TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Karl-Kevin Köörna 206055IADB

Elektriauto laadimise juhtsüsteemi arendamine

Bakalaureusetöö

Juhendajad: Meelis Antoi
Magistrikraad

Ivari Horm
Bakalaureusekraad

Tallinn 2023

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: Karl-Kevin Köörna

05.05.2023

Annotatsioon

Töö eesmärgiks on arendada elektriautolaadijate juhtsüsteem, mis võimaldab Eleport'il laadimisteenust korraldada. Juhtsüsteemi arenduse vajadus tekkis eelnevalt kasutatud kolmanda osapoole tähistamata teenuse mittesobilikuks muutumisest.

Juhtsüsteem on tsentraalne server, mille peamine ülesanne on vahendada sõnumeid laadijate ja teiste osapoolte vahel. Osapoolte alla kuuluvad laadijad, ärisüsteem, välisteenused ja kasutajarakendused.

Töö käigus uuriti laadimisteenusega seotud komponente ja osapooli ning nende püstitatud nõudeid juhtsüsteemi vastu. Arvestades tekkinud nõudeid, disainiti ja arendati juhtsüsteem, mis saab kõikide seotud osapooltega liidestuda.

Töö lõpuks valmis juhtsüsteem, mis vastas kõikidele töös püstitatud nõuetele ja võimaldab kasutajatel Eleport'i laadijates laadida nii kaardiga kui ka mobiilirakendusega. Töö käigus sai juhtsüsteem testitud tootmiskeskkonnas kasutamise jaoks vastuvõetavale tasemele.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 32 leheküljel, 6 peatükki, 19 joonist, 2 tabelit.

Abstract

Development of a Backend System for Electric Vehicle Chargers

The objective of the thesis is to develop a control system for electric vehicle chargers, which in turn enables Eleport to operate an electric vehicle charging service. The necessity for such a system arose due to the inadequacy of a previously utilized third-party white label service.

The control system functions as a central server, facilitating the exchange of messages between chargers and other parties. Aforementioned parties include chargers, the business system, external services and user applications.

During the work, components and parties involved in providing a charging service were studied, along with their requirements for the control system. Taking into account the arisen needs, a control system was designed and developed that can integrate with all the relevant parties.

During the course of the work, a control system was developed that fulfilled all the requirements set forth and allows users of Eleport chargers to charge their electric vehicles using either a card or a mobile application. The control system also underwent testing to an extent, which deemed it acceptable for use in a production environment.

The thesis is in Estonian and contains 32 pages of text, 6 chapters, 19 figures, 2 tables.

Lühendite ja mõistete sõnastik

API	rakendusliides, reeglistik, mille alusel rakendused üksteisega suhtlevad [1]
B2B	*Business to Business*, firmadevaheline kaubandus [1]
CH	*Central Hub* või *Clearing House*, ettevõtte ja/või tarkvara roll, mis tegeleb rändluse teenuse korraldamisega [2]
CPO	*Charge Point Operator*, ettevõtte ja/või tarkvara roll, mis haldab laadijaid [3]
CSO	*Charging Station Owner*, laadija (ja laadimisasukoha) omanik [4]
DSO	*Distribution System Operator* ehk elektriteenusepakkuja [3]
eMSP	*eMobility Service Provider*, ettevõtte ja/või tarkvara roll, mis haldab kliente [3]
EV	*Electric Vehicle* ehk elektrisõiduk
EVSE	*Electric Vehicle Supply Equipment*, elektriauto laadija komponent, mis varustab autot elektriga
HTTP	*HyperText Transfer Protocol*, andmevahetusprotokoll Internetis dokumentide vahetamiseks [1]
JSON	*JavaScript Object Notation*, JavaScript'i alamhulgal põhinev standardne andmevahetusvorming [5]
OCPI	*Open Charge Point Interface*, suhtlusprotokoll rändluse korraldamiseks [6]
OCPP	*Open Charge Point Protocol*, laadija ja juhtsüsteemi vaheline suhtlusprotokoll [7]
OICP	*Open InterCharge Protocol*, suhtlusprotokoll rändluse korraldamiseks [6]
RFID kaart	*Radio-Frequency IDentification card*, kontaktivaba kiipkaart, kasutab kaardi sisse paigutatud antenni, mis on ühendatud kiibiga, et edastada kiibis olev informatsioon lugejasse [8]
SOAP	*Simple Object Access Protocol*, komplekt kokkuleppeid XML põhise suhtluse korraldamiseks üle HTTP [1]
standardteek	Programmeerimiskeelde sisseehitatud funktsionaalsus
TLS	*Transport Layer Security*, transpordikihi turbeprotokoll [1]

XML	*Extensible Markup Language*, märgistuskeel struktureeritud andmevahetusvormingute genereerimiseks [5]

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Elektriautode arvukuse kasvu tõttu on järjest rohkematel inimestel vaja oma uut elektriautot laadida, mis omakorda tõstab nõudlust elektriautolaadijate ja laadimisteenuste järele. Laadimistaristu arendamine võimaldab elektriautoomanikel oma autot laadida.

Eestis laadimisteenust pakkuva ettevõtte Eleport OÜ ja ta tütarettevõtte Eleport Innovations OÜ eesmärgiks ongi nimelt laadimise võimaldamine. Tehakse pidevat tööd nii füüsilise taristu laiendamisega kui ka tarkvaralise taristu arendamisega. Eleport oli pikka aega kasutanud oma laadijate juhtimiseks ja haldamiseks kolmanda osapoole pakutud tarkvaralahendust, kuid laadimisvõrgu kasvu ja ärinõuete lisandumisega ei olnud see lahendus enam jätkusuutlik. Eleport soovis luua omaenda lahenduse, mida saab, kolmanda osapoolele lootmata paremini kohandada praeguse laadimismaastiku vajadustele ning millega saab pakkuda laadimisteenust jätkusuutlikult, elimineerides vaheteenustasud ja nende kasvamise ohu.

Käesoleva lõputöö eesmärgiks on arendada elektriautolaadijate juhtsüsteem, mis võimaldab elektriautolaadijatel suhelda välisteenustega ning elektriautoomanikel oma autosid Eleport'i laadijates mugavalt laadida.

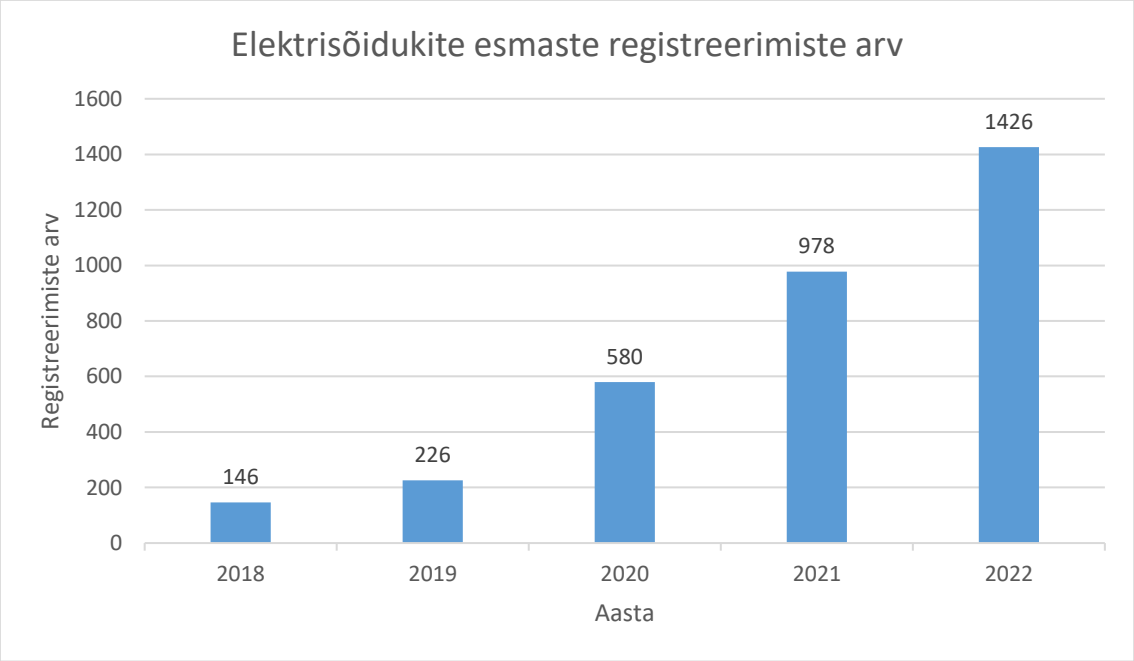
Töö esimeses osas uuritakse ja võrreldakse juhtsüsteemi arendamiseks vajalikke komponente. Töö teises osas analüüsitakse ja seatakse paika andme- ja suhtlusmudelid ning süsteemi üldine ülesehitus. Lõpuks rakendatakse disain tarkvaraarenduses. Lisaks testitakse valminud lahendust, hinnatakse lahenduse pädevust ja leitakse võimalusi edasiarenduseks.

2 Taust

Käesolev peatükk selgitab, millest projekt sündis, mida on vaja projekti jaoks teha ja kuidas projekti tegemisele lähenetakse.

2.1 Probleemi püstitus

Eestis soetatakse iga aasta järjest rohkem elektriautosid. Elektriautode esmaste registreerimiste arv tõusis 2022. aastal võrreldes eelmisega 45% (vt Joonis 1). Lisaks kiitis Euroopa Parlament 2023. aasta veebruaris heaks määruse, mille kohaselt tohib 2035. aastast alates müüa ainult autosid, mis ei tekita CO2 heidet [9]. Elektriautod kuuluvad nende hulka, mistõttu on põhjust uskuda, et elektriautode soetamisnäitajad ei lange.



Joonis 1. Elektrisõidukite esmaste registreerimiste arv aasta lõikes [10]

Kõige lihtsakoelisemad elektriauto laadijad on võrguühenduseta kastid, mis annavad autole voolu ja näitavad oma olekut sisseehitatud ekraanil. Nende kasutamisel on märgatavad puudujäägid võrreldes intelligentsete, võrgu- ja juhtsüsteemiühendusega, laadijatel. Nimelt ei saa lihtsakoeliste laadijatega:

- teostada kaughooldust;
- alustada, lõpetada ja jälgida laadimissessiooni nutiseadmest;
- salvestada, raporteerida ja analüüsida laadimisandmeid;
- korraldada laadimisteenust.

Eleport pakub ettevõttena laadimisteenust, teenuse korraldamise eelduseks on intelligentsete laadijate kasutamine. Intelligentsed laadijad vajad omakorda toimimiseks juhttarkvara, mis oli Eleporti’s enne hallatud kolmanda osapoole poolt. Kuna Eleport ei soovi enam kasutada kolmanda osapoole pakutud lahendusi, on tema laadimistaristusse vaja arendada uus juhtsüsteem, mille algne versioon käesoleva lõputöö raames valmib.

2.2 Lähtetingimused

Ettevõtte, kus autor töötab, on arendamas elektriautode laadimisteenust. Arenduse tuum hõlmab endas juhtsüsteemi ehk vajadust arendada juhtserveri tarkvara, mis enda külge ühendunud laadijaid haldab. Kuna ettevõttel on juba olemas enda mobiilirakendused, kliendiportaal ja laadijad ning kolmandad osapooled, kellele peab ka andmeid jagama, peab loodav süsteem arvestama olemasolevate nõuetega.

Eelmainitust ilmnevad süsteemile järgmised põhinõuded:

- süsteem peab toetama suhtlusprotokolle, mida olemasolevad laadijad toetavad;
- laadimist peab saama alustada ja lõpetada füüsilise kaardiga;
- laadimist peab saama alustada ja lõpetada välisest rakendusest;
- laadimise andmed peavad jõudma andmebaasi ja kasutajarakendustesse;
- laadimise andmed peavad jõudma välistesse teenustesse;
- peab olema võimalik juurde arendada rändluse tugi.

Lisas 2 ja 3 on plokkskeemina kujutatud laadimise protsessi ülevaade, kui laadimine on alustatud füüsilise RFID (ingl. k. *Radio-Frequency IDentification*) kaardiga või välisest

mobiilirakendusest. Plokkskeemis on kujutatud laadija, juhtsüsteemi ja mobiilirakenduse vaheline suhtlus. Skeemil kujutatu hõlmab esimest nelja põhinõuet.

2.3 Metoodika

Viidi läbi firmasisene intervjuu, mille põhjal koguti informatsiooni olemasoleva süsteemi ja uue süsteemi vajaduste kohta. Analüüsiti laadimisteenuse korraldamiseks vajalikke osapooli ja protokolle ning kasutati MoSCoW prioriteetimise meetodit, et leida arendusele asjalik skoop. Viidi läbi iteratiivne ja inkrementaalne tarkvaraarendus. Kuna laadijate käitumine on ettearvamatu, toimus testimine alguses uuriva testimise tehnika järgi. Hiljem toimus testimine väljakujunenud testiprotokolli põhiselt.

Projekti lõpus tehti ülevaade saavutatud olukorrast ja süsteemi pädevusest ning toodi välja eelnevalt uuritu põhjal edasiarenduse võimalused.

3 Teoreetilised alused

Teades põhinõudeid on vaja uurida arenduseks vajalikke komponente ja valmislahendusi.

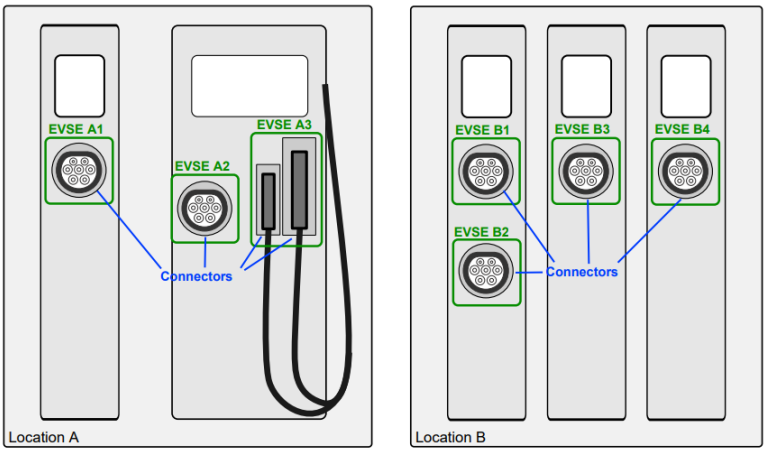
3.1 Laadijad ja nende võrgud

Elektriauto laadija (ingl. k. *electric vehicle charger* ehk *EV charger*) on kui adapter, mis annab elektriautodele ligipääsu elektrivõrgule ehk liidestab elektriauto elektrivõrku, ning vajadusel muundab elektrivõrgust saadud voolu auto akule sobivaks [11]. Levinud on kolm peamist laadija tüüpi: esimese taseme, teise taseme ja DC laadijad (vt Tabel 1). Laadija tüüp juhtsüsteemi toimimist otseselt ei mõjuta, kuid laadimiskiiruse erinevuse tõttu peab arvestama, et laadimissessioonid võivad laadija piiratud võimsuse tõttu kohati väga kaua kesta. Lisaks on võimalik, et autoomanikul endal pole võimalik sessiooni lõpetada ja autole järgi minna kohe kui laadimine lõppeb.

Tabel 1. Laadijate tüüpide võrdlus [12]

Tüüp	**Kiirus**	**Eesmärk**
Tase 1	Aeglane	Kodulaadija, kuid esineb ka avalikes kohtades
Tase 2	Keskmine	Kodulaadija või avalik laadija
DC	Kiire	Avalik laadija

Laadija ise on kast, mis sisaldab üht või mitut EVSE’t (ingl. k. *Electric Vehicle Supply Equipment*). EVSE on laadija komponent, mis annab autole elektrit. Laadijal võib olla üks või mitu EVSE’t. Laadijas saab korraga laadida nii mitu autot, kui mitu EVSE’t laadijal on. EVSE küljes on omakorda üks või mitu erinevat tüüpi pistikut (ingl. k. *connector*), mille abil on võimalik EVSE külge ühendada elektriauto (vt Joonis 2). Juhtsüsteemi vaatepunktist peab arvestama, et ühel laadijal võib korraga toimuda mitu laadimist ning üks laadimine võib korraga muuta mitme pistiku olekut.

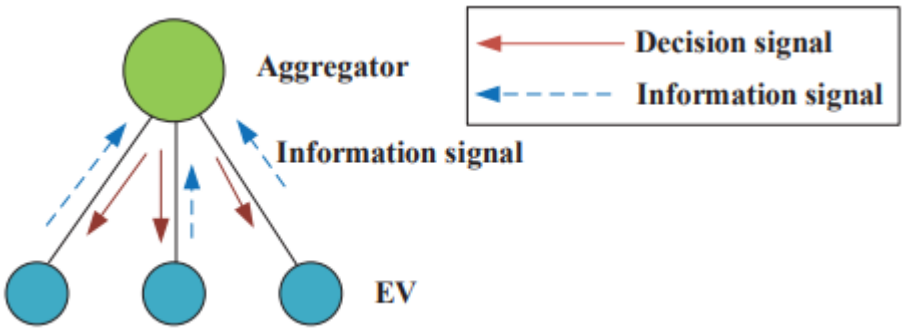


Joonis 2. Laadija topoloogia skeem [13]

Laadijatel on tavaliselt sisseehitatud veebiliides, mille abil saab neid seadistada. Erinevatel laadijatel on ka erinevad juhtpaneelid ja seadistusvõimalused, mille tõttu ei saa juhtsüsteem eeldada, et iga seadistusest tingitud funktsionaalsus igal laadijal olemas on. Veebiliideses saab üldiselt seadistada ka juhtserverit, mille külge laadija ühendub ja oma toetatud protokolli alusel suhtleb.

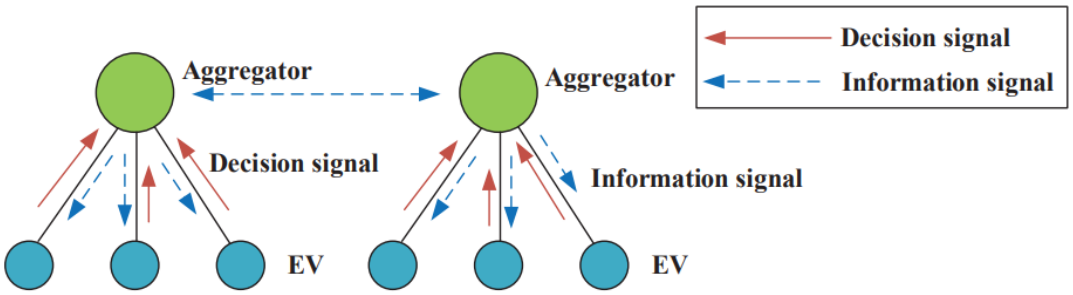
Laadijad võivad opereerida juhtsüsteemi ühenduseta võrguväliselt, mis juhul on elektriauto omanikul ülemvõim laadimise korraldamise üle. Sellist lahendust nimetatakse ka kohalikul kontrollil põhinevaks laadimisstrateegiaks [14]. Selline lahendus sobib näiteks kodukasutuseks, kus laevad ainult usaldatud isikud, ei ole vaja teostada kaughooldust, korraldada kliendipõhist arveldust ega liidestuda teiste teenustega või rakendustega. Laadimisteenuse pakkumiseks ei ole eelmainitud lahendus aga sobilik, kuna laadimisprotsessi ei ole võimalik piisavalt granulaarselt juhtida ja jälgida. Seetõttu jäävad juhtsüsteemi ühenduseta laadijad ka käesoleva töö skoobist välja. Kõikidel järgmistel lahendustel on laadija juhtsüsteemiga ühenduses.

Tsentraalsel kontrollil põhinev strateegia näeb ette, et laadijad on ühendunud ühe juhtserveri külge, mis teeb laadimisprotsesside üle otsuseid [14]. Laadimisteenuse korraldamiseks saadavad laadijad laadimissessiooni kohta andmeid serverile, mis teeb nende andmete ja äriliste piirangute põhjal otsuseid (vt Joonis 3). Joonisel on kujutatud juhtserver kui *agg*regator ja laadija ning elektriauto kui EV (ingl. k. *Electric Vehicle*). Lisaks võimaldavad juhtserveriga lahendused kergemalt korraldada dünaamilist käitumist ja liidestusi, mida peaks vastasel juhul kasutaja ise seadistama ja haldama.



Joonis 3. Tsentraalsel kontrollil põhinev laadimisstrateegia [14]

Hajutatud ja hierarhilisel kontrollil põhinevad strateegiad näevad ette, et mitmed laadijate kogumid on ühendunud erinevate juhtsüsteemide külge, mis suhtlevad omavahel otse või läbi tsentraalse süsteemi (vt Joonis 4) [14]. Eelmainitud juhtsüsteeme võib omada üks suur ettevõte või ka erinevad ettevõtted. Laadijavõrk kujuneb üldiselt selliseks, kui ettevõtted lasevad oma klientidel üksteise laadijates laadida, mille tõttu peavad nende süsteemid ka üksteisega suhtlema. Hajutatud ja hierarhilist topoloogiat on mõttekas rakendada mitme väiksema ettevõtte koostöös rändluse korraldamiseks või ühe väga suure ettevõtte siseselt, et hajutada koormust ja tõsta töökindlust. Nende topoloogiate loomisel aitavad kaasa konkreetselt defineeritud rändlusprotokollid, mida ei pea tingimata ainult mitme ettevõtte vaheliseks rändluseks kasutama.



Joonis 4. Hajutatud kontrollil põhinev laadimisstrateegia [14]

3.2 Laadimisteenuse osapooled

CPO (ingl. k. *Charge Point Operator*) on ettevõtte ja/või tarkvara roll, mis juhib ja haldab laadijavõrku ning pakub teistele osapooltele liidestusi, et laadijatega suhelda [3]. Enamus CPO tegevusest toimub taustal ning ei ole kliendile otseselt nähtav.

eMSP (ingl. k. *eMobility Service Provider*) on ettevõtte ja/või tarkvara roll, mis tegeleb kliendile laadimisteenuse kättesaadavaks tegemisega [3]. eMSP vastutuse alla kuulub arveldus, avalikud rakendusliidesed ja kasutajaliidesed. Suur osa eMSP tegevusest on kliendiga otseselt seotud ja talle ka nähtav.

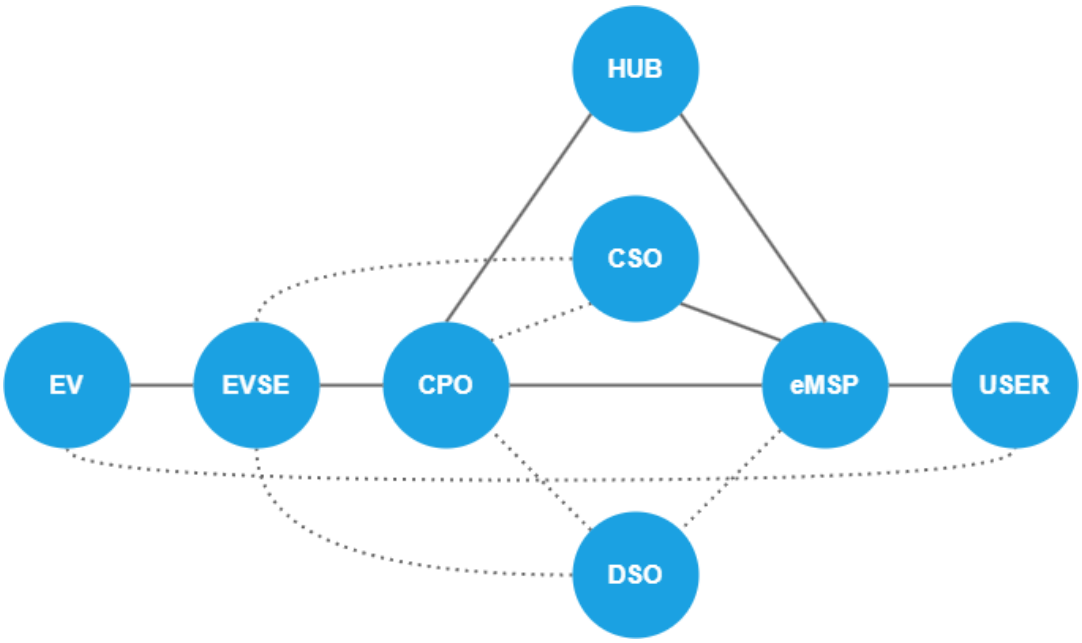
DSO (ingl. k. *Distribution System Operator*) ehk elektriteenusepakkuja on ettevõtte, kes jagab laadimisettevõttele elektrienergiat [3]. DSO võib saata CPO'le uuendusi elektrivõrgu seisundi kohta, et süsteem töötaks efektiivsemalt. CPO kohustustes võib olla ka vajadus saata DSO'le energiaraporteid.

HUB või CH (ingl. k. *Central Hub* või *Clearing House*) on ettevõtte ja/või tarkvara roll, mis tegeleb rändluse teenuse pakkumisega ehk võimaldab kasutajatel laadida teiste ettevõtete laadijates [2]. CH on vahendusteenus mitmete CPO'de ja eMSP'de vahel. CH ei ole tingimata vajalik rändluse korraldamiseks, kuna osapooled võivad teineteisega ka otse suhelda, kuid CH olemasolu elimineerib vajaduse kõikide osapooltega eraldi liidestuda.

CSO (ingl. k. *Charging Station Owner*) ehk laadija omanik on laadija ja asukoha omanik, kes on ühendanud oma laadija CPO süsteemiga ning lubab oma laadijas eMSP kaudu seatud piirangutega autojuhtidel laadida [4].

Käesolevas töös arendatav juhtsüsteem on CPO rollis. eMSP süsteem, millega juhtsüsteem suhtlema hakkab, on ettevõttes teine arendusprojekt, mis jääb käesoleva töö skoobist välja. CSO'd määravad eMSP kasutajaliideses oma laadijatele hinna, laadimiseks lubatud kasutajad, lahtioleku ajad ja muud parameetrid, millega CPO peab arvestama. DSO rollis on Elering, kellele peab saatma energiaraporte.

Joonisel 5 on kujutatud kõik eelkirjeldatud laadimisteenuse osapooled ja nendevahelised seosed.



Joonis 5. Laadimisteenuse osapooled

On olemas ka B2B (ingl. k. *Business to Business*) tähistamata (ingl. k. *white label*) teenused, mis täidavad CPO ja eMSP rolli ning seeläbi pakuvad firmadele ümberkujundatavat laadimisvõrgu halduseks mõeldud valmislahendust, mida on võimalik kasutada oma teenuse osutamiseks. Üks sellistest valmislahendustest on Virta. Valmislahendused aitavad ettevõttel jõuda kiiremini turule ning pakkuda stabiilset teenust, kuid kui ettevõtte ärinõuded (siinjuhul laadimisvõrgu suhtes) kasvavad, võivad tähistamata teenused jääda oma funktsionaalsuse poolest puudulikuks ja teenustasude poolest ebasobivaks. Lisaks, sõltub tähistamata teenust kasutava ettevõtte peaprotsess otseselt teise firma tehtud otsustest, mis võivad olla aeglased või ebasobivad [15]. Eelmainitud põhjuste tõttu eemalduski Eleport tähistamata teenuse kasutamisest, millest omakorda tekkis vajadus käesolevas töös kirjeldatud juhtsüsteemi järele.

3.3 Juhtsüsteemi suhtlus laadijaga

Laadija ja juhtsüsteemi vaheliseks suhtluseks on olemas standardiseeritud ja vabalt kättesaadava spetsifikatsiooniga OCPP (ingl. k. *Open Charge Point Protocol*) protokoll [7]. Enamus laadijatootjaid lisavad OCPP toe otse oma laadijasse ja kui juhtsüsteem on loodud suhtlema ka üle sama protokolli saab toimuda suhtlus kõikide OCPP laadijatega. Kui on soov toetada mitme firma laadijaid, siis on parim toetada ka OCPP protokolli. Firmaomase protokolli väljatöötlemine nõuab väga suurt investeeringut ja ka laadijate

endi tootmist. Lisaks, OCPP asemel või kõrvalt teiste protokollide toetamine ei ole asjalik, kuna enamus turul olevaid laadijaid juba kasutavad OCPP protokolli ja paljud teiste ettevõtete protokollid on olemuselt firmaomased (ingl. k. *proprietary*).

Et toetada kõiki ettevõttes töös olevaid laadijaid peab toetama OCPP protokolli versioone 1.5 ja 1.6, mis ilmusid aastal 2012 ja 2015 [7]. Varasemad OCPP versioonid ei ole enam üldises kasutusel ja uuemad OCPP versioonid ei ole veel laias kasutuses.

OCPP protokoll defineerib hulga sõnumeid, mida laadija ja juhsüsteem omavahel saatma peavad. Ettevõttes olemasolevad laadijad näevad ette, et on vaja toetada protokolli kaht versiooni: OCPP-1.5 ja OCPP-1.6. Eelmainitud versioonide ühed peamised erinevused on andmevorming ja sõnumite andmeväljad. Esimene versioon kasutab XML (ingl. k. *Extensible Markup Language*) andmevormingut ja teise versioon JSON (ingl. k. *JavaScript Object Notation*) andmevormingut. Tuleb otsustada, mis sõnumite tüüpe peab süsteem toetama. Selgema pildi saamiseks saab kasutada MoSCoW prioriteetimise meetodit, mille järgi vajadused jaotuvad nelja kategooriasse: peab olema, peaks olema, võiks olla, ei hakka olema [16]. Käesolevas töös peab kahte esimesse kategooriasse paigutatud sõnumite tüüpe kindlasti toetama.

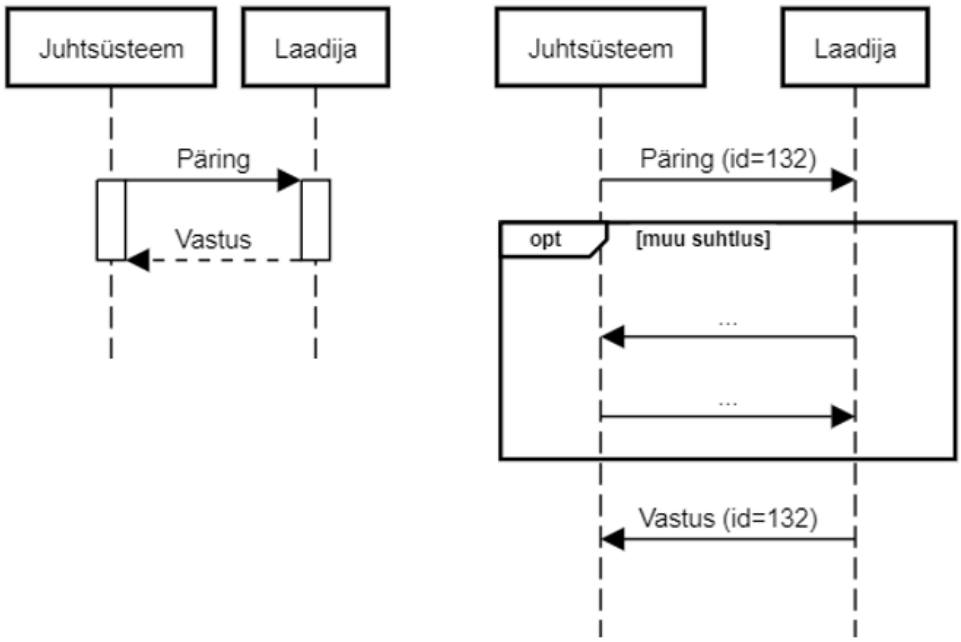
- **Peab olema**
    - *BootNotification* – Laadijad peavad saama süsteemi külge ühenduda.
    - *Heartbeat* – Ühendus peab püsima ja süsteemil peab olema võimalik kontrollida laadija olekut.
    - *Authorize* – Süsteem peab läbiviima laadimisõiguse kontrolli.
    - *StartTransaction* ja *StopTransaction* – Laadimissessiooni peab saama alustada ja lõpetada.
    - *RemoteStartTransaction* ja *RemoteStopTransaction* – Laadimissessiooni peab saama alustada ja lõpetada välisest rakendusest.
    - *MeterValues* – Laadimissessiooni infot peab salvestama.
    - *StatusNotification* – Pistikute olekumuutus peab ilmnema kasutajarakendustesse.

- **Peaks olema**
    - *Reset* – Tehniline tugi peaks saama halvaks läinud siseoleku korral laadija taaskäivitada.
    - *UnlockConnector* – Klienditugi peaks saama kinni jäänud pistiku lahti sundida.
    - *ChangeAvailability* – Avarii või eriolukorra puhul peaks saama laadijas keelata laadimise.
    - *GetConfiguration* ja *SetConfiguration* – Laadijate seadistust peaks saama hallata tsentraalselt.
- **Võiks olla**
    - *TriggerMessage* – Võiks saada sundida laadijat oma olekut uuesti saatma, kui on viimastes andmetes kahtlus.
    - Kõik püsivaraga seotud sõnumid laadijate uuendamiseks üle kaugühenduse.
    - Kõik broneeringutega seotud sõnumid, et saaks broneerida laadimisaja päeva jooksul või teel laadijasse.
- **Ei hakka olema**
    - *DataTransfer* – Kõik suhtlus käib OCPP protokolli järgi, juhtsüsteemil ei ole vaja laadijalt muid andmeid vastu võtta.
    - Kõik laadimisprofiiliga seotud sõnumid, kuna see funktsionaalsus ei ole laadimise juures esmatähtis.
    - Kõik lokaalse õiguskontrolliga seotud sõnumid, kuna selle rakendamine üle terve võrgu ei ole jätkusuutlik.

Osad sõnumid on defineeritud kui sessiooniga seotud sõnumid (ingl. k. *transaction-related messages*), mida laadija peab uuesti saatma, kui juhtsüsteemilt ei tule positiivset vastust. Siinjuhul on oht, et juhtsüsteemi negatiivse vastuse tõttu jääb laadija kinni

lõputusse või väga pikka tsüklisse, mis aja jooksul laadija tavapärane funktsionaalsus võib olla häiritud. Laadija sõnumeid töödeldes peab arvestama sellega, et nende sõnumite töötlemise puhul ainult peaprotsessi takistavad vead lõppevad laadijale veateate saatmisega.

OCPP-1.5 suhtlus käib üle sünkroonse SOAP API (tuntud ka kui OCPP-S), kus päring ja vastus on tihedalt seotud, vastus on saadetud sama HTTP (ingl. k. *HyperText Transfer Protocol*) päringu vastuses (vt Joonis 6) [7]. OCPP-1.6 suhtlus käib asünkroonselt üle püsiva WebSocket ühenduse (tuntud ka kui OCPP-J), kus päringud ja vastused liiguvad üksteisest sõltumatult ning on seotud sõnumiidentifikaatoriga (vt Joonis 6) [7].



Joonis 6. OCPP sünkroonne (vasakul) ja asünkroonne suhtlus (paremal)

Suhtlus kasutab kindlat sünkroonsuse põhimõtet. Laadija ja juhtsüsteem ei tohi teineteisele saata uut päringut enne kui eelmisele päringule pole saadud vastust või eelmisele päringule on oodatud vastust kauem, kui seadistatud maksimum [7]. Eelmainitu tõttu võib juhtsüsteemis tekkida olukord, kus laadijale proovitakse saata mitu sõnumit korraga, mille puhul peab enne järgmise sõnumi saatmist ootama eelmise vastust, ehk tekitama sõnumite järjekorra, nagu laadijas on vastupidise suhtluse teostamiseks vastavalt ka oma saatmise järjekord.

Suhtluse turvalisust aitab tagada autentimisvõtme olemasolu ja suhtluse toimumine üle TLS (ingl. k. *Transport Layer Security*) transpordikihi turbeprotokolli [7]. TLS'iga toimub andmeedastus krüpteeritult, mis takistab teistel osapooltel suhtluse pealtkuulamist [17]. Autoriseerimisvõti kirjutatakse juhtsüsteemi andmebaasi ja seadistatakse ka laadijasse. Laadija ühendumisel peab juhtsüsteem kontrollima päringu päises saadetud võtme vastavust. Võtme mittevastavuse korral paneb juhtsüsteem ühenduse kinni. Tänu sellele, et suhtlus on juba TLS'iga krüpteeritud ei näe võõrad osapooled suhtluse ajal võtit. Eelmainitud võtted kooskõlastavad süsteemi OCPP teise taseme turvaprofiiliga [7] [18]:

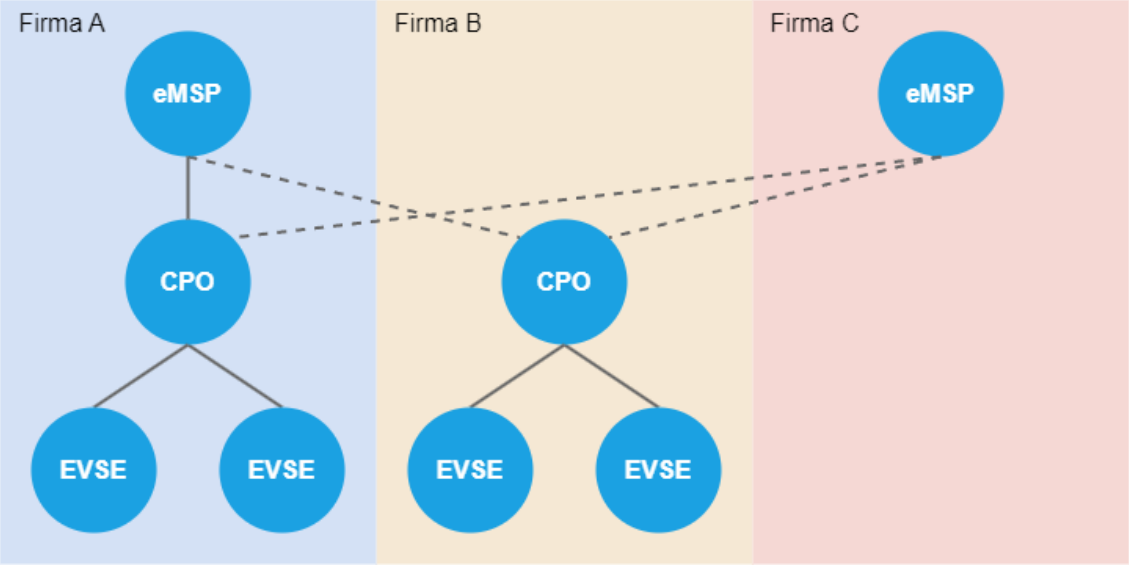
- juhtsüsteemi identiteedis veendumine (kasutades serveri sertifikaati);
- laadija identiteedis veendumine (kasutades autentimisvõtit);
- krüpteeritud suhtlus (kasutades TLS'i).

Kolmas turvaprofiil näeb ette laadijate identiteedikontrolliks kliendisertifikaatide kasutamist. Seda on praktikas raske korraldada, kuna kõik laadijad ei toeta kliendisertifikaadi lisamist ning juhtsüsteem peab ise ka võimaldama sertifikaadi kontrolli. Praktikas on teisest turvaprofiilist vähemalt süsteemi esialgses versioonis piisav, kuna nii teine kui ka kolmas profiil mõlemad alandavad aktsepteeritaval määral samu turvariske [18].

3.4 Rändluse korraldamine

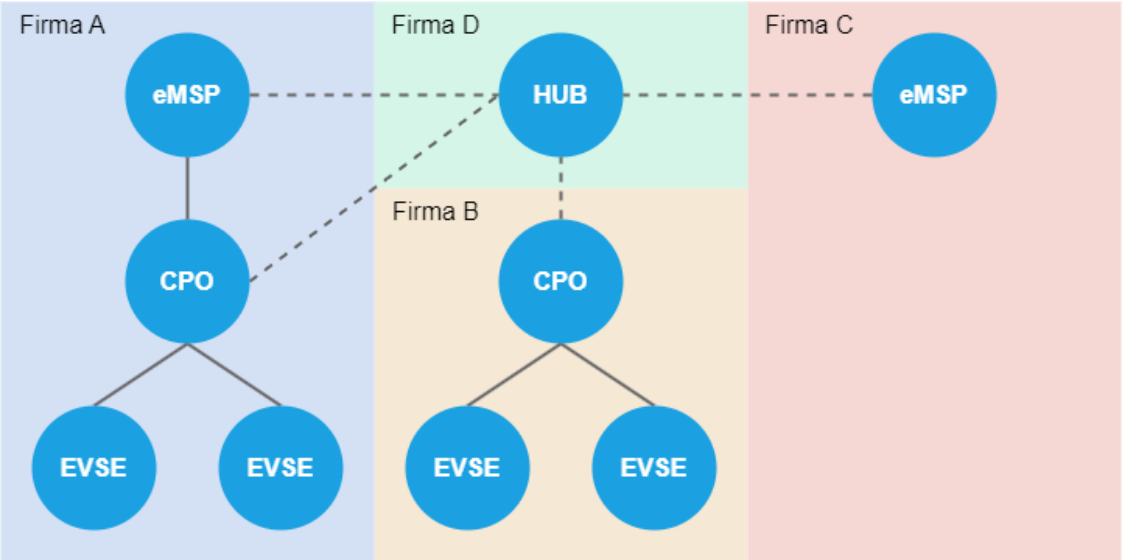
Rändlus võimaldab ühe ettevõtte kliendil laadida teise ettevõtte laadijas. Lisaks, võimaldab rändlus saata laadijate andmeid ühtses vormingus erinevatesse navigatsiooni- ja kaardisüsteemidesse, kus kasutajad saavad näha mitmete ettevõtete laadijaid ning rändluse kaudu neis laadida. Kuigi juhtsüsteemi esialgne versioon ei nõua rändluse väljaarendust, peab see võimaldama rändluse arendamist tulevikus. Selleks tasub uurida rändlusega seotud teooriat, et suunata süsteemi disain tulevaseks arenduseks sobivaks. Rändluse korraldamiseks on olemas mitmeid laialt levinud protokolle [6], kuid ettevõte, kelle jaoks käesolevat juhtsüsteemi luuakse, on oma nõuetest lähtuvalt valimas OCPI (ingl. k. *Open Charge Point Interface*) ja OICP (ingl. k. *Open InterCharge Protocol*) vahel.

OCPI on 2014. aastal Hollandi laadimisteenuse pakkujate koostöös loodud rändlusprotokoll. Tänapäeval on protokoll hallatud sihtasutuse EVRoaming poolt, mis on koostöös mitmete suurte Euroopa laadimisteenuse pakkujatega [19]. Protokoll toetab suhtluse korraldamist üle otseühenduse (ingl. k. *peer to peer*) (vt Joonis 7), läbi vahendaja (ingl. k. *hub*) või kasutades nende mõlema kombinatsiooni [13]. Otseühendus võimaldab osapooltel kasutada omavahel kokkulepitud protokollilaiendusi ja kindlat protokolli versiooni [13]. Protokollilaiendused on eriti kasulikud, kui lisaks väliste süsteemidega suhtlemiseks on sama protokoll kasutuses ka ühe ettevõtte süsteemide vahelises suhtluses. Lisaks, kuna sõnumid ei liigu alati läbi vahendaja, kui üks osapooltest on maas, saab suhtlus kõikide teiste osapooltega jätkuda.



Joonis 7. Rändlus korraldatud üle otseühenduse

OICP on 2012. aastal rändlusteenust pakkuva firma Hubject poolt loodud rändlusprotokoll [20]. Protokoll toetab suhtluse korraldamist ainult läbi vahendaja (vt Joonis 8), milleks on Hubject ise [20]. Hubject'i ülesehitus võtab enda rehkendada palju loogikat, mida peaks muidu iga osapool ise haldama. Suhtlemine läbi vahendaja lihtsustab ka paljude osapooltega liidestamist ning säästab arendusvaeva, kuna iga osapoole liidestusanomaaliate asemel peab tegelema ainult vahendajaga liidestamise omadega. Kui mõlemad süsteemid on vahendajaga ühendunud, saab peale osapooltevahelise lepingu sõlmimist rändlust korraldada lisaarenduseta.



Joonis 8. Rändlus korraldatud läbi vahendaja

Mõlemad protokollid tuvastavad osapooli unikaalse riigi- ja osapoolekoodi kombinatsiooniga, kasutavad JSON andmevormingut ja sertifikaadipõhist autentimist. Mõlema protokolli laadimissessiooniga seotud suhtlus sarnaneb ülesehituselt OCPP omaga. Esineb olukord, kus OCPP käsu peale peab saatma välja vastava käsu rändlusesse. Kõigi kolme protokolli andmeväljad erinevad piisavalt palju, et vajada andmeteisendust. Mõlemad rändlusprotokollid näevad ette, et osa andmeid (nagu andmed teiste osapoolte laadijate kohta) peab perioodiliselt lokaalsesse andmehoidlasse alla tõmbama, et rakendused saaksid nende kohta kiiremini infot pärida ning suure kasutajate arvuga rakendused teiste osapoolte servereid üle ei koormaks. Vastupidiselt peab ka teistele osapooltele oma andmeid raporteerima.

3.5 Juhtsüsteemi liidestused

Selgub, et juhtsüsteem peab suhtlema laadijaga, eMSP süsteemiga, võrguoperaatoriga ja rändlusvahendaja või -osapoolega. CPO vaatepunktist kuuluvad eMSP rolli kõik projektid, mida ettevõtte klient otse kasutab ehk kliendiportaal, makseportaal ja mobiilirakendused.

Kliendiportaalis saab laadija omanik hallata laadijaid, laadimisõigusi ning hindu ja jälgida laadijate olekut ning laadimisandmeid. Sellest esinevad juhtsüsteemile konkreetsemad nõuded ja täpsustused:

- juhtsüsteem peab otsuste tegemisel arvestama kliendiportaalis tehtud muudatusega nagu hinnad ja laadimisõigused:
    - ei tohi lasta laadida kasutajal, kellel pole selle jaoks õigust;
    - ei tohi lasta laadida muu hinnaga kui kliendiportaalis määratud.
- juhtsüsteem peab kliendiportaali käsu peale otse laadijale halduskäske saatma:
    - halduskäskude saatmine ei tohi mõjutada teisi süsteeme, mis laadijale käske saadavad (nagu mobiilirakendusest laadimise alustamine);
    - kui laadija pole juhtsüsteemiga parasjagu ühenduses, peab haldusoperatsiooni, kui võimalik, teistmoodi läbi viima ja/või olukorrast kasutajale teatama (nagu laadimissessiooni peatamine).
- laadija oleku muutusel peavad andmed kliendiportaali välja jõudma:
    - kui laadija ühendus katkeb või taastub, peab see info kajastuma kliendiportaalis;
    - kui laadija olek muutub peab see info kajastuma kliendiportaalis.
- laadimissessiooni andmed peavad kliendiportaali välja jõudma:
    - kui laadimine algab, lõppeb või kulgeb, peab see kajastuma kliendiportaalis.

Makseportaal ja mobiilirakendused on CPO vaatepunktist funktsionaalselt väga sarnased, kuna mõlemast saab alustada ja lõpetada laadimist ning mõlemasse peab reaalajas saatma sessiooni kohta infot. Vahe on selles, et makseportaalis saab laadida kasutajata. Nende rakenduste kõik muud, laadimissessioonidega otseselt mitte seotud, vajadused on täidetud eMSP serveri kaudu. Juhtsüsteem peab arvestama aga järgmisega:

- saadud päringu alusel peab juhtsüsteem alustama või lõpetama laadimissessiooni;
- laadimissessiooni andmed peavad jõudma kasutajaliidesesse reaalajas (näiteks laetud koguse näitamiseks);

- laadijate olekumuutus peab jõudma kasutajaliidesesse reaalajas (näiteks mobiilirakenduses laadijade kaardil näitamiseks).

Iga mingi aja tagant peab võrguoperaatorile saatma laetud koguse kohta raporteid. Sarnaseid osapooli, kus peab iga mingi aja tagant andmeid üleslaadima või alla tõmbama, võib tulevikus tekkida mitu. Näiteks rändluse puhul tekib juba vajadus regulaarselt alla tõmmata teiste osapoolte laadijaid. Siinjuhul peab olema ettevaatlik, kui eelmine andmevahetus on veel töötlemisel ei tohi saata või pärida samu andmeid uuesti.

Rändluse lisandumisel on vajadused sarnased võrguoperaatori ja mobiilirakenduste vajadustega. Lisanduvad ka teiste osapoolte andmed, mida peab lokaalselt salvestama.

3.6 Loodav süsteem

Kolmanda osapoole lahendusest oma lahendusele ülemineku perioodil sai kirjutatud juhtsüsteemi prototüüp Go programmeerimiskeeles. Aegamööda hakkasid selguma aga keele puudujäägid. Esiteks, tema minimaalne standardteek, mille tõttu pidi tihti kirjutama või otsima teeke andmestruktuuridele ja funktsionaalsusele, mis oleks võinud olla keelde sisseehitatud. Teiseks on Go'l hulga omapäraseid võtteid [21], mis raskendab arendust ja arendajate liikumist projektide vahel. Keeruka rakenduse arendamisel osutusid Go puudujäägid olema suurema kaaluga kui Go näiline lihtsus, elegantsus ja kiirus.

Ettevõtte teised projektid olid juba majutatud Microsoft Azure pilveteenuses ja kirjutatud C# programmeerimiskeeles. Kuna nii Azure kui ka C# on mõlemad Microsoft'i tooted on nendevaheline liidestus võrreldes Go'ga tipptasemel. Näiteks on vajaduse korral võimalik kasutada ametlikke teeke, et liidestada enda C# põhine lahendus mõne Azure'i logimis-, analüütika- või andmehoiuteenusega [22]. Võrreldes Go'ga on C#'il väga korralik ja laiaulatuslik standardteek ning andmetöötlusfunktsioonid. C# põhine projekt annab võimaluse kasutada jagatud andmetüüpide definitsioone kõikide olemasolevate C# projektide vahel, elimineerides ümberkirjutamise ja keelte eripäradest tuleva vaeva. Lisaks on C#'il olemas ka ametlik ja laiaulatuslik veebiteenuste loomiseks mõeldud raamistik ASP.NET Core, mis lihtsustab konfigureerimise, andmete valideerimise, rakendusliidese struktureerimise ja reaalajas suhtluse korraldamise [23].

Ettevõttel oli juba enne kasutusel MongoDB andmebaasisüsteem, täpsemalt, Azure Cosmos DB dokumendipõhine andmebaasisüsteem MongoDB päringuliidesega.

Dokumendipõhistel andmebaasidel on head omadused kiiresti muutuvatele ja pidevas arengus olevatele süsteemidele. Range struktuuri kirjelduse puudumine võimaldab dokumentide andmevälju kergelt lisada, muuta ja kustutada [24]. Lisaks võib seotud andmeid dokumentide sisse otse manustada [24], mis on kasulik omaduste või väärtuste nimekirjade puhul, näiteks asukoha lisaväärtusteenuste nimekirja või laadija EVSE'de ja pistikute omaduste kirjelduse puhul. C#'i jaoks on olemas ka teek MongoDB liidesega suhtlemiseks, mis võimaldab kirjutada nii tooreid kui ka tugevalt tüübitud (ingl. k. *strongly typed*) päringuid [25].

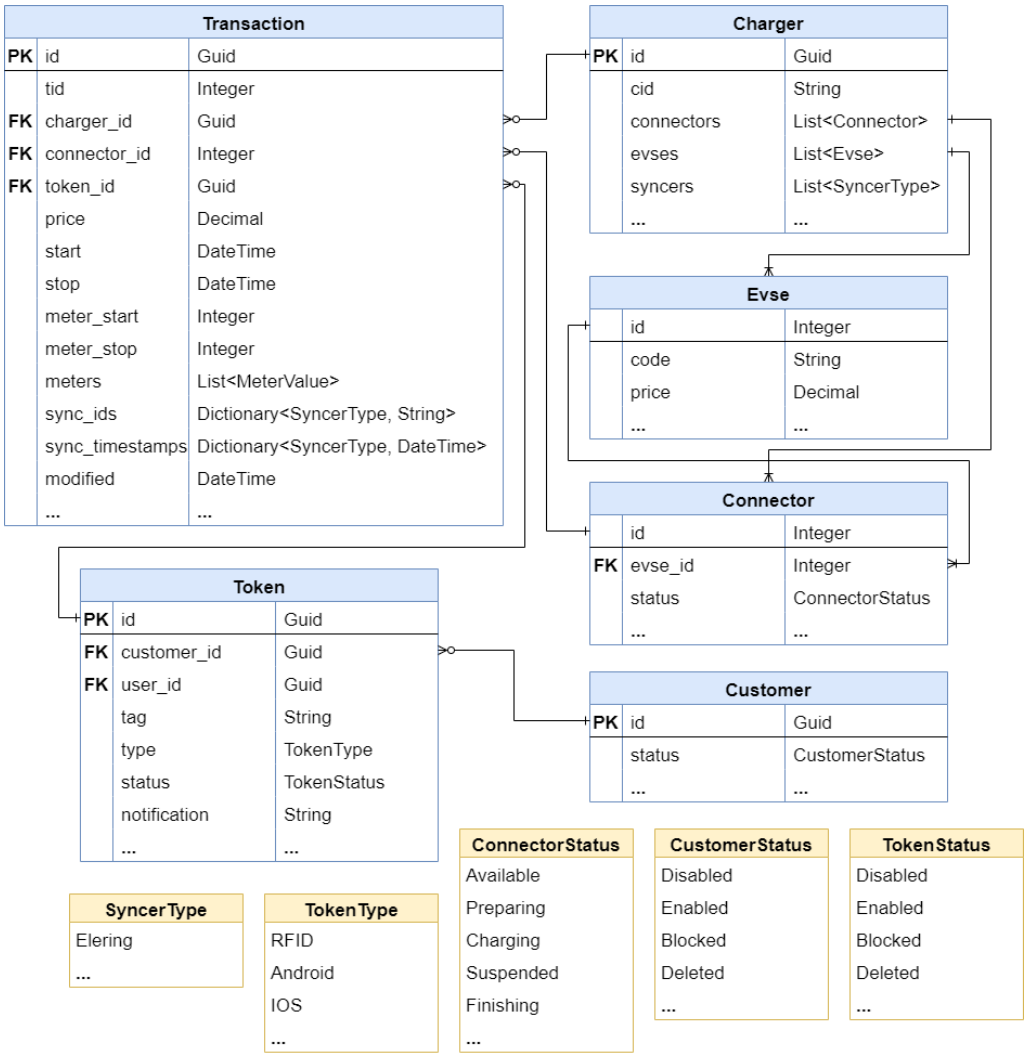
Hea ühilduvuse tõttu ettevõtte olemasolevate projektidega sai juhtsüsteem kirjutatud ka C# programmeerimiskeeles. Kuna ettevõtte andmebaas oli juba MongoDB põhine tuli juhtsüsteem liidestada ka olemasoleva andmebaasiga.

4 Süsteemi disain

Käesolevas peatükis selgitatakse välja süsteemi disain, toetudes teoreetilistest alustest selgunud asjaoludele.

4.1 Andmemudel

Andmemudelist (vt Joonis 9) on välja jäetud arvelduse ja haldusega seotud olemid, millest juhtsüsteem otsest huvi ei tunne. Kujutatud on peamised olemid, mis võimaldavad juhtsüsteemil laadimist korraldada.



Joonis 9. Andmemudeli ülevaade juhtsüsteemi suhtes

Andmemudeli üheks põhiolemiks on laadija, mis vastab füüsilisele laadijale. Laadijasse seadistatakse tema identifikaator, mis laadija ühendamisel viiakse kokku laadijaolemis oleva *cid* välja väärtusega. Kuna EVSE'd ja pistikud on laadija lahutamatud osad ehk alati seotud konkreetse füüsilise laadijaga ning laadijate vahel liigutamatud, on vastavad olemid manustatud otse laadija olemi sisse. Kuna laadija nummerdab oma pistikud ise, on olemi primaarvõtmeks seatud numbriline väärtus, mis täitub laadija saadetud info järgi. Edaspidi hakkab laadija saatma oma pistikute oleku kohta teateid, mille väärtus salvestatakse vastava pistiku manustatud olemisse. Kuna laadijad ise EVSE'id ei raporteeri, saab laadija omanik need manuaalselt luua, määrata neile pistikud ja hinna.

Kasutajatele väljastatakse turvamärgid, mis annavad oma oleku põhjal laadimisõiguse. Turvamärk võib olla füüsiline RFID kaart või ka mobiilirakendus. Nende vahel eristamiseks on turvamärgil defineeritud tüüp. Aktiivse laadimissessiooni kohta teate saatmiseks mobiilirakendusse on mobiilirakendustest pärit turvamärkidel ka teatemärk ehk identifikaator, mille järgi võib konkreetsele telefonile infot saata. Kuna kasutaja igal seadmel on erinev turvamärk ja iga turvamärgi teatemärk viitab tagasi konkreetsele seadmele on info saatmine laadimist alustanud seadmele lihtsalt korraldatav. Info saatmine ühe kasutaja kõikidele seadmetele on võimalik kõikide kasutajale kuuluvate aktiivsete turvamärkide kokku kogumisega ja sealt teatemärkide välja lugemisega.

Kui laadija alustab sessiooni, salvestatakse see kohe andmebaasi. Edaspidi kui laadija saadab sessiooni kohta uut infot, uuendatakse olemasolevat sessiooni andmebaasis. Kui laadija lõpetab sessiooni, lisatakse andmebaasis olevale sessioonile lõpuaeg ja laetud kogus. Kuna laadijad toetavad sessioonide identifitseerimist ainult 32-bitise arvu järgi, on igal sessioonil lisaks süsteemsele unikaalsele identifikaatorile ka laadijaga seotud identifikaator. Eelmainitud identifikaator pole unikaalne, kuid ei tohi ühe laadija aktiivsete sessioonide piires korduda. Ühes laadijas ei tohi olla mitu aktiivset sessiooni sama identifikaatoriga, vastasel juhul ei teaks laadija ega süsteem, millisele sessioonile nende vahel liikuvad andmed kuuluvad.

Sessiooni olemis on suur hulk välju, mida andmemudeli joonises konkreetselt välja pole toodud. Need väljad on enamasti teistest olemitest pärit väljade koopiad, mis on mõeldud laadimishetkel kehtivate väärtuse jäädvustamiseks. Eesmärk on ära hoida olukorda, kus näiteks laadija nime muutumisel, muutub see ka kasutaja kauges laadimisajaloos. See

tekitab kasutajale segadust, kuna ajal millal kasutaja laadis, oli see laadija teise nimega. Selle võttega säilitatakse sessioonide terviklikkus ja ajalooline kontrollitavus.

Laadija olemis defineeritud sünkerid ütlevad, mis osapooltele selles laadijas toimunud sessioonide infot saatma peab. Sessiooni olemitel on sünkimisväljad, mis jälgivad, kas sessiooni info on ettenähtud osapooltele juba saadetud või mitte (vt Tabel 2). Eelmainitud andmevälju saab kasutada ka teistel olemitel, näiteks kui tekib vajadus teise osapoolega sünkroniseerida sessiooniandmete asemel laadijate endi andmeid.

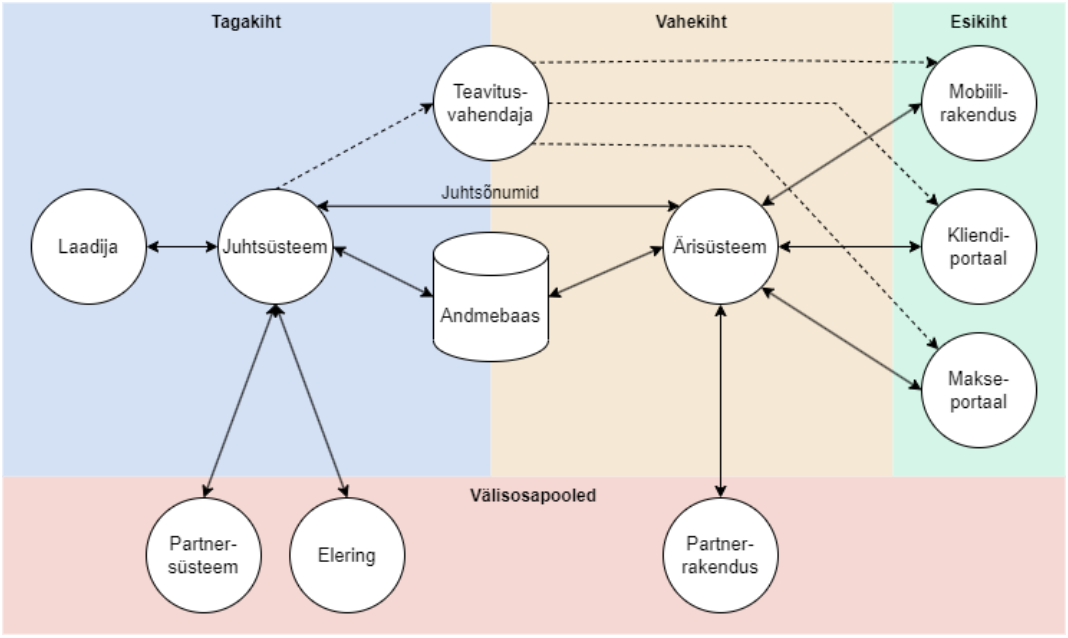
Tabel 2. Sünkimisväljade olekutabel

Väli ***syncers***	**Väli** ***sync_ids***	**Väli** ***sync_timestamps***	**Järeldus**
[ ]	{ }	{ }	Andmeid ei saadeta.
[Elering]	{ }	{ }	Andmed peab saatma.
[Elering]	{ Elering: 123 }	{ }	Andmed on saatmisel, seostatud väärtusega 123.
[Elering]	{ Elering: 123 }	{ Elering: 23-03-16T12:01:01 }	Andmed on saadetud, vajavad uuendamist kui on muutunud peale viimast ajatemplit.
[ ]	{ Elering: 123 }	{ Elering: 23-03-16T12:01:01 }	Andmed peab kustutama.
[ ]	{ Elering: 123 }	{ }	Andmed on kustutamisel.
[ ]	{ }	{ }	Andmed on kustutatud, andmeid ei saadeta.

4.2 Suhtlusmudel

Mobiilirakendused, kliendiportaal ja makseportaal suhtlevad ärisüsteemiga, mis omakorda loeb ja uuendab andmeid juht- ja ärisüsteemile ühises andmebaasis. Juhtsüsteemi ja ärisüsteemi vahel toimub enamus andmevahetust läbi eelmainitud ühise andmebaasi (vt Joonis 10). Ärisüsteem suhtleb juhtsüsteemiga otse ainult juhul, kui ärisüsteemi kasutaval eesrakendustel on vaja otse laadijale sõnumeid saata, näiteks uue laadimissessiooni alustamiseks või laadija kaughalduseks. Sellisel juhul vahendab ärisüsteem eesrakenduse ja juhtsüsteemi vahel päringuid. Päringute vahendamine võimaldab ärisüsteemil teha andmete ja kasutajaõiguste eelkontrolli ning vajadusel piirata

päringute arvu, et juhtsüsteemi mitte ülekoormata. Tagakihis asuvad teenused on kui ettevõtte sisevõrk, kuhu välisvõrgu rakendustest otse päringuid ei tehta. Lisas 4 on kujutatud osa mobiilirakendusest laadimise alustamisega kaasnev suhtlus.



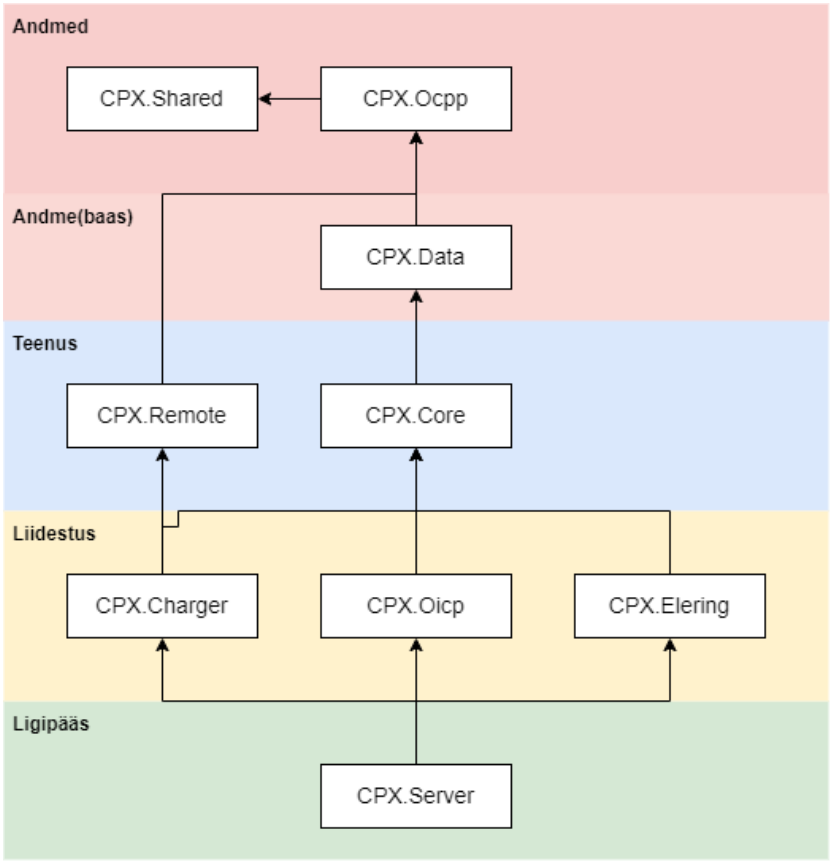
Joonis 10. Suhtlusmudeli ülevaade

Peale laadija oleku muutust või sessiooni algust, salvestab juhtsüsteem andmed süsteemide vahelisse andmebaasi, mille kaudu eesrakendused, sealhulgas kliendiportaal, saavad kätte uuenenud andmed. Sellisel moel uute andmete kättesaamine ei ole reaalaegne, kuna juhtsüsteem ei teavita uute andmete olemasolust, vaid iga rakendus peab ise andmebaasi pärima. Reaalajas uute andmete kättesaamiseks on kasutusel avaldaja-tellija suhtlusmudel, mis võimaldab juhtsüsteemil läbi vahendaja teavitada mitmeid huvitatud rakendusi, nendest rakendustest ise otseselt teadmata [26]. Sõnumivahendaja kasutamine ei tekita otsest sõltuvust juhtsüsteemi ja eesrakenduste vahel [26]. Reaalajas uuenduste jaoks eraldi sidekanali kasutamine aitab vähendada ka andmebaasi koormust. Avaldaja-tellija (ingl. k. *pub-sub*) mudeli kasutamine sobib siin olukorras paremini kui sõnumijärjekorra (ingl. k. *message queue*) mudeli kasutamine, kuna avaldaja-tellija näeb üldiselt ette, et iga kuulaja saab sõnumi kätte, kuid sõnumijärjekord näeb üldiselt ette, et ainult üks tellijatest saab sõnumi kätte [27].

Lõppenud sessioonide andmed saadetakse olenevalt vajadusele välisteenustele. Suhtlus välisteenustega toimub enamasti taustal, teistest programmivooludest sõltumatult. Kuid vajadusel võib suhtlus välisteenustega toimuda ka teistes programmivooludes.

5 Juhtsüsteemi arendus

Arendusprojekt koosneb mitmest omavahel seotud C# alamprojektist. Alamprojektid aitavad visualiseerida projekti struktuuri ja paremini hallata rakenduskihtide vahelisi seoseid (vt Joonis 11). Käesoleva juhtsüsteemi arhitektuur langeb kõige rohkem kokku laialtlevinud N-kihilise arhitektuuriga, mille kohaselt igal kihil on oma ülesanne ja ülemised kihid sõltuvad alumistest [28]. Teine laialtlevinud arhitektuur on „puhas arhitektuur“, mille kohaselt on arenduse tuumas defineeritud kõikide osade abstraktsioonid, mida tuuma ümber olevad kihid, üksteise detaile täpselt teadmata, teostavad ja omavahel kasutavad [28]. Kuigi puhas arhitektuur võimaldab suurtes projektides rakenduse osade väljavahetamist teisi osi puutumata, nõuab ta rohkem arendusaega ja kasvatab koodibaasi kiiremini, kui traditsiooniline N-kihiline mudel, mis siinjuhul tekkis projekti kallal töötamise jooksul loomulikult (vt Joonis 11).



Joonis 11. Juhtsüsteemi kihilise arhitektuuri ülevaade

Andmekihis on defineeritud valdkonda kirjeldavad andmemudelid ja andmebaasiga suhtlemiseks vajalikud hoidlad (ingl. k. *repository*) (vt Joonis 11). Hoidla on klass, mille ülesanne on suhelda andmebaasiga, teisendada baasi ja hoidla kasutaja vahel andmeid ning kujutada ennast ülejäänule rakendusele kui üldist andmekogu [29]. Eelmainitu aitab piirata koodi ülesandeid hallatavasse tükkidesse.

```
23 usages
public class ChargerRepository {

    private readonly IMongoCollection<Charger> _chargers;

    public ChargerRepository(IMongoClient client, IOptions<DatabaseConfig> databaseConfig) {
        var db = client.GetDatabase(databaseConfig.Value.Name);
        _chargers = db.GetCollection<Charger>(DatabaseConfig.ChargerCollection);
    }

    7 usages
    public async Task<Charger?> GetByEvseCodeAsync(string evseCode) {
        var cursor = await _chargers.FindAsync(
            Builders<Charger>.Filter.ElemMatch(c => c.Evses, e => e.Code == evseCode)
        );

        return await cursor.FirstOrDefaultAsync();
    }
```

Joonis 12. Laadija hoidla näide

Teenuskihis on defineeritud abstraktsed klassipõhjad, mida teenus- ja liidestuskihis kasutada saab. Näiteks, kui juhtsüsteemil endal või mõnel liidestusel on vaja passiivselt mingis intervallis andmeid vahetadata või aktiivselt reageerida laadija saadetud sõnumitele (vt Joonis 13), on olemas vastavad klassipõhjad, mis laiendamisel annavad teostuseks juhised kätte ja kindlustavad, et süsteemi osad töötavad ühtemoodi.

```
11 usages  7 inheritors
public abstract partial class AbstractLogger {

    1 usage  4 overrides
    public virtual Task LogStatusNotificationRequestAsync(LogContext<StatusNotificationRequest> ctx) {
        return Task.CompletedTask;
    }

    1 usage  4 overrides
    public virtual Task LogStartTransactionRequestAsync(LogContext<StartTransactionRequest> ctx) {
        return Task.CompletedTask;
    }

    1 usage  6 overrides
    public virtual Task LogStopTransactionRequestAsync(LogContext<StopTransactionRequest> ctx) {
        return Task.CompletedTask;
    }

    1 usage  4 overrides
    public virtual Task LogMeterValuesRequestAsync(LogContext<MeterValuesRequest> ctx) {
        return Task.CompletedTask;
    }

}
```

Joonis 13. Liidestustele mõeldud logija klassipõhja näide

Teenus- ja liidestuskihis on kõik vajalik, et juhtsüsteem tervikuna kui ka liidestused töötaks, sealhulgas klassipõhjade teostused ja muud teenusklassid, mis vastutavad äriloogika eest [30]. Suur osa teenusklasside tegevusest on seotud andmemudelitega. Tihti esineb vajadus tuletada andmemudeli andmete põhjal samu koondandmeid. Teenusklasside paisumise ja koodi kordumise vältimiseks on andmemudelitesse endisse viidud valdkonnaloogikaga seotud andmetöötlused, näiteks OCPP poolt defineeritud mõõtmistulemuste hulgast konkreetsete väärtuste pärimine (vt Joonis 14). Eelmainitu aitab ära hoida antimalli, mida Martin Fowler kirjeldas oma 2003. aasta artiklis „Anemic Domain Model“, kus andmemudelites on ainult andmeväljad ja teenusklassid on täis liigset tegevust [31].

```
5 usages
public class MeterValue {

    4 usages
    public DateTime Timestamp { get; set; }

    7 usages
    public List<SampledValue> SampledValues { get; set; } = new();

    5 usages
    public uint? EnergyImport {
        get {
            var sampledValue = SampledValues
                .FirstOrDefault(sv => sv is {
                    Measurand: Measurand.EnergyActiveImportRegister,
                    Phase: Phase.None
                });

            if (sampledValue == null) return null;

            var value = decimal.Parse(sampledValue.Value);
            if (sampledValue.Unit == UnitOfMeasure.KWh) value *= 1000;
            return (uint) value;
        }
    }
```

Joonis 14. OCPP mõõtmistulemuse andmemudeli klassi näide

Üheks liidestuseks liidestuskihis on laadija ise. Laadijaga suhtlemiseks on loodud eraldi sõnumitöötleja klassid, kus igale laadija sõnumitüübile on defineeritud erinev käitumine. Näiteks autoriseerimissõnumi puhul viiakse läbi vajalikud õigusekontrollid, mille põhjal saadetakse laadijale vastus. Üks sõnumitöötleja vastutus on kutsuda välja teenuskihi definitsiooni alusel liidestuskihis teostatud klasse. Näiteks olekusõnumi puhul kutsutakse logiteenuse abil välja kõik registreeritud logijad (vt Joonis 15). Eelmainitud logijate seas on andmebaasilogija, mis salvestab uuenenud oleku andmebaasi ja teatelogija, mis avaldab uuenenud oleku rakendusliidestesse. Mitmete logijaklasside kasutamine aitab hoida konkreetse ülesandega seotud koodi koos ning kuna logiteenus kutsub logijaid välja ainult abstraktsiooni põhjal ei leki liidestuste detailid rakenduste teistesse osadesse.

```
3 usages
public partial class Handler {

    1 usage
    private async Task HandleStatusNotificationRequestAsync(Message msg) {
        var req = msg.ParseRequest<StatusNotificationRequest>();

        await _loggerService.LogStatusNotificationRequestAsync(new LogContext(req/*, ... */));

        var res = new StatusNotificationResponse();
        await ReplyAsync(res);
    }
```

Joonis 15. Laadija sõnumitöötleja klassi näide

Ligipääsukihis elavad rakendusliidest defineerivad klassid, mis võtavad vastu päringu, töötlevad seda kasutades eelmistes lõikudes mainitud klasse ja saadavad välja vastuse (vt Joonis 16). Ligipääsukihti läbib kõik suhtlus, mis pärineb väljastpoolt juhtsüsteemi, sealhulgas laadijate ühendused ja ka ärisüsteemi päringud. Ligipääsukihis püütakse kinni ka erandid (ingl. k. *exception*) ja vastendatakse kinnipüütud veateatele rakendusliidesele asjakohane vastuse sisu.

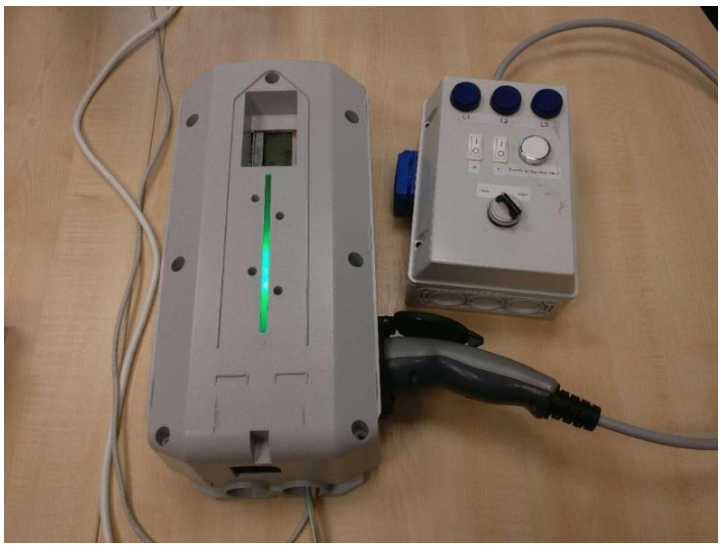
```
[ApiController]
[Route( template: "[controller]")]
public class RemoteController : ControllerBase {

    private readonly ChargerService _chargerService;
    private readonly RemoteConfig _remoteConfig;

    public RemoteController(ChargerService chargerService, IOptions<RemoteConfig> remoteConfig) {
        _chargerService = chargerService;
        _remoteConfig = remoteConfig.Value;
    }

    [HttpPost( template: "start/{evseId}")]
    public async Task<IActionResult> StartAsync([FromRoute] string evseId, [FromBody] StartRemoteRequest req) {
        try {
            // ...
            return Ok(/* ... */);
        }
        catch (ChargerService.NotFoundException) {
            return NotFound();
        } // ...
    }
```

Joonis 16. Rakendusliidese klassi näide

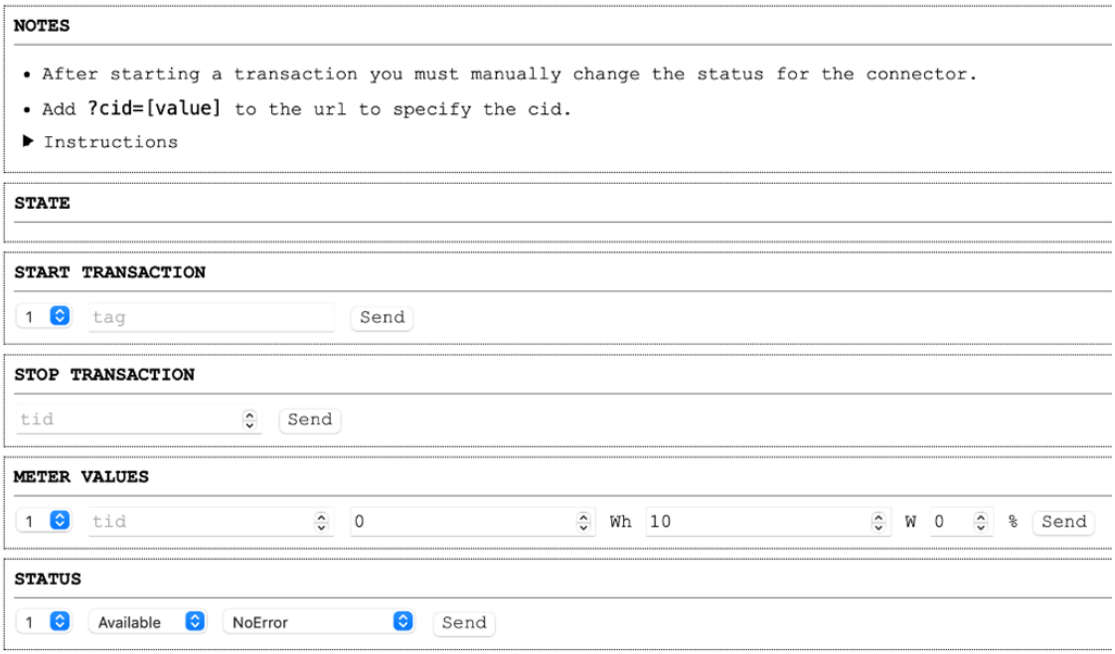
5.1 Testimine

Testialuseks olid laadija ja juhtsüsteemi vahelised protsessid. Testimine toimus nii riistvaralise (vt Joonis 18) kui ka spetsiaalselt käesoleva juhtsüsteemi testimise jaoks

arendatud tarkvaralise laadija vastu (vt Joonis 17). Tarkvaraline laadija sobis äriloogika testimiseks, näiteks testimiseks, kas etteantud kasutajal on laadimisõigus või mitte. Riistvaraline laadija sobis juhtsüsteemi ja laadija vahelise suhtlusprotokolli testimiseks, näiteks testimiseks, mis vormingus ja milliseid mõõtmistulemusi konkreetne laadija toetab ning kas ja kuidas võib konkreetne laadijamudel katki minna. Testima pidi ka ärisüsteemile mõeldud rakendusliidest. Rakendusliidese testimine toimus peamiselt Insomnia rakendusliidese testimisrakendusega ja lõpuks ka ärisüsteemi endaga.



Joonis 17. Füüsiline laadija ja võltsauto



Joonis 18. Tarkvaralise laadija kasutajaliides

Füüsilisi laadijaid testiti spetsiaalselt loodud testiprotokolli järgi (vt Joonis 19). Testiprotokoll aitas välja selgitada, et juhtsüsteemi ja konkreetse laadijamudeli vaheline suhtlus toimub veatult.

Nr	Action	Result	Done	Note
1	Point charger to "[redacted]" and reset	Connects to CPX, updates last seen and last boot in CPM	☑	
2	Show invalid card	Nothing happens	☑	
3	Show valid card	Starts charging	☑	
4	Check charger in app	Status updated due to push	☑	
5	Check charger in CPM	Connector status reflects actual status	☑	
6	Check transaction in CPM	Transaction exists with correct price	☑	
7	Show invalid card	Keeps charging	☑	
8	Show valid card	Stops charging	☑	
9	Check charger in CPM	Connector status reflects actual status	☑	
10	Check transaction in CPM	Transaction exists with correct meter and stop info	☑	
11	Remote start with invalid tag	Nothing happens	☑	
12	Remote start with valid tag from app	Starts charging, updates due to push	☑	
13	Remote stop with invalid tid	Nothing happens	☑	
14	Remote stop with valid tid from app	Stops charging, updates due to push	☑	
15	Check transaction in CPM	Transaction exists with correct price	☑	
16	Change availability to unavailable	Availability changes	☑	
17	Change availability to available	Availability changes back	☑	
18	Point charger back to live and save without reset	Nothing happens	☑	
19	Remote reset	Charger resets	☑	

Joonis 19. Juhtsüsteemi ja laadija testiprotokoll

Testimise käigus pärines enamus vigu laadijate käitumiserinevustest, mistõttu projekti valmides tuli testida kogu funktsionaalsus iga laadija mudeliga eraldi. Edaspidi uute mudelite lisandumisel, tuleb ka läbi viia needsamad testid, et olla kindel, et laadija ja juhtsüsteem omavahel kokku sobivad. Testimisperioodi lõpuks said kõik leitud vead parandatud ning ei esinenud vigu, mis oleksid nõudnud suuremahulist ümberarendust.

5.2 Tulemus

Kõik peatükk 2.2. püstitatud põhinõuded said arenduse lõpuks rahuldatud. Valminud juhtsüsteem võimaldab:

- laadijatel enda külge ühenduda ja endaga suhelda;
- laadimist alustada ja lõpetada nii RFID kaardiga kui ka välisest rakendusest;

- salvestada laadimisandmeid andmebaasi;
- saata laadimisandmeid välisteenustesse ja kasutajarakendustesse passiivselt või aktiivselt;
- arendada rändluse tugi juba teostatud programmiosadele toetudes.

Kasutaja saab alustada laadimist mobiilirakendusest enne, kui kaabli ühendamiseks autost väljub, olles eemal saab kasutaja jälgida mobiilist laadimise infot nagu aku protsent ja hetke laadimisvõimsus, mille põhjal saab kasutaja otsustada, millal on aeg autole järgi minna. Hiljem saab kasutaja mobiilirakendusest vaadata laadimissessioonide ajalugu. Eelmainitud funktsionaalsuse toimimist võimaldab valminud juhtsüsteem.

Üks süsteem, millega valminud juhtsüsteem integreerub on makseportaal, mis sai loodud ühe eelneva bakalaureuse lõputöö raames eesmärgiga pakkuda klientidele võimalus laadida Eleport'i laadijates, platvormil püsikasutajat omamata [32]. Juhtsüsteemi vaatest on makseportaal üks kasutajarakendus, kust saab laadimist alustada ja lõpetada ning kuhu laadimisandmed peavad välja jõudma.

Valminud juhtsüsteem teenindab praegusel ajal pidevalt üle 200 laadija nii Eestis kui ka Lätis ja varsti ka Leedus. Poolas asuvad laadijad lisatakse süsteemi aastal 2025. Laadimisjaamade arv, mis ühendatakse süsteemi, kasvab kolme aasta jooksul umbes 4500 seadmeni. Tänaseks on süsteem olnud tootmiskeskkonnas kasutusel alates 2022. aasta juunist ning selle aja jooksul on Eleport'i laadijates tehtud üle veerand miljoni euro suurune käive. Laetud umbes üks miljon kWh elektrienergiat autodesse, mis on üle viie miljoni kilomeetri sõite, mis on märkimisväärselt vähendanud CO2 heitmeid transpordi sektoris.

5.3 Edasiarenduse võimalused

Nagu põhinõuetes mainitud, pidi süsteem võimaldama rändluse juurdearendust. Üks edasiarenduse võimalus ongi OICP ja/või OCPI protokolli toe lisamine, et pakkuda veelgi laiaulatuslikumaid laadimisvõimalusi. Lisaks on tulevikus oodata ka laadijaid, mis toetavad uut OCPP-2.0 protokolli. Uute laadijatega suhtlemiseks tuleb juhtsüsteemil toetada ka protokolli uut versiooni.

Uuemad elektriautod ja laadijad toetavad *Vehicle to Grid* ja *Plug & Charge* funktsionaalsust. *Vehicle to Grid* näeb ette, et elektriauto annab läbi laadija elektrit võrku tagasi [33]. *Plug & Charge* näeb ette, et auto suudab ise juhtsüsteemile ennast tuvastavat infot saata, mis tähendab, et laadimine saab alata kohe ning kasutaja ei pea näitama RFID kaarti ega mobiilirakendusest ise laadimist alustama [34]. Kuigi eelmainitud süsteemid arvestavad OCPP protokolliga võivad nad vajada süsteemis täpsustusi.

6 Kokkuvõte

Käesoleva töö eesmärgiks oli luua juhtsüsteem, mis võimaldab Eleport'i klientidel oma elektriautot laadida ning Eleport'il laadimisteenust pakkuda, kasutamata kolmanda osapoole tähistamata lahendust. Tähistamata teenuse kasutamine ei olnud jätkusuutlik, kuna arendus oli aeglane ja laadijavõrgu laienedes kasvas vahendustasu märgatavalt.

Juhtsüsteemile seatud nõuded said täidetud ja süsteem vastuvõetavas mahus testitud. Juhtsüsteem on juurutatud tootmiskeskkonda ja laadijaid teenindamas. Muu hulgas integreerub valminud juhtsüsteem ühe eelneva bakalaureuse lõputöö raames valminud makseportaaliga.

Töö tulemusena valmis lisaks juhtsüsteemile ka tarkvaraline laadija juhtsüsteemi edaspidiseks testimiseks ning testiprotokoll juhtsüsteemi ja uute füüsiliste laadijate edaspidiseks ühilduvuse testimiseks. Lisaks täienesid ka autori ja ettevõtte teadmised elektriautolaadijatest ja laadimisega seotud protsessidest.

Kuna ettevõttel on soov jätkata lõputöö skoobist välja jäänud rändluse arendusega, jätkab käesoleva töö autor mainitud lisafunktsionaalsuse arendamisega.

Kasutatud kirjandus

[1] H. Vallaste, „e-Teatmik: IT ja sidetehnika seletav sõnaraamat,“ [Võrgumaterjal]. Available: http://vallaste.ee. [Kasutatud 02 02 2023].

[2] ElaadNL, „EV Related Protocol Study V1.1,“ Arnhem, 2017.

[3] Netherlands Enterprise Agency, „Electric Vehicle Charging - Definitions and Explanation,“ 01 2019. [Võrgumaterjal]. Available: https://www.rvo.nl/sites/default/files/2019/01/Electric%20Vehicle%20Charging%20-%20Definitions%20and%20Explanation%20-%20january%202019_0.pdf. [Kasutatud 04 03 2023].

[4] HM Government, „Taking Charge: The Electric Vehicle Infrastructure Strategy,“ 2022. [Võrgumaterjal]. Available: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/1065576/taking-charge-the-electric-vehicle-infrastructure-strategy.pdf. [Kasutatud 04 03 2023].

[5] Cybernetica, „Andmekaitse ja infoturbe leksikon,“ [Võrgumaterjal]. Available: https://akit.cyber.ee/. [Kasutatud 04 03 2023].

[6] M. van der Kam and R. Bekkers, "Comparative Analysis of Standardized Protocols for EV Roaming," Eindhoven, 2020.

[7] Open Charge Alliance, „Open Charge Point Protocol 1.6,“ 2015.

[8] Kaardiekspert OÜ, „RFID tooted,“ [Võrgumaterjal]. Available: https://kaardiekspert.ee/products/rfid-kaardid/. [Kasutatud 24 02 2023].

[9] Euroopa Parlament, „"Eesmärk 55": uued sõiduautod ja kaubikud 2035. aastaks heitevabaks,“ 14 02 2023. [Võrgumaterjal]. Available: https://europarl.europa.eu/news/et/press-room/20230210IPR74715/eesmark-55-uued-soiduautod-ja-kaubikud-2035-aastaks-heitevabaks. [Kasutatud 23 02 2023].

[10] Transpordiamet, „Sõidukite statistika,“ 01 02 2023. [Võrgumaterjal]. Available: https://www.transpordiamet.ee/soidukite-statistika. [Kasutatud 23 02 2023].

[11] Technology Connections, „Electric Car Chargers Aren't Chargers at All,“ 23 09 2020. [Võrgumaterjal]. Available: https://youtu.be/RMxB7zA-e4Y. [Kasutatud 30 03 2023].

[12] U.S. Department of Transportation, „Electric Vehicle Charging Speeds,“ 02 02 2022. [Võrgumaterjal]. Available: https://www.transportation.gov/rural/ev/toolkit/ev-basics/charging-speeds. [Kasutatud 03 03 2023].

[13] EVRoaming Foundation, „OCPI 2.2.1: Open Charge Point Interface,“ 06 10 2021. [Võrgumaterjal]. Available: https://evroaming.org/app/uploads/2021/11/OCPI-2.2.1.pdf. [Kasutatud 03 02 2023].

[14] Fraunhofer-Institute for Energy Economics and Energy System Technology IEE, Kassel, „Smart Charging Strategies and Technologies for Electric Vehicles,“ 11

2021. [Võrgumaterjal]. Available: https://changing-transport.org/wp-content/uploads/2021_Smart_Charging_Strategies_and_technologies_for_Electric_Vehicles.pdf. [Kasutatud 03 03 2023].

[15] S. Tailor ja P. Tailor, „White labeling,“ [Võrgumaterjal]. Available: https://learnmarketing.net/whitelabeling.htm. [Kasutatud 09 03 2023].

[16] Agile Business Consortium, „Chapter 10: MoSCoW Prioritisation,“ 01 2014. [Võrgumaterjal]. Available: https://www.agilebusiness.org/dsdm-project-framework/moscow-prioririsation.html. [Kasutatud 04 03 2023].

[17] Cloudflare, „TLS Basics,“ [Võrgumaterjal]. Available: https://internetsociety.org/deploy360/tls/basics/. [Kasutatud 30 03 2023].

[18] T. Flatau, „Open Charge Point Protocol Security Explained,“ 17 10 2022. [Võrgumaterjal]. Available: https://wevo.energy/white-papers/open-charge-point-protocol-ocpp-security-explained/. [Kasutatud 09 03 2023].

[19] EVRoaming Foundation, „About Us,“ [Võrgumaterjal]. Available: https://evroaming.org/about-us. [Kasutatud 05 05 2023].

[20] Hubject, „hubject/oicp: Open Intercharge Protocol,“ [Võrgumaterjal]. Available: https://github.com/hubject/oicp. [Kasutatud 03 02 2023].

[21] S. Wallez, „Go: the Good, the Bad and the Ugly,“ 10 04 2018. [Võrgumaterjal]. Available: https://bluxte.net/musings/2018/04/10/go-good-bad-ugly/. [Kasutatud 02 03 2023].

[22] Microsoft, „Azure for .NET Developers,“ [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/azure/. [Kasutatud 18 04 2023].

[23] Microsoft, „ASP.NET Overview,“ [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/aspnet/overview. [Kasutatud 05 05 2023].

[24] MongoDB, Inc., „Data Modeling Introduction - MongoDB Manual,“ [Võrgumaterjal]. Available: https://mongodb.com/docs/manual/core/data-modeling-introduction. [Kasutatud 27 01 2023].

[25] MongoDB, Inc., „MongoDB C# Driver,“ [Võrgumaterjal]. Available: https://mongodb.com/docs/drivers/csharp/current/. [Kasutatud 10 03 2023].

[26] Microsoft, „Publisher-Subscriber Pattern,“ [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/azure/architecture/patterns/publisher-subscriber. [Kasutatud 17 03 2023].

[27] D. Xiang, „Software Messaging Patterns - Queues vs. Publish/Subscribe (Pub/Sub),“ 22 08 2021. [Võrgumaterjal]. Available: https://davidxiang.com/2021/08/22/software-messaging-queues-pub-sub/. [Kasutatud 17 03 2023].

[28] Microsoft, „Common Web Application Architectures,“ 01 11 2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures. [Kasutatud 02 03 2023].

[29] Microsoft, „Design the Infrastructure Persistence Layer,“ 22 02 2023. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design. [Kasutatud 23 03 2023].

[30] M. Fowler, „P of EAA: Service Layer,“ [Võrgumaterjal]. Available: https://www.martinfowler.com/eaaCatalog/serviceLayer.html. [Kasutatud 23 03 2023].

[31] M. Fowler, „Anemic Domain Model,“ 25 11 2003. [Võrgumaterjal]. Available: https://www.martinfowler.com/bliki/AnemicDomainModel.html. [Kasutatud 23 03 2023].

[32] M. Kivi, „Veebirakenduse loomine elektriautode laadimiseks külalisena Eleporti laadimispunktides,“ Tallinn, 2022.

[33] Virta, „Vehicle-to-Grid (V2G): Everything You Need to Know,“ [Võrgumaterjal]. Available: https://www.virta.global/vehicle-to-grid-v2g. [Kasutatud 24 03 2023].

[34] Driivz, „What is Electric Vehicle Plug & Charge?,“ 04 03 2022. [Võrgumaterjal]. Available: https://driivz.com/glossary/plug-and-charge/. [Kasutatud 24 03 2023].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

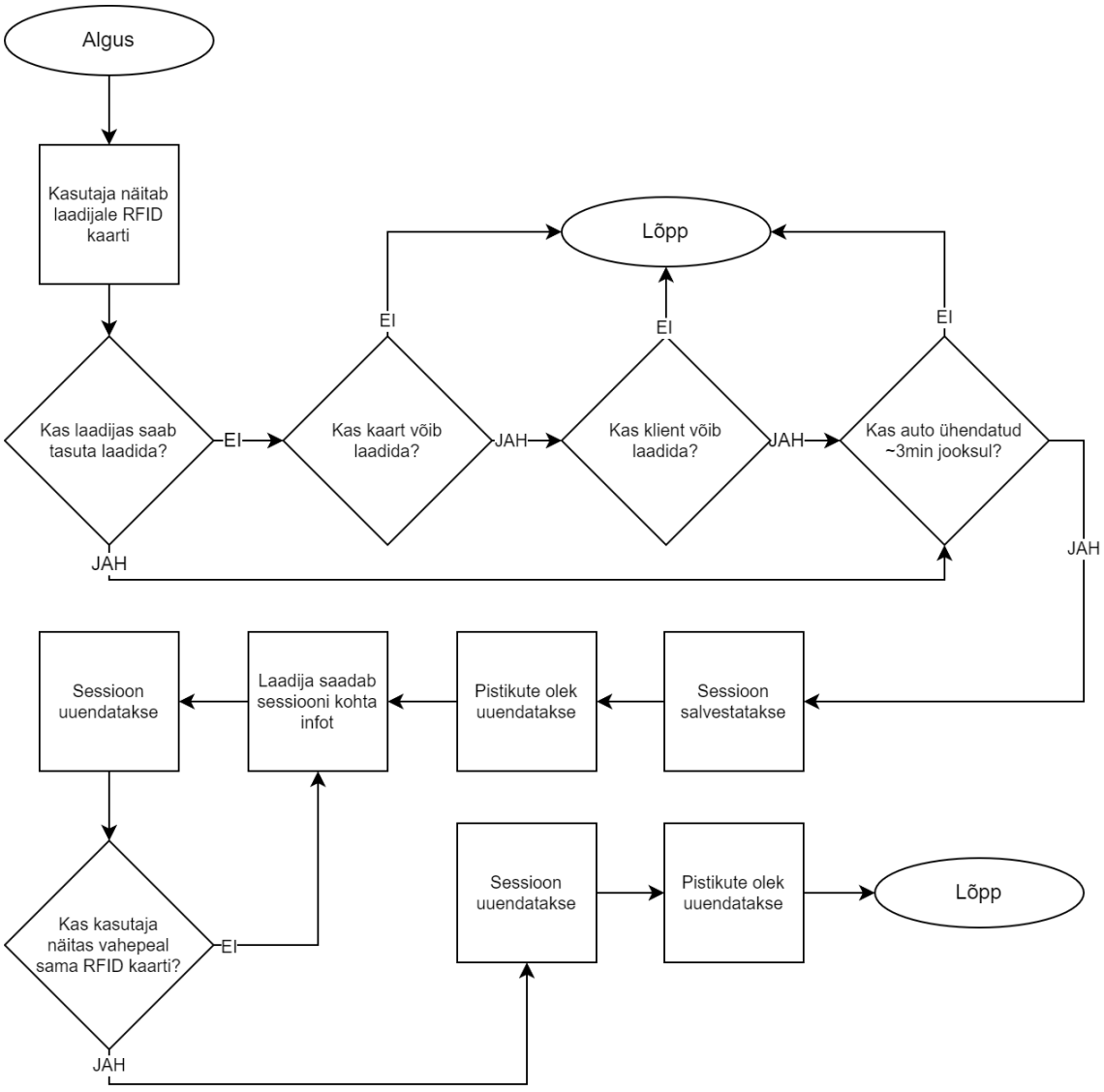
Mina, Karl-Kevin Köörna

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Elektriautode laadimise juhtsüsteemi arendamine“, mille juhendajad on Meelis Antoi ja Ivari Horm
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

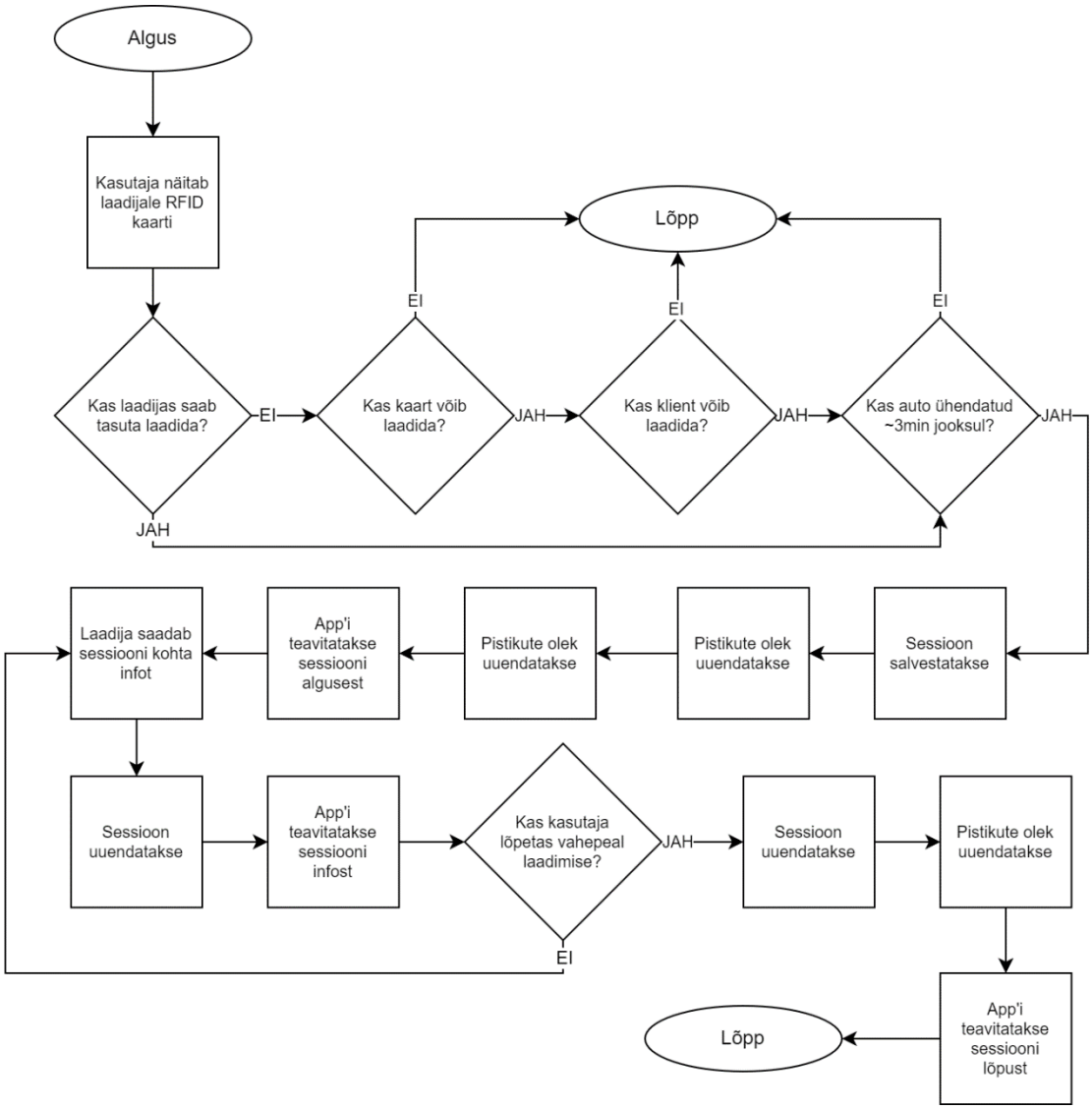
05.05.2023

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Laadimisprotsessi plokkskeem (kiipkaart)



Lisa 3 – Laadimisprotsessi plokkskeem (mobiilirakendus)



Lisa 4 – Mobiilirakendusest laadimise alustamisel toimuva suhtluse näide