TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Jaan Susi 162769IABM
Martin Rajur 162745IABM
Annett Lymar 17278IABM

Eesti Kirjandusmuuseumi jaoks rahvaviiside noodistuse standardi juurutamine ja infosüsteemi edasiarendus

Magistritöö

Juhendaja: Erki Eessaar
PhD

Tallinn 2021

Autorideklaratsioon

Kinnitame, et oleme 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.

Autorid: Jaan Susi, Martin Rajur, Annett Lymar

17.05.2021

Annotatsioon

Lõputöö eesmärgiks on Eesti Kirjandusmuuseumile rahvaviiside haldamise infosüsteemi uue versiooni kavandamine ja realiseerimine. Töö arendab edasi 2020. aastal bakalaureusetöö tulemusena loodud rahvaviiside infosüsteemi. Ülesandepüstitus on koostatud Eesti Kirjandusmuuseumi poolt. Loodav tarkvara peab lihtsustama rahvaviisidega seotud informatsiooni haldamist.

Üheks suuremaks ülesandeks on rahvaviiside noodistuse standardi juurutamine. Noodistuse standard aitab noodistuse kodeerimise teel digitaliseerida rahvaviise selliselt, et lisaks viisiga seotud metaandmetele salvestatakse ka selle korrektne laialdaselt kasutatud standardile vastav noodistus. Loodud noodistusi on hiljem võimalik kasutada täiendavateks uurimusteks ning analüüsiks. Standardi valimiseks kasutati Analüütiliste Hierarhiate Meetodit (Saaty meetodit) ja juurutatavaks standardiks valiti ABC notatsioon. Töö tulemusena loodi teisendusutiliit (https://github.com/nitramra/EKM_converter) olemasolevas kodeeringus noodistuste teisendamiseks valitud standardile vastavaks noodistuse esituseks. Olemasolevad noodistused kanti uues kodeeringus uude süsteemi versiooni üle.

Töö käigus loodi uus infosüsteemi versioon, tuginedes juba eelnevalt tehtud süsteemianalüüsile ja andmebaasi disainile. Lisaks eelnevalt realiseeritud funktsionaalsuse uuel platvormil uuesti realiseerimisele, loodi võimalus viiside juurde kodeeringute lisamiseks ja helifailide ettemängimiseks.

Kirjandusmuuseumile antakse üle tarkvara lähtekood ja teisendatud andmed vastavalt andmebaasi struktuurile. Tarkvara arendamisel lähtuti põhimõttest, et edaspidine süsteemi hooldus ja edasiarendus oleks lihtsasti teostatav antud lõputööga mitteseotud arendajate poolt.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 137 leheküljel, 10 peatükki, 108 joonist, 14 tabelit.

Abstract

Implementation of a Music Notation Standard and Further Development of the Folk Tunes Information System for the Estonian Literary Museum

The goal of this thesis was to design and develop a new version of the folk tunes information system for the Estonian Literary Museum. The current work continues the work that was started by a bachelor thesis in 2020 that resulted with the first version of the information system software. To achieve the goal additional assessment was done to find missing functionality. The resulting software must simplify information management. The authors reimplemented the designed functionality from the first version by using new tools – Node.js, Loopback 4, and React instead of CakePHP in the first version. The database design and platform (PostgreSQL) remained the same. However, the database was extended with new base tables. For the development process Scrum development methodology framework was used.

In addition, a music notation standard was selected. The standard is used to encode folk tunes so that these could be analysed, rendered, and converted into music. We used Analytic Hierarchy Process (Saaty method) for the selection task. The selected standard was ABC notation standard. Most of the already coded tunes from the previous system were converted to the selected encoding by using a custom-made converter utility (https://github.com/nitramra/EKM_converter). Thus, the already done work regarding encoding manuscripts and audio was not lost.

The result of the work is a software that can be used to store metadata regarding different tunes, attach encoded musical notation to the tunes, reference the original materials from Kivike information system, which contains the original documents, and change data about multiple tunes at the same time. In the new system version tunes can be played within the web browser based on the stored encoding of the tune. The system logs data changes of tunes.

The resulting software, including the source code, will be given to the Estonian Literary Museum. In the development a principle that the result must be well maintainable and extendable by developers who did not create the original version was applied.

The thesis is in Estonian and contains 137 pages of text, 10 chapters, 108 figures, 14 tables.

Lühendite ja mõistete sõnastik

ABC Notation	Noodistuse kirjeldamise standard.
ADR	Action Design Research, disaini tegevusuuring. Meetod teadusuuringute läbiviimiseks, mis kombineerib tegevusuuringu ja disainiteaduse põhimõtteid. Eesmärgiks on luua tehniline artefakt, seda paralleelselt juurutades ja tulemusi jälgides ning lõpuks sellest kõigest midagi uut õppides.
AHP	Analüütiliste hierarhiate protsess e meetod. Otsustusmeetod, mida kasutatakse alternatiivide võrdlemiseks kriteeriumite alusel. Tuntud ka looja nime järgi ehk Saaty meetodina. Võimaldab muuta muidu subjektiivsetel hinnangutel põhinevat otsustamist objektiivsemaks.
Alusheli	Alusheli on muusikas diatoonilise helisüsteemi heli, mida tähistatakse Vana-Kreeka tähtnotatsiooni eeskujul tähega A, B, C, D, E, F või G. Mõiste "alusheli" (Stammton) on pärit saksa keeleruumist ning saksa süsteemis on alushelid C, D, E, F, G, A ja H.
Arhivaal	Arhiivi dokument mida säilitatakse arhiivis püsivalt. Dokument, millele arhiiv on hinnanud säilitamist väärivaks.
Artefakt	Inimeste poolt loodud tehis, kunstlik looduses loomulikul kujul mitte esinev asi.
ASCII	Levinud märgikood lihtteksti esituseks. [1]
Autentimine	Protsess, millega kontrollitakse kasutaja identiteeti läbi mandaadi või sessiooniloa valideerimise.
Autoriseerimine ehk volitamine	Protsess, millega kontrollitakse autenditud kasutaja ligipääsuõiguseid ressurssidele.
Bemoll	Heli madaldamine poole tooni võrra, tähistatakse noodipea ette lisatava märgiga. [2]
CRUD	*Create, Read, Update, Delete*. Andmete lisamine, lugemine, uuendamine ja kustutamine on tüüpilised kasutaja tegevused andmebaasirakendustes.
CSV	Tekstifaili tüüp, sisaldab vorminguta andmeid. Kirjeid eraldab reavahetuskood ja kirje välju eraldavad komad. [1]
Deskriptiivne (noodistus)	Muusika ülesmärkimise viis, mis kirjeldab kuidas teose esitus kõlab.
Diatooniline helistik	Helistik mis moodustub ainult seitsmest alushelist (c, d, e, f, g, a, b) ilma kõrgenduste (diees) ja madaldusteta (bemoll).

Diees	Heli kõrgendamine poole tooni võrra, tähistatakse noodipea ette lisatava märgiga. [2]
Dockerfile	Tarkvara konteineri käitamiseks vajalike käskluste kogum
DOM	*Document Object Model* (DOM). Dokumendi objektimudel on mitmeplatvormiline ning keelest sõltumatu viis objektide veebilehel esitamiseks.
EKM	Eesti Kirjandusmuuseum.
ERA	Eesti Rahvaluule Arhiiv.
GDPR	EU *General Data Protection Regulation*. Euroopa Liidu isikuandmete kaitse üldmäärus.
Helivältus	Noodikirja märk, mis tähistab heli või vaikuse kestust.
HTTP	Andmevahetusprotokoll, mida kasutatakse Internetis selleks, et vahetada dokumente.
HTTPS	*Hypertext Transfer Protocol Secure*. Protokoll krüpteeritud teadete edastamiseks arvutivõrkudes.
ISKE	Eesti riigi infosüsteemide kolmeastmeline turvameetmete kogum, mille eesmärk on tagada infosüsteemide piisava tasemega turvalisus.
JavaScript	Objektorienteeritud programmeerimiskeel dünaamilise veebisisu skriptide loomiseks. [1]
JSON	*JavaScript Object Notation*. JavaScriptil põhinev lihtsustatud, inim- ja masinloetav andmevahetusvorming
JWT	*JSON Web Token*. JSON vormingu põhine loa standard õiguste turvaliseks jagamiseks kahe osapoole vahel.
Kivike (KIVIKE)	Eesti Kirjandusmuuseumi infosüsteem ja failihoidla, mis sisaldab digitaliseeritud arhiivimaterjale ja metaandmeid. [3]
MIDI	*Musical Instrument Digital Interface*. Käsitleb digitaalseid käske, mis on seotud muusikalise teosega (nt tempo, helikõrgus, helitugevus jne).
Metaandmed	Kirjeldavad andmed objekti kohta.
MS Access	Microsofti arendatud töölaua andmebaasisüsteem, mis kuulub kontoripaketi MS Office koosseisu. Võimaldab kasutada SQL andmemudelit ja andmebaasikeelt.
MusicXML	Noodistuse kirjeldamise standard.
Noodi pikkus	Tähistab heli kestust.

Noodistus	Noodistus on muusikaheli visuaalne vorm. Selle kirjeldamiseks kasutatakse noodikirja. [4]
Oktav	Vahemikku kahe teineteisele lähima helirea esimese astme (c) vahel nimetatakse oktaviks. [2]
ORM	*Object-relational mapping*. Tehnika andmete teisendamiseks mitme erineva mitteühilduva süsteemi vahel, kasutades selleks objektorienteeritud programmeerimiskeelt.
PHP	*Hypertext Preprocessor* Skriptimiskeel, mida kasutatakse serveripoolse lahendusena dünaamiliste veebilehtede loomiseks.
PID	Arhivaali unikaalne identifikaator Kivikese repositooriumis.
PostgreSQL	Avatud lähtekoodiga ja tasuta pakutav serveri andmebaasisüsteem, mis pakub kasutajatele SQL andmebaasikeelt ja võimalust luua andmebaasi SQL andmemudeli põhjal.
Repositoorium	Andmebaas, kus hoitakse digitaalseid säilikuid.
Rütmitüüp	Rütmitüüp on analüüsi teel saadud mudel, mis näitab üldistatult viisi rütmilist struktuuri teatud muusika kontekstis. Rütmitüüpe saab kasutada noodistamise alusena.
REST	Representational state transfer. Tarkvaraarhitektuuri laad, mis seab piiranguid veebirakenduse loomisele. [5]
Scrum	Tarkvaraarenduse paindmetoodika e agiilne metoodika.
SOAP	*Simple Object Access Protocol*. Struktuursete andmete vahetamise protokoll erinevate veebiteenuste vahel.
SQL	Relatsioonilist andmemudelit realiseeriv andmebaasikeel, mis võimaldab muuhulgas nii andmete käitlemist, andmebaasiobjektide haldamist, tehingute juhtimist kui ka õiguste jagamist.
SSL	*Secure Socket Layer* Krüptograafiline protokoll ehk turbeprotokoll.
UML	*Unified Modeling Language*, ühtne (unifitseeritud) modelleerimiskeel. Visuaalne (mudelid esitatakse diagrammidena e skeemidena) üldotstarbeline (erinevates valdkondades kasutatav) keel, mida kasutatakse eeskätt info- ja tarkavarsüsteemide struktuuri ning käitumise modelleerimiseks.
URL	*Uniform Resource Locator* ehk internetiaadress
XML	*Extensible Markup Language*, Keel mis defineerib reeglid mille alusel saab luua dokumente mis on nii inim- kui masinloetavad. [1]
XSS-rünnak	Saitideülene skriptimine on ründemeetod, mis võimaldab süstida veebisaidile pahatahtlikku koodi.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

“Eesti kultuuripoliitika eesmärk on kujundada loovust väärtustav ühiskond, hoides ja edendades eesti rahvuslikku identiteeti, uurides, talletades ja kandes edasi kultuurimälu ja luues soodsad tingimused elujõulise, avatud ning mitmekesise kultuuriruumi arenguks ja kultuuris osalemiseks.” [6]. Kultuuripoliitika eesmärkide täitmiseks on vaja infotehnoloogilisi süsteeme, mis võimaldavad kultuurimälu organiseerida ja kasutada, sh kõigile huvilistele võimalikult mugavalt kättesaadavaks teha.

1.1 Taust ja probleem

Eesti Kirjandusmuuseumi (edaspidi EKM) Eesti Rahvaluule Arhiivil (edaspidi ERA) puudub ühtne tarkvara rahvaviiside ning nende andmete hoidmiseks ja haldamiseks. Kogutud rahvaviisid – üleskirjutused ja helisalvestised – on suuremalt jaolt hoiul füüsilises arhiivis Tartus. Suur osa käsikirjalistest noodistustest ja heliarhiivist on digitaliseeritud, kuid seda hiiglaslikku, sadadesse tuhandetesse ühikutesse ulatuvat materjalikogu ei ole tänapäevaselt digitaalsel kujul korrastatud ega üldsusele kättesaadavaks tehtud. Rahvaviiside säilitamiseks ja kasutamiseks on oluline luua moodne infotehnoloogilisi vahendeid kasutav infosüsteem, mis toetaks pärimusmuusika levikut ja oleks avalikkusele kättesaadav. See infosüsteem peaks võimaldama hallata ja avaldada nii rahvaviise kui ka metaandmeid nende kohta ning võimaldama neid omavahel seostada.

Varasemalt on EKM selles osas teinud koostööd Tallinna Tehnikaülikooli bakalaureuseõppe üliõpilastega, kes analüüsisid ning arendasid esimese versiooni uuest Rahvaviiside infosüsteemist [7]. Tehtud töö käigus koguti ja esitati põhjalikult nõudeid ning loodi süsteemi esialgne versioon, mis on põhiliselt mõeldud rahvaviiside metaandmete salvestamiseks. CakePHP vahendit kasutades loodi rakenduse prototüüp, mis võimaldas kõikide andmete haldamist, kuid sundis peale EKM-i jaoks ebaoptimaalseid töövoogusid. Käesoleva töö autorid jõudsid seisukohale, et EKM-ile sobiva rakenduse loomiseks tuleks arvestada EKM-i töövoogudega juba platvormi valikul ning tarkvara esitluskihi ja lüüsikihi eraldamine võimaldaks paremini EKM-i

soovitud töövooge realiseerida. Seega selgus vajadus rakenduse platvorm välja vahetada ja sellest tulenevalt rakendus sellel uuesti realiseerida. Lisaks jäi prototüübis realiseerimata mitmeid funktsionaalsuseid, mis on vajalikud rakenduse laialdaseks kasutuselevõtuks. Ülevaade lisatavatest funktsionaalsustest on peatükis 6.

Ülesandepüstitus on koostatud EKM-i poolt. Töö tulemusena plaanitakse lahendada järgmised enne käesoleva töö tegemise algust süsteemis esinenud puudused.

- Puudub võimalus muuta mitut viisi korraga.
- Puudub võimalus lisada viiside juurde kodeeringuid.
- Puudub võimalus hoida viiside juures noodifaile.
- Puudub võimalus helifaile ette mängida.
- Puudub integratsioon või viitamise võime repositooriumile Kivike.
- Täiendamist vajab võimalus kasutajaliidest eri keeltesse tõlkida.

Antud töö tulemusena tekib uus keskkond rahvaviiside haldamiseks, mis võimaldab siduda olemasolevad üksteisega nõrgalt seotud infosüsteemid ja andmebaasid ning samuti käesoleva töö sisendiks oleva bakalaureusetöö tulemusena kavandatud andmebaasirakenduse. Eesmärk on kokku viia rahvaviiside alane mitmekülgne info: rahvaviiside käsikirjad, helisalvestused, laulutekstid ja metaandmed. Selle tulemusena muutuvad kergesti kättesaadavaks rahvalaulud kui viisist ja tekstist koosnevad tervikud koos metaandmetega. Oluline on ka muuta see info otsitavaks koos omavaheliste seostega. Teisene eesmärk on ka selle info kättesaadavaks tegemine laiemale avalikkusele, võimaldades arhiivis olevat informatsiooni kõikide huvilistega jagada.

Käesoleva lõputööga samaaegselt arendatakse ühes bakalaureuse lõputöös rahvaviiside otsingu funktsionaalsust, millele käesolev töö annab sisendit. Seetõttu on oluline, et valmiv lahendus võimaldaks hilisemate liidestuste lisamist. Tööd tehes peab arvestama, et arhiiv on erinevatele välistele osapooltele tulevikus uuringuobjektiks ning seetõttu on suur tõenäosus, et aja jooksul lisatakse sinna väliste komponentide näol täiendavat funktsionaalsust.

1.2 Töö kirjeldus

Töö peamine eesmärk oli realiseerida infosüsteem, mis võimaldab rahvaviise ja nendega seotud olulist infot korrastatud ja otsitaval kujul talletada ja kuvada, kasutades tänapäevaseid tehnoloogiaid ning praktikaid. Töö sisuks on nii olemasoleva funktsionaalsuse uuel platvormil realiseerimine kui ka süsteemi nii andmebaasi kui ka rakenduse osa laiendamine. Süsteemi andmebaasiplatvorm ei muutu – see on endiselt PostgreSQL. Realiseerimisega seotud otsuseid tehes jälgiti seda, et kasutatavad tarkvaralised lahendused ja standardid oleksid vabavaralised ja vastavalt EKM-i soovidele tasuta.

Antud töö raames loodi EKM-i soovitud rahvaviiside infosüsteemi tarkvara koos nende poolt soovitud funktsionaalsustega. Lisaks täiendati rakenduse dokumentatsiooni, et tulevikus lihtsustada täiendavaid arendusi ning tagamaks, et antud tarkvara vastaks tänapäeva nõuetele. Dokumenteeriti nii arenduse, infoturbe, autoriõiguste kui isikuandmete kaitse üldmääruse [8] ja sellest tuleneva Eesti isikuandmete kaitse seaduse [9] vaated. Loodud rakendus on mõeldud kasutamiseks nii organisatsiooni sisestele kui välistele kasutajatele.

Täiendavalt oli vajadus teisendada olemasolev noodistuste andmekogu selliseks, et seda oleks võimalik avalikkusega jagada ning mille põhjal oleks võimalik hiljem täiendavat analüüsi teha. Selle saavutamiseks oli vaja leida rahvaviiside noodistuste kodeerimiseks sobiv standard ning kogu olemasolev andmekogu teisendada vastavalt valitud standardile.

2 Metoodika

Käesolev peatükk annab ülevaate objektist, töö protsessist ja kasutatud tööriistadest.

2.1 Ülevaade objektist

EKM on teadus- ja arendusasutus, mis haldab eesti kultuuriloole olulisi arhiive, nende seas Eesti Rahvaluule Arhiivi (ERA). ERA ülesanneteks on pärimuskultuuri jäädvustamine, säilitamine ja uurimine ning ka selle kõikidele huvitatud osapooltele kättesaadavaks tegemine.

„ERA tegeleb eesti regilaulu, muinasjuttude, kohapärimuse, Eesti asunduste, tänapäeva- ja lastefolkloori, rahvausundi, etnomusikoloogia ning eesti folkloristika ajaloo uurimisega.“ [10] ERA-s töötavad teadurid teevad erinevaid uuringuid, mille tulemusena avaldatakse erinevat tüüpi teoseid nagu näiteks monograafe ja artiklite kogumikke. Lisaks on arhiivil oma digitaliseerimise stuudio. Selle eesmärk on toetada arhiivi erinevates digitaliseerimisega seotud küsimustes. [10]

Umbes 200 aasta jooksul on arhiivi kogunenud suurusjärgus 1.5 miljonit lehekülge käsikirju, üle 60 000 foto, üle 188 000 helisalvestuse ja 1800 videosalvestuse. Arhiivis hoiustatakse nii suulist kunstiloomet kui erinevaid vaimse rahvakultuuriga seotud teoseid. Arhiiv ei piirdu vaid eesti päritolu teostega vaid sisaldab ka erinevate rahvusvähemuste folkloori puudutavaid teoseid. Nende teoste originaalid on talletatud ERA kartoteekide süsteemis ning läbi digitaliseerimise on osa neist kättesaadavad ka digitaalselt läbi Kivikese infosüsteemi. [11]

Inimeste tegevustest säilivad erinevad teosed Neid teoseid tuleks pikaajaliselt säilitada, sest need on väärtuslikud. Näiteks annavad teosed võimaluse tulevikus uurida ja mõtestada seda, milline oli teoste loojate argipäev. Teoseid võib esitada erinevates keskkondades ja erineval viisil kodeerituna. Teose talletamise viisiks võib olla näiteks paberkandja, filmilint, foto, helisalvestis või digitaalne fail. [12]

Arhiivide eesmärgiks ei ole ainult erinevate arhivaalide kogumine, arhiveerimine ja avalikkusele kättesaadavaks tegemine. Tähtis on ka nende pikaajaline säilitamine ja konserveerimine. Kõige selle ülesandeks on tagada hinnalise teabe ja kultuuripärandi pikaajaline säilimine ja tulevastele põlvedele edastamine. [12]

Sellel ajal kui ERA allus ENSV Hariduse Rahvakomissariaadile oli arhiivi säilikutele ligipääs reguleeritud ja piiratud. Seda tehti peamiselt selleks, et kaitsta erinevate isikute, ettevõtete kuid peamiselt riigi huve. Samuti muudeti sellel ajal ajaloosäilikuid ning materjale viidi teistesse fondidesse, tekste kustutati jne. Arhivaalide puhul on tähtis nende võimalikult pikaajaline säilitamine selleks, et võimaldada uurijatel ja huvilistel säilikuid kasutada ja samas säilitada originaalteos võimalikult heas seisukorras. On tähtis, et originaalsäilikuid käsitletaks võimalikult vähe ning seda tehtaks vaid spetsiaalsetel säiliku pikaajalist säilimist tagavatel tingimustel. [13]

Paraku on Eesti arhiivides kultuuripärandi digitaliseerimisele lähenetud üldiselt projektipõhiselt. 2021. aasta kevadel on käimas viis aastat kestev projekt “Kultuuripärandi digiteerimine 2018-2023". Selle eesmärgiks on kultuuripärandi digitaliseerimine ressursse säästvalt ja digitaliseeritud pärandi ühtlustatud säilitamine ning avaliku juurdepääsu tagamine. [14]

Eelnevast on näha, et arhiivide sisu digitaalselt kättesaadavaks tegemine on tähtis. Lisaks arhivaalide säilitamisele on oluline see, kuidas võimaldada digitaliseeritud andmetele ligipääsu. Säilikute digitaliseerimisel ning analüüsil on võimalik luua kodeeritud informatsiooni, mida siis saab siduda digitaliseeritud säilikuga. Sellist tüüpi kodeeritud informatsiooni on võimalik kasutada huvipakkuvate säilikute leidmiseks ning erinevateks analüüsideks.

Töö tulemusena uuel viisil kodeeritavad viisid on suuremalt jaolt vanemate rahvalaulude ehk regilaulude viisid, millest suurem osa on kogutud 19. ja 20. sajandil. Laulude esitus oli kohati kõnelähedane, neid võidi esitada nii retsiteerides ehk poollauldes kui ka uuemal kombel diatoonilises helistikus lauldes.

2.2 Ülevaade töö protsessist

Analüüsi aluseks on võetud olemasolev rahvaviiside haldamise rakendus ja seda selgitav bakalaureusetöö [7], mille tulemusel antud rakendus loodi. Lisaks saadi tellijalt nimekiri

soovidega selle kohta, millist funktsionaalsust soovitakse kasutada ning kuidas on siiani vastavaid ülesandeid lahendatud. Koostöös ja kooskõlas EKM-iga loodi lõpptulemuse visioon ning selle realiseerimisel kasutati võimalikult palju juba loodud analüüsi ja realisatsiooni, soovides pakkuda maksimaalselt lisaväärtust just uute funktsionaalsuste ja võimaluste lisamise läbi, ilma juba tehtud tööd kordamata.

Arendusmeetodina kasutati Scrum raamistikku. Tegemist on paindmetoodikaga, mis aitab meeskondadel struktureeritult tööd planeerida ja selle arengut jälgida. Scrumi peamine eesmärk on kogemuste kaudu õppimine läbi probleemide lahendamise ja läbi selle pidev arenemine. [15] Scrumi üks põhimõtetest on enesejuhtimine. See tähendab, et iga meeskonna liige vastutab temale määratud ülesannete tähtsuse järjekorra määramise eest. Selline lähenemine suurendab meeskonna iseseisvust, mis omakorda tõstab meeskonnaliikmete üldist tööviljakust. [16] Antud põhimõte iseloomustab hästi käesoleva töö autorite tegutsemist, sest meeskonna liikmed asuvad eri linnades. Kogu töö kirjutamise vältel toimus regulaarne omavaheline suhtlus, mille käigus leiti probleemidele ühiselt lahendusi. Igal esmaspäeva hommikul toimus retrospektiiv, milleks oli Skype vahendusel kõne, mille raames räägiti mis eelneval nädalal toimus, millised olid edusammud, takistused ja lahendamata probleemid. Samuti seati sellel kohtumisel eesmärke järgmiseks nädalaks.

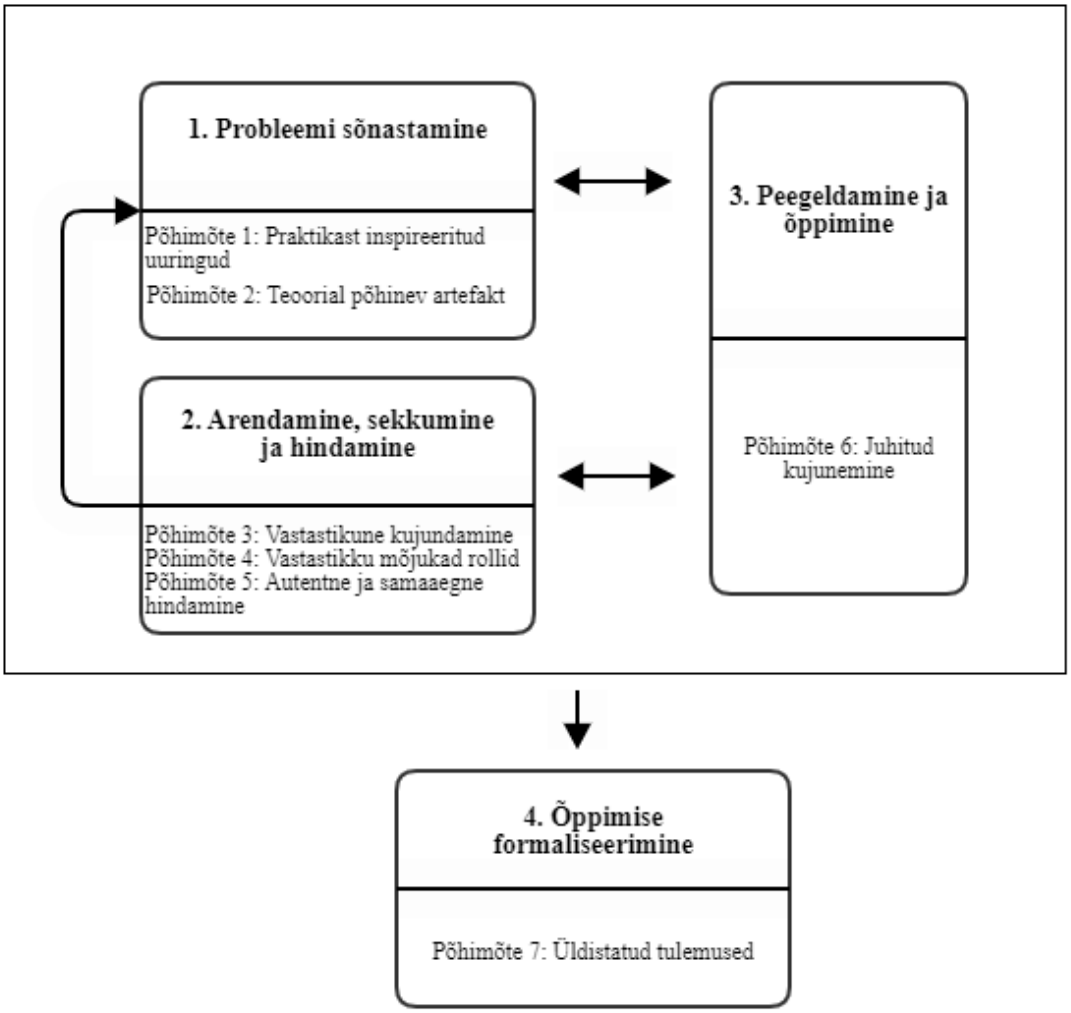
Töö näol on tegemist disaini tegevusuuringuga (ADR ehk *action design research*). [17] Töö eesmärgiks on luua tehnilisi artefakte (töötav tarkvara ja selle dokumentatsioon). Eesmärgi saavutamiseks toimub pidev arendus ning paralleelselt üritatakse selle tulemust kasutusele võtta. Jälgitakse kasutajate tööd ning nende tagasisidega arvestatakse järgmiste iteratsioonide arendamisel. Hindamine toimub arendamisega samaaegselt ning selle tulemused mõjutavad arendust.

2.2.1 Disaini tegevusuuring

Disaini tegevusuuring on uurimismeetod ettekirjutava disainiteadmise loomiseks, luues ja hinnates IT tehiseid e artefakte organisatsioonilises keskkonnas. See keskendub kahele väljakutsele.

1. Mingis konkreetses organisatsioonilises olukorras esinenud probleemide lahendamine.
2. Probleemidega tegelemiseks mõeldud IT tehise loomine ja hindamine.

Nende kahe väljakutse nõutavad vastused annavad meetodi, mis keskendub tehise ehitamisele, sekkumisele ja hindamisele. [18]



Joonis 1 Disaini tegevusringi protsess.

ADR meetod sisaldab kindlaid etappe (Joonis 1):

1. **Etapp 1: probleemi sõnastamine.** Esimese etapi käivitajaks on probleemi teadvustamine, mis annab tõuke uurimisprobleemi sõnastamiseks (vt jaotis 2.2.2). Etapi tulemusel määratakse kindlaks probleemi ulatus. Ühe näitena kriitilisest lahendamist vajavast küsimusest võib tuua osalevate organisatsioonide pikaajalise pühendumuse tagamise edasise uurimise käigus. Lisaks määratakse antud etapis kindlaks uurimisrühma liikmete rollid, vastutus ja üleüldse projekti skoop. [17]
   Esimeses etapis tuleb järgida järgnevaid põhimõtteid.
   Põhimõte 1: Praktikast inspireeritud uuringud – nõuab probleemide vaatlemise kui teadmiste loomise võimaluse kasutamist. Disaini tegevusuuring

otsib võimalusi tehnoloogiliste ja organisatsiooniliste valdkondade ristumiskohas. Ühe konkreetse organisatsiooni probleemide lahendamisele lisaks tuleb luua teadmisi, mida saab probleemide klassi lahendamiseks rakendada paljudes erinevates olukordades ja organisatsioonides.

Põhimõte 2: Teoorial põhinev artefakt – loodav artefakt peab probleemi püstitamisel sõnastatud probleeme lahendama ja tegema seda valdkonna teoreetiliste teadmistega arvestades.

2. **Etapp 2: arendamine, sekkumine ja hindamine.** Teises etapis kasutatakse eelmises etapis sõnastatud probleemide kirjeldusi, et luua lahenduse esimene versioon. Selle etapi tulemus on artefakti realisatsioon. Selle etapi jooksul toimub jooksvalt pidev hindamine ja artefakti ümberkujundamine. Lisaks selgub ka artefakti kavandamisest või organisatsioonilisest sekkumisest tuleneva innovatsiooni mõju. [17] Etappi vaadeldakse nii tehnikatele kui ja organisatsioonile orienteeritult.

   Põhimõte 3: Vastastikune kujundamine – selle põhimõtte kohaselt mõjutavad IT artefakt ja selle organisatsiooniline kasutuselevõtt vastastikku üksteise arengut.

   Põhimõte 4: Vastastikku mõjukad rollid – see tähendab, et projektis osalejad õpivad teineteiselt. Disaini tegevusuuringu läbiviijad toovad oma teadmised teooriast ja praktikud lisavad enda vaatenurgast praktilisi teadmisi. Erinevad vaatenurgad võivad üksteisega konkureerida või hoopis üksteist täiendada.

   Põhimõte 5: Autentne ja samaaegne hindamine – tegutsetakse paralleelselt. See tähendab, et artefakti hindamine toimub selle disainimisega (tehnilise kavandamisega) ja realiseerimisega samal ajal, st see ei ole ehitamisele järgnev eraldi etapp.

3. **Etapp 3: Peegeldamine ja õppimine**

   Antud etapis toimuvad tegevused samaaegselt eelnimetatud etappidega. Tähelepanu on suunatud eksperdi poolt püstitatud algsele disainile ja paralleelselt ka organisatsiooni ümberkujundamise protsessile. Loodav artefakt peab olema mõlemaga kooskõlas. [19]

   Põhimõte 6: Juhitud kujunemine – loodav artefakt peab arvestama nii uurija loodud algset disaini kui ka paralleelselt toimuva organisatsiooni ümberkujundamise protsessi tulemustega.

4. **Etapp 4: Õppimise formaliseerimine -** Neljandas ja viimases etapis toimub õppimise tulemusel saadud teadmisete formaliseerimine. Need viiakse kindlast

kontekstist saadud keskkonnast üle üldisemale tasemele, kus õpitut saab rakendada ka üldisemat tüüpi probleemide jaoks.
Põhimõte 7: Üldistatud tulemused – organisatsioonispetsiifilised lahendused viiakse üle üldisemale tasemele.

2.2.2 Disaini tegevusuuringu rakendamisest käesolevas töös

Töö koostamisel lähtuti läbivalt disaini tegevusuuringu etappidest.

Palju aega kulus EKM-i kui organisatsiooni tundmaõppimisele ja muusikalise tausta uurimisele. Esmalt keskendusid töö autorid probleemide mõistmiseks vajalikele baasteadmiste tekitamisele. Seejärel kaardistati lahendamist vajavad probleemid, tutvuti organisatsiooni eripäradega, pandi kirja EKM-i soove ja ootusi ning loodi esmane visiooni projekti lõpptulemusest. Töö autoritest kaks on varasemalt muusikaga kokku puutunud muusikakoolis ja pilliõppes. Kokkupuude oli viimati 15 aastat tagasi. Seetõttu oli tarvis muusikateooriat uuesti meelde tuletada.

Kaasatud osapoolteks olid EKM spetsialistid, kelleks olid digitaalarhivaar Olga Ivaškevitš ja vanemteadur Taive Särg, kes võtsid disainimise protsessist aktiivselt osa. Lisaks kaasati ka üks käesoleva töö sisendiks oleva bakalaureuse lõputöö koostajatest – Kristi Seemen, kes oli juba süsteemi ja eesti rahvaviisidega tuttav.

Uurimisrühma arvates oli peamiseks probleemiks eelnevalt realiseeritud lahenduse ebapiisavus, sest noodistuste kuvamiseks, jagamiseks ja noodistuste haldamiseks puudus süsteemis piisav võimekus. Esimene versioon disainist koostati organisatsiooni igapäevatöö praktikast lähtudes. Peale seda kui EKM-i soovid ja valdkonnaspetsiifilised nõuded olid leitud, sõnastati peamised eesmärgid ja uurimisküsimused. Ülesandepüstituse lõpliku sõnastuse kokku leppimine osutus aga keerukaks kuna eri osapooltel tekkis eriarvamusi ja soovide lahknevust. Samuti oli vaja pidevalt jälgida uue infosüsteemi arendamise skoobi vastavust lähteülesandele. Valiti välja esmased tööriistad, mida probleemide lahendamiseks kasutada.

Lähtudes uue süsteemi ideest ja visioonist alustati rakenduse loomist. Et tagada rakenduse parim võimalik realisatsioon, mis vastaks tellija soovidele, peeti iganädalaselt kõikide osapooltega koosolek, kus hinnati tehtut ja otsustati, mis suunas jätkata. Soovide

muutuste tõttu, oli tarvis ka esmaselt välja pakutud disaini muutmine. Näiteks muutusid aja jooksul prioriteedid, mis järjekorras soove realiseerida.

Järgnevalt esitatakse soovide muutumise näide. Esimestel koosolekutel räägiti sellest, et PID (PID ehk arhivaali unikaalne identifikaator Kivikese allsüsteemis) on rahvaviiside digitaliseerimise juures väga tähtis komponent. Igal viisil pidi olema hallatav PID, mis peab olema koos viisiga kuvatud. Mitu kuud hiljem, kui töö autorid tutvustasid viisi muutmise funktsionaalsust, tekkis EKM töötajatel küsimus, miks PID viisi juures muudetav on, lisades juurde, et see peab olema tarkvara poolt genereeritud, mitte kasutaja hallatav väärtus.

Viimaks toimub lõpptulemuse formuleerimine. Loodud süsteem antakse EKM-ile üle koos lähtekoodi ja andmebaasi importfailidega. Töö tulemusena lahendati üheltpoolt konkreetse organisatsiooni probleem. Teisalt jõuti ka tulemusteni, mida saab kasutada edasiste arenduste baasina.

2.3 Tööriistade kirjeldus

Informatsiooni salvestamiseks ja failide hoidmiseks kasutati Microsoft OneDrive pilveteenust. See võimaldas erinevatest seadmetest tööle ja materjalidele ligi pääseda ja neid täiendada. Microsoft Onedrive võimaldab mitmel kasutajal jagatavaid dokumente samaaegselt täiendada. Faili alla laadides saab seda otse muuta Microsoft Wordis. Teised kasutajad näevad reaalajas tehtavaid muudatusi ja saavad teisest seadmest samaaegselt dokumenti muuta. Microsoft Wordis toimetamine säilitab dokumendi vorminduse ning kogu dokumendile tehtavate muudatuste ajalugu. Tänu sellele on võimalik vältida juhuslikke andmekadusid ning vajadusel taastada dokumendist eelnev seis.

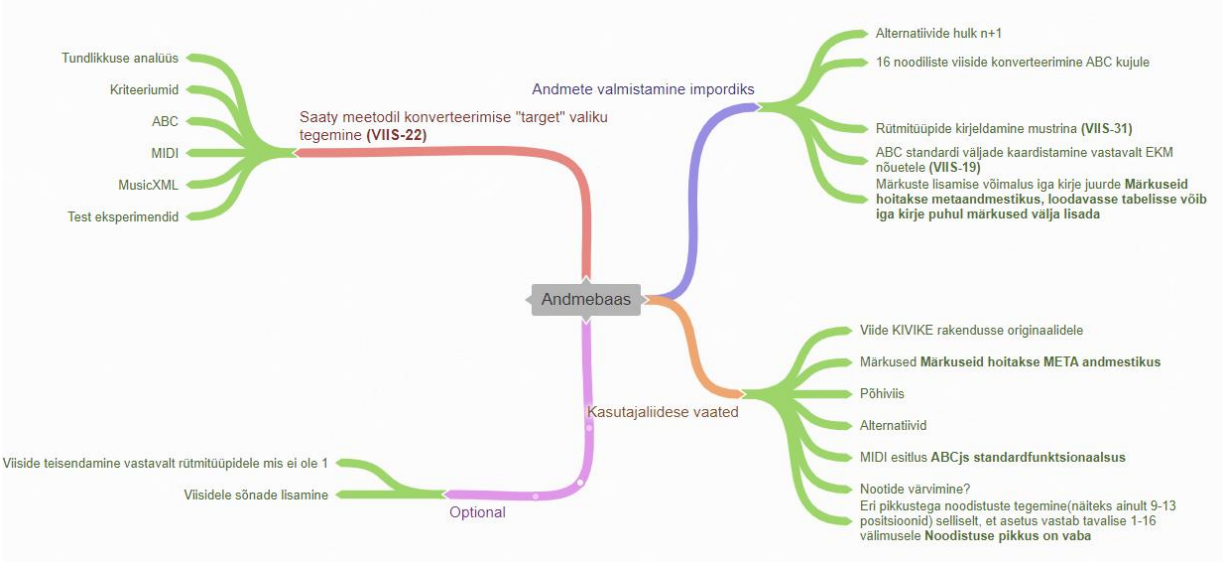
Omavaheliseks suhtlemiseks ja koosolekute pidamiseks oli kasutusel Microsoft Skype . Skype kasuks otsustati kuna töö teostajad olid magistriõppe algusest saadik seda info jagamiseks kasutanud. Lisaks pakkus see vahend võimalust omavahel faile jagada ning audio- ja videokõnesid pidada, mis oli omavahelise info vahetamise jaoks tähtis.

Tellijaga tehti videokõnesid Google Meet videosideteenust kasutades. See lahendus võeti üle eelmiselt meeskonnalt, kes samamoodi EKM-iga aktiivselt koosolekuid pidas. Koosolekutest eraldi protokolli ei peetud, kuid lisaks koosolekutele kasutati e-kirju passiivseks suhtlemiseks EKM ja antud töö autorite vahel. Koosolekuid EKM-i

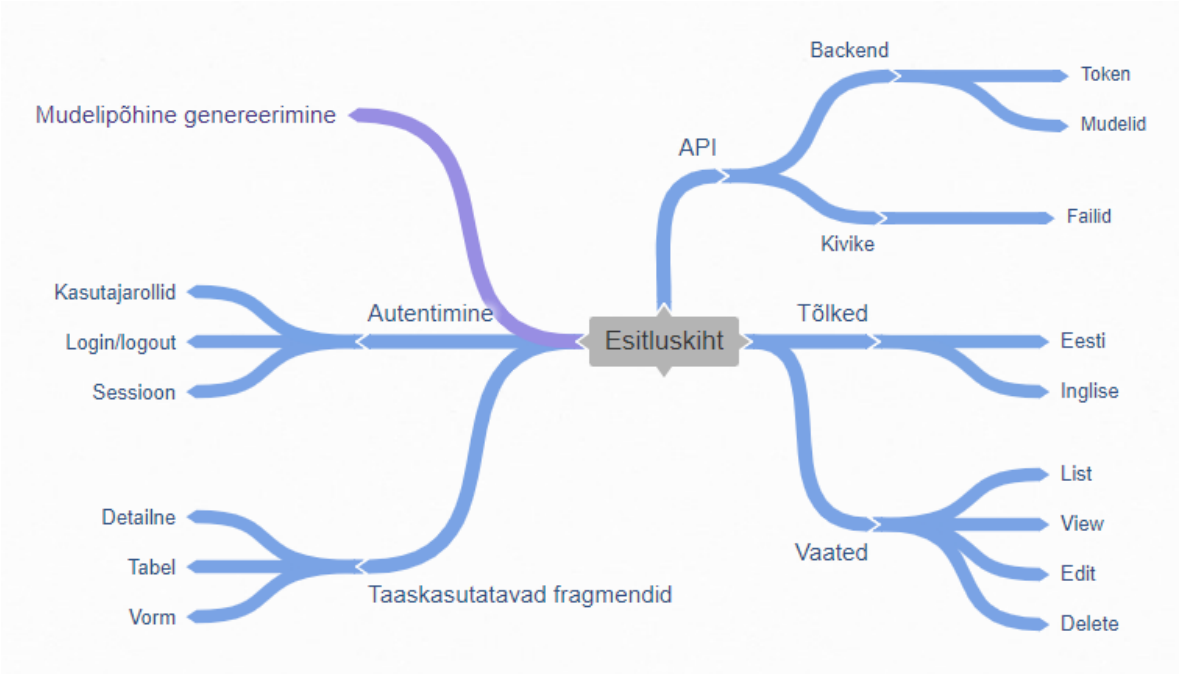
töögrupiga peeti iganädalaselt neljapäeviti. Selle käigus jagati informatsiooni projekti kulgemise kohta ning vahetati mõtteid erinevate töö käigus tekkinud küsimuste osas. Koosolekutel võeti vastu otsuseid edasiste tegevuste ja arendussuundade osas. Eraldiseisva jagatud infohoidla järgi puudus vajadus. Ühistöödeks, nagu EKM viisi andmebaasi noodistuste kodeerimine ja metaandmestiku parendamine, kasutati Google Sheets rakendust, milles hallati nii mustandeid kui ka lõplikku puhtandit.

Suhtluseks juhendajaga kasutati MS Teams tarkvara. Teamsi kohtumistel vaadati korduvalt lõputööd üle ja defineeriti edasised suunad, kuidas tööga edasi minna. MS Teams võimaldas ekraanijagamist, mis lihtsustas ühiselt dokumendiga tutvumist.

Ideede ja mõtete kaardistamiseks kasutati vabavaralist rakendust nimega Coggle. Selle eeliseks võrreldes Google Docsiga on idee visualiseerimine mõttepuuna. Töö planeerimise algfaasis mõeldi läbi töö sisu ja leiti selle erinevad valdkonnad. Näiteks loodi mõttepuu andmebaasi osadest (Joonis 2) ja esitluskihi osadest (Joonis 3). See aitas anda ülevaate sellest, mida peab realiseerima. Vastavalt vajadusele tehti mõttepuid ümber – lisati ja muudeti harusid.



Joonis 2 Väljavõte coggle.it mõttekaardilt.



Joonis 3 Esitluskihi kuva coggle.it mõttekaardilt.

Süsteemi visuaalseks modelleerimiseks kasutatakse ühtset modelleerimiskeelt (Unified Modelling Language), lühendiga UML. Visualiseerimiseks kasutasime tegevusdiagramme, mis kirjeldavad protsesse, tuues välja toimingute järjekorra ning otsustuspunktid. Diagrammide loomiseks kasutati tasuta rakendust diagrams.net. [20] ja tasulist rakendust Gliffy. [21]

Arendustöö käigus on vaja teha otsuseid – näiteks milliseid vahendeid või metoodikaid kasutada. Antud töös kasutati nootide kodeerimise standardi valimiseks Saaty meetodit (*Analytic Hierarchy Process, AHP*) [22]. Tegemist on hierarhilise analüüsimeetodiga, mis aitab süstemaatiliselt leida alternatiivide hulgast parima võimaliku lahenduse. Meetodi loojaks on matemaatik Thomas L. Saaty. Peamiselt kasutatakse seda meetodit otsustusprotsesside toetamiseks. See aitab teha keerukaid otsuseid ja muuta neid ratsionaalsemaks, pakkudes võimalust hinnata alternatiive mitme kriteeriumi alusel. Meetod põhineb kriteeriumite paariviisilistel võrdlustel ja alternatiivide paariviisilistel võrdlustel kriteeriumite suhtes. Saaty meetodi abil otsuste tegemiseks on saadaval erinevaid programme. Käesolevas töös kasutatakse tasuta programmi Web Hipre. [23]

Rakenduse lähtekoodi hoiustamiseks kasutatakse Github koodirepositooriumit, läbi mille toimus omavaheline koodibaasi sünkroniseerimine ning talletamine. Täiendavalt kasutati sealset versioonikontrolli. Lähtekoodi lokaalses masinas kompileerimiseks ning edukaks

käitamiseks on vajalikud järgnevad komponendid: lähtekood, PostgreSQL andmebaasisüsteem, Node.js mootor. Lihtsuse ning ühtse koodistiili hoidmiseks otsustati, et kasutatakse koodiredaktorit Visual Studio Code [24], mille jaoks loodi soovitatavate pistikute nimekiri ning nende sätted. Pistikute nimekirja ning sätteid hoiti koos lähtekoodiga kaustas *.vscode*, mille koodiredaktor VS Code automaatselt ära tundis ning seejärel vastavaid pistikuid installeerida soovitas. Pistikute installeerimisel rakendusid vastavad sätted automaatselt.

Koodi kompileerimise ning rakenduse käivitamise lihtsustamiseks loodi käskluste jada, mis ühendati kokku *npm start* skripti. Selle skripti käsurealt käitamisel kas esitluskihi või lüüsikihi kaustas kompileeritakse kood ning käivitatakse vastav rakendus automaatselt.

Rakenduse avalikult kättesaadavasse serverisse ülekandmiseks loodi lüüsikihile ja esitluskihile *Dockerfile*, mis sisaldab endas juhiseid rakenduse kompileerimiseks. Docker konteineri tõmmis ehitatakse ning laetakse Docker Hub keskkonda käsuga *npm run docker:release*, kust seejärel on võimalik seda alla laadida serverisse ning konteinerina käitada.

3 Teoreetiline taust

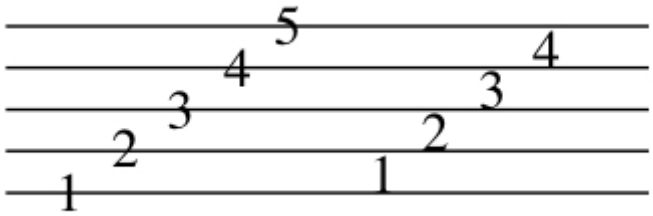
Selles peatükis tutvustatakse antud töös kasutatavaid mõisteid, mille tundmine on rahvaviiside infosüsteemi täienduste tegemiseks ja ka käesoleva kirjatöö mõistmiseks vajalik.

3.1 Muusika kirjeldamine noodikirjas

Läänelikus kultuuris kasutatakse muusika üleskirjutamiseks ühtset märgisüsteemi ehk noodikirja. Noodikiri võimaldab heliloojal edastada teavet oma teose kohta. See on oluline muusikule, kes esitab teiste autorite teoseid. Lisaks sisaldub selles teavet, milliseid noote mängida ja kui kiiresti või aeglaselt seda teha ning informatsiooni dünaamika ja tämbri kohta. Samas ei anna ükski noodikiri edasi kogu informatsiooni esituse kohta, vaid see kandub edasi elavas traditsioonis.

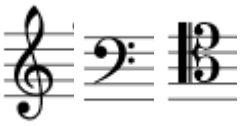
Charles Seeger, Ameerika Ühendriikide muusikateadlane, helilooja ja folklorist, jagas noodikirja kaheks: preskriptiivne ja deskriptiivne noodikiri. Esimene on suunatud muusika esitajale (solistile, dirigendile jt) ja annab juhiseid kuidas tuleks teost esitada, eeldusel, et esitaja on teoste stiiliga tuttav. Teine noodikirja tüüp, ehk deskriptiivne noodikiri, edastab detailsemalt muusikateose iseloomu. [25]

Noodikiri koosneb sümbolitest ja märgistustest, mis aitavad muusikuid kompositsiooni esitamisel seda võimalikult täpselt teha. Leidub erinevaid noodistusi, millest kõige klassikalisem nootide esitus on noodid noodijoonestikul (Joonis 4). Noodijoonestik koosneb viiest üksteise kohal asuvast horisontaalsest joonest ja neljast joonevahest. Jooned ja joonevahed tähistavad alushelide asukohti (C, D, E, F, G). Noodid, mis ulatuvad väljapoole noodijoonestikust, märgitakse abijoontele noodijoonestiku kohale või alla. [26]



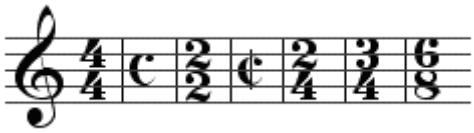
Joonis 4 Klassikaline noodijoonestik.

Teose täpsele helikõrgusele viitab iga noodijoonestiku alguses asuv noodivõti. Põhiliselt kasutatakse kolme erinevat noodivõtit, milleks on G-, F- ja C-võti (Joonis 5). Kui muusikateose vältel helikõrgus muutub, siis kirjutatakse sellesse kohta uus võti [26]



Joonis 5 G-, F- ja C- noodivõti.

Noodivõtme järele lisatakse taktimõõt. Takt on teose ajalise struktuuri osa, mis eristatakse üksteisest püstjoontega ehk taktijoontega (Joonis 6). Taktimõõt on esimesel noodijoonestikul. Kui noodivõti märgitakse iga noodistiku algusesse, siis taktiga nii ei tehta. Reeglina koosneb taktimõõt kahest arvust. Ülemine viitab löökide arvule taktis, alumine löögile vastavat vältust. [23]

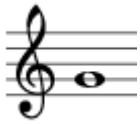
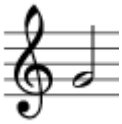
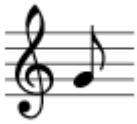
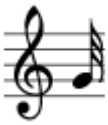
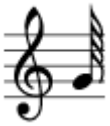


Joonis 6 Näited taktimõõtudest, mis asetsevad taktijoonte vahel.

Topelt taktijoont kasutatakse muusikateose osade eraldamiseks. Topelt joon noodistiku lõpus tähistab teose lõppu.

Noot on märgistus, mis paikneb noodijoonestikul ja näitab heliastme kõrgust ja vältust. Noot koosneb neljast osast – noodipea, noodivars, noodilipud ja talad. Heli pikkust ehk vältust näitab noodi kuju (Tabel 1).

Tabel 1 Noodipikkused.

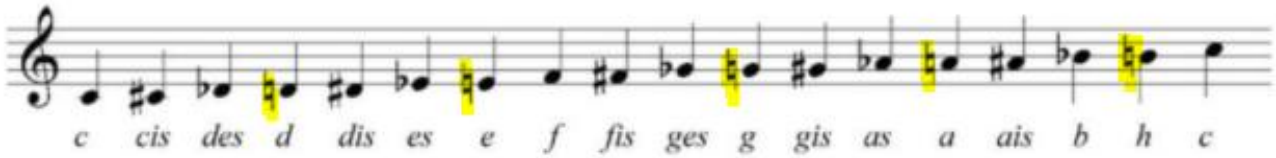
	tervenoot
	poolnoot
	veerandnoot
	kaheksandiknoot
	Kuueteistkümnendiknoot
	kolmekümnekahendiknoot
	kuuekümneneljandiknoot

Kui heli puudub, siis on tegemist pausiga. Sarnaselt nootidele on ka pauside tähistustel erinevad kujud ja pikkused. Pausi kestvused on tähistatud erinevate märkidega. Nootide loogikat jätkates on olemas täispaus, poolpaus, veerandpaus, kaheksandikpaus, kuueteistkümnendikpaus, kolmekümnekahendikpaus ja kuuekümneneljandikpaus.

Pauside sümbolid on väljatoodud Joonis 7. [27]

Joonis 7 Pausid noodistikul.

Heli muutmine ehk altereerimine tähistab heli kõrguse muutmist poole- või täistooni võrra. Heli kõrgendamist tähistatakse noodi ette asetatud dieesiga, madaldamist tähistatakse bemolliga Heli alteratsiooni tühistamist märgitakse bekaariga (Joonis 8). Heli kõrgendamisel lisatakse tähtnimetuste puhul liide –is (näiteks dis, cis, gis jne), madaldamise puhul aga (e)s (näiteks des, ges, as jne). Heli h on erandiks – seda tähistatakse pooletoonilisel madaldamisel tähisega b.



Joonis 8 Heli kõrgendamine ja madaldamine.

Tempo on heliteose esitamise kiirus, mille tähistamiseks kasutatakse kahte lähenemist. [28]

- Itaaliakeelseid tempotähiseid, mis väljendavad kujundlikult teose esitamise tempot nagu *largo, adagio, andante, presto, allegro* jne. Näiteks, *largo* tähistab väga, väga aeglast muusikapala esitamist, *allegro* aga kiiret, liikuvat esitamist.

- Löökide arvu minutis, mille puhul määratakse teose esitamise eeldatav kiirus täpselt. Selliselt muusika esitamiseks kasutatakse täpselt õige tempo saavutamiseks üldiselt metronoomi.

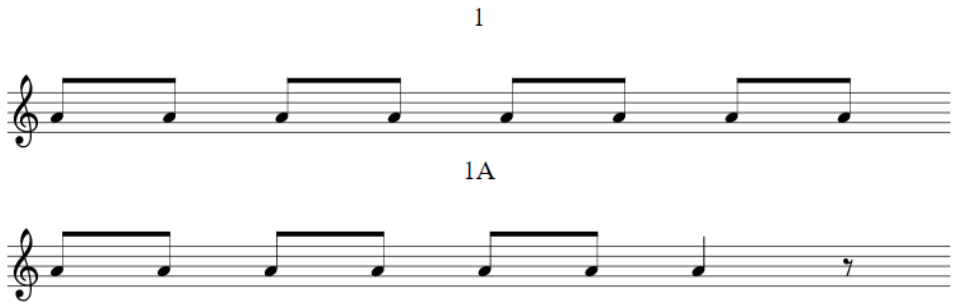
EKM-ilt saadud lähteandmetes kasutati läbisegi mõlemat liiki lähenemist. Oli noodistusi, millele oli märgitud tempo numbriliselt peale, kui ka selliseid kuhu oli kirjutatud kujundlik tempo esitamise kiirus nagu näiteks “Ruttu”, “Aeglaselt” või “Alleg”.

3.2 Eesti Kirjandusmuuseumi rahvaviiside kodeerimise põhimõtted.

Eesti Kirjandusmuuseumi (EKM) Eesti Rahvaluule Arhiivil (ERA) on ulatuslik folkloorikogu, kuhu on talletatud Eesti rahvaviise erineval kujul. Arhiivis on kirjeid nii helifailidena kui käsikirjadena. Selleks, et neid kodeerida ja käsikirjalisi noodistusi kuulatavaks muuta, on vaja leida viis kuidas andmeid salvestada selliselt, et kasutusele võetav lahendus oleks rahvamuusika eripäradega arvestades piisavalt paindlik. Läänelik klassikaline muusika on arenenud teatud reeglitele vastavalt ja muutunud mõnevõrra jäigaks osaliselt seetõttu, et nootidega kirjutatud muusika peab vastama kindlale struktuurile ja põhimõtetele. Eri rahvaste rahvamuusika on aga arenenud sõltumata läänelikust muusikatraditsioonist ja sellele vastavast noodikirjast. Seega ei ole rahvamuusikat sageli võimalik suruda klassikalise muusika helikeelele vastava noodikirja raamidesse. Seetõttu on vajalik leida võimalus kirjeldada muusikat nii nagu see esitati, milleks kasutatakse tavaliselt täiendatud läänelikku noodikirja.

Praeguseks on EKM kodeerinud juba pea 5000 viisi ning kasutuses on üle 60 erineva rütmitüübi. Lisaks on tekkinud soov uues loodavas lahenduses kodeerida viisid vastavalt nende tegelikule kujule vajaduseta jäigalt rütmitüüpi jälgida. Rütmitüübid säilivad viisi kodeeringute juures kui metaandmed, mida ei kasutata noodistuse loomiseks ega võrdlemiseks noodistuse vastamise osas rütmitüübile.

Rütmitüübiks loetakse noodistuse kuju, mis määrab nootide ajalised tunnused. Rütmitüübi osaks ei ole aga üksikute nootide kõrgused. Selle tõttu on edaspidi rütmitüüpide näidete juures kõik noodistatud selliselt, et iga noot vastab helikõrgusele A (Joonis 9).

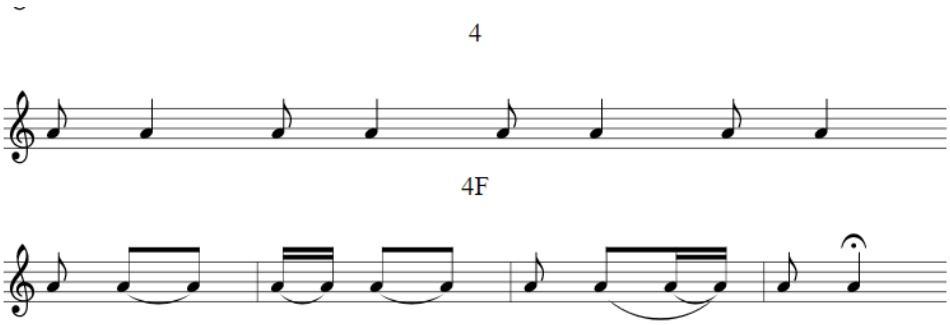


Joonis 9 Näide rütmitüübist.

Olemasolev MS Access andmebaas (Tabel 7 ja Lisa 2), mis sisaldab antud töö skoobiks olevaid viise, sisaldab andmeid, mis on suures osas kodeeritud vastavalt I. Rüütli välja

töötatud mudelile [29]. Seal on kirjeldatud kaherealised, 16 kohalised viisid ning iga viisi kohta on kirjeldatud selle rütmitüüp. Statistiliselt oli suurima esinemiste arvuga rütmitüüp "rütmitüüp 1", mida oli antud valimis 85% kirjetest. Kui siia juurde lisada mitmest rütmitüübist koosnevad viisid (erinevate rütmitüüpide kombinatsioonid), siis on protsent veelgi suurem.

Üheks olemasoleva andmebaasi puuduseks oli andmebaasi tabelite veergude piiratud arv. Kuna andmebaasi struktuurid loodi vaid kas 8- või 16- kohaliste viisidega arvestades, siis ei võimaldanud see salvestada erineva või muutuva pikkustega viise. Rütmitüüpide puhul, mis olid teistsuguse pikkusega, kasutati loovat lähenemist, kombineerides ühele väljale mitu nooti või loobuti teatud nootide kodeerimisest. Seetõttu ei olnud olemasolev lahendus piisav kogu arhiivis olevat viisimaterjali kodeerimiseks. Rütmitüübid jäävad loodava lahenduse puhul küll alles metaandmetena, kuid otsene sõltuvus noodistusest kaob, ehk rütmitüübi muutmisel noodistus ei muutu.



Joonis 10 Näide rütmitüübist ja selle variatsioonist.

Näitena rütmitüübist ja selle alamtüübist, mis olemasolevasse Accessi-põhisesse lahendusse ei mahu, on rütmitüüp 4 ja selle variatsioon 4F mis on nootide hulgalt väga erineva pikkusega (Joonis 10). Kui rütmitüüp 4 koosneb kaheksast noodist, siis rütmitüüp 4F koosneb juba 13 noodist.

Andmete kodeerimisel MS Access andmebaasis kasutati I. Rüütli koostatud rahvaviiside noodistuse formaliseerimise põhimõtteid. *„Meloodia kodeeritakse tähtnimetuses vastava vormi järgi, kus igal meetrilisel positsioonil on oma väli. 1. oktaavi ei märgita, 2. oktaavi iga noodi ees seisab tähis 2, väikese oktaavi puhul -, suurel oktaavil –2. Kui üht positsiooni täidab kaks või enam noodi (nn silbi-jaotused), eraldatakse need kaldjoonega /. Samas positsioonis samal helikõrgusel korduvaid juhuslikke lisahelisid ei märgita"*

[29]. Autorimärkusena märgime juurde selle, et reaalses andmebaasis oli väiksema oktavi puhul ees tähis 1.

Selleks, et luua uus lahendus, mis ei oleks enam piiratud olemasoleva lahenduse piirangutega, tuli esmalt leida nootide kodeerimise standard, mille alusel viise kodeerida. Hetkel kasutatakse noodistuste koostamiseks Finale nimelist tarkvara [30], mis võimaldab EKM-i töötajatel kodeerida erinevaid noodistusi ning teha kujundust (eripikkustega noodiread, sõnade/pealkirjade/viidete lisamist) vastavalt soovile. Töö tegemise hetkel paistis tarkvara vajadustele vastavat. Puuduseks aga on see, et tegemist on tasulise tarkvaraga ning failiformaat, milles Finale noodistusi salvestab, ei ole ilma selle tarkvarata kasutatav.

3.3 Alternatiivsete noodistuste kodeeringute ülevaade

Alternatiividena uuriti võimalusi eksportida Finalest noodistust erinevatele kujudele. Finale toetab ametlikult nii MusicXML kui MIDI formaati, mis mõlemad on avatud standardid ning, mis võimaldavad noodistusi avada ning uurida ilma tasulist tarkvara kasutamata. Lisaks nendele lisati võrdlusesse ka avatud standard ABC notatsioon , mis võimaldab hõlpsasti muusikat teksti kujul kodeerida. MusicXML <=> ABC teisenduseks on võimalik kirjutada programm või võtta teisenduse tegemiseks kasutusele mõni vabalt kasutatav programm. Sama kehtib ka MIDI <=>ABC ja MIDI<=>MusicXML teisenduste puhul. Seega võib selles võtmes lugeda neid kodeeringuid samaväärseteks. Valimist jäi välja NIFF failitüüp kuna tegemist on binaarse formaadiga, mis vajab kasutamiseks eraldi rakendust ning on raskemini loetav kui tekstipõhine kodeering. Samuti jäi vaatluse alt välja Lilypond kodeering, mis pole väga laialdaselt levinud. [31]

3.3.1 MusicXML

MusicXML puhul on tegemist XML baasil failiformaadiga, mis on mõeldud muusikalise noodistuse kirjeldamiseks. Tegemist on avatud, põhjalikult dokumenteeritud ja vabalt kasutatava standardiga, mis on litsentseeritud “W3C Community Final Specification Agreement” litsentsiga. [32] Antud standardi arendamisega tegeles “W3C Music Notation Community Group”. [33] Töö kirjutamise hetkel (2021. aasta kevad) on antud standardi viimane versioon 3.1, mis avalikustati 2017. aasta detsembris. MusicXML-i standardi esimene versioon 1.0 avalikustati 2004. aastal, mis näitab selgelt standardi olulisena püsimist. MusicXML standardit toetavad mitmed erinevad noodistuse

töötlemise programmid, kaasa arvatud Finale, mis on EKM-i puhul üks põhilistest kodeerimise rakendustest. MusicXML, nagu kõik XML-il baseeruvad formaadid, on inim- ja masinloetavad, ning kuigi see pole mõeldud tekstiredaktoriga kirjutamiseks, on sellest võimalik ilma eritarkvara kasutamata lugeda selliseid metaandmeid nagu näiteks märksõnad. Antud projekti skoobis olid põhiliselt küll noodistused kus sõnu ja silpe ei viidud noodistustega kokku, kuid sellise võimalikkuse olemasolu on EKM-ile tulevaste arhivaalide digitaliseerimise puhul tähtis.

3.3.2 MIDI

MIDI on 1980. aastate alguses loodud ja 1982. aastal määratletud andmevahetuse (-edastuse) protokoll (keel), mis võimaldab elektroonilistel instrumentidel, arvutitel ja teistel seadmetel (näiteks MIDI-t toetavad valgustusseadmed teatrites) omavahel suhelda, üksteist juhtida ja sünkroniseerida. Eelduseks on, et seadmed toetavad MIDI standardit ja nende vahel on MIDI-ühendus (standardne MIDI-kaabel või näiteks USB-MIDI kasutajaliides) või WIDI-ühendus ("traadita" MIDI ehk Wireless MIDI). MIDI-ks nimetatakse ka kaabli või WIDI kaudu edastatavat MIDI teavet. [34]

MIDI standard käsitleb digitaalseid käske – sündmuste teateid ja nende edastamist, mis hõlmavad teavet näiteks tempo, helikõrguse, oktavi, helitugevuse, modulatsiooni, helipanoraamis paiknemise jne kohta. [34]

Standardne MIDI failiformaat võimaldab muusikalisi jadasid salvestada ja kuvada ning erinevate süsteemide ja rakenduste vahel üle kanda. Standardi arendamise ja hooldamisega tegeleb mittetulundusühing MMA (MIDI Manufacturers Association), mille eesmärk on luua MIDI standard ja jälgida selle ühilduvust erinevate MIDI toodete vahel. [35]

MIDI failid on masinloetavad, ehk MIDI faili tekstiredaktoriga avades ei ole võimalik ilma eriteadmisi omamata sisust aru saada. Seetõttu on MIDI failide kasutamise puhul vajalik kasutada ka täiendavat programmi, mis teisendab MIDI faili sisu inimloetavale kujule. MIDI faili on võimalik lisada ka teksti, mistõttu on võimalik luua ja kuvada noodistusi, mis sisaldavad noodistust ja teksti korraga. [34]

3.3.3 ABC notatsioon

ABC notatsioon on inim- ja masinloetav, lihtsustatud muusika kodeering. Lihtsustamine tähendab, et see kasutab tähti a-g, A-G ja z selleks, et tähistada vastavaid noote ja pause. Oktavite vahel liikumise, dieesi, bemolli jms lisade tähistamiseks kasutatakse täiendavaid märke. ABC kodeeringu autoriks on Chris Walshaw [36] ja selle eesmärk oli luua noodistuse standard, mida oleks lihtne kodeerida kasutades ASCII märke. Standard loodi selleks, et oleks võimalik kodeerida rahvaviise ja traditsioonilisi Lääne Euroopa viise. Need on tavaliselt ühehäälsed meloodiad, mida on võimalik kodeerida ühele reale. Hilisemate arenduste käigus laiendati selle standardi võimekust, lisades täiendavaid funktsioone, mille tulemusena on kodeeritud esitusse võimalik lisada veelgi enam informatsiooni. [37]

Kuna ABC notatsiooni kodeering on tekstipõhine, siis on nootide ABC vormingus kodeerimiseks võimalik kasutada ükskõik millist tekstiredaktorit. Lisaks sellele on olemas mitmeid rakendusi, mille abil on võimalik teistest kodeeringutest nagu MIDI ja MusicXML teisendada kodeeringuid ABC kujule.

Kodeeringu veel üheks oluliseks omaduseks on inimloetavus ja -kirjutatavus. Lühikese aja jooksul on inimesel võimalik õppida noodistust otse ABC kodeeringus kirjutama ning puudub vajadus täiendavate teisendusi tegevate rakenduste järele. Selle tulemusena on võimalik saada kodeeritud muusika andmemaht väga väikeseks.

ABC notatsiooniga kodeeritud rahva ja pärimusmuusika uurimiseks on olemas avalik andmebaas, kus on abc kujul kodeeritud umbkaudu 630 000 viisi. [38]

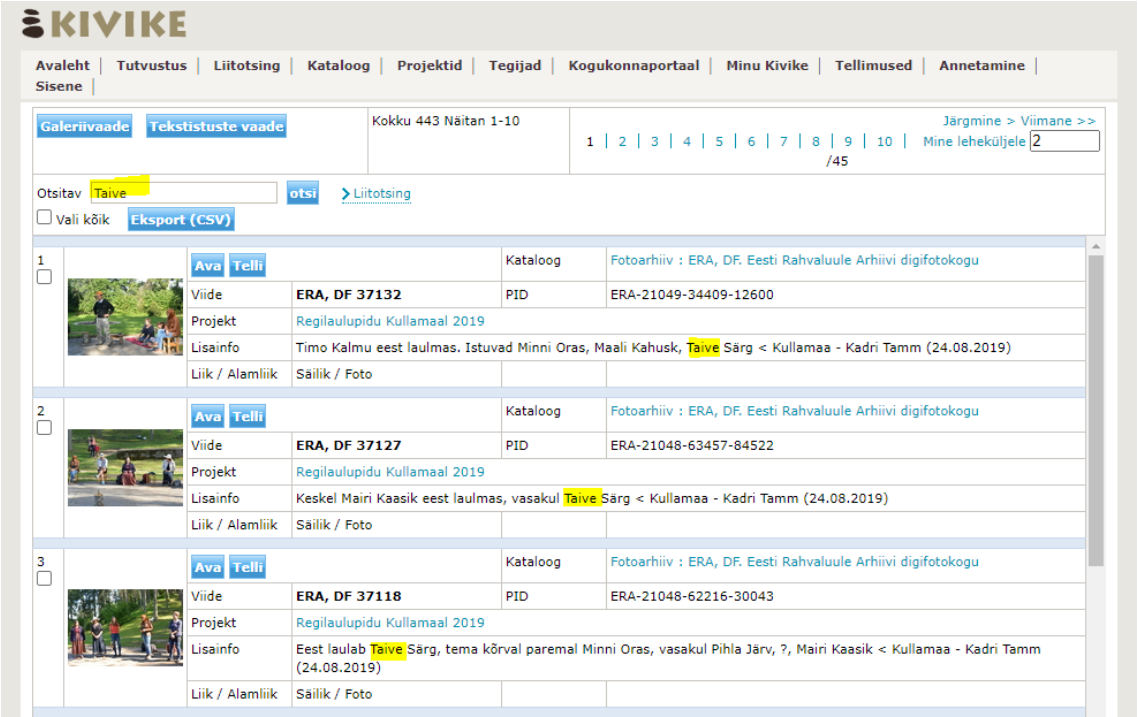
3.4 Kivike

Repositoorium e materjalide varamu Kivike on infosüsteem, milles on kahte tüüpi teavet: digitaliseeritud arhiivimaterjalid ja nende metaandmed. Failihoidlas leidub ka viiteid sellistele materjalidele, mida digitaliseeritud kujul veel Kivikesse lisatud ei ole. Selliseid faile ja andmeid lisavad EKM töötajad käsitsi jooksvalt süsteemi. [3]

Viidete loomise võimalus loodava rahvaviiside infosüsteemi ja kivikese arhivaalide vahel oli üks EKM-i soovitud funktsionaalsustest. Sellise viitamise abil on võimalik kokku viia viise nendega seotud arhivaalidega nagu näiteks käsikirjad või helisalvestised.

Repositooriumi kasutajaliidese ja kirjandusmuuseumi andmebaaside kaudu on võimalik Kivikeses olevaid materjale otsida. Huvi korral (näiteks õppetöö eesmärkidel) saab neid faile kasutada, kuid selle jaoks peab huvilisel olema kasutajakonto, millega ta Kivikese infosüsteemi sisse logib. Nii on võimalik vaadata faile ja tutvuda andmetega, kuulata helisalvestisi ja vajadusel ka tellida endale koopiaid.

Kivikese infosüsteemist saab materjale otsida kas liht- või liitotsinguga. Lihtotsing asub Kivikese avalehel ja see sobib siis kui otsija täpselt ei tea, mis on otsitava teose nimi (Joonis 11). Otsinguvälja tuleb lisada märksõna või numbrikombinatsioon, mille järgi otsitakse kõikidelt säiliku, pala või failiga seotud väljadelt. Lihtotsinguga saab otsida täpse fraasi järgi, kui see on asetatud jutumärkidesse.

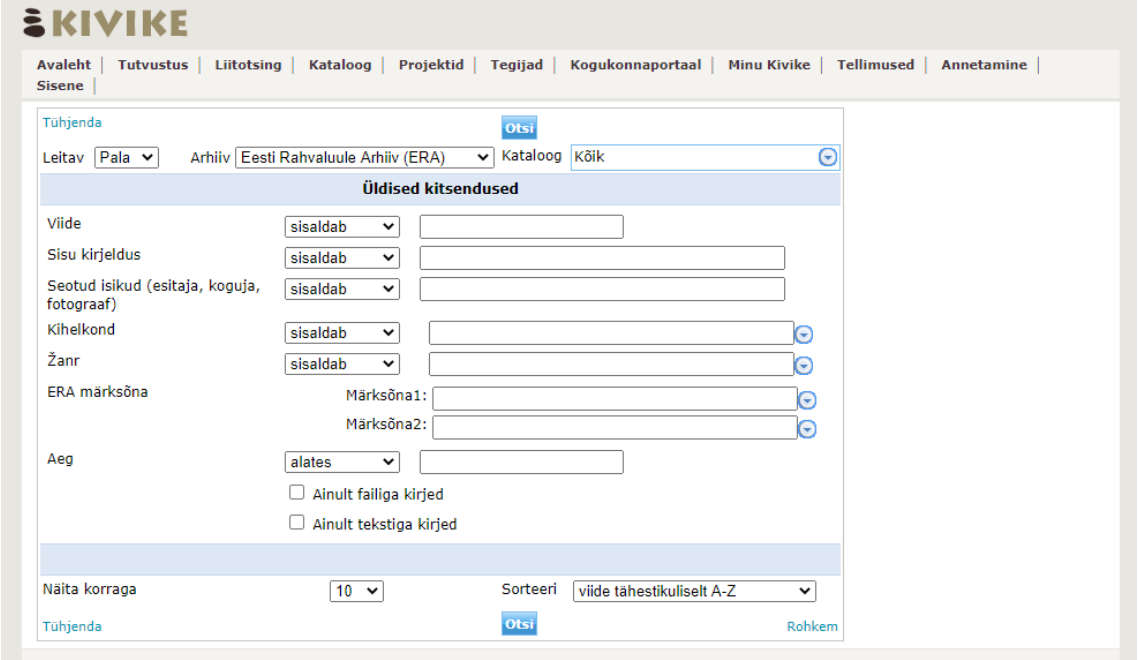


Joonis 11 Näide Kivikese lihtotsingust.

Liitotsing on eelmisest täpsem, otsida saab erinevate parameetrite alusel (Joonis 12). Kui vaikimisi pakutust ei piisa, siis saab otsinguväljasid ka juurde lisada. Nii on võimalik saada küllaltki täpne otsingutulemus. Kui rippmenüüst täpsustada arhiiv, millest otsida, siis muutuvad otsinguväljad vastavalt valitud arhiivile. Arhiivid on Arhiivraamatukogu, Eesti Kultuurilooline Arhiiv, Eesti Rahvaluule Arhiiv ja Folkloristika osakond (Kevad 2021 seisuga).

Otsida saab ka spetsiifiliselt ainult tegija järgi. Tegijaks võib olla üksikisik või kollektiiv. Tegijad on autorid, esitajad, kogujad ja fotograafid. Iga teos on seotud isikutega lihtsustamaks teoste otsingut juhul kui autor või esitaja on teada

Andmeid leiab ka projektide järgi sorteerides. Valida saab järgmiste kategooriate vahel: välitööd, üritused ja suuremahulised digitaliseerimisprojektid.



Joonis 12 Kivikese liitotsing.

3.5 Saaty meetod

Saaty meetod ehk analüütiliste hierarhiate meetod (*Analytic Hierarchy Process, AHP*) on välja töötatud USA matemaatiku Thomas Lorie Saaty poolt. Meetod loodi 1970ndatel aastatel ja selle peamine eesmärk on toetada keerukate otsuste tegemist läbi alternatiivide paariviisilise võrdlemise. Seda kasutatakse otsuste tegemiseks, millel on mitu kriteeriumit ja alternatiivi. Meetodi kasutamine toetab valikute põhjendamist ja aitab tõestada, et tegemist on õige otsusega. [22].

AHP on:

1. Analüütiline – otsuseid saadakse matemaatiliste arvutuste põhjal ning tehakse loogika abil.

2. Hierarhiline – hinnanguid saab struktureerida ja erinevatele tasanditele paigutada.

3. Protsess – otsuste tegemisel on menetluslik iseloom, meetod näeb otsuseni jõudmiseks ette kindla tegevuste järjekorra.

AHP võimaldab teha otsuseid meeskondades ja muuta otsustamine ja tulemuste sõnastamine arusaadavaks. [39]

Esmalt modelleeritakse probleem hierarhiana, et jõuda ühisele arusaamale protsessi eesmärgist. Hierarhia koosneb mitmetest tasemetest ja elementidest ning aitab probleemi lahti mõtestada ja paremini mõista. Kõige kõrgemal tasemel on eesmärk (probleem), selle all on ühel või mitmel tasemel kriteeriumid ning kõige alumisel tasemel on alternatiivid, mille hulgast soovitakse valik teha. Teise sammuna hinnatakse kriteeriumeid paariviisiliselt, et leida nende suhteline osatähtsus. Hinnang antakse Saaty hinnangute skaala järgi, mis koosneb sõnalistest hinnangutest, millest igaühele vastab arvuline väärtus. Sõnaliselt on subjektiivset arvamust lihtsam edastada. Arvuline skaala aitab seejärel kriteeriumite osatähtsust arvutada. [22]

Saaty hinnangute skaala ehk fundamentaalskaala on esitatud tabelis Tabel 2:

Tabel 2 Saaty hinnangute skaala.

Väärtus	**Hinnang**
1	Võrdtähtis
3	Mõõdukas paremus
5	Oluline paremus
7	Väga tugev paremus
9	Äärmuslik või ekstreemne paremus

*ülejäänud väärtused (2, 4, 6 ja 8) on vahepealsed hinnangud [22].

Järgmise sammuna keskendutakse alternatiividele ja nende tähtsuse arvutamisele. Alternatiive võrreldakse paarikaupa ning seda tehakse iga kriteeriumi korral eraldi. Lõpptulemuse arvutamiseks kasutatakse alternatiivide võrdlemisel saadud suhtelise

headuse hinnanguid kriteeriumite suhtes ja kriteeriumite osatähtsuseid. Selliselt leitakse iga alternatiivi suhteline headus, mis on esialgselt seatud probleemi lahendamiseks parim valik.

Grupiotsuste tegemisel on kirjeldatud erinevaid konsensuse meetodeid. Vahet tehakse kahel erineval lähenemisel: Eristatakse deterministlikku ja statistilist lähenemist. [40]

Deterministlik grupiotsustamine ehk omaväärtuse meetod: Otsustamisel osalevad isikud asuvad geograafiliselt tihedalt koos ja saavad osaleda arutelus. Rühma liikmetel on otsuste tegemisel üldjuhul võrdne tähtsus. Kui mõni grupiliige on olenevalt valdkonnast teistest liikmetest targem, peaks ta teisi vastavalt mõjutama õige suunise leidmiseks. Sellisel juhul tuleks talle omistada otsustusprotsessis suurem kaal.

Statistiline grupiotsustamine puhul on osalevad isikuid on palju. Igal osalejal on võrdne tähtsus.

Deterministlik grupiotsustamine on kohaldatav olukordadele, kus rühm on väike, statistiline kui kaasatud on laiem osa elanikkonnast. Antud töös kasutati deterministlikku lähenemisviisi, sest väikese rühma tõttu oli võimalik iga otsustamises osalejaga individuaalselt suhelda ja hinnanguid läbi arutada ja põhjendada.

Grupiotsuste tegemisel peab üksikisikute eelistusi paremusjärjestusse seadma ja konsensuse leidma (konsensus ehk üldsuse üldise kokkuleppe protsess). Konsensuse tüüpe saab jagada kolmeks: spontaanne, tekkiv ja manipuleeritud konsensus.

Spontaanne üksmeel tekib traditsioonilistes kogukondades, kus on levinud samad vaated ja arusaamad. Tekkiv konsensus toimub mittetraditsioonilistes ühiskondades. Kuulatakse kõigi seisukohad ära, mille järel iga inimene hindab oma tekkinud seisukohta. Nii moodustub nö avalik arvamus. Kui suurem osa otsustajatest on veendunud ühest vaatest, siis on tekkinud enamus piisavalt jõuline, et vähemused selle seisukoha omaks võtaks. Manipuleeritud konsensus on olemas ühiskondades, kus teoreetiliselt võib tekkida tekkiv konsensus, kuid see sõltub sellest, kes ühiskondlike veenmisi ja vahendeid kontrollib. [40]

Antud töös kasutati otsuste tegemisel ja põhikriteeriumite võrdlemisel tekkiva konsensuse grupiotsustamist. [41]

4 Analüüs

Selles peatükis esitatakse nõuded süsteemi täiendustele ja parandustele. Nõuete mõistmiseks on vajalik teoreetiline taust on esitatud peatükis 3.

4.1 Funktsionaalsed nõuded

Kõik eelnevalt bakalaureuseõppe tudengite poolt kogutud ja realiseeritud nõuded (vt [7]) säilisid ka loodavas infosüsteemi versioonis. Täiendavalt lisandusid neile järgnevates alajaotustes kirjeldatud nõuded.

4.1.1 Viiside noodistuste haldus

Viiside juurde peab olema võimalik lisada kodeeritud noodistust. Kodeeritud noodistus peab asuma andmebaasis ning olema kogu süsteemi ulatuses kodeeritud vastavalt samadele põhimõtetele. Viisi külge peab olema võimalik lisada ühte või mitut kodeeritud noodistust ning iga kodeeritud noodistuse külge peab olema võimalik lisada ühte või mitut variatsiooni.

4.1.2 Otsing

Otsida peab saama viisi metaandmete hulgast. Otsingu tulemusena peab tagastatama asjakohased viisid. Otsingu tulemustele peab saama rakendada massimuutmist.

4.1.3 Massimuutmine

Massimuutmise abil peab olema võimalik mitmel viisil samaaegselt väljade infot uuendada. Enne muutmist peab olema võimalik otsida viise erinevate väljade alusel ning otsingu tulemuses valida kuvatavaid välju. Muutmiseks peab olema võimalik valida kõiki viisi andmevälju. Võimalik peab olema nii uue väärtuse lisamine, olemasoleva väärtuse asendamine kui olemasoleva väärtuse eemaldamine. Massimuutuse muudatust tuleb enne selle rakendamist eraldi vaates kuvada ning alles peale kinnituse saamist rakendada.

4.1.4 Mitme keele tugi

Kasutajaliideses peab olema mitme keele tugi. See tähendab, et peab olema võimalik nii väljanimede kui klassifikaatorite nimede tõlkimine erinevatesse keeltesse. Väljanimede

tõlkimine võib olla rakenduse loogikasse sisse kirjutatud kuid klassifikaatorite tõlget peab saama klassifikaatori põhiselt teha, st erinevad nimed peavad olema andmebaasis.

4.1.5 Viidete lisamine Kivikese infosüsteemi

EKM soovib viisi kirjete juurde võimalust lisada viiteid Kivikese repositooriumis olevatele arhivaalidele. Esialgu oli soov, et Kivikese infosüsteemi SOAP rakendusliidese kaudu tehtaks päringuid Kivikese infosüsteemi ning vastuseid kuvataks viisi juures eraldi sektsioonis. Antud ideed testides ilmnes asjaolu, et otsingu tulemused ei olnud sellise lähenemise kasutamiseks piisavalt usaldusväärsed. Oli juhtumeid kus vastuseid ei tulnud või kus vastusesse tuli rohkem kirjeid kui esialgu arvati.

Läbirääkimiste tulemusel jõuti otsuseni, et iga viisi juures hakatakse hoidma täpset arhivaali PID-i koos kirjeldusega. Selle tulemusena ei ole võimalik küll automaatset otsingut viisiviite järgi teha, kuid on võimalik tekitada kindel seos arhivaali ja viisi vahel. Lisaks on võimalik kirjeldada iga viite puhul täpselt selline liik nagu toimetaja soovib. Liikide näideteks on käsikiri ja esitlus.

4.1.6 Andmemuudatuste logimine

Viiside andmete lisamisel, muutmisel ja kustutamisel peab sellest maha jääma jälg, et oleks teada, kes, millal ja millise muudatuse tegi ning milline oli selle sisu.

4.2 Mittefunktsionaalsed nõuded

Käesoleva töö sisendiks olevas bakalaureusetöös [7] olid sõnastatud rahvaviiside infosüsteemi mittefunktsionaalsed nõuded.

Nendele nõuetele lisaks pidasime vajalikuks lisada nõuded, mis tulenevad EKM-i ISKE M-taseme meetmetest. Täpsem ülevaade meetmete kohta on jaotises 4.3.1.

Samuti sõnastasime nõude tarkvarale, et süsteemi realiseerimiseks tuleks kasutada vaid avatud ja tasuta tarkvara ning standardeid, mille juurutamine ei tooks kaasa täiendavaid oste ei tarkvara ega litsentside näol.

4.2.1 Paroolipoliitika

Paroolipoliitika seadmisel lähtuti Microsofti soovitustest tugevatele paroolidele [42] ning kehtestati järgnevad nõuded infosüsteemi tarkvaras kasutatavatele paroolidele.

- Vähemalt 14 märgi pikkune.
- Parool peab sisaldama vähemalt ühte suurtähte.
- Parool peab sisaldama vähemalt ühte väiketähte.
- Parool peab sisaldama vähemalt ühte numbrit.
- Parool peab sisaldama vähemalt ühte sümbolit, mis pole numbrimärk ega täht.

Antud parooli nõuded vastavad Microsofti soovitatud miinimumile ning ületavad ISKE meetme M 2.11 nõudeid. [43]

4.3 Süsteemi vastavus regulatsioonidele

Loodav infosüsteem sisaldab endas andmeid, millele rakenduvad erinevad seadustest ja määrustest tulenevad kohustused ja piirangud. Järgnevalt on lahti selgitatud erinevate loodavat infosüsteemi mõjutavate kohustuste ja piirangute olemust ja nende rakendumist loodavas infosüsteemis. Sellist analüüsi ei ole loodavale süsteemile varem tehtud.

4.3.1 Infoturve

„Eesti Kirjandusmuuseumi arengukava aastateks 2019-2022“ kohaselt rakendab EKM infoturbe meetmeid, mis peavad tagama EKM-i poolt kogutud materjalide ja andmete kaitse. EKM vastab ISKE keskmisele turbeastmele (M) [44].

ISKE-s on defineeritud kolm turbeastet: madal (L), keskmine (M) ja kõrge (H). Turbeaste määratakse andmete turvaklasside kaudu. Nende määramisel lähtutakse andmete konfidentsiaalsusest, terviklikkusest ja käideldavusest [45].

ISKE turbetase M tähendab seda, et tagatud peavad olema L taseme ja M taseme meetmete rakendamine. Kuna loodav rakendus on veebirakendus, siis kehtib sellele moodul B 5.21 Veebirakendused. [46]

“Kogutud andmete turvalisus on tagatud sellises ulatuses, et probleemide korral, saaks EKM oma põhitegevuste täitmiseks normaalselt tööd jätkata. Turvameetmed on proportsioonis võimaliku kahjuga, mis võib tuleneda meetmete puudulikkusest. Võimalikult väikesena üritatakse hoida häiriv toime EKM-i tegevusele” [44].

Töötajate teavitamine ja harimine turvaeesmärkidest toimub perioodiliselt. Korraldatakse koolitusi ja avalikult kättesaadavad on ka harivad materjalid infoturbe teemadel. Erikoolitusi saavad EKM-i töötajad, kes on andmeturbega seotud tööpositsioonidel [44].

Sellest järeldame, et EKM-i töötajatel on baasteadmised infoturbe põhimõtetest ning EKM-i poolt on juurutatud M taseme saavutamiseks nõutud moodulid, mis on välja toodud kui soovitatavad moodulid B 5.21 moodulile. Töö autorid eeldavad seda kuna EKM-is on kasutusel ka teisi veebirakendusi. Seetõttu keskendub töö ainult veebirakendusi otseselt puudutavate nõuete analüüsile ning täitmisele.

Vastavalt eesmärgiks seatud turbeastmele M on vaja rakendada järgnevaid meetmeid [47].

M 2.1 (L) IT kasutajate vastutuse ja reeglite kehtestamine – Meede on organisatoorne. Meetme rakendamiseks eeldatakse ettevõtte siseselt protsesside defineerimist ja juurutamist. EKM-i vastutusalas on kõikide meetmest tingitud äriprotsesside kirjeldamine ja juurutamine.

M 2.8 (L) IT-rakendustele ja andmetele pääsuõiguste andmine – Üleantavas rakenduses on defineeritud erinevad rollid, mille alusel on piiratud ligipääs erinevatele rakenduse funktsioonidele. Rakenduse üleandmise raames antakse üle ka administraatori õigustega kasutaja. Üleandmise hetkest läheb rakenduse pääsuõiguste jagamine üle EKM-i vastutusalasse ning eeldame, et EKM-il on paigas õiguste jagamise protsessid.

M 2.11 (L) Paroolide kasutamise reeglid – Üleantavas rakenduses kasutatakse paroolipoliitikat vastavalt jaotisele 4.2.1. Paroolide reeglid on seatud vastavalt ISKE nõude M 2.11 soovitusele.

M 2.31 (L) Volitatud kasutajate ja õiguste profiilide dokumenteerimine – Meetme sisu nõuab organisatsioonilt õiguste kirjeldamise ja jagamise dokumentatsiooni olemasolu ning juurutatud protsessi. Loodava lahenduse puhul on kasutajarollid kirjeldatud jaotises 5.8 ja rollide jagamise/jälgimise protsess on EKM-i hallata.

M 2.34 (L) IT-süsteemi muutuste dokumenteerimine – Töö valmimisel antakse üle täielik lähtekood ning hilisemate muudatuste dokumenteerimise ja haldus on EKM-i vastutusalas.

M 2.35 (L) Teabe hankimine turvaaukude kohta – Loodud tarkvara uuendatakse üleandmise hetkel viimasele versioonile ning üleandmise hetkest läheb tarkvara uuendamise kohustus üle EKM-ile. Tarkvara uuendamise ja turvaaukude seire on EKM-i organisatoorne vastutus.

M 2.62 (L) Tarkvara vastuvõtuprotseduurid – Tegemist on organisatoorse meetmega, täpsemad vastuvõtu tingimused peavad olema EKM-il dokumenteeritud ning jälgitavad.

M 2.63 (L) Pääsuvolituste kehtestamine – Pääsuvolituste jagamine ja kontroll on EKM-i protsess. Antud töö raames defineeriti erinevad kasutajarollid ning anti üle administraatori õigustega kasutaja. Hilisem meetme vastavuse jälgimine jääb EKM-i vastutusalasse.

M 2.80 (L) Tüüptarkvara nõuete kataloogi koostamine – Tüüptarkvara nimekirja loome on EKM-i ülesanne ning jääb antud projekti skoobist välja.

M 2.273 (L) Turvalisust mõjutavate paikade ja värskenduste kiire paigaldamine – Turvapaikade paigaldamine on organisatoorne meede. Loodav lahendus sisaldab lisaks käsitsi kirjutatud lähtekoodile ka komponente/pistikprogramme, mida on võimalik pakihalduriga uuendada. Uuendamise protsess aga on meede, mida tuleb EKM-il endal kasutusele võtta.

M 2.363 (L) Kaitse SQL-süstimise vastu – Üle antava infosüsteemi lüüsikiht kasutab pistmiku *loopback-connector-postgresql*, mis puhastab automaatselt kõik süsteemi poolt genereeritud SQL laused. Kasutajatel ei ole võimalik käitada ise defineeritud lauseid.

M 2.486 (L) Veebirakenduste ja veebiteenuste arhitektuuri dokumenteerimine – Antud lõputöö sisaldab täidab kõiki antud meetmes kirjeldatud dokumentatsiooni nõudeid ning seetõttu võib antud lõputööd pidada piisavaks meetme täidetuks tunnistamiseks piisavaks.

M 2.487 (L) Veebirakenduste arendamine ja laiendamine – Tegemist on organisatoorse nõudega, mis rakendub edasiste arenduste tegemisel. Käesoleva töö raames lähtuti üldistest nõuetest ning täiendavad arendused peavad olema reguleeritud vastavalt organisatsiooni arenduspoliitikale.

M 2.488 (L) Veebiliikluse jälgimine – Antud projekti raames ei ole soovi antud meetmes kirjeldatud statistikat korjata ning seetõttu seda meedet antud projekti raames ei rakendata.

M 3.5 (L) Turvameetmete koolitus – Turvateadlikkuse tõstmine on EKM-i kui organisatsiooni kohustus.

M 4.78 (L) Konfiguratsioonimuudatuste hoolikas teostamine – Konfiguratsioonimuudatuste hilisem jälgimine peab olema EKM-i protsessi osa ning kuna turbetase M on asutusel saavutatud, siis loeme selle täidetuks.

M 4.392 (L) Autentimine veebirakendustes – Rakenduses kasutatakse autentimist ning paroolipoliitika vastab ISKE meetmele **M 2.11 (L) Paroolide kasutamise reeglid.** Parooli meeldejätmise funktsionaalsust ei ole juurutatud. Täiendava autentimise nõue parooli vahetamisel on kehtestatud. Rohkem kriitilisi funktsionaalsusi töö autorid ei tuvastanud, kuna puudub masskustutamise võimekus, mida meetme korral tuuakse välja kui kriitilist funktsionaalsust, mille puhul võiks kasutada täiendavat autentimist.

M 4.393 (L) Sisestuste- ja väljastuste põhjalik valideerimine veebirakendustes ja veebiteenustes – Rakenduse lüüsikihis toimub sisestuste- ja väljastuste valideerimine mudelites defineeritud tüüpide alusel. Kõik väärtused kontrollitakse sisestuste puhul enne andmebaasiga suhtlust ning väljastuse korral koheselt pärast andmebaasist vastuse saamist. Ebakorrapärasuse tekkimisel tagastatakse asjakohane vastus.

M 4.394 (L) Seansihaldus veebirakendustes – Veebirakenduse seanss on esitluskihis defineeritud kui JWT luba. Seansi info on allkirjastatud lüüsikihi poolt ning seda kontrollitakse iga autentimist vajava päringu korral lüüsikihi poolt. Luba väljastatakse 96 tunniks. Selle aegumise korral tagastab lüüsikiht vastava vastuse ning nõuab uuesti mandaadi saatmist uue loa loomiseks.

M 4.395 (L) Tõrkekäsitlus veebirakendustes ja veebiteenustes – Veebirakendustes tagastatakse kasutajale vaid kindlaid veateateid, mis ei sisalda informatsiooni rakenduse loogika/andmebaasikeele lausete kohta. Tõrke ilmnemisel tagastatakse kasutajale veateade eeldefineeritud vormi alusel ning täiendavat infot süsteemi SQL lause/vastuste/seadistuste kohta kliendile ei tagastata. Tõrke ilmnemisel soovitud tegevus katkestatakse ning tegevuse edasist käitlemist ei toimu.

M 4.396 (L) Veebirakenduste kaitsmine keelatud automaatkasutuse eest – Antud meetme puhul ei ole rakendatud täiendavaid aegviiteid parooli sisestamise võimaluste puhul. Autorid ei pidanud vajalikuks lisada rakendusele CAPTCHA testi, mille abil veendutakse, et vastaja on inimene. Samuti eeldavad autorid, et EKM-il on kasutusel täiendavad turvarakendused, mis kaitsevad rakendust automatiseeritud rünnete vastu.

M 4.398 (L) Veebirakenduste turvaline konfiguratsioon – Rakenduse loogikas on lubatud vaid need HTTP meetodid, mis on rakenduse tööks vajalikud. Iga otspunkt on kaitstud täiendavate rollipõhiste piirangutega. Kogu rakendusega toimuv suhtlus on turvatud krüpteeritud transpordikanalitega. Konfiguratsioonifailid on salvestatud väljaspool veebi juurkataloogi ning haldamiseks vajaliku tarkvara seadistus on EKM-i haldusalas.

M 4.399 (L) Andmete ja sisu kontrollitud lisamine veebirakendustesse – Failide üleslaadimine on lubatud vaid autenditud kasutajatel. Suunamisel rakenduse välisele lehele avatakse uus aken lehe sisuga. *HTTP-Request referrer* andmevälja kasutatakse kuna välised lehed, millele suunatakse, on kindlaks määratud süsteemi poolt ning kasutajad ei saa ise URLe sisestada.

M 4.401 (L) Konfidentsiaalsete andmete kaitse veebirakendustes – Andmevahetus kasutaja ja rakenduse vahel toimub ainult krüpteeritud transpordikanali kaudu, failide üleslaadimist ei lubata ning failide alla laadimise funktsionaalsus ei paljasta konfidentsiaalset informatsiooni.

M 4.402 (L) Juurdepääsukontroll veebirakendustes – Pääsuõiguste kontrollimisel lähtutakse kasutajale omistatud rollidest. Kõik õiguste omistamised rakenduses saavutatakse läbi rollide omistamise. Volituskomponent on seadistatud arvestama kõikide otspunktidega, ehk iga otspunkti puhul on määratletud selle kasutamiseks vajaminev roll.

M 4.404 (L) Veebirakenduste turvalise loogika kavandamine – Loodav kliendirakendus on üles ehitatud JavaScript-il põhineval raamistikul. Sellest tulenevalt antud rakenduse puhul seda meedet ei rakendata.

M 4.406z (L) Clickjacking-rünnete tõkestamine - *x-frame-options SAMEORIGIN* päiseparameeter on rakendatud.

M 5.66z (L) SSL-i/TLS-i kasutamine kliendis – SSL sertifikaatide tellimine ja kasutuselevõtt on EKM-i ülesanne.

M 5.168 (L) Taustsüsteemide turvaline sidumine veebirakenduste ja veebiteenustega – Taustsüsteemide turvaline sidumine ning täiendavate piirangute seadmine/jälgimine on EKM-i ülesanne ning ei ole antud projekti skoobis.

M 5.169 (L) Veebirakenduse süsteemiarhitektuur – Loodud lahendus on üles ehitatud vastavalt meetme soovitustele. Arhitektuuri joonis on jaotises 5.5.

M 2.64 (M) Logifailide kontroll – Kontrolli logifailide üle teostab EKM. Seetõttu jääb meetme kontroll antud töö skoobist välja.

M 2.110 (M) Andmeprivaatsuse suunised logimisprotseduurides – Andmeprivaatsuse kontroll on EKM-i protseduuriline meede, mis jääb antud töö skoobist välja.

M 4.176 (M) Autentimismeetodite valimine veebilehtede jaoks – Kasutusel on blanketil põhinev autentimine koos SSL-iga, mida võib pidada meetme täidetuks lugemiseks piisavaks.

M 4.397 (M) Veebirakenduste turvet puudutavate sündmuste logimine – Kontrolli logifailide ja logimise seadistuse üle teostab EKM. Seetõttu jääb meetme kontroll antud töö skoobist välja.

M 4.400 (M) Turbe seisukohalt oluliste andmete väljastamine veebirakendustes – Lüüsikihist väljastatakse eeldefineeritud veateated. Esitluskiht oskab antud veateateid tõlgendada kasutajatele arusaadavateks veateadeteks sõltuvalt kontekstist. See on hea kuna sellisel lähenemise puhul pole välisel ründajal võimalik tuvastada milline moodul veateadet väljastab. Ülejäänud meetme nõuded on organisatoorsed ning antud töö skoobist väljas.

M 4.403 (M) Cross-Site Request Forgery (CSRF, XSRF, Session Riding) tõkestamine – Lüüsikiht loob autenditud kasutajatele JWT vormingus loa, mis on võltsimise vältimiseks ka allkirjastatud. Kõikide autentimist või autoriseerimist vajavate tegevuste puhul kontrollitakse väljastatud luba ning selle terviklikkust allkirja vastu.

M 4.405 (M) Ressursside blokeerimise (DoS-rünnete) tõkestamine veebirakendustes ja veebiteenustes – Organisatoorne meede. Täiendavat funktsionaalsust WAF/cloudflare teenusega ei juurutatud. Punkti rakendamine jääb EKM-i otsustada.

M 5.150 (M) Läbistustestide läbiviimine – Läbistustestimine on suur ettevõtmine ning jäi antud töö skoobist välja. Autorid eeldavad, et järgmisel korral kui toimub läbistustestimine, siis lisatakse loodav infosüsteem testimise skoopi.

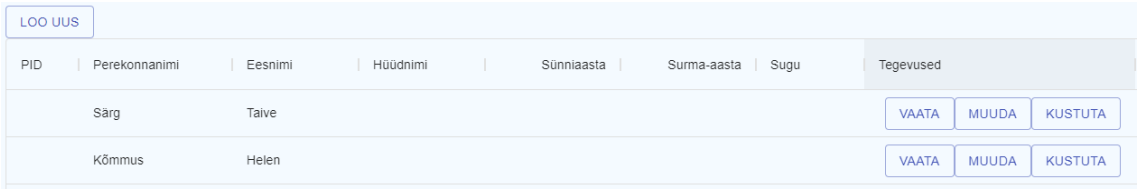
M 5.177 (M) SSL-i/TLS-i kasutamine serveris – Sertifikaatide haldus ja poliitika on EKM-i halduspädevuses ning antud projekti skoopi ei kuulu.

4.3.2 Andmekaitse ja GDPR

Euroopas kehtestatud isikuandmete kaitse üldmäärus, edaspidi GDPR (*General Data Protection Regulation*), reguleerib füüsiliste isikute andmete töötlemist. Eesti isikuandmete kaitse seadus [48] võtab selle üldmääruse põhimõtted Eestisse üle.

Isikuandmed on mistahes andmed, mille kaudu on võimalik isikut tuvastada. Need andmed väljendavad näiteks isiku füüsilisi, majanduslikke, kultuurilisi ja ka sotsiaalseid omadusi ning kuuluvusi. Andmeid, mis kasvõi kaudselt võivad viidata konkreetsele isikule, tuleb vaadelda isikuandmetena. Ka mitteidentifitseeritavaks muudetud või krüpteeritud isikuandmed, mida saab kasutada isiku tuvastamiseks, jäävad isikuandmeteks. Kui andmed on muudetud anonüümseks ja nendest ei ole enam võimalik kuidagi üksikisikut tuvastada, siis neid ei loeta enam isikuandmeteks. [48]

Isikuandmed, mida EKM teoste juures kasutab on näiteks nimi ja perekonnanimi, hüüdnimi, elukoht, sünni- ja surma-aasta, sugu (Joonis 13).



Joonis 13 Isikute vaade.

Kui digitaliseeritud objektidele antakse juurdepääs igale kasutajale, siis isikuandmete kaitse seab digitaliseerimise kontekstis arenduse tulemusele piiranguid. Kehtivate piirangute olemasolul on objekti metaandmed avalikud, kuid isikuandmed ei tohi avalikult kättesaadavaks tehta.

Andmesubjektil (isik, kelle andmeid töödeldakse) on õigus esitada vastuväiteid töödeldavate andmete kohta. See õigus ei kehti siis, kui andmeid töödeldakse avalike huvide täitmiseks.

GDPR-i kohaselt on andmesubjektil õigus nõuda enda kohta käivate andmete kustutamist. Teiste sõnadega on tal õigus olla unustatud. Avalikes huvides arhiveerimise, teadus- või ajaloouuringute kontekstis on antud seaduse punkti täpsustatud. See tähendab, et andmete kustutamist ja selle õigust ei rakendata kui see moonutab töötlemise eesmärki [49]. Selle kohta on isikuandmete kaitse seaduse eelnõusse lisatud eraldi lõik:

Isikuandmete kaitse seadus §7 lg 2 „Käesoleva paragrahvi lõikes 1 nimetatud andmesubjekti õigusi võib piirata selleks, et mitte ohustada arhivaalide seisundit, nende autentsust, usaldusväärsust, terviklikkust ja kasutatavust.“

Kui infosüsteemiga seotud kasutajad vahetuvad, siis tekib vajadus kontode blokeerimise järele. Seda saab teha deaktiveerimise teel, sest kasutajaid füüsiliselt ära kustutada ei tohi. Kontode kustutamine on välistatud, sest kasutajad on seotud dokumentidega. Tulevikus võib tekkida vajadus näha dokumentide muutmisajalugu, mistõttu peab iga muudatuse kohta säilima taasesitatav kirje muudatusest, selle tegemise ajast ja tegijast.

Deaktiveeritud kasutaja ei saa süsteemi sisse logida. Tema kasutajakonto jääb nimekirjas „Kasutajad“ nähtavaks ja juurde on märgitud kasutaja staatus, milleks antud juhul on deaktiveeritud. Deaktiveeritud kasutajat saab uuesti aktiveerida.

4.3.3 Autoriõigus

Eesti Vabariigi põhiseadusele tuginedes on Eesti riik kohustatud tagama rahvuse ja kultuuri säilitamise läbi aegade. Kultuuriruumi säilitamiseks on Kultuuriministeerium koostanud kultuuripärandi digitaliseerimise tegevuskava aastateks 2018-2023, mille eesmärgiks on Eesti kultuuripärandi digitaliseerimine ja ühtselt säilitamine. See tähendab, et info on rahvusvaheliselt kättesaadav virtuaalkeskkonnas ehk elektroonilistes arhiivides. [50] EKM-i vaatenurgast on see äärmiselt oluline, sest suur osa rahvaviisidest on säilitatud vaid paberkandjal suurtes arhiivides. Digitaliseerimine aitab tagada selle terviklikku säilimist ja kättesaadavaks tegemist kõikidele huvigruppidele.

Loodud rahvaviiside infosüsteem säilitab suurtes kogustes informatsiooni, mille puhul vajab andmekaitse ja autoriõiguste tagamine eritähelepanu. Kogutavaks informatsiooniks

on autori(te) kohta käiv lühikirjeldus, viisi noodistik, laulusõnad ja helifailid. Seaduse järgi tekivad autoriõigused, kui teos vastab teatud kriteeriumitele, mis on Autoriõiguse seaduses (AutÕS) eraldi välja toodud. [51] Näiteks peab teos olema originaalne ja seda peab saama mingil moel taasesitada (AutÕS § 7 lg 3).

Seadus kohaldatakse teostele, mille loojateks on Eesti Vabariigi kodanik või siin alaliselt elav isik. Siiski on rahvaviisid erandiks ja neid ei saa vaadelda nagu tavalist muusikateost, sest tegemist on rahvaloominguga (AutÕS § 5). Rahvaloomingu teoste kaudu säilib meie kultuuripärand. Autoriõiguse seaduse järgi on rahvaloominguteosed vabalt kasutatavad, sest tegemist on üldsusele suunatud teostega. Nendele ei kohaldu autoriõiguslik kaitse, mis tähendab, et kõik huvigrupid peaksid saama Eesti kultuuripärandisse kuuluvaid rahvaloominguteoseid vabalt kasutada. See tagab meie kultuuripärandi ja identiteedi edasikandmise ja levimise [51].

AutÕS § 20 käsitleb teoste vaba kasutamist avaliku arhiivi või muuseumi poolt. Selles on eraldi välja toodud ka fakt, et teose autori nõusolekuta on avalikul arhiivil või muuseumil õigus teost reprodutseerida. See on eelkõige mõeldud selle jaoks, et asendada hävinud või kasutamiskõlbmatuks muutunud teoseid. EKM-i arhiivi paberkandjal olevad noodistused muutuvad ajaga paratamatult kasutuskõlbmatuks. Samuti tohib teoseid säilitamise eesmärgil digitaliseerida (AutÕS § 20 lg 1 p 4). Isiklikud õigused ei ole digitaliseerimise järgselt kuidagi mõjutatud, sest teose sisu selle käigus ei muudeta. [51]

Rahvaviisile on olemasolul lisatud alati autori nimi, teose pealkiri ja loomise või talletamise aeg. Juhul kui see info puudub, siis on tegemist orbteostega. Sellisel juhul on tegemist küll teostega, mis on autoriõigustega kaitstud, kuid mille õiguste omanikku ei ole suudetud tuvastada. Avalikule asutusele on lubatud selliseid teoseid üldsusele kättesaadavaks teha kui see on kultuurilistel eesmärkidel vajalik. Orbteose regulatsioon kehtib teosele, mida talletatakse avalikus arhiivis (nt raamatukogu, teadus- ja haridusasutustes) (AutÕS § 27^{3} lg 3).

EKM kogutud rahvaviiside autoriõigused jagunevad suures plaanis kahte rühma:

1) teosed, millel puuduvad autoriõigused,

2) teosed, mille autoriga peab sõlmima autorilepingu.

Autorileping sõlmitakse kõikide uute autoritega kelle looming jõuab EKM-i arhiivi. Autorileping on kokkulepe teose looja ja teose kasutaja vahel, mille kohaselt autor annab teisele poolele õigused oma teose kasutamiseks (AutÕS § 48 lg 1). Leping peab olema kirjalikus vormis. Selles fikseeritakse teose täpne kirjeldus (vorm, maht, nimetus jms), õigused, mida autor kasutajale üle annab, teose kasutamise viis ja lepingu kehtivuse algus- ja lõpptähtaeg (AutÕS § 48^{1} lg 1 p 1-4).

4.4 Muutused EKM-i töökorralduses

Organisatsiooni siseselt lisandub tarkvara, millega EKM töötab. Süsteem koondab kogu informatsiooni ühte kohta ja lihtsustab viiside haldust, eemaldades vajaduse kasutada Exceli dokumente ning MS Access andmebaasi. See kiirendab tööprotsessi kuna kõik andmed on ühest kohast kättesaadavad.

Uue tarkvara kasutusele võtmine muudab rahvaviiside digitaliseerimise töövoogu. Lisaks viisiga seotud andmete sisestamisele on nüüd võimalik lisada ka viisi kodeering, mis võimaldab viisi juurde kuvada selle noodistuse. Rahvaviisi kodeerimise osa on sellisel kujul uus funktsionaalsus, millega EKM varem tegelenud ei ole. Töövoogu on vaja organisatsioonisiseselt juurutada, et arhivaalide digitaliseerimine sujuks takistusteta. Takistuseks võib olla töötajate teadmatus uue kodeerimise viisi osas.

5 Disain

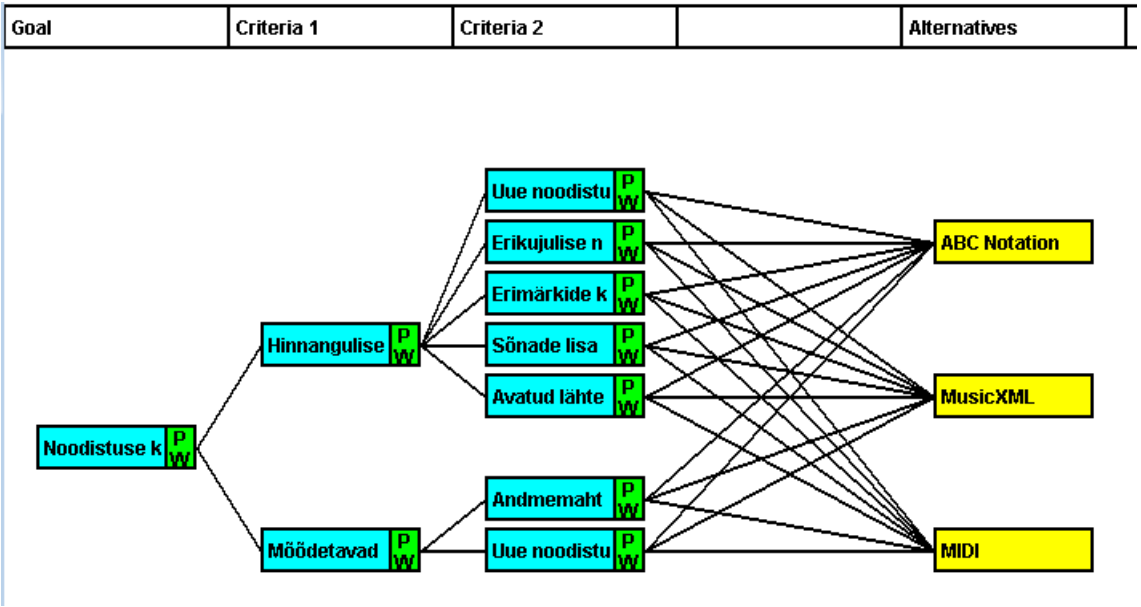
Selles peatükis esitatakse loodava süsteemi tehniline kavand ning samuti tehnilise lahenduse loomiseks kasutatavate vahendite valiku põhjendus.

5.1 Nootide kodeerimise standardi valimine

Nootide kodeerimise standardi valimiseks kasutati Saaty meetodit (vt jaotist 3.5). Alternatiivideks olevaid standardeid kirjeldati jaotises 3.3. Hinnangute andmisel lähtuti järgmistest põhimõtetest.

- Hinnanguliste kriteeriumite puhul kasutati grupiotsustamist. Paariviisiliste võrdluste puhul andsid hinnanguid kõik rühma liikmed ja hinnang määrati tekkiva konsensuse grupiotsustamisega.
- Mõõdetavate kriteeriumite puhul lähtuti hinnangute andmisel mõõtmistulemustest.

5.1.1 Otsustusmudel



Joonis 14 Otsustusmudeli struktuur.

Autorid koostasid Web-Hipre tarkvara kasutades otsustusmudeli struktuuri, mille abil tekkis joonisel (Joonis 14) nähtav hierarhia.

Hinnangulisteks kriteeriumiteks loeti kriteeriumid, mille puhul puudus lihtsalt mõõdetav meetod erinevate alternatiivide hindamiseks. Seetõttu kasutati hinnanguliste kriteeriumite puhul grupiotsustamist.

Mõõdetavateks kriteeriumiteks loeti kriteeriume mille puhul oli võimalik luua mõõdetava tulemusega test ning testi tulemuse põhjal alternatiive hinnata.

5.1.2 Nootide kodeerimise standardite põhikriteeriumite võrdlus

Joonis 15 Nootide kodeerimise standardite hinnanguliste kriteeriumite võrdlus.

Kaal nootide kodeerimisel oli jaotatud selliselt, et hinnangulised kriteeriumid on mõõdukalt tähtsamad kui mõõdetavad kriteeriumid. Selle tingis asjaolu, et hinnangulised kriteeriumid sisaldavad endas tähtsaid funktsionaalseid tingimusi, mis on valiku tegemisel mõõdukalt olulisemad kui mõõdetavad tulemused (Joonis 15).

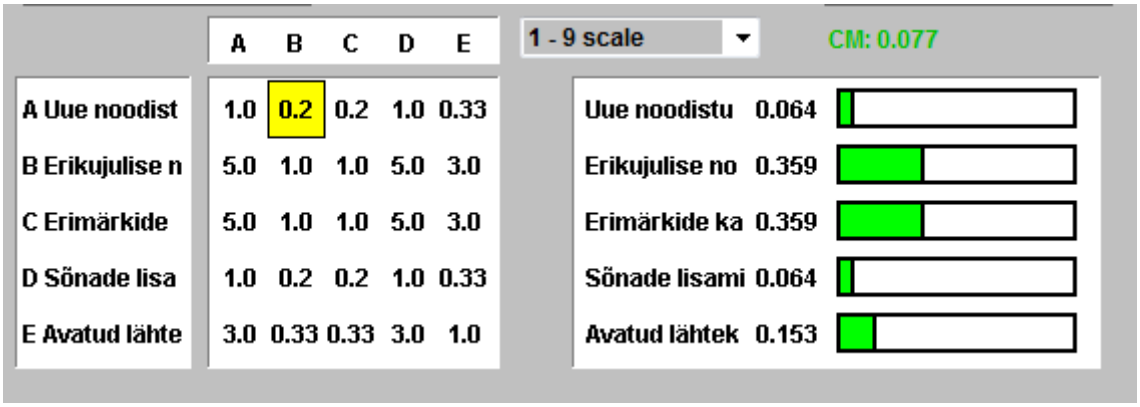
5.1.3 Hinnangulised kriteeriumid ja nende võrdlus

Tabel 3 sisaldab ülevaadet hinnangulistest kriteeriumitest koos nende põhjendustega.

Tabel 3 Hinnangulised kriteeriumid ja nende kirjeldus.

Kriteerium	**Kirjeldus**
Uue noodistuse sisestamise lihtsus	Andmete sisestamise lihtsus on arhiivi omamise puhul tähtis. Arhivaalide loomise puhul on tähtis, et teoste sisestamine oleks võimalikult lihtne ning kiire protsess. Kui teose sisestamine vajab palju täiendavaid interaktsioone ning kasutusel on

Kriteerium	Kirjeldus
	erinevad tarkvarad, siis sellest tulenevalt võib õpikõver olla liiga järsk ja igapäevane töö võib seetõttu olla tülikas.
Erikujulise noodistuse sisestamise võimalikkus	Rahvaviisid ei järgi tavapärast muusikalist struktuuri ning esineb klassikalise muusika põhireeglitele mittevastavat ülesehitust. Sellest tulenevalt on vaja leida lahendus, mis võimaldaks noote kodeerida vastavalt sellele kuidas neid esitati, ilma, et oldaks piiratud noodistuse "korrektse" kujuga.
Erimärkide kasutamise võimalikkus	Noodistuse puhul on erinevate erisuste märkimiseks kasutusel lisamärgid. Kasutusele võetav lahendus peab omama võimalikult laialdast tuge lisamärke noodistuse juures salvestada ja neid ka hiljem kasutajaliideses kuvada.
Sõnade lisamine noodistusele	Kuigi hetkel ei ole noodistuste juurde sõnu lisatud, on sellise funktsionaalsuse omamine EKM-ile tähtis. Sõnade puhul on tähtis, et oleks võimalik sõnu ja silpe siduda täpsete nootidega.
Avatud lähtekoodiga ja tasuta tarkvaraga standard	Kuna soov on antud süsteemi realiseerimiseks kasutada vaid avatud ja tasuta tarkvara ning standardeid, siis on viiside kodeerimise puhul oluline see, et antud lahenduse kasutamine ei tohi vajada täiendavaid oste ei tarkvara ega litsentside näol.



Joonis 16 Hinnangulised kriteeriumid.

Uue noodistuse sisestamise lihtsus *vs.* Erikujulise noodistuse sisestamise võimalikkus – EKM töötajate jaoks on noodistuse sisestamise lihtsus tähtsal kohal (Joonis 16). Sisestatavaid andmeid on palju. Andmete toimetajate ja suuremamahuliste sisestamiste korral uute sisestajate jaoks on noodistamise lihtsus tähtis. Samas tuleb tähtsamaks kriteeriumiks pidada erikujulise noodistuse sisestamise võimalikkust. Seetõttu on erikujulise noodistuse sisestamise võimalikkus oluliselt tähtsam kui protsessi lihtsus.

Uue noodistuse sisestamise lihtsus *vs.* Erimärkide kasutamise võimalikkus – Erimärkide kasutamise võimalikkus rahvaviiside infosüsteemis võimaldab rahvaviisi täpsemalt kodeerida. Samas on erimärke vaja harva kasutada. Seetõttu on erimärkide kasutamise võimalikkus mõõdukalt tähtsam kui uue noodistuse sisestamise lihtsus.

Uue noodistuse sisestamise lihtsus *vs.* Sõnade lisamine noodistusele – Sõnade lisamine noodistusele on funktsionaalsus, mida EKM soovib tulevikus kasutama hakata ning seetõttu on sellise funktsionaalsuse omamine tähtis. Kuna see on aga harva kasutatav funktsionaalsus, siis on see uue noodistuse sisestamise lihtsusega võrdtähtis.

Uue noodistuse sisestamise lihtsus *vs.* Avatud lähtekoodiga ja tasuta tarkvaraga standard – Kui võtta arvesse, et avaliku sektori asutuses on ressursid piiratud ning töökiirusest on tähtsam pikaajaline kulu, siis loeme avatud lähtekoodiga ja tasuta tarkvaraga standardit mõõdukalt tähtsamaks.

Erikujulise noodistuse sisestamise võimalikkus *vs.* Erimärkide kasutamise võimalikkus – Erikujuline noodistus võimaldab noodistust kujutada selliselt nagu teos esitati või noodistuse kirjapanija seda kirjeldas. Erimärgid aga võimaldavad noodistusele lisada detaile, mis aitavad teose taasesitamisel, seda võimalikult täpselt teha. Kuna mõlemad on sama tähtsad aspektid, siis loeme neid võrdtähtsateks.

Erikujulise noodistuse sisestamise võimalikkus *vs.* Sõnade lisamine noodistusele – Erikujulise noodistuse sisestamise võimalikkus on oluliselt tähtsam kuna loodava lahenduse esmane eesmärk on võimaldada detailset noodistust ja sõnade lisamise võimalus on sellega võrreldes oluliselt vähemtähtsam.

Erikujulise noodistuse sisestamise võimalikkus *vs.* Avatud lähtekoodiga ja tasuta tarkvaraga standard – Erikujulise noodistuse sisestamine on oluliselt tähtsam. Juurutatava standardi toetatav funktsionaalsus on tähtsam selle kasutamise kulust.

Erimärkide kasutamise võimalikkus *vs.* Sõnade lisamine noodistusele – erimärkide kasutamine on oluliselt tähtsam kui sõnade lisamine. Ilma erimärkideta ei ole võimalik noodistust korrektselt kirja panna.

Erimärkide kasutamise võimalikkus *vs.* Avatud lähtekoodiga ja tasuta tarkvaraga standard – Erimärkide kasutamise võimalikkus on oluliselt tähtsam. Juurutatava standardi toetatud funktsionaalsus on tähtsam selle kulust.

Sõnade lisamine noodistusele *vs.* Avatud lähtekoodiga ja tasuta tarkvaraga standard – kuna sõnade lisamise näol on tegemist lisafunktsionaalsusega, siis võrreldes avatuse ja tasu puudumisega on see mõõdukalt vähemtähtis.

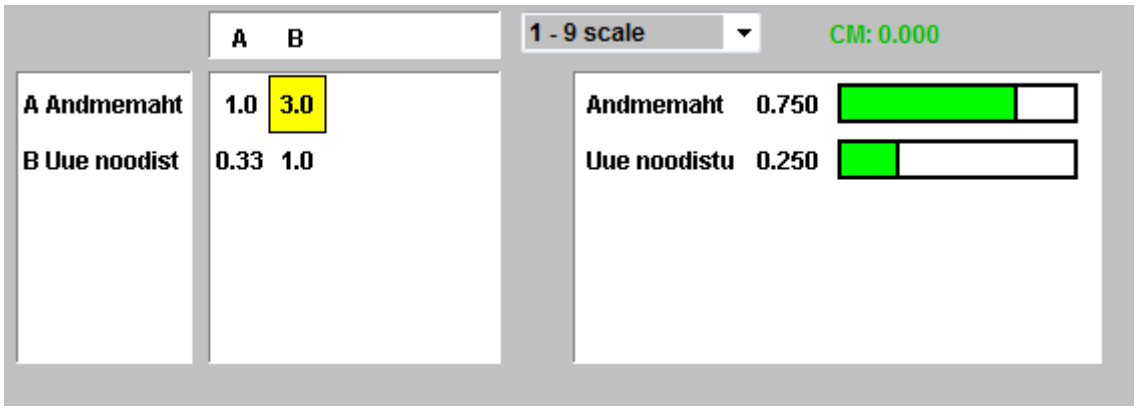
5.1.4 Mõõdetavad kriteeriumid ja nende võrdlus

Tabel 4 sisaldab ülevaadet mõõdetavatest kriteeriumitest koos nende põhjendustega.

Tabel 4 Mõõdetavad kriteeriumid ja nende võrdlus.

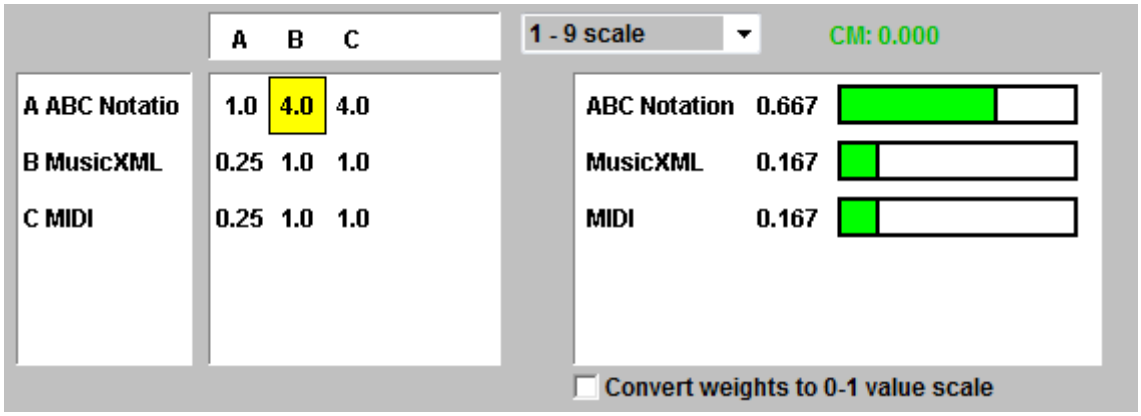
Kriteerium	**Kirjeldus**
Andmemaht	EKM arhiivi eeldatav suurus on üle 40 000 arhivaali ning iga arhivaali kohta võib tekkida andmebaasi n+1 kirjet. Kirjete mahtu suurendavad variatsioonid. Seega eeldatav kirjete hulk on suhteliselt suur. Seetõttu on tähtis jälgida loodava lahenduse tuleviku andmemahtu.
Uue noodistuse sisestamise kiirus	Mõõdetav parameeter mis mõjutab kasutajakogemust ning andmete sisestamist on uute kirjete lisamise kiirus. Seetõttu on tähtis mõõta kui kiiresti on võimalik uut noodistust sisestada.

Andmemaht *vs.* Uue noodistuse sisestamise kiirus – Kuna luuakse arhiivi, mille eesmärk on andmete pikaajaline säilitamine, siis on andmemaht mõõdukalt tähtsam kui uue noodistuse sisestamise kiirus (Joonis 17)



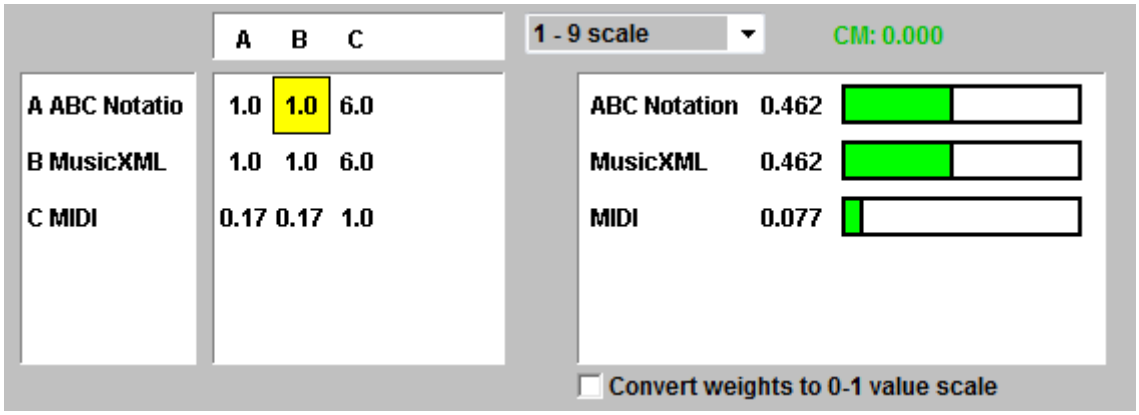
Joonis 17 Mõõdetavad kriteeriumid.

5.1.5 Võrdlus hinnanguliste kriteeriumite alusel



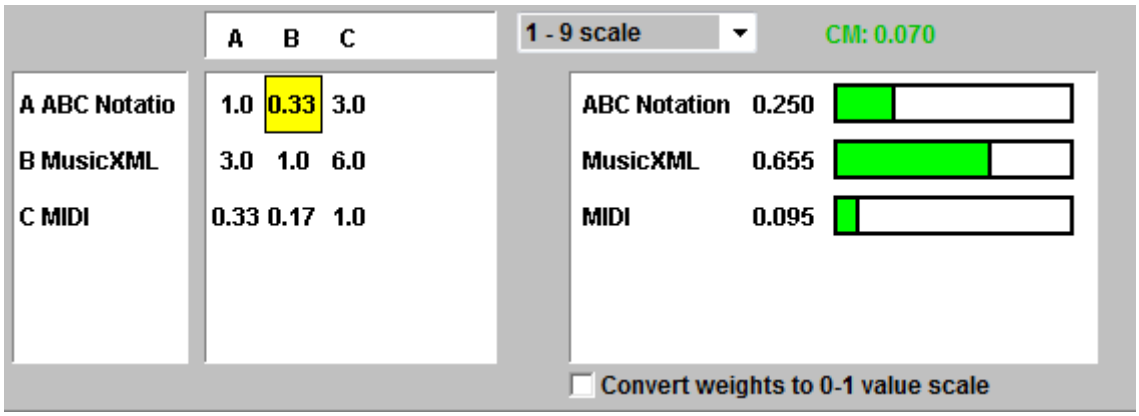
Joonis 18 Alternatiivide võrdlus uue noodistuse sisestamise lihtsuse alusel.

Joonis 18 näitab, et uue noodistuse sisestamise puhul hindasime ABCd kõige kõrgemalt kuna selle puhul on võimalik sisestada noodistust kasutades ASCII märke ning puudub vajadus täiendavate väliste rakenduste järgi. Erimärgistus ja muu analoogne on lisatav kasutades lihtsat süntaksi ning ABC notatsiooni standardil on põhjalik dokumentatsioon, kus on kirjeldatud kogu vajalik kodeering. [52] MIDI ja MusicXML märgiti võrdväärseteks kuna mõlemal puhul on tarvis kasutada välist tarkvara ning erinevus noodistuse sisestamise lihtsuses tuleb kasutatavast tarkvarast, mitte standardist.



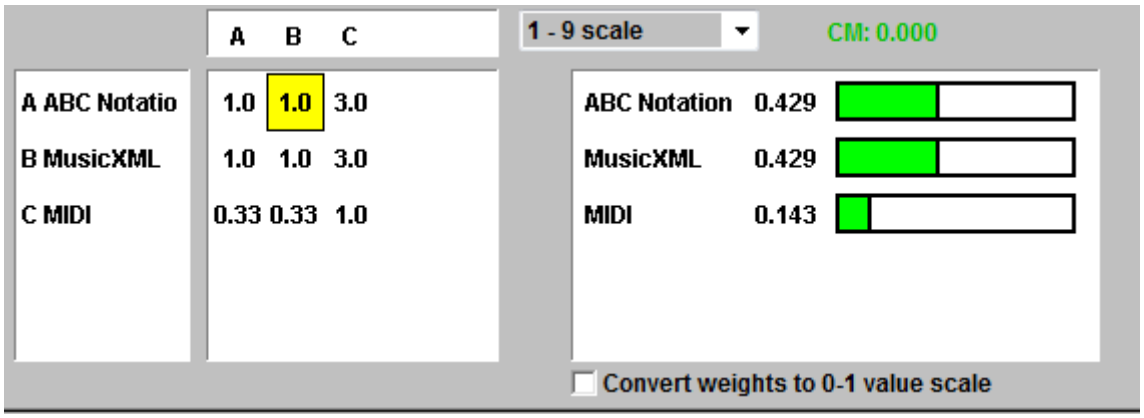
Joonis 19 Alternatiivide võrdlus erikujulise noodistuse sisestamise võimalikkuse alusel.

Joonis 19 näitab, et ABC ja MusicXML on erikujulise noodistuse lisamise osas samaväärseid võimalusi pakkuvad, mistõttu on nende paarile Saaty skaala järgi hindeks antud 1 ehk võrdselt oluline. MIDI ei võimalda noote piisava täpsusega üles märkida. Näitena võib tuua, et süsteem ei tee vahet näiteks ½ ja ¼ nootidel. Samuti on MIDI puuduseks see, et see ei võimalda teha vahet erinevatel nootidel, vaid väljendab vaid nende kõrgust. See tähendab, et näiteks noodid f-diees ja g-bemoll on MIDI puhul samasugused.



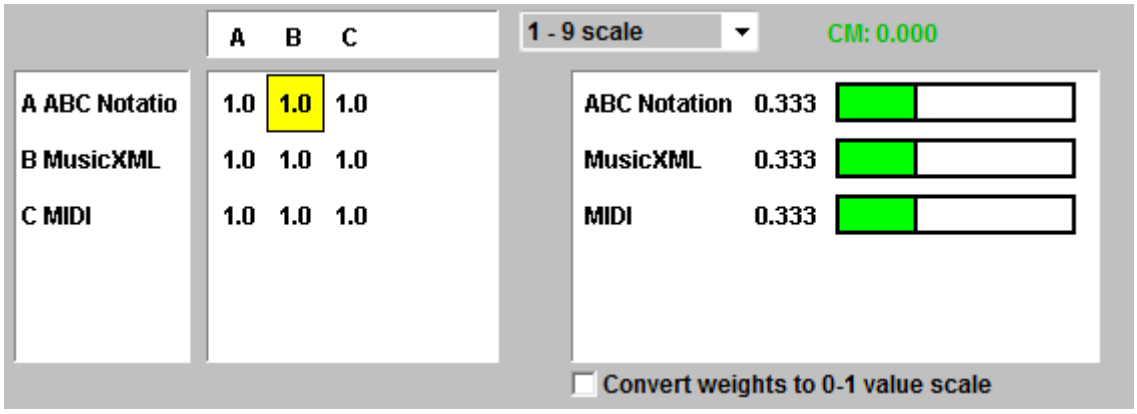
Joonis 20 Alternatiivide võrdlus erimärkide kasutamise toe alusel.

Joonis 20 näitab, et erimärkide kasutamist toetab kõige enam MusicXML [53]. Võrreldes ABC-ga ja MIDI-ga on MusicXML Saaty skaala järgi väga tugevalt eelistatud. Samas, kui võrrelda ABC kodeeringut MIDI-ga, siis on ABC tugevalt eelistatud.



Joonis 21 Alternatiivide võrdlus sõnade noodistusele lisamise alusel.

Joonis 21 näitab, et kuigi sõnade lisamine on võimalik kõikide kodeeringute puhul ei õnnestunud testimise (jaotis 5.1.6) ja teisendamise käigus MIDI vormingu puhul sõnade kaasa toomine. Seetõttu, kuigi põhimõtteliselt on seda võimalik teha, ei ole see sama lihtne ega mugav nagu MusicXML ja ABC puhul.

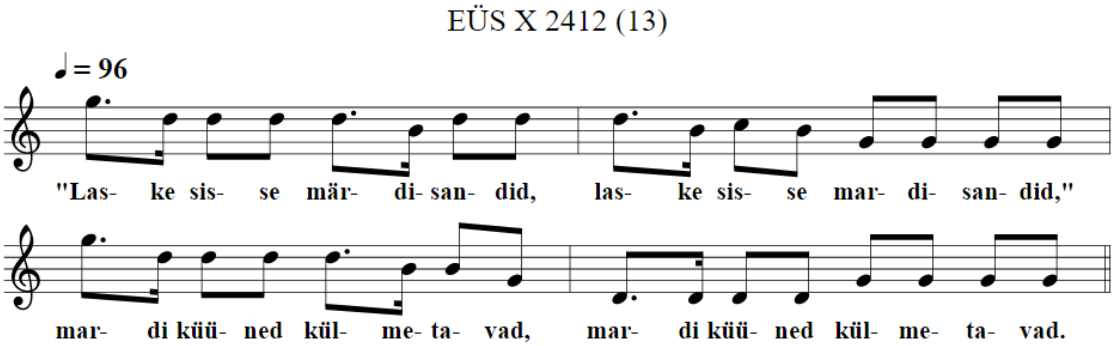


Joonis 22 Alternatiivide võrdlus avatud lähtekoodiga ja tasuta tarkvaraga standardi alusel.

Joonis 22 näitab, et kõiki valimisse sattunud kodeeringuid saab kasutada tasuta ja avatud lähtekoodiga programmide abil. ABC standardis muusika kodeerimiseks sobib igasugune tekstiredaktor. Selleks, et kodeeritavat noodikirja ka visualiseerida, on võimalik kasutada vabavaralist rakendust nagu näiteks EasyABC [54] MusicXML ja MIDI vormingus muusika kirjutamiseks saab kasutada MuseScore [55] nimelist tarkvara, mis on vabaks kasutamiseks “GNU General Public License” [56] litsentsi alusel. MuseScore salvestab küll andmeid oma formaadis, kuid omab MIDI/MusicXML formaadi tuge import/eksport funktsioonide kaudu. Täiendavalt on MuseScore’il ka pistikprogramm ABC kodeeringus muusika importimiseks. [57]

5.1.6 Võrdlus mõõdetavate kriteeriumite alusel

Kuna noodistusi on vaja hoiustada andmebaasis või failisüsteemis, siis on tähtsaks kriteeriumiks iga noodistuse andmemaht. Antud hinnangu andmiseks loodi näidisviis ning selle andmemahtu võrreldi iga paari puhul eraldi. Näidisviisiks on mardilaul, mille me saime EKM-ilt (Joonis 23).



Joonis 23 Näidisviis andmemahu võrdlemiseks.

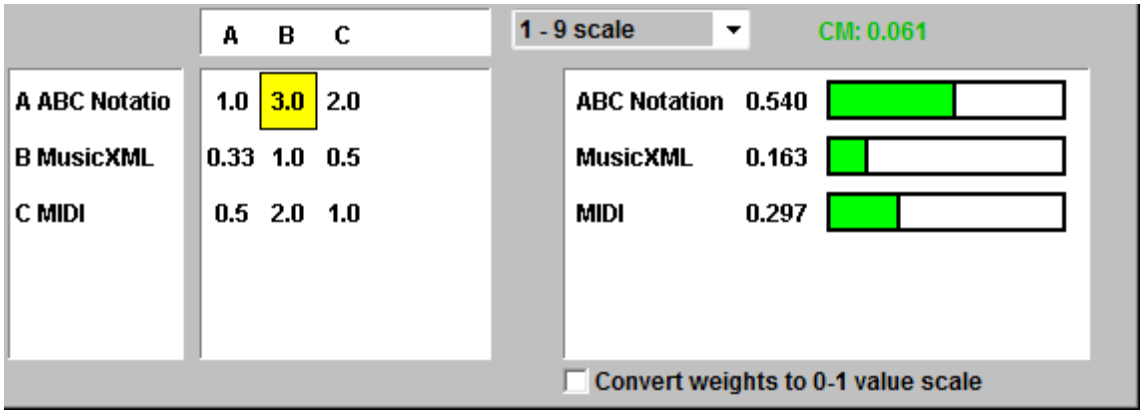
Antud viisi kasutati hindamaks sellise viisi esituse andmemahtu erinevates kodeeringutes.

Tabel 5 Viisi esituse andmemahud erinevates kodeeringutes.

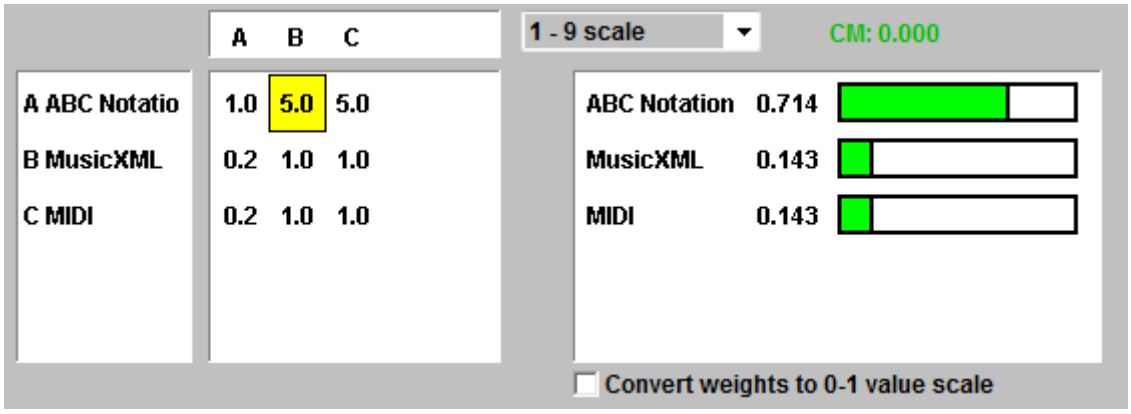
Kodeering	**Suurus baitides**
ABC	274B
MusicXML	14935B
MIDI	291B (märkusena ei õnnestunud teksti juurde lisada, seega koos tekstiga on eeldatav maht suurem; samuti oli MIDI-ks teisendamise tulemusest osa informatsioonist kadunud ning MIDI seetõttu kasutamatu)

Vastavalt eelpooltoodud tulemustele näitab Tabel 5, et ABC kodeeringus esitus on üle 50 korra väiksema andmemahuga kui MusicXML. MIDI on küll tulemuste järgi vaid 5% suurem kui ABC, kuid MIDI vormingusse ei õnnestunud testimise käigus sõnu juurde lisada. MIDI ja MusicXML puhul kasutati tarkvara MuseScore, milles viis kodeeriti ning siis kasutati eksportimise funktsiooni, et teha teisendus vastavas kodeeringus esituseks.

Hilisema faili avamise käigus aga avastati, et MIDI faili puhul puudusid sealt sõnad. Seetõttu andsime hinnangud, mida on näha joonisel Joonis 24.



Joonis 24 Alternatiivide võrdlus andmemahu alusel.



Joonis 25 “Uue noodistuse sisestamise kiirus” Saaty skaalal hinnangud.

Joonis 25 on näha uue noodistuse sisestamise kiiruse võrdlus. Mõõtmiseks sisestas üks töö autoritest Joonis 23 näha oleva noodistuse kasutades ABC kodeeringu jaoks tekstiredaktorit ja MIDI/MusicXML kodeeringu jaoks MuseScore tarkvara, mis võimaldab graafilise liidese abil noodistust sisestada. Enne mõõtmist harjutas see autor mõlema tarkvara peal noodistuse sisestamist. Sisestamise kiirust tuleb mõlema lähenemise puhul pidada algaja sisestamise kiiruseks, kuna testijal puudus eelnev kogemus nii ABC kodeeringu kasutamise kui MuseScore tarkvara kasutamisega. Testi tegemiseks olid autoril olemas nii noodipilt kui nootide kõrgused ja spikker erinevate nootide kujutamiseks. Enne aja mõõtmist harjutas autor mõlema lahenduse puhul sisestamist kolmel korral. Seetõttu võib võrdlust lugeda võrdsetel alustel tehtud võrdluseks, mille käigus ei olnud kumbki lahendus eelisseisus.

Mõõdetud tulemused on toodud tabelis Tabel 6.

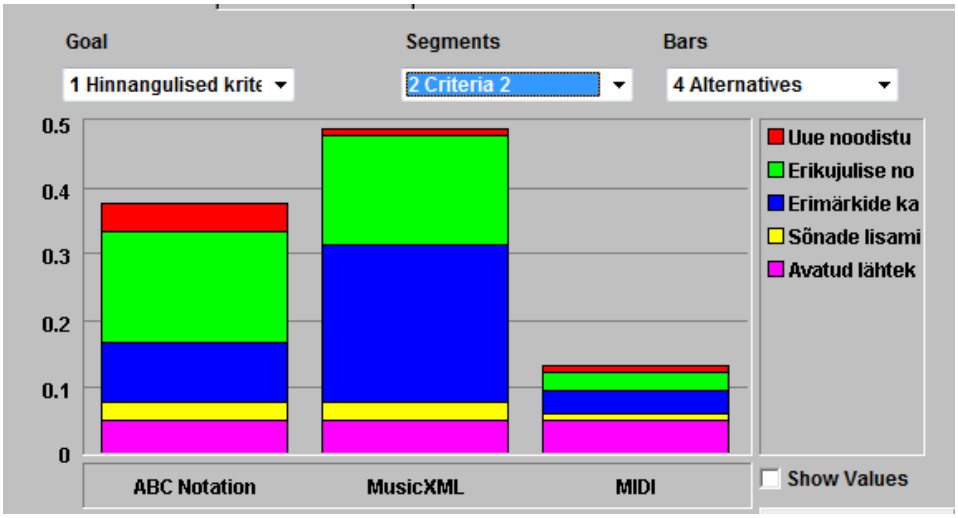
Tabel 6 Uue noodistuse sisestamise kiirused.

Standard	**Aeg**
ABC notatsioon	1min 24sek
MusicXML ja MIDI	2min 08sek

Vastavalt tabelis väljatoodud aegadele on näha, et MusicXML ja MIDI kodeeringus nootide sisestamise kiirus on 52% väiksem kui ABC kodeeringus. Seetõttu on ABC oluliselt parem kui MusicXML ja MIDI. MusicXML ja MIDI on omavahel võrdluses selles aspektis samaväärsed.

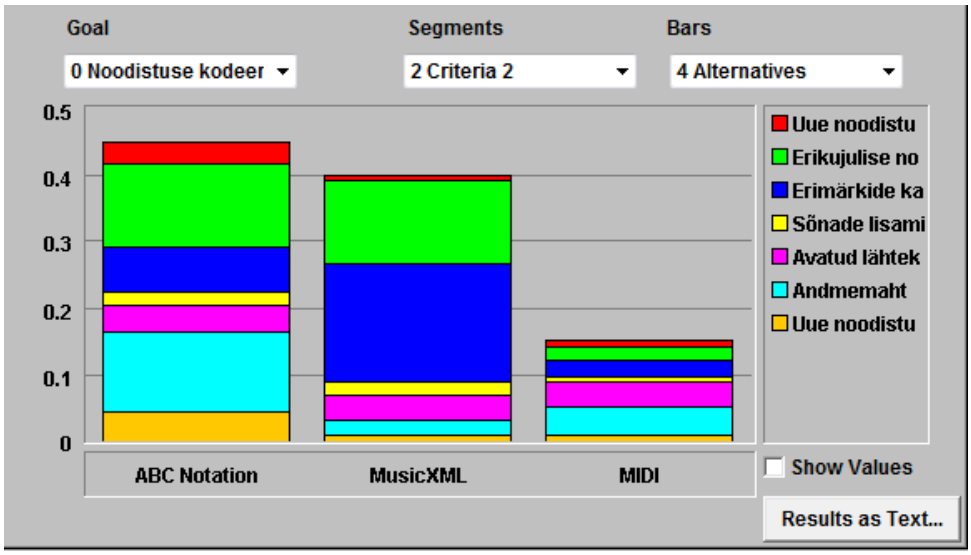
5.2 Saaty meetodil otsustamine

Järgnevalt analüüsitakse otsustusmudeli tulemusi.



Joonis 26 Hinnangulised kriteeriumid.

Kui me vaatleme ainult hinnangulisi kriteeriumeid, siis on näha, et MusicXML on alternatiividest kõige sobivaim valik (Joonis 26). Lähedal MusicXML-ile on ka ABC notatsioon, mis on funktsionaalsuselt veidi kehvem just erimärkide kasutamise osas, kuid väga lähedal liidripositsioonile. Kui me aga võtame arvesse ka mõõdetavad kriteeriumid, siis pilt muutub.

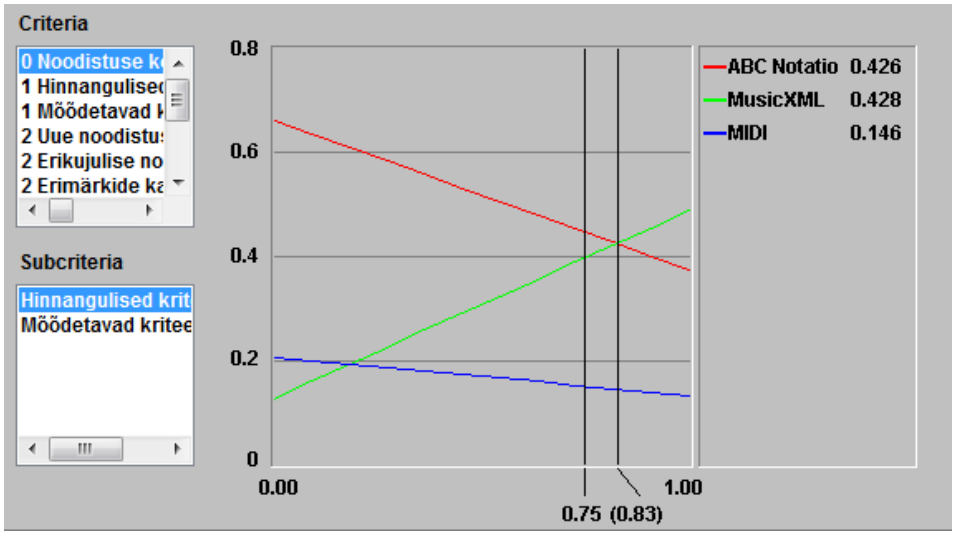


Joonis 27 Otsustusmudeli rakendamise tulemused.

Kui me vaatleme nii hinnangulisi kriteeriumeid kui ka mõõdetavaid kriteeriumeid, siis on näha, et MusicXML ei ole enam kõige sobivam variant (Joonis 27). See tuleneb sellest, et andmesalvestusformaadina on MusicXML vajaminev andmemaht palju suurem ning seetõttu ei ole see antud projekti raames kasutusele võtmiseks kõige mõistlikum standard. MusicXML-i võlu seisneb suurte ja keerukate heliteoste kirjeldamises, mille puhul ABC notatsioon võib jääda hätta. Võrreldes ABC notatsiooni kasutamisel tekkivat andmemahtu MusicXML-iga oli erinevus 54 kordne ABC notatsiooni kasuks. MIDI tulemusi varjutasid tema muud probleemid (vt Tabel 5).

Jooniselt on võimalik järeldada, et otsustusmudeli abil osutus parimaks alternatiiviks ABC notatsioon.

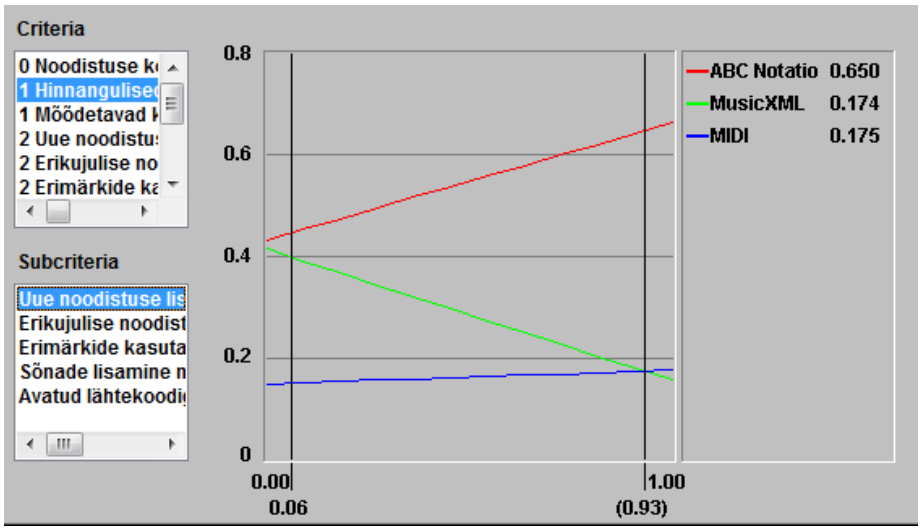
5.2.1 Põhikriteeriumite tundlikkuse analüüs



Joonis 28 Põhikriteeriumite tundlikkuse analüüs.

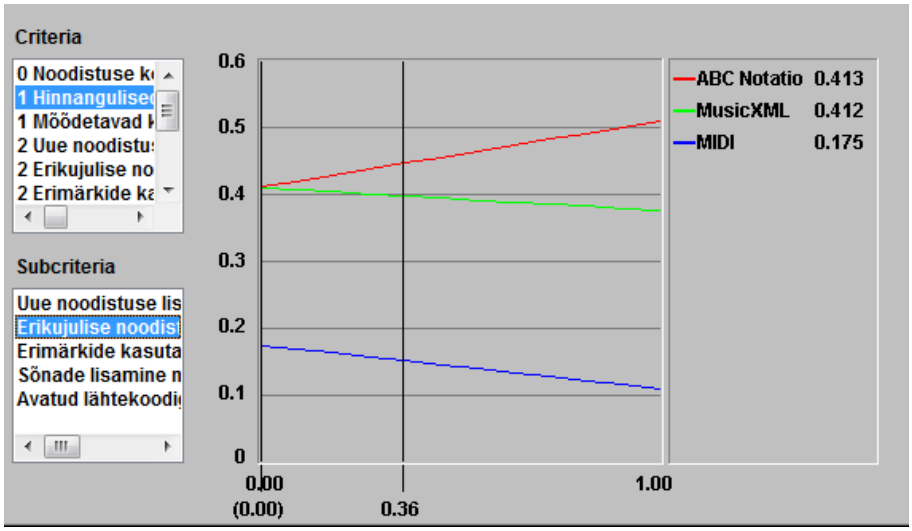
Joonis 28 on näha põhikriteeriumi tundlikkuse analüüsi graafik, mille abil on meil võimalik järeldada, et hinnanguliste kriteeriumite osatähtsuse muutmisega on meil võimalik muuta tulemust. Kui kriteeriumi osatähtsust tõsta 0.75 pealt 0.83 peale, siis on ABC notatsiooni asemel võitjaks hoopis MusicXML. Isegi peale sellist muutmist on MIDI igal juhul mittesobilik. Erinevuse leidmiseks leidsime esiteks kaalude suhte enne muudatust milleks on 0,75/(1-0,75) = 3 ja peale muudatust 0,83/(1-0,83) = 4,9. Selleks, et MusicXML muutuks võitjaks, tuleks hinnanguliste kriteeriumite tähtsust tõsta ümardatult 4,9/3 = 1.6 korda.

5.2.2 Alamkriteeriumite tundlikkuse analüüs



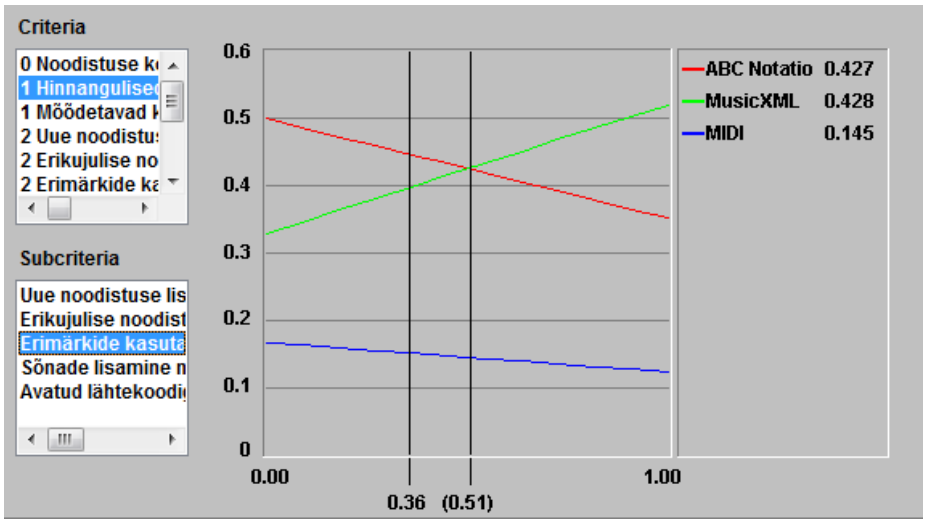
Joonis 29 “Uue noodistuse sisestamise kiirus” kriteeriumi tundlikkuse analüüs.

Joonis 29 “Uue noodistuse sisestamise kiirus” kriteeriumi tundlikkuse analüüs näitab uue noodistuse sisestamise kiiruse tundlikkuse analüüsi graafikut. Jooniselt järeldub, et juhul kui tõsta uue noodistuse sisestamise kiirust 0.06 pealt 0.93 peale ei muutu küll võitja, kuid muutub teise ja kolmanda koha järjestus ja MIDI on teise valikuna eelistatum MusicXML ees. Erinevuse leidmiseks leidsime esiteks kaalude suhte enne muudatust milleks on 0,06/(1-0,06) = 0,06 ja peale muudatust 0,93/(1-0,93) = 13,28. Selleks, et MIDI saaks teise koha, tuleks kriteeriumi osatähtsust suurendada ümardatult 13,28/0,66 = 208 korda.



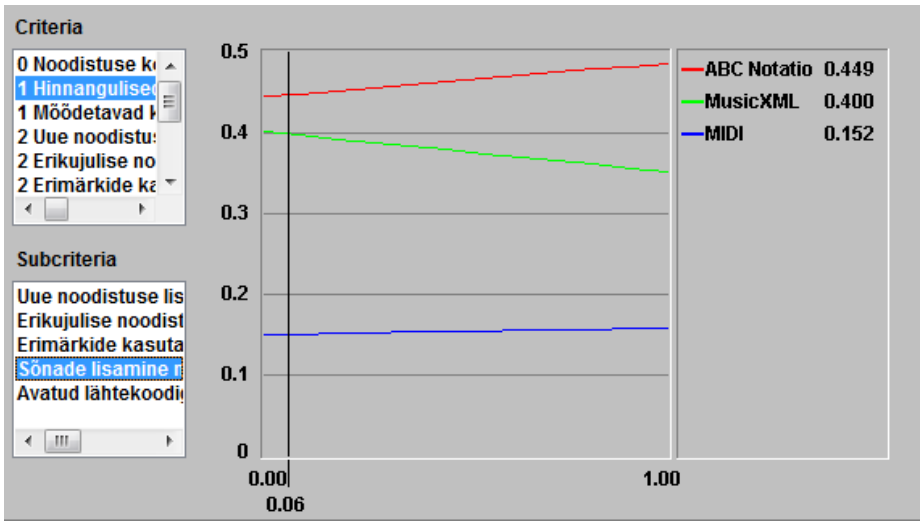
Joonis 30 “Erikujulise noodistuse sisestamise võimalikkus” kriteeriumi tundlikkuse analüüs.

Joonis 30 näitab, et isegi juhul kui muuta erikujulise noodistuse sisestamise osatähtsust 0.00 peale, siis osutuks ABC notatsioon võitjaks. See tähendab, et isegi kui erikujulise noodistuse sisestamise võimalikkuse välja jätta osutuks võitjaks ABC notatsioon kuigi erinevus on 0.001



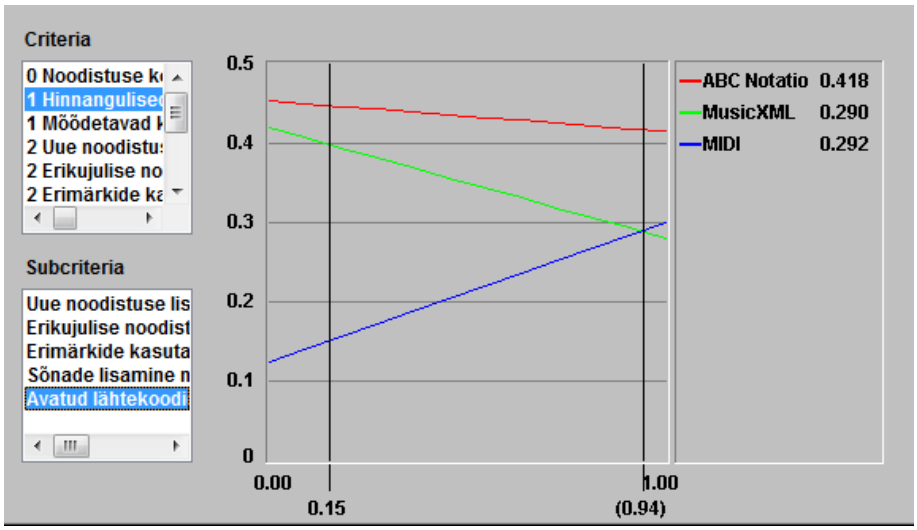
Joonis 31 “Erimärkide kasutamise võimalikkus” kriteeriumi tundlikkuse analüüs.

Joonis 31 näitab, et juhul kui tõsta erimärkide kasutamise osatähtsust 0.36 pealt 0.49 peale, siis muutub sobivaimaks alternatiiviks MusicXML. Erinevuse leidmiseks leidsime esiteks kaalude suhte enne muudatust milleks on 0,36/(1-0,36) = 0,56 ja peale muudatust 0,51/(1-0,51) = 1,04. Selleks tuleks erimärkide kasutamise osatähtsust suurendada ümardatult 1,04/0,56 = 1.85 korda.



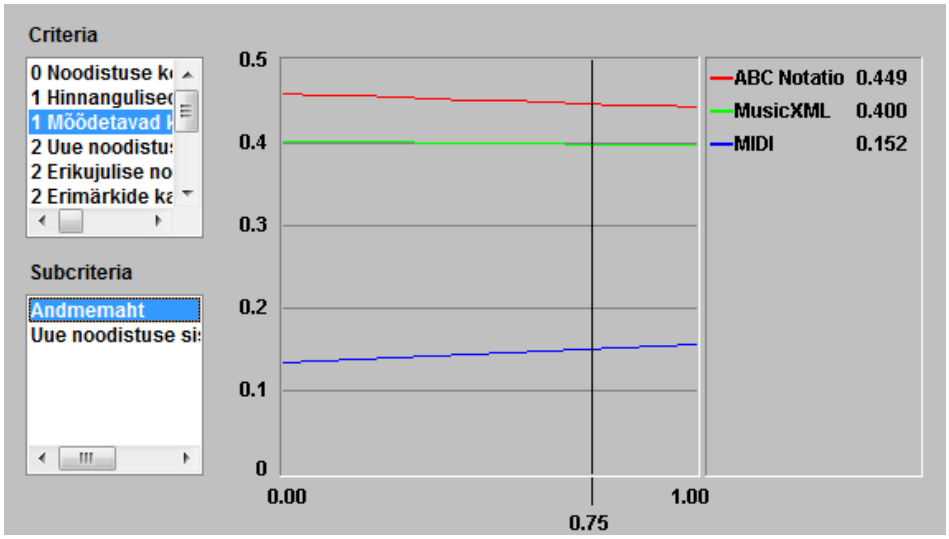
Joonis 32 “Sõnade lisamine noodistusele” kriteeriumi tundlikkuse analüüs.

Joonis 32 näitab, et sõnade lisamise osakaalu muutmine valiku järjestust ei muuda



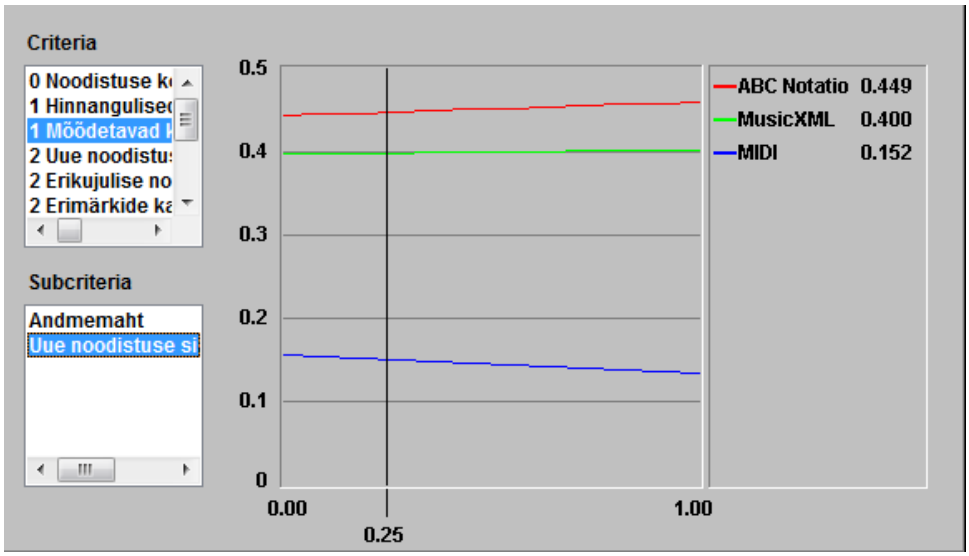
Joonis 33 “Avatud lähtekoodiga ja tasuta tarkvaraga standard” kriteeriumi tundlikkuse analüüs.

Joonis 33 näitab, et juhul kui muuta ilma litsentsita kasutamise võimaluse osatähtsus 0,15 pealt 0,94 peale, ei muutuks võitja, kuid muutub teise ja kolmanda koha järjestus ja MIDI on teise valikuna eelistatum MusicXML ees. Erinevuse leidmiseks leidsime esiteks kaalude suhte enne muudatust milleks on 0,15/(1-0,15) = 0,18 ja peale muudatust 0,94/(1-0,94) = 15,67. Selleks tuleks erimärkide kasutamise osatähtsust suurendada ümardatult 3,35/0,09 = 88,8 korda.



Joonis 34 Andmemahu kriteeriumi tundlikkuse graafik.

Joonis 34 näitab, et andmemahu osakaalu muutmine valiku järjestust ei muuda.



Joonis 35 Laadimisaja kriteeriumi tundlikkuse graafik.

Joonis 35 näitab, et sõnade andmete laadimisaja osakaalu muutmine valiku järjestust ei muuda.

5.3 Esitluskihi platvormi valik

Antud töö autoritel on kõige rohkem kogemusi platvormiga ReactJS. Veendumaks, et pole alternatiive, millel on selgelt tugev eelis selle platvormi ees, uuriti ka teisi platvorme. Alternatiivide leidmiseks kasutati populaarse teadmiste jagamise platvormi Stackoverflow iga-aastase arendajate seas läbi viidava uuringu 2020. aasta tulemusi. [58] Sellest selgus, et neli kõige armastatumat veebiarendus platvormi on .NET Core, ReactJS, Vue.js ning Express.

Esitluskihi valimisel otsustasid autorid kõrvale jätta .NET Core ning Express platvormid kuna nende puhul on tegu platvormidega, mis ei eralda rakendustena esitluskihti ning lüüsikihti, vaid serveerivad kasutajaliidese ning vajalikud otspunktid ühest rakendusest. Antud töö eesmärgiks on luua süsteem, mida saaks kasutada ka programmatiliselt päringute tegemiseks, mistõttu sobivad paremini platvormid, kus otspunktid on avaldatud põhifunktsionaalsusena, mitte tagantjärgi lisatuna.

ReactJS ning Vue.js vahel valides leiti Vue.js dokumentatsioonist artikkel mis väidab, et mõlemad kasutavad virtuaalset DOM-i, pakuvad vaate komponente ning keskenduvad teegi tuumikule, jättes probleemid nagu marsruudi valik ning globaalse oleku haldamine

kolmandate osapoolte pistikutele. [59] Lisaks öeldakse, et ReactJS ning Vue.js on skoobilt sarnased ning nenditakse, et ReactJS kasutajaskond on laiem.

Arvestades autorite kogemuste ning leidudega otsustati jätkata ReactJS platvormiga.

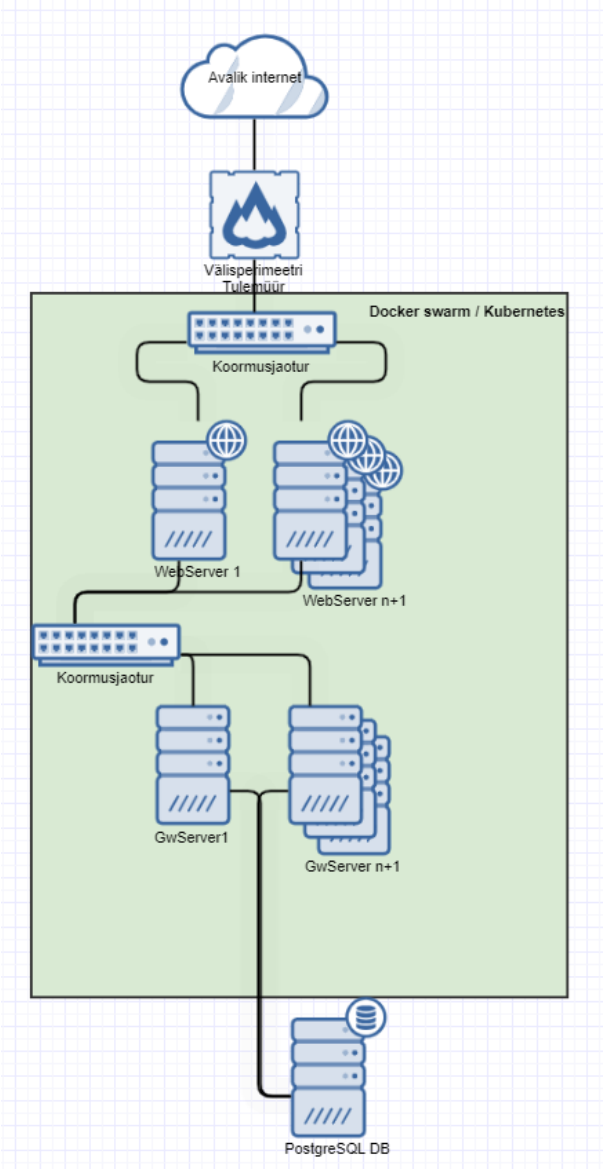
5.4 Lüüsikihi platvormi valik

Nagu ka esitluskihi puhul, selgus, et kõige rohkem kogemust on autoritel Loopback platvormiga. Sarnaselt esitluskihi alternatiividega leidsid autorid eelmainitud uuringust võimalikeks lüüsikihi platvormi alternatiivseteks kandidaatideks Loopback kõrvale .NET Core, Spring Boot ja Express platvormid.

Express platvorm on minimalistlik baasplatvorm, millega saab vahevarasid lisades kiiresti süsteeme ehitada. Kuna aga Express ei toeta otseselt andmetüüpide kontrollimist ning andmebaasiga suhtlust, siis leiti, et on efektiivsem kasutada platvormi, mis pakuks neid võimalusi kohe alguses, mitte kolmandate osapoolte pistikutena. Sellel põhjusel jäeti valikust Express kõrvale.

Valides allesjäänud kolme variandi vahel ei leidnud autorid ühtegi selget eelist, mis oleks ühe platvormi teiste ees esile tõstnud. Põhiline erinevus seisnes programmeerimiskeeles ning seetõttu otsustati jääda juba tuttava Loopback platvormi juurde.

5.5 Uue lahenduse tehniline arhitektuur



Joonis 36 Tuleviku visioon tehnilisest disainist.

Magistritöö raames loodav infosüsteemi tarkvara (Joonis 36) on selliselt üles ehitatud, et seda oleks võimalik tulevikus vajaduse korral kiiresti ja lihtsalt horisontaalselt skaleerida. Kuigi tegemist on infosüsteemiga, mille eeldatav üheaegne kasutajate hulk on väike, on võimalik loodud lahendust ka mujal kasutada ning loodav lahendus võimaldab põhimõtteliselt sellist lähenemist kasutada. Uus lahendus vastab ISKE nõudele M 5.169 Veebirakenduse süsteemiarhitektuur [60] ja koosneb kolmest kihist.

1. Esitluskiht – esitluskiht on ehitatud selliselt, et see on olekuvaba (*stateless*), mis tähendab, et veebikihis ei hoiustata andmeid, mida on süsteemi toimimiseks vaja. Veebikiht on esitluskiht, mis kasutab veebilüüsi andmebaasist andmete

pärimiseks. Sellise lähenemise tulemusel on võimalik soovi korral, horisontaalset skaleerimist rakendades, paralleelselt kasutada n+1 veebikihi rakendusserverit, ilma, et andmete kasutamisel/salvestamisel tekiks konflikte. Tulenevalt esialgsest vähesest kasutajate hulgast rakendatakse lahenduses põhimõtet n=0, mis tähendab, et veebikihi rakendusservereid on ainult üks.

2. Lüüsikiht – Lüüsikihi eesmärk on esitluskihi ja andmebaasi vahel andmete edastamine ja teisendamine selliselt, et oleks tagatud kõik nii turvalisuse kui esitluskihi jaoks vaja olevad nõuded. Turvalisuse tagamiseks rakendab lüüsikiht nii autentimist kui rollipõhist autoriseerimist iga piiratud suhtluse korral ja lisaks talletab ka andmebaasis andmete muutmiseks käivitatud lause ning selle teinud kasutaja. Lüüsikihte on võimalik samuti suuremale koormusele vastu pidamiseks horisontaalselt skaleerida. Antud lähenemise puhul on taaskord kasutusel n+1 loogika, mille tulemusena on võimalik vastavalt süsteemide koormusele süsteemi dünaamiliselt skaleerida. Tulenevalt esialgsest vähesest kasutajate hulgast rakendatakse lahenduses põhimõtet n=0, mis tähendab, et lüüsikihte on ainult üks.

3. Andmebaasikiht – Andmebaasina on antud süsteemis kasutusel PostgreSQL andmebaas. Kuna andmete terviklus ja turvalisus on kõrged prioriteedid, siis on andmebaasi ees lisaturbe pakkumiseks lüüsikiht, mis aitab vähendada andmebaasimootori võimalike turvanõrkuste vastu suunatud rünnakute riski. Andmebaasiga suhtlemiseks luuakse andmebaasi kasutaja, kellele antakse täpselt nii vähe õiguseid kui funktsioonide täitmiseks on vaja. Kuna loodav rakendus hakkab kasutama olemasolevat EKM-i andmebaasi, siis antud töö raames eeldavad autorid, et andmebaas on korrektselt turvatud ning töös täiendavalt sellele tähelepanu ei pöörata.

5.6 Olemasoleva andmebaasi nootide ABC kujule teisendamise ja kodeerimise põhimõtted

Selleks, et võimaldada EKM-il rahvaviiside andmebaasi kasutada ühtsel ja uut standardit järgival põhimõttel, ilma juba tehtud tööd kaotamata, oli vaja kirjeldada põhimõtted, mille alusel on võimalik olemasolev andmestik viia ABC standardit järgivale kujule. Enne töö alustamist uuriti, kas on juba olemas mingi rakendus, mille abil oleks võimalik teisendust automatiseerida kuid seda ei leitud. Olemasolev andmestik oli kodeeritud peaasjalikult

vastavalt I. Rüütli koostatud noodistuste formaliseerimise põhimõttele. Selle alusel oli loodud töö allikmaterjaliks olnud MS Access andmebaas, mille struktuur on näha lisas 2. Selles andmebaasis hoiti viiside andmeid tabelis nimega *Viis*, mille veerge on võimalik grupeerida järgnevalt (Tabel 7):

Tabel 7 MS Access andmebaasi tabelite veerud.

Funktsioon	**Veergude nimetused**
Väärtused seoste loomiseks	IDviis, FKey, VNr, Nr
Helilised tunnused	võti, alter, tugiheli, kõrgus, tempo, takt
Helikõrguste positsioonid	P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16
Märkused	vMärkusi, viis

Andmebaasi tabelis *Viis* oli kokku 4941 rida, millest üle 95% olid vastavalt andmebaasi loomise põhimõtetele korrektselt sisestatud. Umbes 5% oli aga selliseid ridu, mis sisaldasid vaid kommentaare või olid kirjeldatud vastavalt heli muutusele. See tähendab, et iga noodi kõrguse asemel oli märgitud viis varieeruvusarvudena.(Joonis 37)

-4	[-4][-2]	2	0	2	2	0	-4	-4	-4	2	0	2	2	0	0
[-3][3]	[-3][3]	4	5	4	2	0	-3	[-3][3	[-3][:	4	5	4	2	0	0
3	3	3	3	3	3	0	-4	3	3	3	3	3	3	0	0
-3	-3	4	4	2	2	0	-3	-3	-3	4	4	2	2	0	0

Joonis 37 Pilt andmebaasi tabelist "Viis" mitte sobiva kodeeringuga.

Kuna selliste viiside hulk oli väike ning puudus täpne arusaam, mis põhimõtetel selline kodeering loodi, jäeti sellised read teisendamata.

2e	2d	2d	2c	2c	e	e	g	g	a	a	h[g]	h[g]	2c	2c
2e	2d	2d	2c	2c	e	e	g	g	g	g	2c	2c	2c	2c
g	a	f	e	d	d	e	f	f	g	f	e	d	g	g

Joonis 38 Pilt andmebaasi tabelist "Viis" korrektsete kodeeringutega.

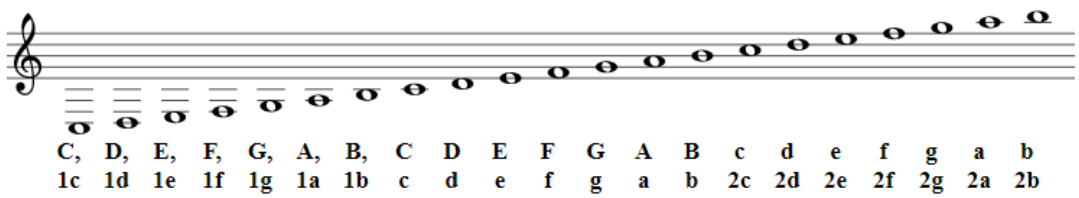
Korrektselt kodeeritud viiside puhul (Joonis 38) oli igale positsioonile märgitud vähemalt üks täht, mis tähistas antud positsioonil oleva noodi kõrgust. Vastavalt sellele

informatsioonile ja I. Rüütli poolt sõnastatud põhimõtetele oli võimalik koostada vastavustabel, mille alusel teisendada noodistus ühest vormist teise. Noodistuse puhul tähistas täht noodi kõrgust, eesliide 2 tähendas, et tegemist on ühe oktavi võrra kõrgema noodiga ja eesliide 1 tähendas, et tegemist on ühe oktavi võrra madalama noodiga. Selle info põhjal oli võimalik luua järgnev teisenduse loogika.

Tabel 8 Noodistuse teisendamise loogika.

Esialgne tähis	Uus tähis	**Esialgne tähis**	Uus tähis	**Esialgne tähis**	Uus tähis
a	A	**1a**	A,	**2a**	a
b	B	**1b**	B,	**2b**	b
c	C	**1c**	C,	**2c**	c
d	D	**1d**	D,	**2d**	d
e	E	**1e**	E,	**2e**	e
f	F	**1f**	F,	**2f**	f
g	G	**1g**	G,	**2g**	g
h	B	**1h**	B,	**2h**	b

Tabel 8 esitatud vastavusi vaadates on näha, et kuigi esialgses kodeeringus kasutati ka nooti h, vastab ABC noodikirjas sellele b ning selline teisendus on eesmärgipärane. ABC standardi puhul on võimalik oktavites liikuda allapoole, lisades noodi lõppu koma ja ülespoole, lisades ülakoma. Oktavi võrra kõrgemate nootide märkimiseks oleks olnud võimalik väikese tähe asemel märkida ka ülakoma uue tähise juurde, kuid lihtsamaks ja kiiremaks sisestamiseks võeti kasutusele võimalus oktavi võrra kõrgema noodi märkimiseks väikese tähega. Antud skaalaga on kaetud kolme oktavi kõrgune helivahemik, mis oli ka vahemik, mis oli esindatud lähteallikaks olevas andmestikus (Joonis 39).



Joonis 39 Oktavi tähistamine ABC standardi järgi.

Antud lähenemine tagab, et juhul kui tulevikus on soov kodeerida laiema heliulatusega teoseid, siis on selleks võimekus olemas. Eelnevalt kasutuses olnud lahendus eeldas, et maksimaalne heliulatus on kolm oktavit, kuid enam seda piirangut ei ole.

Lisaks eelpooltoodule on võimalik noodi helikõrgust muuta kasutades järgnevaid lisatähiseid.

Tabel 9 Helikõrguse muutmiseks kasutatavad tähised.

Tähendus	**Originaal**	**ABCkuju**
Diees	#	^
Bekaar	-bekaar	=
Bemoll	b	_

Tabel 9 on näha nii originaalkuju, mis võeti aluseks teisenduse vajaduse tuvastamiseks, kui ABC kodeeringu kuju. Kuigi nii bemoll kui noot b olid tähistatud b tähega, ei tekitanud see teisendamisel probleemi, kuna bemolli ei saa ilma kaasneva noodita esineda ning seetõttu otsiti teisendamisel bemolli kombinatsioonina kaasnevast noodist ehk kujul ba, bb, bc jne.

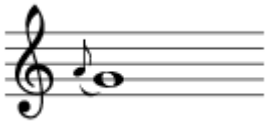
Kolmanda lisana olid andmebaasis kasutusel variatsioonid. Nende tähistamiseks olid osade nootide juurde märgitud kandiliste sulgude sisse täiendavad noodid (Joonis 40).

Joonis 40 Helikõrguse variatsioonid.

Variatsioon tähendab, et esitaja on teose esitamisel viisikordustel teatud kohtades nootide helikõrgusi muutnud. Viisi noodistaja on esituse info jäädvustamiseks mõnikord üles

märkinud varieeruvad noodid. See tähendab, et kui esitamise ajal on juhuslikes kohtades nootides variatsioonid ehk ühel korral on potsioonidel 5, 6 noodid BE, siis variatsiooni korral võisid olla samal positsioonil noodid CD.

Üks uue andmebaasi soovitud tulemitest on luua eeldused üle kogu andmestiku otsimiseks. Selleks, et seda võimaldada, peavad, variatsioonid olema terviklikud ehk sisaldama kõiki noote, mitte ainult varieeruvaid osi. Ehk juhul, kui me säilitaks variatsioonid osalise noodistusena või märkides ühele positsioonile kaks nooti, siis oleks sellise otsingu tegemine keerulisem ning tulemus oleks lugejale keerulisem aru saada. Seda illustreeris ka asjaolu, et EKM-iga iganädalastel koosolekutel ei olnud võimalik kokku leppida viisi kuidas variatsiooni põhiviisiga samal ajal kuvada, ilma, et kasutaks märgistust mis tegelikkuses tähendab midagi muud. Kokkuleppel EKM-iga otsustati, et variatsioonid kirjeldatakse täies pikkuses ning salvestatakse noodistuse juurde eraldi kirjena.



Joonis 41 Eellöögi tähistamine.

Neljandaks olid andmebaasis kasutusel eellöögid (Joonis 41). Eellöök on heli, mis esitatakse enne meloodiaheli. Eellöögid olid andmebaasis tähistatud sulgudega (), kuid ABC kujul on nende tähistamiseks kasutusel looksulud. Seega tehti see teisendus juhul kui sellised märgid leiti.

Joonis 42 Helivältused.

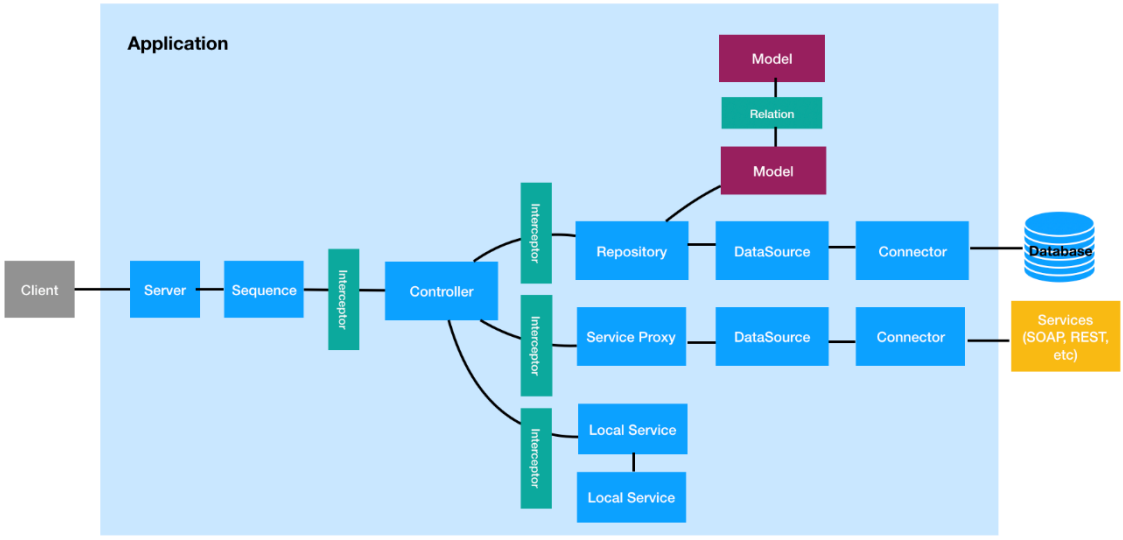
Viiendaks olid andmebaasis osa noote kiiremad noodid, mis tähendas, et ühel positsioonil asus kaks nooti, mis olid eraldatud kaldkriipsuga. Kui üldiselt olid kõik noodid helivältusega 1/8 nooti (Joonis 42), siis kiiremad noodid on sellest ühe astme võrra kiiremad ehk 1/16 nooti. Sellisel juhul muudeti mõlemal pool kaldkriipsu asuvad noodid 1/16 noodiks lisades nende järele kaldkriipsu “/”.

5.7 Viiside haldamise rakenduse arhitektuur

Selles jaotises antakse ülevaade töö tulemusena loodud rakenduse tehnilisest arhitektuurist.

5.7.1 Lüüsikiht

Lüüsikiht on mõeldud vahekihiks esitluskihi ja andmebaasi vahele andmete turvaliseks käitlemiseks. Antud töö raames on lüüsikiht loodud Node.js mootoril ning Loopback 4 platvormil.



Joonis 43 Loopback 4 komponendid [61].

Loopback 4 koosneb järgmistest komponentidest (Joonis 43): server, jada, püüdja, kontroller, repositoorium, mudel, andmeallikas, ja pistmik.

- Server realiseerib sissetulevad protokollid. Antud juhul realiseeriti REST protokoll läbi HTTPS kihi. Server kuulab defineeritud otspunkte, edastab päringu jadale ning tagastab jadast saadud vastuse.

- *Sequence* ehk jada on vahevara tegevuste järjestus, mis kontrollib, kuidas määratud server vastab sissetulevatele päringutele. Jada ainuke eesmärk on moodustada päringule vastus, käitades kindlas järjekorras temas defineeritud vahevarad.

- *Interceptor* ehk püüdja on taaskasutatav funktsioon, mida saab siduda meetodite välja kutsumisega. Näiteks saab logidesse märkmeid teha enne ja pärast meetodi

väljakutsumist või muundada ja valideerida meetodi parameetrite väärtuseid (argumente) ning väljundeid.

- Kontroller realiseerib otspunktide äriloogikat, toimides sillana otspunktide ning mudelite vahel. Lisades kontrollerile ning ta funktsioonidele dekoratsioone, saab siduda omavahel otspunktid kontrolleri funktsionaalsusega. Kontrolleri funktsioonid kasutavad äriloogika rakendamiseks ainult päringust saadud objekte ning selle kontrolleriga seotud teenuseid.

- Mudel kirjeldab ärivaldkonna objekte, defineerides selleks andmetüüpe, nimetusi ning piiranguid. Seda kasutatakse põhiliselt andmevahetuseks läbi kogu lüüsikihi, näiteks andmete pärimisel andmeallikast või andmebaasi lisatavate andmete valideerimisel. Lisaks luuakse mudeli ning kontrolleri põhjal otspunktide OpenAPI spetsifikatsioon.

- Repositoorium pakub andmeallikas defineeritud andmebaasi andmetele juurdepääsu, tagastades ja sisestades andmeid mudelis defineeritud tüüpide põhiselt. See tähendab, et andmebaasist küsitakse korrektsete andmetüüpidega väärtused ning andmebaasi sisestamisel kontrollitakse, kas väärtused on samuti õiget tüüpi. Põhimõtteliselt seotakse omavahel mudel ja andmeallikas ning pakutakse meetodeid mudeli objektide haldamiseks andmebaasis kasutades CRUD operatsioone.

- Andmeallikas defineerib andmebaasiga ühendumiseks kasutatava pistmiku ning andmebaasiserveri mandaadi. Seda kasutatakse andmebaaside sidumiseks repositooriumitega läbi ORM teenuse ning spetsiifilise andmebaasi pistmiku. Antud süsteemi puhul rakendatakse ainult ühte pistmikku: *loopback-connector-postgresql*.

5.7.2 Lüüsikihi otspunktid

Lüüsikihi otspunktid on defineeritud kontrollerites spetsiifiliste meetodi dekoratsioonidega. Kõik andmeid muutvad meetodid on kaitstud autentimise ning autoriseerimisega

Kõik lüüsikihi kontrollerid realiseerivad sama tüüpi otspunkte: sisesta üks, loenda, võta kõik, uuenda kõik, võta üks, uuenda üks, asenda üks, kustuta üks.

Antud töös loodi järgmised otspunktid:

- Viiside otspunkt.
- Isikute otspunkt.
- Klassifikaatorite otspunktid (29 tükki).
- Autentimise, registreerimise ning kasutajate otspunktid.

5.7.3 Esitluskiht

Esitluskiht on realiseeritud React platvormil. See koosneb järgnevatest osadest: komponendid, elemendid, mudelid, teenused ja tõlked.

- Komponent on eraldiseisev, isoleeritud ning taaskasutatav rakenduse osa, mis kuvab DOM-i ning rakenduse komponente etteantud sisendi põhjal. Komponendi elutsükkel koosneb loomisest, uuendamisest, eemaldamisest ning veahaldusest. Komponendi uuendamise puhul vahetatakse välja ainult muutunud elemendid, mitte kogu komponent.
- Element on komponendi ehituseks kasutatav muutumatu tükk. Elemente ei muudeta, vaid kustutatakse ning taasluuakse.
- Mudel on objekt, mis kirjeldab lüüsikihist päringu tulemusel saadud objekti parameetreid ning seda kuidas nende väärtuseid kasutajale kuvada. Mudeli andmete põhjal luuakse päringuks filter, mille edastamine otspunkti kirjeldab kontrollerile alammudelid, mida lisaks põhimudelile tagastada.
- Teenus on üle esitluskihi kasutatav objekt, mis sisaldab endas korduvkasutatavat funktsionaalsust, näiteks autentimine ja päringute tegemine.
- Tõlkimine sooritatakse esitluskihis baassõne alusel komponendi tasemel, kasutades selleks tõlketeenust. Vastavalt teenusele edasi antud kontekstist muudetakse algsõne kuvatavasse formaati.

5.7.4 Esitluskihi kasutajaliidese mudelipõhine lähenemine

Selleks, et ühtlustada erinevate vaadete ja päringute loomist ning lihtsustada esitluskihi vaadete muutmist andmestruktuuri muutumisel, otsustati vaadete komponendid luua abstraktsed, mistõttu on neid võimalik kasutada koos erinevate mudelitega. Selline lähenemine võimaldab näiteks tabeli komponenti kasutada nii viiside kui ka atribuutide kuvamise vaadetes ilma, et oleks vajadus uuesti defineerida objekti väljad ning sellega muud kaasas käivad omadused.

```
export const TuneModel = ModelService.GenerateDefaults({
    apiPath: 'tunes',
    list: {
        fields: [
            { field: 'tuneReference', headerName: 'tune.tuneReference', width: 170 },
            { field: 'textReference', headerName: 'tune.textReference', width: 170 },
            { field: 'soundReference', headerName: 'tune.soundReference', width: 170 },
            { field: 'videoReference', headerName: 'tune.videoReference', width: 170 },
            { field: 'catalogue', headerName: 'tune.catalogue', width: 170 },
            { field: 'nations', headerName: 'tune.nation', selector: 'title', width: 170 },
            { field: 'languages', headerName: 'tune.language', selector: 'title', width: 100 },
            { field: 'countries', headerName: 'tune.country', selector: 'title', width: 100 }
        ]
    },
```

Joonis 44 Viisi tabelis kuvamiseks kasutatava mudeli väljavõte.

Vaate komponent ootab sisendiks andmemudelit (Joonis 44) ning mudeli defineeritud struktuuri järgivat objekti. Oodatav struktuur luuakse mudeli põhjal ning lisatakse lüüsikihi päringule. Vaste saamisel loob komponent vastavalt mudelile elemendid, mille sisse võetakse väärtused mudeli põhjal etteantud objektist.

```
<Grid item container direction='row'>
    {
        TuneMelodyModel.view.fields.filter(x => !x.hidden && x.type !== 'player').map((field, k) => {
            return (
                <AssetPropertyElement key={k} title={t(field.headerName)} value={melody[field.field]} />
            );
        })
    }
    <Grid item><Button onClick={() => alert('Varsti tuleb')} variant='outlined'>{t('melody.export')}</Button></Grid>
</Grid>
<Grid item>
    <PlayerViewComponent elementData={melody} index={i.toString() + j.toString()} />
</Grid>
```

Joonis 45 Dünaamiliselt ning staatiliselt kuvatavad vaate komponendid.

Ainsad andmeid kuvavad komponendid, mis sisaldavad endas lisaks mudelipõhiselt loodud komponentidele ka püsivaid komponente, on viisi haldamiseks mõeldud komponendid. Püsivaks komponendiks loeme siinkohal komponenti, mis ei ole loodud viisi mudeli põhjal, vaid on vaatesse sisestatud komponendi lähtekoodi tasemel. Joonis

45 on väljavõte viisi vaatamise vaate lähtekoodist, kus on näha vaate dünaamilist aspekti esimeses Grid komponendis ning staatilist teises Grid komponendis.

5.7.5 Autentimine ja autoriseerimine

Kuna tegu on aktiivsel sisul põhineva esitluskihiga, siis on eriti tähtis, et kõik lüüsikihi otspunktid oleksid kaetud alati korrektsete autentimis- ning autoriseerimisreeglitega. Aktiivse sisuga esitluskihid on täielikult pöördprojekteeritavad ning nende käitumist annab kliendi veebilehitsejas käigu pealt muuta. Näiteks saaks autentimist ning autoriseerimist teostava funktsiooni *CanAccess* muuta alati tagastama tõeväärtust TRUE ning seeläbi anda juurdepääsu esitluskihi osadele, kuhu kasutajal muidu ligipääsu ei oleks. Kuna aga antud rakenduse puhul ei ole kriitiline peita rollipõhiseid vaateid ründajate eest, siis sellest ei tulene märgatavat täiendavat turvariski. Kõik andmete käsitlemise ning muutmisega seonduv toimub lüüsikihis, millele pääseb ligi ainult vastavat autorisatsiooni omades.

Autentimine

Lüüsikiht kasutab autentimiseks *User* ning *UserCredentials* klasse ning neile vastavaid andmebaasi tabeleid. *User* klassi objektid sisaldavad endas informatsiooni kasutaja kohta ning *UserCredentials* klassi objektid sisaldavad viidet kasutajale ning kasutaja parooli räsi.

Autentimise teenus, *UserManagementService*, pakub *verifyCredentials* meetodit, millega on võimalik kontrollida kasutaja mandaati sisse logimisel, andes argumentidena ette meiliaadressi ning kasutaja parooli. Vastava meiliaadressiga kasutaja kohta tehakse kõigepealt üldiste andmete päring ning selle õnnestumisel küsitakse lisaks ka kasutaja parooli räsi. Järgnevalt võrreldakse parooli ja räsi ning vastavuse korral tagastatakse leitud kasutaja andmed. Kui mingil põhjusel ei leita andmebaasist kasutajat, tema parooli või ei klapi sisestatud parool räsiga, siis tagastatakse koheselt päringu vastus üldsõnalise sõnumiga ebakorrektse mandaadi kohta.

Loa genereerimise teenus *JWTService* pakub *generateToken* ning *verifyToken* meetodeid. *generateToken* meetod loob loa, mis sisaldab kasutaja andmeid ning rolle, loodud luba kasutatakse autentimist ning autoriseerimist vajavate päringute tegemisel. v*erifyToken*

meetodit kasutatakse lüüsikihi turvatud otspunktidesse sissetulevate päringute valideerimisel.

UserController on *login*, *signup* ning muid kasutaja CRUD otspunkte defineeriv kontroller. Kontrolleri otspunktid on kaetud klassipõhise dekoratsiooniga (Joonis 46) ning on lubatud vaid administraatori õigustega kasutajatele. Ainsaks erandiks on *login* otspunkt, mis ei oma ligipääsu piirangut. *login* otspunkt kasutab *UserManagementService* meetodit *verifyCredentials* ning *JWTService* teenuse *generateToken* funktsionaalsust.

```
@authenticate('jwt')
@authorize({
  allowedRoles: ['admin'],
  voters: [basicAuthorization],
})
export class UserController {
  constructor(
```

Joonis 46 UserController klassi autentimise ning autoriseerimise dekoratsioonid.

Esitluskihis on autentimise lihtsustamiseks defineeritud *AuthService*, mis võimaldab kasutada meetodeid *Login* ja *Logout*. *Login* meetod saadab lüüsikihi *login* otspunkti mandaadi, saades vastuseks korrektse loa, salvestab selle veebilehitseja mällu. Kasutades *Logout* meetodit kustutatakse veebilehitsejast luba ning kasutaja suunatakse tagasi pealehele.

Esitluskihi päringutele loa lisamine toimub kasutades vahevara. Igale lüüsikihile esitatud päringule pannakse võimalusel kaasa veebilehitsejast leitud luba.

Autoriseerimine

Autoriseerimiseks kasutab lüüsikiht *basicAuthorization* teenust, mis lisatakse otspunktide kontrolleritele dekoratsiooniga (Joonis 46). Teenus eeldab, et kasutaja on autenditud, ning ootab sisendiks kasutaja informatsiooni. Vastavalt kasutaja rollidele tagastab teenus otsuse, mille põhjal saab kontroller käituda.

Esitluskihis kasutatakse autoriseerimiseks teenust *AuthService*, mis avaldab lisaks autentimiseks mõeldud *Login* ja *Logout* meetoditele ka *CanAccess* meetodi. *CanAccess* meetodit kasutatakse komponentide poolt loa olemasolu ning loas sisalduvate rollide, kontrollimiseks. Meetodi vastuse põhjal saab komponent otsustada, kas kasutaja peaks nägema vastavaid komponente ning kuvada vaated vastavalt sellele otsusele (Joonis 47).

```
<Divider />
{AuthService.CanAccess(['editor', 'admin']) &&
<Grid item container direction='row'>
  <Grid item xs={2}>
    <Typography variant='h5'>{t('tune.verification')}</Typography>
  </Grid>
  <AssetPropertyElement title={t('tune.verified')} value={assetData.verified}
 />
  <AssetPropertyElement title={t('tune.verifiedBy')} value={assetData.verifie
dBy} />
  <AssetPropertyElement title={t('date.created')} value={assetData.created} /
>
  <AssetPropertyElement title={t('date.modified')} value={assetData.modified}
 />
</Grid>
}
```

Joonis 47 Esitluskihi AuthService teenuse kasutusnäide.

5.8 Kasutajaliidese rollid

Kasutajaliidese rollide loomisel lähtuti sellest kuidas infosüsteemi esimese versiooni autorid olid seda planeerinud. Seega on rakenduses kolm erinevat rolli ning järgnevalt on lühidalt kirjeldatud iga rolli õigused.

5.8.1 Tuvastamata kasutaja

Tuvastamata kasutaja on kasutaja, kellel on piiratud ligipääs infosüsteemile. Tuvastamata kasutajal on õigus vaadata kõiki andmeid välja arvatud muudatuste logid ja kasutajakontodega seotud informatsioon. Samuti on tuvastamata kasutajal õigus ainult andmeid vaadata.

5.8.2 Toimetaja

Toimetaja roll omab kõiki õiguseid, mis on tuvastamata kasutajal ning lisaks nendele ka õigust teha süsteemis muudatusi. Toimetajal on õigus muuta kõiki viiside ja klassifikaatoritega seonduvaid andmeid. Toimetajal on lisaks õigus vaadata viiside ja klassifikaatorite puhul muudatuste ajalugu. Toimetajal aga ei ole õigust vaadata kasutajakontodega seotud infot.

5.8.3 Administraator

Administraatoril on õigused teha kõike mida Tuvastamata kasutaja ja Toimetaja ning lisaks nendele on tal õigus vaadata ja muuta ka kasutajakontodega seotud infot.

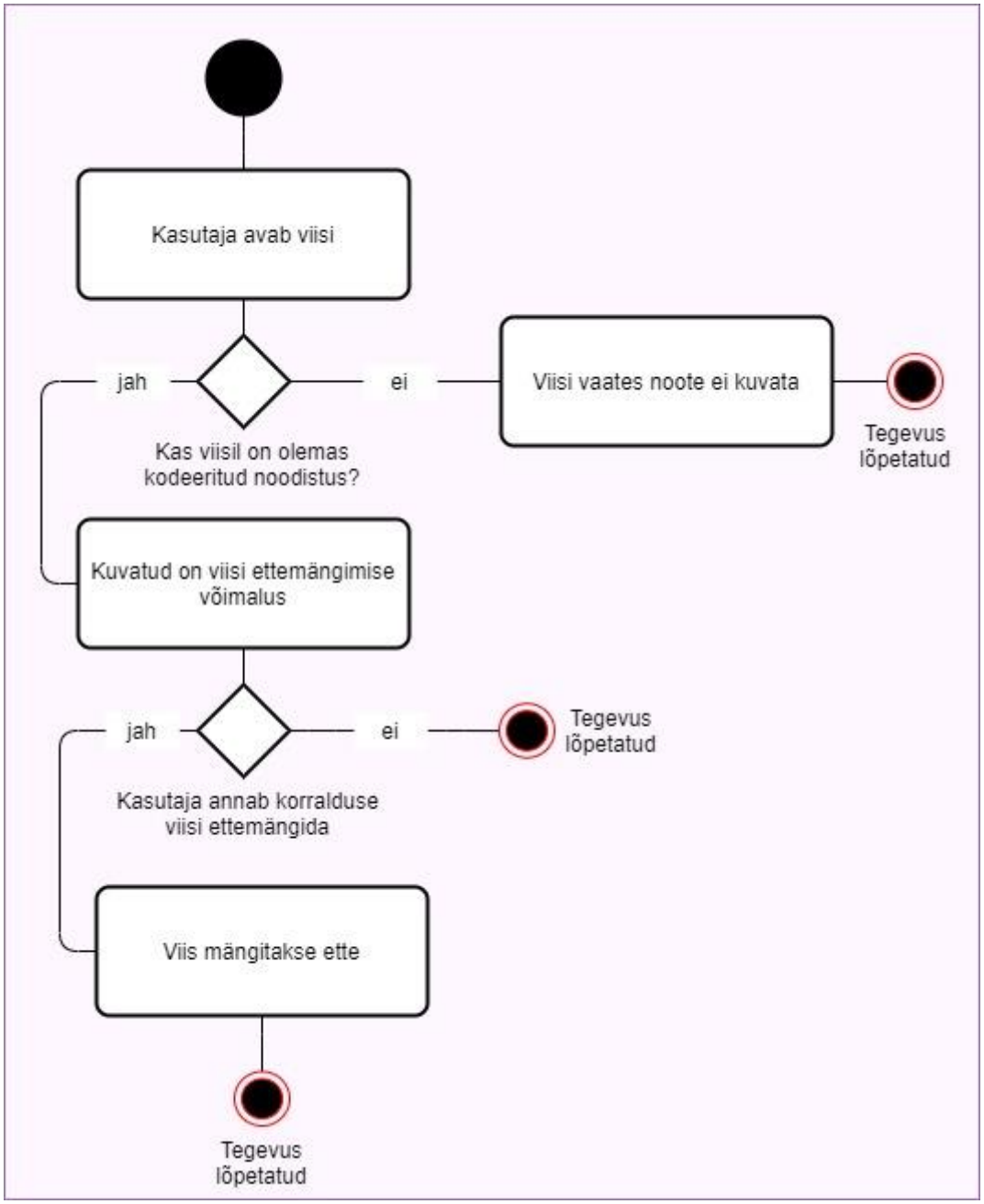
6 Realisatsioon

Järgnevalt kirjeldame antud lõputöö raames tehtud arendusi ning nende tulemusi. Peatükk keskendub põhiliselt veebiliideses tehtud muudatustele

6.1 Realiseeritud uus funktsionaalsus

Antud töö raames otsustati olemasolev platvorm CakePHP välja vahetada Node.JS mootoril toimiva Loopback 4 ja React.js platvormide vastu. Selle tulemusena taasloodi eelnevalt realiseeritud funktsionaalsus ning täiendati rakendust vastavalt lisandunud funktsionaalsetele nõuetele (vt jaotis 4.1).

6.1.1 Viisi kuulamine



Joonis 48 Viisi kuulamise tegevusdiagramm.

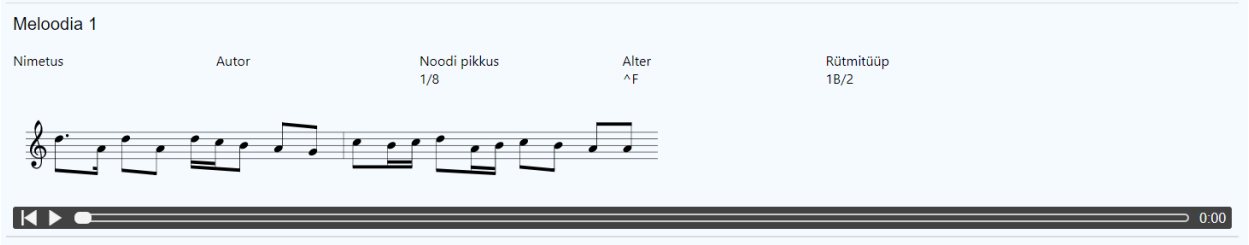
Viisi kuulamise töövoog on järgmine:

- Kasutaja avab viisi, mille heliteost ta kuulata tahab.
- Rahvaviisil, millel on olemas kodeeritud noodistus, on helifaili kuulamise võimalus. Lisatud on ka noodistuse pilt.

- Kui kodeeritud noodistik ei ole viisile lisatud, siis kuulamise võimalust ei ole. Samuti puudub noodistuse pilt (Joonis 48).

- Kui kodeeritud noodistus eksisteerib ja noodistus pildina kuvatud on, siis tekib noodistuse alla heliriba, millega helifaili ettemängida (Joonis 49).

- Kui ettemängimise alustamiseks nuppu vajutada, siis mängitakse viis ette.



Joonis 49 Viisi noodistik ja ettemängimise funktsionaalsus.

Loodud infosüsteemis kodeeriti kõik viisid kasutades ABC standardit. Tänu sellele on võimalik genereerida kodeeritud noodistusest noodistiku kuva ja helifailide tekitamiseks kasutada erinevaid teeke. Kuna EKM-i arhiivis on teoseid nii dokumentide kui helifailidena, siis keskendus antud lõputöö vaid kodeeritud noodistuste kuvamisele ja ettemängimisele. Originaalidest helifaile otse loodud lahenduses ei kuvata ega kanta ette. Nende kuulamiseks on vaja kasutada viiteid Kivikese infosüsteemi, kust on võimalik helifaile alla laadida või ette mängida. Lõputöö raames realiseeritud lahendus võimaldab vaid kodeeritud noodistuste põhjal heli tekitada.

Selleks, et ABC kodeeringust genereerida nii noodistust kui ka helifaili, kasutati ABCjs nimelist tarkvara. Selle autoriks on Paul Rosen ning tarkvara on kasutatav MIT litsentsiga [62], mille alusel on võimalik antud tarkvara kasutada, muuta ja jagada tasuta ning anda üle õigused antud tarkvara kasutamiseks ka EKM-ile. [63]

Peatükis 6.5 kirjeldatakse abiprogrammi, mille abil tekitati esialgne hulk kodeeritud noodikirju. Lisaks noodikirjale on noodistuse kuvamise puhul tähtsad ka järgnevad parameetrid.

- Noodivõti.

- Alter.

- Tempo.
- Noodipikkus.
- Tugiheli.
- Kõrgus.
- Takt.

Nende aluseks võeti MS Access andmebaasi tabeli *Viis* veerud *IDviis*, *võti*, *alter*, *tugiheli*, *kõrgus*, *tempo* ja *takt*. Selle tulemusena tekitati Google Sheets rakenduses tabel, mille külge liideti EKM-i poolt loodud kontrollitud ja korrigeeritud parandatud viiside metaandmestikust viisiviide (viide arhivaalile). Tulemusena tekkis tabel, milles oli võimalik alustada andmete kontrolli ja väljade osas, mis tuleb teisendada ABC noteeringu vormingusse, ka teisendust. Google Sheets rakenduses analüüsiti andmestikku ja otsiti vigadega väärtuseid ning loodi vastavustabelid, mille abil teisendati lähteandmete kirjed uuele kujule. Juhul kui sisestamisel oli tekkinud vigu või oli ebamäärasust, siis tehti tabelis käsitsi parandusi. Näiteks oli mõne tempo tähise juures kasutusel küsimärgid või oli sisestamisel tehtud vigu.

Vastavustabelid loodi järgnevalt: Esiteks kasutati kõikide erinevate variatsioonide leidmiseks veergudest *tempo*, *tugiheli* ja *alter* Google Sheets funktsiooni *Unique*, mille abil oli võimalik välja tuua kõik ühes veerus olevad unikaalsed väärtused ja *Countif* funktsiooni abil nende esinemiste arv. Järgnevalt võeti see info aluseks ning tekitati vastavustabel kus olid kõrvuti vanad väärtused, mis saadi *Unique* funktsiooni abil ja uued väärtused. Kolmandaks tehti esialgses tabelis parandusi juhul kui kirjed olid vigased.

Selleks, et andmeid süsteemi importida, võeti aluseks kontrollitud ja parandatud Google Sheets tabel ning kasutades *Vlookup* funktsiooni kombineeriti kõikidest väljadest uus tabel, mille puhul esialgsed väärtused asendati teisendatud väärtustega. Näiteks juhul kui esialgu oli *tempo* veerus väärtus “Kaunis ruttu”, siis teisenduse tulemusena asendati see tempoga “1/4=136”. Teisenduste vastavustabeli koostamisel oli kõige suurem roll EKM töötajatel, kuna neil oli olemas vajalik ligipääs nii arhivaalidele kui arusaam kuidas tuleks väärtuseid teisendada.

Lõpliku importfaili loomise aluseks võeti noodi kodeerimise utiliidi väljund ning sellele lisati väärtused *clef*, *alter*, *tugiheli*, *korgus*, *tempo*, *takt*, *note_length*. Lõpliku importfaili fragment on esitatud Joonis 50.

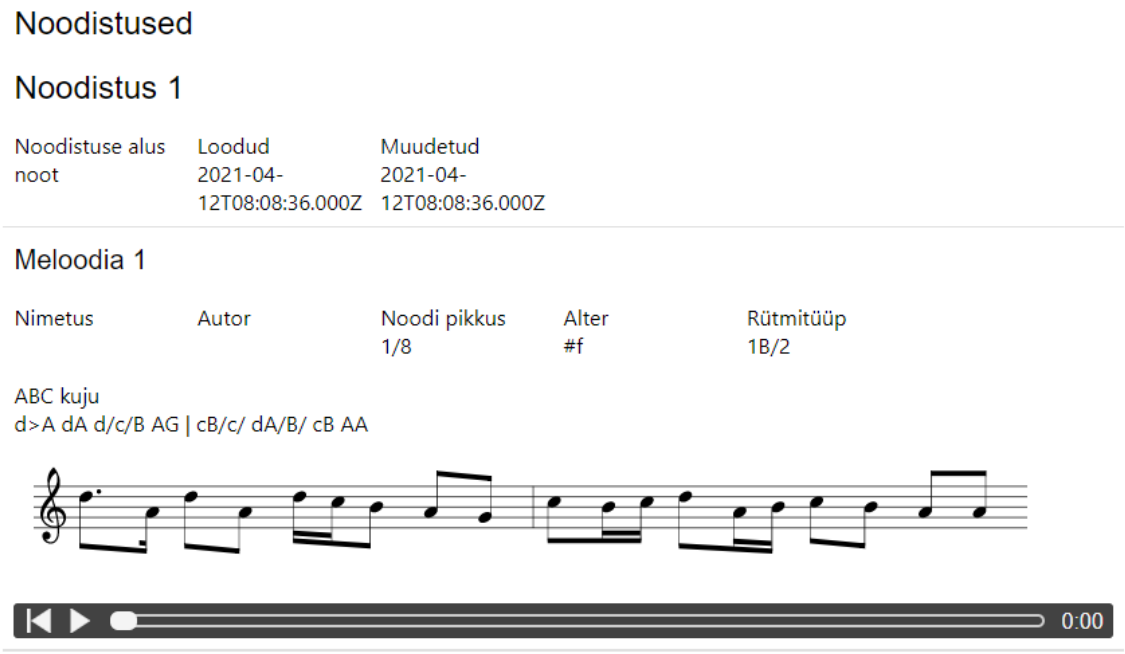
	A	B	C	D	E	F	G	H	I	J	K	L
1	id	reference	variation	rytm_type	melody	clef	alter	tugiheli	korgus	tempo	takt	note_length
2	1	245	1	1	dB BG BA AF | cA AF AG GG	&	^F	g	b			1/8
3	2	245	2	1	dB BG BA A!mark!G | cA AF AG GG	&	^F	g	b			1/8
4	3	246	1	1	dd cB dB BA | cB AG BG GG	&		g				1/8
5	4	247	1	1	dd dc BA AB | cB cA AG GG	&	^F	g	g1=Fis			1/8
6	5	4492	1	1	GD cc BA GE | ED DA AF GG	&	^F	g	h	1/4=84	24	1/8
7	6	4492	2	1	GD cc BA GE | ED !mark!B!mark!F A	&	^F	g	h	1/4=84	24	1/8
8	7	4563	1	1	Bd Bc BG dB | AG cB AG GG	&	^F	g				1/8
9	8	4563	2	1	Bd Bc BG dB | AG cB !mark!BG GG	&	^F	g				1/8
10	9	248	1	1	dB GG BA FA | cA FF GG GB	&	^F	g				1/8

Joonis 50 Lõpliku importfaili näide.

Selle tulemusena kombineeriti kogu MS Access andmebaasis olnud info ühte tabelisse, mida oli võimalik loodud lahenduse andmebaasi importida.

Selle info põhjal on võimalik iga viisi kohta genereerida piisava detailsusega noodistus. Noodistuse põhjal on võimalik luua helifaili, mida saab otse veebilehitseja ette mängida (Joonis 51).

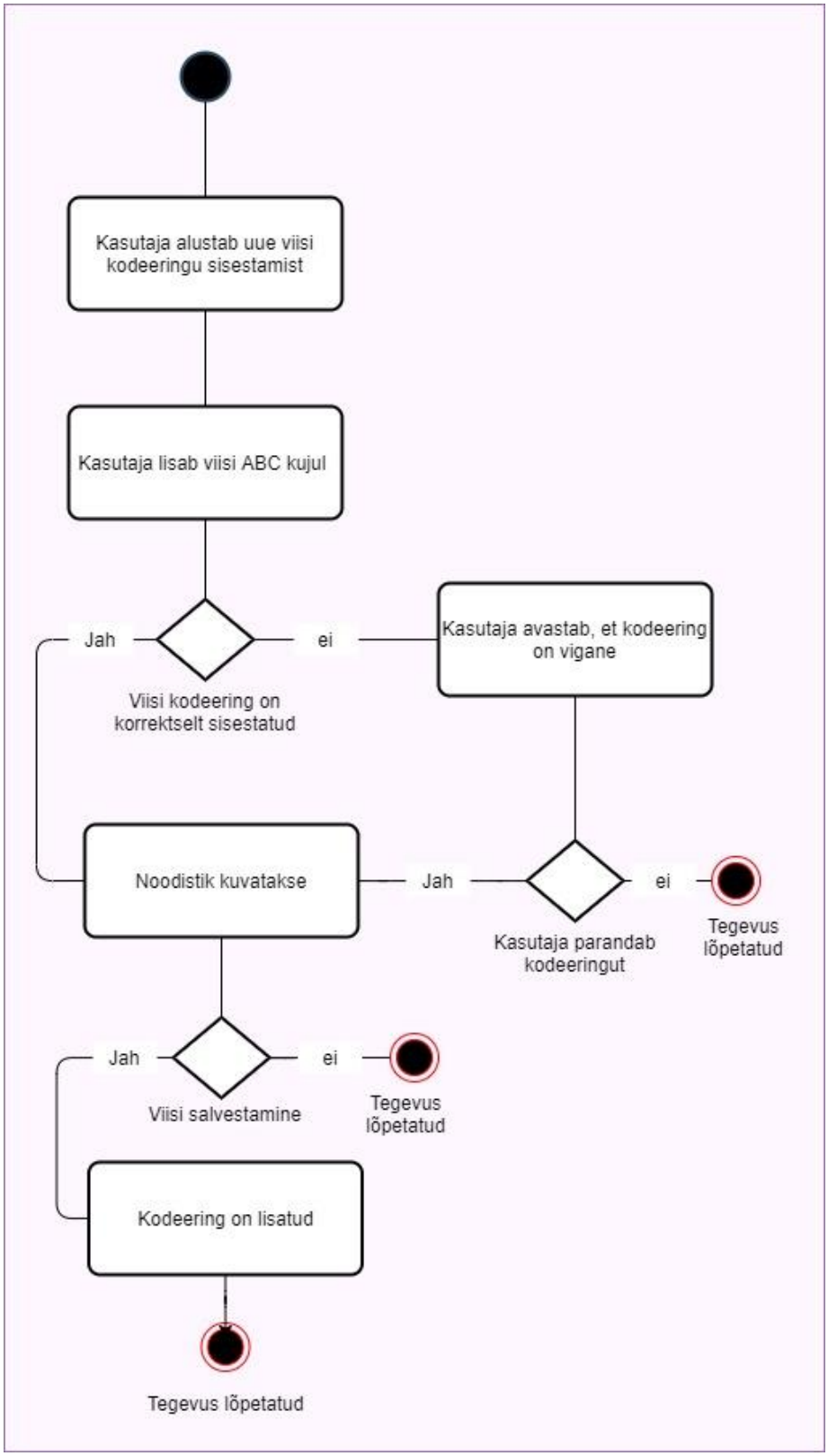
Realisatsioonis kuvatakse viisi vaates eraldi sektsiooni nimega „Noodistused“. Seal on võimalik kirjeldada erinevaid noodistamisi, mis sellele viisile on tehtud ning näha on ka täiendav metaandmestik.



Joonis 51 Noodistuse kodeeringu põhjal tekitatud viisi kuulamine.

Võimalus ühe arhivaali kohta sisestada mitut noodistust oli EKM-i poolne nõue. See tuleneb sellest, et ühte arhivaali võisid kodeerida erinevad isikud. See on eriti tähtis sellistel puhkudel kui noodistuse loomine käib helisalvestisest, mille korral võivad ühte ja sama heliteost erinevad kodeerijad erinevalt üles märkida. Seetõttu on oluline, et seda infot oleks võimalik ekraanil kuvada. Noodistuse all on eraldi heliriba, millele vajutades on võimalik lasta süntesaatoril viisi ette kanda.

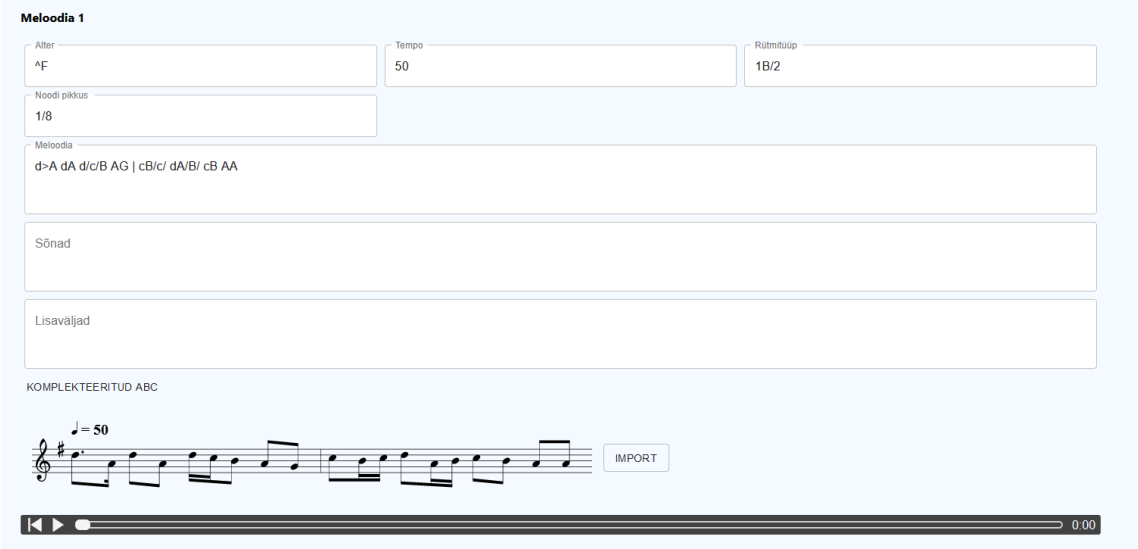
6.1.2 Viisi kodeeringu sisestamine



Joonis 52 Viisi kodeeringu sisestamise tegevusdiagramm.

Viisi kodeeringu sisestamise töövoog on järgmine (Joonis 52):

- Sisse logitud kasutaja alustab uue kodeeringu sisestamist viisi vaatest vajutades nuppu „loo uus“ .
- Kodeeringut tuleb sisestada ABC kujul.
- Kui kodeering on korrektselt vastavalt standardile sisestatud, siis kuvatakse viis noodistikuna. Kui kodeering ei vasta nõuetele, siis noodistust ei kuvata ja sisse tuleks viia vastavaid parandusi.
- Kui noodistik on kuvatud ja kasutaja on veendunud selle korrektsuses, siis tuleb lisatud andmeid salvestada.
- Salvestamisega lõpeb viisi kodeeringu sisestamise protsess.



Joonis 53 Viisi kodeeringu sisestamise vaade.

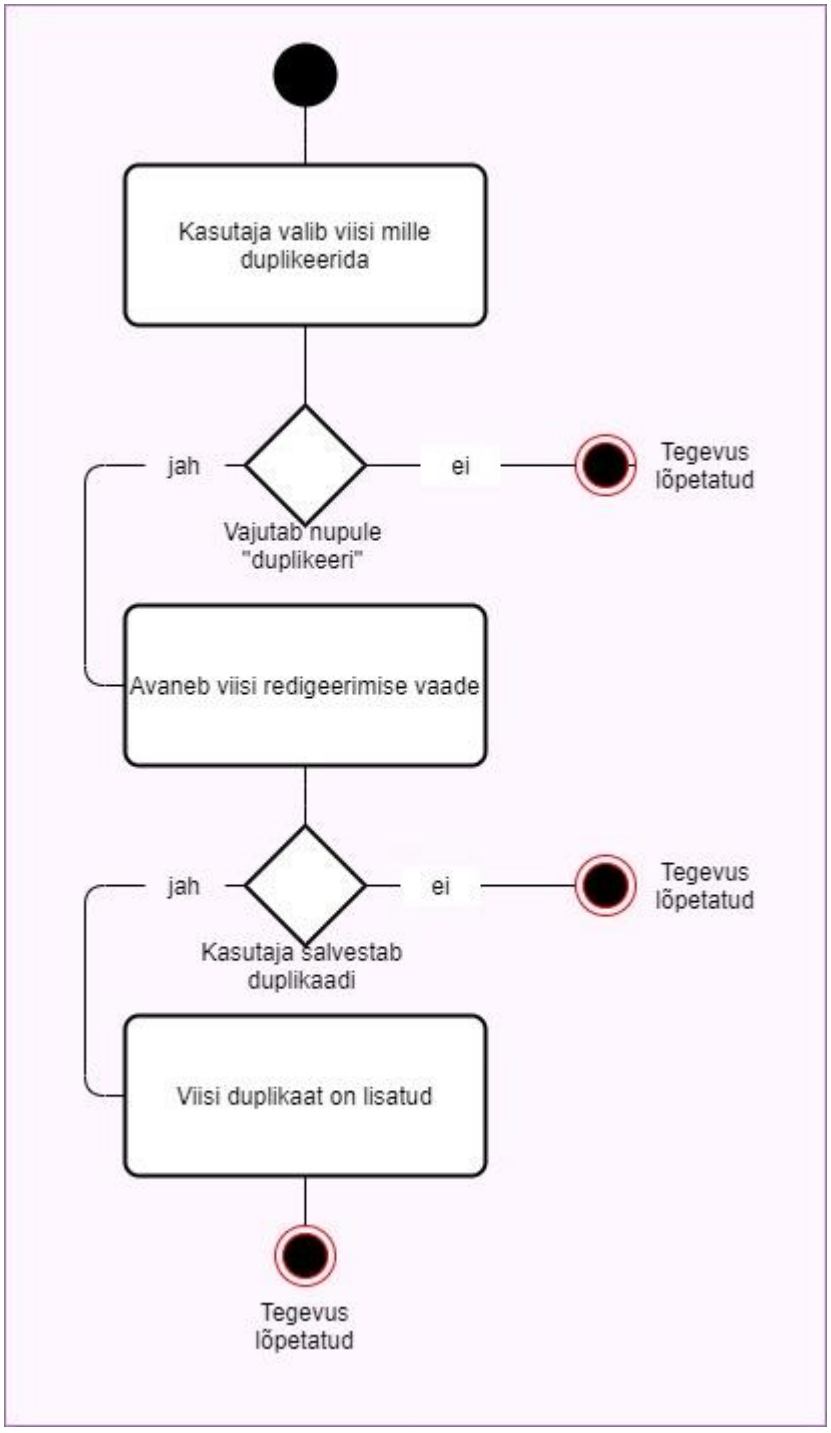
Viiside kodeerimiseks võeti kasutusele ABC standard, mis tähendab, et viise on võimalik lihtsalt loetava tekstina esitada. Viiside noodistuse kuvamiseks võeti kasutusele ABCjs nimeline tarkvara, mille puhul on võimalik reaalajas näha kodeeritava noodistuse visuaalset esitust (Joonis 53). Lisaks sellele on võimalik noodistusele lisada infot nagu näiteks viited ja sõnad, selleks, et veelgi täiendada arhivaali kohta digitaalselt hoiustatavat informatsiooni.

Viisi kodeeringu sisestamiseks on loodud vorm, kus on võimalik määrata väärtused eelpool mainitud parameetritele nagu helivõti, tempo ja noodipikkus. Lisaks on eraldi väljad, mille abil on võimalik sisestada nii kodeeritud noodistus kui ka sõnad.

Kodeerimise sisestamise ajal kuvatakse reaalajas loodavat noodikuva ning selle abil on võimalik aktiivselt jälgida kodeeringu vastavust soovitud tulemusele.

Kodeeringu sisestamisel puuduvad kontrollid kuna rahvaviisid ei allu tavapärastele muusikalistele põhimõtetele ning üks EKM-i soovidest oli vabaneda noodistuste sisestamisel kõikidest piirangutest. Kontrolliks võib pidada näiteks seda, et kas noodistus vastab seatud taktimõõdule. Seetõttu ei ole kodeeringu salvestamisel rakendatud kontrolle ning lähtutakse põhimõttest, et kodeeringu sisestaja peab olema piisavalt pädev hindamaks kas tulemus vastab tema soovile.

6.1.3 Viisi duplikeerimine



Joonis 54 Viisi duplikeerimise tegevusdiagramm.

Viisi duplikeerimise e kloonimise puhul kopeeritakse kogu viisi sisu uue viisi loomise vaatesse, milles on võimalik teha kõikides väljades soovitud muudatusi (Joonis 54). Siinkohal on tähtis mainida, et uus, duplikeeritud, viis salvestub alles peale kasutaja poolt salvestamise nupu vajutamist ning süsteemi poolt kohe koopiat viisist ja kõikidest sellega

seotud kirjetest automaatselt ei tehta. Põhimõtteliselt toimub uue viisi loomine vana viisi andmetega. Uut viisi ei seostata andmebaasi tasemel originaalse viisiga.

Joonis 55 Näide viisi duplikeerimise võimalusest.

Duplikaadi tegemiseks tuleb liikuda viisi vaatesse ning vajutada nuppu “kopeeri” (Joonis 55). Selle tulemusel avaneb viisi redigeerimise vaade, kust on näha kõik originaalviisi väärtused ning päises on hoiatusteade, et tegemist on duplikaadi vaatega. See on mõeldud vältimaks olukordi kus tekib segadus viisi muutmise vaate eesmärgi osas. Muudatuste salvestamiseks andmebaasi on kasutajal vajalik vajutada salvestamise nuppu ning selle tulemusena salvestatakse antud kirje andmebaasi ning sealt edasi käitub duplikeeritud viis haldamise mõttes samamoodi kui imporditud või käsitsi sisestatud viis.

6.1.4 Viisile sündmuste ajaloo lisamine

Viisi sündmuste ajaloo salvestamise põhjaks kasutati @sourceloop/audit-log [64] pistikprogrammi, mida modifitseeriti antud rakenduses kasutamiseks. Antud pistikprogrammi eesmärk on luua NodeJS rakendustes võimalus *create*, *update*, *replace* ja *delete* tegevuste logimiseks. Vahemärkusena võib välja tuua, et *update* uuendab esitluskihi poolt saadetud väljasid, *replace* operatsioon aga asendab terve kirje täielikult esitluskihi poolt saadetud objektiga. Pistikprogramm töötab selliselt, et eelnevate tegevuste puhul salvestatakse json kujul andmebaasi kogu http käsuga kaasas olnud massiiv. Kuna meie soovisime hilisema lugemise lihtsustamiseks ja andmemahu kokkuhoiuks salvestada ainult kasutaja poolt tehtud muudatusi (st mitte samaks jäänud väärtuseid), siis tegime järgneva täienduse:

```
let diffAfter: any = diff(before, after);
let diffBefore: any = {};
Object.entries(diffAfter).forEach(
  ([key, value]) => (diffBefore[key] = (before as any)[key]),
);
```

Joonis 56 Audit log muudatuste filtreerimise lähtekood.

Selle tulemusena filtreeriti välja (Joonis 56) kõik muutunud json struktuuri elemendid ning tekitati uus massiiv, mille sisuks on vaid muutunud andmete algsed ning uued väärtused.

```
const auditLog = new AuditLog({
    actedAt: new Date(),
    // eslint-disable-next-line @typescript-eslint/no-explicit-any
    actor: (user?.id as any).toString() ?? '0',
    action: Action.UPDATE_ONE,
    before: diffBefore,
    after: diffAfter,
    entityId: before.getId(),
    actedOn: this.entityClass.modelName,
    actionKey: opts.actionKey,
    ...extras,
    });
```

Joonis 57 Audit log lähtekoodi näide.

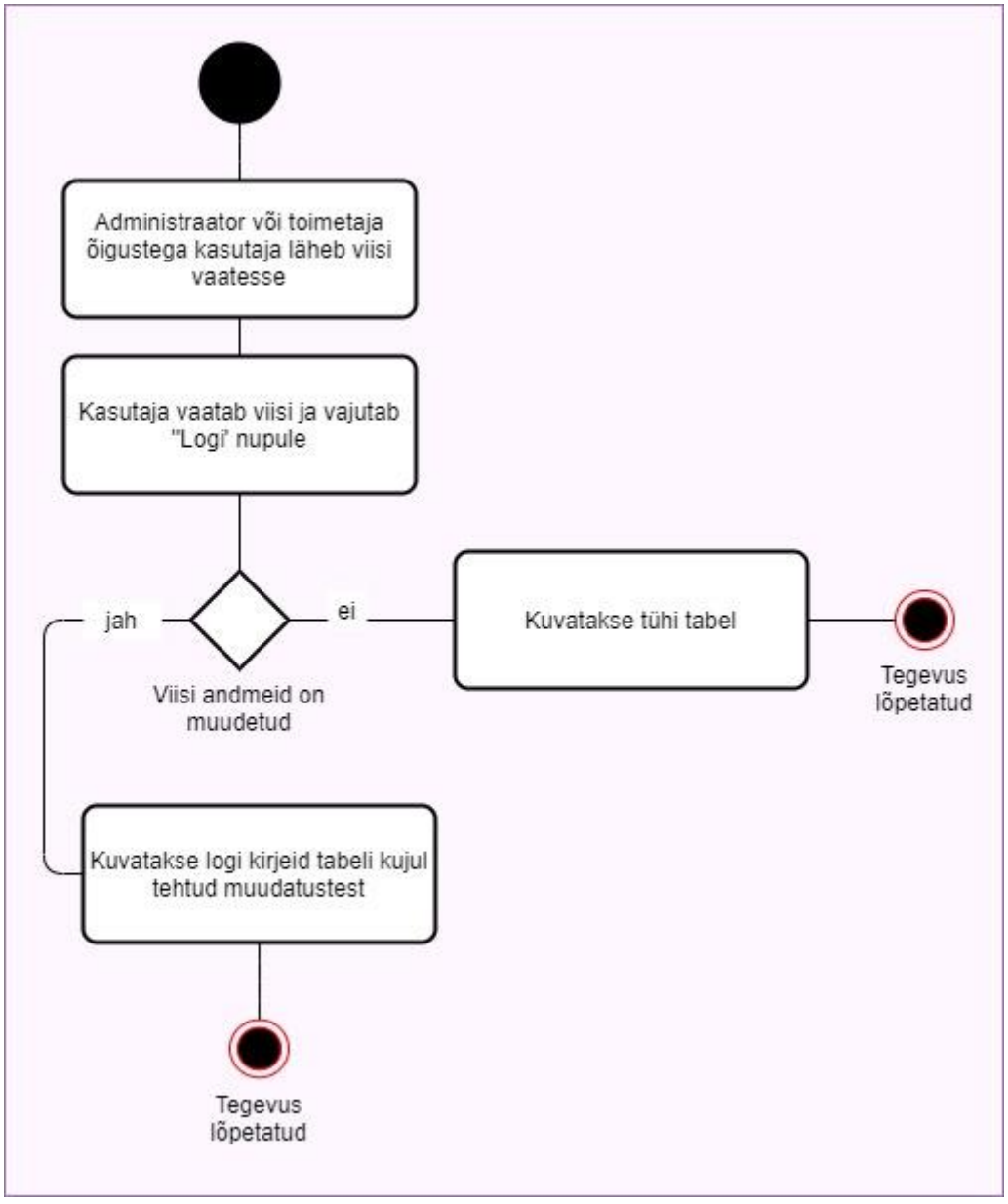
Joonis 57 on näha lüüsikihis loodud objekt *update* operatsiooni puhul. Andmebaasi lisatakse järgnev info.

- Rakenduse kasutaja, kes tegevuse tegi.
- Ajatempel, millal tegevus tehti.
- Kirjeldus, mida ja mis objektiga tehti.
- Välja muutmiseelne ja muutmisjärgne väärtus.

Selleks, et neid andmeid salvestada, loodi eraldi andmebaasi tabel (vt jaotis 6.4). Töö autorid on teadlikud, et see tabel võib kasvada üle aja suureks ning andmemahu vähendamiseks täiendati pistikprogrammi selliselt, et oleks võimalik salvestada vaid muudetud väärtused ilma vajaduseta tervet kirjet salvestada. Kui tulevikus tekib probleem liigselt kasvanud andmemahuga, siis on võimalik luua protsess, mille tulemusel salvestatakse kas viimased kümme kirjet iga viisi kohta või realiseerida vanade logikirjete regulaarne kustutamine. Antud lahenduse analüüs ning juurutamine on aga antud töö skoobist väljas.

Logi nägemiseks lisandus administraatori ning toimetaja õigustega kasutajate viisi vaatesse nupp, mida vajutades avaneb vaade kust on näha kogu antud viisiga seotud logi sisu. Logisse salvestatakse kõik viiside sisestamised, muutmised ning kustutamised. Andmete vaatamiste kohta informatsiooni ei logita.

6.1.5 Viisi sündmuste ajaloo vaatamine



Joonis 58 Sündmuste logi vaatamise tegevusdiagramm.

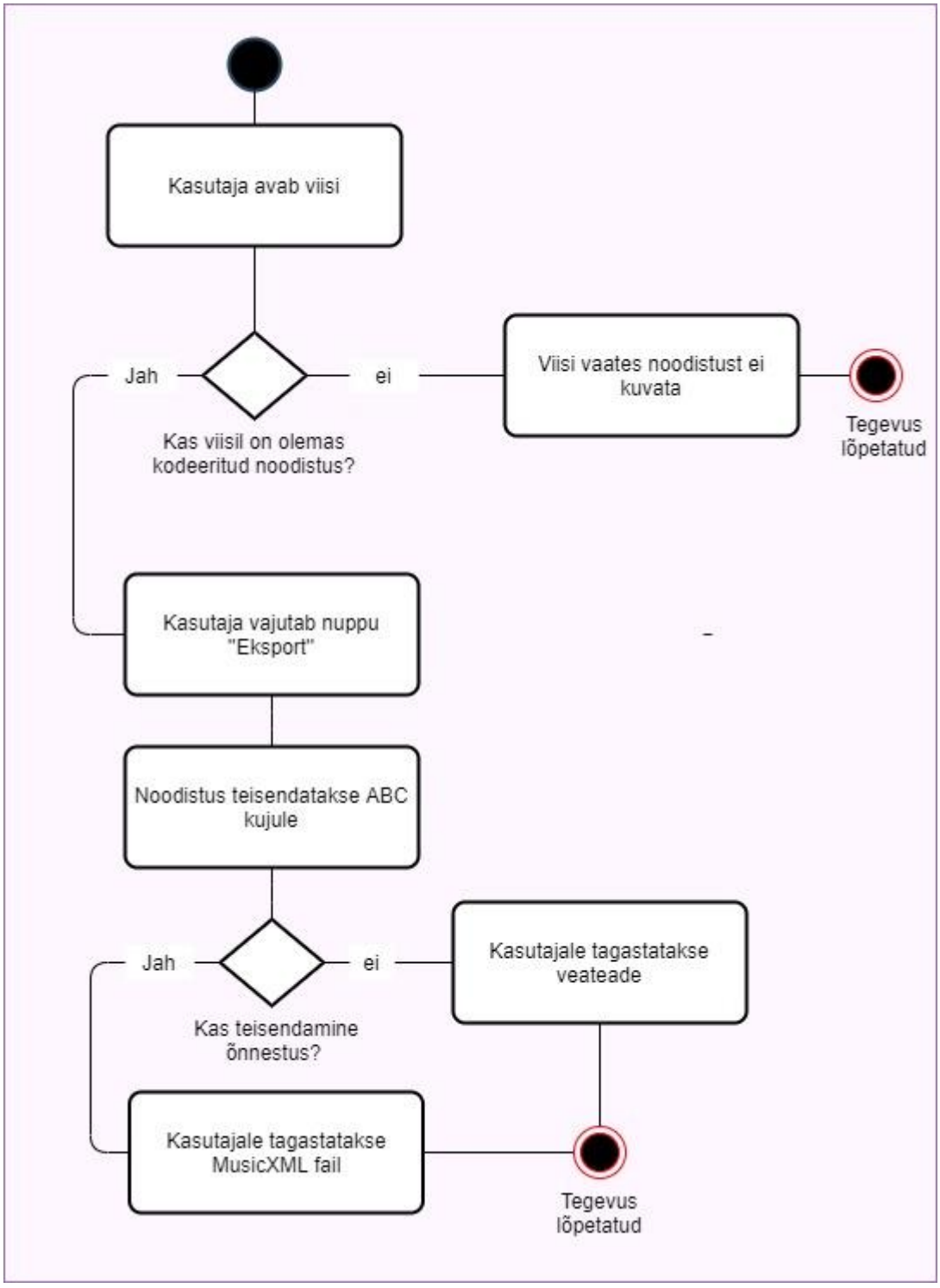
Sündmuste vaatamise protsessi kirjeldus (Joonis 58):

- Logi nägemiseks peab kasutaja olema administraatori või toimetaja roll.
- Kasutaja valib viisi, mille kohta ta logi vaadata soovib.

- Viisi vaates on nupp nimega „Logi“.
- Juhul, kui keegi on eelnevalt viisi andmeid muutnud, siis sisaldab logi informatsiooni muudetud andmete kohta (Joonis 59).

Joonis 59 Näide logist.

6.1.6 Viisi eksport/import

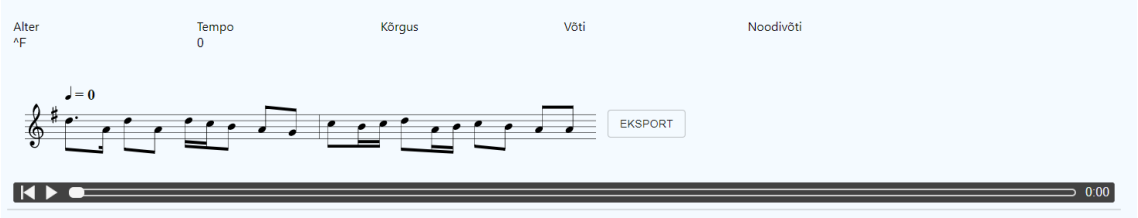


Joonis 60 Viisi eksportimise tegevusdiagramm.

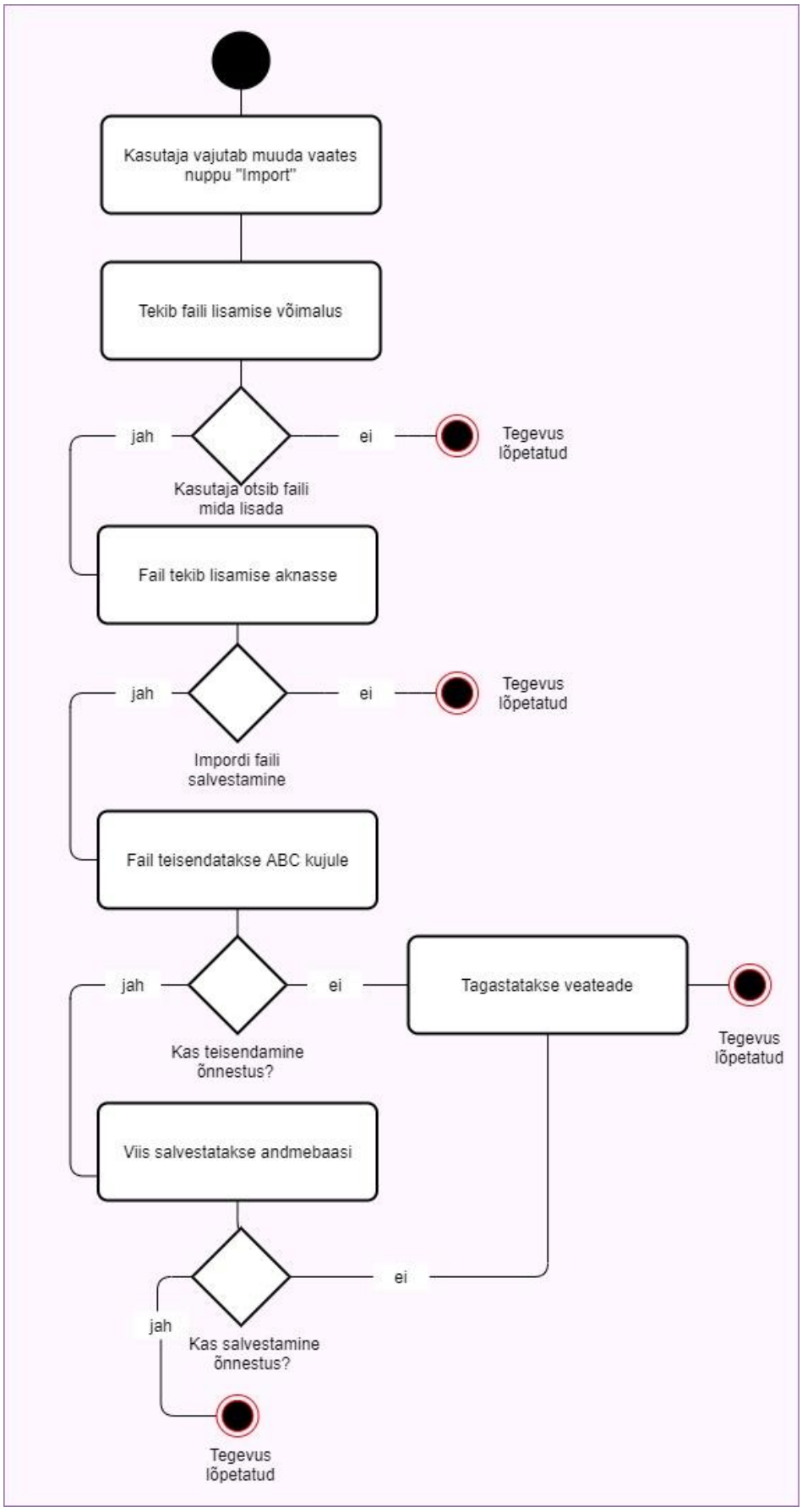
Viisi eksportimise töövoog (Joonis 60) on järgnev.

- Kasutaja avab viisi, mille helifaili ta soovib enda seadmele eksportida ehk alla laadida.

- Kui viisile on lisatud kodeeritud noodistik, siis on viisi vaates olemas pilt noodistikust ja helifaili kuulamise ja salvestamise võimalus. Kui viisil puudub kodeeritud noodistik, siis puudub ka helifail, mida saab alla laadida.
- Kui noodistus on olemas tekib selle juurde nupp nimega „eksport“ (Joonis 61).
- Kui kasutaja vajutab nupule salvestatakse MusicXML fail tema seadmesse.



Joonis 61 Näide viisi eksportimise võimalusest.



Joonis 62 Viisi importimise tegevusdiagramm.

Viisi importimise töövoog (Joonis 62) koosneb järgnevatest etappidest.

- Õigustega sisse logitud kasutaja valib viisi, millele ta soovib faili lisada.
- Kasutaja vajutab viisi muutmiseks nuppu „muuda“. Avatakse viisi muutmise vaade. Antud vaates tekib nupp nimega „import“. Importida saab MusicXML formaadis faile.
- Kasutaja vajutab nupule ja tekib faili lisamise võimalus.
- Kasutaja salvestab lisatud faili. Kui ta faili ei lisa, ega salvesta, siis on tegevus lõpetatud.
- Pärast salvestamist on faili import lõpetatud.

Viiside ekspordi ja impordi jaoks on kasutusel xml2abc ja abc2xml programmid, mille abil on võimalik teisendada MusicXML formaadist andmeid ABC formaati ja vastupidi. Eksport ning import funktsionaalsuse vajadus tulenes Finale tarkvarast kui selgus, et sellega ei ole võimalik salvestada viiside esitust ABC kodeeringus. Kuna märkimisväärset osa EKM arhivaalidest on võimalik eksportida MusicXML formaati, lisati loodud tarkvarasse selline funktsionaalsus. Antud lähenemise abil on võimalik juba kodeeritud andmestikud eksportida MusicXML kujule ning need siis teisenduse läbi uude süsteemi sisse laadida.

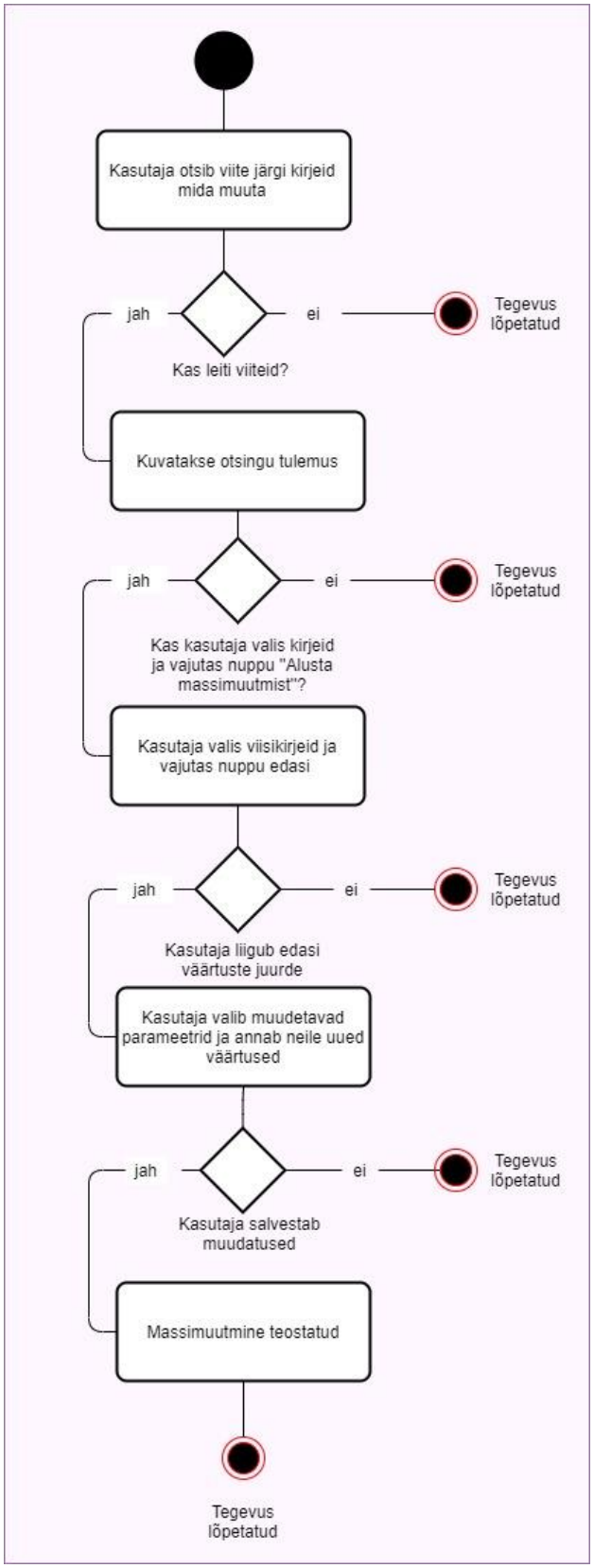
Xml2abc [65] ja abc2xml [66] on vabalt kasutatavad GNU GPL litsentsi abil. Antud litsentsi viide (Joonis 63) on osa rakenduse lähtekoodist.

```
Copyright (C) 2012-2018: W.G. Vree
Contributions: M. Tarenskeen, N. Liberg, Paul Villiger, Janus Meuris, Larry Myerscough,
Dick Jackson, Jan Wybren de Jong, Mark Zealey.

This program is free software; you can redistribute it and/or modify it under the terms of the
Lesser GNU General Public License as published by the Free Software Foundation;

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See the Lesser GNU General Public License for more details. <http://www.gnu.org/licenses/lgpl.html>.
```

Joonis 63 Xml2abc ja abc2xml programmide litsentsi väljavõte.

Sellest tulenevalt on võimalik eelpool väljatoodud rakendusi kasutada selleks, et MusicXML kodeeringuga andmeid teisendada ABC kujule ja vastupidi.

6.1.7 Mitme viisi korraga muutmine



Joonis 64 Mitme viisi korraga muutmise tegevusdiagramm.

Mitme viisi korraga muutmise töövoog (Joonis 64) on järgnev.

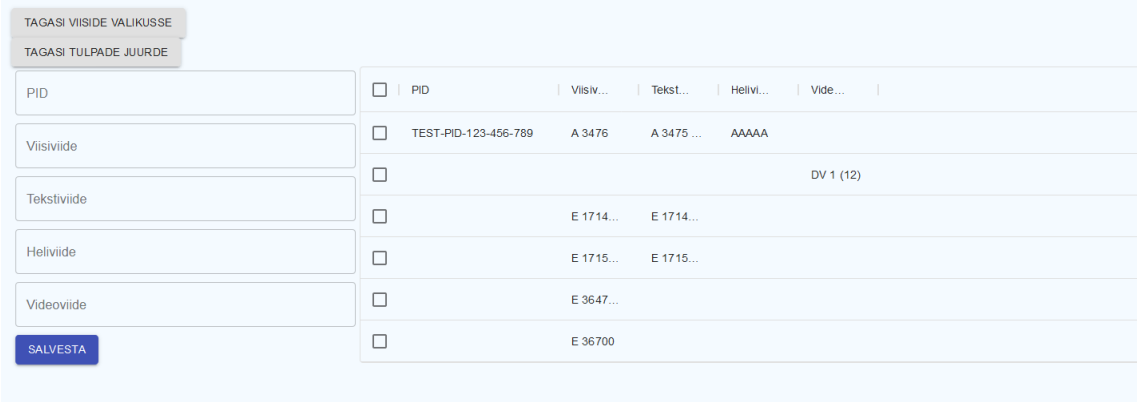
- Kasutaja otsib viite järgi kirjed, mida muuta.
- Kasutaja teeb kirjete valiku otsingu tulemuste seast ning vajutab nupule „Alusta massimuutmist“.
- Kasutaja valib muudetavad väärtused ning vajutab nupule „Edasi väärtuste juurde“.
- Kasutaja sisestab muudatused, valib muudetavad kirjed ning vajutab nuppu „Salvesta“.
- Massimuutmise kinnitus kuvatakse ning nupul „kinnita“ vajutamisel massimuutmine teostatakse.

Massmuutmise vajaduse analüüsimisel jõudsid autorid koos EKM-iga arusaamisele, et mitme viisi korraga muutmiseks tuleb kasutajal sooritada otsing ning kasutajaliideses päringu vastetega tabelis tuleb märkida need kirjed, mille puhul soovitakse muudatust teha. Selleks tuleb vastavad kirjed otsingu tulemuses märkeruudu abil märgistada ning vajutada nuppu “Alusta massmuutmist”.

Tulemusena peab avanema vaade kus on võimalik valida välju (parameetreid, veerge), mida soovitakse eelnevalt valitud kirjete puhul muuta. Valides välja, tuleb järgmisena valida kas soovitakse välja sisu asendada või täiendavat infot lisada. Juhul kui tegemist on tekstiväljaga, siis lisatakse uus tekst olemasoleva väärtuse lõppu. Juhul kui tegemist on massiiviga, siis lisatakse täiendav väärtus massiivi lõppu. Juhul kui on tegemist asendusega, siis asendatakse välja sisu uue sisuga, olenemata sellest, mis liiki väljaga on tegemist ja mis selle sisu oli. Muudatuste kinnitamiseks tuleb vajutada salvestamise nuppu ning selle tulemusena avaneb ülevaade kirjetest kuhu muudatust rakendatakse ja muudatustest, mis sinna rakenduvad. Kui see kinnitatakse, siis tehakse andmebaasis vastavad muudatused ning kasutaja viiakse tagasi otsingu lehele.

Eelpool kirjeldatud funktsionaalsusest on praeguseks realiseeritud massmuutmine lihtsustatud vormis. Valides otsingu tulemustest massmuutmiseks soovitud viisid, avaneb viisi väljade valiku vaade. Pärast väljade valiku tegemist ning “edasi” nupule vajutamist avaneb uus vaade, kus kuvatakse tekstilahtrid muudetavate väljade väärtuste jaoks ning tabel, mille veergudeks on eelnevalt valitud väljad. Pärast tekstilahtrite täitmist ning

tabelist viiside valimist (Joonis 65) tuleb vajutada salvestamise nuppu, mille tagajärjel rakendatakse valitud viisidel *update* operatsioon ning kirjutatakse antud väljad viisides üle.



Joonis 65 Massmuutmise vaade.

6.1.8 Viidete lisamine Kivikese infosüsteemile

Selleks, et võimaldada loodud infosüsteemi kirjete sidumist Kivikese süsteemis registreeritud kirjetega, lisati viisi vaatesse võimalus kirjete juurde lisada Kivikese objektide PID-e, mille abil on võimalik kasutajaid suunata täpse arhivaali juurde. See on tähtis, kuna üks EKM-i soovidest oli võimalus hakata kokku viima loodava infosüsteemi kirjeid Kivikeses asuvate originaalteostega. Selleks lisati viisi vaatesse eraldi sektsioon nimega “Välisviited” (Joonis 66), kuhu on võimalik lisada otseviiteid arhivaalidele. Nende kasutamiseks on vaja omistada arhivaalile kirjeldav nimetus ning arhivaali täpne PID.

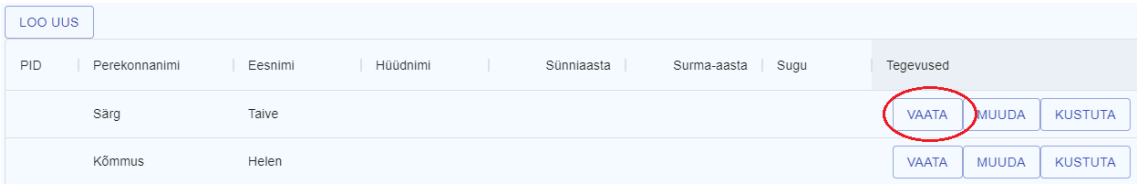
Joonis 66 Välisviited viisi vaates.

“Ava Kivikeses” nupule vajutades suunatakse kasutaja Kivikese veebilehele, kus avatakse vastava PID-iga arhivaal.

6.2 Olemasoleva funktsionaalsuse uusversioon

Järgnevalt kirjeldatakse bakalaureusetöös [7] realiseeritud funktsionaalsuste uusversiooni käesoleva töö tulemusena valminud süsteemis.

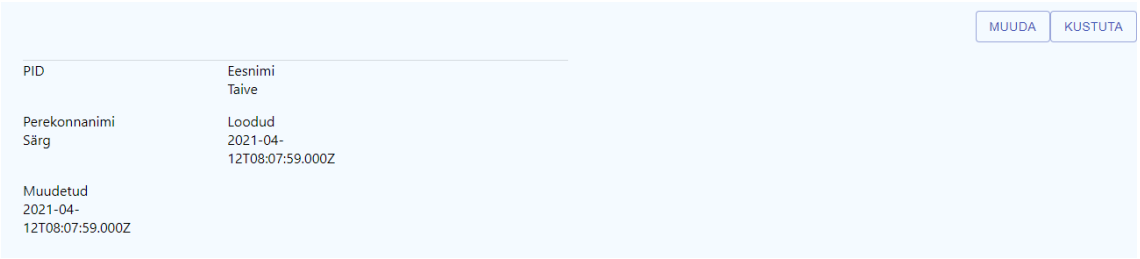
6.2.1 Isikute vaatamine



Joonis 67 Vaade "Isikute vaatamine".

Isikute vaates on võimalik näha rahvaviisidega seotud isikuid. Kuvatakse PID, isiku ees- ja perekonnanimi ning olemasolul ka hüüdnimi, sünni- ja surma-aasta ja sugu. Antud vaates on võimalik isikuid lisada, nendega seotud informatsiooni vaadata ja muuta. Võimalik on ka isikute deaktiveerimine (Joonis 67).

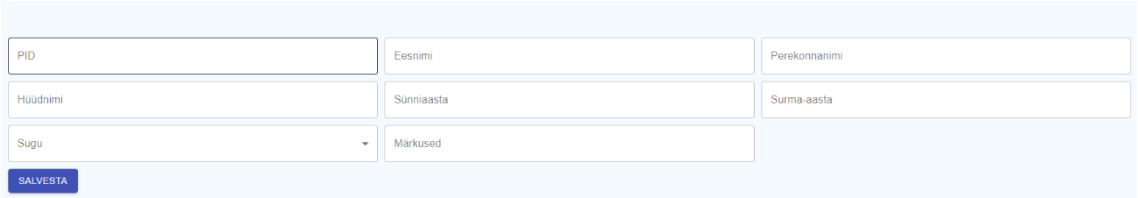
Vajutades nupule “Vaata” avaneb vaade isiku detailvaatega (Joonis 68). Detailvaates on olemas veel muutmise ja deaktiveerimise funktsioon.



Joonis 68 Isikute detailvaade.

6.2.2 Isikute lisamine

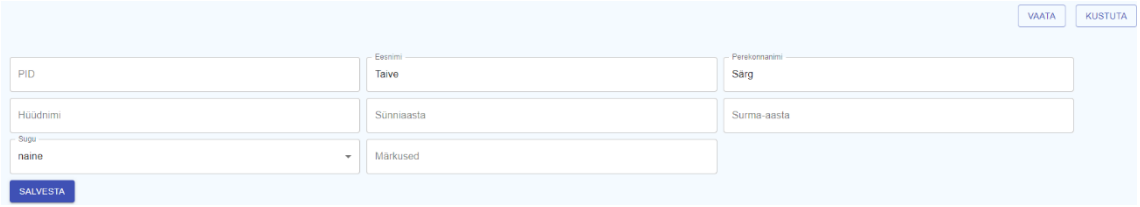
Isikute vaatest on võimalik uut isikut luua. Selleks tuleb vajutada nupule „Loo uus“ (Joonis 67). Seejärel avaneb vaade, kus saab isikuga seotud informatsiooni vastavasse välja lisada. Täitmiseks pakutavad väljad on: ees- ja perekonnanimi, hüüdnimi, sünni- ja surma-aasta ning sugu. Lisaks on olemas ka „Märkused“ väli, kuhu saab olemasolul lisainformatsiooni lisada. PID on kasutusel selleks, et isikut kindla viisiga siduda. Kui andmeväljad on täidetud, siis tuleb sisestatud andmed salvestada vajutades nupule „Salvesta“. (Joonis 69)



Joonis 69 Vaade "Isiku lisamine".

6.2.3 Isikute muutmine

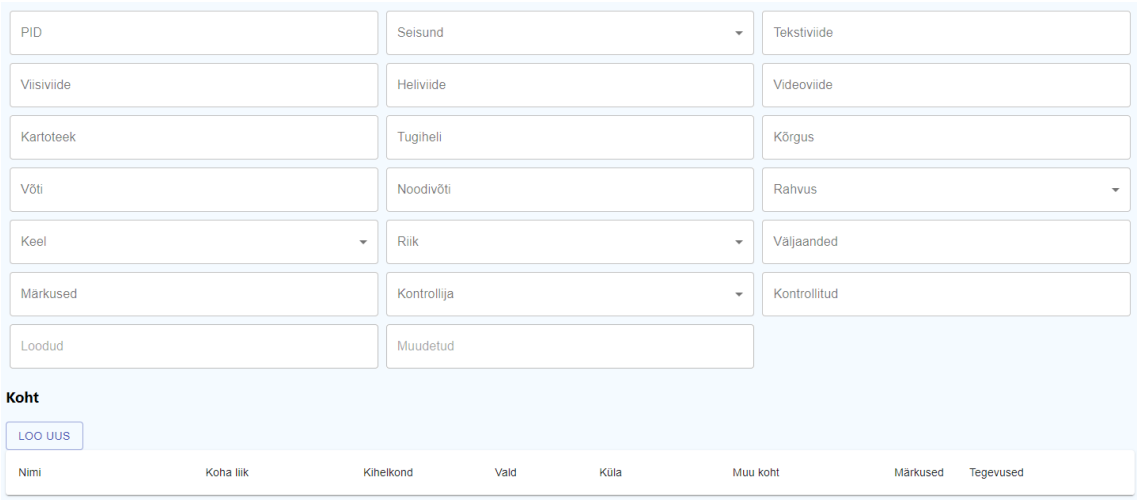
Isikutega seotud andmeid on võimalik muuta nii isikute vaatest (Joonis 67), kui ka isiku detailvaatest (Joonis 68). Muutmiseks avaneb vaade väljadega, kus on osad juba olemasoleva informatsiooniga eeltäidetud (Joonis 70). Peale andmete lisamist ja muutmis tuleb toimingut kinnitada vajutades nupule „Salvesta“.



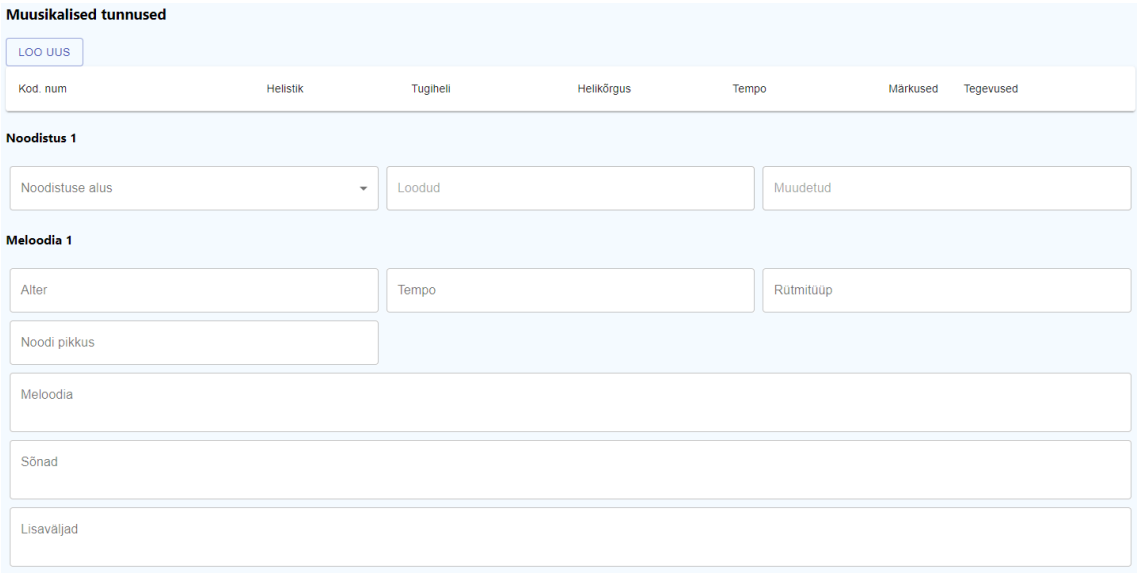
Joonis 70 Vaade "Isikute muutmine".

6.2.4 Viisi lisamine

Uue viisi lisamisega saab alustada viisi vaatest (Joonis 74). Lehe vasakus nurgas on nupp „Loo uus“, mille vajutamisega alustatakse viisi lisamise protsessi (Joonis 71, Joonis 72). Avaneb vaade kõikide viisiga seotud väljadega. Viisi lisamise protsess lõpeb salvestamisega.



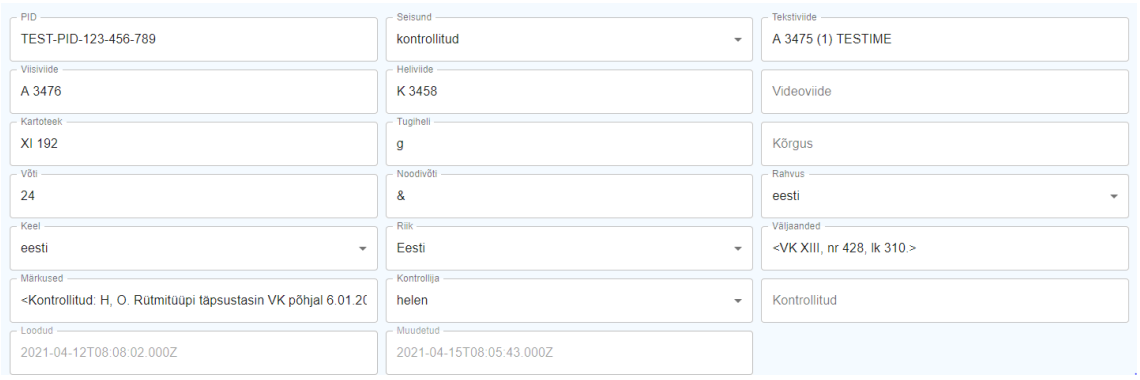
Joonis 71 Vaade "Viisi lisamine".



Joonis 72 Vaade "Viisi lisamine".

6.2.5 Viisi muutmine

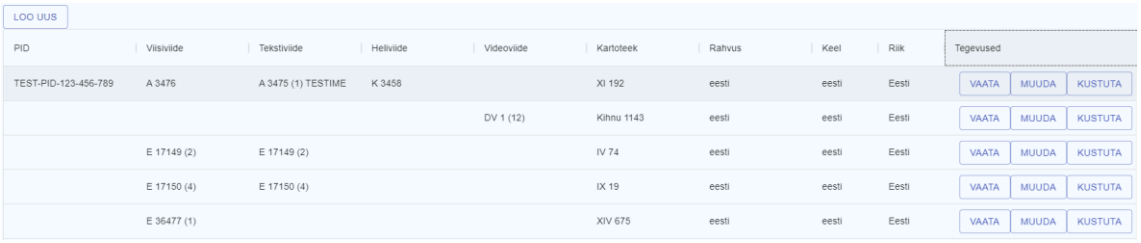
Viisi muutmisega alustatakse kas viisi vaatest (Joonis 74) või viisi detailvaatest (Joonis 75) vajutades nupule „Muuda“. Seejärel avaneb vaade (Joonis 73) kõikidest viisiga seotud väljadest. Eeltäidetud on need väljad, millele on andmed juba sisestatud. Neid välju saab täiendada, sh väljast väärtust kustutada. Tühjadele väljadele saab informatsiooni juurde lisada. Protsess lõpeb „Salvesta“ nupu vajutamisega. Kui antud vaatest lahkuda enne „Salvesta“ vajutamist, siis ei salvestata andmeid süsteemi.



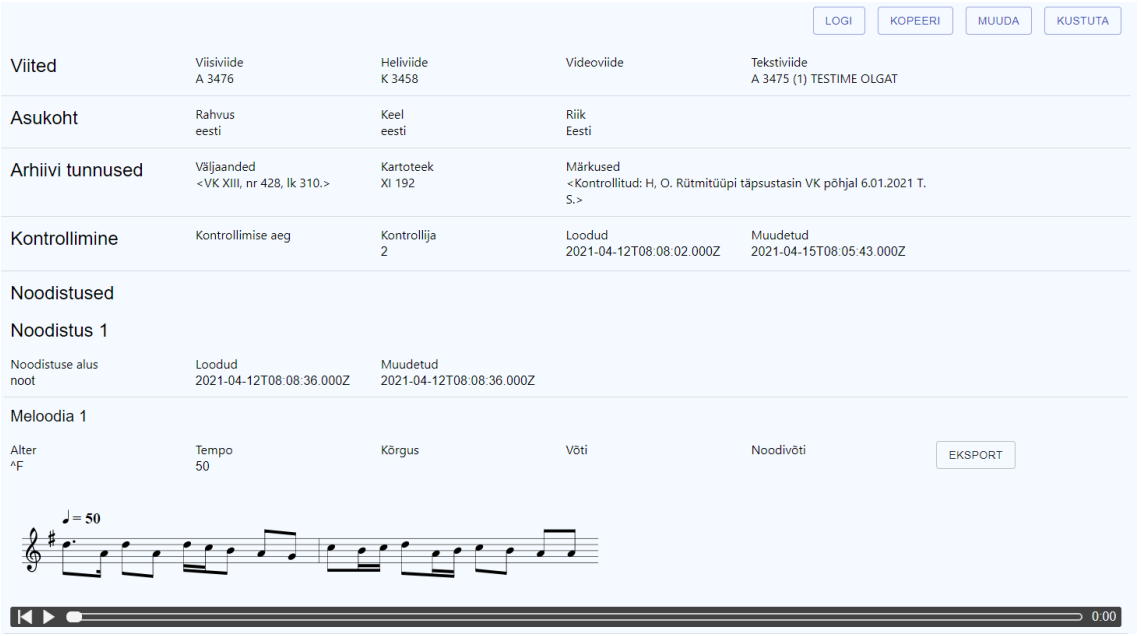
Joonis 73 Vaade "Viisi muutmine".

6.2.6 Viisi vaatamine

Viisi vaatamist alustatakse viisi vaatest (Joonis 74) või otsingu vaatest, kui kindel viis on leitud. Mõlemal juhul tekib paremasse serva „Vaata“ nupp, mis suunab kasutajat viisi detailvaatesse (Joonis 75).



Joonis 74 Viisi vaade.



Joonis 75 Vaade "Viisi detailvaade".

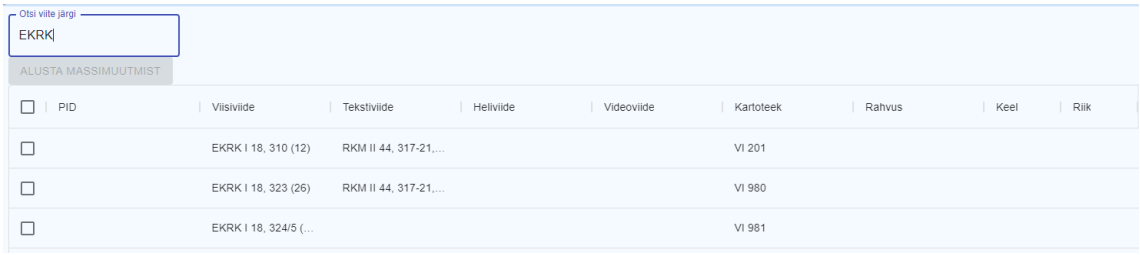
6.2.7 Viisi kustutamine

Viisi kustutamiseks tuleb viisi vaates (Joonis 76) vajutada nupule „Kustuta“, mille järel viis koos sellega seotud andmetega kustutatakse.

Joonis 76 Viisi vaate kustuta nupp.

6.2.8 Viisi otsimine

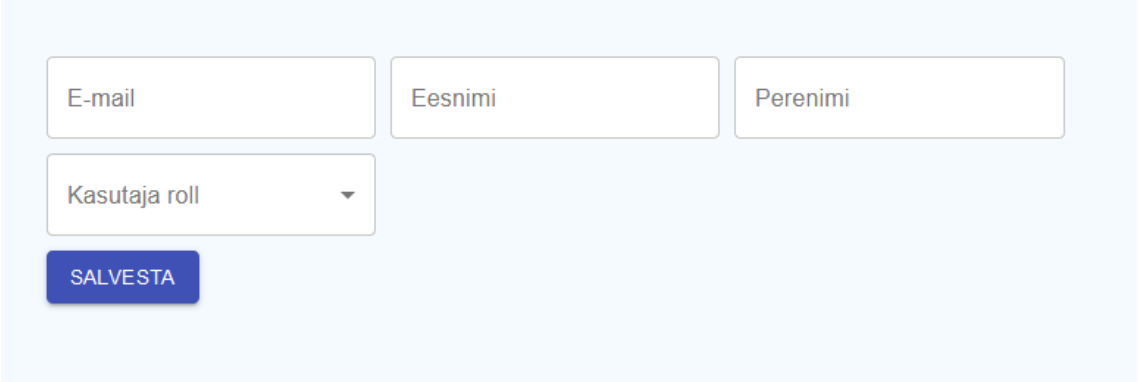
Viisi otsimiseks tuleb minna otsingu vaatesse kust avaneb võimalus viisi viite järgi otsida. Sisestades otsingulahtrisse viite tekib tulemuste tabel (Joonis 77).



Joonis 77 Otsingu tulemuste kuvamine tabelina.

6.2.9 Kasutaja lisamine

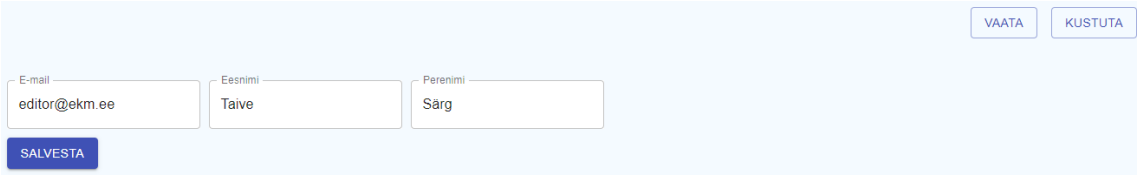
Kasutaja lisamiseks tuleb olla sisse logitud ja vastavate õigustega kasutaja. Kasutajaid saab lisada administraator. Selleks tuleb avada kasutajate vaade (Joonis 80). Vajutades nupule „Loo uus“ avaneb vaade uue kasutaja andmete sisestamiseks (Joonis 78). Kasutaja lisamine lõpeb andmete salvestamisega.



Joonis 78 Kasutaja lisamise toiming.

6.2.10 Kasutaja muutmine

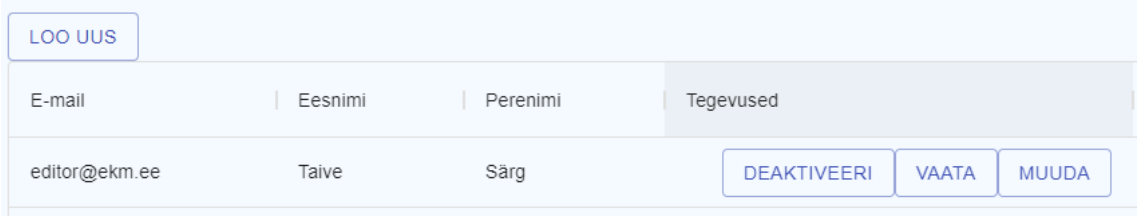
Kasutaja muutmist saab alustada kasutaja vaatest (Joonis 80) ja kasutaja detailvaatest (Joonis 81). Vajutades nupule „Muuda“ avaneb vaade koos kasutaja andmetega: meiliaadress, eesnimi ja perekonnanimi). Väljad on olemasoleva infoga eeltäidetud ja muutmiseks kuvatud (Joonis 79). Protsess lõpeb andmete salvestamisega.



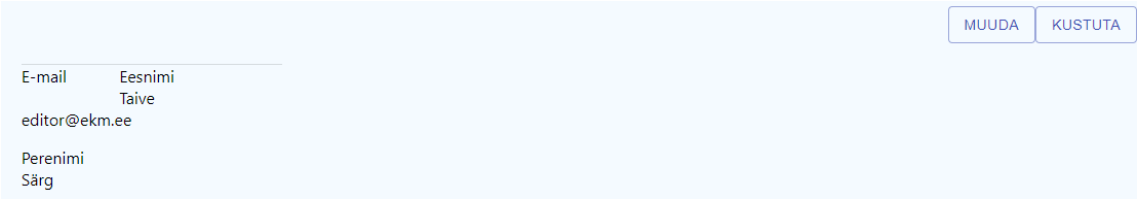
Joonis 79 Vaade "kasutaja muutmine".

6.2.11 Kasutaja vaatamine

Kasutajad saab vaadata kasutaja vaatest. Korraga kuvatakse kasutaja meiliaadress ja ta ees- ja perenimi (Joonis 80). Vajutades nupule „Vaata“ liigub kasutaja edasi kasutaja detailvaatesse (Joonis 81).



Joonis 80 Vaade "Kasutaja vaatamine".



Joonis 81 Kasutaja detailvaade.

6.2.12 Kasutaja aktiveerimine/deaktiveerimine

Kasutajate aktiveerimine ja deaktiveerimine toimib kasutaja vaatest, vajutades nuppu „Deaktiveeri“ (Joonis 80). Kui kasutaja on deaktiveeritud, siis muutub antud nupp aktiveerimise nupuks.

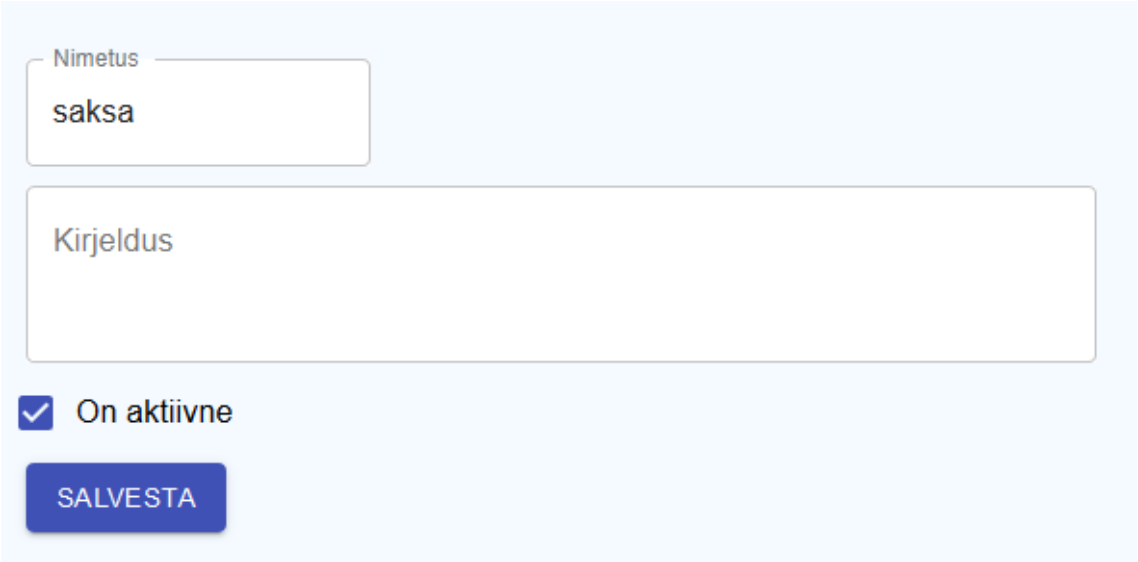
6.2.13 Klassifikaatori väärtuste vaatamine

Klassifikaatori väärtuste vaatamine toimub klassifikaatorite kaupa. Minnes klassifikaatorite vaatesse, saab valida, milliseid klassifikaatori väärtuseid vaadata. Klassifikaatorite vaates on võimalik neid vaadata, muuta ja kustutada. Samuti on võimalik lisada uusi klassifikaatori väärtuseid (Joonis 82).

Joonis 82 Klassifikaatorite vaatamine.

6.2.14 Klassifikaatorite väärtuste lisamine

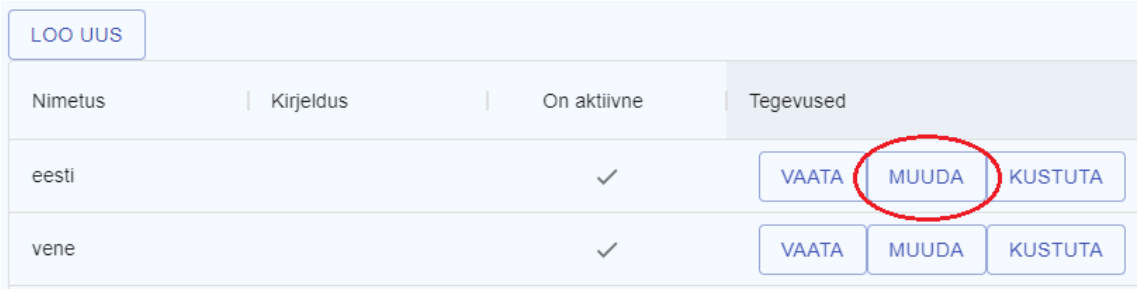
Lisamine toimub klassifikaatorite vaates vajutades nupule „Loo uus“. Pärast nupule vajutamist avaneb vaade väljadega klassifikaatori väärtuse kirjeldamiseks (Joonis 83).



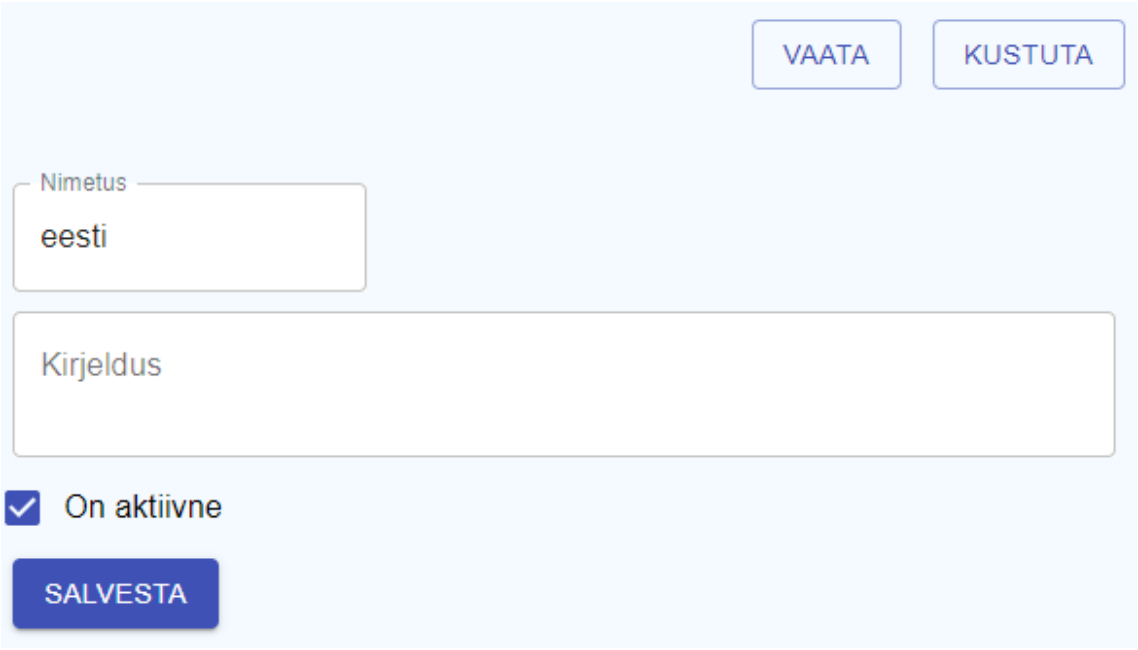
Joonis 83 Klassifikaatori lisamine.

6.2.15 Klassifikaatori väärtuste muutmine

Klassifikaatori väärtuste muutmine toimub klassifikaatorite vaates (Joonis 84). Muuda nupule vajutades avaneb vaade väljadega „nimetus“ ja „kirjeldus“, kus on võimalik täiendada klassifikaatori väärtusega seotud informatsiooni. Protsess lõppeb salvestamisega (Joonis 85).



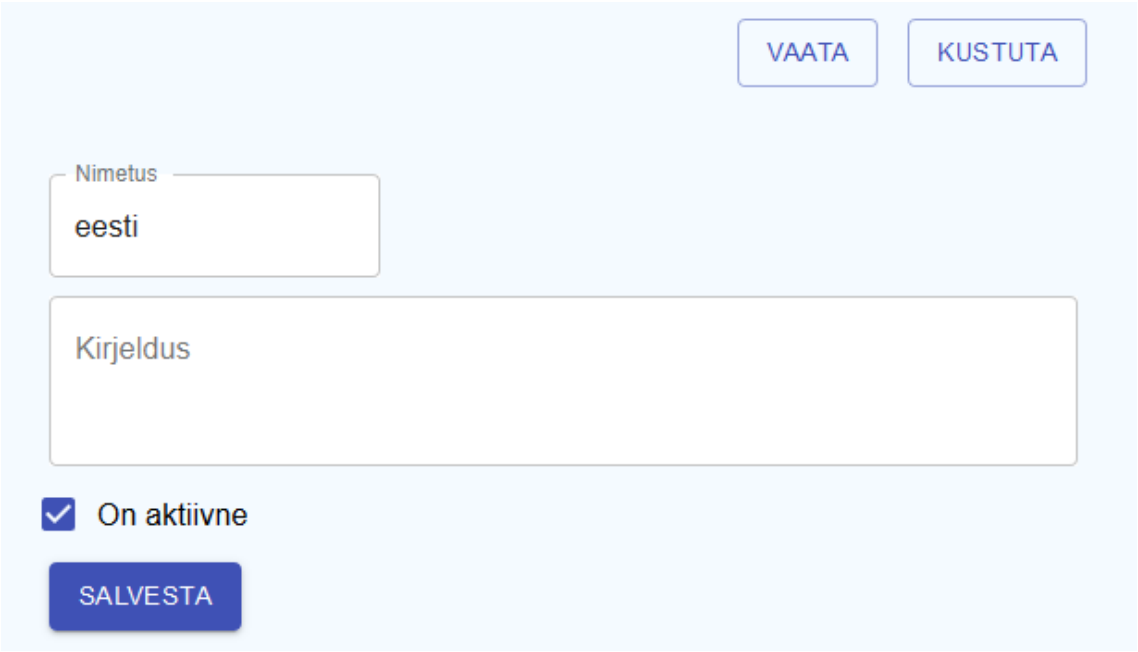
Joonis 84 Klassifikaatori väärtuse muutmine.



Joonis 85 Klassifikaatori väärtuse muutmise vormi väljad.

6.2.16 Klassifikaatori väärtuse aktiveerimine/deaktiveerimine

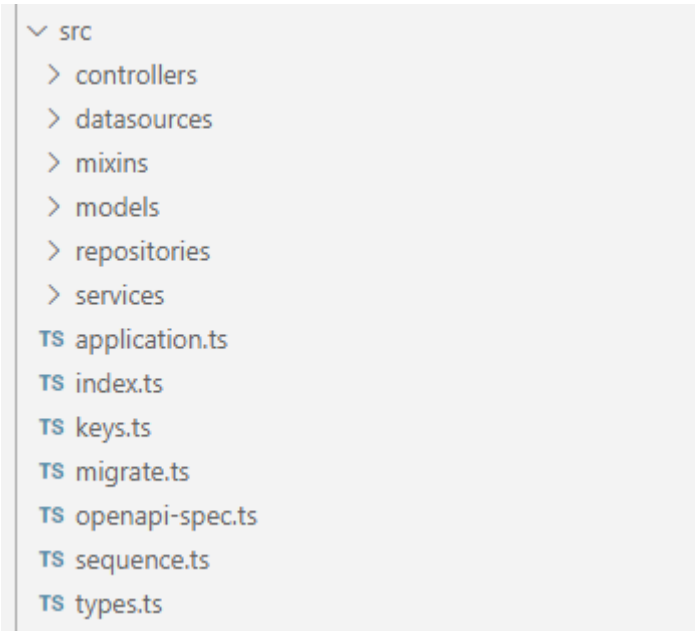
Klassifikaatori väärtuse aktiveerimiseks ning deaktiveerimiseks tuleb alustada selle muutmist. Valides klassifikaatorite vaatest „Muuda“ funktsionaalsuse, saab toimingut alustada. Kolmas lahter, nimetusega „on aktiivne“, on kasutusel selleks, et klassifikaatori väärtust aktiveerida ning deaktiveerida. Seda tehakse märkeruudus valiku tegemisega. (Joonis 86).



Joonis 86 Klassifikaatori väärtuse aktiveerimine/deaktiveerimine.

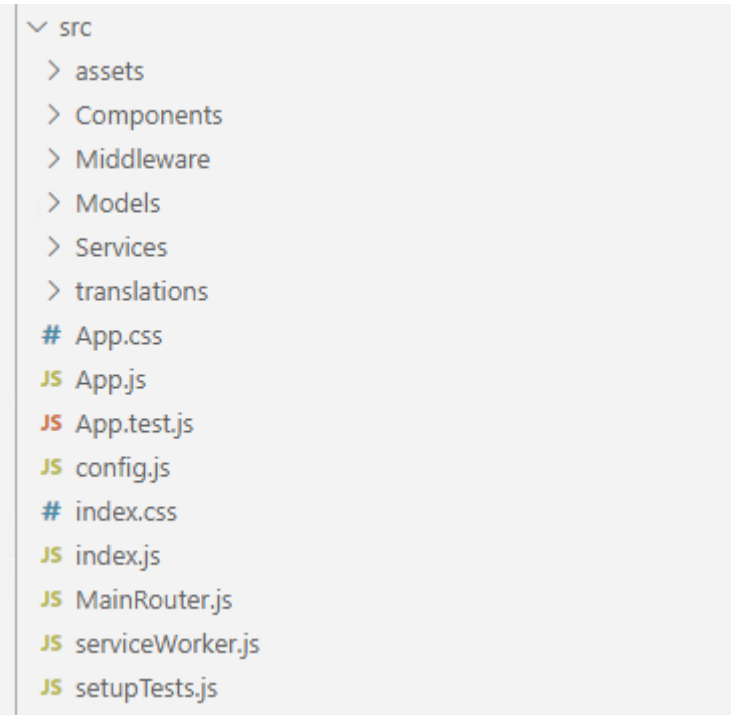
6.3 Rakenduse lähtekood

Rakenduse lüüsikihi ning esitluskihi lähtekood juhindub taaskasutuse põhimõttest. Tagamaks komponentide ühtlus ja muutmise lihtsus on kogu mitmekordselt kasutatav funktsionaalsus nii lüüsikihis kui esitluskihis viidud teenustesse ning kasutajaliidese komponendid on defineeritud eraldiseisvalt vaadetest.



Joonis 87 Lüüsikihi lähtekoodi struktuur.

Lüüsikiht on loodud programmeerimiskeelega TypeScript, mis kompileeritakse JavaScriptiks ning seejärel minimiseeritakse ning segatakse. Segamise e inetamise all mõeldakse *uglify* funktsionaalsust, mis nimetab lähtekoodi muutujad ühetähelisteks ning kaotab kõik tühimikud ja kommentaarid koodist. Nagu on näha Joonis 87, siis rakenduse lähtekood on jagatud kuute kausta. Otspunkte loovad kontrollerid on defineeritud kaustas *controllers*, kontrollerid kasutavad mudeleid kaustas *models* ning repositooriume kaustas *repositories*. Repositooriumitele on defineeritud andmeallikad, mis on kirjeldatud kaustas *datasources*. Kontrollerite poolt kasutatavad teenused on kaustas *services*. Kaustas *mixins* asub logimise funktsionaalsuse jaoks repositooriumi funktsionaalsust laiendav klass.



Joonis 88 Esitluskihi lähtekoodi struktuur.

Esitluskiht põhineb programmeerimiskeelel JavaScript. JavaScript keelt ei kompileerita kuid nagu ka lüüsikihi puhul minimiseeritakse ning segatakse kood inimese poolt loetamatuks, selleks, et suurendada rakenduse turvalisust. Erinevus lüüsikihiga tuleneb sellest, et esitluskihi kood laetakse alla kliendi veebilehitsejasse, kuid lüüsikihi koodi ei ole võimalik avalikult näha.

Joonis 88 on näha, et esitluskihi lähtekood jaguneb viite kausta. Komponendid asetatakse kausta *Components* grupeerides neid kasutuskoha järgi vastavatesse alamkaustadesse. Komponentidega luuakse vaateid vastavalt mudelitele, mis on defineeritud kaustas *Models*. Komponentide poolt kasutatavad teenused ning tõlked on vastavalt kaustades *Services* ja *translations*. Teenustele ja komponentidele rakendatakse kaustas *Middleware* asuvad vahevarad. *assets* kaust sisaldab endas vaid staatilisi ressursse.

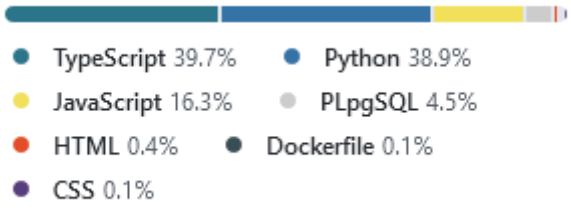
6.3.1 Lähtekoodi statistika

Projekti koodirepositooriumis asuvates failides on kokku 99 223 rida.

Sellest 69 784 on lüüsikihi rakenduses ning 23 292 esitluskihi rakenduses, ülejäänud read asuvad muudes projektiga seotud failides. Eraldi tuleb välja tuua veel *package_lock.json* failid ning konverteerimiseks kasutatavad Pythoni programmid (*abc2xml* ja *xml2abc*, mida töö autorid ei kirjutanud), millel on siin suur mõju – kahe programmi peale kokku

umbes 40 000 rida. Need eemaldades jääb rakenduste lähtekoodi ridade arvuks umbkaudselt 53 000 rida, millest ligikaudu 35 000 on SQL skriptid, mis on kasutusel viisi andmete sisestamiseks. See tähendab, et loodud rakenduste jaoks loodi autorite poolt umbes 18 000 rida koodi.

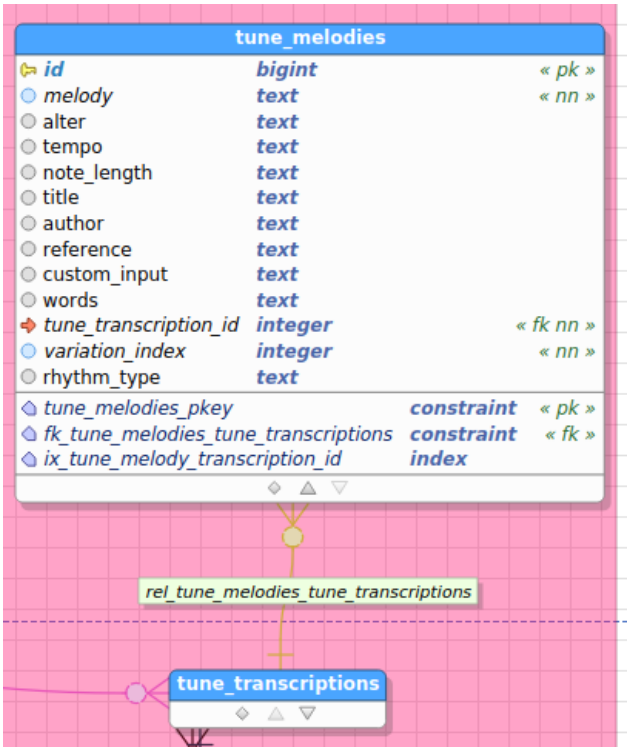
Valdaval enamus rakenduste programmeerimiskeele jaotusest oli lüüsikihi TypeScript, millele järgnes esitluskihi JavaScript. Github statistika (Joonis 89) on konverteerimisprogrammide tõttu küll tõstnud Pythoni teisele kohale, kuid muus osas vastab jaotus autorite poolt loodud lähtekoodi omavahelistele suhetele.



Joonis 89 Github statistika programmeerimiskeelte jagunemise kohta.

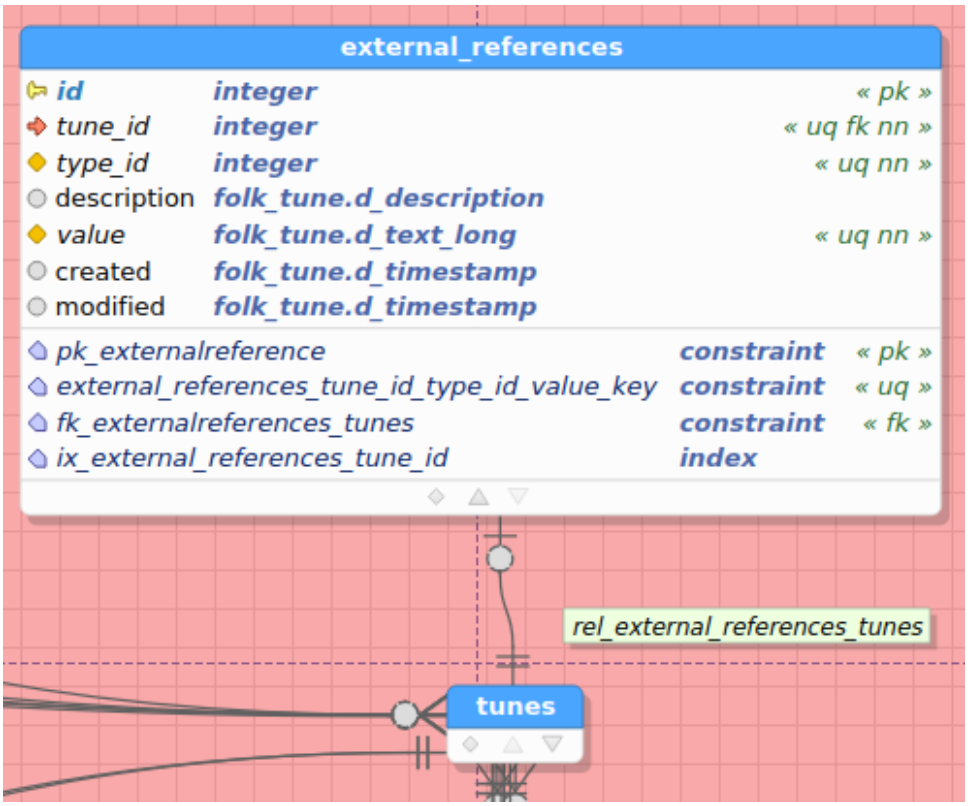
6.4 Andmebaasi struktuuri muutmine

Seoses täiendava funktsionaalsuse lisamisega tekkis vajadus laiendada andmebaasi selliselt, et oleks võimalik realiseerida kogu eesmärgiks seatud funktsionaalsus. Selleks loodi andmebaasi järgmised täiendavad baastabelid. Kõikides järgnevalt kirjeldatud baastabelites, kus veerg *id* on täisarvu tüüpi, kasutatakse võtmeväärtuste genereerimiseks PostreSQL-i arvujada generaatorite mehhanismi (veergude defineerimiseks kasutati SERIAL/BIGSERIAL notatsiooni).



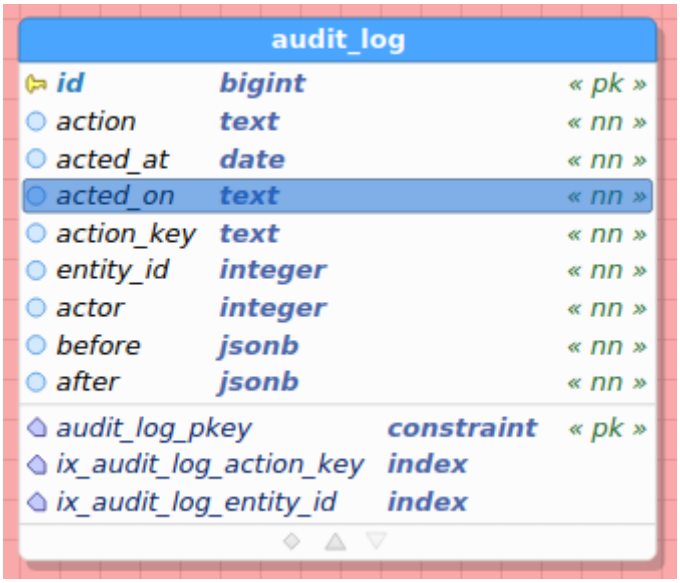
Joonis 90 *tune_melodies* tabeli struktuur.

Tune_melodies tabeli (Joonis 90) eesmärk on hoida ABC kujule kodeeritud noodistust koos noodistusega kaasas käiva informatsiooniga. *Tune_melodies* tabelis on defineeritud veerud vastavalt sellele, mis informatsiooni on ABC kujul noodistuse loomiseks vaja. *Tune_melodies* on seotud *Tune_transcriptions* tabeliga selleks, et oleks võimalik siduda noodistus noodi kodeerimisega. *Tune_transcriptions* tabel on omakord seotud *Tunes* tabeliga, mille kaudu tekib seos noodikirja ja viisi vahel. Tabelisse tekitati indeks veerule *tune_transcription_id*, et kiirendada viisivaates andmete pärimist. Täiendavalt lisati veergudele *tune_transcription_id*, *variation_index* ja *melody* NOT NULL piirang.



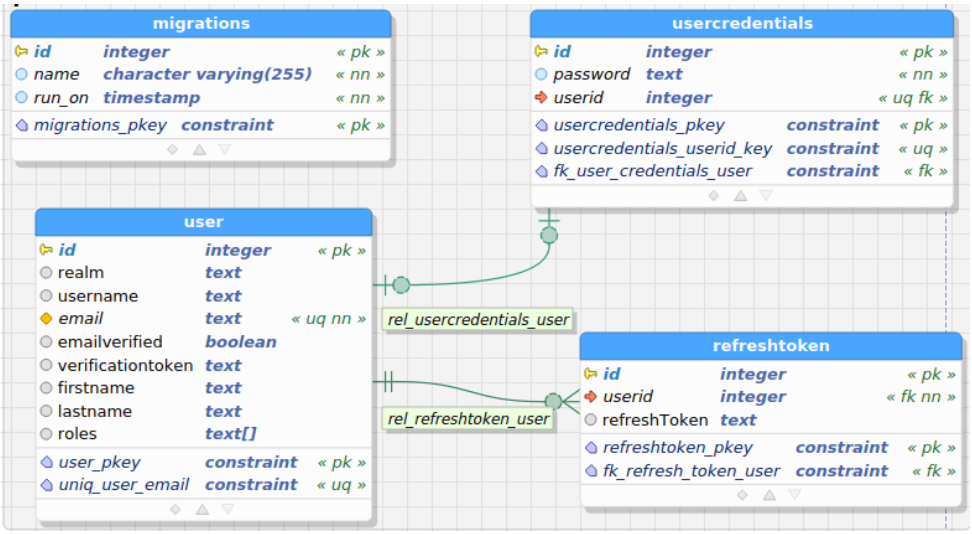
Joonis 91 *external_references* tabeli struktuur.

External_references tabeli (Joonis 91) eesmärk on võimaldada välisviidete lisamist viisile. Tabelis hoitakse viiteliiki (*type_id*), kirjeldust (*description*) ja väärtust (*value*). *type_id* lisati selleks, et kui tulevikus on soov luua täiendavaid viiteliike, siis on selleks valmidus olemas. Antud töös oli ainult üks viiteliik, milleks oli Kivikese viide. *value* ei ole mitte aadress, vaid selle sisuks on PID, mille põhjal genereeritakse Kivikese objektile viitav URL. Tabelile loodi indeks veeru *tune_id* põhjal, mis kiirendab viidete pärimist viisi vaates. Lisaks on piirang NOT NULL seatud veergudele *tune_id*, *type_id* ja *value*.



Joonis 92 *audit_log* tabeli struktuur.

Audit_log tabeli (Joonis 92) eesmärk on luua võimalus viiside muudatuste ajaloo salvestamiseks ning kuvamiseks. *action_key* väärtusega määratakse tabel, milles muudatus tehti ning *entity_id* alusel määratakse muudetud tabeli rea *id*. Kuna logitabelisse tulevad kokku andmed muudatuste kohta erinevatest tabelites, siis pole *entity_id* välisvõtme kitsendusega kaetud. Tabelis on loodud indeks väljade *action_key* ja *entity_id* väärtuste põhjal. *before* ja *after* väljad sisaldavad json objekte, mille sisuks on ainult muudetud väljadega seotud info. *before* väljas on esialgne väärtus, *after* väljas hoitakse väärtus, millega esialgne väärtus asendati. *actor* välja salvestatakse muudatuse teinud kasutaja *id*.



Joonis 93 tabelite *user*, *migrations*, *usercredentials*, *refreshtoken* andmebaasi struktuur.

Tabelis *user* hoitakse kasutajakontodega seonduvat informatsiooni nagu kasutajatunnus, meiliaadress, eesnimi, perekonnanimi ja rollid. Sellega on seotud eraldi tabelid *usercredentials* kus hoitakse kasutaja parooli ja *refreshtoken* kus hoitakse kasutajale genereeritud loa asendust. *migrations* tabelis hoiab lüüsikiht informatsiooni andmebaasil rakendatud migratsioonide kohta (Joonis 93). Igal käivitamisel kontrollib lüüsikiht rakendatud migratsioonide vastavust migratsioonidega lüüsikihi koodis.

6.5 EKM viiside kodeerimise utiliit

Selleks, et automatiseerida noodistuse teisendus ühelt kujult teisele ning minimeerida käsitsi andmete teisendamist, kirjutati teisendusprogramm (utiliit). Programmi eesmärgiks on teisendada EKM-i rahvaviiside andmebaasi tabeli *Viis* veergudes olev informatsioon ABC notatsiooni standardile vastavale kujule. Programm realiseerib funktsionaalsused, mida järgnevalt on kirjeldatud.

- Loodav rakendus peab sisendfailist kõik read läbi töötlema ning nende baasil genereerima ABC noodistuse.
- Juhul kui leitakse väärtused, millele puudub vaste vastavustabelis, siis tuleb tagastada välja väärtus märkides see <> märkide vahele.
- Utiliit peab arvestama sellega, et lähteandmetes on variatsioon märgitud kandiliste sulgude [] vahele ning seal olevad väärtused tuleb panna eraldi reale koos märkega, et tegemist on variatsiooniga.
- Utiliit peab tuvastama noodikõrguseid kolmes oktavis ja vahet tegema nende tähiste vahel.
- Utiliit peab tuvastama eellööke, kiiremaid lööke, dieesi, bemolli ja bekaari ning nende tähised asendama ABC kodeeringu tähistega.
- Utiliit peab oskama vastavalt rütmitüübile luua rütmitüübile vastavat noodistruktuuri.
- Utiliit peab tulemused salvestama CSV faili.

Selle utiliidi keeleks valiti go [67] kuna see oli töö autoritele tuttav ning selle abil oli järelikult võimalik kõige lihtsamini ja kiiremini antud funktsionaalsus realiseerida. Programmis on kokku 334 füüsilist koodirida. Noodistuse kodeerimiseks kulus programmil kolm sekundit. Programm oli põhimõtteliselt mõeldud ühekordseks käivitamiseks kuid töö käigus selgus, et kodeerimist oli vaja teha läbi korduvalt. Põhjuseks oli näiteks see et töö käigus hinnati algandmete kvaliteeti ning tehti andmetes käsitsi parandusi . Andmetes avastatud puudusteks olid näiteks kirjavead (üks sulg ühte moodi ja teine teistmoodi), üleliigsed märgid ning rakenduse jaoks tundmatud tähekombinatsioonid. Teiseks laiendati töö käigus automaatselt kodeeritavate rütmitüüpide ja kombinatsioonide hulka. Kuna see programm võib olla tulevikus kasulik ka teistele muusika kodeerimisega tegelejatele, siis programmi lähtekood on avaldatud GitHubi salves, mille aadress on: https://github.com/nitramra/EKM_converter

Programmi käivitamiseks on vaja programmiga samasse kausta tekitada kaks faili. Fail *Viis.csv* peab vastama struktuurile *ViisStruct* (Joonis 94).

```
//ViisStruct Viis.csv struktuur
type ViisStruct []struct {
    IDviis string `csv:"IDviis"`
    FKey   string `csv:"FKey"`
    VNr    string `csv:"VNr"`
    P1     string `csv:"P1"`
    P2     string `csv:"P2"`
    P3     string `csv:"P3"`
    P4     string `csv:"P4"`
    P5     string `csv:"P5"`
    P6     string `csv:"P6"`
    P7     string `csv:"P7"`
    P8     string `csv:"P8"`
    P9     string `csv:"P9"`
    P10    string `csv:"P10"`
    P11    string `csv:"P11"`
    P12    string `csv:"P12"`
    P13    string `csv:"P13"`
    P14    string `csv:"P14"`
    P15    string `csv:"P15"`
    P16    string `csv:"P16"`
}
```

Joonis 94 *ViisStruct* struktuur.

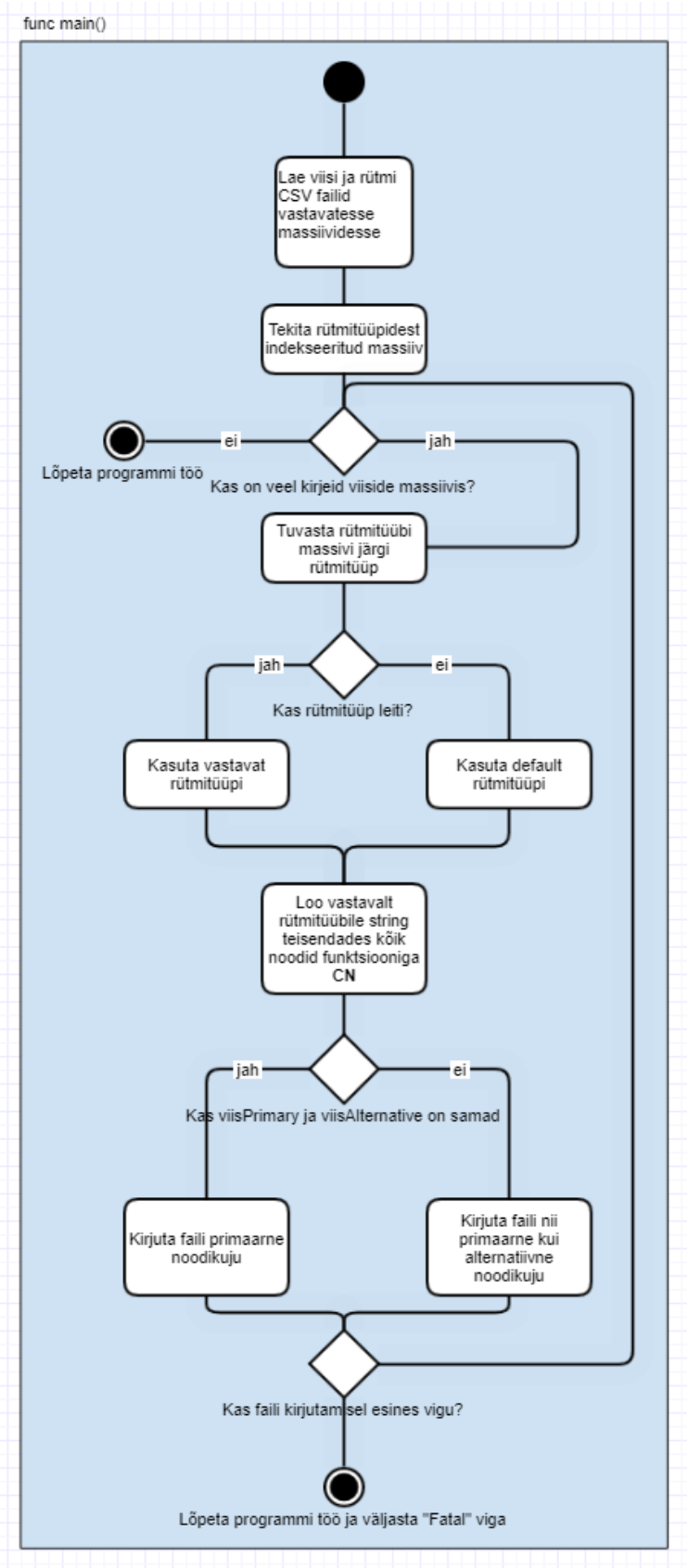
Lisaks sellele peab olema fail *rytmityybid.csv*, mis peab vastama struktuurile *RytmiStruct* (Joonis 95):

```
//RytmiStruct rytmityybid.csv struktuur
type RytmStruct []struct {
    Fkey      string `csv:"Fkey"`
    Nr        string `csv:"Nr"`
    Rytmiliik string `csv:"Rytmiliik"`
}
```

Joonis 95 RytmiStruct struktuur.

Juhul kui need tingimused on täidetud, saab programmi käivitada.

6.5.1 Funktsioon main()



Joonis 96 Funktsioon main() tegevusdiagramm.

Programmi käivitamisel käivitatakse funktsioon *main* ning laetakse eelpool kirjeldatud failide sisu massiivi kasutades *gocsv.UnmarshalFile* funktsiooni, mis võimaldab CSV faili otse massiivi laadida (Joonis 96). Selleks, et teisendamisel hõlpsasti leida rütmitüüp, teisendatakse CSV baasil massiiv map[string]string tüüpi massiiviks, mille tulemusena on võimalik viisi ID järgi leida sellele vastav rütmitüüp (Joonis 97).

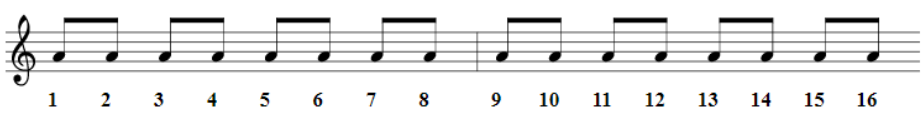
```
//RytmiMap rütmitüübi otsimiseks
RytmiMap := make(map[string]string)

for _, r := range RytmiArray {
  RytmiMap[r.Fkey] = r.Rytmiliik
}
```

Joonis 97 RytmiMap struktuuri loomine.

Järgnevalt alustatakse kogu *ViisiArray* massiivi läbitöötamist for tsükliga. *ViisiArray* on massiiv kus asuvad kõik viisid vastavalt eelnevalt defineeritud struktuurile *ViisStruct*. "switch RytmiMap[q.FKey]" funktsiooni abil otsitakse rütmitüüpide massiivist vastava viisi rütmitüüpi. Võrdlemiseks kasutatakse case funktsiooni ning juhul kui tegemist on defineeritud rütmitüübiga, siis koostatakse noodistus vastavalt rütmitüübile. Muul juhul kasutatakse vaikimisi rütmitüüpi, mis tähendab, et kogu teisenduse tulemus tagastatakse pika jadana. Selle mõte on, et hilisemal käsitsi kodeerimisel on lihtsam noodistust luua lisamise abil ja ei ole vajadust märkide/tühikute eemaldamiseks. Noodistus koostatakse tekstielementide kokku liitmise teel ning järgnevalt on toodud neli erinevat näidet:

1. Rütmitüüp 1 ehk kõige levinum. Antud juhul on tegemist rütmitüübiga kus mõlemad variandid ja viisi pooled kodeeritakse ühtemoodi. Rütmitüübid on üldiselt kaheksakohalised, kuid andmebaasis olevad kirjed olid jaotatud 16 kohale. Seega on antud näite puhul nii positsioonid 1–8 kui 9–16 kodeeritud vastavalt rütmitüübile 1 (Tabel 10).

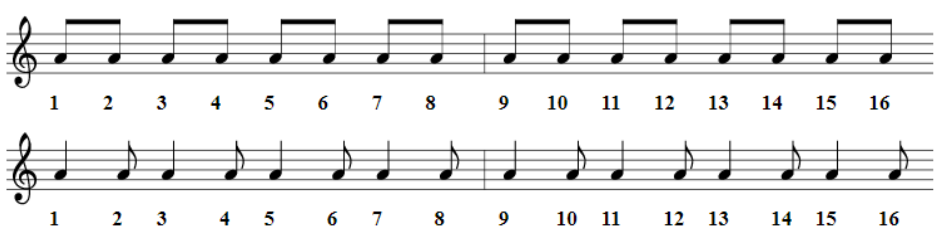
Tabel 10 Rütmitüübi näide, kus mõlemad variandid ja viisi pooled kodeeritakse sama moodi.

ABC kuju	AA AA AA AA | AA AA AA AA
Noodistus	

Lähtekood	case "1":
	  viisPrimary = CN(q.P1, 1) + CN(q.P2, 1) + " " + CN(q.P3, 1) + CN(q.P4, 1) + " " + CN(q.P5, 1) + CN(q.P6, 1) + " " + CN(q.P7, 1) + CN(q.P8, 1) + " | " + CN(q.P9, 1) + CN(q.P10, 1) + " " + CN(q.P11, 1) + CN(q.P12, 1) + " " + CN(q.P13, 1) + CN(q.P14, 1) + " " + CN(q.P15, 1) + CN(q.P16, 1)
	  viisAlterna = CN(q.P1, 2) + CN(q.P2, 2) + " " + CN(q.P3, 2) + CN(q.P4, 2) + " " + CN(q.P5, 2) + CN(q.P6, 2) + " " + CN(q.P7, 2) + CN(q.P8, 2) + " | " + CN(q.P9, 2) + CN(q.P10, 2) + " " + CN(q.P11, 2) + CN(q.P12, 2) + " " + CN(q.P13, 2) + CN(q.P14, 2) + " " + CN(q.P15, 2) + CN(q.P16, 2)

2. Rütmitüüp “1 ja 9” puhul on tegemist sellise variandiga kus variatsioon on teistsuguse rütmitüübiga. Selle näite puhul vastab *viisPrimary* rütmitüübile 1 ja *viisAlterna* rütmitüübile 9. Allpool on näha, et tekib kaks kirjet, millest esimene vastab rütmitüübile 1 ning teine vastab rütmitüübile 9 (Tabel 11).

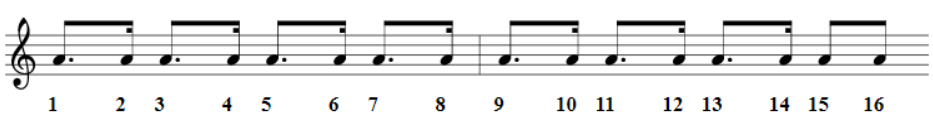
Tabel 11 Rütmitüübi näide, kus variatsioon on teistsuguse rütmitüübiga.

ABC kuju	AA AA AA AA | AA AA AA AA A2 A A2 A A2 A A2 A | A2 A A2 A A2 A A2 A
Noodistus	

Lähtekood	case "1 ja 9":
	  viisPrimary = CN(q.P1, 1) + CN(q.P2, 1) + " " + CN(q.P3, 1) + CN(q.P4, 1) + " " + CN(q.P5, 1) + CN(q.P6, 1) + " " + CN(q.P7, 1) + CN(q.P8, 1) + " | " + CN(q.P9, 1) + CN(q.P10, 1) + " " + CN(q.P11, 1) + CN(q.P12, 1) + " " + CN(q.P13, 1) + CN(q.P14, 1) + " " + CN(q.P15, 1) + CN(q.P16, 1)
	  viisAlterna = CN(q.P1, 2) + "2 " + CN(q.P2, 2) + " " + CN(q.P3, 2) + "2 " + CN(q.P4, 2) + " " + CN(q.P5, 2) + "2 " + CN(q.P6, 2) + " " + CN(q.P7, 2) + "2 " + CN(q.P8, 2) + " | " + CN(q.P9, 2) + "2 " + CN(q.P10, 2) + " " + CN(q.P11, 2) + "2 " + CN(q.P12, 2) + " " + CN(q.P13, 2) + "2 " + CN(q.P14, 2) + " " + CN(q.P15, 2) + "2 " + CN(q.P16, 2)

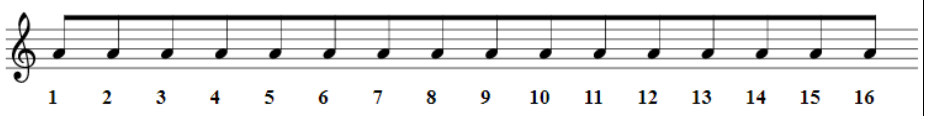
3. Rütmitüüp “9B/9F” puhul on tegemist rütmitüübiga kus esimene pool vastab ühele tüübile ning teine pool teisele. Sellised rütmitüübid olid kaldkriipsuga eraldatud (Tabel 12).

Tabel 12 Rütmitüübi näide, kus esimene pool vastab ühele tüübile ning teine pool teisele.

ABC kuju	A>A A>A A>A A>A | A>A A>A A>A AA
Noodistus	
Lähtekood	`case "9B/9F":` ` viisPrimary = CN(q.P1, 1) + ">" + CN(q.P2, 1) + " " + CN(q.P3, 1) + ">" + CN(q.P4, 1) + " " + CN(q.P5, 1) + ">" + CN(q.P6, 1) + " " + CN(q.P7, 1) + ">" + CN(q.P8, 1) + " | " + CN(q.P9, 1) + ">" + CN(q.P10, 1) + " " + CN(q.P11, 1) + ">" + CN(q.P12, 1) + " " + CN(q.P13, 1) + ">" + CN(q.P14, 1) + " " + CN(q.P15, 1) + CN(q.P16, 1)` ` viisAlterna = CN(q.P1, 2) + ">" + CN(q.P2, 2) + " " + CN(q.P3, 2) + ">" + CN(q.P4, 2) + " " + CN(q.P5, 2) + ">" + CN(q.P6, 2) + " " + CN(q.P7, 2) + ">" + CN(q.P8, 2) + " | " + CN(q.P9, 2) + ">" + CN(q.P10, 2) + " " + CN(q.P11, 2) + ">" + CN(q.P12, 2) + " " + CN(q.P13, 2) + ">" + CN(q.P14, 2) + " " + CN(q.P15, 2) + CN(q.P16, 2)`

4. Rütmitüüp *default* on kasutusel juhul kui tegemist on rütmitüübiga, mille kohta puudub kodeering. Antud lähenemine oli vajalik kuna rütmitüüpide hulk oli suur ning palju oli rütmitüüpe, milles oli alla viie kirje. Seetõttu hinnati mõistlikumaks sellised noodistused käsitsi kodeerida. Samuti kehtis see lähenemine noodistuste puhul kus rütmitüüp oli määramata või vigane (Tabel 13).

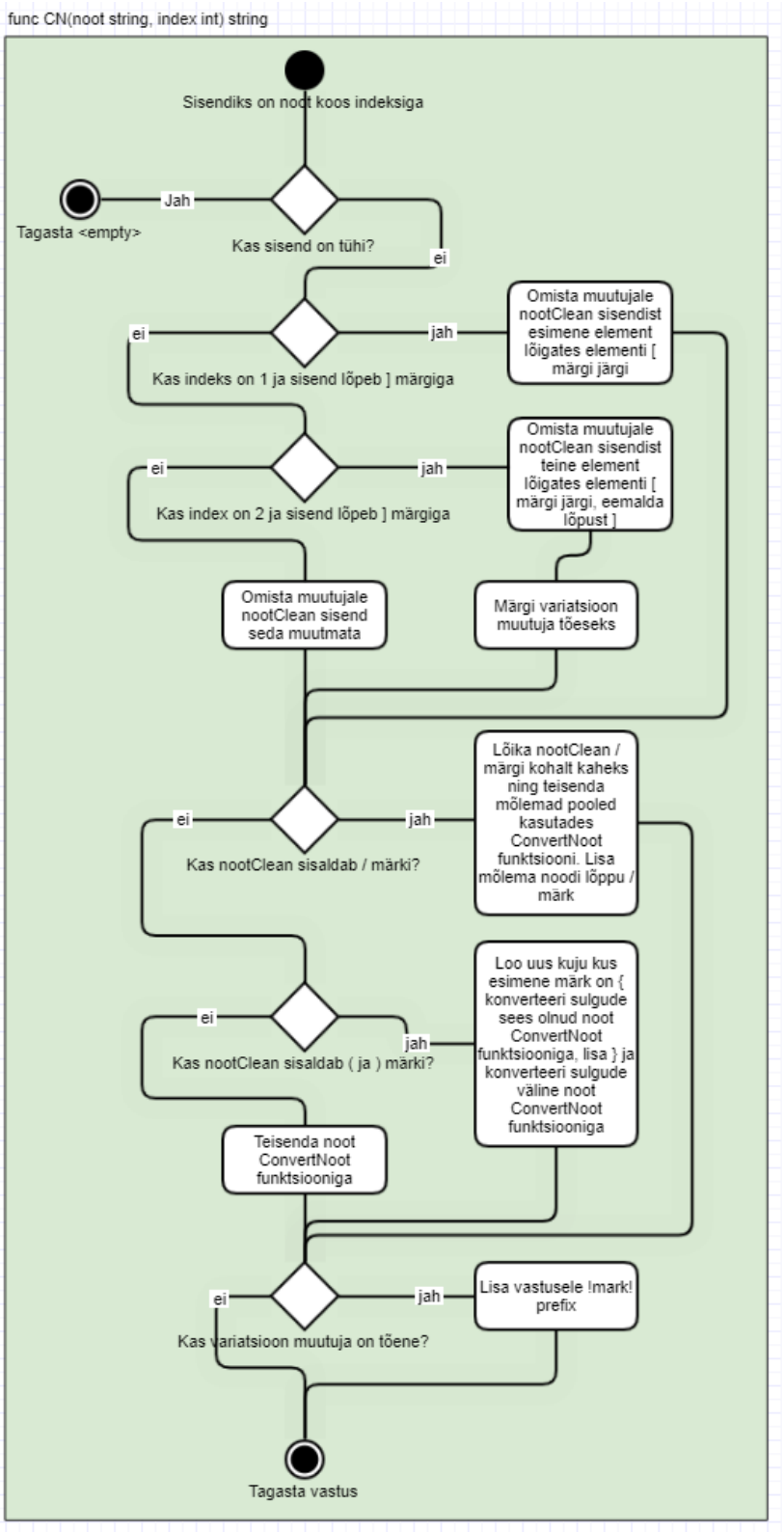
Tabel 13 Näite rütmitüüp default-ist.

ABC kuju	AAAAAAAAAAAAAAAA
Noodistus	

Lähtekood	`default:`
	`  viisPrimary = CN(q.P1, 1) + CN(q.P2, 1) + CN(q.P3, 1) + CN(q.P4, 1) + CN(q.P5, 1) + CN(q.P6, 1) + CN(q.P7, 1) + CN(q.P8, 1) + CN(q.P9, 1) + CN(q.P10, 1) + CN(q.P11, 1) + CN(q.P12, 1) + CN(q.P13, 1) + CN(q.P14, 1) + CN(q.P15, 1) + CN(q.P16, 1) + "; manual"`
	`  viisAlterna = CN(q.P1, 2) + CN(q.P2, 2) + CN(q.P3, 2) + CN(q.P4, 2) + CN(q.P5, 2) + CN(q.P6, 2) + CN(q.P7, 2) + CN(q.P8, 2) + CN(q.P9, 2) + CN(q.P10, 2) + CN(q.P11, 2) + CN(q.P12, 2) + CN(q.P13, 2) + CN(q.P14, 2) + CN(q.P15, 2) + CN(q.P16, 2) + "; manual"`

Iga teisenduse lõpus võrreldakse *viisPrimary* ja *viisAlterna* väärtuseid ning kui need erinevad, siis kirjutatakse mõlemad kirjed faili. Kui need on samad, siis kirjutatakse faili ainult *viisPrimary*. Peale seda kontrollitakse kas faili kirjutamisel leiti viga ning kui ei leitud, siis jätkatakse järgmise tsükli kirjega kuni kirjed saavad otsa.

6.5.2 Funktsioon CN(noot string, index int)string



Joonis 98 Funktsioon CN(noot string, index int)string tegevusdiagramm.

Funktsioon CN eesmärk on noodi teisendamine (Joonis 98). Nimi CN valiti põhjusel, et see oleks võimalikult lühike. Autorid soovisid, et eelmises jaotises kirjeldatud rütmitüübi

teisendamine oleks lihtsamini loetav ja mahuks ühele ekraanile. *Index* parameeter on kasutusel selleks, et aru saada kas tegemist on *viisPrimary* või *viisAlterna* rütmitüübi reaga.

CN funktsioon saab sisendiks string tüüpi argumendi (väärtuse). Esiteks kontrollitakse kas string on tühi. Juhul kui on, siis tagastatakse <empty> väärtus ja lõpetatakse funktsiooni töö. < ja > märkide lisamise eesmärk oli lihtsustada hilisemat otsingut, selleks, et leida hõlpsasti üles teisendamata jäänud elementidega noodistused.

Juhul kui sisend ei ole tühi, ehk selle pikkus on suurem kui 0, siis vaadatakse kas tegemist on variatsioonidega kirjega. Selleks vaadatakse juhul kui index=1 stringi viimast elementi. Juhul kui see on “]”, siis on teada, et tegemist on variatsioonidega kirjega ning kasutades funktsiooni *strings.Split* poolitatakse string “[“ märgi juurest ning võetakse esimene element ning see omistatakse muutujale *nootClean*. Kui index=2, siis otsitakse samuti kas noot lõppeb märgiga “]” ning kui jah, siis võetakse *strings.Split* funktsiooni abil teine element ning eemaldatakse viimane element milleks oleks]. Juhul kui] märki ei leita, siis omistatakse kogu sisendiks olnud string muutujale *nootClean* (Joonis 99).

```
//Alternatiivide puhul võtame välja esimese ja teise alternatiivi
if index == 1 && string(noot[len(noot)-1:]) == "]" {
  nootClean = (strings.Split(noot, "["))[0]
} else if index == 2 && string(noot[len(noot)-1:]) == "]" {
  tmpval = (strings.Split(noot, "["))[1]
  nootClean = string(tmpval[:len(tmpval)-1])
  variatsioon = true
} else {
  nootClean = noot
}
```

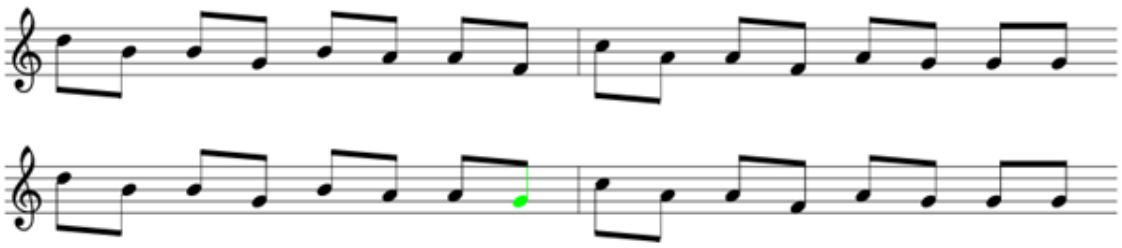
Joonis 99 Alternatiivide töötlemine.

Järgmiseks vaatame kas tegemist on väljaga kuhu on pandud kaks kiiremat nooti. Juhul kui on kiiremad noodid, siis need on eraldatud “/” märgiga. Kui selline märk leitakse *nootClean* muutuja väärtusest, siis eraldatakse *strings.Split* funktsiooniga need teineteisest ning iga noodi puhul kutsutakse välja funktsioon *ConvertNoot*. Juhul kui leitakse nii algus kui lõpu sulud “(“ “)”, siis asendatakse sulud looksulgudega ning sulgude sees ja sulgude järel olevale väärtusele kutsutakse välja funktsioon *ConvertNoot* (Joonis 100).

```
//Ehitame vastuse vastavalt sellele mis elemendid meile sattusid
if strings.Contains(nootClean, "/") == true {
  //Kui on kiirem noot siis teeme nii
  vastus = ConvertNoot(strings.Split(nootClean, "/")[0]) + "/" +
ConvertNoot(strings.Split(nootClean, "/")[1]) + "/"
} else if strings.Contains(nootClean, "(") == true &&
strings.Contains(nootClean, ")") == true {
  //Kui on eellöök siis teeme nii
  vastus = "{" + ConvertNoot(strings.Split(strings.Split(nootClean, "(")[1],
")")[0]) + "}" + ConvertNoot(strings.Split(nootClean, ")")[1])
} else {
  vastus = ConvertNoot(nootClean)
}
```

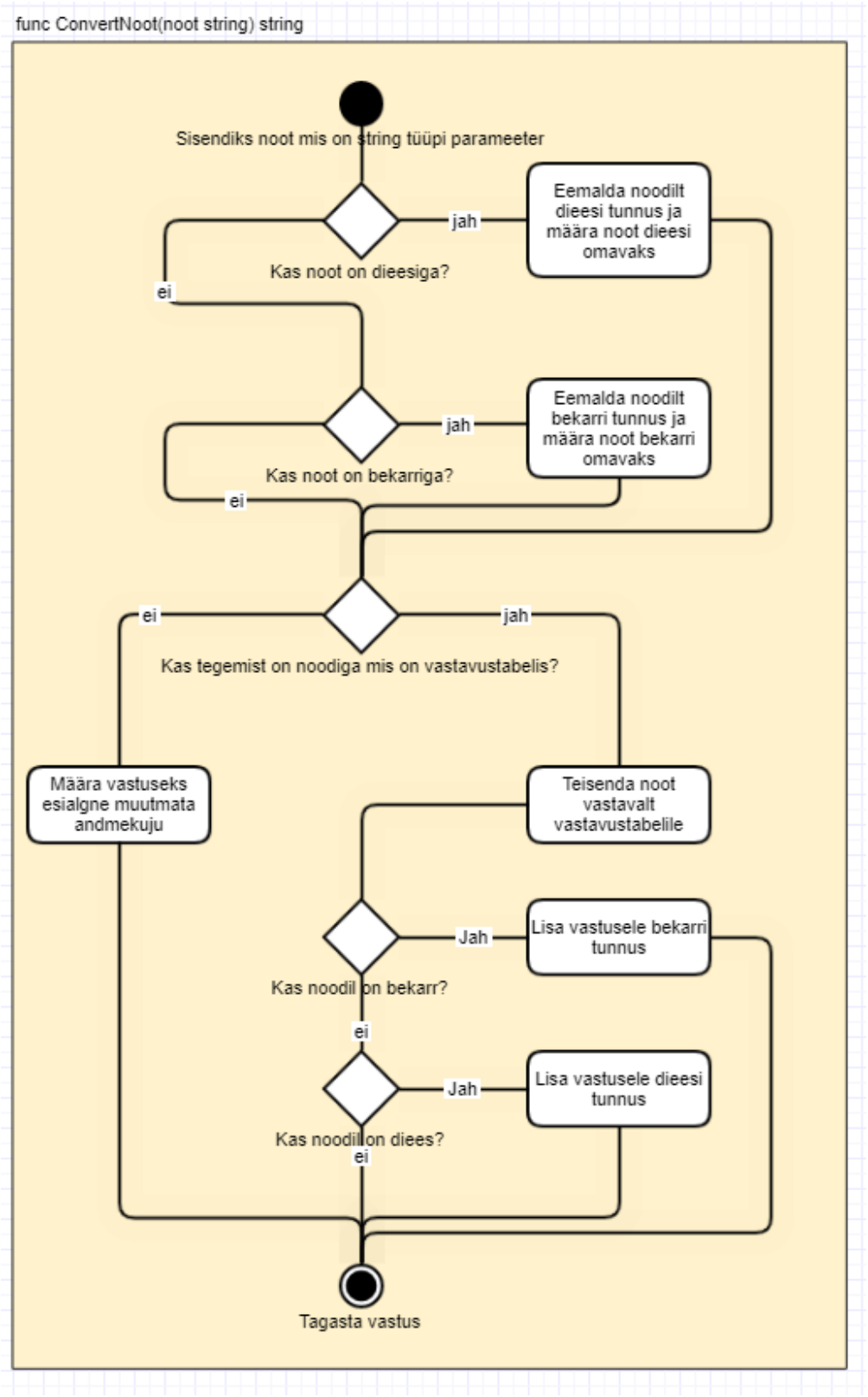
Joonis 100 CN funktsiooni vastuse koostamine.

Viimase asjana kontrollime, kas tegemist oli variatsiooniga ning kui oli, siis lisame vastuse ette “!mark!” stringi, mille tulemusena noodistust visualiseerides värvitakse variatsiooniks olnud noot teist värvi.(Joonis 101)



Joonis 101 Noodi värvimine teist värvi.

6.5.3 Func ConvertNoot(noot string) string



Joonis 102 Func ConvertNoot(noot string) string.

Funktsiooni *ConvertNoot* eesmärk on teisendada noot ABC standardile vastavale kujule. CN funktsiooni läbimise tulemusena on tekkinud stringitüüpi väärtus, kust on eemaldatud kõik üleliigne ning alles on jäänud vaid teisendamist vajav string (Joonis 102).

Esimese asjana kontrollitakse, kas tegemist on dieesiga. Selleks otsitakse sisendväärtusest "#" märki. Juhul kui see leitakse, siis see eemaldatakse väärtusest ning määratakse diees parameetri väärtuseks TRUE.

Järgmisena kontrollitakse, kas tegemist on bekaariga, otsides stringist väärtust "-bekaar". Juhul kui see leitakse, siis see eemaldatakse ning määratakse bekaar parameetri väärtuseks TRUE.

Koodis ei ole arvestatud võimalusega, et nii bekaar kui diees võiksid korraga eksisteerida, kuna bekaar on dieesi tühistus ning selline variant ei saa eksisteerida.

Järgmisena käivitatakse juba puhastatud noodil *switch* funktsiooni abil teisendus. Selleks on loodud vastavustabel nagu on kirjeldatud tabelis (Tabel 8). Lisaks vastavustabelis olevatele kirjetele on teisenduses kasutusel ka bemolli kirjed ehk tähis b. Kuna tähis b vastab ka noodile B, siis lisati kõik lähteandmestikus olevad kombinatsioonid teisenduse tabeli lõppu (Joonis 103).

```
case "ba":
  nootOutput = "_A"
case "bh":
  nootOutput = "_B"
case "bd":
  nootOutput = "_D"
case "be":
  nootOutput = "_E"
```

Joonis 103 bemolliga kirjete teisendamine.

Juhul kui eelnevalt tuvastati, et tegemist on dieesi või bekaariga, siis lisatakse vastav tähis noodile (Joonis 104).

```
if diees == true {
  return "^" + nootOutput
}
if bekaar == true {
  return "=" + nootOutput
}
```

Joonis 104 diees ja bekaar tähiste määramine.

Kui eelneva teisenduse väljundi pikkus on suurem kui null, siis tagastatakse teisenduse tulemus. Muul juhul tagastatakse *noot* parameetri väärtus ehk sisendiks olnud väärtus “<” ja “>” märkide vahele panduna, selleks, et selliseid ebaõnnestunud teisendusi oleks hilisemas tulemuses võimalik otsida ning parandusi käsitsi teha.

7 Tulemuste valideerimine ja nendest õppimine

Tulemusi valideeriti jooksvalt käesoleva töö käigus iganädalaste koosolekute raames. Lisaks küsiti enne töö esitamist EKM-ilt hinnangut tehtud tööle ning sooritati põhjalik vastuvõtutestimine.

7.1 EKM-i hinnang tehtud tööle

„Arvamus Eesti Rahvaviiside Andmebaasi arendamise kohta

Üliõpilased Martin Rajur, Jaan Susi ja Annett Lymar arendasid edasi Eesti Rahvaluule Arhiivi veebipõhist Eesti Rahvaviiside Andmebaasi. Selle eelmine järk oli valminud koostöös TalTechiga 2020. aastal, mil loodi netipõhine infosüsteem rahvaviiside haldamiseks, keskendudes viiside metaandmetele.

Praeguse rühma, - kes alustas tööd andmebaasiga 2020. aasta sügisel ja jõuab nüüd kaitsmiseni, - eesmärgiks oli luua kasutajaliides, lisada andmebaasi kodeeritud viisid koos kodeeringu andmetega, luua multimuutmise võimalus ning välisviidete funktsionaalsus, mis võimaldab lisada seoseid teiste andmebaasidega. Need on küllalt keerukad ülesanded, arvestades andmebaasi struktuuri ja ulatust. Kõik ülesanded on täidetud ja andmebaasile on loodud eelpool nimetatud funktsioonid.

Tahaksin tunnustada ja tänada kolmikut, kes töötas huvi ja pühendumusega, süvenedes ka sisuliselt andmebaasis hallatava materjali – rahvaviiside ja nende noodistuste – iseärasustesse. Eriti tõstan esile, et ülesandele läheneti loovalt ja viiside kodeeringut kohandati sel moel, et viisid muutusid (vabavara ABC editor vahendusel) ka ekraanil automaatselt kuvatavaks ja kuulatavaks. Mõningaid pisiasju saaks edaspidi viimistleda, kuid see on suure ülesande puhul tavaline ega vähenda kuidagi tehtud töö väärtust.

Märkusena lisan, et meie algsetest lootustest andmebaasi osas ei valminud veel otsingusüsteem, sest üliõpilane, kes asus seda looma, lükkas vastava töö kaitsmise ja ülesande täitmise sügisesse. See ei puuduta kuidagi praeguse rühma tööd, ainult seletab, miks otsinguga seotud osa andmebaasis on sisustamata.

Kui see pole ka õige koht – siiski soovin eraldi tänada möödunud aastal Infotehnoloogia Teaduskonna lõpetanud Kristi Seemenit, kes jäi nõuandjana koos meiega tööle andmebaasi juurde, ja aitas uuel rühmal varasema tööga tutvuda ja andmeid migreerida vanast andmebaasist uude. Aitäh, Kristi.

Taive Särg

Eesti Kirjandusmuuseumi Eesti Rahvaluule Arhiivi vanemteadur

14.05.2021“

7.2 Integratsioonitestid

Autorid viisid perioodiliselt läbi integratsiooniteste. Nende käigus tehti lähtekoodi külmutamine ning võeti eraldi aeg ainult testimiseks. Testimine toimus ettemääratletud testjuhtude alusel, ning viidi läbi autorite poolt samaaegselt, et veenduda tulemuste korrektsuses. Autorid nõustusid omavahel, et integratsioonitestide tulemuste kirja panemine ei ole vajalik, mistõttu tulemused sünkroniseeriti ning seejärel alustati järgmist iteratsiooni. Testimisel kasutati erinevaid veebilehitsejaid, et veenduda rakenduse platvormiüleses toimivuses.

Järgnevas tabelis on välja toodud integratsioonitestide läbi viimiseks kasutatud testjuhud.

Tabel 14Integratsioonitestide testjuhud.

Moodul	**Testjuht**	**Oodatav tulemus**
Logimine	Sisestasin kasutajatunnuse ja parooli	Sisselogimine õnnestus, kasutajale kuvatakse koduleht
	Vajutasin „Unustasin parooli“	Kasutajale kuvatakse parooli lähtestamise vaade
	Sisestasin kasutajatunnuse ja vale parooli	Kasutajale kuvatakse sõnum: „Vigane e-mail/parool“
Kasutajad	Vajutasin kasutajate vaates „deaktiveeri“	Vastav kasutaja deaktiveeritakse

Moodul	**Testjuht**	**Oodatav tulemus**
	Vajutasin kasutajate vaates nupule „vaata“	Kuvatakse kasutaja detailne info
	Vajutasin nupule „muuda“. Muutsin kasutajat. Olemas oli e-posti aadress, lisasin ees- ja perenime. Vajutasin „salvesta“ ning „vaata“	Tehtud muudatused salvestati ning peale vaata nupu vajutamist kuvati kasutaja leht kus oli näha tehtud muudatusi.
	Kasutaja vaatest vajutasin „Loo uus“ Täitsin väljad. Salvestasin	Tekkis uus kasutaja.
Otsing	Lisasin otsingusse viite „EKRK“	Avaneb otsingu tulemuste vaade, kus on näha selle viitega tulemused
	Lisasin otsingusse viite EKRK I 18, 310 (12) ja vajutasin „vaata“	Viisi detailvaade avanes
	Lisasin otsingusse viite EKRK I 18, 310 (12) ja vajutasin „vaata“ ja seejärel „muuda“	Avanes viisi detailvaade muutmise režiimis
	Otsingu vaatest markeerisin viisi viitega A3476. Vajutasin „kustuta“	Viis kustutati infosüsteemist.
Klassifikaatorid	Klassifikaatorite vaatest vajutasin „Loo uus“ ja lisasin uue aktiivse klassifikaatori	Klassifikaatorite nimekirja tekkis uus klassifikaator vastavalt sisestatud informatsioonile
	Klassifikaatori vaatest valisin klassifikaatori muutmise	Klassifikaatori muutused salvestati infosüsteemi

Moodul	**Testjuht**	**Oodatav tulemus**
	Klassifikaatori vaates valisin „Muuda“ ja võtsin kirje juurest ära märgistuse „on aktiivne“	Klassifikaator ei ole enam aktiivne ning ei ole kasutatav
	Klassifikaatori vaatest valisin klassifikaatori kustutamise	Klassifikaator eemaldatakse süsteemist juhul kui see pole ühegi viisi juures kasutusel.
	Klassifikaatori detailvaatest vajutasin nupule „kustuta“	Klassifikaator eemaldatakse süsteemist juhul kui see pole ühegi viisi juures kasutusel.
Isikud	Isiku vaatest uue isiku lisamine	Lisatakse uus isik
	Isiku detailvaate vaatamine	Kuvatakse isiku detailvaade koos kõikide väljadega
	Isiku detailvaatest kustutamine	Isik kustutatakse juhul kui see isik ei ole ühegi viisi juures kasutusel
	Isiku vaatest kustutamine	Isik kustutatakse juhul kui see isik ei ole ühegi viisi juures kasutusel
Tõlked	Olen otsingu vaates ja muudan EST => ENG	Kogu infosüsteemi tarkvara kasutajaliidese keel muutub ingliskeelseks. Kõik väljanimed on keelt muutnud
Viisid	Viisi lisamisel kuvatakse noodistust vastavalt selle lisamisele	Viisi sisestamisel on abc kodeeringut sisestades näha igat lisatavat nooti selle lisamise hetkel. Lisaparameetrite

Moodul	Testjuht	Oodatav tulemus
		väärtuste seadistamisel muutub noodikuva automaatselt.
	Viisi salvestamine toimib	Viisi mistahes osa muutes kõik muudatused salvestatakse ning on näha viisi vaates
	Vajutades „eksport“ nuppu genereeritakse musicXML fail, mis esitatakse kasutajale allalaadimiseks	Vajutades „eksport“ nuppu peab toimuma abc noodistuse teisendamine musicXML kujul, mida peab olema võimalik kasutaja arvutisse laadida
Logimine	Muudatuste kohta peab tekkima logi	Viisi muudatuste kohta peab tekkima logi, mida olema võimalik näha viisi andmete juures
Autoriseerimine	Ilma sisse logimata ei tohi olla võimalik näha kasutajate infot ega kasutada muuda/kustuta/aktiveeri funktsioone	Ilma sisse logimata peavad olema peidetud kõik muutmisega seotud funktsioonid

7.3 Tulevikus kasutatavad tulemused

Antud töö tulemusena tekkis tarkvara, mis sisaldab võimekust muusikaliste teoste üles märkimiseks koos seotud viidete ja muu isikustatud/isikustamata andmetega. Loodud lahenduse kasutajaskonda on võimalik laiendada, lubades sellele ligipääsu teistele analoogset lahendust vajavatele kultuuri- ja haridusasutustele. Lahendus võimaldab sisestada eripikkusega noodistust koos sõnadega. Seetõttu pole piiratud muusika liik, mida selles keskkonnas on võimalik salvestada. Samuti on lahenduse disainimisel arvestatud võimalusega lisaks infosüsteemi Kivike viidetele võimaldada teiste infosüsteemide viiteid. Kasutusele võetud ABC kodeering on standard, mille põhimõtted ja reeglid on avalikult kättesaadavad ning mida kasutatakse nii erinevates rakendustes kui ka teadustöödes. Seega on autorid veendunud, et kasutusele võetud standard on selline, mida kasutades on võimalik teiste teadusasutustega andmeid vahetada ilma vajaduseta andmeid teisendada.

Töö käigus on EKM-i töötajad ja üks praktikant alustanud kodeerimist ABC noteeringus selleks, et teha MS Access andmebaasis esinevate puudulike kirjete kodeerimine käsitsi. Autoritel on hea meel märkida, et praeguseks on lisaks automaatselt teisendatud kodeeringutele tekkinud ka märkimisväärne hulk käsitsi kodeeritud teoseid, seega peavad autorid valikut õnnestunuks. Autorite arvates on ABC kodeeringu näol tegemist muusika baasteadmisi omava inimese jaoks lihtsa võimalusega noodistamiseks ning olemasolev dokumentatsioon on piisav selleks, et muusikalise haridusega inimene on võimeline vähese vaevaga kodeerimist alustama.

Viiside kodeeritud ja struktureeritud andmebaasi saab võtta aluseks tuleviku teadusuuringutes. Iga viisi puhul on olemas suurel hulgal erinevat metaandmestikku ja noodistused, mida on võimalik erinevat liiki uuringutes kasutada. Tänu lihtsamale andmete kättesaadavusele võib loota, et mugav kasutajaliides ning tulevikus tekkiv sarnaste viiside otsingu lahendus kutsuvad inimesi rohkem huvi tundma eesti rahvaviiside vastu.

7.4 Mida sellest tööst veel õppida?

Töö kirjutamisel kasutati Onedrive Word dokumenti, et kõik meeskonna liikmed saaksid sama dokumenti üheaegselt redigeerida. Selline lähenemine osutus väga kasulikuks, kuid kahjuks oli sellel lähenemisel ka mitmeid puuduseid.

Eelisena toome välja integratsiooni MS Word töölauarakendusega, mille tõttu oli võimalik pilves asuvat dokumenti avada töölaual samaaegselt tarkvara täisfunktsionaalsust kasutades. Muudatusi sünkroniseeriti pidevalt pilve ning dokumendist loodi mitmeid automaatseid versioone, mida probleemide ilmnemisel oli võimalik viitena dokumendi eelmisest versioonist kasutada.

Puuduseks antud lahenduse puhul tooksime esiteks välja, et kuigi on võimalik samaaegselt dokumenti muuta, tekkisid meil üsna tihti sünkroniseerimise konfliktid, mida oli vaja käsitsi lahendada.

Teise probleemina toome välja, et kui dokumendis kasutati suures hulgas pilte, tabeleid ja välisviiteid, siis perioodiliselt tehti dokumendis viidete uuendamist ning selle toimumise ajal hangus dokument kõikide kasutajate jaoks. Kui viidet või pealkirja lisati olemasolevate kirjete vahele, mille tulemusena muutus numeratsioon, siis esines sellist katkestust liiga sageli ja see hakkas tööd segama.

Kolmanda probleemina mis ilmnes just MS Word töölaua tarkvara ja veebiversiooni ristkasutusega tulenes funktsioonist „Track changes“ (jälgi muutuseid). Seda kasutades tekkis korduvalt olukordi, kus teadmata põhjustel muutusid stiilid ning selle tulemusel ka vormingud. See tähendas omakorda seda, et tekkis vajadus käsitsi stiile parandada või üle kirjutada. Lisaks märkasid autorid, et „Track changes“ funktsioon toimis perioodiliselt kus ühel kasutajal seda rakendati ja teisel mitte, mistõttu oli raske aru saada, mis režiimis dokument on.

Neljandaks juhtus paaril korral nii, et osa tekstist kadus ära või pealtnäha suvalistesse kohtadesse tekkisid juurde üksikud tähed. Ühel korral oli kaks lehte teksti liikunud tabeli välja sisse, mille tulemusena oli suurem osa sellest informatsioonist hävinenud kuid õnneks tänu automaatsele versioneerimisele, oli eelnev seis võimalik taastada.

Kokkuvõtvalt loeme antud lahendust ühistöö loomisel ikkagi väärtuslikuks ning olenemata puudustest andis sellise lahenduse kasutamine töö kirjutamisel lisandväärtust

ning kiirendas ühistöö protsessi. Samuti puudus sellise lähenemise puhul täiendav vajadus dokumendi erinevaid versioone hallata ning neid omavahel sünkroniseerida.

8 Edasised uurimisvõimalused seoses tehtud tööga

Seoses loodud infosüsteemi ja kodeerimisstandardi kasutuselevõtuga teevad töö autorid järgmised ettepanekud tulevasteks uurimusteks.

8.1 I. Rüütli töö analüüsi uus iteratsioon

Ingrid Rüütel koostas töö "Eesti rahvaviiside tüpoloogia (1.01.2004−31.12.2007)" mille raames loodi Eesti rahvaviiside tüpoloogiline süsteem. Selleks loodi originaalne arvutimeetod, mis võimaldab diferentseerida meloodiatüüpe, selgitada nende vahelisi seoseid, modelleerida üksikute meloodiatüüpide struktuuri, uurida muutlikust ning koostada generatiivseid grammatikaid. [68] Nimetatud töös on viitatud ka kaherealiste refräänita viiside andmebaasile, mis käesoleva töö käigus teisendati ABC standardile vastavale noodistusele. Käesoleva töö ja eelneva bakalaureusetöö tulemusena on selles andmebaasis olevad andmed kontrollitud ning ühtlustatud. Juba loodud algoritmide ja metoodikate abil on võimalik analüüsi uuesti teha, kaasates ka aja jooksul lisanduvaid noodistusi.

8.2 Häälegeneraatori abil laulude esitamine

EKM-iga iganädalaste koosolekute pidamise käigus arutati ka võimalust lisaks noodistuse alusel heli genereerimisele ka laulude esituse genereerimist. Praegu realiseeritud lahendus sisaldab ainult noodistusi ilma sõnadeta, kuid süsteemi realisatsioon võimaldab ka lisada laulusõnu. Selle tulemusena tekib aja jooksul kirjeid, mille puhul on olemas nii noodistus kui sõnad. Selle info põhjal on teoorias võimalik genereerida ka laulu esitus. Seega võib üks antud infosüsteemi väljunditeks tulevikus olla ka rahvaviiside ja laulude digitaalse esituse põhjal rahvalaulu digitaalne genereerimine ja taasesituse võimaldamine. M. Blaauw ja J. Bonada esitasid artikli “A Neural Parametric Singing Synthesizer Modeling Timbre and Expression from Natural Songs” [69], milles nad kirjeldasid oma WaveNet sügaval närvivõrgul baseeruvat mudelit laulu sünteesimiseks.

8.3 Masinõppe abil uute viiside loomine

Loodud andmebaasi noodistustel põhinev üks võimalikke tuleviku uurimissuundi võiks olla võimalus luua masinõppe abil uusi viise. Selle võimaluse loob käesoleva töö tulemusena loodud kodeeritud noodistuse kogum, mida saaks kasutada masinõppe mudeli treenimiseks. O.F. Jansen kirjeldab oma bakalaureusetöös "Training Sequence Generative Adversarial Nets to Compose Music in the abc-notation." [70], kuidas luua ABC kodeeringus kodeeritud viiside põhjal luua uusi viise. Antud töö kasutab viiside kodeerimiseks ABC kodeeringut, mis on ka kodeering, millele viidi üle EKM-i viisid. Seetõttu võiks antud töö rakendamine EKM-i andmestiku peal olla lihtsam. Täiendavalt on B. L. Sturm, J. F. Santos ja I. Korshunova avaldanud artikli "Folk music style modelling by recurrent neural networks with long short term memory units" [71], mis keskendub just rahvamuusikale. Nagu ka eelnevas töös on ka siin kasutatud noodistuse loomiseks ABC kodeeringut.

8.4 Sarnaste viiside otsimine

Selleks, et võimaldada viiside otsimist viisi struktuuri alusel, on vaja rakendada mingit meetodit, mille abil võrrelda otsingu aluseks olevat noodistust andmebaasis oleva noodistusega. Sellist tüüpi otsingu tegemisel on tähtis, et otsing oskaks leida ka sarnaseid viise ning oskaks arvestada noodistuse struktuuri ja põhimõtetega. Antud teemal on C. Walshaw, kes on ühtlasi ABC kodeeringu autor, kirjutanud artikli "A Visual Exploration of Melodic Relationships within Traditional Music Collections" [72], milles vaadeldakse meloodiliselt sarnaste viiside leidmiseks meloodilisi sarnasusi. Sellise lähenemise puhul oleks võimalik otsingu puhul kasutada meloodilist sarnasust selleks, et lisaks täpsetele vastetele kuvada ka sarnaseid vasteid. Lisaks sellele on sama autor kirjutanud ka artikli "A statistical analysis of the ABC music notation corpus: exploring duplication" [73], milles vaadeldakse korduvaid kodeeringuid ning sarnasuste leidmist. Üks võimalik näidis sarnasest realisatsioonist oleks *Folk tune finder* veebirakendus, mis võimaldab klaveri abil noodistust sisestada ning selle põhjal viise otsida. Antud veebirakendus kasutab andmetena ABC notatsioonis kodeeritud andmeid tänu millele võib olla sarnasuste otsimine lihtsam. [74]

8.5 Helifailide põhjal noodistuse loomine

EKM-i ERA arhiivis on nii käsikirju kui helisalvestisi. Selleks, et kiirendada arhivaalide teisendamist kodeeritud noodistuseks, võiks uurida helisalvestiste automaatse kodeerimise võimalust. Sel teemal on kasutajauuringu pealkirjaga "AUTOMATIC MUSIC TRANSCRIPTION AND ETHNOMUSICOLOGY: A USER STUDY" [75] [76] teinud A. Holzapfel ja E. Benetos. Antud uuringus hindavad nad automaatse muusika kodeerija (*Automatic Music Transcription*) kasulikkust rahvamuusika kodeerimiseks.

8.6 Optilise märgituvastuse kasutamine automaatseks noodistuse loomiseks

EKM-i ERA arhiivis on suurem osa arhivaalides käsikirjad, mis on ka osaliselt digitaliseeritud ehk eksisteerivad pildi kujul. A. Othman, ja C. Direkoğlu kirjeldavad oma teadusartiklis võimalustest rakendada pildi põhjal noodistuse loomiseks optilist märkide tuvastamise süsteemi koos masinõppega. [77] Sarnase või sama metoodika rakendamisel võiks olla võimalik arhiivi kodeerimist märkimisväärselt kiirendada.

9 Kokkuvõte

Lõputöö eesmärgiks oli edasi arendada rahvaviiside infosüsteemi Eesti Kirjandusmuuseumile. Uue süsteemi versiooni loomise aluseks võeti süsteemi esialgne versioon, mis oli bakalaureusetöö raames varasemalt üliõpilaste poolt loodud. Ülesandepüstitus koostati EKM-i poolt ja töö autorite eesmärk oli kõikide soovide realiseerimine.

Töö koostamisel lähtusid autorid disaini tegevusuuringu põhimõtetest, eesmärgiga luua EKM-i probleeme lahendav tehniline artefakt (töötav süsteem) ning seda järk-järgult kasutusele võtta. Meeskonnatöö korraldamiseks kasutati arendusmetoodikana Scrum raamistikku.

Töö tulemusena juurutati ühtne nootide kodeerimise standard, tänu millele saab EKM rahvaviise digitaliseerida ja ka noodistikku kuvada. See lahendus tekitas võimaluse viisi juurde tekitada helifaile, mida kasutajad saavad otse veebilehitsejas kuulata. Standardi valimiseks kasutati Analüütiliste Hierarhiate Meetodit (Saaty meetodit) ja juurutatavaks standardiks valiti ABC notatsioon.

Töö tulemusena loodi teisendusutiliit (https://github.com/nitramra/EKM_converter) olemasolevas kodeeringus noodistuste teisendamiseks valitud standardile vastavaks noodistuse esituseks. Olemasolevad noodistused kanti uues kodeeringus uude süsteemi versiooni üle. See tähendab, et rahvaviiside noodistuste kogumisel ja kirjapanemisel juba tehtud suur töö ei läinud kaotsi.

Töö tulemusena tekkis uus rahvaviiside infosüsteemi versioon. Selles on uuel platvormil (Node.js, Loopback 4 ja React) realiseeritud eelmises infosüsteemi tarkvara versioonis realiseeritud funktsionaalsus. Andmebaasiplatvorm (PostgreSQL) jäi samaks, kuid andmebaasi lisandus uusi tabeleid. Valminud lahendus võimaldab talletada kõik rahvaviisidega seotud andmed ühes kohas. Lisaks loodi võimalus muuta mitut viisi korraga, lisada viisidele kodeeritud noodistus, viidata arhivaalidele Kivikese repositooriumis, mängida ette helifaile ja logida viisidega toimunud andmemuudatusi. Viisifailide hoiustamise vajadus lahenes sellega, et ABC notatsiooni hoitakse andmebaasis. Lõpuni jäi realiseerimata võimalus valida infosüsteemi tarkvara kasutajaliidese keel, massmuutmise laiendatud realisatsioon ning viisi kodeeringu eksport

ja import. Otsingu funktsionaalsus realiseeriti minimaalsena, kuid süsteem ehitati üles sellisena, et sinna oleks võimalik tulevikus detailsemat otsingu funktsionaalselt lihtsalt lisada.

Töö arenduse käigus kasutatud testimiskeskkond on saadaval aadressil https://viisid.xmt.ee kuni 2021. aasta lõpuni. Töökeskkonna viide lisatakse peale andmete lõplikku valideerimist ning rakenduse juurutamist EKM-i andmebaaside nimekirja, mis asub aadressil https://www.kirmus.ee/et/info/andmebaasid.

10 Kasutatud kirjandus

[1] „ANDMEKAITSE JA INFOTURBE LEKSIKON,“ Cybernetica, [Võrgumaterjal]. Available: https://akit.cyber.ee/term/8673-ascii. [Kasutatud 21 05 2021].

[2] „Muusikateooria õpik,“ [Võrgumaterjal]. Available: http://mt.ema.edu.ee/. [Kasutatud 05 05 2021].

[3] „Kivike,“ Eesti Kirjandusmuuseum, [Võrgumaterjal]. Available: http://kivike.kirmus.ee/. [Kasutatud 16 05 2021].

[4] I. Bent, D. Hughes, R. Provine, R. Rastall, A. Kilmer, D. Szendrei, T. Payne, M. Bent ja G. Chew, „Notation. Grove Music Online.,“ 2001. [Võrgumaterjal]. Available: https://www-oxfordmusiconline-com.ezproxy.utlib.ut.ee/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000020114. [Kasutatud 08 05 2021].

[5] „REST - Wikipedia,“ [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/REST. [Kasutatud 16 05 2021].

[6] „Kultuuripoliitika põhialused aastani 2020,“ [Võrgumaterjal]. Available: https://www.riigiteataja.ee/aktilisa/3140/2201/4002/RKo_lisa.pdf. [Kasutatud 05 05 2021].

[7] M. Paroll, K. Avloi ja K. Seemen, „Rahvaviiside infosüsteemi tarkvara arendamine Eesti Kirjandusmuuseumile. Tallinna Tehnikaülikool. Bakalaureusetöö.,“ 2020..

[8] „EUROOPA PARLAMENDI JA NÕUKOGU MÄÄRUS (EL) 2016/679,,“ Euroopa Liidu Teataja, 27 04 2016. [Võrgumaterjal]. Available: https://eur-lex.europa.eu/legal-content/ET/TXT/HTML/?uri=CELEX:32016R0679. [Kasutatud 04 12 2020].

[9] „Isikuandmete kaitse seadus. Riigi Teataja.,“ [Võrgumaterjal]. Available: https://www.riigiteataja.ee/akt/IKS. [Kasutatud 04 12 2020].

[10] E. Kirjandusmuuseum, „Eesti Rahvaluule Arhiiv,“ [Võrgumaterjal]. Available: https://www.kirmus.ee/eesti-rahvaluule-arhiiv. [Kasutatud 09 05 2021].

[11] „Eesti Rahvaluule arhiiv,“ [Võrgumaterjal]. Available: http://folklore.ee/era/ava.htm. [Kasutatud 16 05 2021].

[12] T. Kannik, „Arhiivide kirjastustegevus Eestis: olukorra kaardistus. Tallinna Tehnikaülikool. Magistritöö.,“ 2017.

[13] K. Kulasalu, „Ropp ja riigivastane: rahvaluulekogude tsenseerimisest Eestis hilisstalinismi perioodil. Tartu Ülikool. Magistritöö,“ 2013.

[14] Rahvusarhiiv, „Kultuuripärandi digiteerimine,“ [Võrgumaterjal]. Available: https://www.ra.ee/projektid-ja-koostoo/kultuuriparandi-digiteerimine/. [Kasutatud 2021 05 2021].

[15] P. A. G. Permana, „Scrum Method Implementation in a Software,“ *International Journal of Advanced Computer Science and Applications,* kd. 6, 2015.

[16] L. Rising ja N. S. Janoff, „The Scrum Software Development Process for Small Teams,“ IEEE SOFTWARE, 2000.

[17] M. Sein, O. Henfridsson, , S. Purao, M. Rossi ja R. Lindgren, „Action design research. MIS Quart. 35,“ pp. 37-56, 2011.

[18] A. Haj-Bolouri, S. Purao, M. Rossi ja L. Bernhardsson, „Action Design Research as a Method-in-Use:“.

[19] T. Henriques ja H. ONeill, „ACTION DESIGN RESEARCH – Logical Data Model,“ 2020.

[20] „diagrams.net,“ [Võrgumaterjal]. Available: https://www.diagrams.net/. [Kasutatud 07 04 2021].

[21] „Gliffy,“ Perforce Software, Inc., [Võrgumaterjal]. Available: https://www.gliffy.com/. [Kasutatud 10 03 2021].

[22] T. L. Saaty, „Decision making with the analytic hierarchy process,“ Pittsburgh, Int. J. Services Sciences, 2008.

[23] „Web-HIPRE Global Decision Support,“ Helsinki University of Technology, [Võrgumaterjal]. Available: http://hipre.aalto.fi/. [Kasutatud 10 05 2021].

[24] „Visual Studio Code - Code Editing. Redefined,“ [Võrgumaterjal]. Available: https://code.visualstudio.com/. [Kasutatud 16 05 2021].

[25] C. Seeger, „Prescriptive and Descriptive Music-Writing,“ *The Musical Quarterly,* kd. 44, pp. 184-195, 1958.

[26] M. Seiffert, Sammelbände der internationalen Musikgesellschaft 11, 1909-10.

[27] T. Homburger, „Notenlesen verständlich erklärt,“ [Võrgumaterjal]. Available: https://www.bonedo.de/artikel/einzelansicht/notenlesen-verstaendlich-erklaert.html. [Kasutatud 03 05 2021].

[28] S. Fischer, „Tempo ja retooriline ajastamine 17.–18. sajandiinstrumentaalmuusika esituses. Eesti Muusika- ja Teatriakadeemia.,“ 2020.

[29] I. Rüütel, „Eesti Regivärsiliste Rahvalaulude Muusikaline Tüpoloogia,“ %1 *Võim & Kultuur 2*, M. Kõiva, Toim., Tartu, EKM Teaduskirjastus, 2006.

[30] „finale.,“ [Võrgumaterjal]. Available: https://www.finalemusic.com/. [Kasutatud 02 05 2021].

[31] L. Leis, „Diginootide meloodiaanalüsaator. Tartu Ülikool. Bakalaureusetöö.,“ 2020.

[32] „W3C,“ [Võrgumaterjal]. Available: https://www.w3.org/community/about/process/final/. [Kasutatud 10 05 2021].

[33] „MusicXML 3.1 For Developers,“ [Võrgumaterjal]. Available: https://www.musicxml.com/for-developers/. [Kasutatud 16 05 2021].

[34] „MIDI,“ [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/MIDI. [Kasutatud 16 05 2021].

[35] „Web Archive: MIDI AboutMMA,“ [Võrgumaterjal]. Available: https://web.archive.org/web/20160116011459/http://www.midi.org/aboutus/aboutmma.php. [Kasutatud 16 05 2021].

[36] „about abc notation,“ [Võrgumaterjal]. Available: http://abcnotation.com/about. [Kasutatud 16 05 2021].

[37] „ABC notation,“ [Võrgumaterjal]. Available: https://abcnotation.com/. [Kasutatud 16 05 2021].

[38] „Music notation info – ABC,“ [Võrgumaterjal]. Available: http://www.music-notation.info/en/formats/abc.html. [Kasutatud 03 05 2021].

[39] CRGRAPH, „Analytischer Hierarchieprozess,“ 2019. [Võrgumaterjal]. Available: http://www.versuchsmethoden.de/AHP.pdf.

[40] I. Basak ja T. Saaty, „Group decision making using the analytic hierarchy process,“ *Mathematical and Computer Modelling,* kd. 17, nr 4-5, pp. 101-109, 1993.

[41] L. Puu, „Sobiva andmeaida lahenduse väljapakkumine ettevõttele Hanza Mechanichs Tartu. Tallinna Tehnikaülikool. Magistritöö.,“ 2017.

[42] „Create and use strong passwords,“ Microsoft, [Võrgumaterjal]. Available: https://support.microsoft.com/en-us/windows/create-and-use-strong-passwords-c5cebb49-8c53-4f5e-2bc4-fe357ca048eb. [Kasutatud 06 05 2021].

[43] „ISKE kataloogid/7 Kataloog M/M2/M 2.11 - ISKE Portaal - Versioon 8_06,“ [Võrgumaterjal]. Available: https://iske.ria.ee/8_06/ISKE_kataloogid/7_Kataloog_M/M2/M_2.11. [Kasutatud 16 05 2021].

[44] „EESTI KIRJANDUSMUUSEUMI ARENGUKAVA AASTATEKS 2019–2022,“ [Võrgumaterjal]. Available: https://www.kirmus.ee/sites/default/files/2019-11/EKM_arengukava_IT-lisaga_2019-2022.pdf. [Kasutatud 16 05 2021].

[45] „Infosüsteemide turvameetmete süsteem ISKE,“ [Võrgumaterjal]. Available: https://www.ria.ee/et/kuberturvalisus/infosusteemide-turvameetmete-susteem-iske.html. [Kasutatud 16 05 2021].

[46] K. Viik, „Mittefunktsionaalsete nõuete määratlemine turvalise tarkvaraarenduse hankimiseks Eesti avalikus sektoris. Tallinna Ülikool. Magistritöö.,“ 2017.

[47] „ISKE Portaal,“ Riigi Infosüsteemi Amet, [Võrgumaterjal]. Available: https://iske.ria.ee/8_00//ISKE_kataloogid/5_Kataloog_B/B5/B_5.21. [Kasutatud 10 05 2021].

[48] „Isikuandmete kaitse seadus,“ [Võrgumaterjal]. Available: https://www.riigiteataja.ee/akt/104012019011. [Kasutatud 16 05 2021].

[49] AS PricewaterhouseCoopers Advisors, „Pikaajalise säilitamise arhitektuuri ärianalüüs Lõpparuanne,“ 2018.

[50] Kultuuriministeerium, „Kultuuripärandi digiteerimine 2018-2023" tegevuskava,“ 2018.

[51] „Autoriõiguse seadus,“ [Võrgumaterjal]. Available: https://www.riigiteataja.ee/akt/128122011005. [Kasutatud 05 05 2021].

[52] „The abc music standard 2.1 (Dec 2011),“ [Võrgumaterjal]. Available: http://abcnotation.com/wiki/abc:standard:v2.1. [Kasutatud 16 05 2021].

[53] „This index lists the different MusicXML elements, attributes, and entities.,“ [Võrgumaterjal]. Available: https://www.musicxml.com/for-developers/alphabetical-index/. [Kasutatud 16 05 2021].

[54] „Guide to EasyABC,“ [Võrgumaterjal]. Available: http://easyabc.sourceforge.net/. [Kasutatud 16 05 2021].

[55] „Free music composition and notation software | MuseScore,“ [Võrgumaterjal]. Available: https://musescore.org/en. [Kasutatud 16 05 2021].

[56] „GNU General Public License | MuseScore,“ [Võrgumaterjal]. Available: https://musescore.org/en/about/gnu-general-public-license. [Kasutatud 16 05 2021].

[57] „ABC Import | MuseScore,“ [Võrgumaterjal]. Available: https://musescore.org/en/project/abc-import. [Kasutatud 16 05 2021].

[58] „2020 Developer Survey,“ [Võrgumaterjal]. Available: https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-web-frameworks-loved2. [Kasutatud 03 03 2021].

[59] „Comparison with Other Frameworks,“ Vue.js, [Võrgumaterjal]. Available: https://vuejs.org/v2/guide/comparison.html. [Kasutatud 03 03 2021].

[60] „ISKE kataloogid/7 Kataloog M/M5/M 5.169 - ISKE Portaal - Versioon 8_00,“ [Võrgumaterjal]. Available: https://iske.ria.ee/8_00//ISKE_kataloogid/7_Kataloog_M/M5/M_5.169. [Kasutatud 16 05 2021].

[61] „LoopBack 4,“ [Võrgumaterjal]. Available: https://loopback.io/doc/en/lb4/Concepts.html. [Kasutatud 03 04 2021].

[62] „The MIT license,“ [Võrgumaterjal]. Available: https://opensource.org/licenses/MIT. [Kasutatud 16 05 2021].

[63] „Can I use this in my own site?,“ [Võrgumaterjal]. Available: https://www.abcjs.net/#can-i. [Kasutatud 16 05 2021].

[64] „@sourceloop/audit-log - npm,“ [Võrgumaterjal]. Available: https://www.npmjs.com/package/@sourceloop/audit-log. [Kasutatud 16 05 2021].

[65] „xml2abc,“ [Võrgumaterjal]. Available: https://wim.vree.org/svgParse/xml2abc.html. [Kasutatud 16 05 2021].

[66] „abc2xml,“ [Võrgumaterjal]. Available: https://wim.vree.org/svgParse/abc2xml.html. [Kasutatud 16 05 2021].

[67] „The Go Programming language,“ [Võrgumaterjal]. Available: https://golang.org/. [Kasutatud 16 05 2021].

[68] I. Rüütel, „"Riiklik programm: Eesti keel ja rahvuslik mälu (EKRM)" projekt EKRM04-35,“ Eesti Teadusinfosüsteem, [Võrgumaterjal]. Available: https://www.etis.ee/Portal/Projects/Display/13a5fc92-3a6d-4761-a4f8-47b925f2be76. [Kasutatud 15 03 2021].

[69] M. Blaauw ja J. Bonada, „A Neural Parametric Singing Synthesizer Modeling Timbre and Expression from Natural Songs,“ *Applied Sciences,* kd. 7, nr 12, 2017.

[70] O. Jansen, „Training Sequence Generative Adversarial Nets to Compose Music in the abc-notation.,“ Utrecht University Repository, 2017.

[71] B. Sturm, J. F. Santos ja K. Iryna, „Folk music style modelling by recurrent neural networks with long short term memory units,“ Ghent University Department of Electronics and information systems, Ghent, 2015.

[72] „A Visual Exploration of Melodic Relationships within Traditional Music Collections,“ IEEE, 06 12 2018. [Võrgumaterjal]. Available: https://ieeexplore.ieee.org/abstract/document/8564207. [Kasutatud 11 03 2021].

[73] C. Walshaw, „A statistical analysis of the ABC music notation corpus: exploring duplication,“ University of Greenwich, Istanbul, 2014.

[74] „folk tune finder,“ [Võrgumaterjal]. Available: https://www.folktunefinder.com/. [Kasutatud 16 05 2021].

[75] A. Holzapfel ja E. Benetos, „Automatic music transcription and ethnomusicology: a user study,“ Queen Mary University of London, 2019.

[76] „Creative Commons — Attribution 4.0 International,“ [Võrgumaterjal]. Available: https://creativecommons.org/licenses/by/4.0/. [Kasutatud 16 05 2021].

[77] A. Othman ja C. Direkoğlu, „Recognizing Musical Notation Using Convolutional Neural,“ *European Journal of Science and Technology,* pp. 283-290, 2020.

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

Meie, Annett Lymar, Martin Rajur ja Jaan Susi

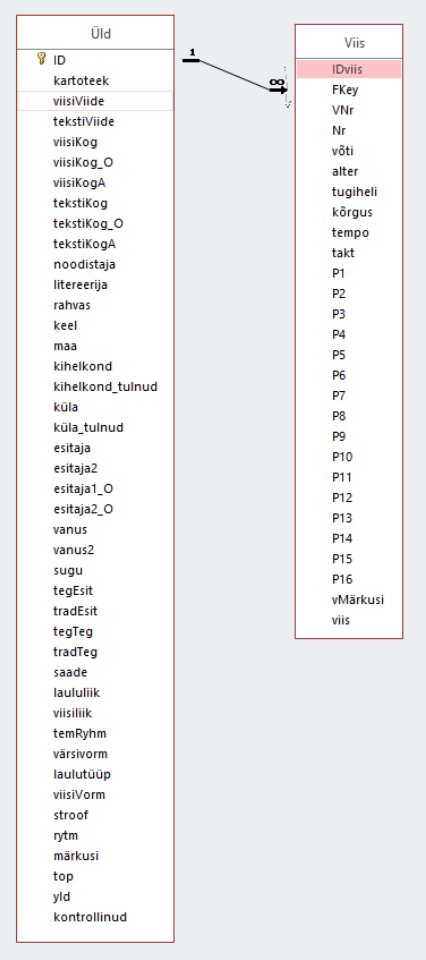
1. Anname Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Eesti Kirjandusmuuseumi jaoks Rahvaviiside noodistuse" , mille juhendaja on "Erki Eessaar"
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse 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. Oleme teadlikud, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autoritele.
3. Kinnitame, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

10.05.2021

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 – Andmete algallika andmebaasi disain

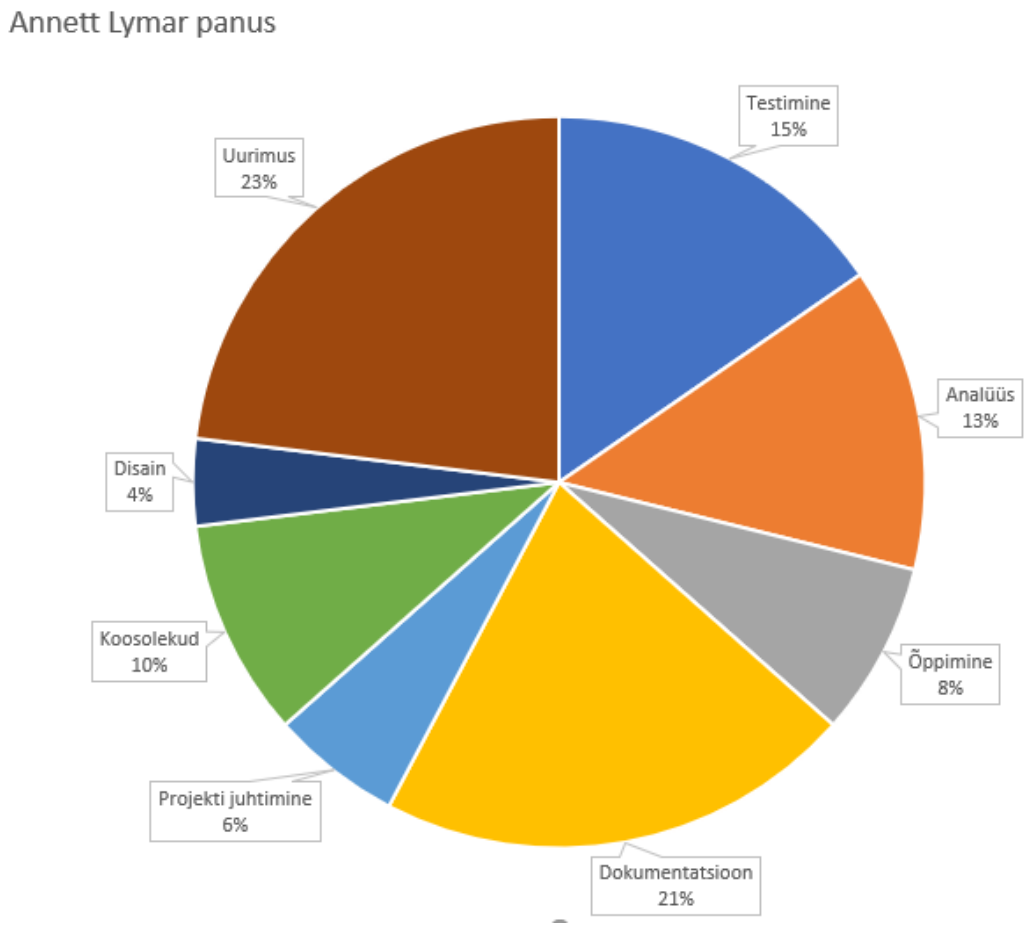
Joonisel on MS Access andmebaasi struktuur (Joonis 105), mis oli andmete teisendamisel aluseks.



Joonis 105 MS Access struktuur

Lisa 3 – Annett Lymari panuse kirjeldus ja eneseanalüüs

Lõputöö kirjutamisel olid minu peamisteks rollideks projektijuhtimine, dokumentatsiooni koostamine ja testimine. Ülesannete protsentuaalselt ajakasutust on näha joonisel (Joonis 106).



Joonis 106 Annett Lymar panus

Projektijuhina korraldasin meeskonnasiseseid koosolekud ja veebikohtumisi tellijaga. Olen projektijuhina IT-sektoris mõnda aega töötanud, tänu millele on mul varasem kogemus tarkvaraarenduse juhtimisest ja meeskonna juhtimisest. Jälgisin, et nädalaks seotud eesmärgid oleks täidetud ja et meeskond liiguks ajakavas seatud tulemuste suunas. Sellele aitas kaasa valitud metoodika, mida jälgiti kogu töö kirjutamise vältel.

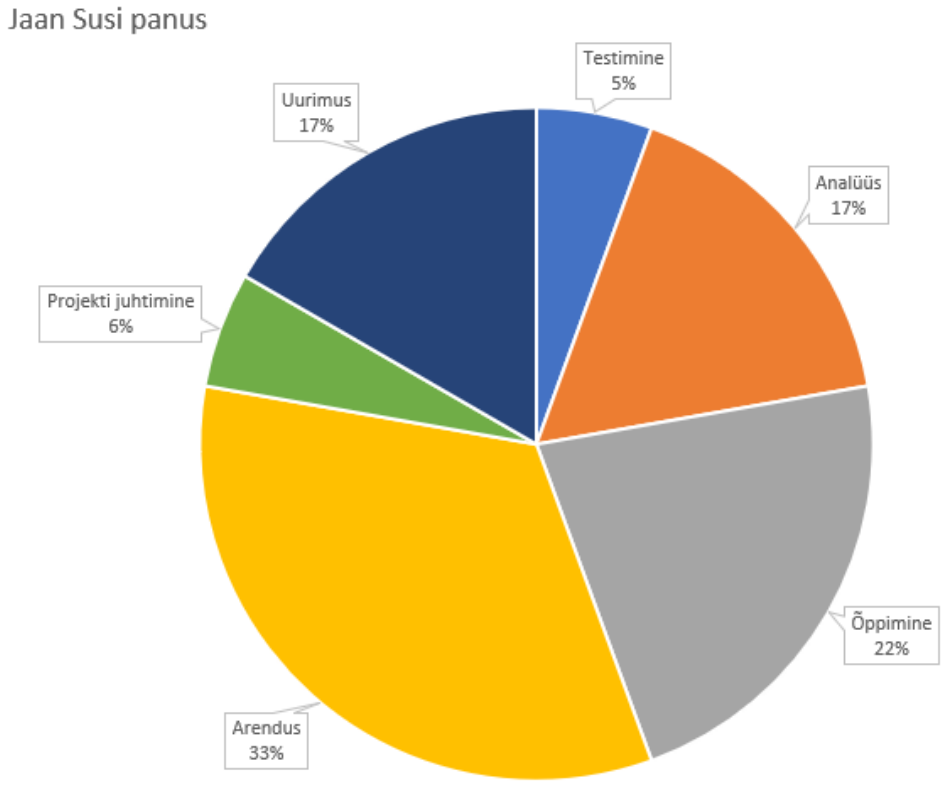
Lisaks oli minu ülesandeks jooksvalt rakenduse funktsionaalsuste testimine ja selle dokumenteerimine, mis aitas arendajal vajadusel parandusi teha. Testisin kasutajaliidest, ja üritasin end panna uue kasutaja rolli, et mõista süsteemi keerukust ja leida viise, kuidas lõpptarbijale süsteemis orienteerumine võimalikult mugavaks teha.

Eesti rahvaviiside valdkond ei olnud mulle varasemalt tuttav. Olen Saksamaal õppinud neli aastat klaverimängu ja nootide lugemist, mis suures plaanis oli ühtne siinsete standarditega. Klaverit mängisin 15 aastat tagasi, mistõttu oli mul vaja teadmisi värskendada.

Suur osa ajast kulus uurimisele ja dokumentidega tutvumisele. Teadusartiklite analüüsile kulus palju aega ja iga teema kohta ei leitudki teadustööd, millele viidata. See tegi lõputöö kirjutamist põnevaks, sest see viitas teema uudsusele.

Lisa 4 – Jaan Susi panuse kirjeldus ja eneseanalüüs

Antud töös kujunesid minu peamisteks ülesanneteks rakenduste analüüs ning arendusprotsessi juurutamine, juhtimine ning arendamine. Ajakasutus jagunes põhiliselt arendusele, sellega seoses tehnikate ning praktikate õppimisele ja rakenduse analüüsi koostamisele (Joonis 107).



Joonis 107 Jaan Susi panus

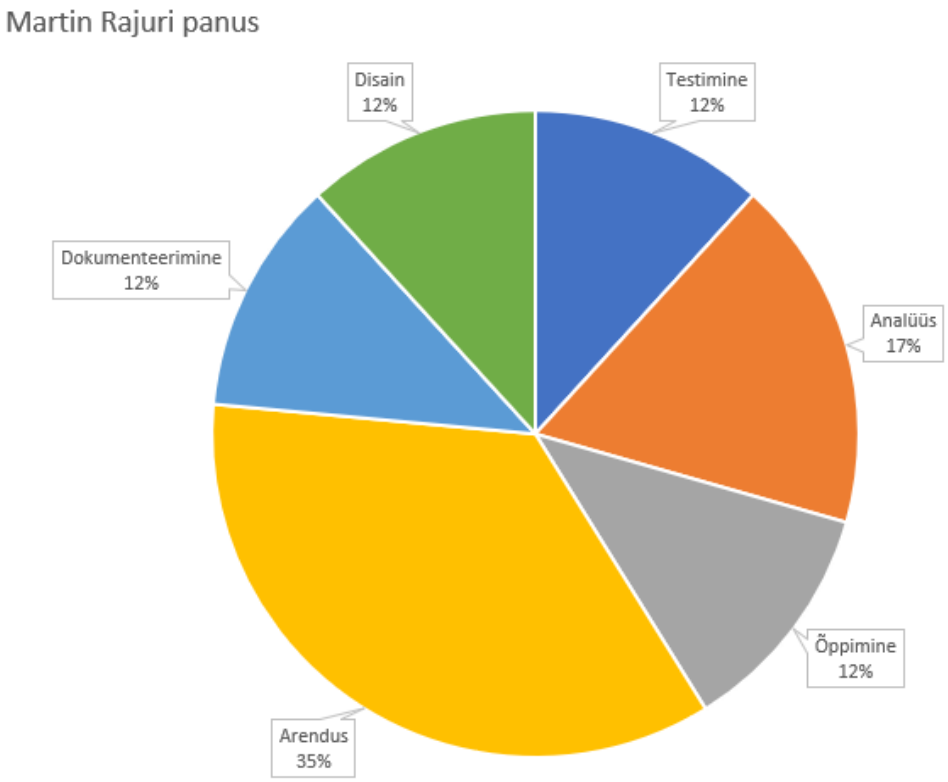
Olles varasemalt kokku puutunud muusikaga ning noodistusega muusikakoolis 15 aastat tagasi, oli EKM-i vajaduste mõistmine minu jaoks projekti alguses lihtsam kui teistele. Projekti käigus tuli küll meelde tuletada noodistiku eripärad, kuid EKM-iga vestlemisel mõistete ning põhimõtete arusaamiseks minul eraldi muusikateooria õppimist vaja ei olnud.

Kuna tegelen igapäevaselt arendustööga, võtsin enda vastutada infosüsteemi rakenduse arenduse juhtimise ning arendusprotsessi üles seadmise. EKM-iga koosolekutel oli minu põhiliseks eesmärgiks aru saada EKM-i vajadustest just rakenduste töövoogudest lähtuvalt ning sellele põhinedes koostada rakenduste arhitektuur ning ülesehitus.

Realiseerisin ka abstraktse vaadete loomise süsteemi esitluskihti, mida oleks kaasautoritel ilma koodi duplikeerimise ohuta võimalik kasutada kiiresti vaadete loomiseks.

Lisa 5 – Martin Rajuri panuse kirjeldus ja eneseanalüüs

Käesoleva projekti raames kujunesid minu põhilisteks rollideks uue lahenduse disain ja arendus. Minul kui meeskonna ainsal liikmel, kellel puudus eelnev muusikaline haridus, oli vaja muusikateooria algpõhimõtted selgeks õppida, selleks, et oleks võimalik muusikalist noodistust sisaldavates teemades kaasa rääkida. Projekti käigus jagunes minu tehtud töö vastavalt Joonis 108



Joonis 108 Martin Rajuri panus

Töö algusfaasis oli minu põhiroll uue platvormi disain ning võimalike alternatiivide otsimine. Kuna mul on pikaajaline kogemus taristu ja platvormi administraatori ning arhitektina, siis oli huvitav disainida võimalikult skaleeritavat, kuid samas võimalikult väikese kuluga realiseeritavat lahendust. Teisalt muusikateooria õppimine, rahvamuusika eripärade ja noodistuse tekkepõhjuste kohta uurimine oli minu jaoks antud töös kõige huvitavam osa. Eelnevalt ei olnud ma teadlik erinevatest võimalikest noodistamise viisidest ning kasutusel oleva noodikirja piirangutest pärimusmuusika kirjeldamisel.

Realisatsiooni faasis oli minu kanda nii arendaja kui testija roll. Koostöös teiste meeskonna liikmetega arendasin lahenduse funktsionaalsust ning samas ka testisin. Kuna arendamise osas on mul olemas küll eelnev kogemus, kuid ma ei pea ennast heaks arendajaks, siis tegelesin peaasjalikult olemasolevate funktsionaalsuste laiendamise ja täiendamisega ning testimise ja vigade otsimisega.

Uue kodeeringu otsimise ja juurutamise osas aitasin koostada detailset lähteülesande kirjeldust ning koostöös EKM-iga tegelesime noodistuse kodeerimise ja selle valideerimisega. Loodud rakendus läbis mitmeid iteratsioone ja muudatusi ning tihti oli vaja võrrelda programmeerimiskeeles programmeerimise ja Exceli funktsionaalsuste rakendamise võimalikku ajakulu ja kasu. Selle tulemusena jõudsime andmetel põhinevate otsuste (näiteks ei teisendatud rütmitüüpe mis olid harva esinevad, osade teisenduste jaoks kasutati programmeerimise asemel vastavustabeleid jne) tulemusel hübriidlahenduseni, mille puhul kasutati andmebaasi sisendi andmiseks kombinatsiooni nii programmeerimiskeeles programmi kirjutamisest kui Exceli funktsioonide kasutamisest.

Uurimuse osas keskendusin peamiselt rahvaviisidele ja noodistustele, kuna see oli peamine teema, mille osas mul puudusid mul vajalikud teadmised. Töö käigus lugesin palju erinevat kirjandust, mis aitas mul paremini mõista ja hinnata erinevate alternatiivide ning lähenemiste sobivust EKM-i jaoks loodavas infosüsteemis. Töö käigus ilmnes ka mitmeid huvitavaid teemasid, mida meie loodud infosüsteemi poolt kogutavate andmetega on tulevikus võimalik teha.