TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut

ITI70LT

Olavi Ojamaa 122196 IAPMM

Ühendatud aadressiandmete haldamine ja otsimine ohutuskriitilises süsteemis

Magistritöö

Juhendajad: Juhan-Peep Ernits

Peeter Lump

Tallinn 2014

Autorideklaratsioon

Olen koostanud antud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud. Käesolevat tööd ei ole varem esitatud kaitsmisele kusagil mujal.

Autor: Olavi Ojamaa

Annotatsioon

Magistritöö eesmärgiks on parandada ja edasi arendada Häirekeskuse töös kasutatava geoandmete rakenduse asukoha andmete haldamise ja otsimise võimekust.

Töö tulemusena valmis rakenduse uus iteratsioon, mis võimaldab andmeid erinevatest allikatest koguda, uuendada ja ühtselt otsida. Geoandmete infosüsteemi uues versioonis parandati märgatavalt nii otsingualgoritmi kui ka andmete struktuuri, mille tulemusena paranes otsingu kiirus ja andmete leitavus. Aeglaste päringute osakaal langes 5% pealt 0,01%-ni ja andmete leitavus paranes 96,4% pealt 99,96%-ni. Paranenud tulemused saavutati põhiliselt otsingu tüübile vastavate indeksite kasutuselevõtu ja tõhusama otsingusõne töötlemise tulemusena.

Samuti loodi töö käigus testsüsteem, mille kaudu kontrollitakse jooksvalt rakenduse korrektset töötamist vastavalt analüüsis püstitatud nõuetele.

Magistritöö on kirjutatud eesti keeles ning sisaldab teksti 60-1 leheküljel, 8 peatükki, 25 joonist ja 5 tabelit.

Abstract

The purpose of this thesis is to improve and further develop the unified location data management system as a part of a safety critical system used by Estonian Rescue Board.

As a result of this thesis a new iteration of the Geodata application was developed. Significant new functionalities include the ability to correctly collect and automatically update location data from different sources and enable a unified search of the locations and addresses. The improved data structure and a new search algorithm yielded better performance for search queries. The percent of slow queries dropped from 5% to 0.01% and the search algorithm is now able to find 99.96% of addresses compared to 96.4% of the previous iteration. The improved results are mainly achieved by using address type specific indexes and a more efficient way of processing search terms.

To further the reliability of the system an integrated test functionality was developed to monitor the performance of the application continuously.

The thesis is written in Estonian and contains 60 pages of text, 8 chapters, 25 figures, and 5 tables.

Lühendite ja mõistete sõnastik

HK	Häirekeskus
SMIT	Siseministeeriumi infotehnoloogia- ja arenduskeskus
Asukoht	Objekt, mida süsteemist otsitakse. Asukohad on aadressid, maanteed ja teised reaalselt eksisteerivad objektid, kus HK mõistes võib sündmus aset leida
EHAK	Eesti haldus- ja asustusjaotuse klassifikaator
ADS	Maaameti poolt hallatav aadressiandmete andmekogu
Kasutaja	Infosüsteemi kasutav Häirekeskuse töötaja.
Päringu koht	Otsitud asukoha järjekorra number tagastatud otsingutulemuste hulgas

Sisukord

Autorideklaratsioon 2
Annotatsioon 3
Abstract 4
Lühendite ja mõistete sõnastik 5
Sisukord 6
Jooniste nimekiri 8
1. Sissejuhatus 9
1.1. Töö taust ja eesmärk 10
2. Nõuete analüüs 12
2.1. Asukohtade otsingu nõuded 12
2.2. Asukohtade haldamise nõuded 13
2.3. Süsteemi töökindluse ja jõudluse nõuded 13
3. Asukohtade sisendandmed 14
3.1. Maaameti andmed 14
3.1.1. Maaameti andmete struktuur 15
3.1.2. Maaameti andmete uuendamine 16
3.2. Regio andmed 16
3.2.1. Regio andmete struktuur 17
3.2.2. Regio andmete uuendamine 18
3.3. Sisendandmete analüüs 19
3.4. Nõuded järgnevate liidestujate sisendandmetele 20
4. Asukoha operatiivstruktuur 22
4.1. Operatiivandmete struktuur 22
4.2. Andmete teisendamine ja uuendamine 24

5. Asukohtade otsing....28
5.1. Asukohtade tabeli indekseerimine....28
5.2. Asukohtade otsingu loogika....29
5.3. Asukohtade sorteerimine....35
5.4. Asukohtade loendi esitamine kasutajale....35
6. Asukohtade otsingu testsüsteem....37
6.1. Testsüsteemi kirjeldus....37
6.2. Testsüsteemi tehniline lahendus....38
7. Tulemused....41
7.1. Vana aadressi otsing....41
7.2. Uus aadressi otsing....44
7.3. Vana ja uue otsingu võrdlus....46
8. Kokkuvõte....48
Kasutatud kirjandus....50
Lisa 1 – Süsteemi esialgsed nõuded....52
Nõuded Häirekeskuse infosüsteemi asukoha sisestus süsteemile....52
Vormi täitmise üldnõuded....52
Aadress tüüpi asukoha sisestamine....53
Maantee tüüpi asukoha sisestamine....54
Objekt tüüpi asukoha sisestamine....55
Koordinaat tüüpi asukoha sisestamine....56
Märkuste sisestamine....56
Asukohtade otsingu süsteemi lisanõuded....57
Lisa 2 – Asukohtade uuendamise lähtekood....58
Lisa 3 – Aadresside tasemete tähendused....60

Jooniste nimekiri

Joonis 1 Maaameti andmete struktuur 15
Joonis 2 Regio andmete struktuur 17
Joonis 3 Operatiivandmete strukuur 22
Joonis 4 Asukoht tabeli taseme nimetuste täitmine 25
Joonis 5 Korteri välja teisendamine 25
Joonis 6 Indeksid 28
Joonis 7 Numbriliste sõnede teisendamine 29
Joonis 8 Otsinguks kasutava teksti koostamine 31
Joonis 9 EHAK piirang 32
Joonis 10 EHAK kitsenduse otsimine 32
Joonis 11 Lähiaadressi otsing 32
Joonis 12 EHAK piirang lähiaadressi otsimisel 33
Joonis 13 EHAK taseme aadresside otsing 34
Joonis 14 Koordinaadi ja raadiuse piirang 34
Joonis 15 Asukohtade sorteerimine otsingutulemuses 35
Joonis 16 Test otsingusõne koostamine 39
Joonis 17 Otsingu kontrollimine 39
Joonis 18 Vana otsingu kiiruse jaotuvus 42
Joonis 19 Vana otsingu kohtade jaotuvus 42
Joonis 20 Uue otsingu ajaline jaotuvus 44
Joonis 21 Uue otsingu kohtade jaotavus 45
Joonis 22 Punktide ja komade töötlemine 45
Joonis 23 Otsingusõne töötlemine regulaaravalidsega 46
Joonis 24 Otsingute ajaline võrdlus 47
Joonis 25 Otsingute aadresside leitavuse võrdlus 47

1. Sissejuhatus

Siseministeeriumi infotehnoloogia- ja arenduskeskus (edaspidi SMIT) on Siseministeeriumi hallatav riigiasutus, mis tegeleb ministeeriumi valitsemisala IKT teenuste haldamise ja arendamisega. Üheks laialdast kasutust leidvaks komponendiks SMIT-i poolt hallatavates süsteemides on geoandmete alamsüsteem (võrgurakendus), mis on läbinud mitmeid arendusiteratsioone.

Geoandmete rakenduse funktsionaalsuse juurde kuulub aadresside, maanteede, objektide otsing ja uuendamine ning pöördgeokodeerimine. Lisaks sellele vahendab rakendus teekonnaotsinguid ja geoandmete andmebaasis hoitakse kaardikihtide andmeid.

Geoandmete rakendus on ehitatud Grails [1] platvormil. Grails on Groovy [2] programmeerimiskeelt kasutav veebirakenduse platvorm, mis on loodud eesmärgiga muuta arendusprotsess efektiivsemaks. Seda eesmärki püütakse saavutada eelkõige kasutades põhimõtet *programmeerimine tava järgi* [3]. Grailsi eelisteks on tugev integreeritus Java programmerimiskeelega, sisseehitatud Hibernate [4] ja Spring [5] tugi, kasutajaliidese automaatse genereerimise võimalus ja vähene konfigureerimise vajadus. [6]

Andmeid hoitakse rakenduses PostgreSQL andmebaasis, seejuures otsingud toimuvad kasutades *Searchable pluginat* [7]. Selle tarkvaratüki kaudu on võimalik ära kasutada *Lucene* indeksite [8] ja *Compass* otsingu [9] võimalusi kiireks tekstipõhiseks päringuteks.

Geoandmete rakenduse funktsionaalsust kasutavad Päästeameti, Häirekeskuse (edaspidi HK) ja tulevikus ka korrakaitse asutuste infosüsteemidesse kuuluvad rakendused *REST* päringute ja *SOAP* veebiteenuste kaudu.

1.1. Töö taust ja eesmärk

Magistritöö tulemusena täiendatakse SMIT-i poolt hallatava Häirekeskuse infosüsteemi kuuluvat geoandmete ohutuskriitilist rakendust. Töö keskendub aadressiandmete haldamise ja otsimise parandamisele ning samuti valmib rakenduse korrektsust ja töökindlust kontrollida võimaldav testsüsteem.

Päästeameti ja Häirekeskuse töös on väga oluline kiiresti otsida kõikvõimalikke asukohti Eesti riigis. Senimaani on erinevat tüüpi asukohtade otsimine toimunud mitme päringu kaudu, see tähendab et aadresse, objekte ja maanteid on käsitletud eraldiseisvate objektidena. Käesoleva töö raames töötatakse välja ühtne andmestruktuur nende andmete haldamiseks ja otsimiseks. Sündmuse asukohta süsteemist otsides ei pea HK töötaja enam mõtlema, millise asukoha tüübiga on tegu.

Uus andmete ülesehitus eraldab erinevatest allikatest pärinevad töötlemata asukohad operatiivstruktuurist, mille põhjal otsing toimub. Välja töötatakse uus andmemudel ja täiendatakse liidestust väliste süsteemidega. Selle tulemusena paigutatakse erineval kujul olevad aadressid ning loodusesse ja avalikku ruumi kuuluvad objektid ühte tabelisse. Selline lähenemine võimaldab lisaks juba mainitud efektiivsemale otsimisele ka andmeid paremini hallata - asukohtade lisamine välistest süsteemidest operatiivstruktuuri toimub kindla kokkuleppe järgi. Lisaks lubab selline lahendus käsitsi muuta ja lisada asukohti.

Töö tulemusena parandatakse ka asukohtade otsingualgoritmi. Lisaks olemasolevas loogikas olevate vigade parandamisele ja erinevat tüüpi objektide arvestamisega tulemuste leidmisel, lisatakse võimalus piirata tulemusi mingi kindla asukohapunkti ja raadiuse järgi. Selle funktsionaalsuse mõte on piirata otsingu tulemusi Häirekeskusesse helistanud abivajaja kõneasukoha järgi. Uus lahendus võimaldab otsida ka kehtetute, see tähendab vananenud ja enam mitte kasutuses olevate andmete hulgast. Selline võimekus tuleb kasuks, kui abivajaja mäletab mingi asukoha vana nime, mis enam tegelikult kehtiv pole.

Kuna asukoha otsingu korrektne töötamine on hädavajalik osa hädaabikõnede teenindamisel, siis on oluline garanteerida rakenduse töökindlus. Magistritöö raames luuakse testsüsteem, mis kontrollib kõikide asukohtade leitavust ja uuenemist. Samuti võimaldab testsüsteem koguda statistikat raskesti otsitavate või vigaste andmete kohta

ning kontrollida süsteemi järjepidevalt. Kuigi aadressiandmed võivad tunduda staatilistena, tehakse neisse tihti täiendusi. Näiteks lisatakse uusi tänavaid, parandatakse olemasolevaid aadresse või muudetakse neid lähtuvalt muutunud halduskorraldusele valdade liitumise tulemusel. Loodav süsteem võimaldab pärast andmete uuenemist kontrollida, et kõik, nii muutmata kui ka lisandunud või muudetud asukohad on leitavad.

2. Nõuete analüüs

Esialgsed süsteemile püstitatud nõuded on välja toodud magistritöö lisas 1. See analüüs pärineb projekti ametlikust dokumentatsioonist ja seda täiendas Peeter Lump magistritöö „Ohutuskriitilise asukohtade otsingusüsteemi arendus“ (2013) [9] raames.

Olemasoleva geoandmete rakenduse kasutamise tulemusena on esile kerkinud uued soovid ja senise lahenduse kitsaskohad. Otsingute puhul on suund liikuda edasi ühtse andmestruktuuri ja täpsustavate päringute suunas. Samuti on kvaliteetse aadressiotsingu teenuse pakkumiseks oluline hoida andmeid ajakohasena. Geoandmete rakendus peab olema võimeline asukohti automaatselt uuendama ning lubama kasutajaliidese kaudu andmeid hallata. See võimekus oli olemas ka varem, kuid esialgse lahenduse puuduse tulemusena ei olnud alati kõik aadressid SMIT-i mõistes kehtivad ja süsteemist leitavad. Seega on oluline, et andmete korrektsus oleks kontrollitav ja tagatud igal ajahetkel. Järgnevalt on välja toodud nõuded, mis on eelnevaga võrreldes lisandunud või muutunud.

Järgnevates alampeatükkides on välja toodud muutunud või lisandunud nõuded.

2.1. Asukohtade otsingu nõuded

1. Kasutaja saab otsida erinevat tüüpi asukohti ühelt sisestusväljalt.
2. Asukohaotsingut on võimalik piirata koordinaadi ja raadiuse järgi.
3. Kui HK-sse helistanud inimene on positsioneeritud, siis süsteem pakub otsingutulemusi sõltuvalt selle kõne asukohast (vaikimisi on see valik väljas).
4. Kasutajal on võimalik piirata otsingutulemusi EHAK (Eesti haldus- ja asustusjaotuse klassifikaator) [7] järgi, kasutades selleks haldusjaotuse nimetust. Näiteks kui otsida „Tartu linn Pargi tn“, siis otsing peab toimuma Tartu linna piiranguga.
5. Kasutajal on võimalik otsida asukohta kehtetute aadresside hulgast (vaikimisi väljas).
6. Maja numbri kuvamise piirang „Sõpruse pst 2“ ei tagasta sõpruse pst 20, 21, 220 jne.

2.2. Asukohtade haldamise nõuded

1. Maaameti aadresse uuendatakse perioodiliselt üle x-tee ADS andmekogust [8].
2. Kohanimeregistri andmeid uuendatakse andmete tarne tulemusena kasutajaliidese kaudu.
3. Regio maanteid uuendatakse andmete tarne tulemusena kasutajaliidese kaudu.
4. Regio objekte uuendatakse andmete tarne tulemusena kasutajaliidese kaudu.
5. Välistelt partneritelt saadavad andmed teisendatakse ühtsele kujule (ptk. 4).
6. Kasutajaliidese kaudu on võimalik asukohtade andmeid muuta.
7. Iga asukoha kohta on võimalik sisestada täpsustusi vastavale andmeveerule (kasutajale kuvatakse asukoha valimisel see täpsustus – vajalik näiteks raskesti ligipääsevate objektide puhul).

2.3. Süsteemi töökindluse ja jõudluse nõuded

1. Asukohtade otsingus on võimalik otsingusõnega piirata otsingut nii, et otsitav asukoht on loendi alguses esimese 20 kirje hulgas. Kasutajale kuvatakse kuni 50 otsingu vastet. (Otsingusõne on lähiaadressiga aadressi puhul lähiaadress ja EHAK viimane tase – selleks on üldiselt vald või linn, kuhu aadress kuulub. Ilma lähiaadressita asukoha puhul on otsingusõneks EHAK viimane tase).
2. Testsüsteem kontrollib üle kõikide asukohtade leitavuse vastavalt nõudele 2.3.1 ja toob välja probleemsed andmed.
3. Testsüsteem mõõdab iga aadressi leidmiseks kuluvat aega.
4. Pärast perioodilist uuendamist ja andmete muutmist kontrollib rakendus muutunud asukohtade leitavust. Probleemide korral saadetakse e-mail administraatorile ja probleemsed asukohad kuvatakse kasutajaliidese abil.
5. Kasutajaliidese kaudu on võimalik jälgida asukohtade muutuste logi.

3. Asukohtade sisendandmed

Geoandmete rakendus kogub asukohtade andmeid mitmest allikast. Need andmed on erineva formaadi ja sisulise tähendusega. Sisendandmete analüüsi tulemusena on võimalik eristada oluline informatsioon ebaolulisest ja saavutada arusaam, millise täpsusastme ja võimalustega saab hiljem operatiivstruktuuri ning sellega seotud otsingualgoritmi üles ehitada.

3.1. Maaameti andmed

Algandmed saadakse Maaametist [10] kindla kuu algusseisuga ftp serverist kokkulepitud struktuuriga csv failide kujul. Seejärel toimub nende andmete perioodiline uuendamine üle X-tee.

Senimaani on Maaameti andmetest kasutatud ainult nende poolt pakutud aadresse (1,2 miljonit kehtivat kirjet). Magistritöö käigus lisanduvad ka Kohanimeregistri [11] andmed (45 tuhat kirjet).

Maaameti ADS andmekogust pärinevatest tabelitest on geoandmete rakenduse jaoks olulised *aadressiobjekt*, *aadressid*, *aadress_komponendid* ja *adob_aadress*.

Aadressiobjekt on maaga seotud objekt, millele on määratud aadress või millele aadressi määramise kohustus või võimalus tuleneb õigusaktist. Aadressiobjektiks on maaga püsivalt seotud objekt looduses, millel on ruumikuju ja mille peal või sees saab elada, töötada, midagi hoiustada, mida mööda liikuda jms. seoses olla ja sellest tulenevalt on vajadus sellele kohale või objektile (ruumi piirkonnale) osundada inimkeeles harjumuspärasel viisil ehk aadressiga.

Aadressiobjektil on alati olemas vähemalt üks aadress, kuid leidub ka mitme aadressiga aadressiobjekte (näiteks tänavate ristumise nurkadel asuvad ehitised), millele on määratud paralleelaadressid. *Aadressiobjektide* elutsükli kujutamine infosüsteemides on heas kooskõlas neile vastavate reaalsete objektide tekkimise ja kadumisega looduses.

Aadressiks on inimkeeles loetav ja mõistetav tekst, mis osundab aadressiobjektile. Ühele aadressile saab vastata mitu aadressiobjekti, sest mitu objekti looduses (näiteks mitu hoonet) võivad omada sama aadressi. Seega väljendades asukohta vaid aadressi täpsusega ei pruugi me üheselt objekti (nt hoonet) looduses määratleda (näiteks maal asuv elamu koos kõrvalhoonetega omavad kõik ühesugust nime, linnas asuv elamu koos kuuridega omavad sama tänavanimest ja numbrist koosnevat lähiaadressi). [13]

Aadressi ja objekti vaheline mitu-mitmele seos kajastub tabelis *adob_aadress*.

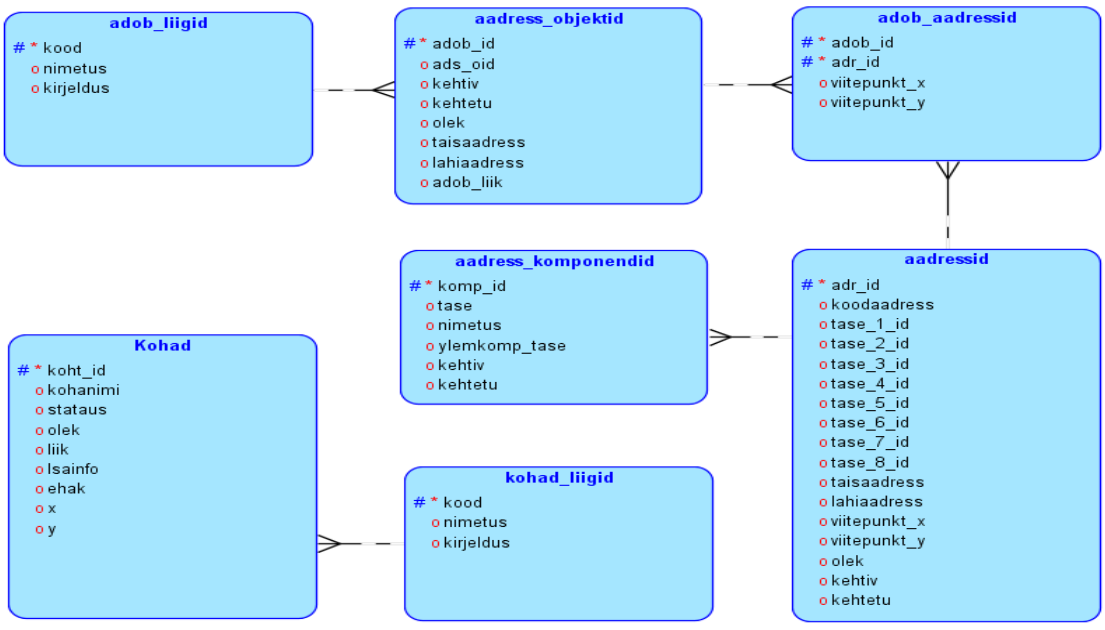
Iga aadress koosneb *aadressi komponentidest*, mis on jaotunud 8 astmelisse hierarhiasse. Aadressi tasemete tähendus on välja toodud Lisas 3. Aadressikomponendid on aadressi osad, millest aadressid kokku pannakse.

Maaameti aadressiandmete detailsem kirjeldus on antud ADS spetsifikatsioonis [14].

Kohanimeregistrist saadakse erinevat liiki objektid (tabel *„Kohad“*), mille Maaamet on mitmest allikatest omandatud. Suure osa andmetest moodustavad mitteametlikud kohanimed ja looduslikud objektid (järved, jõed, mäed).

3.1.1. Maaameti andmete struktuur

Järgneval joonisel (Joonis 1) on kujutatud need tabelid ja veerud, mis maaameti andmete hulgast kasutusele võetakse.



Joonis 1 Maaameti andmete struktuur

3.1.2. Maaameti andmete uuendamine

Aadressiandmed on pidevalt ajas muutuvad ja nende ajakohasena hoidmine on oluline võimalikult täpse asukoha otsingu tagamisel. Aadresside, objektide ja nendevaheliste seoste uuendamiseks kasutatakse Maaameti X-tee teenuseid.

Senimaani on uuendatud ainult *aadressid* tabelit, kuid ainult sellest ei piisa et tagada andmete terviklikus (ptk 3.3). Rakenduse uues versioonis kasutatakse objektide uuendamiseks teenust ADSobjmuudatused.v3 (spetsifikatsioon [14] ptk 5.7), aadresside uuendamiseks ADSaadrmuudatused.v3 (spetsifikatsioon [14] ptk 5.8) ja nendevaheliste seoste muudatuste jaoks ADSobjaadrmuudatused.v2 (spetsifikatsioon [14] ptk 5.9).

Need teenused edastavad kõik andmetega seotud muudatused alates esialgsest impordist (milleks on mingi kindla kuu esimene päev) ning selle kaudu tagatakse, et SMIT-is olevad aadressiandmed vastavad samale seisule, mis on Maaametis.

Sisuliselt küsitakse teenuste kaudu tabelitega seotud andmete lisamise, muutmise ja kustutamise operatsioone, mis seejärel geoandmete rakenduse andmebaasis täidetakse. Muudatuste ajalist järjekorda ja viimase muudatuse aega peetakse meeles logide id kaudu (küsitakse muudatused alates viimasest kirjest). Muudatuste alguse saab määrata ka kuupäevaliselt, seega kindla ajalise seisuga andmete muudatusi saab teenuste kaudu küsida alates impordile vastavast kuupäevast.

Kohad tabeli uuendamine toimub andmeimpordi kaudu. Kasutajaliidese kaudu laetakse üles kohanimede väljavõte. Uusi andmeid võrreldakse vana seisuga ja uuendatakse vastavalt.

3.2. Regio andmed

Objektid on erinevat tüüpi asukohad, millele on antud nimi ja ruumipunkt. Objekt võib asuda mingil kindlal aadressil (poed, koolid, restoranid), aga võivad olla ka aadressidest sõltumatud (bussipeatused, looduslikud objektid). HK mõistes on objektid orientiirid, mille järgi inimesed oma asukohta oskavad määratleda. Sündmusi registreerides ei osata tihti öelda täpset aadressi, kuid teatakse nimetada näiteks lähedal asuvat bussipeatust või poodi.

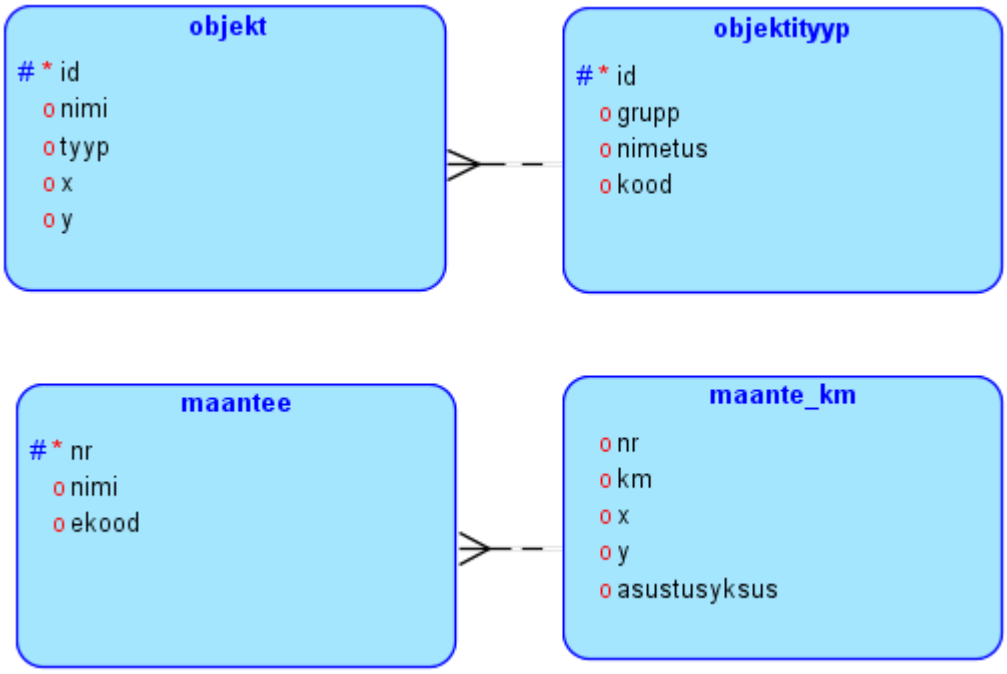
Selliste erinevat liiki asukohad pärinevad Regio poolt hallatavatest objektidest, mis on jaotatud mitmesse suuremasse gruppi - näiteks majutus, meelelahutus, sport, toitlustus. Neid andmeid tarnitakse SMIT-ile lepingu alusel regulaarse ajaperioodi järel shape [15] ja csv failide kujul.

Lisaks objektidele tarnitakse Regiolt maanteed ja maanteede kilomeetripostid. Maanteedel toimunud õnnetused antakse asukoha mõttes tihti kilomeetriposti täpsusega. Lisaks sellele on võimalik nende kaudu määrata ligikaudu inimese asukoht kui ta väidab, et on teatud kilomeetri kaugusel mingist linnast.

Regio maanteed võetakse senimaani paralleelselt kasutuses olnud Maanteeameti andmete vastu kasutusele seetõttu, et erinevalt Maanteeameti teelõikudest on Regio andmed kilomeetriposti täpsusega. Sellisel kujul maanteed ja nende punktid sobivad oma täpsuse tõttu paremini sündmuse registreerimise kohana.

3.2.1. Regio andmete struktuur

Andmete struktuur koosneb *objekt* ja *objektityyp* tabelitest (Joonis 2)



Joonis 2 Regio andmete struktuur

3.2.2. Regio andmete uuendamine

Senimaani on objektide uuenemine toimunud käsitsi – .shp failidest saadakse andmed, mis sql skripti abil baasi laetakse. Seejärel on vaja uuesti luua indeksid, et kasutajale uuenenud objektid ja maanteed leitavad oleks.

Uue lahenduse korral laetakse rakenduse kasutajaliidese kaudu üles shape või csv failid, mis seejärel vastavad andmebaasi kirjed uuendab.

Täisautomaatse uuendamise kasutuselevõttu pole ette näha, sest see nõuaks lisaarendust välistelt partneritelt

3.3. Sisendandmete analüüs

Aadressiandmete korrektsuse tagamisel lähtutakse aadressiobjekti elutsüklist. Kuna just aadressiobjektid, mitte aadressid, on reaalselt eksisteerivad kohad Maaameti mõistes, siis nende olemasolu järgi peab toimima otsinguandmete kehtivus. Eelmises geoandmete rakenduse versioonis lähtuti ainult aadressidest ja nende kehtivusest. Selline lähenemine tekitas aga olukorra, kus mingil ajahetkel võivad süsteemis mitteleitavad olla tegelikult reaalselt olemasolevad asukohad.

Näiteks valdade ühinemise käigus muutuvad vanad aadressid kehtetuks ja mõne aja pärast tekivad asemele uued kehtivad aadressid. Samal ajal on aga kehtetu aadressiga seotud objekt endiselt aktiivne. Seetõttu on vaja aadresside kehtivuse juures kontrollida, kas sellega on veel seotud mõni kehtiv objekt. Kui aadress on määratud kustutamisele ja pole enam ühegi objektiga seotud, võib selle kehtetuks märkida. See on ka põhjuseks, miks geoandmete rakendus peab aadressi, aadressobjekti ja nende vahelist seost säilitama ja ajakohasena hoidma.

Eelmise lahenduse korral sooritati tekstilisi otsinguid täisaadressi ja lähiaadressi väljade järgi. Sellise struktuuri järgi oli raske üles leida selliseid aadresse, mille lähiaadressis olev sõne sisaldub ka aadressi esimeses pooles. Näiteks otsing ’harju tänav’ tagastas kõik tänavad Harju maakonnas või ’mustamäe tee’ kõik teed Mustamäe linnaosas. Selle vältimiseks tuleb eraldada EHAK taseme ja lähiaadressi otsing ning otsingualgoritmi juures kasutusele võtta märksõnad (populaarsemad aadresside mõistes tähendust omavad sõnad – maantee, puiestee, tänav jne).

Teine probleem kahe välja otsingu puhul on see, et otsides märksõna ’harju’ järgi, tagastatakse kõik harju maakonnas olevad vallad, külad, linnad. See ilmselt pole soovitud tulemus ja selle parandamiseks on vaja otsida märksõnade järgi eraldi esimese kolme taseme hulgast.

Tugev argument loomaks eraldi indeksid esimese kolme taseme jaoks on ka see, et neid on vähe võrreldes aadresside koguhulgaga. Pole mõtet otsida maakonda 1,2 miljoni aadressi hulgast. Aadress_komponent tabeli abil saab iga taseme numbri kohta luua selle tekstilise nimetuse. Aadresside hulk tasemete järgi on välja toodud järgnevas tabelis 1.

Tabel 1 Aadresside tasemed

Taseme nr	1	2	3	4	5	6	7	8
Aadresside koguarv	16	215	4683	741	16878	350764	250780	584529

Senimaani on toimunud tänavate ja majade otsingud lähiaadressi veeru järgi, mis on täidetud umbes 1,2 miljonil aadressil (v.a. EHAK aadressid). See tähendab, et otsides mingit kindlat tänavat, tehakse päring üle 1,2 miljoni rea, kuigi tänavaid on ainult ligikaudu 17 tuhat. Seega oleks mõistlik lähiaadressi põhjal luua mitu indeksit, mis vastavad mingile kindlale kirjakujule (sisaldavad ainult tähti, sisaldab numbrit, on korteri tunnusega).

Maaameti *kohad* andmetes on olemas objekti nimi, koordinaadid ja EHAK kood, mis viitab millisesse haldusjaotusesse see koht kuulub. Need andmed on piisavad, et integreerida kohad samalaadsesse struktuuri, nagu on *aadressid* tabelis.

Samad andmed on olemas ka Regio objektide ja maanteede andmete puhul, kuid EHAK kood tuleb küsida eraldi koordinaadi järgi Maaameti X-tee teenuse ADSaadrotsing.v1 kaudu, mis tagastab koordinaadi järgi EHAK kuuluvuse.

3.4. Nõuded järgnevate liidestujate sisendandmetele

Tulevikus lisanduvate andmete osas on kohustuslikeks väljadeks nimetus ja koordinaadid. Lisaks võiks need andmed olla klassifitseeritud selliselt, et hiljem võimaldada tüübi järgi otsingut.

Kuna andmed hakkavad tulevikus lisanduma paljudest erinevatest allikatest, siis operatiivstruktuuriga liidestumisel on oluline neid eraldada mõnest teisest allikast pärinevatest andmetest. Selle jaoks on vaja, et välisest allikast saadakse andmetega kaasa ka selle identifikaator, mis püsib objekti eluea jooksul muutumatu.

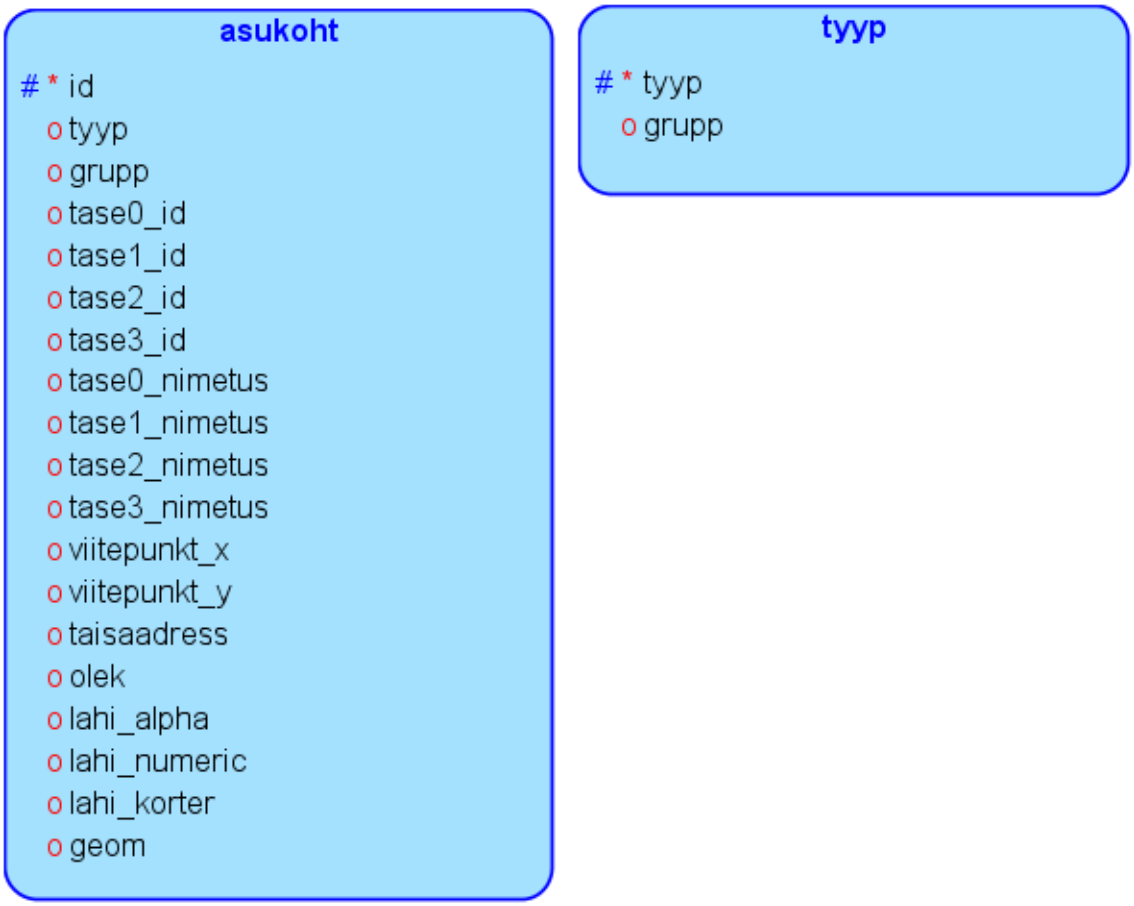
Erinevatest kohtades saadud andmed hoitakse eraldi tabelites ja neil võivad olla korduvad *id* numbrid. Kui andmed teisendatakse operatiivstruktuuri, siis saavad andmed identifikaatori ette allika tunnuse. Näiteks Maaameti aadresside puhul kasutatavad

adr_id-d ette tähekombinatsiooni *ADR*. Selle lahenduse puhul ei pea geoandmete rakendus ise identifikaatoreid genereerima, mistõttu on andmete uuendamine ja terviklikkuse tagamine üle mitme keskkonna lihtsustatud.

4. Asukoha operatiivstruktuur

Eelmises peatükis käsitletud geoandmed teisendatakse ühtseks andmetabeliks. Selle tabeli ülesehitus peab võimaldama efektiivset asukohtade otsingut ja andmete haldamise protsessi vastavalt peatükis 2 kirjeldatud nõuetele. Järgnevates alampeatükkides põhjendatakse, kuidas selline struktuur välja kujunes.

4.1. Operatiivandmete struktuur



Joonis 3 Operatiivandmete strukuur

Asukoht tabeli (Joonis 3) kirje *id* moodustab algallikast pärinev identifikaator, millele on ette lisatud kolmekohaline pärinevuse tunnus. Selle abil on lihtsasti võimalik iga kirjet seostada allikaga ja seetõttu kergendada andmete uuendamist vastavalt peatükis 2.2 olevatele nõuetele.

Tüüp välja kasutamine võimaldab tulevikus pärida asukohti kindla liigi järgi. Maaameti aadressid on kõik ühte tüüpi aga Regiost pärinevad objektid on jagatud kategooriatesse, mida võib hiljem kasutada tulemuste filtreerimiseks. Näiteks tüübi piirangu järgi saaks leida üles kõik sularahaautomaadid Tallinna Kristiine linnaosas. Mitu tüüpi võivad moodustada grupi, näiteks mälestusmärgid, looduslikud objektid, transport. Tüübid on kirjeldatud ka eraldi tabelis ja need lisatakse käsitsi. Praeguses lahenduses hakkavad nende alla kuuluma Regio objektide ja Maaameti kohtade tüübid.

Taseme id-d viitavad, millistest komponentidest aadressid koosnevad. Need andmed on vajalikud otsingutulemuste piirkonna järgi filtreerimise ja sorteerimise jaoks (nõue 2.1.4). Näiteks otsides kõiki aadresse, millel on *tase1_id* 37, piiratakse otsing Harju maakonna täpsuseks. Tasemed 1-3 pärinevad Maaametist (Maakond, vald, linn, küla). *Tase0_id* on lisatud sellel põhjusel, et tulevikus lisanduvad lisaks Eesti asukohtadele ka Läti aadressid.

Esimese kolme taseme nimetused on eraldi veeruna välja toodud selleks, et võimaldada eraldi iga taseme järgi otsimist. Selline tasemete lahku löömine võimaldab otsida näiteks kindlat valda, ilma et tulemuste sisse satuks ekslikke ridu. Endise ainult *tasemete id* ja *täisaadressi* järgi otsingu puhul seisnes probleem selles, et otsides mingi kindla sõna järgi polnud võimalik piirata just konkreetsesse tasemesse kuuluvaid aadresse. Näiteks otsides sõna „Harju“ järgi, tagastatakse Harju maakond ja ka kõik teised vallad, linnad, külad, kus on täisaadressi esimese poole sees sõna Harju. Neid teisi valetulemusi on aga keeruline välja sorteerida, sest pole võimalik kindlaks teha kas otsingusõna sisaldus EHAK aadressi esimeses, teises või kolmandas nimetuse osas.

Veeru lahiaadress järgi on moodustatud kolm eraldi atribuuti, mille alusel otsing sooritatakse: *lahi_alpha, lahi_numeric* ja *lahi_korter*. Vana süsteemi puhul otsiti kõiki aadresse *lahiaadress* veeru abil (umbes 1,6 miljonit kirjet). Olenemata sellest, kas tegemist on lihtsa tänava nimetuse, maja või korteriga, otsiti alati ühest suurest indeksist. Uue lahenduse järgi on aadresside täpsustav osa eraldatud kolmeks indeksiks sisuliselt välistava kirjapildi järgi (peatükk 4.2).

Lahi_alpha (umbes 310 000 kirjet). indeksisse kuuluvad kõik sellised lähiaadressid, mis ei sisalda ühtegi numbrit. Sisuliselt kuuluvad siia alla kõik tänavad, väiksemad asukohad ja paljud objektid.

*Lahi_numer*ic (umbes 300 000 kirjet) sisaldab asukohti, mis sisaldavad numbrit, aga ei oma korteri tunnust. Sellisteks andmeteks on enamasti majad ja linnadest või ametlikest tänavatest eemal olevad objektid.

Lahi_korter (umbes 590 000 kirjet) väli täidetakse juhul, kui *lahiaadress* veerg sisaldab korterile viitavat tunnust. Selleks on numbriline string, mis sisaldab korterite või majaosade eraldamise tunnust „-“.

Siinkohal on oluline märkida, et sellist kolmeks sisuliseks osaks eraldamist toetab ka välja töötatud otsingualgoritm. Samade tunnuste alusel, mille põhjal lähiaadress tükeldati, valitakse millisest indeksist otsida.

Uute kasutusele võetud indeksite puhul arvestatakse ka sellega, et koordinaate mitteomavad aadresse ei indekseerita. Taseme nimetuse indeks on täidetud vaid nendel kirjetel, mille puhul on tegemist EHAK aadressiga – oluline võit nii indekseerimisel kui ka otsingu kiirusel. See tähendab et tänavatel ja majadel pole *tase1_nimetus* veerg täidetud. Tänu sellistele muudatustele vähenes indeksite suurus 39% - 726mb vs 445mb.

4.2. Andmete teisendamine ja uuendamine

Asukohtade andmete valdava enamuse moodustavad Maaameti aadressid. Esialgse andmete tarne tulemusena saadakse mingi kindla kuu algusaja seisuga aadresside andmed. SQL skriptidega luuakse struktuur ja kopeeritakse andmed csv failidest. *Aadressid* esialgne seis kantakse *Asukoht* tabelisse üle SQL skriptiga.

Pärast aadresside sisestamist *Asukoht* tabelisse, toimub andmete töötlemine ja teisendamine ülevaatlikult järgnevalt:

1) Aadressi komponentide hulgast leitakse EHAK taseme aadressidele taseme nimetused ja andmebaasis täidetakse *tase1_nimetus*, *tase2_nimetus* ja *tase3_nimetus*.
2) Andemete analüüsi käigus tuvastatud sisulist väärtust mitteomavate aadresside eemaldamine andmebaasist (tänavate lõigud, sisuliselt korduvad kirjed – sama kirjapilt).
3) Andmebaasis korrastatakse täisaadressi ja lähiaadressi kirjapilt (lisatakse tühikud punktide ja komade järele, eemaldatakse üleliigsed tühikud).

4) Andmebaasis täidetakse *lahi_alpha*, *lahi_numeric* ja *lahi_korter* väljad vastavalt lähiaadressi sisule.
5) Andmebaasis täidetakse geomeetria veerg, mille põhjal saab hiljem teostada keskpunkti ja raadiuse piirangu.
6) Rakenduse käivitamisel luuakse Lucene indeksid asukoht tabeli veergude jaoks, mis osalevad asukohtade otsingu päringus.

Taseme nimetuse veerg täidetakse aadresside puhul, mille jaoks see tase on aadressi viimane. Näiteks Harju maakond, Tallinna linn puhul on *tase1_nimetus* täitmata, aga *tase2_nimetus* täidetud. Omakorda kõik Tallinna linna kuuluvad aadressid ei oma *tase2_nimetus* veerul sisu. Joonisel 4 on väljatoodud selle teisendamise SQL laused.

```
update asukoht set tase1_nimetus = (select nimetus from aadress_komponendid
where komp_id = asukoht.tase1_id) where tase2_id is null and tase3_id is null
and
lahiaadress is null and olek = 'K' and viitepunkt_x is not null;
update asukoht set tase2_nimetus = (select nimetus from aadress_komponendid
where komp_id = asukoht.tase2_id) where tase3_id is null and
lahiaadress is null and olek = 'K' and viitepunkt_x is not null;
update asukoht set tase3_nimetus = (select nimetus from aadress_komponendid
where komp_id = asukoht.tase3_id)
where lahiaadress is null and olek = 'K' and viitepunkt_x is not null;
```

Joonis 4 Asukoht tabeli taseme nimetuste täitmine

Lahi_alpha täidetakse selliste aadresside puhul, millel lähiaadress ei sisalda ühtegi numbrit.

Lahi_numeric puhul otsitakse üles read, mis sisaldavat numbrit aga ei sisalda korteri tunnust. See tähendab et lähiaadress ei tohi sisaldada numbrilist sõne, mis sisaldab sidekriipsu.

Lahi_korter puhul leitakse üles kõik korteri tunnustega aadressid. Vastav teisendamise lause on välja toodud joonisel 5. Seejuures tasub märkida, et pole oluline kas nende ülesleitavate aadresside puhul kõik kirjed on tegelikult korterid – oluline on ainult kirjapilt ja erinevate indeksite kasutamise ainus põhjus on otsingu optimeerimine.

```
update asukoht set lahi_korter = lahiaadress where substring(lahiaadress from
'-\S*\d|\d\S*-') is not null and olek = 'K' and viitepunkt_x is not null;
```

Joonis 5 Korteri välja teisendamine

Maaameti X-tee teenuste kaudu hoitakse andmed geoandmete andmebaasis ajakohasena. Pärast aadresside uuendamist iga öö kantakse muudatused üle *Asukoht* tabelisse. Olenevalt aadressi sisust täidetakse asukoht tabelis vastavad veerud, ning pärast uuendusi reindekseeritakse muudatused. Selle protsessi lähtekood on olemas töö Lisas 2.

Kohtade lisamisel ja muutmisel laetakse kasutajaliidese kaudu üles csv fail. Csv faili põhjal luuakse SQL skript, mis tühjendab koht tabeli ja sisestab kõik andmed või lisab ja muudab ainult lisandunud kirjed sõltuvalt sellest, millise valiku rakenduse kasutaja tegi. Seejärel teisendatakse kohanimeregistri andmed Asukoht tabelisse. Koordinaatide järgi leitakse aadressi EHAK osa ja esimese kolme taseme id'd, et hiljem oleks võimalik otsingu juures neile piiranguid rakendada. Seejuures jäätakse andmetest välja sellised asukohad, mis aadresside hulgas on juba olemas.

Regio objektide puhul laetakse samuti üles csv fail ja luuakse SQL skript. Erinevalt aga *Koht* tüüpi andmetest määratakse objektidele aadressi tekst lähiaadressi täpsusega. Kuna paljud sama nimega objektid (pangaautomaat, parkla) asuvad sama linna sees, siis on oluline, et neid oleks võimalik otsida ka näiteks tänava ja ka maja numbri järgi. Selleks et numbrit sisaldavad objektid oleksid leitavad nii maja täpsusega kui ka lihtsalt linna täpsusega otsides, on vajalik need lisada mõlemasse tähestikulisse ja numbrilisse indeksisse.

Maanteede puhul määratakse esimesed kolm taset kilomeetri postidele. See võimaldab otsida näiteks mingi maantee lõiku valla täpsusega. Ka kilomeetripostide puhul on vajalik sisestada maantee nimi koos kilomeetri numbriga mõlemasse indeksisse.

Objektide uuendamine käib perioodiliselt kasutajaliidese kaudu. Kasutaja laeb üles csv faili, milles sisalduvad andmed salvestatakse Objekt tabelisse. Seejärel konverteeritakse andmed Asukoht tabelisse. Sellise üldise põhimõtte järgi uuendatakse ka teisi tulevikus lisanduvaid asukoha sisendandmeid, mille puhul pole välja töötatud automaatset uuendamist.

Teiste andmete puhul, millel ei ole kaasas aadressi, tuleb Asukoht tabelisse konverteerimise käigus üles leida EHAK tasemed. Seda tehakse Maaameti X-tee teenuse kaudu, mis tagastab tasemelise kuuluvuse koordinaadi järgi.

Asukoha lähiaadressi osa moodustatakse objekti nimest ja tüübist ning lisatakse sobivasse indeksisse. Eelnevalt võib olla vajalik uute tüüpide sisestamine. Asukoht *id*-ks määratakse uue objekti id koos vastava eesliitega

Rakenduses on lisaks ettenähtud võimalus käsitsi Asukoht kirjeid muuta. See võimalus on eelkõige mõeldud kiirete andmeparanduste jaoks.

5. Asukohtade otsing

Asukoha otsingu võib liigendada kaheks suuremaks osaks. Kõigepealt töödeldakse otsingusõne, mille käigus selgub otsitava asukoha liik ning samuti tehakse kindlaks otsingule mõjuvad piirangud. Seejärel sooritatakse töödeldud sõne põhjal otsing indeksist ja sorteeritakse tulemus.

Otsingu tehniline teostus on lahendatud loogika osas kasutades *Grails* rakendust, mis on kirjutatud keeles *Groovy*. Asukohtade andmeid hoitakse *PostgreSQL* andmebaasis, kuid kiire otsingu tagamiseks kasutatakse mälus hoitavaid *Lucene* indekseid, millest tehakse päringuid kasutades *Compass* otsingut. Tehnoloogiate tutvustus ja viited on antud peatükis 1.

5.1. Asukohtade tabeli indekseerimine

Lucene indeksisse lisatakse veerud, mille põhjal otsing toimub. Need on tasemete üks kuni kolm taseme nimed ja identifikaatorid, lähiaadressi põhjal moodustatud veerud ja koordinaadid (Joonis 6).

```
static searchable = {
       tase1  index: 'not_analyzed'
       olek index: 'not_analyzed'
       tase1Nimetus index: 'analyzed', converter: 'aadressConverter'
       tase2Nimetus index: 'analyzed', converter: 'aadressConverter'
       tase3Nimetus index: 'analyzed', converter: 'aadressConverter'
       lahiNumeric index: 'analyzed', converter: 'aadressConverter'
       lahiKorter index: 'analyzed', converter: 'aadressConverter'
       taisaadress index: 'analyzed', converter: 'aadressConverter'
       lahiAlpha index: 'analyzed', converter: 'aadressConverter'
       tase2 nullValue: -1, index: 'not_analyzed'
       tase3 nullValue: -1, index: 'not_analyzed'
       koordX nullValue: -1, index: 'not_analyzed'
       koordY nullValue: -1, index: 'not_analyzed'
}
```

Joonis 6 Indeksid

Tasemete, oleku ja koordinaatide indekseerimiseks pole vaja analüsaatorit kasutada (*not_analyzed*), see tähendab et andmeid ei töödelda vaid lisatakse otse indeksisse.

Asukohtade tekstiliste väärtuste puhul aga kasutatakse enda loodud analüsaatorit, et täita nõuet 2.1.6. Kuna indeksist otsitakse märksõnade järgi iga sõne algusest alates, siis töötlemata numbrilised sõned põhjustaksid suure hulga soovimatute asukohtade tagastamise. Otsides „Sõpruse pst 2“ tagastaks ka „Sõpruse pst 20“, „Sõpruse pst 21“ jne. Üks võimalus selle probleemi lahendamiseks on polsterdada numbrid prefiks nullidega - näiteks „Vilde tee 68-31“ viia indeksis kujule „Vilde tee 00068-00031“. Samamoodi tuleb hiljem töödelda ka otsingusõne, et päringuid korrektselt toimiks.

Sellise teisendamisesüsteemi loomiseks kasutatakse Lucene indeksite konverterit, mis töötleb indeksis olevaid kirjeid eelmises lõigus mainitud põhimõtte järgi. Teisendamise funktsiooni lähtekood on Joonisel 7.

```
public static String formatAadressTekstWithNumbers(String aadress){
	Matcher m = patternNumbrilineTekst.matcher(aadress)
	StringBuffer sb = new StringBuffer()
	while (m.find()) {
		m.appendReplacement(sb, StringUtils.leftPad(m.group(), 5, '0'))
	}
	m.appendTail(sb)
	return sb.toString()
}
```

Joonis 7 Numbriliste sõnede teisendamine

5.2. Asukohtade otsingu loogika

Otsingu algoritm jaguneb järgnevateks üldisteks sammudeks

1) Otsingusõne töötlemine.
2) Päringu sõnede loomine töödeldud otsingusõnest ja otsitava asukoha tüübi määramine (numbri ja korteri tunnuse sisalduvuse kontroll).
3) EHAK piirangu kontrollimine ja vastavate otsingusõnede loomine.
4) EHAK tasemete otsimine piirangu alusel.
5) Otsitava asukoha tüübile vastavast indeksist otsimine (tähestikuline, numbriline või korter). EHAK piirangute arvestamine (EHAK piirangu sõne peab sisalduma kas lähiaadressis või asukoht peab kuuluma vastavasse haldusüksusesse).
6) Raadiuse piirangu olemasolul rakendatakse koordinaadi ja raadiuse järgi tulemuste piiramine.
7) Asukohtade sorteerimine.

Päringu puhul on alati täidetud tingimus et kõik sõnad, mis kasutaja otsingväljale sisestas, peavad sisalduma ka tagastavates asukohtades. Sellise lähenemise puhul ei tagastata asukohti, mis kattuvad ainult ühe otsingsõnaga mitmest. Samas ainult sellisest lihtsast lähenemisest ei piisa, sest ainult täisaadressi järgi otsides ei ole võimalik eristada, millise aadressiteksti osas otsingusõnega kattuvus on. Selle probleemi lahendab lähiaadressist otsimine, mille kaudu saab kontrollida et vähemalt üks osa sisulisest otsingsõnest sellega kattub. Sisulise otsingusõne all on mõeldud sellist sõne, mis viitab asukoha nimele, mitte liigile. Seetõttu on lähiaadressist otsimisel vajalik eemaldada kasutaja poolt sisestatud sõnadest liigi sõnad. Nende hulka kuuluvad näiteks tänav, maantee, puiestee, metskond.

Kõigepealt eemaldatakse mitteolulised tähemärgid (komad, punktid, alakriipsud jne), mis ei esine üheski asukohas ja võisid olla kasutaja poolt sisestatud ekslikult.

Nende sõnade järgi, millega otsitakse lähiaadressist (*kontrolltekst*) eemaldatakse veel lisaks liigi sõnad (mnt, linn, küla, maantee, talu, tn jne).

Töödeldud sõnad viiakse Lucene päringu süntaksile vastavale kujule [16], et neid otsingutermina kasutada. Seejuures analüüsitakse termi koostamisel, kas sõnad sisaldavad numbreid ja sidekriipse. Selle järgi tehakse valik, kas otsida nii-öelda tavalisi, numbrilisi või korter asukohti. Selle töötlemise käigus arvestatakse ka teatud eranditega, mis sidekriipsu ja numbreid sisaldavate aadressinimetuste juures välja on tulnud. Indeksis hoitakse tähestikulisi sidekriipsu sisaldavaid sõnu eraldi, aga numbrite puhul jääb sidekriips alles.

Otsitekst kasutatakse täisaadressi järgi otsimiseks ja *kontrollteksti* kasutatakse lähiaadressist otsimiseks. Joonisel 8 on kood, mille abil luuakse *otsitekst*

```
for (int i = 0; i < otsinguSonad.size(); i++) {
	sona = otsinguSonad[i].trim()
	boolean numeric
	if (StringUtils.isNotBlank(sona)){
		numeric = false
		if (sona.matches(".*\\d.*")) {
			numeric = true
			alpha_numeric = true
			if (sona.contains("-")) {
				alpha_korter = true
			}
		}
		if (!numeric) {
			sona = sona.replaceAll('-|/', ' ').replaceAll(" +", "
			")trim()
		}
		if (sona.contains("'")) {
			otsitekst += sona.replaceAll("'.*", "") + "* "
		} else if (sona.matches("\\d*/\\d*-\\D*")) {
			sona = sona
		} else if (sona.matches("\\D*-\\D*/\\d*") || sona.matches("\\D*-
		\\D*-\\D*\\d*")) {
			sona = sona.replaceFirst("-", "* ")
		} else if (sona.matches("\\d*/\\d*-\\D*")) {
			sona = sona.replaceFirst("-", "* ").replaceFirst("-", "
			")
		} else if (sona.matches("-\\d*|\\d*-")) {
			sona = sona.replaceFirst("-", "")
		}
		if (StringUtils.isNotBlank(sona)) {
			List testList = sona.split(" ")
			if (testList.last().size()==1 &&
			!testList.last().matches("\\d") && i ==
			(otsinguSonad.size()-1)) {
				otsitekst += sona + ' '
			} else {
				otsitekst += sona + '* '
			}
		}
	}
}
```

Joonis 8 Otsinguks kasutava teksti koostamine

Järgnevalt otsitakse kasutaja poolt sisestatud sõnadest EHAK piiranguid. Kui kasutaja on sisestanud mõne valla või linna koos vastava liiginimetusega (nt „Tartu linn“), siis selle põhjal hiljem piiratakse otsingutulemusi vastavate EHAK tasemete järgi. Kui sisestatud sõnades leitakse viide taseme tüübile, siis lisatakse see vastavasse EHAK piirangu tingimusse (Joonis 9).

```
if (sona.matches("maakond") && kontrollSonad[i-1]!="") {
	ehakPiirang = true
	ehak1Piirang += kontrollSonad[i-1].replaceAll("/|-", " ") + '* ' +
	kontrollSonad[i] + "*"
}
```

Joonis 9 EHAK piirang

Kui kasutaja oli sisestanud EHAK piirangu, siis minnakse haldusjaotuse id'd otsima (Joonis 10).

```
if (ehak1Piirang!="") {
	ehakList1 = Asukoht.searchEvery(reload: false) {
		must(queryString(ehak1Piirang.trim(), [useAndDefaultOperator:
		true, defaultSearchProperty: "tase1Nimetus"]))
	}
}
```

Joonis 10 EHAK kitsenduse otsimine

Kui otsingusõne sisaldab numbrit hakatakse asukohta otsima *lahi_numeric* või korteri puhul *lahi_korter* indeksist (Joonis 11). Kui otsingusõne numbrit ei sisaldanud, siis tehakse sama päring *lahi_alpha* indeksist. Otsingusüntaksi kirjeldus on leitav *Searchable plugina* dokumentatsioonist [17].

```
if (alpha_numeric) {
	must {
		if (alpha_korter) {
			addShould(queryString(kontrolltekst,
			[useAndDefaultOperator: false, defaultSearchProperty:
			"lahiKorter"]))
		} else {
			addShould(queryString(kontrolltekst,
			[useAndDefaultOperator: false, defaultSearchProperty:
			"lahiNumeric"]))
		}
	}
	must(queryString(otsitekst, [useAndDefaultOperator: true,
	defaultSearchProperty: "taisaadress"]))

}
```

Joonis 11 Lähiaadressi otsing

Koos lähiaadressi vastavalt väljalt otsimisega lisatakse päringule ka EHAK *id*-de piirang, mis eelnevalt oli välja otsitud. Kuna eksisteerib aadresse ja objekte, mille nimi kattub mõne EHAK taseme nimega, siis peab tagastavad aadressid kas kuuluma vastavasse haldusjaotusesse või piirangu sõne peab sisalduma lähiaadressis (Joonis 12).

```
must { // AND

	if (ehakList1) { // esimese taseme kontroll
		ehakList1.each {
			piirang += it.tase1 + "* "
		}
		if (StringUtils.isNotBlank(kontrolltekst)) {
			should { // OR
				// 1. piirang: piirangu sõne kuulub esimesse
				// tasemesse
				addMust(queryString(piirang,
				[useAndDefaultOperator: false,
				defaultSearchProperty: "tase1"]))
				// 2. piirang: piirangu sõne sisaldub
				// lähiaadressis
				if (alpha_korter) {
					(queryString(kontrolltekst,
					[useAndDefaultOperator: false,
					defaultSearchProperty: "lahiKorter"]))
				} else {
					addMust(queryString(kontrolltekst,
					[useAndDefaultOperator: false,
					defaultSearchProperty: "lahiNumeric"]))
				}
			}
		}
	}
...
```

Joonis 12 EHAK piirang lähiaadressi otsimisel

Tähestikulise otsingu juurde lisandub veel ainult EHAK taset sisaldavate aadresside otsing (Joonis 13). Iga EHAK taseme järgi otsitakse eraldi otsingusõne kattuvust ja lõpuks kontrollitakse, et kõik kasutaja poolt sisestatud sõnad on olemas ka tagastavates aadressides.

```
aadressList += Asukoht.searchEvery(reload: false) {
	must {
		addShould(queryString(kontrolltekst, [useAndDefaultOperator:
		false, defaultSearchProperty: "tase1Nimetus"]))
		addShould(queryString(kontrolltekst, [useAndDefaultOperator:
		false, defaultSearchProperty: "tase2Nimetus"]))
		addShould(queryString(kontrolltekst, [useAndDefaultOperator:
		false, defaultSearchProperty: "tase3Nimetus"]))
	}
		must(queryString(otsitekst, [useAndDefaultOperator: true,
		defaultSearchProperty: "taisaadress"]))
}
```

Joonis 13 EHAK taseme aadresside otsing

Kui otsingufunktsioonile on ette antud koordinaadi ja raadiuse parameetrid filtreeritakse *PostGis* [23] piirangu abil otsingust välja sellised aadressid, mis vastava koordinaadi ja raadiuse sisse jäävad (Joonis 14)

```
if (radius) {
	def idList = aadressList.collect {it.id}
	List koord = circle.replaceAll("\\(|\\)|,","").split(" ")
	aadressList = Asukoht.createCriteria().list {
		sqlRestriction "ST_DWithin(the_geom,
		ST_SetSRID(ST_Transform(ST_SetSRID(ST_Point(${koord[1]}::float,
		${koord[0]}::float),4326),3301), 3301), $radius)"
		inList("id", idList)
	}
}
```

Joonis 14 Koordinaadi ja raadiuse piirang

Vana algoritmiga võrreldes on uue lahenduse sisuline eelis see, et otsingud toimuvad väiksema andmehulga peal. Samuti kasutatakse vähem piiravaid kitsendusi, mis varem teostati lisaindeksite kaudu (taseme ja koordinaadi olemasolu). Uue lähenemise juures on ühendatud andmete ülesehitus sellega, kuidas neid otsitakse. Täpselt sama kriteeriumi alusel, millega andmed erinevatesse indeksitesse liigendati, hinnatakse pärast ka kasutaja sisendit, mille järgi otsing toimub. Otsingukiirust parandab kindlasti ka asjaolu, et numbreid koheldakse indeksis teistest sõnedest erinevalt. Selle tulemsena ei tagastata numbriliste aadresside otsimisel selliseid numbreid sisaldavaid asukohti, mida tegelikult ei otsitud. EHAK taseme ja koordinaadi järgi piiramine on samuti uus funktsionaalsus.

5.3. Asukohtade sorteerimine

Otsinguga tagastatud asukohad on vaja sorteerida enne kui need kasutajale kuvatakse. Sorteerimise põhimõte võrreldes vana lahendusega sisuliselt ei muutu. Vastavalt Lisas 1 oleva analüüsi järgi sorteeritakse asukohad piirkonna, taseme ja lõpuks nimetuse tähestikulise järjekorra järgi (Joonis 15).

```
aadressList = aadressList.sort { a, b ->
	(b.tase1 in omaMaakond) <=> (a.tase1 in omaMaakond) ?:
	compareTase(b.tase1Nimetus, a.tase1Nimetus) ?:
	compareTase(b.tase2Nimetus, a.tase2Nimetus) ?:
	compareTase(b.tase3Nimetus, a.tase3Nimetus) ?:
	a.lahiAlpha.length() <=> b.lahiAlpha.length() ?:
	ac.compare(a.lahiAlpha, b.lahiAlpha) ?:
	a.taisaadress <=> b.taisaadress
}

public int compareTase(String s1, String s2) {
	if (s1) {
		return 1
	} else if (s2) {
		return -1
	} else {
		return 0
	}
}
```

Joonis 15 Asukohtade sorteerimine otsingutulemuses

Ainsa olulise muudatusena oli vajalik täiendada tähestikulist sorteerimist. Kui kasutaja sooviks järjekorras näha numbreid „1, 1a, 1b, 10“, siis tavapärane sorteerimine asetab numbrid alati tähtedest ette. Uus sõnede sorteerimine algoritm arvestab selle eripäraga ja järjestab numbrilised sõned korrektselt. Lahenduse loomisel lähtusin Dave Koelle „The Alphanum Algorithm“ [18] põhimõttest ja lisasin sellele juurde võimekuse sorteerida ka null väärtusi ja sõnu mis sisaldavad erimärke (näiteks side- ja kaldkriipsud) ja tühikuid.

5.4. Asukohtade loendi esitamine kasutajale

Põhiline ajend uue lahenduse loomiseks oli vajadus kõiki erinevat tüüpi asukohti ühtselt otsida ja kasutajale kuvada. Leitud asukohti kuvatakse ühtselt, see tähendab et alati rakendub sama sorteerimise põhimõte. Maanteede ja objektide puhul kuvatakse lisaks asukoha nimele ka selle tüüp. Samuti kuvatakse endiselt 50 esimest kirjet.

Asukoht tabeli struktuur võimaldab hiljem ka hõlpsalt realiseerida asukoha tüübi järgi otsimist. Näiteks on võimalik otsida kõiki sularahaautomaate või parklaid märksõnade järgi filtreeritud piirkonnas.

6. Asukohtade otsingu testsüsteem

Asukohtade otsingu testsüsteem on rakendusse integreeritud võimekus, mis kontrollib otsingute korrektset töötamist. Kontrolliga tagatakse, et asukohad on leitavad, see tähendab et otsingusõnele vastav tagastatud kirje on esimese viiekümne tulemuse hulgas. Päringu aeg peab serveri poole peal jääma alla 100 millisekundi (Esialgne analüüs – Lisa 1).

6.1. Testsüsteemi kirjeldus

Otsingu testimise esimene eesmärk on kontrollida, kas otsitav asukoht tagastatakse. Lisaks sellele on oluline otsitava asukoha paiknevus tagastavate andmete hulgas ning päringute kiirus ja stabiilsus. Süsteem suudab kontrollida nii kõikide andmebaasis olevate asukohtade leitavust kui ka järjepidevalt andmete muutumise käigus uusi asukohti testida.

Kõik asukohad läbi käies kontrollitakse mitmenda tulemusena asukoht tagastatakse ja kui kaua päring aega võttis. Selle informatsiooni põhjal on võimalik üles leida asukohad, mis pole hästi ülesleitavad või võtavad kauem aega.

Otsingusõne koostatakse EHAK aadresside puhul viimase täidetud taseme nimetuse järgi ja lähiaadressi puhul lisatakse lähiaadressile, mis koosneb ainult ühest sõnast, viimane EHAK taseme osa. Sellest ei pruugi aga alati piisata ja valla, linna või küla tase on vaja lisada ka selliste lähiaadresside puhul, mida leidub Eestis palju (näiteks samanimelised tänavad mitmes haldusüksuses: Kooli tn, Pargi tn).

Kõikide asukohtade üle kontrollimine võtab aega mitu tundi, seega tulemused on mõttekas kirjutada tekstifaili või andmebaasi tabelisse ja tulemuste kuvamist mitte ootama jääda. Asukoht tabelis on olemas eraldi veerg, millesse kirjutatakse konkreetse asukohaga esinenud probleem. Probleemid on antud klassifikaatoritega: pole esimese viiekümne tulemuse hulgas, pole esimese kahekümne tulemuse hulgas, päring aeglane. Pärast testimise lõppu saab tulemuse kätte csv failist ning kasutajaliidese kaudu on võimalik kuvada kõik probleemsed asukohad vastavalt probleemi kirjeldusele.

Teist tüüpi kontrollimine on perioodiline ja toimub pärast igaöist aadresside uuenemist. Aadressid teisendatakse asukohtadeks ja lisandunud ning muutunud asukohad tuleb üle kontrollida. Probleemseid asukohti on võimalik näha kasutajaliidese kaudu.

Ainult muutunud asukohtade kontrollimine aga ei taga, et mõni varem ülesleitav asukoht tulevikus uute andmete lisandumise tulemusena tagastavatest asukohtadest välja ei jääks. Seega sooritatakse perioodiliselt uuesti ka täiskontrolli.

Lisaks sellele on magistritöö raames realiseeritud vana ja uue algoritmi võrdlus. Kõikide aadresside leitavuse kontroll tehakse vana struktuuri ja algoritmi kasutades, seejuures seekord moodustatakse otsingu sõne lähiaadressi järgi, mitte ei kasutata täisaadressi. Selline lähenemine on märksa realistlikum, sest aadresse ei otsita üldjuhul valla ja maakonna täpsusega. Selline lähenemine võimaldab kontrollida andmete sorteerimist päriselule lähedastel tingimustel.

6.2. Testsüsteemi tehniline lahendus

Kuna testsüsteem kontrollib kõikide asukohtade leitavust ja päringu kiirust, siis indekseid eraldi kontrollida pole vaja. Näiteks *Luke* [19] või *CheckIndex* [20] vahendiga on võimalik kontrollida indeksite ülesehitust ja vaadelda erinevat statistikat, kuid see ei aita tagada seda, kas indeksit täidavad oma eesmärki ehk aitavad leida asukohti.

Kuna täiskontrolli üle kõigi asukohtade tehakse harva, sisuliselt ainult pärast andmete alglaadimist, siis ma ei pidanud vajalikuks kasutada selle loomiseks eraldi platvormi või rakendusliidest. Järjepidevalt toimuvad kontrollid on lihtsasti realiseeritavad kasutades *Quartz* [21] pluginat, mis võimaldab kontrollimist rakenduse töötamise ajal perioodiliseks muuta

Testsüsteem on integreeritud rakendusse ühe komponendina. Täiskontroll on võimalik teostada kasutajaliidese kaudu, mis vastava teenuse meetodi käivitab. Asukohad tabelist loetakse kümne tuhande kirje kaupa andmed sisse. Iga asukoha põhjal luuakse testpäring. Päringu sisuks on lähiaadress tingimuslikult koos EHAK taseme viimase osaga (Joonis 16). Lähiaadressi mitteomavate aadresside puhul on otsingusõne ainult vastava aadressi EHAK taseme viimane osa.

```
temp = aadress.lahiaadress
sonad = temp.trim().split(" ")
if (sonad.size()==1 && !sidekriips && tyyp != "objekt") {
	String replaceString = aadress.lahiaadress
	List temp2 = aadress.taisaadress.replaceFirst(", $replaceString\$",
	"").split(",")
	String viimaneEhak = temp2.get(temp2.size()-1)
	otsisona = viimaneEhak + " " + temp
}
```

Joonis 16 Test otsingusõne koostamine

Päringule tagastatakse järjestatult viiskümmend Asukoht objekti ja nende hulgast hakatakse konkreetset rida otsima. Tulemuse puhul jäetakse meelde kirje tagastamise kiirus ja asukoht (Joonis 17). Lisaks sellele peetakse abimuutujate abil statistikat selle üle, kui palju on aeglaseid päringuid. Vastavalt analüüsile (Lisa 1) on aeglane päring selline, mis võtab serveri poole peal aega üle 100 millisekundi. Teiseks kontrollitakse, kas otsitav kirje asub järjestuses mittesobival positsioonil - vastavalt kas kaugemal kui kahekümnes või viiekümnes positsioon. Probleemi esinemise korral tehakse märge vastavale Asukoht tabeli reale, et hiljem oleks võimalik ülekontrollida ainult probleemsed andmed.

```
startTime = System.nanoTime();
estimatedTime
aadressidOtsingList = leiaAadress(otsisona, omaMaakond, true, 50)
estimatedTime = System.nanoTime() - startTime;
td = (estimatedTime / 1000000).toInteger()
for (int i = 0; i < aadressidOtsingList.size(); i++) {
	if (aadressidOtsingList[i].id == aadress.id) {
		koht = i
		break
	}
}
```

Joonis 17 Otsingu kontrollimine

Pärast kontrolli lõppu tagastatakse koondraport, mis sisaldab välju: raporti aeg, kontrollitud asukohtade arv, keskmine päringu aeg, esimese 20 kirje hulgast väljas, esimese 50 kirje hulgast väljas, aeglased päringud.

Samuti on võimalik tulemuste täpsemaks analüüsimiseks ja esitamiseks väljastada csv fail koos iga päringu tulemusega. Lisaks on võimalik testida asukoha tüüpe eraldi, et

tuvastada probleeme tüübi järgi ja paremini süsteemi edasi arendada (pole vaja kõiki andmeid kontrollida, kui probleeme esineb mingit kindlat tüüpi asukohtadega).

Järjepidev kontroll teostakse perioodiliselt pärast andmete uuenemist. Andmebaasist päritakse kõik asukohad, mis on eelnevaga võrreldes muutunud (asukoht tabelis hoitakse viimati muudetud aega). Probleemid kirjutatakse logi tabelisse ja neid on kasutajaliidese kaudu võimalik vaadata. Seejärel on võimalik vajaduse korral andmeid käsitsi parandada ning teavitada kasutajaid ja andmete haldajat probleemidest

7. Tulemused

Geoandmete rakenduse uue iteratsiooni valmimisel arvestati vana süsteemi probleemidega ja lisandunud nõuetega. Uue versioon töötati välja samal ajal nii andmeid ja nõudeid analüüsides kui ka erinevaid tehnilisi lahendusi katsetades.

Töömahu mõttes moodustas süsteemi arendamise juures valdava osa just andmete analüüs ja süsteemi disain. Enne lõplikku algoritmi väljatöötamist prooviti läbi alternatiivseid lahendusi nii tehnoloogiate (PostgreSQL Full Text Search [22]), kui ka erinevad variandid andmete struktuuris (mitme tabeli lahendus, aadressi tasemete põhised indeksid). Maaameti X-tee teenustega liidestumine, andmete teisendamine, testsüsteemi loomine ning Lucene indeksite infrastruktuuri ja otsingualgoritmi programmeerimine olid ülejäänud aeganõudvad tööülesanded.

Uut süsteemi on vaja katsetada ja võrrelda selle võimekust vana lahendusega. Järgnevates alampeatükkides vaadeldakse eraldi uut ja vana otsingut ning võrreldakse tulemusi.

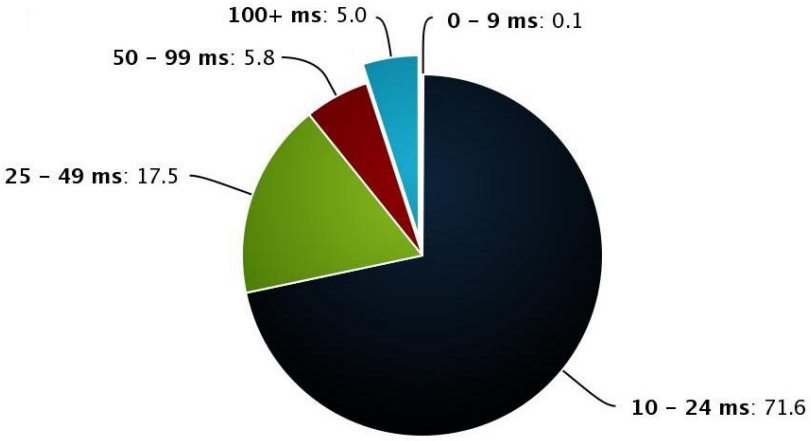
7.1. Vana aadressi otsing

Kontrollimise käigus käidi läbi kõik ligikaudu 1,2 miljonit kehtivat aadressi. Järgnevas tabelis 2 on välja toodud päringutele kulunud aja ja tagastamise koha statistika.

Tabel 2 Vana otsingu statistika

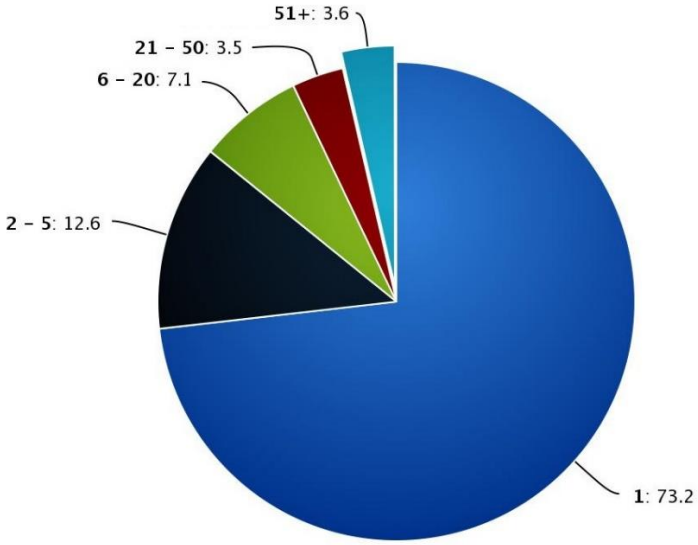
Keskmine päringu aeg	Päringu aja usaldusvahemik	Keskmine päringu koht	Päringu koha usaldusvahemik
32,86 ms	0,15ms	2,23	0,02

Olgugi et keskmine päringu kiirus jääb alla analüüsis kehtestatud maksimumile, on aeglaseid päringuid siiski 5% (Joonis 18). Valdav osa päringute kiirusest jääb kümne ja kahekümne viie ms vahele, sealjuures alla kümne ms päringuid on kõigest 0,1%.



Joonis 18 Vana otsingu kiiruse jaotuvus

Aadresside paiknevus otsingutulemuste hulgas on küllaltki hea, kuid problemaatiline on lubamatult suur tagastamata tulemuste arv – 3,6% aadressidest pole leitavad (Joonis 19).



Joonis 19 Vana otsingu kohtade jaotuvus

Aeglaseid päringuid uurides selgub, et lisaks EHAK aadresside otsimisele on probleeme ka 7 taseme aadressidega. Tabelis 3 on välja toodud probleemsete päringute osakaal aadressi tasemete kaupa.

Tabel 3 Aeglaste päringute statistika

Taseme nr	1	2	3	4	5	6	7	8
Aeglaste päringute arv	15	29	231	1	428	4392	18990	5465
Aeglaste päringute osakaal	93,8%	13,5%	5,0%	0,1%	2,5%	1%	7,6%	1,0%

Tabelis 4 on välja toodud mitte leitavate aadresside jaotuvus tasemete järgi. Ligikaudu 9% tänavatest ei ole otsingusüsteemist leitavad kasutades selleks ainult tänava nime. Suurim probleem on aga teise ja kolmanda taseme aadressidega, nende mitteleitavus on tõsine puudujääk. Ei olnud näiteks võimalik leida Viljandi ja Tartu valda ja Harju küla.

Tabel 4 Mitte leitavate aadresside statsitka

Taseme nr	1	2	3	4	5	6	7	8
Tagastamata päringute arv	0	5	40	3	1539	2416	13454	5648
Tagastamata päringute osakaal	0%	2,3%	0,9%	0,4%	9,1%	0,7%	5,4%	1,0%

Mitte leitavaid aadresse on enim Harju, Tartu ja Viljandi maakonnas – vastavalt 3952, 2494 ja 2158.

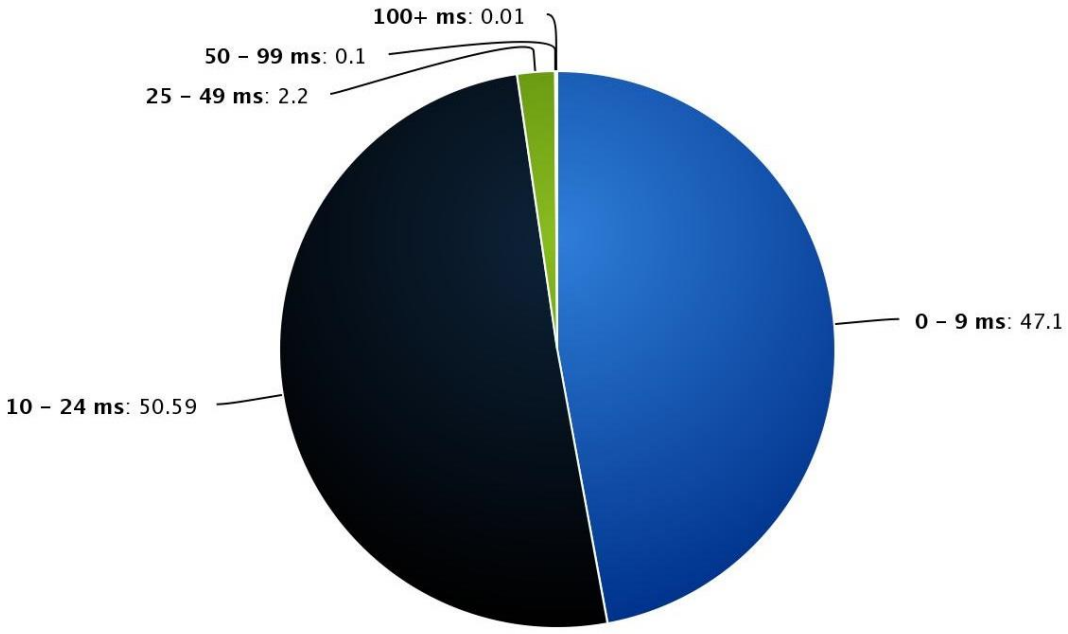
7.2. Uus aadressi otsing

Uue teostuse korral on päringu keskmine aeg 13,78 millisekundit, kusjuures mediaan päringuaeg on 16 ms. Keskmine päringu aeg ja koht on kirjas tabelis 5.

Tabel 5 Uue otsingu statistika

Keskmine päringu aeg	Päringu aja usaldusvahemik	Keskmine päringu koht	Päringu koha usaldusvahemik
13,78 ms	0,05ms	0,67	0,03

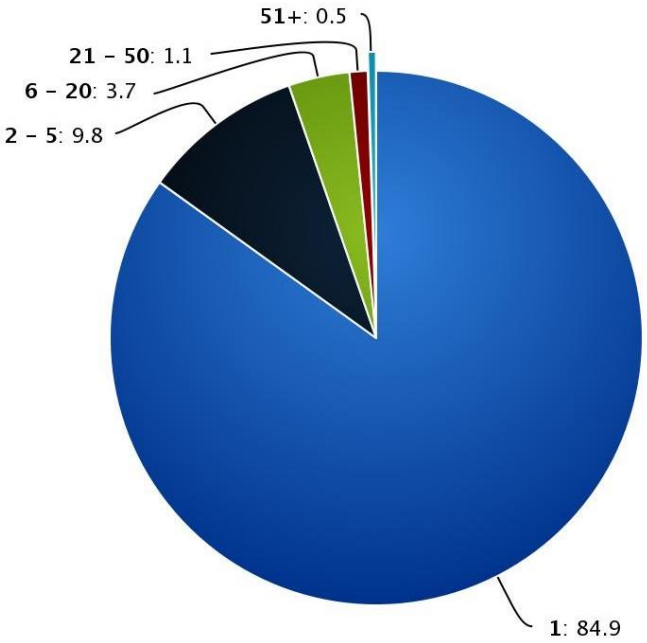
Seega keskmine päringu aeg jääb alla lubatut ja kõigest 0,01% päringutest võtavad aega rohkem kui 100 millisekundit. Päringute ajaline jaotuvus on joonisel 20.



Joonis 20 Uue otsingu ajaline jaotuvus

Mõõtmisest järeldub, et nõue 2.3.1 on täidetud – ainult 1% tagastatud asukohtadest on väljaspool esimest 20 kirjet. Esialgsete tulemuste järgi aga ei ole leitav 0,5%

aadressidest (Joonis 21). See arv oli lubamatult kõrge ja oli vajalik leida probleemi põhjus ja lahendus.



Joonis 21 Uue otsingu kohtade jaotavus

Selgus, et suur osa probleemseid aadresse on seotud sellega, kuidas töödeldakse otsingusõnes punkte ja komasid – need eemaldatakse, sest ei oma otsimise juures tähendust. Näiteks vigase kujuga aadressi puhul „Ritsu 10-Krt.9“ ei ole võimalik seda otsida, sest kui punkt kustutatakse jääb otsingusõnesse alles lihtsalt „Krt9“. Kuna analüüsi käigus tehti selgeks et üleliigsed kirjavahemärgid tuleks otsingusõnest kindlasti eemaldada, oli vaja andmeid parandada. Iga koma ja punkti järele lisati tühik ja hiljem eemaldati lisatühikud (Joonis 22).

```
update asukoht set taisaadress = replace(taisaadress, '.', '. ');
update asukoht set lahiaadress = replace(lahiaadress, '.', '. ');
update asukoht set taisaadress = replace(taisaadress, ',', ', ');
update asukoht set lahiaadress = replace(lahiaadress, ',', ', ');
update asukoht set taisaadress = replace(taisaadress, '  ', ' ');
update asukoht set lahiaadress = replace(lahiaadress, '  ', ' ');
```

Joonis 22 Punktide ja komade töötlemine

Lisaks oli probleeme ka keerulise kujuga side- ja kaldkriipse sisaldavate aadressidega. Kuna indeksis hoitakse eraldusmärkidega sõnu vastavalt kas koos või eraldi sõltuvalt sellest, millised selle osad numbreid sisaldavad, siis tuleb ka otsingusõne koostamisel selliste eripäradega arvestada. Keerulisemate aadresside puhul viidi sisse regulaaravaldise kontrollid, et korrektne otsinguterm koostada. Mõned näited sellest kontrollist on välja toodud joonisel 23.

```
if (sona.matches("\\D*-\\D*/\\d*") || sona.matches("\\D*-\\D*-\\D*\\d*")) {
	sona = sona.replaceFirst("-", "* ")
} else if (sona.matches("\\d*/\\d*-\\D*")) {
	sona = sona.replaceFirst("-", "* ").replaceFirst("-", " ")
} else if (sona.matches("-\\d*|\\d*-")) {
	sona = sona.replaceFirst("-", "")
}
```

Joonis 23 Otsingusõne töötlemine regulaaravalidsega

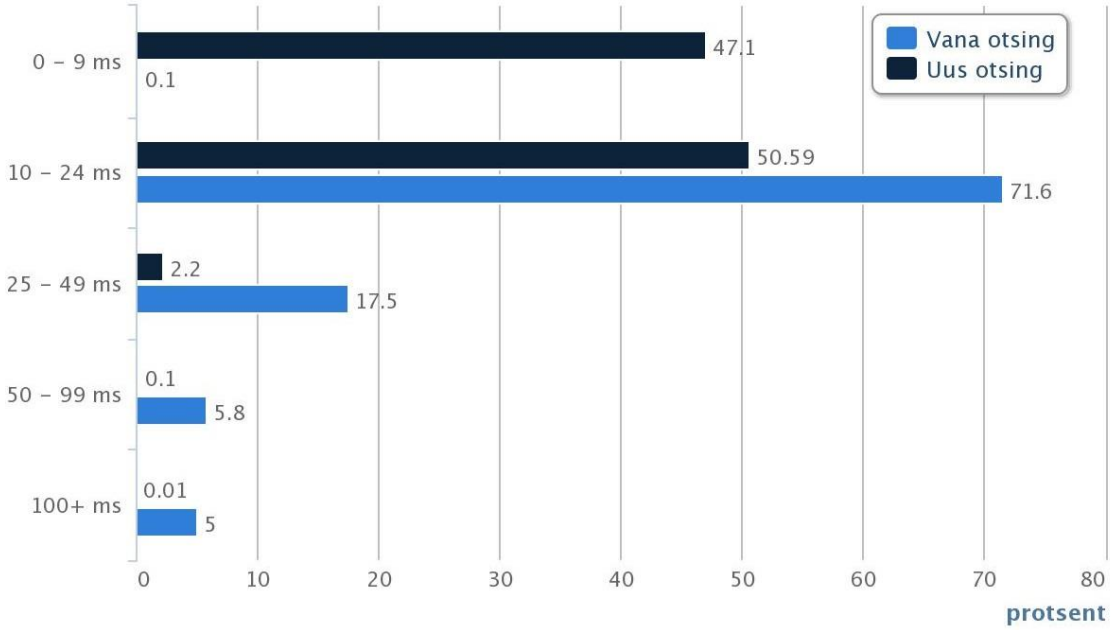
Nende kahe meetme rakendamise tulemusena on nüüd leitavad 99,96% aadressidest. See tähendab, et ainult 469 aadressi pole leitavad. Viimistlemise ruumi veel on ja nende probleemsete aadressidega tegeletakse edasi.

Objektide puhul olid leitavad kõik kirjed, peale mõne üksiku vigase objekti. Vigased olid näiteks objektid „Mill¬onamägi“ või „Pad¬assaar“. Neid kirjeid oli vaja käsitsi muuta ja teavitada andmete omanikku. Kohanimeregistri *kohtade* ja *maanteede* andmetega probleeme ei olnud.

7.3. Vana ja uue otsingu võrdlus

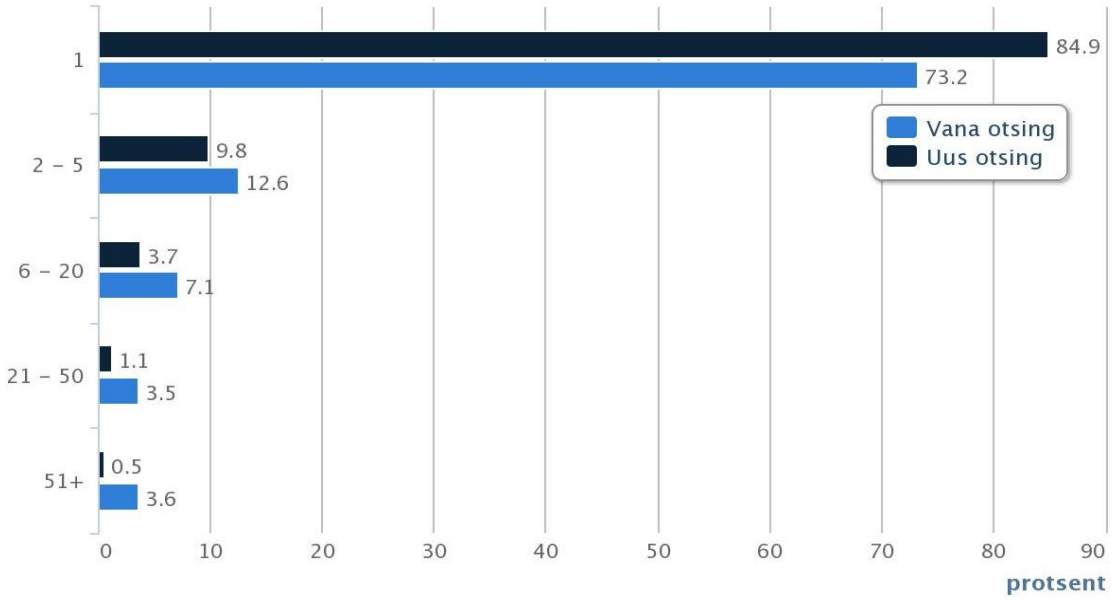
Vana ja uut otsingut võrreldakse kahe kriteeriumi alusel: otsingu kiirus ja aadresside leitavus ehk paiknevus tagastatavas loendis.

Kiiruse osas on uue lahenduse eelis oluline eelkõige selle osas, et väga vähe on aeglaseid päringuid (üle 100 ms). Uue otsingu puhul jäävad peaaegu pooled päringud alla 10 ms ja ülejäänud pool alla 50 ms. See viitab sellele, et ka andmete hulga kasvamise puhul on lahendus sobiv. Ajaline võrdlus on joonisel 24.



Joonis 24 Otsingute ajaline võrdlus

Leitavuse osas on vana ja uus otsing esmapilgul küllaltki sarnased (Joonis 25). Kriitilise tähtsusega aga on siinjuures *51+* kategooria, ehk aadressid, mida ei leita. Esialgu oli see uue otsingu puhul 0,5%, kuid täiendavate andmeparanduste ja regulaaravaldise kontrolli tõhustamise tulemusena langes see arv 0,04% peale. Ilmeka arvuna võib välja tuua, et uue algoritmi puhul on tagastamata aadresside hulk 90 korda väiksem.



Joonis 25 Otsingute aadresside leitavuse võrdlus

8. Kokkuvõte

Magistritöö eesmärk oli parandada ja täiendada geoandmete rakenduse asukoha andmete haldamist ja otsimist. Kuna see funktsionaalsus on osa Häirekeskuse ohutuskriitilisest infosüsteemist, oli lisaks vajalik välja töötada kontrollmehhanism rakenduse korrapärase toimimise jälgimiseks.

Maaametist pärinevate aadresside uuendamiseks töötati välja uus lahendus, mis arvestab korrektselt nende andmete muutumisega. Uuendamise protsessis lähtutakse päriselus eksisteerivatest objektidest, erinevalt eelnevalt kasutuses olnud aadressi nime põhisest lähenemisest. Uus lahendus kooskõlastati Maaametiga ja on käesoleva töö raames realiseeritud.

Üks olulisemaid ajendeid otsingusüsteemi täiendamiseks oli muuta erinevate asukohtade leidmine ühtseks, ühel andmestruktuuril põhinevaks süsteemiks. Seetõttu arendati töö käigus välja ühtne asukoha haldamise võimekus, mis tähendab et erinevatest allikatest pärinevad andmed koondatakse ühtsesse struktuuri. Andmete teisendamine operatiivkujule on Maamati aadresside puhul automaatne ja teiste allikate puhul pool-automaatne – andmete uuendamiseks laetakse üles kindlaksmääratud struktuuriga csv fail, mille sisu töödeldakse ja teisendatakse *Asukoht* tabelisse.

Kõik aadresside ja loodusesse kuuluvate objektidega seotud andmed hoitakse nüüd ühes *Asukoht* tabelis, mille struktuur töötati välja samaaegselt uue otsingualgoritmi loomisega. Varem toimus otsimine struktuuri põhjal, mis oli väliselt partnerilt etteantud ja sobis hästi andmete haldamiseks, kuid mitte paindliku otsimise toetamiseks (Maaameti *Aadressid* tabel). Uue struktuuri ja algoritmi väljatöötamise tulemusena töötab otsing vanaga võrreldes märkimisväärselt paremini. Aeglaste päringute osakaal vähenes 5%-ilt 0,01%-ini. Seejuures võtab kõikide Eestis leiduvate asukohtade otsimise kiirus umbes poolel juhul aega alla kümne millisekundi ja ülejäänu puhul alla kahekümne viie millisekundi. Ka aadresside üldine leitavus paranes oluliselt. Kuigi keskmine leitavus oli hea ka vana süsteemi puhul, siis muret tekitas fakt, et otsing ei tagasta 3,6% kirjetest. Uue lahenduse puhul oli see arv esialgu 0,5%, kuid täiendavate

meetmete ja andmeparanduste tulemusena saadi see näitaja 0,04% peale. See tähendab, et uue lahenduse puhul on tagastamata aadresside hulk 90 korda väiksem.

Need leitavuse ja päringu aja mõõtmise tulemused pärinevad testsüsteemist, mis loodi rakenduse korrektse toimimise kontrollimiseks. Rakendusse integreeritud funktsionaalsus võimaldab kontrollida kõikide asukohtade leitavust, probleeme analüüsida ja kasutajat intsidentidest teavitada. Pärast igaöist aadresside uuendamist kontrollitakse muutunud asukohtade leitavust ja tagastatakse raport rakenduse haldajale. Seega kõikide asukohtade leitavus on igal ajahetkel kontrollitav ja lisaks sellele on võimalik rakenduse volitatud kasutajal ka andmeid parandada, kui peaks probleeme esinema.

Ühe huvitava edasiarendamise suunana tasuks katsetada süsteemi skaleeruvust. Otsingu kiiruse mõõtmise head tulemused viitavad sellele, et süsteem peaks töötama hästi ka suurema andmehulgaga. Samuti oleks oluline rakendusele luua realistlikud koormustestid (olgugi, et vana lahenduse testimine andis positiivse tulemuse).

Magistritöös arendatud süsteem plaanitakse kasutusele võtta Häirekeskuse infosüsteemi järgmise arendusetapi käigus.

Kasutatud kirjandus

1. Grails
   http://www.grails.org/ (06.03.2014)
2. Groovy
   http://groovy.codehaus.org/ (06.03.2014)
3. Convention over Configuration.
   http://softwareengineering.vazexqi.com/files/pattern.html (06.03.2014)
4. Hibernate
   http://hibernate.org/orm/ (06.03.2014)
5. Spring
   http://projects.spring.io/spring-framework/ (06.03.2014)
6. Grails-i dokumentatsioon
   http://grails.org/doc/2.2.4/ (06.03.2014)
7. EHAK
   http://metaweb.stat.ee/view_xml.htm?id=3760026 (01.04.2014)
8. ADS – aadressiandmete süsteem
   https://www.ria.ee/teejuht/riigi-infosusteemi-olemus-ja-komponendid/aadressandmete-susteem-ads (01.04.2014)
9. Magistritöö „Ohutuskriitilise asukohtade otsingusüsteemi arendus“ Peeter Lump 2013
10. Maaamet
    http://www.maaamet.ee/ (02.04.2014)
11. Aadressiandmete register
    http://geoportaal.maaamet.ee/est/Andmed-ja-kaardid/Aadressiandmed-p112.html (02.04.2014)
12. KNR - kohanimeregister
    http://www.maaamet.ee/index.php?page_id=505 (02.04.2014)
13. ADS-iga liidestumise juhend versioon 1.5.1
    http://geoportaal.maaamet.ee/docs/aadress/ADS-ga_liidestumise_juhend_ver_1_5_1.pdf (02.04.2014)
14. ADS spetsifikatsioon
    http://geoportaal.maaamet.ee/docs/aadress/ADS_spetsifikatsioon_27-03-2012.pdf?t=20120328113511 (02.04.2014)
15. Shp fail
    http://wiki.openstreetmap.org/wiki/Shapefiles (03.04.2014)
16. Lucene päringu süntaks
    http://lucene.apache.org/core/2_9_4/queryparsersyntax.html (21.04.2014)
17. Grails searchable otsingu süntaks
    http://grails.org/Searchable+Plugin+-+Methods+-+search (21.04.2014)

18. Dave Koelle „The Alphanum Algorithm“
    http://www.davekoelle.com/alphanum.html (21.04.2014)
19. Luke – Lucene indeksite töövahend
    http://www.getopt.org/luke/ (23. 04. 2014)
20. CheckIndex (08.05.2014)
    http://solr.pl/en/2011/01/17/checkindex-for-the-rescue/
21. Quartz Plugin (22.05.2014)
    http://grails.org/plugin/quartz
22. PostgreSQL Full Text Search (22.05.2014)
    http://www.postgresql.org/docs/9.3/static/textsearch.html
23. PostGis (23.05.2014)
    http://postgis.net/

Lisa 1 – Süsteemi esialgsed nõuded

Nõuded Häirekeskuse infosüsteemi asukoha sisestus süsteemile

Infosüsteemi asukoha sisestuse süsteem võimaldab rakenduse kasutajal sisestada asukohta erinevatel meetoditel. Rakendus võimaldab sisestada märksõnu, mis lahenduvad aadressiks, objektiks või maantee osaks. Lisaks võimaldab rakendus sisestada koordinaate, mis pöörd geo-kodeeritakse aadressiks. Valitud aadressi juurde on võimalik lisada märkuseid.

Vormi täitmise üldnõuded

- Kui aadress on võimalik geo-kodeerida või pöörd geo-kodeerida, siis tuleb seda jooksvalt teha. Ehk kui aadressist on võimalik koordinaat leida ning vastupidi, siis tuleb seda teha. Koordinaatideks teisendatud aadressi või koordinaatide sisestamisel tekkinud aadressi on võimalik kaardile visualiseerida.
- Sisestatud asukohtade ajalugu on vaja säilitada ning ajaloolised asukohad tuleb kuvada selleks ette nähtud tabelis. Vajalik on näha aadressi ajalugu, maanteed, objekti; vastavalt ka muutmise aega ja muutjat (kasutajanimi). Ajaloolist aadressi on võimalik taastada aktuaalse aadressina.
- Asukohtade soovituslike väljade üldnõue: kuna asukohast peab alati saama koordinaadi tekitada, siis valida saab ainult olemasolevaid majasid, kortereid, talusid, tänavaid, asustusi, maanteid ja objekte st vaba-sisestust teha ei saa ja nö uusi kirjeid luua pole võimalik.
- Asukohtade soovituslike väljade üldreegel: kasutaja on süsteemis seotud piirkonnaga, piirkonnad on süsteemis omakorda seotud maakondadega. Soovituslikel väljadel tuleb pakutavatest vastetest leida nende aadressi järgi kasutaja piirkonnaga seotud kirjed ning sorteerida need pakutavas loetelus nö eraldi ja esimestena st rippmenüüs üleval pool. Seejuures tuleb need nö oma piirkonna valikuid kuvada läbiva suure tähega.

- Kui välja sisestus ei vasta nõuetele, siis kuvatakse välja taust punaselt. Näiteks aadress-tüüpi asukoht on sisestamata, maantee kilomeetrinumbrit pole olemas jne.

Aadress tüüpi asukoha sisestamine

Andmevälja kasutatakse maakonna, asula, linna, valla, küla, tänava, talu, maja või korteri sisestamiseks.

Kasutaja trükib otsitava märksõna või selle esitähed ning kasutajale kuvatakse sobivaid vasteid. Näiteks „Sõpruse“. Otsingut teostatakse alati sõne algusest, ei otsita keskelt ega lõpust.

Sisestada võib ka mitu märksõna osa eraldades need tühikuga. Sellisel juhul toimib iga trükitud sõnaosa täiendava kitsendusena. Näiteks „tar pa laeva“. Maakonna ja valla/asula taseme vahele ei pea trükkima koma tähist, vaid piisab tühikust.

Maja numbri osas kitsenduse seadmiseks tuleb trükkida number, näiteks „Sõpruse 12“. Korterinumbreid pakutakse kui numbri järgi on trükitud „-„ või „/“ märk, näiteks „Sõpruse 12-„

Kasutajale soovituste tegemiseks kuvatakse aadresse põhimõttel suuremalt haldusüksuselt väiksemale suunaga ehk standardsel kujul. Erandiks on aadresside kuvamisel lühendamise nõue ehk tuleb rakendada sõne-töötlust: sõna „maakond“ tuleb asendada lühendiga „mk“; sõna „linnaosa“ ei näidata. Võimalik peab olema üks konkreetne aadress soovitustest valida ning seda aadressina kuvada.

Kasutajale soovituste tegemisel tuleb esimese sorteerimisena tõsta oma piirkonnas asuvad aadressid ettepoole ning kuvada läbivalt trükitähtedega. Järgmisena tuleb aadressid kõigi ADS tasemete järgi tähestikuliselt sorteerida (suunas üldisemalt täpsemale). Ehk Harjumaa, Järvamaa jne. Kõige lõpus on Viljandi ja Võrumaa. Järgmine tase on vald/linn mis tuleb samamoodi sorteerida kuni kõige madalama tasemini välja.

Peale sobiva aadressi kasutaja poolt valimist tuleb rakendada sõna-töötlust. Sõna „maakond“ tuleb asendada lühendiga „mk“; sõna „linnaosa“ ei näidata. Kasutajale tuleb aadressi kuvada põhimõttel väiksemalt haldusüksuselt suuremale.

Peale aadress tüüpi asukoha sisestamist/valimist tuleb kontrollida kas sellel asub eeldefineeritud objekt. Kui aadressil ei asu ühtegi objekti, siis ei tehta midagi. Kui aadressil asub ainult 1 objekt, siis valitakse see automaatselt objekti otsingu väljale. Kui aadressil asub mitu objekti, siis objekti otsingu välja ei täideta, kuid selle järel kuvatakse eraldi sobivate objektide valikute rippmenüü.

Maantee tüüpi asukoha sisestamine

Andmevälja kasutatakse maantee valimiseks (põhi-, tugi- või kõrvalmaantee, sh rambi/ühendustee).

Kasutaja trükib otsitava märksõna või selle esitähed ning kasutajale kuvatakse sobivaid vasteid. Näiteks „Tartu“. Otsingut teostatakse alati sõne algusest, ei otsita keskelt ega lõpust.

Sisestada võib ka mitu märksõna osa eraldades need tühikuga. Sellisel juhul toimib iga trükitud sõnaosa täiendava kitsendusena. Näiteks „tar jõh“ leiab „JÕHVI-TARTU-VALGA“ maantee.

Kasutajale soovituste tegemiseks kuvatakse maanteid ametliku nimetuse, Eesti mnt koodi ja rahvusvahelisi A- ja B-klassi teede koodi nagu E263 järgi. Võimalik peab olema üks konkreetne aadress soovitustest valida ning seda aadressina kuvada. Kuvamise kuju: „maantee nimi (eesti maantee kood / rahvusvaheline kood)“. Näide: „TALLINN – TARTU – VÕRU -LUHAMAA (2/E263)“.

Maantee-tüüpi aadressi valimise tühistamiseks tuleb maantee-otsingu välja sisu kustutada (vastavalt kaovad ära ka KM ja lõik väljad) või kui aadress-tüüpi väljalt valitakse uus väärtus.

Maantee kilomeetrinumbri sisestamise väli. Väljale saab sisestada ainult numbrit.

Kui kasutaja sisestab numbri, mida vastaval teel ei esine kuvatakse välja taust punast värvi.

Kui kasutaja sisestab numbri, mis esineb vastaval teel, siis väljalt lahkudes arvutatakse sisestatud andmete alusel (Tee ja KM) välja konkreetne lõik maanteel.

Välja kuvatakse ainult juhul kui on valitud maantee ja on sisestatud nimele ja numbrile vastav väärtus (ehk leidub lõik mis vastab eelpool nimetatud tingimustele).

Valitud maantee ja KM-nr sisestamise järel tuleb automaatselt valida KM-le vastav lõik ning sellest tulenevalt täita automaatselt aadress-tüüpi aadressi väli ning taustal EHAK väärtus.

Klassifikaatorina tuleb kuvada lisaks valitud lõigule ka eelnevad ja järgnevad 2 lõiku, ehk kokku 5 lõiku.

Objekt tüüpi asukoha sisestamine

Andmevälja kasutatakse eeldefineeritud objekti (pood, bussipeatus, kõrgendatud ohuallikaga objekti – ATeS, vms) sisestamiseks.

Otsida saab nii objekti nime, aadressi kui selle alamosa järgi.

Kasutaja trükib otsitava märksõna või selle esitähed ning kasutajale kuvatakse sobivaid vasteid. Näiteks „Sää“ leiab kõik Säästumarketid. Otsingut teostatakse alati sõne algusest, ei otsita keskelt ega lõpust.

Sisestada võib ka mitu märksõna osa eraldades need tühikuga. Sellisel juhul toimib iga trükitud sõnaosa täiendava kitsendusena. Näiteks „sää juh“.

Kasutajale soovituste tegemiseks kuvatakse objekte nimi-aadress kujul. Näiteks: Viru hotell (Viru väljak 4, Kesklinna, Tallinna linn, Harju mk), Järve Selver (Pärnu mnt 238, Nõmme, Tallinna linn, Harju mk). Võimalik peab olema üks konkreetne objekt soovitustest valida ning seda aadressina kuvada. Soovituste tegemisel tuleb kasutajale kuvada objekti tüüpi ning selle järgi peab olema võimalik ka otsingut kitsendada.

Maanteel asuva objekti, näiteks bussipeatuse valimisel täidetakse lisaks aadressi väljale ka maantee väli kilomeetrinumbri täpsusega.

Juhul kui aadress-tüüpi väljalt sisestatakse aadress ning sellel aadressil asub üks objekt, siis valitakse objekt välja. Kui aadressil asub mitu objekti, siis kuvatakse objektide valiku andmeväljal kõiki valikuid.

Välja kuvatakse ainult juhul kui sisestatud on aadress kus asub mitu objekti. Vaikimisi ei valita ühtegi objekti, vaid kuvatakse sobivaid objekte valikus (kui suur maja, siis juhtum ei pruugi olla üldse seotud objektiga). Peale objekti valimist lisatakse valitud objekt objekt-tüüpi asukoha sisestamise veerule.

Objekti valimise järel tuleb automaatselt täita aadress-tüüpi asukoha EHAK väärtus.

Koordinaat tüüpi asukoha sisestamine

Asukoha koordinaate on võimalik sisestada kahes koordinaatsüsteemis L-Est (XY) ja WGS84 (NE). Ühes süsteemis koordinaatide sisestamise järel arvutatakse koheselt ka teise süsteemi koordinaadid.

Väljaspool HK teeninduspiirkonda asuvaid koordinaate ei saa sisestada.

Koordinaatide sisestamise järel teostatakse pöörd geo-kodeerimine ning leitakse sisestatud koordinaatidele lähimad aadressid. Süsteem pakub aadressid välja rippmenüüna.

Märkuste sisestamine

Sisestatava info osas kasutajale piiranguid ei seata.

Lahtrisse sisestatakse täpsustavat infot nagu uksekood, korrus vms. Samuti kui patsiendid/vigastatud asuvad märgitud aadressi ees tänaval, siis saab kirjutada, et asuvad maja ees/taga, „õues kuri koer“ vms.

Asukoha täpsustuse väli peab teksti sisestamisel pikenema koos teksti sisestusega ning kustutamisel peab kasti uuesti väiksemaks tegema.

Asukohtade otsingu süsteemi lisanõuded

- Eranditult kõik kehtivad asulad, linnad, vallad, külad, tänavad, talud, majad ja korterid on leitavad kasutaja poolt sisestatavate märksõnade järgi;
- Eranditult kõik kehtivad põhi-, tugi- või kõrvalmaanteed on leitavad kasutaja poolt sisestatavate märksõnade järgi;
- Eranditult kõik kehtivad objektid (poed, bussipeatused, ATeS objektid, ujumiskohad, lõkkeasemed, vms) on leitavad kasutaja poolt sisestatavate märksõnade järgi;
- Aadresside otsingus on võimalik märksõnadega piirata otsingut nii, et otsitav aadress on loendi alguses esimese 50 kirje hulgas. Kasutajale kuvatakse kuni 100 otsingu vastet;
- Maanteede otsingus on võimalik märksõnadega piirata otsingut nii, et otsitav maantee on loendi alguses esimese 10 kirje hulgas. Kasutajale kuvatakse kuni 20 otsingu vastet;
- Objektide otsingus on võimalik märksõnadega piirata otsingut nii, et otsitav objekt on loendi alguses esimese 30 kirje hulgas. Kasutajale kuvatakse kuni 50 otsingu vastet;
- Kõik otsingud tagastavad tulemuse nii, et kasutaja ei pea ootama. Päring saadetakse serverisse kui kasutaja teeb muudatuse märksõnade sisestamise väljale ning on muudatuse tegemise ajutiselt või lõplikult peatanud ning märksõnade sisestamise väljal on vähemalt kaks tähemärki. Märksõnade muutmise peatamine tähendab, et kasutaja ei ole 0,3 sekundi jooksul teinud märksõnade väljale muudatust. Kasutajale tuleb kuvada otsingu tulemus järgneva 0,2 sekundi jooksul, mis on jagatud kaheks võrdseks osaks – 0,1 sekundit on aega serveri tööks ning 0,1 sekundit brauseri poolseks tööks. Võrgu latentsust ei arvestata.
- Otsingusüsteemi kasutaja võib sisestada märksõnu mistahes järjekorras ning sellest ei sõltu otsingu tulemus.

Lisa 2 – Asukohtade uuendamise lähtekood

```
public void updateAsukoht() {
	List<Aadress> aadressUuendused = Aadress.findAllByTeisendatud(false)
	Date now = new Date()
	// Abistav list reindekseerimise jaoks
	List<Asukoht> muutunudAsukohad = new ArrayList()
	aadressUuendused.each { adr ->
		boolean alpha_numeric = false
		boolean alpha_korter = false
		Asukoht asukoht = Asukoht.get("ADR"+adr.id)
		if (adr.olek == 'T' && asukoht?.olek == 'K') { // Kehtetuks
		määratud aadress, mis asukoht tabelis on kehtiv
			if(!Adob.findByAdrId(adr.id)) { // kontroll, kas aadressi
			ja objekti vahel on seos
				asukoht.olek = "T"
				asukoht.muudetudKpv = new Date()
				adr.teisendatud = true
				asukoht.save()
				muutunudAsukohad.add(asukoht)
			} else {
				// adob seos alles, asukoht jääb kehtima
			}
		} else if (adr.olek == 'K') { // muutunud/lisandunud aadress on
		kehtiv
			if (!asukoht) { // kui pole tabelis, tuleb luua uus
			asukoht
				asukoht = new Asukoht()
				asukoht.muudetudKpv = now
				asukoht.id = "ADR" + adr.id
			}
			// põhiandmete täitmine
			asukoht.tase1 = adr.tase1
			asukoht.tase2 = adr.tase2
			asukoht.tase3 = adr.tase3
			asukoht.taisaadress = adr.taisaadress
			asukoht.lahiaadress = adr.lahiaadress
			asukoht.koordX = adr.koordX
			asukoht.koordY = adr.koordY
			if (adr.lahiaadress) { // lähiaadress olemas
			// kontrollitakse kas tegemist on alpha, numeric, või
			korter tüüpi lähiaadressiga
				def kontrollSonad = adr.lahiaadress.split(" ")
				kontrollSonad.each { sona ->
					if (sona.matches(".*\\d.*") &&
					sona.contains("-")) {
						alpha_korter = true
					} else if (sona.matches(".*\\d.*")) {
						alpha_numeric = true
					}
				}
				// täidetakse vastav veerg
				if (alpha_korter) {
					asukoht.lahiKorter = adr.lahiaadress
				} else if (alpha_numeric) {
```

```
                                asukoht.lahiNumeric = adr.lahiaadress
                        } else {
                                asukoht.lahiAlpha = adr.lahiaadress
                        }
                } else { // lähiaadress puudu, EHAK aadress
                // sõlutvalt viimase taseme olemasolust täidetakse vastav
                veerg
                        if (adr.tase3) {
                                asukoht.tase3Nimetus =
                                AadressKomponent.get(adr.tase3).nimetus
                        } else if (adr.tase2) {
                                asukoht.tase2Nimetus =
                                AadressKomponent.get(adr.tase2).nimetus
                        } else {
                                asukoht.tase1Nimetus =
                                AadressKomponent.get(adr.tase1).nimetus
                        }
                }
                asukoht.olek = "K"
                asukoht.muudetudKpv = now
                adr.teisendatud = true
                adr.save()
                asukoht.save()
                muutunudAsukohad.add(asukoht)
        }
    }
    // muduatused salvestatakse baasi
    def hibSession = sessionFactory.getCurrentSession()
    if(hibSession != null) hibSession.flush()
    Sql sql = new Sql(dataSource)
    // muutunud asukohtade geomeetria veeru täitmine
    sql.execute("update asukoht set the_geom =
    ST_SetSRID(ST_MakePoint(viitepunkt_x, viitepunkt_y), 3301) where
    the_geom is null and viitepunkt_x is not null;")
    // muutunud asukohtade reindekseerimine
    if (!muutunudAsukohad.isEmpty()) {
        Asukoht.reindex(muutunudAsukohad)
    }
}
```

Lisa 3 – Aadresside tasemete tähendused

Järgnev aadressi tasemete illustratsioon pärineb ADS andmekoguga liidestumise juhendist [13].