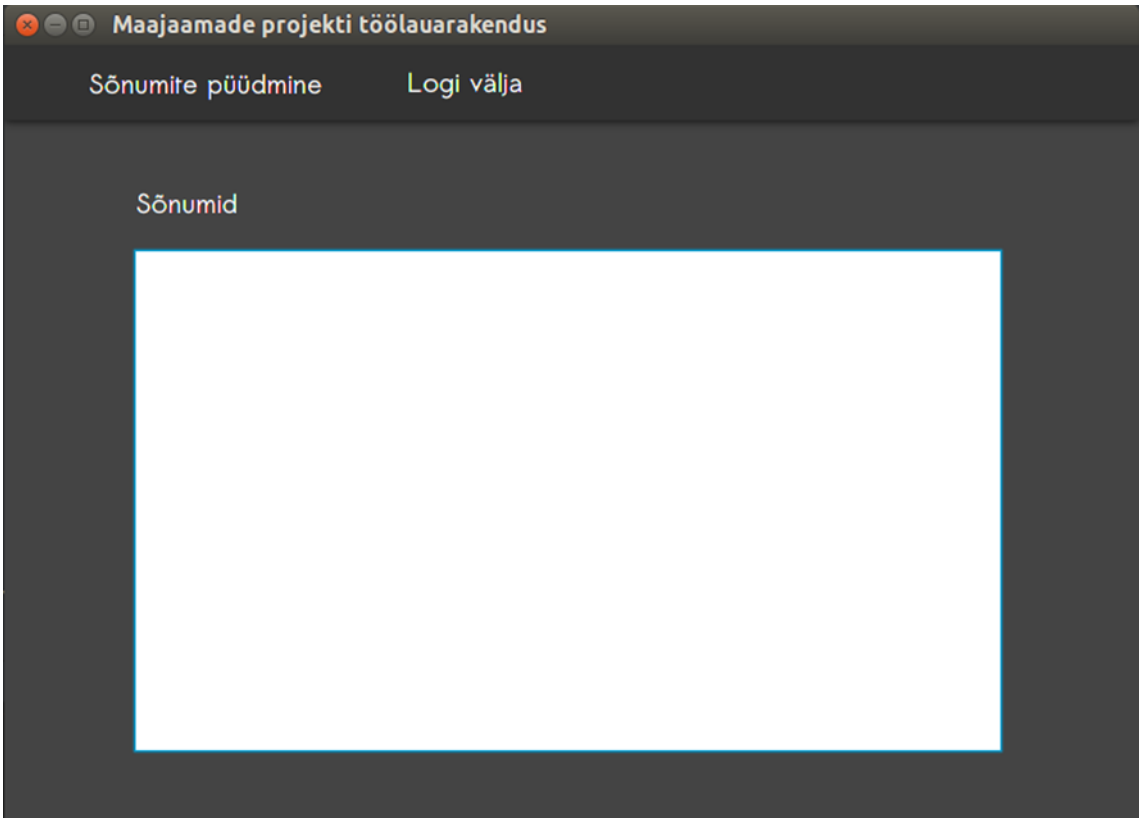
sisestamiseks, vale parooli või kasutajanime korral kuvab rakendus veateateid ning ei lase sisse logida. Praegusel hetkel ei ole veel realiseeritud autentimine loodud API-ga, kuna nagu eelnevalt mainitud, keskendus autor süsteemi teiste komponentide loomisele.



Joonis 17. Töölauarakenduse sisselogimise vaade koos veateadetega.

Sõnumite vaates on olemas menüü, milles on nupp töölauarakendusest välja logimiseks. Vaade on viimistlemata, kuna autor ei teadnud, milliseid andmeid tuleb rakenduses kuvama hakata. Väljalogimise nupule vajutades logitakse kasutaja rakendusest välja ning juhitakse tagasi sisselogimise leheküljele.

Töölauarakendusega seotud funktsionaalsetest nõuetest sai töö käigus realiseeritud ainult üks – rakendusest väljalogimise funktsionaalsus. Sisselogimise jaoks lõi autor vaate, mis tuleb tulevikus ühendada API sisselogimise teenusega. Lisaks sellele lõi autor ka vaate sõnumite jaoks. Suurem osa töölauarakendusega seotud funktsionaalsetest nõuetest olid aga seotud antenni ning raadioga suhtlemisega ning seetõttu ei ole nõuded ka praeguseks hetkeks täidetud.



Joonis 18. Töölauarakenduse viimistlemata sõnumite vaade.

Töölauarakendus vastab kõikidele seotud mittefunktsionaalstele nõuetele, jar-faili saab käivitada nii Windows, Linux kui ka macOS operatsioonisüsteemidel ning loodud kasutajaliides on eestikeelne.

7 Edasise arengu võimalused

Käesolevas peatükis kirjeldab autor maajaamade projekti jaoks loodud süsteemi edasise arengu võimalusi.

Veebilehe peamiseks edasise arengu võimaluseks oleks kõigi olemasolevate API teenuste kasutamine ning vajaliku info veebilehel kuvamine. Tuleks kuvada iga kooli poolt saadetud sõnumeid ning kinni püütud sõnumeid ja laulutükke. Veebilehel võiks olla võimalus laulutükkide kuulamiseks ning tervikliku laulu kokku panemiseks. Lisaks sellele peaks veebileht piirama kasutajate õigusi vastavalt sellele, kas nad on sisse või välja logitud ning kas nad on koolid või administraatorid. Administraatorkasutajad võiksid veebilehel näha informatsiooni projektis osalevate koolide kohta, näiteks osalevate koolide nimekirja.

Oluliseks edasise arengu võimaluseks on ka statistika kuvamine. API-sse tuleks luua teenused, mille abil saaks pärida informatsiooni selle kohta, kui palju laulutükke on iga projektis osalev kool kinni püüdnud. Lisaks sellele võiksid API-s olla teenused, mis tagastavad info selle kohta, millised koolid on kõige rohkem sõnumeid saatnud ning kinni püüdnud. Kosmoselaulu püüdmise edetabel ning statistika sõnumite kohta motiveeriks õpilasi maajaamade projektis aktiivselt osalema ka pärast antenni ja raadio ehitamist ning töötubade lõppu.

Veebileht võiks tulevikus olla ka reklaamimisplatvormiks, veebilehel võiks olla informatsioon satelliidiprogrammi raames toimuvate ürituste kohta, lisaks sellele võiks veebilehel olla ka sektsioon huvitavate artiklite jaoks, et külastajad saaksid olla kursis satelliidi ning meeskonnaliikmete käekäiguga. Avalehel võiks kuvada infot satelliidi asukoha ja järgmiste ülelendude aegade kohta. Lisaks sellele võiks avalehel olla ka satelliidi poolt maale saadetud mõõtetulemusi ning fotosid.

Veebilehele võiks tulevikus lisada ka fotogalerii, mille erinevates albumites saaksid lehe kasutajad näha satelliidi tehtud pilte, fotosid meeskonna töödest ja tegemistest, maajaamade projekti töötubadest ja TTÜ maajaamast ning antennidest. Fotod on hea viis lehe külastajates huvi äratamiseks ning lisaks sellele aitaksid fotod satelliiditehnoloogiaid paremini kooliõpilastele illustreerida.

Töölauarakendusega seotud edasise arengu võimalusi on kõige rohkem, kuna rakenduse realisatsioon jäi SDR-raadio ning koolilaste poolt ehitatud antenniga suhtlemise tarkvara puudumise tõttu poolikuks.

Esimeseks ning kõige lihtsamaks edasiarenduse võimaluseks on API sisselogimise teenuse kasutamine autentimiseks. Lisaks sellele võiks API-t kasutades realiseerida sõnumite staatuse uuendamise teenuse, et märkida koolide poolt kinni püütud sõnumid kätte saaduks. Tuleks realiseerida ka teenus, mis saadab API-ile informatsiooni kinni püütud laulutükkide kohta.

Pärast SDR-raadio ning antenniga suhtlemise tarkvara integreerimist töölauarakendusega saaks kuvada ka antenniga kinni püütud informatsiooni. Lisaks TTÜ satelliidi poolt edastatavatele sõnumitele ning kosmoselaulu tükkidele võiks töölauarakendus kuvada informatsiooni ka teiste ülelendavate satelliitide kohta. Teiste satelliitide informatsiooni kuvamine võimaldab paindlikumat töötubade korraldamist koolides, sest TTÜ satelliidi ülelendude arv ühe päeva jooksul on piiratud, lisaks sellele saaks töötubasid korraldada ka siis kui TTÜ satelliit pole veel kosmosesse jõudnud. Töölauarakendusse võib tulevikus lisada ka veebilehel olevaid funktsionaalsusi, näiteks uute sõnumite lisamist või statistika kuvamist.

8 Kokkuvõte

Käesoleva töö eesmärkideks oli luua andmebaasitäiendus maajaamade projektiga seotud info hoiustamiseks, luua API projekti informatsiooni haldamiseks, luua veebileht, mida saaksid kasutada nii maajaamade projektist osavõtvad koolid, projekti korraldajad kui ka TTÜ Satelliidiprogrammist ja maajaamade projektist huvitatud inimesed, ning luua töölauarakendus, mida koolid saaksid kasutada satelliitidelt saadud sõnumite vaatamiseks.

Esimese eesmärgi saavutamiseks selgitati välja maajaamade projekti tegevuste ning kasutajatega seotud informatsioon, mida tuleks andmebaasis hoida. Loodi olemisuhte diagrammid andmebaasitabelite planeerimiseks. Lisati uus andmebaasiskeem TTÜ satelliidi missiooni juhtimise andmebaasi ja lisati loodud andmebaasiskeemi kõik planeeritud tabelid.

Teise eesmärgi saavutamiseks loodi uus API maajaamade projekti jaoks. API loomise ulatus suurenes töö käigus, sest tuli luua uus avaliku veebi projekt. API-s realiseeriti kõik funktsionaalsetest nõuetest lähtuvad teenused. Testiti ühiktestidega loodud äriloogikat ning kontrollerklasside meetodeid. Turvati API otspunkte autentimise ning autoriseerimise abil. Lisaks sellele testiti käsitsi kõiki API otspunkte ning teenuste toimimist nii korrektsete kui väärade sisendite korral.

Kolmanda eesmärgi saavutamiseks loodi uus veebileht. Veebilehe loomine osutus planeeritust mahukamaks, kuna tuli luua uus avaliku veebi veebilehe projekt. Veebilehe külastajad saavad lehele sisselogimata lugeda TTÜ satelliidi ning maajaamade projektiga seotud informatsiooni, tutvuda statistikaga, külastada avalehte ning sisselogimise lehte. Veebilehele sisselogitud koolid saavad näha sõnumite ning kosmoselaulu vaateid ning saata sõnumeid. Veebilehele sisselogitud projekti korraldajad saavad süsteemi lisada uusi koole ning administraatorkasutajaid. Töö käigus jäid implementeerimata sõnumite ja kätte saadud laulutükkide nimekirjade kuvamine ning administraatorite võimalus näha koolidega seotud informatsiooni.

Neljanda eesmärgi saavutamiseks loodi töölauarakenduse kasutajaliides, milles on koolidel võimalik sisse logida, välja logida ning sõnumite vaadet näha. Töö teiste elementide realiseerimise mahu suurenemise ning antenni ja raadioga suhtleva tarkvara puudumise tõttu jäid realiseerimata töölauarakenduse API-ga suhtlemine ning töölauarakenduse ühendamine antenni ja raadioga suhtleva tarkvaraga.

Kasutatud kirjandus

[1] 2017/2018. Õppeaasta arvudes
[WWW] https://www.hm.ee/sites/default/files/2017-2018_oppeaasta_arvudes.pdf (08.05.2018)

[2] Comparison with Other Frameworks
[WWW] https://vuejs.org/v2/guide/comparison.html (08.05.2018)

[3] Database Schema Recommendations for an Application
[WWW] https://wiki.postgresql.org/wiki/Database_Schema_Recommendations_for_an_Application (01.05.2018)

[4] L3 Documentation
[WWW] https://gitlab.cs.ttu.ee/tut-satellite/mcs/wikis/l3-documentation (17.04.2018)

[5] REST Resource Naming Guide
[WWW] https://restfulapi.net/resource-naming/ (14.04.2018)

[6] Twitter
[WWW] https://en.wikipedia.org/wiki/Twitter (01.05.2018)

[7] Factor, P. Schema Based Access Control for SQL Server Databases
[WWW] https://www.red-gate.com/simple-talk/sql/sql-training/schema-based-access-control-for-sql-server-databases/ (01.05.2018)

[8] Neuhaus, J. Angular vs. React vs. Vue: A 2017 comparison
[WWW] https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176 (07.05.2018)

[9] Pot, J. What Are Electron Apps, and Why Have They Become So Common?
[WWW] https://www.howtogeek.com/330493/what-are-electron-apps-and-why-have-they-become-so-common/ (06.05.2018)

Lisa 1 – Koolidega seotud teenused

HTTP meetod	**URI**	**Teenuse kirjeldus**
GET	*/schools*	Tagastab loetelu kõigist koolidest.
GET	*/schools/{id}*	Tagastab identifikaatorile vastava kooli.
GET	*/schools/username/{username}*	Tagastab kasutajanimele vastava kooli.
POST	*/schools/sign-up*	Lisab süsteemi uue kooli ehk kasutaja.

Lisa 2 – Sõnumitega seotud teenused

HTTP meetod	**URI**	**Teenuse kirjeldus**
GET	*/messages*	Tagastab loendi kõigist süsteemi lisatud sõnumitest.
POST	*/messages*	Lisab uue sõnumi andmebaasi.
GET	*/messages/{id}*	Tagastab parameetriks olevale identifikaatorile vastavad sõnumid.
POST	*/messages/{id}?state={state}*	Muudab identifikaatorile vastava sõnumi staatuse parameetriga kaasa antud staatuseks.
GET	*/messages/sender/{id}*	Tagastab kõik identifikaatorile vastava kooli poolt saadetud sõnumid.
GET	*/messages/receiver/{id}*	Tagastab kõik identifikaatorile vastavale koolile saadetud sõnumid.
GET	*/messages/receiver/{id}/received*	Tagastab kõik identifikaatorile vastava kooli poolt kätte saadud sõnumid.

Lisa 3 – Laulutükkidega seotud teenused

HTTP meetod	**URI**	**Teenuse kirjeldus**
GET	*/song-pieces*	Tagastab loendi kõigist laulutükkidest.
POST	*/song-pieces*	Lisab andmebaasi uue laulutüki.
GET	*/song-pieces/{id}*	Tagastab identifikaatorile vastava laulutüki.
POST	*/song-pieces/{id}*	Muudab identifikaatorile vastava laulutüki saadetuks.
GET	*/song-pieces/{id}/sending-time*	Tagastab identifikaatorile vastava laulutüki saatmisaja.
GET	*/song-pieces/sent*	Tagastab kõik saadetud laulutükid.
GET	*/song-pieces/not-sent*	Tagastab kõik saatmata laulutükid.
GET	*/song-pieces /school/{id}*	Tagastab kõik konkreetse kooli poolt kinni püütud laulutükid.
POST	*/song-pieces/{song-piece-id}/school/{school-id}*	Muudab laulutüki kooli poolt kinni püütuks.

Lisa 4 – Statistikaga seotud teenused

HTTP meetod	**URI**	**Teenuse kirjeldus**
GET	*/statistics*	Tagastab maajaamade projektiga seotud statistikat.