TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Keijo Raamat 202817IADB

Muinsuskaitseameti arheoloogiliste leidude registri prototüüp

Bakalaureusetöö

Juhendaja: German Mumma
MSc

Tallinn 2023

Autorideklaratsioon

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

Autor: Keijo Raamat

15.04.2023

Annotatsioon

Käesoleva lõputöö eesmärk on leida parim lahendus Muinsuskaitseameti probleemile arheoloogiliste leidude haldamisel ja arendada välja parima lahenduse prototüüp.

Töö esimeses osas kirjeldatakse detailselt lahti taust: mis valdkonnaga tegu ning mis on põhilised tegurid problemaatilise olukorra tekkeks.

Teises osas käsitletakse probleemi. Analüüsitakse erinevaid lahendusi ja tehakse valik, millega on otstarbekam probleemi lahendada. Veel selgitatakse välja, mis sammud on vajalikud prototüübi valmistamiseks kasutades kasutajalugude kaardi metoodikat.

Lõputöö keskses osas tehakse kaalutletud otsused töövahendite valikuks ja kirjeldatakse loodava prototüübi disaini. Tuuakse koodi näiteid ja antakse ülevaade arendusprotsessist.

Kogu töös on läbivalt kasutatud välearenduse metoodikaid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 24 leheküljel, 6 peatükki, 16 joonist, 2 tabelit.

Abstract

Prototype of Archaeological Findings Register for National Heritage Board

The aim of this thesis is to find the best solution for the problem faced by the National Heritage Board in managing archaeological findings and to develop a prototype for the best solution.

The first part of the thesis provides a detailed background of the field analysed, and a choice is made regarding the most feasible solution. The necessary steps for creating the prototype using the user story mapping methodology are also outlined.

The central part of the thesis involves making informed decisions on selecting the tools and describing the design of the prototype to be developed. Code examples are provided, and an overview of the development process is given.

Agile development methodologies are used throughout the entire thesis.

The thesis is in Estonian and has 24 pages of text, 6 chapters, 16 figures, 2 tables.

Lühendite ja mõistete sõnastik

AL-ÜV	Arheoloogilise leiu üleandmise-vastuvõtmise akt
bot	Programm, mis automatiseerib ja/või jäljendab inimtoiminguid
eepos	Suurem kasutajalugu, *epic*
CI/CD	*Continious Integration/Continious Delivery* pidevintegratsioon ja pidevedastamine
git	Versioonihaldustarkvara
GitHub	Giti veebiserver koos CI/CD vahenditega
integratsioonitest	Test, mis kontrollib mitme funktsiooni koostööd
MVC	Mudel-Vaade-Kontroller disainimuster
pipeline	Töövoog, tarkvarakonveier – järjestikku ühendatud protsessid
Saaty AHP	Thomas L. Saaty väljatöötatud Analüütiline hierarhia protsessi meetod
teenistuja	Riigi ametiasutuse töötaja.
teek	Kolmanda osapoole loodud programmide kogum, lisatakse oma rakenduse osaks.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Muinsuskaitseametil on vägagi tähtis roll Eesti mineviku uurimise koordineerimisel ja kultuuripärandi säilitamisel. Selline tegevus sisaldab endas suurel hulgal andmete kogumist ja töötlemist lisaks füüsiliste objektide haldamisele. Need on küllaltki ajamahukad tegevused. Mõningaid tegevusi saab aga infotehnoloogiliste vahenditega hõlpsamaks muuta.

Käesolev bakalaureusetöö otsib ühele andmete kogumise ja töötlemisega seotud probleemine lahendust. Töös koostatakse Muinsuskaitseameti arheoloogiliste leidude haldamise töömahukuse vähendamiseks rakenduse prototüüp Leidude haldamise töömahukus seisneb leidude registreerimises, vastuvõtmise-üleandmise akti koostamises ja seadustest tulenevate ajaliste piirangute jälgimisel. Töö eesmärgiks on välja selgitada täpsed Muinsuskaitseameti vajadused ning vastavalt neile arendada registri prototüüp.

Informatsiooni, kuidas seni on Muinsuskaitseametis arheoloogilisi leide registreeritud ja mismoodi kogu töö leidudega välja näeb, kogus autor intervjuust ameti teenistuja Nele Kangertiga. Samuti kasutas autor arheoloog Tuuli Kurisoo ütlusi.

Töös on kasutatud välearenduse metoodikaid: kasutajalugude kaarti, pidevintegratsiooni – pidevedastust. Programmeerimise keel valiti kasutades Saaty AHP metoodikat.

Lõputöö on kirjutatud kolmes peatükis. Esmalt on kirjeldatud sissejuhatavalt tausta, seejärel täpsustakse probleemi ning viimasena antakse ülevaade probleemi lahenduse realiseerimisest. Sellele järgneb kokkuvõte.

2 Taust

Peatükis kirjeldatakse valdkonda, mille probleemi lahendama asutakse. Tuuakse välja seadustest tekkivad nõuded ja kirjeldatakse täpsemalt lõputööd puudutavat probleemset töölõiku Muinsuskaitseametis.

2.1 Mis on arheoloogia, leid ja kultuuriväärtus. Muinsuskaitseameti roll

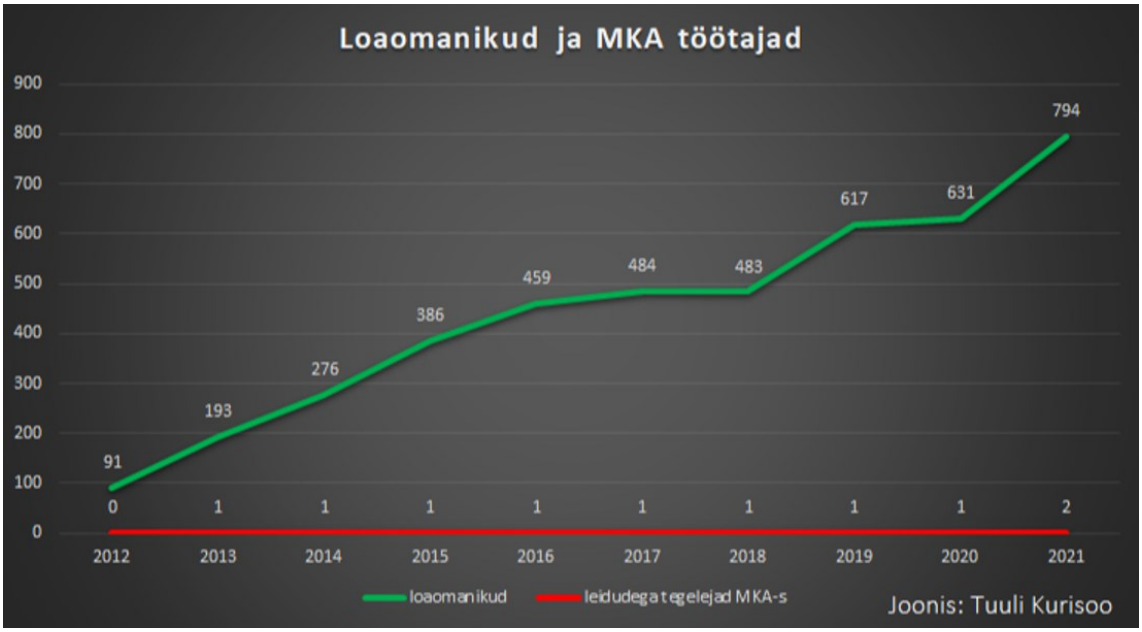
Arheoloogia on teadus, mis uurib ühiskonda inimtegevuse tagajärjel tekkinud materiaalsete tõendite põhjal [1]. Materiaalsed tõendid ehk arheoloogilised leiud on maasse, maapinnale, ehitisse, veekogusse või selle põhjasetetesse ladestunud või peidetud arheoloogiline, sealhulgas ajaloolise, kunstilise, teadusliku või muu kultuuriväärtusega inimtekkeline ese või esemete kogum, millel ei ole omanikku või mille omanikku ei ole võimalik kindlaks teha [2]. Leidude kultuuriväärtuse tuvastamine on tulenevalt Muinsuskaitseseadusest Muinsuskaitseameti kohustus.[2] Esemete olemuse hindamiseks võetakse aluseks leidude infoväärtus, nende määrang (liik, dateering) ja kontekst (eseme leidmise situatsioon). Seega on arheoloogiliste leidude puhul olulised nii ajalised kui ka ruumilised andmed.

Arheoloogiliste esemete leidmine toimub peamiselt kolmel viisil: arheoloogiliste välitööde, otsinguloaga tegutsevate harrastajate või juhuleidmise kaudu. Välitöid teevad ameti loa alusel teadusasutuste või eraettevõtete arheoloogid. Välitööde käigus leitud esemed ja info talletatakse teadusasutuste juures. Otsinguloaga harrastusarheoloogid on aga kohustatud arheoloogilised esemed üle andma ametile, sest kõik kultuuriväärtusega leiud kuuluvad riigile.

2.2 Arheoloogiliste leidude teekond Muinsuskaitseametis

Eestis on järjest populaarsemaks muutunud metallidetektori abil kultuuriväärtusega esemete otsime. See on lisaks kodanike liikumisharjumistele otseselt mõju avaldanud ka ameti arheoloogiliste leidudega tegelevate teenistujate töökoormusele. Jooniselt 1 on

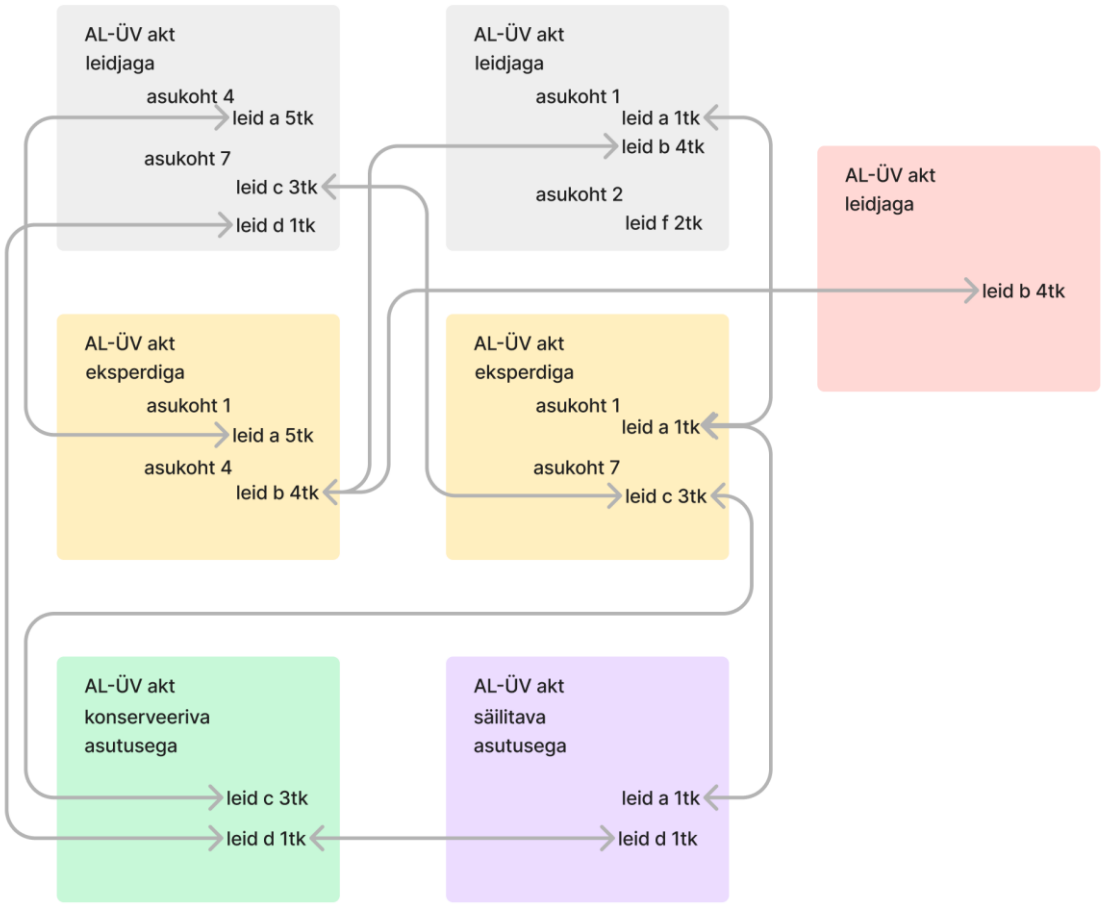
näha, et otsinguloa omanike arv on kasvanud kümne aastases ajavahemikus keskmiselt 87% aastas kui teenistujate arv vaid 20%.



Joonis 1. Otsinguloa omanikud ja leidudega tegelevad töötjad aastatel 2012-2021.

Eduka otsingu tulemusel leitud esemed antakse ametile üle arheoloogilise leiu üleandmise-vastuvõtmise aktiga (AL-ÜV). Selles aktis on üles tähendatud leidja andmed, otsingud (AL-ÜV koostatakse tihti mitmes kohas toimunud otsingu kohta) ja sealt leitud esemed. Leidjaga koostatud AL-ÜV-s on leidude nimetused ja määrangud tihti üldised. Täpsemalt toimub leidude määramine ja dateerimine järgmises etapis, mil ameti poolt palgatud arheoloogid määravad üleantud esemeid. Leiud liigutatakse ekspertiisi järgmise AL-ÜV alusel. Tuleb tähele panna, et ekspertiisi ei liigu leiud enam otsimise põhjal, vaid ekspertiisi suunatakse kogum leidudest. Mõned esemed võivad vajada konserveerimist ja seda nii enne ekspertiisi kuid ka ekspertiisi käigus. Konserveerimist vajavad leiud antakse üle vastavat teenust pakkuvale asutusele üle uue AL-ÜV alusel. Viimases etapis koostakse AL-ÜV akt leiu säilitava asutusega. Vastavalt ekspertiisi tulemusena võivad

leiud liikuda ka tagasi leidjale (need ei oma kultuuriväärtust). Joonisel 2 on näitlikustatud leidude liikumist läbi aktide ametis.



Joonis 2. Näide leidude liikumisest läbi aktide.

Tulenevalt kultuuriministri määrusest [3] on lisaks sellisele andmete liikumise voole seatud mõningad ajalised kriteeriumid. Kultuuriväärtus tuleb määrata 6 kuu jooksul alates leidjaga AL-ÜV sõlmimist. Kui vajalikus osutub konserveerimine või täiendavad uuringud, siis kultuuriväärtus tuleb määrata hiljemalt 3 kuu jooksul peale tööde lõppu.

Kõiki tööks vajalikke akte hallatakse Muinsuskaitseametis Webware loodud dokumendihaldus süsteemis nimega Webdesktop. See rakendus on dokumentide järjehoidmiseks mõeldud, mitte aga dokumentide vahel kirjete liigutamiseks. Seetõttu ei saa hõlpsalt jälgida, millises staatuses leiud (nt. kas ekspertiisis või konserveerimisel). Ametis on kasutusele võetud tabelarvutus programm Microsoft Excel saamaks ülevaadet leidude seisust. Exceli tabelisse on kirjutatud välja kõik aktides olev info mille põhjal saab määrata eseme staatuse. Väikeste mahtude juures see süsteem õigustas end. Kuid suurenenud leidude hulga tõttu on muutunud selle kasutamine väga aeganõudvaks.

3 Probleemi kirjeldus ja lahenduse otsimine

Selles peatükis kaardistatakse Muinsuskaitseameti vajadused. Probleemi lahendamise uurimiseks kasutati esmase meetodina intervjueerimist. Leidudega tegeleval teenistujal paluti selgitada olemasolevat süsteemi ja soove selle muutmiseks. Intervjuu tulemustest välja koorunud funktsionaalsed nõuded on lahti kirjutatud lugude kaardil. Mittefunktsionaalsed nõudeid on kirjeldatud vastavas alampeatükis.

3.1 Probleemi kirjeldus

Muinsuskaitseametis on dokumendipõhine asjaajamine. Sellel on suurimaks puuduseks töömahukas protsess saamaks ülevaadet leitud esemetest. Mõningal määral on leevendust toonud Exceli tabeli kasutamine, kuid kasvanud töömahu juures ei käi teenistujate jõud sellest üle.

Uurides lähemalt, kuidas toimub ekspertiisi protsess selgus, et väga palju andmeid võetakse varasematest töödest. 2020 aastal alustas Tuuli Kurisoo teadusprojekti MetDect, mille käigus uurib ta detektorileide[4]. Selle projekti raames võeti ametis kasutusele ArcGis-il baseeruv kaardirakendus. Sellest kaardirakendusest saadakse kogu Eestit hõlmav ülevaade detektoriga leitud arheoloogilistest esemetest. Andmed leidudest liigutatakse kaardirakendusse eraldi Exceli tabeli kaudu. Osaliselt andmed kaardirakenduse Exceli tabeli ja aktide-leidude tabeli vahel kattuvad.

Tabelarvutuse korral tekib mitmekordset andmete käsitsi kopeerimist ühest tabelist teise (tabelist akti, aktist tabelisse, tabelist tabelisse jne). Autor näeb selles mitut probleemi. Esmalt ajakulu, kui korra on andmed sisestatud, siis ei ole otstarbekas neid uuesti sisestada, isegi kopeeri-kleebi tehnikat kasutades. Teiseks on küllaltki tülikas tagada andmete terviklust. Väga kergelt võib üks rida või lahter jääda kopeerimisest välja ning see eksimus võib jääda märkamatuks.

3.2 Lahenduse valik probleemile

Leidude haldamise probleemi saaks lahendada järgmistel viisidel:

- Koostada täiustatum tabeltöötlusprogrammi lahendus. Sellise lahenduse puhul tekiks mugavaima kasutaja liidese loomisel, seda eriti väiksema ekraaniga seadmest - võib osutuda tulevikus piiravaks. Samuti on küllaltki keeruline kui mitte võimatu lisada kolmanda osapoole lahendusi, näiteks Maa-ameti integreeritav aadressiotsing In-ADS [5].

- Võtta kasutusele mõni sobiv olemasolev süsteem. Intervjuus ameti teenistujaga selgus, et iga riik otsustab oma pärandi eest omasoodu ja ole ühtset seadustest tulenevat süsteemi. Seega naaberriikidest ei oleks sobivat lahendust laenata. Autor uuris ka mõningaid valmis laosüsteeme (sisuliselt on tegu ju arheoloogiliste leidude lao ja logistikaga) kuid ei leidnud ühtegi, mida oleks võimalik muuta ameti vajadustele vastavaks.

- Spetsiaalselt ameti vajadustele ehitatud tarkvara. Kui rakendus arendada tihedas koostöös kasutajatega siis on võimalik tagada rakenduse parim kasutamise kogemus ja seeläbi vähendada aega, mis kulub andmete töötlemisele. Seda lahendust on võimalik muutuvate nõuete korral muuta.

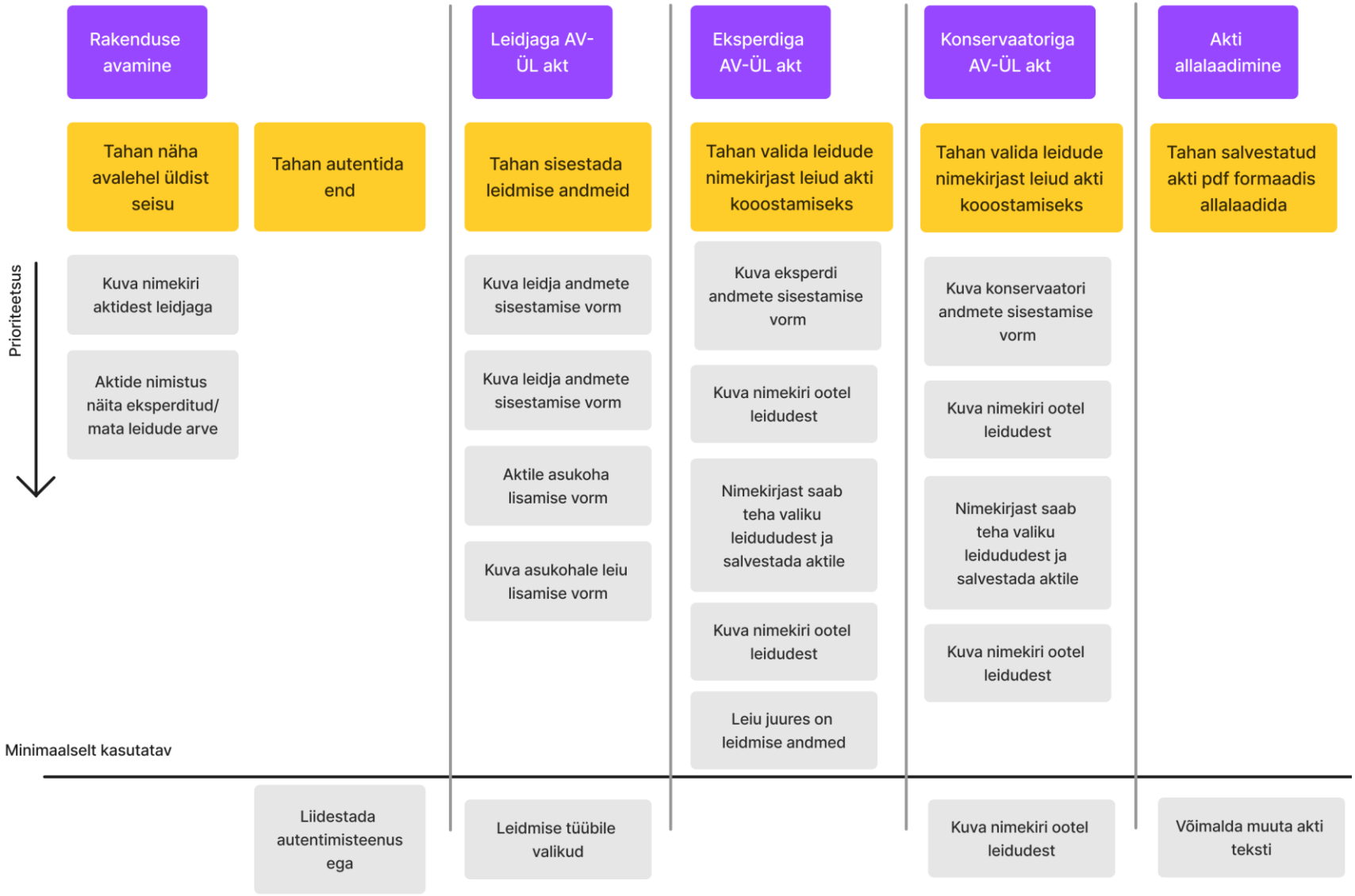
Seega kõige parem lahendus on luua ameti vajadusi silmas pidades iseseisev rakendus. Nii saab kergemalt tagada andmete terviklust aktide koostamisel ja vähendada andmete töötluseks kuluvat aega. Samuti on kergem liidestada erinevaid rakendusi/andmebaase. Autor leiab, et andmed peavad olema inimese teenistuses mitte vastupidi ja see on saavutatav läbi tarkvara, mis on kasutajatele hõlbus kasutada.

3.3 Mittefunktsionaalsed nõuded

Intervjuust ameti teenistujaga selgus, et on soov jagada leidude registri arendus mitmesse etappi. Esimeses osas soovitakse kasutada rakendust ainult aktide koostamiseks ja leidude seisu jälgimiseks. Kuna amet ei ole kindel, kas ja kus serveris oleks ruumi rakenduse käitamiseks siis soovitakse esimeses faasis kasutada rakendust ainult kasutaja arvutist. Kuid, et kasutajaid saaks olema rohkem kui üks on vaja andmeid mingit moodi jagada. Hetke lahenduses on Exceli fail jagatud kasutajate vahel läbi kohtvõrgus oleva failiserveri. Sarnase portatiivsuse saavutamiseks peab rakendus olema kergesti paigaldatav (eelistatult 1 käivitatav rakenduse fail).

3.4 Funktsionaalsed nõuded lugude kaardil

Lugude kaart on välearenduse meetod, kus rakenduse funktsionaalsus struktureeritakse ja jaotatakse prioriteetsuse järjekorda [6]. Selline lähenemine on justkui sillaks kasutaja ja arendaja vahel. Kasutaja räägib loo mida taha tahab teha ja selle tegevusega saavutada ja arendaja selgitab, mis ülesanded on vajalikud selle loo reaalsuseks saamisel. Seejärel prioritiseeritakse ülesanded ja otsustatakse ilma milliste ülesanneteta ei ole lugu ehk rakenduse omadust võimalik kasutada. Sellisel viisil järgitakse välearenduse manifesti põhimõtet: „Lihtsus - ebavajaliku töö tegematajätmise kunst - on väga oluline“ [7]. Joonisel 3 on lugude kaart, millele on autor intervjuu põhjal välja kirjutanud kasutaja lood vormis „Tahan teha seda …“. Need lood on grupeeritud eeposte ehk üldisema sõnastusega funktsionaaluste alla.



Joonis 3. Funktsionaalsed nõuded lugude kaardil.

4 Lahenduse realiseerimine

Selles peatükis selgitakse välja sobivaimad töövahendid prototüübi valmistamiseks ja kirjeldatakse detailsemalt rakenduse ülesehitust ning arenduse protseduure.

4.1 Töövahendite valik

Osad töövahendid nagu näiteks andmebaasi mootor on võimalik arenduse jooksul vahetada kergesti, seda muidugi juhul kui on järgitud vastutuste eraldamise printsiipi. Teised töövahendid, nagu näiteks programmeerimiskeel, on väga raske ja töömahukas vahetada. Seetõttu on töö autor pannud suuremat rõhku sobivaima programmeerimiskeele valikule.

4.1.1 Programmeerimiskeele valik

Valiku langetamiseks kasutas autor Thomas L. Saaty analüütilist hierarhia protsessi (AHP). AHP võimaldab abstraktseid valikuid sisaldavaid keerukaid otsuseid vastu võtta konkreetsete näitajate põhjal. AHP kasutamiseks on vaja formuleerida eesmärk, kriteeriumid ja alternatiivid. Alternatiividele antakse arvuline väärtus vastavalt kriteeriumile kui oluline see on eesmärgi saavutamiseks. Väärtused valitakse kriteeriumite/alternatiivide paaride võrdlemise teel: küsitakse kumb olulisem on ja antakse numbriline väärtus. Vastavalt siis 1 tähistab, et A ja B võrdne paar, 3 tähistab, et A on B-st mõõdukalt parem, 5 tähistab, et A on B-st märkimisväärselt parem, 7 tähistab, et A on B-st oluliselt parem ja 9 tähistab, et A parim. Numbrilised väärtused kirjutakse maatriksina üles ja need arvutatakse kaaludeks. Kaalude põhjal saab väita, milline alternatiiv on parim valik eesmärgile [8].

Alternatiivideks, mille vahel kaaluda moodustati neist keeltest, milles programmeerimises on autoril mingil määral kogemust: Java, C#, Pyhton, Go ja PHP. Kriteeriumiteks valis autor järgmised aspektid:

- Ressursinõudlikkus – väiksem nõudlikkus hoiab hilisemad käitamiskulud madalamad ja ökoloogilise jalajälje väiksemana. Kõige täpsema ressursi nõude

saab vaid valmis rakenduse koormuse all töötades mõõtes. Töö autor võtab indikatiivsete suuruste saamiseks ajakirjas Science of Computer Programming 2021 aastal ilmunud artikli Ranking programming languages by energy efficiency esitatud andmed [9]. Selles artiklis on mõõdetud protsessori kulutatud energiat, aega mis kulus programmi tööks ning mälu kasutust. Kuna need 3 mõõdikut on sarnase iseloomuga – mida väiksem mõõt seda parem, siis võtab autor need kolm arvu kokku üheks nn. skooriks.

- Portatiivsus – vähem nõudeid paigaldamisel lihtsustab tööd ja seeläbi hoiab kulud minimaalsed. Tulenevalt mittefunktsionaalsetest nõuetest on oluline, et rakendust saaks käivitada vähese arvuti kasutamise kogemustega isik. Seega on portatiivsus kergelt paigaldamise vaatevinklist oluline kriteerium. Kuigi pea igas keeles kirjutatud lähtekoodist on võimalik käivitatavat binaarfaili luua, siis hindab töö autor kõrgemalt neid keeli, millest saab kõige lihtsamalt käivitatavat faili luua.

- Autori kogemus – käesoleva töö praktilise osa valmimise seisukohast äärmiselt tähtis. Kõigist kriteeriumitest on tegu väga subjektiivsega ja ei oma olulist tähtsust valmiva rakenduse kasutamisel ega haldamisel.

Tabel 1. Programmeerimiskeelte ressursi vajadused [9].

Keel	Energia (j)	Aeg (ms)	Mälu (Mb)	Skoor
Java	1,98	1,89	6,01	9,88
C#	3,14	3,14	2,85	9,13
Python	75,88	71,90	2,80	150,58
Go	3,23	2,83	1,05	7,11
PHP	29,30	27,64	2,57	59,51

Vastavalt AHP eeskirjale leitakse esmalt kaalud kriteeriumitele. Seejärel alternatiividele iga kriteeriumi kohta. Arvutused on detailsemalt välja toodud Lisas 2.

Kriteeriumite kaalud: (Ressursinõudlikkus, Portatiivsus, Kogemus) = (0,539, 0,297, 0,164)

Tabel 2. Keeltele arvutatud kaalud kriteeriumite kaupa.

Keel / Kriteerium	Java	C#	Python	Go	PHP
Ressursinõudlikkus	0,251	0,251	0,029	0,396	0,073
Portatiivsus	0,101	0,233	0,055	0,568	0,042
Kogemus	0,164	0,074	0,391	0,074	0,296

Korrutades omavahel alternatiividest moodustuva maatriksi kriteeriumite maatriksiga on tulemuseks sobivaima keelete kaalude maatriks.

$$\begin{pmatrix} 0{,}251 & 0{,}101 & 0{,}164 \\ 0{,}251 & 0{,}233 & 0{,}074 \\ 0{,}029 & 0{,}055 & 0{,}391 \\ 0{,}396 & 0{,}568 & 0{,}074 \\ 0{,}073 & 0{,}042 & 0{,}296 \end{pmatrix} \begin{pmatrix} 0{,}539 \\ 0{,}297 \\ 0{,}164 \end{pmatrix} = \begin{pmatrix} 0{,}192 \\ 0{,}217 \\ 0{,}096 \\ 0{,}394 \\ 0{,}1 \end{pmatrix}$$

Kuna Go kaaluks on 0,394, mis on ligi 2 korda suurem kui paremuselt teisele kohal olev C# - 0,217, siis käesoleva töö praktiline osa teostatakse keeles Go.

4.1.2 Raamistiku valik

Raamistik veebi–rakenduse kirjutamiseks ei ole Go puhul hädavajalik. Go standard teekide kogum sisaldab kõiki komponente, mida on vaja veebi-serveri ehitamiseks. Kuid raamistikud lisavad teatud määral arenduskiirust, sest neisse on valmis kirjutatud konkreetsemaid komponente. Näiteks ruuteri ehk päringute jaotur ehitamiseks on komponendid Go standard–teegis olemas, kuid raamistikes on ruuter valmis ja saab keskenduda rakenduse toimimisloogika realiseerimisele.

Kuigi Go on küllaltki uus keel, on sellel mitmeid raamistikke, mille vahel valida. Populaarseimad raamistikud on Gin, Fiber ja Iris. Gin dokumentatsioon [10] koosneb peamiselt näidetest, mis väga suure kogemusega arendajale võib mugav olla, kuid autor leiab, et ta kui väheldase kogemusega arendaja vajab rohkem selgitusi. Iris

dokumentatsioon [11] on küll põhjalik koos näidetega, kuid see on ei ole väga struktureeritud - vajaliku info leidmine nõuab omajagu aega. Fiber dokumentatsioon hõlmab nii näiteid kui ka selgitusi [12]. Dokumentatsioon on struktureeritud ning kergesti aimatav, millises peatükis mida selgitatakse. Autor valib raamistikuks Fiber-i.

4.1.3 Andmebaasi valik

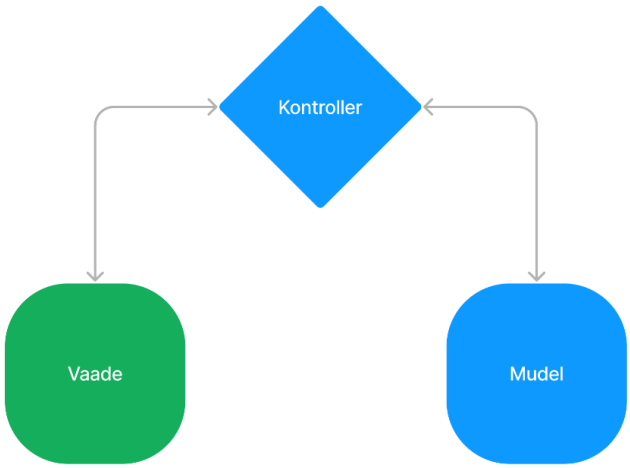
Kasutajatelt rakendusse sisestatavad andmed on väga selgelt struktureeritud, töö autor ei näe, et nende järjepidevus muutuks. Seega struktureeritud päringu keel ehk SQL on hea valik. Tulenevalt prototüübi mittefunktsionaalsest nõudest (portatiivsus) peab andmebaasi rakendus olema kas kasutaja arvutis olemas olema või väga kergesti paigaldatav. Neile kahele tingimusele vastab SQLite [13]. Rakenduse hilisemas elukaare faasis tasuks mõelda suuremat koormust kannatava andmebaasi süsteemi peale. Selle hõlpsaks teostamiseks tuleb rakenduse arhitektuuris arvestada. Samuti on suureks abiks ORM. ORM ehk lahti kirjutatuna Object-Relational Mapping on tehnika sidumaks omavahel rakenduse objektid ja andmebaasi relatsioonid ehk tabelid [14]. Vaatamata sellele, et Go ei ole puritaanlikult võttes objektorienteeritud keel, on siiski andmed seal struktureeritud ja Go-d saab tingimisi objekt orienteeritud keelena vaadelda ja seetõttu on ORM-i kasutamine võimalik. Kuna autor on varasemalt kokku puutunud Go jaoks realiseeritud ORM vahendiga GORM [15] valib selle rakenduses kasutamiseks.

4.2 Rakenduse arhitektuur

„Hea arhitektuur muudab süsteemi hõlpsasti arusaadavaks, hõlpsasti arendatavaks hõlpsasti käideldavaks hõlpsalt rakendatavaks. Peamine eesmärk on minimeerida süsteemi maksumust ja arendaja tootlikust rakenduse kogu elukaare jooksul.“ – Robert C. Martin [16]

Autor valib rakendus arhitektuuriliseks ülesehituseks MVC disaini mustri. MVC on üks lihtsamaid mustreid nii kirjutamise kui loetavuse seisukohast. MVC ehk Model-View-Controller mustri puhul jaotatakse rakendus kolmeks komponendiks - mudeliks, vaateks ja kontrolleriks. Mudel vastutab andmete struktuuri ja andmebaasi salvestamise eest,

vaade kasutajaga andmete vahendamise ja kontroller vaate andmete ja mudeli vahelise suhtluse eest [17]. Joonisel 5 on kujutatud skemaatiliselt MVC disaini mustrit.



Joonis 4. Klassikaline MVC disainimuster.

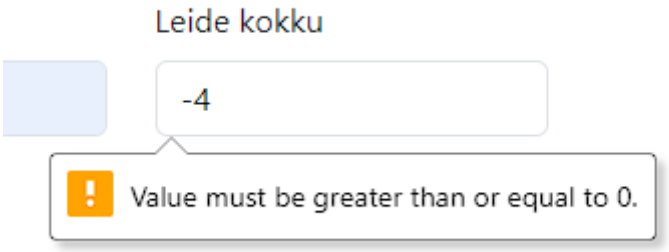
4.2.1 Vaadete ülesehitus

Vaated on võimalik luua kahel viisil: lehitsejas või serveris. Käesolevas töös kasutab autor serveri poolset vaate genereerimist. Seda selleks, et lehitsejas HTML koodi genereerimiseks oleks vaja kasutada lisa tehnoloogiat (JavaScript või WASM), kuid autor leiab, et käesolevale tööle ei lisaks see märgatavat lisandväärtust. Kasutajamugavuse suurendamiseks ja visuaalselt terviklikuma pildi saamiseks kasutab autor Bootstrap tööriistakomplekti. Bootstrap sisaldab nii komponente HTML elementide loomiseks kui ka kujundamiseks [18]. Kasutajamugavus väljendub näiteks vaates sisestatud andmete esmases valideerimises. Joonisel 5 on seatud HTML-i elemendile atribuudid type=“number“ ja min=“0“, mis piiravad kasutajat kogemata mitte numbrilist või negatiivset numbrit leidude koguseks sisestamaks.

```
25      <div class="col">
26        <label for="findingsAmount" class="form-label">Leide kokku</label>
27        <input type="number" min="0" class="form-control" id="findingsAmount" name="findingsAmount">
28      </div>
```

Joonis 5. Bootstrapi vormi validaator.

Joonisel 6 on näha ebasobiva suuruse sisestamisel antavat veateadet. Enne andmete andmebaasi sisestamist kontrollitakse siiski rakenduse sees andmed, et kindlustada andmete kvaliteet.



Joonis 6. Ebasobiva suuruse korral kuvatav veateade.

Vaate genereerimiseks on kasutusel mallid, inglise keeles *templates*. Mallid on jaotatud alam-mallideks, mis kutsutakse välja vastavalt sisendandmetele. Näiteks genereeritakse tabel ja täidetakse andmetega kui vaate genereerimisele on kaasa antud vastav muutuja (Joonis 7.).

```
1   {{template "partials/top"}}
2
3   <h1>Leidmine</h1>
4   {{ template "findings/components/fetchActAttributes" .}}
5
6   {{ if .Locs }}
7   {{ template "findings/components/locationList" . }}
8   {{ end }}
9
10  {{ template "findings/components/addLocation" . }}
11
12  {{template "partials/bottom"}}
13
```

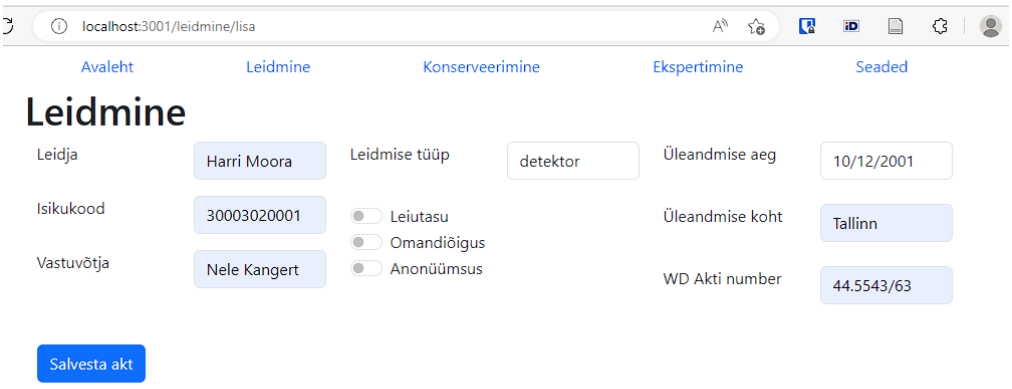
Joonis 7. Vaate malli näidis.

Alam-mallide kasutamine võimaldab mallide taaskasutamist ja struktureerib koodi loetavamaks. Näiteks saab Joonisel 8 olevat alam-malli kasutada igal lehel kus kasutaja soovib näha nimekirja asukohtadest.

```
<table class="table">
  <thead>
    <tr>
      <th scope="col">Maakond</th>
      <th scope="col">Vald</th>
      <th scope="col">Küla</th>
      <th scope="col">KÜ lähinimi</th>
      <th scope="col">KÜ</th>
      <th scope="col">Leide kokku</th>
      <th scope="col"></th>
    </tr>
  </thead>
  <tbody class="table-group-divider">
    {{ $actId := .Act.ID }}
    {{range .Locs }}
      <tr>
        <td>{{.County}}</td>
        <td>{{.Parish}}</td>
        <td>{{.Village}}</td>
        <td>{{.CadastralUnitName}}</td>
        <td>{{.CadastralUnitCode}}</td>
        <td>{{.FindingsAmount}}</td>
        <td>
          <form action="/leidmine/akt/{{$actId}}/eemalda_asukoht/{{ .ID }}" method="POST">
            <button type="submit" class="btn btn-danger">-</button>
          </form>
        </td>
      </tr>
    {{end}}
  </tbody>
</table>
```

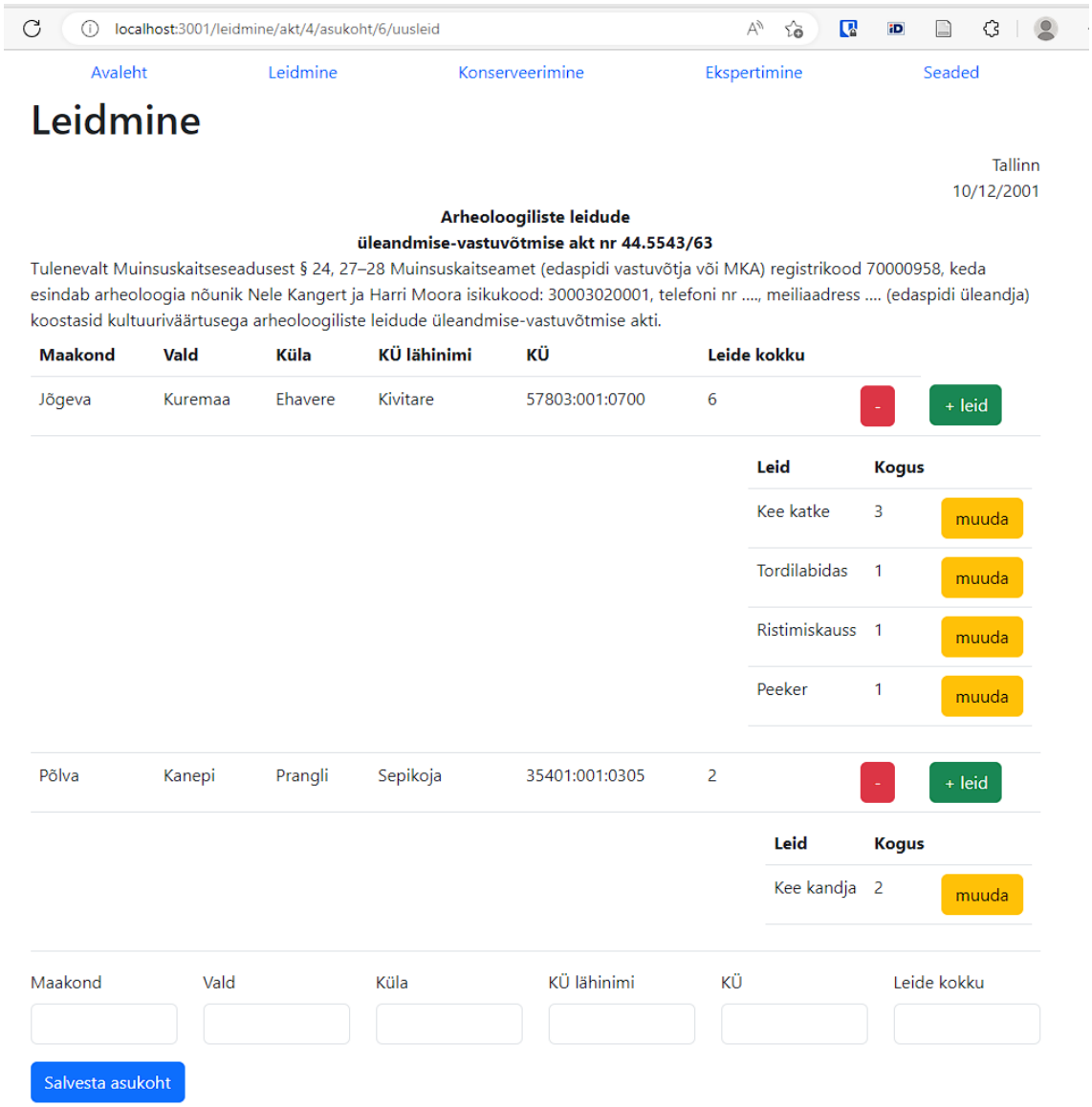
Joonis 8. Alam-malli näide.

Joonisel 9 on näidatud leidjaga akti atribuutide sisestamise vorm.



Joonis 9. Leidjaga akti sisestamise vaade.

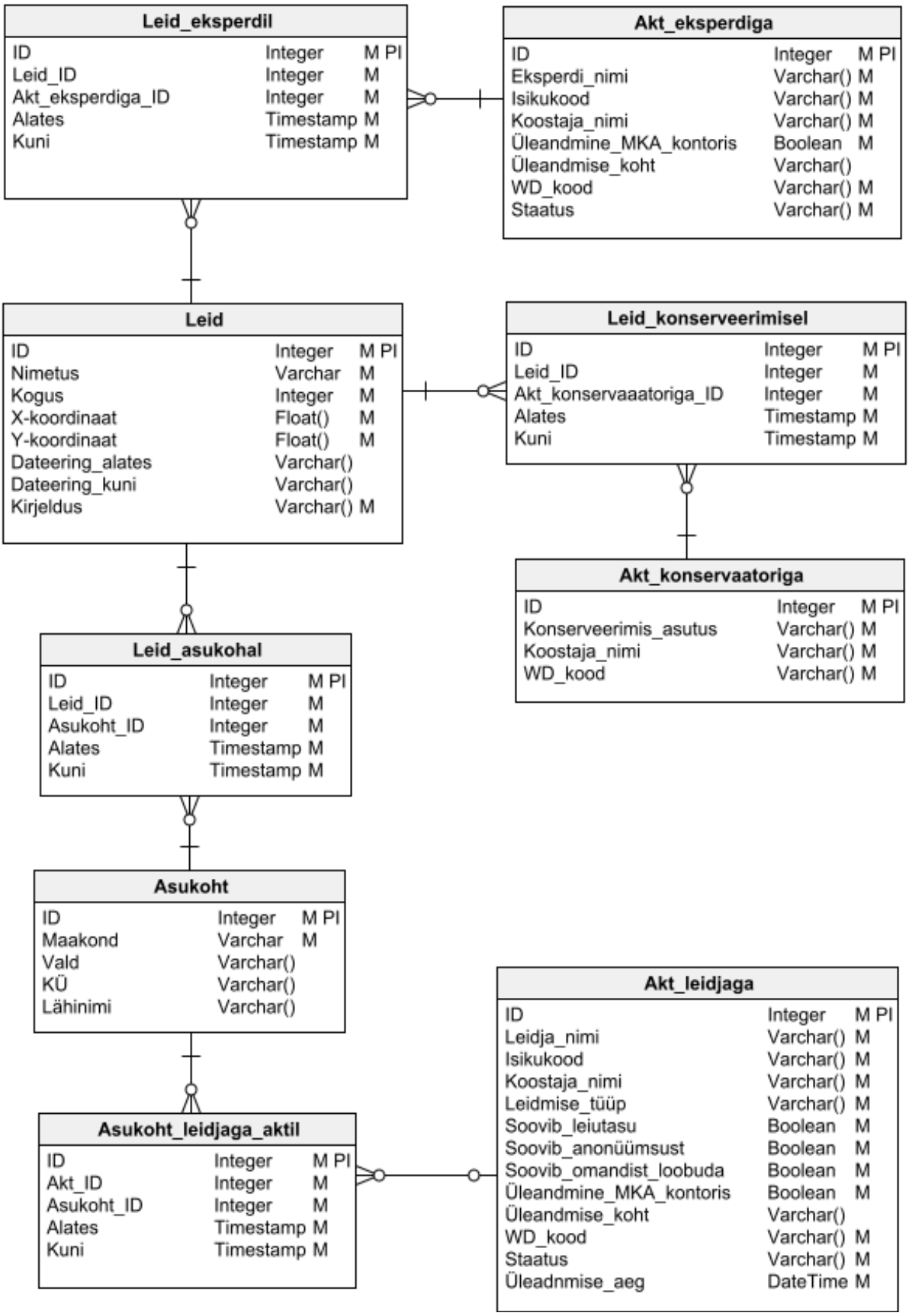
Joonisel 10 on näidatud leidjaga aktile asukohtade ja asukohtadele leidude sisestamise vaade.



Joonis 10. Leidjaga aktile asukohtade lisamise vaade.

4.2.2 Andme mudeli kirjeldus

Mudelite loomise aluseks joonistas autor välja andmebaasi olemi mudeli skeemi (Joonis 11). Andme olemid on omavahel mitu-mitmele seoses. See tähendab, et näiteks ühes leidmise aktil võib olla mitu asukohta ja üks asukoht võib olla mitmel leidmise aktil. Kuna see tekitaks ring-sõltuvuse on kasutusel suhet näitav olem ehk vahetabel. Suhte olemis on kirjas, millal suhe algas ja lõppes. Joonisel 12 on näidis koodist, kus tehakse päritakse leidmise akti identifikaatori järgi läbi vahetabeli kõik asukohad sellel aktil.



Joonis 11. Andmebaasi olemi mudel.

```
11  func GetLocationsByFindingActID(actID string) ([]models.Location, error) {
12          var locs []models.Location
13
14          id, err := strconv.ParseUint(actID, 0, 20)
15          if err != nil {
16                  log.Println("Error converting id from url to int")
17          }
18
19          err = initializer.DB.
20                  Joins("JOIN finding_locations ON locations.id = finding_locations.location_id").
21                  Where("finding_locations.finding_act_id = ?", id).
22                  Find(&locs).Error
23          if err != nil {
24                  log.Printf("Error quering locations for finding act %s", actID)
25                  return nil, err
26          }
27
28          return locs, nil
29  }
```

Joonis 12. Funktsioon päringuga andmebaasi ja tagastamas aktil olevad asukohad.

4.3 Pidevintegratsioon ja pidevedastamine

Pidevintegratsioon ja pidevedastamine on ehk inglise keelse lühendina CI/CD välearenduse metoodikate kogum, mis võimaldavad tõsta tarkvara kvaliteeti ja tootmise kiirust läbi korduvate tegevuste automatiseerimise. Pidevintegratsioon või ka pidevarendus (*continuous integration*) on meetod, mille korral jagatud lähtekoodi hoidlas integreeritakse pidevalt lisandunud kood. Eduka integreerimise eelduseks on automaatselt käivituvad testid[19]. Pidevedastamine (*continuous delivery*) on pidevintegratsiooni pikendus, mille käigus edastatakse lähtekoodist valmis ehitatud rakendus kasutusse.

Käesolevas töös on lähtekoodi versioneerimisvahendiks kasutusel git ja jagatud hoidlana GitHub. Pidevintegratsioon ja pidevedastamine on realiseeritud GitHubi teenuse Actions kaudu. Samuti on seadistas autor teekide ajakohasuse kontrolliks ja automaatseks uuendamiseks tööriista nimega Dependabot. Dependabot võrdleb Go sõltuvuste nimistus (fail nimega go.mod) olevaid versiooni numbreid vastava sõltuvuse viimatise numbriga. Leides uue versiooni loob Dependabot avalduse, nii öelda *pull requesti*, uuema versiooni lisamiseks.

4.3.1 Automaattestid ja pidevintegratsiooni töövoog

Pidevintegratsiooni edukaks toimimiseks koostas automatiseeritud töövoo ehk inglise keeles *pipeline*. See töövoog käivitab iga uue koodi lisandumisel testid. Teste on koostanud töö autor kahte tüüpi: ühik- ja integratsioonitest. Joonisel 13 on andmebaasi

```
44  func Test_GetAllLocations_Should_Return_Slice_Of_Locations_Containg_Given_Location(t *testing.T) {
45      l := models.Location{
46          County:           "Vagula",
47          Parish:           "",
48          Village:          "Liiva",
49          CadastralUnitCode: "44:67",
50          CadastralUnitName: "gemäe",
51          FindingsAmount:    2,
52      }
53      loc, _ := repository.AddLocation(l, db)
54      actual, _ := repository.GetAllLocations(db)
55
56      var isInSlice = false
57      for _, location := range actual {
58          if location.ID == loc.ID {
59              isInSlice = true
60              return
61          }
62      }
63
64      if !isInSlice{
65          t.Errorf("GetAllLocations() returned slice without inserted location %d", loc.ID)
66      }
67
68  }
```

Joonis 13. Asukoha andmebaasi salvestava funktsiooni test.

asukoha andmeid salvestava funktsiooni test. Testis luuakse asukoht ja antakse see paketi repository AddLocation funktsioonile. Seejärel päritakse andmebaasist kõik asukohad ja kontrollitakse kas testi algul sisestatud asukoht on tagastatud asukohtade hulgas.

Integratsiooni testid kontrollivad mitme funktsiooni omavahelist tööd. Joonisel 14 on, integratsioonitest mis edastab läbi http Post tüüpi päringu kontrollerile leidmise akti andmed ja kontrollib, kas need andmed salvestatakse andmebaasi.

```go
func TestFinding_Act_Should_Be_Inserted_Into_DB_When_Form_Is_Submitted(t *testing.T) {
    act := &models.FindingAct{FinderName: "Helka", WDActNumber: "666", RecieverName: "Mikk"}

    // Setup new Fiber app
    engine := html.New("../views", ".tmpl")
    app := fiber.New(fiber.Config{
        Views: engine,
    })
    app.Static("/", "./public")

    // Setup routes
    faC := controllers.FindingActController{DB: db}
    app.Get("/leidmine/lisa", faC.NewFinding)
    app.Post("/leidmine/lisa", faC.CreateFinding)

    // Setup request body
    body := new(bytes.Buffer)
    writer := multipart.NewWriter(body)
    writer.WriteField("finderName", act.FinderName)
    writer.WriteField("WDActNumber", act.WDActNumber)
    writer.WriteField("RecieverName", act.RecieverName)
    writer.Close()

    // Create request
    req := httptest.NewRequest(http.MethodPost, "/leidmine/lisa", body)
    req.Header.Set("Content-Type", writer.FormDataContentType())
    res, err := app.Test(req, -1)

    // Was the request succesful?
    assert.NoError(t, err)
    assert.Equal(t, http.StatusFound, res.StatusCode)

    // Is the new act in DB?
    var a models.FindingAct
    result := db.Where("finder_name = ? AND wd_act_number = ? and reciever_name = ?",
              act.FinderName, act.WDActNumber, act.RecieverName).First(&a)
    if result.Error != nil {
        t.Error("Could not find given act in DB")
    }

}
```

Joonis 14. Integratsioonitesti näidis.

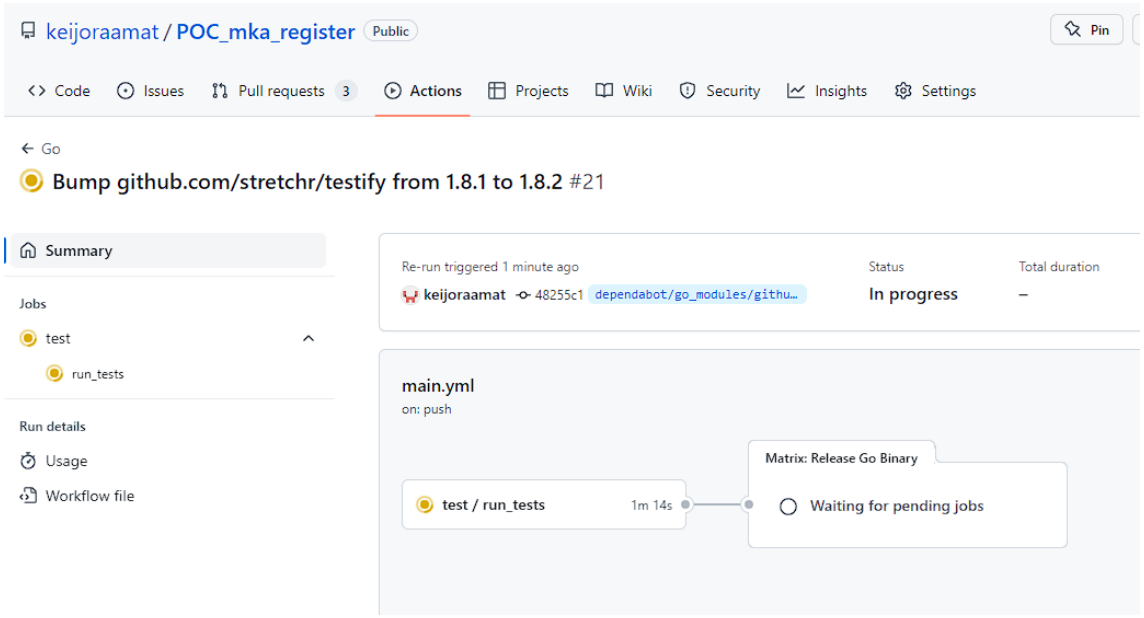
Kõikide testide automaatseks käivitamiseks on koostas autor korduvkasutatava töövoo. Joonisel 15 on kujutatud teste käivitava töövoo konfiguratsioon. Kuna selle voo päästikus, inglise keeles *trigger*, on töövoog (joonisel rida 4 olev märksõna workflow_call) siis on võimalik seda kasutada mitmel juhul – pidevintegratsiooni kui ka pidevedastuse korral.

```yaml
name: Test

on:
  workflow_call:

jobs:

  run_tests:
    runs-on: ubuntu-22.04
    steps:
    - uses: actions/checkout@v3

    - name: Get Go version
      run: |
        echo "GO_VER=`go list -m -f={{.GoVersion}}`" >> $GITHUB_ENV

    - name: Set up Go
      uses: actions/setup-go@v3
      with:
        go-version: ${{ env.GO_VER }}
        cache-dependency-path: go.sum

    - name: Get dependencies
      run: go get .

    - name: Test
      run: go test -v ./...

```

Joonis 15. Automaatselt testide käivitamine.

4.3.2 Pidevedastuse töövoog

Pidevedastuse töövoog kutsub esmalt välja automaattestid. Kui need on edukad ehitatakse lähtekoodist käivitatavad failid erinevatele operatsiooni süsteemidele: Linux, Windows ja Mac OS. Joonisel 16 on fragment GitHub Action töölauast kus on näha, et ehitamise osa ootab testide järgi.



Joonis 16. GitHubi töölaud pidevedastuse töövoost.

5 Lahenduse realiseerimise tulemus

Bakalaurusetöö käigus valmis Muinsuskaitseameti arheoloogiliste leidude registri prototüüp.

Valminud prototüüp täidab kõik seatud mittefunktsionaalsed nõuded – rakendus on kasutaja arvutis ühe kompaktse failina ja ei vaja rohkem arvutialaseid teadmisi kui elementaarsed oskused failide käsitlemisel.

Prototüübil ei ole kõiki funktsionaalsuseid realiseeritud, mis minimaalselt kasutatava tootena kasutaja lugude kaardil märgitud sai. Realiseerimata on aktide koostamine ekspertide ja konservaatoritega kuid neid on tänu prototüübi valmistamise käigus rajatud põhjale kerge lisada.

Prototüüpi demonstreeriti arheoloogia valdkonna inimestele, kellele tundus selle kasutamine lihtne ja ülesehitus hõlpsasti hoomatav.

Kokkuvõtteks saab tõdeda, et selle prototüübi edasi arendus annab Muinsukaitseametile vajaliku töövahendi, mis aitab vähendada korduvaid tegevusi ja seeläbi vähendad töökoormust.

6 Kokkuvõte

Käesoleva lõputöö eesmärgiks oli luua Muinsuskaitseameti arheoloogiliste leidude registri prototüüp. Selle tarbeks uuriti ameti töötajalt ja arheoloogilt mida on vaja täpselt registreerida. Arvesse võeti ka Muinsuskaitseseadusest tulenevad tingimused. Samuti selgitati välja rakenduse kasutajatelt ootused rakendusele.

Lõputöö lahendatava probleemi, arheoloogiliste leidude registreerimise ja haldamise suur töömahukus, on leidnud lahenduse arendatud registri prototüübi näol.

Töös on kasutatud metoodikad: Saaty AHP, pidevintegratsioon ja kasutaja lugude kaart.

Valminud registri prototüüp ei ole oma funktsionaalsutega katmas täielikult kõiki kasutajate vajadusi. Kuid järgnevate funktsionaalsuste lisamine on lihtne ülesanne ega nõua nii palju vaeva kui esmase prototüübi valmistamine. Samuti on prototüüp lahendatud selliselt, et seda saab integreerida Muinsuskaitseameti Kultuurimälestiste registrisse.

Autor õppis lähemalt tundma programmerimiskeelt Go. Olles varem kasutanud seda keelt vaid mõningate liidestuste loomiseks oli veebirakenduse koostamine omaette väljakutseks. Kuid metoodiline lähenemine tõestas, et probleemidest saab jagu üldistades ja murekohtade täpsemas formuleerimises. Samuti pidi autor korduvalt tunnistama, et automaattestide kirjutamine on küll töömahukas, kuid äärmiselt hädavajalik, sest vaid nii suutis autor tuvastada mõningaid eksimusi koodis ja neid parandada.

Kasutatud kirjandus

[1] „Arheoloogiast | Arheoloogia.ee“. https://arheoloogia.ee/arheoloogiast/ (vaadatud 21. märts 2023).

[2] „Muinsuskaitseseadus–Riigi Teataja“. https://www.riigiteataja.ee/akt/MuKS (vaadatud 21. märts 2023).

[3] „Arheoloogilise leiu leidjale leiuautasu määramise kord, otsinguvahendi loaga ja otsinguvahendi kasutamisega seotud nõuded ning nõuded täienduskoolituse läbiviijale–Riigi Teataja“. https://www.riigiteataja.ee/akt/116052019002 (vaadatud 19. aprill 2023).

[4] „MetDect“, *Tallinna Ülikool*, 16. juuli 2021. https://www.tlu.ee/ht/teadusteadusprojektid/metdect (vaadatud 21. aprill 2023).

[5] Maa-amet, „Integreeritav aadressiotsing In-ADS“. https://geoportaal.maaamet.ee/est/Teenused/Integreeritav-aadressiotsing-In-ADS-p504.html (vaadatud 23. aprill 2023).

[6] J. Patton, *User Story Mapping*. O’Reilly Media, 2014. Vaadatud: 15. aprill 2023. [Online]. Available at: https://learning.oreilly.com/library/view/user-story-mapping/9781491904893/

[7] „Agiilse tarkvaraarenduse manifest“. https://agilemanifesto.org/iso/et/manifesto.html (vaadatud 22. aprill 2023).

[8] R. W. Saaty, „The analytic hierarchy process—what it is and how it is used“, *Math. Model.*, kd 9, nr 3, lk 161–176, 1987, doi: https://doi.org/10.1016/0270-0255(87)90473-8.

[9] R. Pereira *et al.*, „Ranking programming languages by energy efficiency“, *Sci. Comput. Program.*, kd 205, lk 102609, mai 2021, doi: 10.1016/j.scico.2021.102609.

[10] „Gin Web Framework“, *Gin Web Framework*. https://gin-gonic.com/ (vaadatud 23. aprill 2023).

[11] kataras, „Iris Web Framework Documentation“. https://www.iris-go.com/docs (vaadatud 23. aprill 2023).

[12] „👋 Welcome | Fiber“, 18. aprill 2023. https://docs.gofiber.io/ (vaadatud 23. aprill 2023).

[13] „SQLite Home Page“. https://www.sqlite.org/index.html (vaadatud 23. aprill 2023).

[14] C. Ireland, D. Bowers, M. Newton, ja K. Waugh, „Understanding object-relational mapping: A framework based approach“, *Int. Lournal Adv. Softw.*, kd 2, nr 2, 2009.

[15] Jinzhu, „GORM“, *GORM*, 21. aprill 2023. https://gorm.io/index.html (vaadatud 23. aprill 2023).

[16] R. C. Martin, *Clean Architecture: A Craftsman’s Guide to Software Structure and Design.* 2017. [Online]. Available at: https://learning.oreilly.com/library/view/clean-architecture-a/9780134494272/ch15.xhtml#:-:text=The%20primary%20purpose,maximize%20programmer%20productivity.

[17] R. Morales-Chaparro, M. Linaje, J. Preciado, ja F. Sánchez-Figueroa, „MVC web design patterns and rich internet applications“, *Proc. Jorn. Ing. Softw. Bases Datos*, lk 39–46, 2007.
[18] M. O. contributors Jacob Thornton, and Bootstrap, „Bootstrap“. https://getbootstrap.com/ (vaadatud 20. aprill 2023).
[19] „Continuous Integration“, *martinfowler.com*. https://martinfowler.com/articles/continuousIntegration.html (vaadatud 9. mai 2023).

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

Mina, Keijo Raamat

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

15.04.2023

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

Lisa 2 – AHP kriteeriumite ja alternatiivide kaalude arvutus

Kriteerium	Ressursi-nõudlikus	Portatiivsus	Kogemus
Ressursi-nõudlikus	1	2	3
Portatiivsus	1/2	1	2
Kogemus	1/3	1/2	1
Kokku	1,833	3,500	6,000

Normaliseeritud maatriks

Kogemus	Ressursi-nõudlikus	Portatiivsus	Kogemus		Kriteeriumi kaalud W
Ressursi-nõudlikus	0,545	0,571	0,500		0,539
Portatiivsus	0,273	0,286	0,333		0,297
Kogemus	0,182	0,143	0,167		0,164
				Kokku	1,000

Kaalutud summad Ws		Järjepidevuse vektor
1,625		3,015
0,894		3,008
0,492		3,004
	keskmine, λ	3,009

Järjepidevuse indeks	0,004604		
Juhuslikkuse indeks	0,58		
Järjepidevuse suhe	0,007939	< 0,1	järelikult on kriteeriumite järjestus pidev

Ressursi-nõudlikkus	Java	C#	Python	Go	PHP
Java	1	1	8	1/2	7
C#	1	1	8	1/2	7
Python	1/8	1/8	1	1/9	1/6
Go	2	2	9	1	7
PHP	1/7	1/7	6	1/7	1
Summa	4,268	4,268	32,000	2,254	22,167

Normaliseeritud maatriks

Ressursi-nõudlikkus	Java	C#	Python	Go	PHP	Kriteeriumi kaalud W
Java	0,234	0,234	0,250	0,222	0,316	0,251
C#	0,234	0,234	0,250	0,222	0,316	0,251
Python	0,029	0,029	0,031	0,049	0,008	0,029
Go	0,469	0,469	0,281	0,444	0,316	0,396
PHP	0,033	0,033	0,188	0,063	0,045	0,073
					Kokku	1,000

Kaalutud summad W_S		Järjepidevuse vektor
1,443		5,743
1,443		5,743
0,148		5,053
2,173		5,492
0,377		5,192
	keskmine, λ	5,445

Järjepidevuse indeks	0,111186		
Juhuslikkuse indeks	1,12		
Järjepidevuse suhe	0,099273	< 0,1	järelikult on keelte järjestus pidev

Portatiivsus	Java	C#	Python	Go	PHP
Java	1	1/5	3	1/7	3
C#	5	1	6	1/5	4
Python	1/3	1/6	1	1/9	2
Go	7	5	9	1	9
PHP	1/3	1/4	1/2	1/9	1
Summa	13,667	6,617	19,500	1,565	19,000

Normaliseeritud maatriks

Portatiivsus	Java	C#	Python	Go	PHP	Kriteeriumi kaalud W
Java	0,073	0,030	0,154	0,091	0,158	0,101
C#	0,366	0,151	0,308	0,128	0,211	0,233
Python	0,024	0,025	0,051	0,071	0,105	0,055
Go	0,512	0,756	0,462	0,639	0,474	0,568
PHP	0,024	0,038	0,026	0,071	0,053	0,042
					Kokku	1,000

Kaalutud summad Ws		Järjepidevuse vektor
0,522		5,155
1,354		5,823
0,276		4,974
3,320		5,841
0,225		5,322
	keskmine, λ	5,423

Järjepidevuse indeks	0,105748		
Juhuslikkuse indeks	1,12		
Järjepidevuse suhe	0,094418	< 0,1	järelikult on portatiivsuse järjestus pidev

Kogemus	Java	C#	Python	Go	PHP
Java	1	3	1/3	3	1/3
C#	1/3	1	1/4	1	1/4
Python	3	4	1	4	2
Go	1/3	1	1/4	1	1/4
PHP	3	4	1/2	4	1
Summa	7,667	13,000	2,333	13,000	3,833

Normaliseeritud maatriks

Kogemus	Java	C#	Python	Go	PHP	Kriteeriumi kaalud W
Java	0,130	0,231	0,143	0,231	0,087	0,164
C#	0,043	0,077	0,107	0,077	0,065	0,074
Python	0,391	0,308	0,429	0,308	0,522	0,391
Go	0,043	0,077	0,107	0,077	0,065	0,074
PHP	0,391	0,308	0,214	0,308	0,261	0,296
					Kokku	1,000

Kaalutud summad Ws		Järjepidevuse vektor
0,837		5,094
0,375		5,067
2,069		5,285
0,375		5,067
1,577		5,320
	keskmine, λ	5,166

Järjepidevuse indeks	0,041612		
Juhuslikkuse indeks	1,12		
Järjepidevuse suhe	0,037154	< 0,1	järelikult on kogemuse järjestus pidev