TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Johan Vahter 179561

KLASSIFIKAATORITE JA GRUPPIDE RAKENDUS DOKUMENTIDE MENETLUSSÜSTEEMILE

Bakalaureusetöö

Juhendaja: Ahti Lohk
Doktorikraad

Tallinn 2020

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: Johan Vahter

18.05.2020

Annotatsioon

Antud bakalaureusetöö lahendab kahte suuremat probleemi. Esimeseks on probleem, et haldajad ei saa ise menetlussüsteemis UUSIS klassifikaatorite väärtusi hallata. Haldamine on raskendatud, kuna puudub ühtne kord ning tõlked ja väärtuste kehtivused on kirjutatud koodi. Teine probleem on mitmete rakenduste ühine vajadus ajakohaste klassifikaatorite järele, mis tuleneb UUSISe tükeldamisest väiksemateks mikroteenusteks ning erinevate dokumentide taotlemiste viimist e-taotluskeskkonda.

Bakalaureusetöö põhieesmärgiks on probleemidest tulenevalt valmis saada klassifikaatorite mikroteenus menetlussüsteemile UUSIS. Antud mikroteenus koosneb kahest osast, millest esimeseks on kasutajaliides, mis toetaks tõlkeid ja klassifikaatorite väärtuste kehtivusi ning teiseks osaks on rakendusliidesed teistele rakendustele ajakohaste klassifikaatori väärtuste küsimiseks.

Käesoleva bakalaureusetöö tulemusena valmis probleemide lahendamiseks kasutajaliides, mis koos andmebaasi tasandile viidud tõlgete ja kehtivustega võimaldab täielikku klassifikaatorite väärtuste haldust ja kaks rakendusliidest, millelt saab gruppide või klassifikaatorite ning kehtivuse kuupäevaga soovitud klassifikaatorite väärtusi küsida.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 29 leheküljel, 5 peatükki, 17 joonist, 1 tabelit.

Abstract

Classifiers and Groups Application for Documents Processing System

Given Bachelor's thesis solves two main problems. The first problem is that administrators cannot manage classifier values in documents processing system UUSIS on their own. Managing classifier values is difficult because there is no specific structure to manage those values, also the translations and the validities are written into the code of the program. Another problem that needs to be solved is that more and more applications need the same up to date classifier values. Demand for those up to date classifier values is increasing because UUSIS is being rewritten into smaller microservices and more document applications are made possible to apply in the e-application environment.

The main purpose of the Bachelor's thesis is to build a classifiers and groups application for document processing system UUSIS. The application consists of two main parts. The first part is a user interface that supports the translations and the validity of classifier values and the second part is an application programming interfaces from where other applications can query up to date classifier values.

As a result of this Bachelor's thesis a user interface and two application programming interfaces were created solving the above-mentioned problems. Classifier value translations and validity were designed into the database and that made it possible for the user interface to support every detail management of a classifier value. One of the application programming interfaces is for querying classifier values with the desired classifiers list and the other is to query similarly with groups list. Both interfaces also support an optional validity date, which when inserted into the request will only respond with classifier values valid on the inserted date.

The thesis is in Estonian and contains 29 pages of text, 5 chapters, 17 figures, 1 table.

Lühendite ja mõistete sõnastik

API	Rakendusliides
DAO	Andmete edastamise objekt
HTML	Veebilehtede märkimiskeel
ID	Tunnus
ISKE	Infosüsteemide turvameetmete süsteem
JSON	Javascripti objekti notatsioon
JSONB	Javascripti objekti binaarne notatsioon
UUSIS	Politsei- ja Piirivalveameti dokumentide menetlussüsteem

Sisukord

Jooniste loetelu

Tabelite loetelu

Sissejuhatus

Politsei- ja Piirivalveameti dokumentide menetlussüsteemil (UUSIS) eksisteerib palju erinevaid klassifikaatoreid, alustades riikidest ja lõpetades seadustega. Kõik need klassifikaatorid sisaldavad väärtuseid, mis omakorda asuvad gruppides. Maailm on pidevas muutumises: küll muutuvad riikide nimed ja küll kehtestatakse mõni uus seadus. Need on ainult üksikud näited paljudest. Kuna UUSISe puhul on tegemist riikliku menetlussüsteemiga, siis kõik eelpool nimetatu mõjutab pidevalt ka klassifikaatoreid, klassifikaatorite väärtusi ning gruppe, millesse need kuuluvad.

Klassifikaatorite väärtuste, gruppide ja tõlgete haldus ei ole keeruline, kuid antud menetlussüsteemis on see mitmete erinevate lahenduste ning lähenemistega keeruliseks aetud. Hetkel lisab, kustutab ja muudab antud väärtusi, kas arendaja, testija või tooteomanik, mis ei tohiks nii olla. Haldurid, kes reaalselt nende väärtuste eest vastutavad, ei saa neid ise hallata, kuna ei ole ühtset korda, kuidas need asjad üles on seatud. Klassifikaatorite väärtuste kehtivused ning ka mõned tõlked on kirjutatud otse koodi, mida haldurid ise muuta ei saa.

Selliselt on juba aastaid hakkama saadud, kuid see ei ole kindlasti parim praktika. UUSIS on hetkel üks suur monoliitrakendus, kuid seda tükeldatakse samm-sammult väiksemateks ja hallatavamateks mikroteenusteks. Kõikidel nendel mikroteenustel on üks ühine vajadus, milleks on klassifikaatorite kasutamine. Lisaks UUSISe tükeldamisele on aina enam võetud suund võimaldada taotleda erinevaid dokumente, lisaks ID-kaardile ja Eesti kodaniku passile, e-taotluskeskkonnast, et inimeste elu kergemaks teha. Praegu toimuvad arendused, et viia elamisloakaardi ja välismaalase passi taotlemise võimalus samuti e-taotluskeskkonda. Sealjuures tekkis taas tugev vajadus klassifikaatorite tõlgete järele, mida enam koodi kirjutada ei soovita.

Antud bakalaureusetöö põhieesmärgiks on valmis saada klassifikaatorite mikroteenus menetlussüsteemile UUSIS. Klassifikaatorite mikroteenus lahendab minimaalselt kaks suuremat eelpool mainitud probleemi. Esimese probleemi, kus haldurid ei saa klassifikaatorite või gruppide muudatusi ise sisse viia, lahendamiseks oleks tarvis teha

kaks suuremat sammu. Alustuseks peab kõik klassifikaatorite väärtuste tõlked, kehtivused ning gruppides eksisteerimise kehtivused viima andmebaasi tasandile. Esimese sammu edukas täideviimine tekitab võimaluse kõikide nende väärtuste halduseks kasutajaliideses, mis enne ei olnud võimalik. Teine samm selle probleemi lahendamiseks on kasutajaliidese ehitamine, mis peab võimaldama vaadata, muuta, kustutada ja lisada gruppe, klassifikaatoreid, keeli, klassifikaatorite väärtusi ning viimaste kehtivusi, tõlkeid ja gruppi kuuluvusi.

Teine probleem, mida klassifikaatorite mikroteenus lahendab, on mitmete rakenduste ühine vajadus ajakohaste klassifikaatorite järgi. Probleemi lahendamiseks on plaanis ehitada kaks eraldi rakendusliidest. Üks, millelt saab küsida klassifikaatorite võtmete järjendi järgi endale soovitud klassifikaatorite väärtusi ning teine rakendusliides, mis tagastab gruppide järjendi järgi klassifikaatorite väärtusi. Mõlemad rakendusliidesed peavad vajadusel arvestama ka klassifikaatori väärtuse kehtivuse kuupäevaga. Rakendusliidesed peavad olema dokumenteeritud rakendusliideste abisüsteemiga Swagger, et arendajatel oleks antud rakendusliidestest hea ülevaade.

Antud bakalaureusetöö koosneb kolmest suuremast osast, millest esimeseks on analüüs, mille põhjal rakendus arendada. Teiseks suureks osaks on rakendus ise koodi kujul, mis omakorda koosneb kahest suuremast osast: kasutajaliidesest ning taustprogrammist. Viimaseks osaks on rakenduse tutvustus, mis annab valminud rakendusest põhjaliku ülevaate.

1 Projektis kasutatavad tehnoloogiad ja programmid

Menetlussüsteem, millele antud bakalaureusetöö autor programmi esmajoones programmeerib, on UUSIS. UUSIS on Politsei- ja Piirivalveameti isikutuvastus- ja menetlusinfosüsteem. UUSISt on kasutatud alates aastast 2002 ning antud infosüsteem puudutab eranditult kõiki inimesi, kes Eestis ametlikult elada või töötada soovivad. Isegi kui kasutada võimalust näiteks ID-kaart või Eesti kodaniku pass e-taotluskeskkonnast taotleda jõuab kogu info UUSISesse ning edasine menetlemine käib selle infosüsteemi kaudu [1].

Tänapäeval on olemas väga palju erinevaid ja väga võimekaid programmeerimiskeeli. Igal programmeerimiskeelel on omad positiivsed ja negatiivsed omadused, kuid enamike maailma populaarsemate keeltega saab juba praktiliselt kõike teha. Antud töös on peamiselt taustprogrammi kirjutamiseks kasutatud Pythonit koos Flaski raamistikuga ning kasutajaliidese kirjutamiseks Typescripti koos Vue.js'i raamistikuga.

1.1 Python

Pythoni programmeerimiskeele idee ning kavand on loodud Guido van Rossumi poolt aastal 1991. Hetkel arendab keelt *Pyhton Software Foundation*. Tänaseks on Pythoni kogukond kasvanud väga suureks ning tegemist on ühe populaarsema keelega kogu maailmas. Viimastel aastatel on Pythoni populaarsus jõudsalt kasvanud tänu tehisintellektide võiduarendamisele [2].

Bakalaureusetöö autor valis Pythoni antud projekti taustprogrammi arendamiseks väga mitmel põhjusel. Esiteks ei ole Pythoni programmeerimiskeele võimekusel piire, sellega on võimalik kõike arendada. Teiseks leidub palju ülevaatlikku dokumentatsiooni iga detaili kohta, mis huvi võib pakkuda. Lisaks on Pythoni kogukond suur ja abivalmis ning mitte ükski küsimus ei jää vastuseta. Uuemad Pythoni versioonid toetavad ka muutujate ja funktsioonide tüüpimist, millega on võimalik koodi kvaliteeti ning vettpidavust oluliselt tõsta.

1.2 Flask

Flask on Pythoni mikroraamistik, mis sai alguse Armin Ronacheri aprillinaljast. Mõte lihtsast ilma piiranguteta raamistikust kogus väga palju populaarsust ning seejärel tehti see ka tõeks [3]. Tänaseks on Flask üks põhilisi Pythoni raamistikke, mida kasutatakse veebiarendusteks [4].

Antud töös kasutab autor Flaski, kuna seda on äärmiselt lihtne üles seada, raamistik on väga paindlik ning materjale, nippe ja ideid, kuidas midagi Flaskiga teha, on võimalik suurema vaevata Internetist leida.

1.3 Typescript

Typescript on Microsofti poolt täiendatud Javascript. Algselt skriptimiseks kirjutatud keelega Javascript on raske suuremaid rakendusi vettpidavalt kirjutada, kuna ilma tüüpimiseta võib rakendus kiirelt loetamatuks, arusaamatuks ning vigaseks muutuda. Eelpool mainitu ongi põhjuseks, miks Microsoft tüüpimise Javascripti tõi ning milleks Typescript loodi [5].

Antud projektis kasutab autor Typescripti just puhta ning kvaliteetse koodi kirjutamise põhimõtte pärast. Javascripti kirjutamine on lihtsam, kuid vigu tuleb oluliselt rohkem ette.

1.4 Vue.js

Vue.js'i loojaks on Even You, kes kirjutas seda Javascripti raamistikku algselt enda tarbeks. Vue.js on loodud veebi kasutajaliideste lihtsaks arendamiseks [6].

Autor valis antud töös Vue.js, kuna tegemist on lihtsasti mõistetava ja paindliku raamistikuga. Vue.js'il on olemas toetus ka enamikele veebirakenduseks tarvilikele tehnoloogiatele ning koostöös Typescriptiga saab sellega väga korralikke rakendusi arendada.

2 Rakenduse analüüs

Antud bakalaureusetöös kirjeldatud rakendus jaguneb suures pildis kaheks. Esimene suurem osa on taustprogramm, mille peamine ülesanne on päring vastu võtta, valideerida, vajadusel andmeid töödelda, andmebaasis muudatusi teha ning sealt andmeid küsida, mida päringu tegijale tagastada, kui kõik klapib. Teiseks suureks osaks on kasutajaliides, mille funktsiooniks on assisteerida haldurit võimalikult mugavalt klassifikaatorite ning gruppide halduses ning kliendile olemasolevate väärtuste kohta ülevaate andmine.

2.1 Rakendus minevikus, olevikus ja pärast antud projekti lõppu

Kasutades üheksa akna analüüsi, näeb antud töös kavandatava rakenduse minevikku, olevikku ning kuhu rakendus selle töö lõpuks jõuab [7].

Tabel 1. Üheksa akna analüüs

	Minevik	**Olevik**	**Tulevik**
Välised tegurid / kasutajad	Süsteemi haldur Arendaja Menetlussüsteem UUSIS (üks rakendus)	Süsteemi haldur Arendaja UUSIS (üks rakendus) E-taotluskeskkond	Süsteemi haldur Arendaja UUSIS (tükeldatult) E-taotluskeskkond (täiendatud)
Süsteem	Klassifikaatorid andmebaasis	Klassifikaatorid andmebaasis ning kehtivused ja tõlked koodis	Klassifikaatorite rakendus
Süsteemi alamkomponendid	Klassifikaatoreid ja gruppe haldab ainult arendaja otse andmebaasis, tõlkeid ja kehtivusi ei saa määrata	Kasutajaliides, mis tõlkeid ja kehtivusi ei toeta. Otsimisel osalise sõnaga peab kasutama % märki. Muudatusi teeb suures osas arendaja.	Kasutajaliides klassifikaatorite ja gruppide halduseks, tõlgete lisamiseks ning kehtivuste määramiseks. Rakendusliides arendajatele

Tabelis 1 toodud analüüs näitab, et mida aeg edasi, seda rohkem rakendusi hakkab samu klassifikaatoreid kasutama. Vältimaks koodikordust ja lugematul arvul erinevaid lahendusi klassifikaatorite haldamiseks, on mõistlik teha rakendus, kust kõik rakendused, kellel on tarvis neid väärtusi kasutada, saavad neid ühest kohast küsida. Praegusel hetkel on võimalik e-taotluskeskkonnas taotleda Eesti kodaniku passi, ID-kaarti ning lühiajalise töötamise registreerimist. Arendamisel on ka tugi, mis võimaldaks e-taotluskeskkonnast taotleda elamisloakaarti ja välismaalase passi. See rakendus toetab ka mitut keelt. Mida aeg edasi, seda suurem nõudlus tekib tõlke funktsionaalsuse järele.

2.2 Rakenduse kasutajate kaardistamine

Iga rakenduse mõte on kellegi elu lihtsamaks, paremaks ja/või arusaadavamaks teha. Selleks, et rakendus võimalikult tõhusaks teha, on üks tähtsamaid punkte kogu rakenduse analüüsi juures aru saada, kes ja millist kasu rakendusest saab. Antud töös kirjeldatud rakenduse kasutajad jagunevad kolmeks, nagu näitas ka üheksa akna analüüs (vt Tabel 1 esimene rida).

2.2.1 Menetlussüsteemi haldurid

Kõige enam saavad antud rakendusest abi menetlussüsteemi haldurid ehk peakasutajad, kes loovad hetkel iga kord töö teostamiseks pileti, kui on vaja muuta mõnda klassifikaatori väärtust või lisada klassifikaatorit gruppi. Pileti tegemine võtab sama palju aega, kui selle muudatuse ise tegemine ning lisaks sellele peab haldur ka ootama, kuni keegi arendajatest tehtud pileti lõpuks realiseerib. Parema ülevaate andmine olemasolevates klassifikaatoritest on samuti üks rakenduse eesmärke.

2.2.2 Arendaja

Pärast antud bakalaureusetöö raames kirjutatud rakenduse valmimist ei pea arendajad enam oma väärtuslikku aega klassifikaatori väärtuste muutmisele kulutama, vaid saavad rahulikult arendustele keskenduda. Arendajad ise saavad kasutada antud töös tehtava rakenduse taustprogrammi API lõpp-punkte, et uutele rakendustele või vanade täiendamisele mugavalt soovitud klassifikaatoreid küsida. Korralik ülevaade, kuidas erinevaid klassifikaatoreid küsida, tuleb eraldi lehele Swaggeri tuge kasutades.

2.2.3 Teised rakendused

Menetlussüsteem on hetkel üks suur rakendus, kuid arendajad juba töötavad selle kallal, et see väiksemateks hallatavamateks osadeks muuta. Antud töös tehtav rakendus saab olema üks nendest nurgakividest, millele kõik need väiksemad rakendused just vajalike klassifikaatorite osas toetuma hakkavad.

2.3 Kasutajalugude kaardistamine

Kasutajalugu on lühikirjeldus sellest, kes, mida ja milleks teeb. Kasutajalugude kirjeldamine aitab kaardistada, milliseid funktsionaalsusi on antud rakendusel tarvis, et kasutajatele maksimaalselt kasulik olla. Kasutajalugude kirjutamine on agiilse arenduse üks põhilisi alustalasid [8].

Kasutajalood menetlussüsteemi haldaja vaatest tulenevalt süsteemi haldaja kasutajakogemusest:

1. Haldajana tahan ma näha olemasolevaid klassifikaatori väärtuseid, et saada ülevaade klassifikaatori väärtustest.

2. Haldajana tahan ma otsida klassifikaatori väärtust, et kiiremini soovitud klassifikaatori väärtus üles leida.

3. Haldajana tahan ma järjestada klassifikaatori väärtuseid, et saada parem ülevaade olemasolevatest klassifikaatoritest.

4. Haldajana tahan ma lisada uut klassifikaatori väärtust, et süsteemis oleksid ajakohased klassifikaatori väärtused.

5. Haldajana tahan ma muuta olemasolevat klassifikaatori väärtust, et süsteemis olevad klassifikaatori väärtused oleksid ajakohased.

6. Haldajana tahan ma olemasolevat klassifikaatori väärtust kustutada, et hoida süsteemis olevaid klassifikaatori väärtuseid ajakohasena.

7. Haldajana tahan ma näha, mis gruppides klassifikaatori väärtus asub, et saada parem ülevaade, millistes gruppides klassifikaatori väärtus asub.

8. Haldajana tahan ma lisada klassifikaatori väärtust gruppi, et hoida gruppide sisu ajakohane.

9. Haldajana tahan ma eemaldada klassifikaatori väärtust grupist, et hoida gruppide sisu ajakohane.

10. Haldajana tahan ma lisada tõlget klassifikaatori väärtusele, et toetada mitmekeelseid süsteeme.

11. Haldajana tahan ma lisada uut keelt, mille tõlget saab klassifikaatori väärtusele lisada, et toetada mitmekeelseid süsteeme.

12. Haldajana tahan ma näha, millal on klassifikaatori väärtus kehtiv, et saada parem ülevaade.

13. Haldajana tahan ma lisada klassifikaatori väärtuse kehtivuse vahemikku, et hoida süsteemis kuvatavad klassifikaatori väärtused ajakohased.

14. Haldajana tahan ma muuta klassifikaatori väärtuse kehtivuse vahemikku, et hoida süsteemis kuvatavad klassifikaatori väärtused ajakohased.

15. Haldajana tahan ma eemaldada klassifikaatori väärtuse kehtivuse vahemikku, et kehtetuid klassifikaatorite väärtusi ei pakutaks.

16. Haldajana tahan ma näha olemasolevaid gruppe, et saada ülevaade gruppidest.

17. Haldajana tahan ma otsida gruppi, et soovitud grupp kiiremini üles leida.

18. Haldajana tahan ma lisada uut gruppi, et süsteemis oleksid ajakohased grupid.

19. Haldajana tahan ma muuta olemasolevat gruppi, et süsteemis oleksid ajakohased grupid.

20. Haldajana tahan ma näha olemasolevaid klassifikaatoreid, et saada ülevaade klassifikaatoritest.

21. Haldajana tahan ma otsida klassifikaatorit, et soovitud klassifikaator kiiremini üles leida.

22. Haldajana tahan ma lisada uut klassifikaatorit, et süsteemis oleksid ajakohased klassifikaatorid.

23. Haldajana tahan ma muuta olemasolevat klassifikaatori kirjeldust, et süsteemis oleksid ajakohased klassifikaatorite kirjeldused.

24. Haldajana tahan ma kustutada olemasolevat klassifikaatorit, et süsteemis oleksid ajakohased klassifikaatorid.

25. Haldajana tahan ma näha kes, millal ja milliseid muudatusi klassifikaatori või grupiga tegi, et saada parem ülevaade.

Kasutajalood arendaja vaatest:

1. Arendaja tahan ma näha, millised API lõpp-punktid antud rakendusel on, et saada ülevaade, kuidas rakendust kasutada.

2. Arendajana tahan ma saada soovitud klassifikaatorid API-i lõpp-punktist päringuga, et saada arendatavale rakendusele vajalikud klassifikaatorid.

3. Arendajana tahan ma saada gruppides olevad klassifikaatorid API-i lõpp-punktist päringuga, et saada arendatavale rakendusele vajalikud klassifikaatorid.

4. Arendajana tahan ma saada kindlal kuupäeval kehtivad klassifikaatorid API lõpp-punktist päringuga, et saada arendatavale rakendusele ajakohased klassifikaatorid.

3 Rakenduse taustprogrammi planeerimine

Taustprogramm on rakenduse süda, mis peab teenindama selle külge ühenduvate rakenduste tööd. Antud töös on tarvis kahte taustsüsteemi osa programmi rakendusliidese osas, andmete töötluskihti, andmete andmebaasist küsimise kihti, autentimist ning andmebaasi.

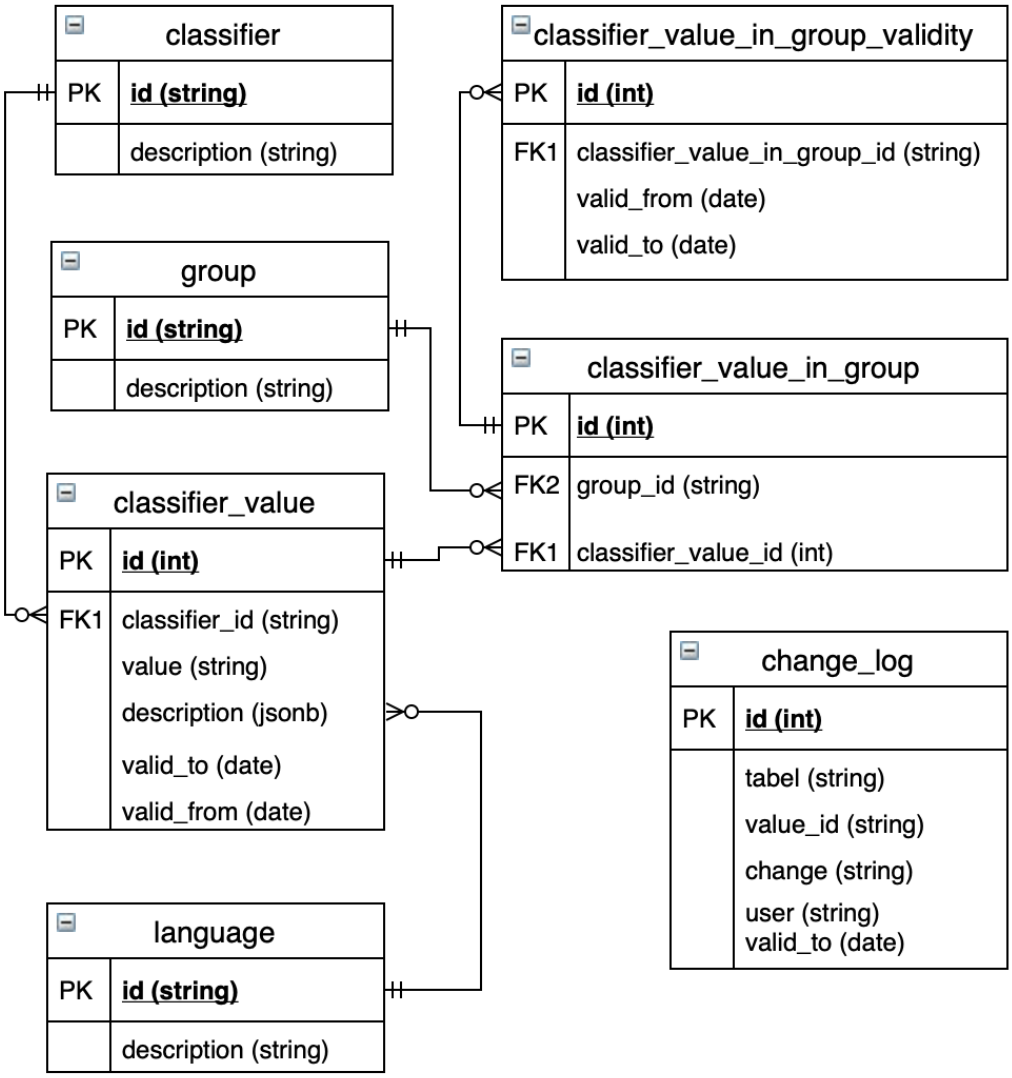
3.1 Andmebaasi mudel

Vastavalt eelnevates peatükkides tehtud analüüsile on vaja koostada andmebaasi mudel, mis toetab kõiki neid nõutavaid funktsionaalsusi, oleks lihtsasti hallatav ning arusaadavalt kirjeldatud. Klassifikaatorid ja grupid on rakenduse põhitegelased, seega mõlemal peab olema oma tabel, mis koosneb kirjeldusest ja ID-st. ID peab nende tabelite puhul olema tekstiline ning võimalikult kirjeldav, et andmebaasist klassifikaatorite väärtusi vaadates oleks paremini aru saada, millega on tegu. Toetavate keelte haldamiseks on mõistlik kasutada sama loogikaga tabelit nagu gruppidel ja klassifikaatoritel.

Klassifikaatorite väärtuste jaoks on keerulisemat tabelit vaja. Kuna neid tuleb lugematul hulgal, siis on mõistlik kasutada põhivõtmena isekasvavat numbrilist ID-d. Tabelil peavad lisaks olema kehtivuste väljad ehk alates mis kuupäevast klassifikaator on kehtiv kuni kuupäevani, mil see kehtib. Seda on tarvis juhul kui toimuvad näiteks nimevahetused. Väli, mis seob omavahel klassifikaatori ja väärtuse, peab antud tabelis ka kindlasti olemas olema. Lisaks on üks väga tähtis väli see, kus hoitakse klassifikaatori nime tõlkeid.

Tarvis on veel tabelit, kust näeb, millised klassifikaatori väärtused millises grupis on ning tabelit, kust näeb, mis ajahetkedel mis klassifikaatorid grupis asuvad. Viimast nendest on tarvis, kuna võib juhtuda olukordi, kus ühel ajahetkel on klassifikaator grupis, aga teisel enam ei ole. Näitena võib tuua Suurbritannia, mis 30.01.2020 seisuga on riik Euroopa Liidus, kuid järgmisel päeval seda riiki enam Euroopa Liidus ei ole.

Muudatuste halduseks on vaja dünaamilist tabelit, kuhu sisestada ükskõik millises tabelis tehtavate muudatuste logi selliselt, et pärast teaks ka seda, millise tabeli väärtuse kohta muudatus kehtis, kes muudatuse tegi ja mida muudatus endast täpsemalt kujutas.



Joonis 1. Andmebaasi tabelite skeem.

Kokku sai planeeritud seitsme tabeliga andmebaasi mudel, mis peaks hõlpsasti tagama kõikide soovitud funktsionaalsuste tagamise antud töös tehtavale rakendusele. Joonis 1 näitab täpselt ära planeeritud tabelite sisu ning nendevahelised seosed. Üks kõige keerukam ülesanne oli klassifikaatori väärtustele tõlgete loogika väljamõtlemine. Võimalusi, kuidas seda kõige mõistlikumalt lahendada, oli kaks, nimelt hoida klassifikaatorite väärtuste tabelis JSONB-ina (Javascripti objekti binaarne notatsioon) või teha eraldi vahetabel keelte ja klassifikaatorite väärtuse vahel. Mõlemad lahendused on sama keerukusega nii kasutajaliidesest haldamise kui ka käsitsi haldamise osas. Antud töös otsustas autor kasutada JSONB-i, et hoida päringutel tehtavate tabelite ühendamiste arvu väiksemana ning kuna rakenduses kasutataval PostgreSQL-i andmebaasil on selle andmetüübi kasutamiseks olemas täielik tugi.

3.2 Taustprogrammi struktuur

Struktuur on iga programmi selgroog. Kui see on korralikult ning loogiliselt üles ehitatud, siis peab programm ka kauem vastu ning sisaldab vähem nõrku kohti. Autor leidis, et põhi taustprogramm on mõistlik jaotada viieks eraldi kihiks, alates andmebaasist kuni päringute haldurini.

Esimeseks kihiks on andmebaas, mis on ühtlasi ka kogu programmi aluseks. Valik oli nelja kõige populaarsema andmebaasi vahel: Oracle, SQLite, MySQL ja PostgreSQL. Oracle andmebaas on küll äärmiselt võimekas, kuid see sai valikust välistatud, kuna seda on keeruline üles seada ning see on kallis. SQLite andmebaasi hakkab rakendus küll kasutama, kuid seda ainult testide jaoks, sest põhirakenduse jaoks ei ole SQLite paraku piisavalt turvaline valik. Kaks kõige reaalsemat varianti olid MySQL ja PostgreSQL, kuid antud töös sai siiski otsustatud PostgreSQL-i versioon 12.2 kasuks. PostgreSQL-i kasuks rääkis fakt, et PostgreSQL toetab rohkemaid andmetüüpe nagu näiteks JSONB ja nende andmetüüpide paremat sidumist. Kui andmed on omavahel piisavalt hästi soetud, siis saab olla kindlam andmete puhtuses. MySQL-i andmebaas on küll kohati kiirem, kuid kuna plaanis on taustprogrammiga andmed korra küsida ning need siis vähemalt tunniks salvestada, siis see eelis kaotab oma väärtuse [9].

Teine kiht on koopia andmebaasi mudelist taustprogrammi ning see on kirjutatud taustprogrammi enda programmeerimiskeele süntaksi järgi. Antud kihti on tarvis, kuna rakendus hakkab kasutama SQLAlchemy tööriista. SQLAlchemy on Pythonile kirjutatud andmebaasiga suhtlemise abiline, mis teeb andmebaasiga suhtluse läbipaistvaks, arusaadavaks ja selgeks objektide kaardistamise abil [10].

Järgmiseks kihiks tuleb andmetele juurdepääsu kiht. Andmetele juurdepääsu kiht ehitatakse antud rakenduse puhul Andmete Juurdepääsu Objekti (DAO) disaini häid tavasid kasutades. Selle kihi põhimõte on andmete küsimine andmebaasist vastavalt vajadusele selliselt, et ülejäänud rakendus andmebaasis otse küsimas ei käi.

Neljandaks osaks rakenduses on äriloogika kiht. Kogu andmete töötlemise loogika asub selles osas. Kiht küsib andmeid andmete juurdepääsu kihist, töötleb need vastavalt vajadusele ning saadab edasi viimasele kihile. Viimane kiht kutsub omakorda äriloogika kihti välja enamjaolt kahel põhjusel. Esiteks sissetulnud andmete valideerimiseks, millele viimane kiht tavaliselt vastust ootab, kas kõik on korras või mitte. Teiseks on nende

sissetulnud andmete töötlemine ning saatmine õigel kujul andmete juurdepääsu kihile vastavalt siis, kas kustutamiseks, küsimiseks või muutmiseks.

Viimaseks kihiks on rakenduse programmeerimisliides (API) [11]. Rakenduse programmeerimisliidese ülesandeks on olla vahelüliks taustprogrammi ja ülejäänud programmide vahel, kes seda taustprogrammi vajavad. Siin toimub päringu vastuvõtmine, vastava päringu õiguse valideerimine, vajalike andmete saatmine äriloogika kihti töötlemiseks ning seejärel vastavalt, kas positiivse või negatiivse vastuse saatmine päringule. Antud töös on see kiht jaotatud kaheks osaks. Esimene osa, kust rakendused soovitud klassifikaatoreid ja gruppe hakkavad küsima ning teine osa on klassifikaatorite ja gruppide haldus kasutajaliidese jaoks tehtav API, mis toetab ka andmete muutmist ja kustutamist.

Lisaks põhi taustprogrammi osale on oluline osa veel ka testidel. Programmides, millega võib töötada palju inimesi, on testid hädavajalikud, et olla kindel programmi töökindluses. Testide kvaliteetsel kirjutamisel selgub üsna kiirelt, kui midagi on tehtud muudatusega katki läinud ja see omakorda säästab paljudest suurematest probleemidest.

4 Rakenduse kasutajaliidese planeerimine

Kasutajaliides on iga rakenduse nägu ja kui see on hästi planeeritud, siis ka iga halduri jaoks hea tööriist. Erinevalt taustsüsteemi planeerimisest tuleb kasutajaliidest planeerides arvestada tugevalt ka visuaalse osaga, mitte ainult koodi loetavuse ja funktsionaalsusega. Rakenduse kasutajaliidese planeerimine jaguneb antud bakalaureusetöös suures osas kaheks: üks osa on programmi sisemise struktuuri paikapanek ning teine kasutajavoo paikapanek.

4.1 Kasutajaliidese programmi struktuur

Programmi struktuuri ülesehitus tuleb suures osas, jälgides Vue.js'i rakenduste ülesehituse häid tavasid. Vastavalt eelnevates peatükkides viidatule on antud projekti puhul kasutatud just seda sama Javascripti raamistikku. Konfiguratsioonidele tuleb oma kaust, mis rakendusega käivitamisel kaasa ei lähe. Konfiguratsioonid loetakse sisse aplikatsiooni käivitamisel. Arenduseks või põhikasutuseks on eraldi failid, mida saab vastavalt vajadusele konfigureerida. Põhirakendus jaguneb seitsme kausta ning nelja juurkaustas asuva aplikatsiooni alusfaili vahel.

Kaustade loetelu ning funktsionaalsuse kirjeldus põhirakenduses:

1. Varad (ingl *assets*) - varade all mõeldaks kõiki üldisemas kasutuses olevaid stiili faile, pilte, ikoone [12].

2. Komponendid (ingl *components*) - komponentide kaustas hakkavad olema kõik rakenduses kasutatavad komponendid. Vue.js raamistik on üles ehitatud just nende samade komponentide baasile, mis selles kaustas olema hakkavad. Komponendina mõeldakse Vue.js'is ühte faili, mis sisaldab vähemalt HTML-i osa, skripti osa, mis antud töös on kirjutatud Typescritpis ja on mõeldud ainult selles failis asuvale komponendile ning stiili osa kuhu on samuti kirjeldatud kõik ainult selles komponendis kehtivad stiilinõuded [12].

3. Ruuter (ingl r*outer*) - selles kaustas hakkab asuma antud rakenduse veebiaadresside loogika [12]. Selles kaustas saab väga edukalt implementeerida ka kasutajate valideerimise osa.

4. Teenused (ingl *services*) - teenuste kausta koguneb äriloogika, mida saab mitmel pool rakenduses kasutada, et ei tekiks palju koodikordusi [12]. Sellesse kausta tulevad näiteks veel ka kõik vajalikud andmete taustprogrammist küsimise loogikad.

5. Ladu (ingl *store*) - antud rakenduses hakkab ladu hoidma kesksest taustsüsteemist küsitud klassifikaatoreid ning gruppe. Ladu võimaldab neid mugavalt küsida ning muuta täpselt vastavalt vajadusele [12].

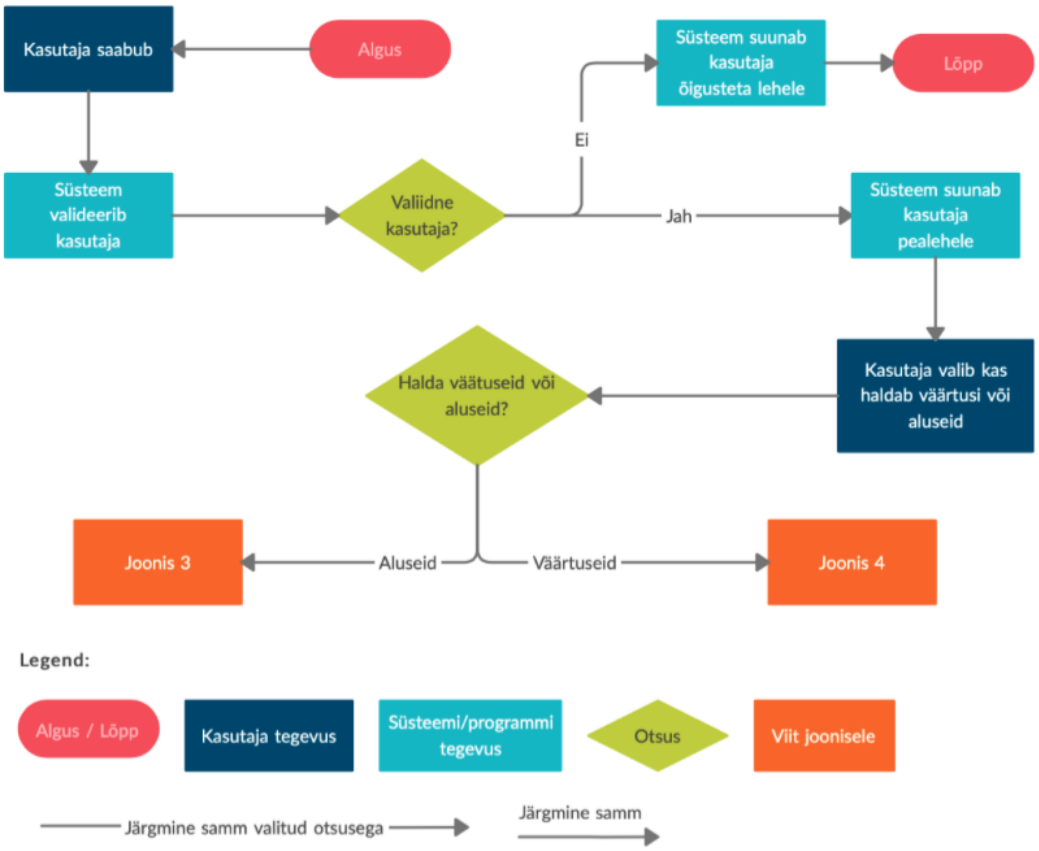
6. Tüübid (ingl *types*) - antud rakendus ehitatakse üles tüübitult, et hoida rakendus arusaadavam ning töökindlam, seega kogunebki sellesse kausta kõikide ise genereeritud andmetüüpide info. Näiteks liidestatakse ära kõik andmetüübid, mida küsitakse taustprogrammist, et need oleksid ühtlased läbi kogu rakenduse.

7. Vaated (ingl *views*) - rakenduse lihtsamaks lugemiseks on veebi erinevad vaated koondatud sellesse kausta. Vaade Vue.js'i mõistes on tegelikult sama, nagu on komponent, mis sisaldab väiksemaid komponente. Hea tava on hoida oma veebiaadressiga komponente selles kaustas [12].

4.2 Kasutajaliidese vooskeem

Kasutajaliidese puhul on kõige tähtsam, et see oleks lihtne, täidaks oma funktsioonid ning oleks arusaadav ka kõige võhikumale arvuti kasutajale. Kogu kasutajaliidese lihtsaks tegemine on aga üks raskemaid ülesandeid. Paremaks arusaamiseks sellest, milline peab kasutajaliides olema ning milliseid protsesse sisaldama, on mitmeid võimalusi.

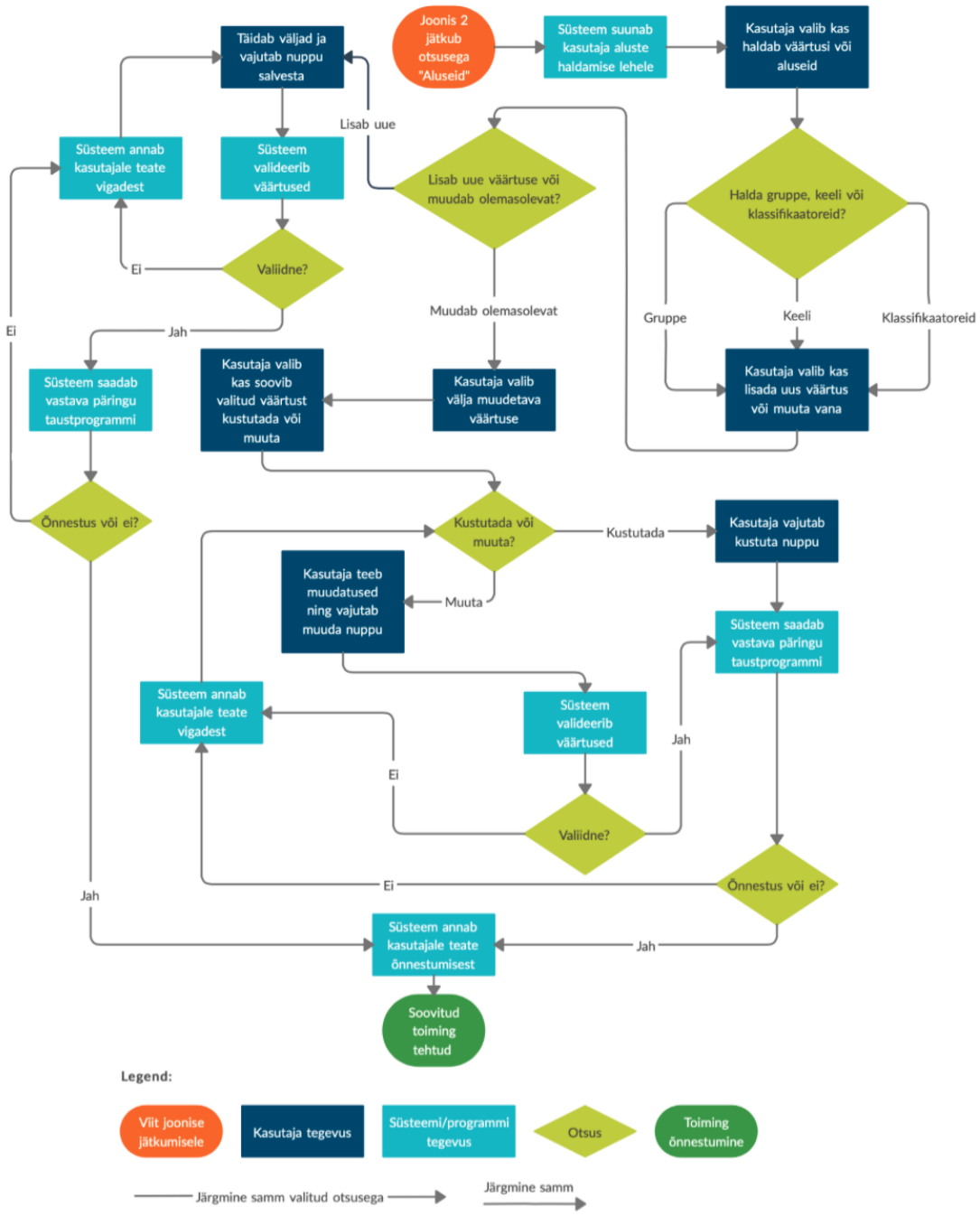
Antud töö jaoks valis autor vooskeemi koostamise meetodi. Vooskeem on, nagu nimigi vihjab, skeem, mis kaardistab kasutaja tegevused, valikud ja otsused, nendele otsustele järgnevad uued võimalused ja nii edasi kuni soovitud protsess on täide viidud. Vooskeemi koostamisel on palju erinevaid eeliseid, mida saab hiljem arendusel ära kasutada. Üks ja kõige olulisem neist on see, et vooskeem annab tervikliku pildi, kuidas asjad kasutajaliideses toimima peavad, kuidas peab süsteem mingites olukordades käituma ning mis funktsionaalsusi kasutajaliides sisaldama peab. Arendajale annab see juhised, kuidas programmi ehitada ning kui keegi peaks arenduse üle võtma, siis on korralik ülevaade programmist tänu vooskeemile alati olemas [13].



Joonis 2. Vooskeemi esimene osa.

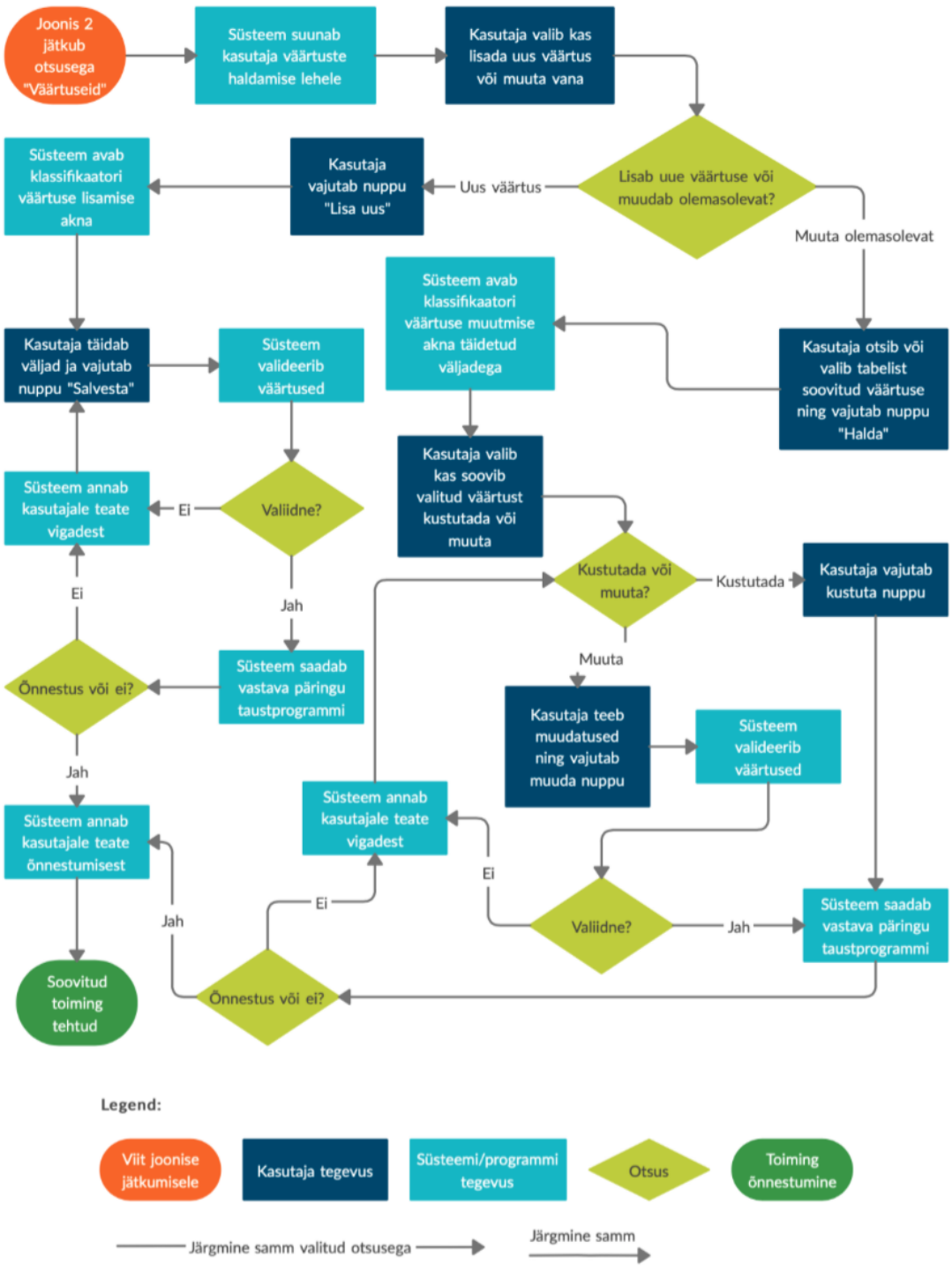
Programmi vooskeem tuli väga mahukas, seega jagas autor selle kolmeks osaks. Joonis 2 kujutab vooskeemi esimest osa, mis on osadest kõige lihtsam ning kirjeldab kasutaja esimest saabumist kasutajaliidesesse. Esimene asi, mida programm teeb, kui kasutaja lehele tuleb, on kasutaja valideerimine. Kasutaja, kellele ei ole õigusi saadetakse lehele, kus on teade „Õigused puuduvad" ja kasutaja seal rohkem midagi teha ei saa. Õiguste olemasolul saab kasutaja minna avalehele, kust saab valida, kas tahab klassifikaatori väärtusi või aluseid muuta.

Joonis 3 kujutab vooskeemi teist osa, mis kaardistab ära kogu süsteemi toimimise loogika, kui kasutaja otsustab minna haldama aluseid. Aluste all on mõeldud gruppe, klassifikaatoreid ning keeli ehk kõiki neid elemente, mida hakatakse süsteemisiseselt liigitamiseks palju kasutama. Klassifikaatori ühe näitena võib tuua klassifikaatori „Riik", mille väärtused on näiteks „Eesti" ja „Suurbritannia". Keel on iseenesest mõistetavalt tõlgete haldamiseks ning gruppidega liigitatakse klassifikaatori väärtusi. Grupi näiteks võib tuua grupi „Euroopa Liit", kuhu kuulub „Eesti", aga „Suurbritannia" sinna ei kuulu.



Joonis 3. Vooskeemi „Aluste" valdamise osa.

Tänu andmebaasi korralikule planeerimisele on võimalik kogu aluste osa väga sarnaselt üles ehitada, kuna kõigi kolme aluse loogika on sama, ainult olemite nimed on erinevad. Joonis 3 illustreerib seda hästi, sest pärast valikut, kas kasutaja soovib gruppe, keeli või klassifikaatoreid hallata, on ülejäänud loogika kõigil nendel valikutel sama. Selliselt on süsteem lihtsam ning saab vältida koodikordusi.



Joonis 4. Vooskeemi „Väärtuste” haldamise osa.

Joonisel 4 kujutatud väärtuste vaade on sisult sarnane Joonisel 3 välja toodud aluste vaate loogikaga. Väärtuste haldamine on küll pisut keerulisem, sest kõikidel alustel oli ainult kaks välja, mida saab muuta või lisada, aga klassifikaatori väärtustel on lisaks veel kehtivused, tõlked ja gruppi kuuluvused, mida saab muuta, kustutada või lisada.

5 Rakenduse tutvustus

Korraliku programmi valmimine nõuab pikka analüüsiprotsessi ning lugematul hulgal programmeerimistunde. Antud bakalaureusetöö raames valminud rakendus on suunatud kasutamiseks kahele kasutajaskonnale ning seetõttu jaguneb ka tutvustuse osa kaheks. Üks osa on suunatud kasutajaliidese kasutamiseks nendele, kes hakkavad klassifikaatoreid ja gruppe haldama ning teine on suunatud arendajatele arendustel abistamiseks.

5.1 Arendajatele suunatud tutvustus

Arendaja aeg on väärtuslik vara ning dokumenteeritud rakendus on abiks kiirele kohanemisele rakendusega. Tutvustus kirjeldab, millised on antud töös valminud rakendusliidesed ning kuidas neid kasutada. Lisaks on ka ülevaade, kus asub Swaggeri dokumentatsioon ning kuidas seda kasutada.

Antud töö probleemi, milleks oli aina enamate rakenduste vajadus kasutada samu ajakohased klassifikaatoreid, lahenduseks sai rakenduse taustsüsteemis loodud kaks rakendusliidest. Esimeseks nendest on rakendusliides, mis tagastab järjendi klassifikaatori väärtustest, kui aadressile parameetriteks kaasa anda kuupäev, millal soovitud klassifikaatorid peavad olema kehtivad ning järjendi klassifikaatori „ID”-dest.

```
http://localhost:5000/api/commondata/classifier_values?valid_on=01.01.2019&cl
assifiers=country,city
```

Joonis 5. Päringu näidis klassifikaatorite „ID”-dega küsides

Antud aadress on näide tehtavast päringust lokaalses masinas eelpool kirjeldatud rakendusliidese pihta. Aadress koosneb kokku kolmest osast. Esimeseks on põhiaadress, kus rakendus asub, teiseks on viide, kus rakendusliides asub ning kolmandaks on parameetrid. Parameetriteks on „*valid_on*” ja „*classifiers*”. Nagu nimigi ütleb, siis „*valid_on*” on parameeter, millega sisestatakse soovitud klassifikaatorite väärtuste kehtivuse kuupäev formaadis „*dd.mm.yyyy*”. Parameetri „*valid_on*” ära jätmisel tagastab rakendus tänasel kuupäeval kehtivad rakendused. Teise parameetriga „*classifiers*” saab sisestada soovitud klassifikaatorite „ID”-d komaga eraldatud järjendina.

```
{
    "city": [
        {
            "classifierId": "city",
            "description": {
                 "EN": "Tallinn",
                 "ET": "Tallinn"
            },
            "id": 11,
            "validFrom": null,
            "validTo": null,
            "value": "555"
            },
    ],
    "country": [
         {
            "classifierId": "country",
            "description": {
                 "EN": "Estonia",
                 "ET": "Eesti",
                 "RU": "Эстония"
            },
            "id": 8,
            "validFrom": null,
            "validTo": null,
            "value": "222"
         },
    ]
}
```


Joonis 6. Päringu vastuse näide Joonisel 5 tehtud päringule

Joonisel 6 on kujutatud näidis, milline tuleb vastus eelpool toodud aadressi puhul. Vastus tuleb JSON-i andmetüübiga. Õnnestumise korral tagastab ka koodi 200, mis tähendab, et päring oli edukas. Vastuse võtmeteks on aadressil parameetriteks antud klassifikaatorite „ID”-d ning nende väärtusteks järjend klassifikaatorite väärtustest, mis on kehtivad parameetris „*valid_on*” määratud kuupäeval.

Teiseks rakendusliideseks on liides, mis tagastab kindlal kuupäeval gruppides asuvad samal kuupäeval kehtivad klassifikaatori väärtused. Aadress, millele päring tehakse on kujutatud Joonisel 7 ja on väga sarnane esimesele.

```
http://localhost:5000/api/commondata/groups?valid_on=01.01.2019&groups=EU,BAL
```

Joonis 7. Päringu näidis gruppide "ID"-dega küsides

Nagu antud aadressilt näha, siis erinevus eelneva aadressiga on rakendusliidese asukoha osa ning ka üks parameetritest. Parameeter „*groups*” erineb esimese rakendusliidese parameetrist „*classifiers*” nii palju, et järjend, mis komaga eraldatult kaasa antakse, on grupi „ID”-dest koosnev mitte enam klassifikaatorite „ID”-dest.

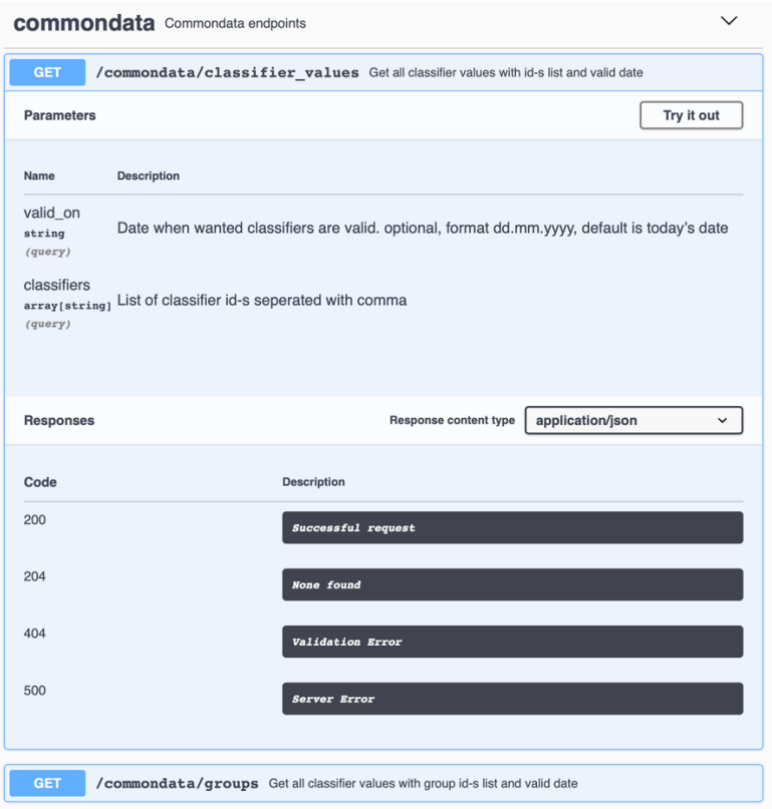
```
{
    "BAL": [
        {
            "classifierId": "country",
            "description": {
                "EN": "Estonia",
                "ET": "Eesti",
                "RU": "Эстония"
            },
            "id": 8,
            "validFrom": null,
            "validTo": null,
            "value": "222"
        }
    ],
    "EU": [
        {
            "classifierId": "country",
            "description": {
                "EN": "Estonia",
                "ET": "Eesti",
                "RU": "Эстония"
            },
            "id": 8,
            "validFrom": null,
            "validTo": null,
            "value": "222"
        }
    ]
}
```


Joonis 8. Vastuse näide Joonisel 7 tehtud päringule

Joonisel 8 on kujutatud vastus gruppide „ID”-dega ning kehtivuse kuupäevaga päringule, mis tuleb taustsüsteemist samuti JSON-i andmetüübis. Võtmeteks selle päringu korral on aga parameetrile „*groups*” kaasa antud gruppide „ID”-d ning nende väärusteks, nagu ka eelmisel rakendusliidesel, list klassifikaatori väärtustest, mis on kehtivad soovitud kuupäeval. Kuupäeva kontroll käib nüüd kahekordselt. Esiteks kontrollib süsteem, et klassifikaatori väärtused oleksid antud kuupäeval kehtivad ning teiseks kontrollib süsteem, et klassifikaatori väärtus oleks antud kuupäeval soovitud grupis. Nagu ka eelmise rakendusliidese puhul on kuupäeva parameetri kaasaandmine vabatahtlik, kui seda mitte kaasa anda, siis kasutab süsteem tänast kuupäeva.

Antud rakenduse kõik rakendusliidesed on kaetud Swaggeri dokumentatsiooniga. Swagger on raamistik, millega dokumenteeritakse rakendusliideseid. Swaggeri abiga on võimalik tekitada rakendusliidestest ülevaade, mis on kasutajaliidese kujul ning sellel on ka tööriistad, millega saab testida kirjutatud rakendusliideseid [14]. Antud taustprogrammi Swaggeri dokumentatsioon asub kahel aadressil, kuna on kaks erinevat rakendusliideste komplekti.

Esimeseks aadressiks on lokaalselt (vt Joonis 9): `http://localhost:5000/api/`



Joonis 9. Swaggeri dokumentatsioon aadressil „/api/”

Teiseks aadressiks on lokaalselt (vt Joonis 10): `http://localhost:5000/api/frontend`

Joonis 10. . Osa Swaggeri dokumentatsioonist aadressil „/api/frontend”

Swagger teeb rakendusliidestega tegelemise arusaadavaks nagu seda näitavad ka näidised Joonistel 9 ja 10. Joonisel 9 on näidatud üks dokumentatsioon peaaegu täies ulatuses. Seal on täpselt kirjeldatud, mis parameetreid rakendusliides vastu võtab ning millised on vastused. Vajutades paremas nurgas olevat nuppu „*Try it out*” avaneb vaade, kus saab sisestada kirjeldatud parameetreid ning päringu sooritades näha, mis tuleb vastuseks.

5.2 Haldajale suunatud kasutajaliidese tutvustus

Antud bakalaureusetöö üheks probleemiks oli, et haldajad ei saa ise klassifikaatoreid hallata. Probleemi lahendamiseks oli vaja teha kaks sammu. Esimeseks sammuks oli andmebaasi struktuuri ülesseadmine selliselt, et tõlked ning kehtivused oleksid andmebaasi tasandil. Taustprogrammi programmeerimisel sai vastavalt Joonisele 1 nii tõlked kui ka kehtivused viidud andmebaasi tasandile. Kõik takistused kasutajaliideses väärtuste haldamiseks sai sellega seljatatud. Teine samm probleemi lahendamiseks oli kasutajaliidese programmeerimine. Kasutajaliides koosneb neljast vaatest. Esimene vaade on seotud kasutaja valideerimisega. Kuna antud kasutajaliidesele saab ligi ainult läbi UUSISe siis kehtib ka sellele kasutajaliidesele ISKE (Infosüsteemide turvameetmete süsteem) kõige kõrgem turvatase.

Vabandage, Teil puudub rakendusele ligipääs!

Joonis 11. Õigused puuduvad vaade

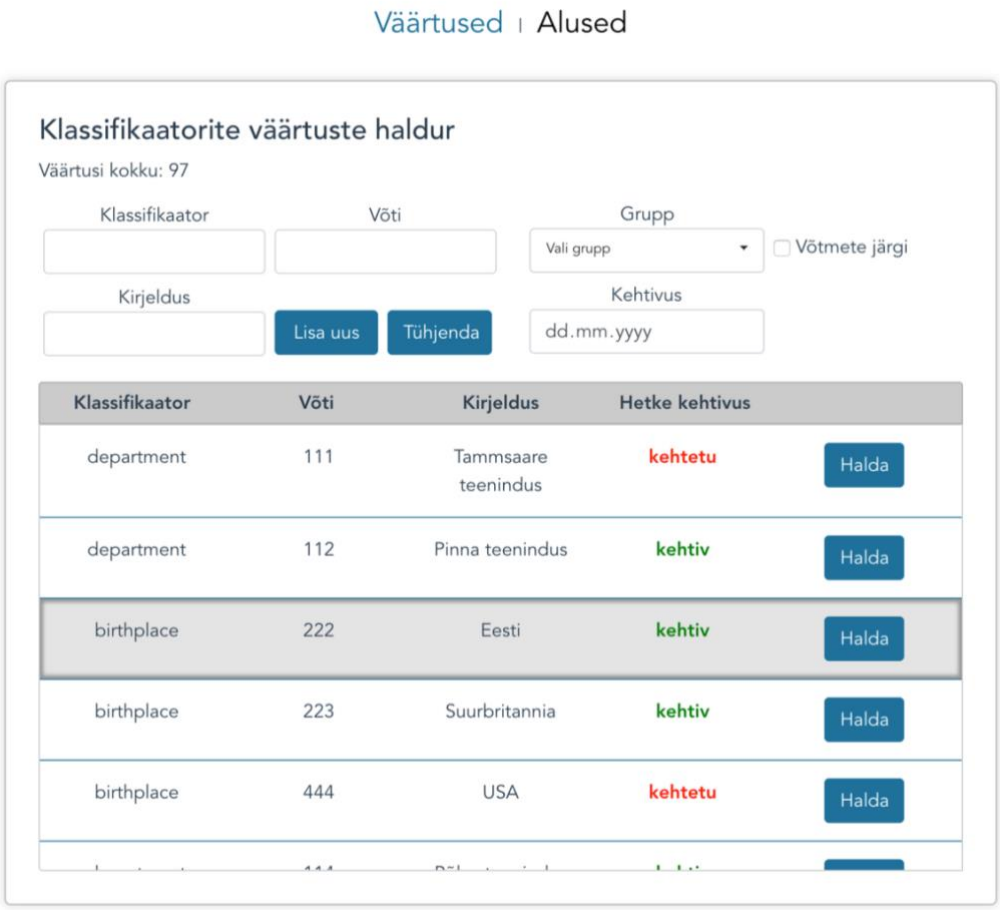
Esimene vaade tehtud kasutajaliideses teavitab kasutajat, kellel ei ole õigusi sellele lehele minna, et tal puudub ligipääs. See vaade on toodud Joonisel 11 ning kasutajal, kes seda vaadet näeb, ei ole muud teha, kui lehelt ära minna, sest ta ei saa lehel midagi teha. Kui aadressivälja kaudu üritatakse ilma õigusteta edasi minna mõnele kasutajaliidese lehele, siis suunatakse kasutaja uuesti sellele samale lehele tagasi.

Õigustega kasutaja pääseb aga kohe pealehele. Pealehe ülesehitus, mis on kujutatud Joonisel 12, on väga minimalistlik. See koosneb ainult pealkirjast, tervitusest sisseloginud kasutajale ning kahest nupust. Üks nendest nuppudest viib kasutaja klassifikaatorite väärtuste halduse lehele ning teine aluste halduse lehele.

Joonis 12. Pealeht

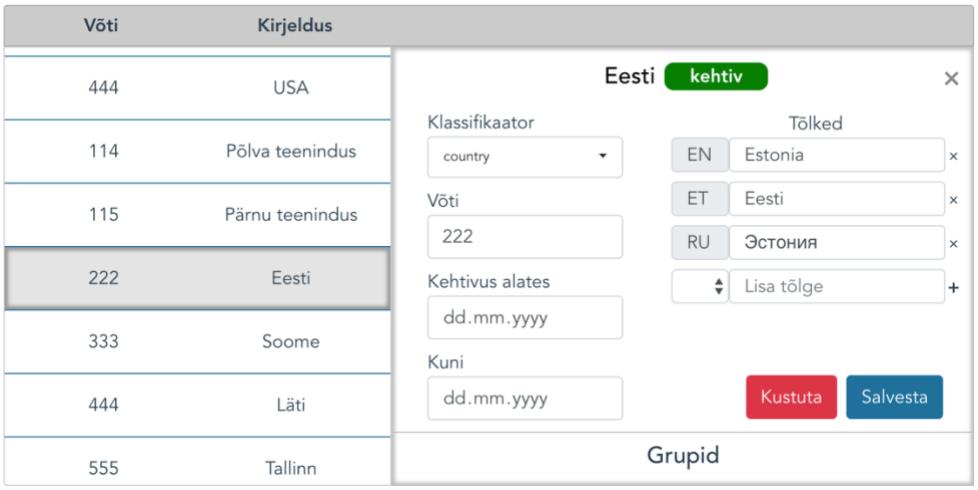
Klassifikaatorite väärtuste halduse vaade, mis on kujutatud Joonisel 13, on juba natukene keerulisem. Lehe üleval asub navigaator, kus saab nii aluste kui ka väärtuste halduse lehtede vahel navigeerida. Kogu ülejäänud lehe vaade on ülesehitatud tabeli põhimõttel. Tabeli päises on välja toodud, kui palju on antud rakenduses erinevaid klassifikaatorite väärtusi. Väärtuste loendurile ning pealkirjale järgneb peamiselt kahe funktsiooniga osa. Nimelt saab antud osas otsida kõikide nende väärtuste seast, mis on salvestatud, kas klassifikaatori, klassifikaatori väärtuse võtme või kirjelduse järgi ning tabel uuendab ennast automaatselt. Lisaks on võimalik võtme või kirjelduse järgi otsida ka gruppi ning leida kõik need klassifikaatori väärtused, mis valitud gruppi kuuluvad. Otsingu osas kuupäeva väljale minnes avaneb väike kalender ning sinna on võimalus, kas ise kirjutada või valida kalendrist soovitud kuupäev. Kuupäeva välja täites näitab tabel ainult neid väärtusi, mis antud kuupäeval kehtivad.

Selleks, et vältida samasuguste väljade topelt tekitamist, sai antud rakenduses ära kasutatud otsinguväljad „Klassifikaator", „Võti" ja „Kirjeldus" uue klassifikaatori lisamiseks. Kirjeldus läheb automaatselt eesti keele tõlkeks ning klassifikaatori väärtus hakkab vaikimisi kehtima päevast, mil see loodi. Lisatud klassifikaatori väärtusele saab pärast lisamist veel täiendavaid väärtusi lisada, muuta või kustutada avanevast täpsema info aknast. Nupp „Tühjenda" puhastab kõik otsingu väljad. Tabelis on iga terve rida kui nupp, mis täidab sama funktsiooni, mis nupp „Halda" ehk avab klassifikaatori väärtuse täpsema info akna selles samas tabelis. Täpsema info akna avanedes on tabel jaotatud kaheks. 2/5 on tabelis väärtuste listi jaoks, mis kuvavad ainult võtit ja kirjeldust. Ülejäänud 3/5 tabelist on klassifikaatori väärtuse täpsema info akna päralt.



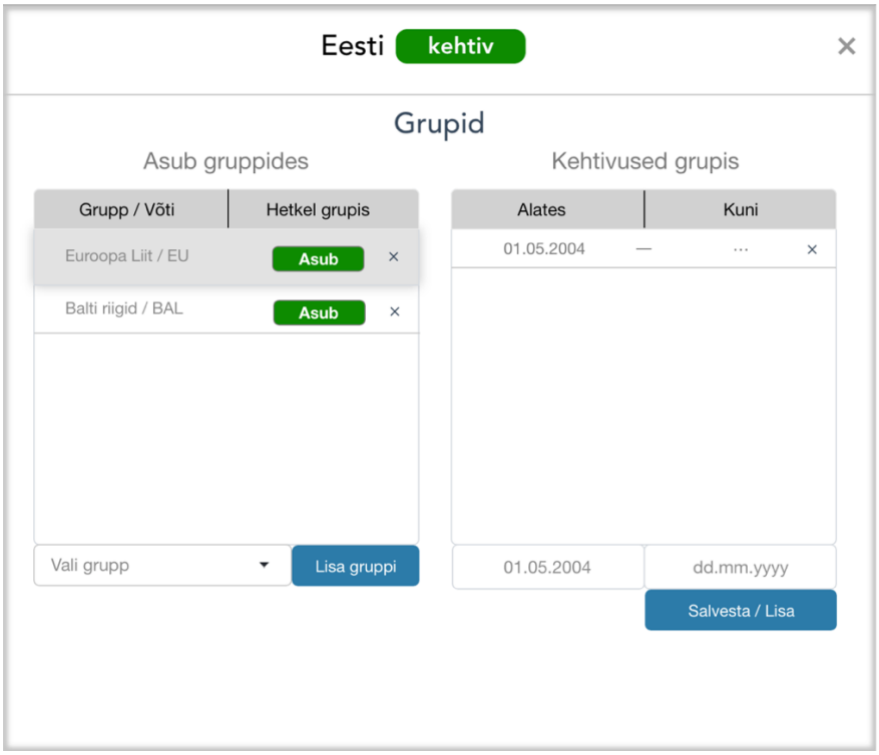
Joonis 13. Klassifikaatorite väärtuste halduse leht

Täpsema info esimeses aknas (vt Joonis 14) on võimalik muuta klassifikaatori väärtuse klassifikaatorit, võtit ning määrata kehtivust. Lisaks on võimalik ka kõiki tõlkeid samas aknas hallata. Klassifikaatori väärtust on võimalik soovi korral kustutada. Täpsema info akna alumises ääres on nupp nimega „Grupid", mis avab gruppide haldamise sektsiooni.



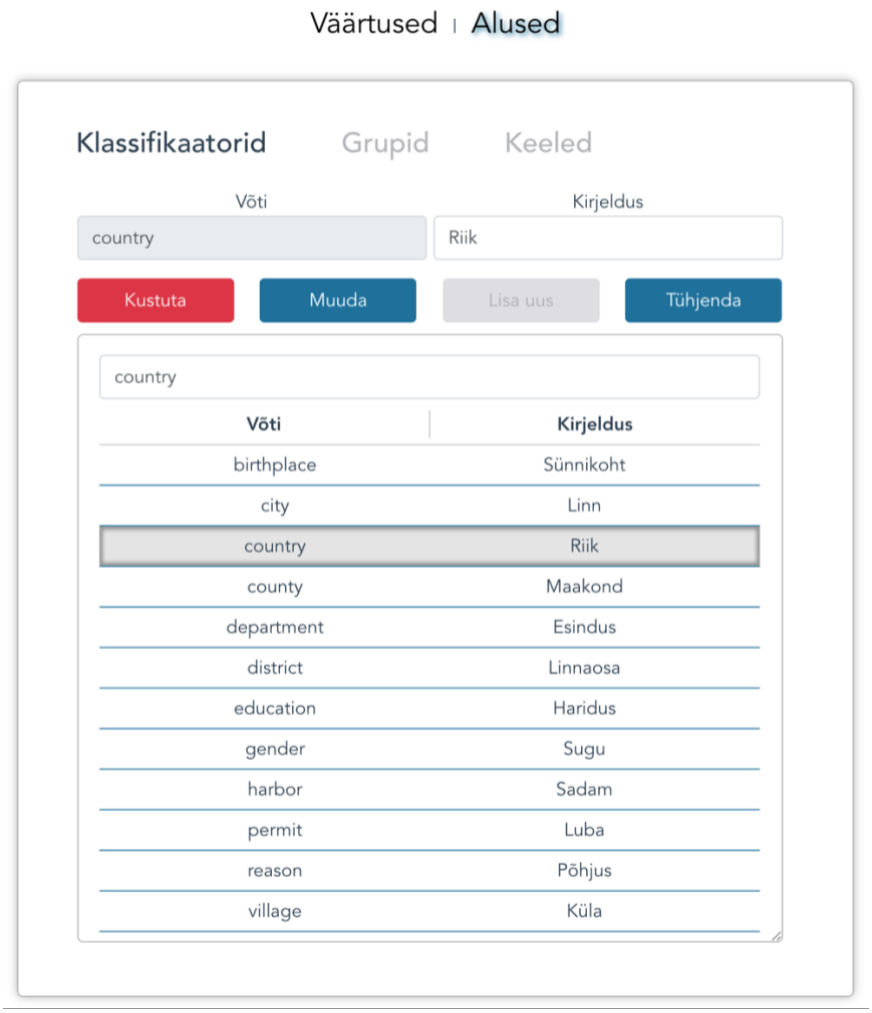
Joonis 14. Klassifikaatori väärtuse täpsema info aken

Joonisel 15 kujutatud gruppide haldamise sektsioon jaguneb kaheks. Vasakul pool on näha, mis gruppides valitud klassifikaatori väärtus asub ning kui teha üks nendest gruppidest aktiivseks, siis parempoolses osas tuleb nähtavale, mis ajahetkedel klassifikaatori väärtus valitud grupis asub. Vaate vasakpoolse osa alumises ääres on võimalik väärtust ka uude gruppi lisada. Gruppi lisades lisab süsteem grupis „Alates“ väärtuseks vaikimisi tänase kuupäev. Parempoolsel osal kuupäevale vajutades täituvad kuupäeva väljad ning neid on vastavalt vajadusele võimalik muuta. Gruppe saab eemaldada grupi kirje lõpus asuvast ristist ning ka kehtivuse eemaldamine on samamoodi lahendatud.



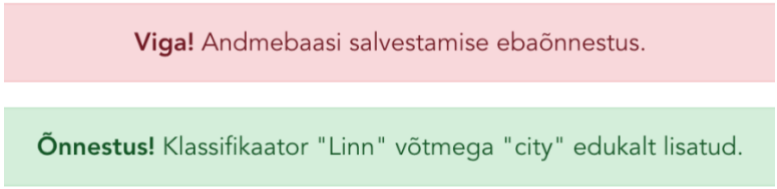
Joonis 15. Klassifikaatori väärtuse gruppide haldamise aken

Kasutajaliidese viimaseks vaateks on aluste haldamise vaade, mida kujutab Joonis 16. Vaate ülemises osas on valik erinevatest alustest. Valides aluste grupi, mida soovitakse muuta, täitub ka allpool olev tabel kõikide valitud alustega. Tabelis on olemas ka otsinguväli, mis arvestab nii aluste väärtusi kui ka kirjeldusi. Otsingut sooritades värskendab tabel automaatselt. Kui tabelist ei ole mitte ühtegi alust valitud, siis saab ülemises osas uut alust lisada. Väljad täites ning vajutades nuppu „Lisa uus”, valideerib süsteem kõigepealt väljad ning kui kõik on korrektne saadab süsteem kirje taustprogrammi ja uus kirje ongi loodud. Kirje valimisel tabelist, täituvad ülemised sisestusväljad „Võti” ja „Kirjeldus” ning aktiivseks muutuvad ka „Kustuta” ja „Muuda” nupud.



Joonis 16. Aluste haldamise vaade

Olemasoleva kirje muutmise korral võtit muuta ei saa, kuna see on suure tõenäosusega seotud erinevate klassifikaatori väärtustega. Ainuke võimalus võtit muuta on kirje kustutada, mis omakorda kustutab ka teistest seotud tabelitest sama kirjega põimitud väljad ning siis uue kirje lisamisega. Kõik sooritatud tegevused on kaetud õnnestumise või ebaõnnestumise teadetega, mida illustreerib Joonis 17. Teated kuvatakse lehtede navigaatori ja lehe vaate vahelisel alal kuni 10 sekundit.



Joonis 17. Veateate ning õnnestumise teate näidis

Kokkuvõte

Antud bakalaureusetöö põhieesmärk oli valmis ehitada rakendus, mis lahendaks kaks suurt probleemi menetlussüsteemis UUSIS. Esimene nendest probleemidest oli, et arendajad, testijad ning tooteomanikud tegelevad klassifikaatorite ja nendega seonduva haldusega, mitte haldurid ning peakasutajad, kes nende eest tegelikult vastutavad. Teine probleem tuleneb UUSISe mikroteenustele üleviimise suunast ning enamate dokumentide taotlemise e-taotluskeskkonda viimisest. Aina rohkem arendatakse rakendusi, millel on vajadus ajakohaste klassifikaatorite väärtuste järgi, kuid puudub ühtne viis, kuidas neid saada.

Bakalaureusetöö esimeseks osaks oli kasutatavate tehnoloogiate valiku ning nii taustprogrammi kui ka kasutajaliidese analüüs. Analüüsi üks punkte oli kaardistada võimalikud rakenduse kasutajad. Kasutajagruppe on antud rakendusel kolm: arendajad, menetlussüsteemi haldurid ning teised rakendused, mis hakkavad antud rakenduse suunas päringuid tegema. Kasutajalugudena sai kaardistatud ka kõik võimalikud toimingud mikroteenuses ning see, mis on nende toimingute eesmärgiks.

Taustprogrammi analüüs ning planeerimine aitas luua selge pildi, milline peab olema andmebaasi struktuur, et toetada kõikide antud töös lahendatavate eesmärkide saavutamist. Andmebaasi struktuuri sai varakult sisse planeeritud esimese probleemi lahendust toetav samm, milleks oli klassifikaatorite väärtuste tõlgete ning kehtivuste klassifikaatori väärtuse külge lisamine. Lisaks ka eraldi tabeli loomine, haldamaks millal klassifikaatori väärtus grupis asub. Tõlked on klassifikaatori väärtuse küljes JSONB andmetüübina ning ka kehtivuse algus ja lõpp asuvad samas tabelis sama kirje küljes kuupäeva andmetüübina. Klassifikaatori väärtuse gruppi kuulumise kehtivuste haldamiseks oli tarvis eraldi tabelit, kuna väärtus võib olla ja võib mitte olla grupis mitmetel erinevatel aegadel. Lisaks toetas taustprogrammi analüüs struktuuri paika panemist. Taustprogrammi struktuuriks sai valitud kihiline struktuur, mis koosnes viiest kihist: andmebaasist, andmebaasi mudeli koopiast taustprogrammis, andmete haldamise kihist, äriloogika kihist ning rakendusliideste kihist.

Kasutajaliidese analüüsis mängis suurt rolli vooskeemide paika panek. Vooskeem sai planeeritud varasemalt mainitud kasutajalugude põhjal ning oli kasuks igal

kasutajaliidese programmeerimise sammul. Samuti aitas kasutajaliidese analüüs paika panna rakenduse koodi struktuuri vastavalt Vue.js headele tavadele.

Rakenduse programmeerimine algas taustprogrammi ülesehitamisest Pythoni programmeerimiskeeles, kasutades ka Flaski mikroteenust ning SQLAlchemy andmebaasi ühendust. Andmebaasiks sai valitud PostgreSQL ning sinna sai ehitatud seitsmest tabelist koosnev andmebaas. Taustprogrammi ehitusega sai lahendatud üks antud töö probleemidest ning tehtud tugi teise probleemi lahenduseks ehk tugi kasutajaliidesele. Probleemiks olnud mitmete rakenduste vajadus ajakohaste klassifikaatorite väärtuste järgi leidis lahenduse rakendusliideste näol. Esimene rakendusliides võtab parameetriks järjendi klassifikaatorite võtmetest ning soovi korral kuupäeva, millal need kehtivad peavad olema ning päringule tagastatakse kõikide soovitud klassifikaatorite väärtused. Teine rakendusliides võtab parameetriks järjendi gruppide võtmetest ning vajadusel kehtivuse kuupäeva ja tagastab vastavates gruppides antud kuupäeval asuvad ning vastava kehtivusega klassifikaatori väärtused. Kõik rakendusliidesed on Swaggeri abiga dokumenteeritud.

Teise suure probleemi lahendamiseks arendas autor kasutajaliidese, mis koosneb neljast erinevast vaatest. Esimene nendest on peavaade, kus saab valida kahe erineva valiku vahel: kas muuta aluseid või klassifikaatori väärtusi. Aluste all on antud töös mõeldud klassifikaatoreid, gruppe ning keeli. Teine vaade tuleb ette juhul kui puudub õigus kasutajaliideses tegutsemisele ning see koosneb teatest „Õigused puuduvad”.

Aluste halduse leht on keerulisem ning koosneb rohkematest elementides. Andmebaasi mudel on selliselt paika pandud, et kõiki aluseid oleks võimalik sarnaselt hallata, kuna kõik need koosnevad väärtustest „ID” ja „*description*”. Lehel asub otsing, kust saab reaalajas otsida, kas gruppide, klassifikaatorite või keelte seast ning kui valida väärtus mida soovitakse muuta, siis täidetakse tekstiväljad vastavate väärtusega automaatselt, et kasutamist lihtsamaks teha. Võimalus on lisada uusi aluseid, muuta olemasolevad ning olemas on ka kustutamise võimalus. Iga toimingu õnnestumisel või ebaõnnestumisel kuvab rakendus ka vastava teate. Kõiki välju valideeritakse, et süsteemi ei satuks vale tüüpi andmed.

Kõige keerulisem vaade rakendusest on klassifikaatorite väärtuste haldamise vaade. Antud töös on see ülesehitatud suure tabelina, kus on kõik need väärtused reas ning tabeli

päises on võimalik reaalajas, kas klassifikaatori võtme, väärtuse võtme, kirjelduse või grupi järgi otsingut teostada ning seejärel tabel uueneb automaatselt. Tabel näitab kohe ka seda, kas klassifikaator on tänasel päeval kehtiv või mitte. Väärtusele vajutades läheb tabel 2/5 suuruseks ning ülejäänud 3/5 ulatuses avaneb valitud väärtuse haldamiseks uus aken. Selles aknas saab kustutada väärtust ennast, tõlget, grupis asumist või grupis asumise kehtivusi. Muuta saab väärtuse klassifikaatorit ning kõiki väärtusi, mis on klassifikaatori väärtusel või mis on klassifikaatori väärtusega seotud. Samuti on võimalik selles aknas lisada klassifikaatori väärtust mõnda soovitud gruppi ja määrata, mis kuupäevade vahemikus väärtus grupis asub. Tõlgete lisamine on lahendatud sarnaselt nagu gruppidesse lisamine ainult ilma kehtivusteta.

Ehitatud kasutajaliides annab võimaluse halduritel ja peakasutajatel klassifikaatoritega tegeleda ilma, et peaks tegema arenduse pileti ning ootama, millal keegi arendajatest selle ära teeb. Haldurid saavad arendatud kasutajaliideses kõik vajalikud toimingud ise ära teha ja sellega ka väärtuslikku aega kokku hoida.

Kasutatud kirjandus

[1] E. Nagel, A. Korsar ja M. Lausmaa, „Arhitektuuridokument,“ 30 01 2017. [Võrgumaterjal]. Available: https://www.riha.ee/api/v1/systems/kmais/files/96134aba-6fa1-417b-b524-213029825279. [Kasutatud 12. aprill 2020].

[2] S. Pramanick, „History of Python,“ [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/history-of-python/. [Kasutatud 02. aprill 2020].

[3] M. Makai, „Flask,“ 2012-2020. [Võrgumaterjal]. Available: https://www.fullstackpython.com/flask.html. [Kasutatud 06. aprill 2020].

[4] „What is Flask used for?,“ 16 November 2019. [Võrgumaterjal]. Available: https://dev.to/amigosmaker/what-is-flask-used-for-2do5. [Kasutatud 12. aprill 2020].

[5] J. Jackson, „Microsoft augments javascript for large scale development,“ U.S. Correspondent, IDG News Service, 1 October 2012. [Võrgumaterjal]. Available: https://www.cio.com/article/2391735/microsoft-augments-javascript-for-large-scale-development.html. [Kasutatud 31. märts 2020].

[6] V. Cromwell, „Between the Wires: An interview with Vue.js creator Evan You,“ 30 May 2017. [Võrgumaterjal]. Available: https://www.freecodecamp.org/news/between-the-wires-an-interview-with-vue-js-creator-evan-you-e383cbf57cc4/. [Kasutatud 31. märts 2020].

[7] M. Cardus, „Quality tools to discover solutions: NINE WINDOWS,“ [Võrgumaterjal]. Available: https://mikecardus.com/quality-tools-to-discover-solutions-nine-windows/. [Kasutatud 11. aprill 2020].

[8] G. Krasadakis, „How and why to write great user stories,“ 22 June 2018. [Võrgumaterjal]. Available: https://www.freecodecamp.org/news/how-and-why-to-write-great-user-stories-f5a110668246/. [Kasutatud 11. aprill 2020].

[9] M. D. a. ostezer, „SQLite vs MySql vs Postgresql a comparison of relational database management systems,“ 19 March 2019. [Võrgumaterjal]. Available: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. [Kasutatud 30. märts 2020].

[10] SQLAlchemy authors and contributors, „SQLAlchemy,“ [Võrgumaterjal]. Available: https://www.sqlalchemy.org. [Kasutatud 05. märts 2020].

[11] P. Gazarov, „What is an api in english please,“ 19 December 2019. [Võrgumaterjal]. Available: https://www.freecodecamp.org/news/what-is-an-api-in-english-please-b880a3214a82/. [Kasutatud 03. aprill 2020].

[12] S. Adittane, „How to structure a vue.js project,“ 12 June 2018. [Võrgumaterjal]. Available: https://itnext.io/how-to-structure-a-vue-js-project-29e4ddc1aeeb. [Kasutatud 10. aprill 2020].

[13] A. Pochimcherla, „FLOWCHARTS,“ [Võrgumaterjal]. Available: http://steamism.com/flowcharts/. [Kasutatud 17. aprill 2020].

[14] M. Rouse, „Swagger,“ August 2019. [Võrgumaterjal]. Available: https://searchapparchitecture.techtarget.com/definition/Swagger. [Kasutatud 29. aprill 2020].