TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Katarina Neff 183336IAAM

AUTOMAATTESTIDE JUURUTAMINE E-RAHVASTIKUREGISTRI ARENDUSKESKKONNAS ELUKOHATOIMINGUTE NÄITEL

Magistritöö

Juhendaja: Nadežda Furs-Nižnikova
MBA

Tallinn
2020

Autorideklaratsioon

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

Autor: Katarina Neff

13.05.2020

Annotatsioon

Magistritöö eesmärk on luua Java programmeerimiskeeles esialgne automatiseeritud testide komplekt e-rahvastikuregistri iseteenindusportaali elukohatoimingutele ning sellega alustada automaattestide juurutamisega e-rahvastikuregistri iseteenindusportaali arenduskeskkonnas. Lisaks püstitab autor eesmärgi kasutada loodud automaattestide lahendust näitena teiste SMITi (Siseministeeriumi infotehnoloogia- ja arenduskeskuse) projektide automatiseerimiseks ning täiendada testimise protsessi automaattestidega seotud tegevustega.

Magistritöös teostatakse ärianalüüs, mille käigus uuritakse ja võrreldakse põhilisi automaattestide lahenduse loomiseks vajalikke tööriistu ja hinnatakse nende vajalikkust ning sobivust e-rahvastikuregistri elukohatoimingute teenuste testimise automatiseerimiseks.

Töö tulemusena kaardistatakse nõuded automaattestidele ja hiljem luuakse nende põhjal esialgne automatiseeritud testide komplekt, mida tulevikus on võimalik edasi arendada e-rahvastikuregistri teiste e-teenuste automatiseerimiseks või kasutada näiteprojektina teistes SMITi osakondades.

2020. aasta veebruaris alustati SMITis magistritöös analüüsitud automaattestide lahenduse juurutamist e-rahvastikuregistri elukohatoimingutele. Automaattestidega soovitakse katta kõik arendatud rahvastikuregistri iseteenindusportaali e-teenused 2020. aasta lõpuks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 52 leheküljel, 7 peatükki, 26 joonist, 10 tabelit.

Abstract

Implementation of Automated Tests in the Development Environment Based on Example of E-Population Register Residence Procedures

The objective of this master's thesis is to create a preliminary automated tests suite in Java programming language for e-population register residence procedures, thus starting the implementation of automated tests in the development environment of e-population register. Furthermore, the author of the thesis has a goal of using the automated tests suite as an example for the automation of other projects in SMIT (The IT and Development Centre of Ministry of the Interior in Estonia).

The thesis includes a business analysis which studies and compares the main tools needed for the automated tests suite, and studies their necessity and suitability for testing automation of e-population register residence procedures.

The author maps the requirements for automated tests and uses these to build a preliminary automated tests suite which can later be used to further develop the automation of other e-services in e-population register or to use the suite as an example project in other departments at SMIT.

In February 2020, SMIT has started implementing the automated tests suite developed for the purposes of this master's thesis in e-population register residence procedures. All developed e-services in e-population register portal are planned to be covered with automated tests by the end of 2020.

The thesis is written in Estonian language and consists of 52 pages including 7 chapters, 26 figures and 10 tables.

Lühendite ja mõistete sõnastik

Automaattest	Automatiseeritud testilugu, mis käivitatakse vastava automatiseerimise tööriistaga [9].
BPMN (Business Process Modelling and Notation)	Äriprotsesside modelleerimiskeel.
Disainimuster (Design pattern)	Üldine lahendus tarkvara kujundamise/arhitektuuri probleemile.
E2E (End-to-End) testimine	Testimise tüüp kõikide rakendusse integreeritud komponentide töötamise kontrollimiseks [16].
Elukohatoimingud	Elukohaandmete registreerimine ja muutmine [5].
JIRA	Atlassiani poolt loodud arendusprojektide haldamise tarkvara.
JIRA pilet	SMITi kontekstis on JIRA tarkvaras vormistatud kasutajalugu.
IDE	Integreeritud programmeerimiskeskkond.
Kasutajalugu	SMITis kasutusel olev nõuete kirjeldamise formaat JIRA keskkonnas.
MoSCoW	Prioritiseerimistehnika, mille abil saavad huvigrupid hinnata nõuete olulisust [38].
Nõue	Dokumenteeritud vajaduse kirjeldus selle kohta, milline süsteem või tulem peab olema, mida võimaldama ja kuidas toimima [9].
Pidev integratsioon	Tarkvara arendamise metoodika, milles tarkvara ehitamine, ühiktestid ning integratsioonitestid käivitatakse iga kord, kui uus kood repositooriumisse üles pannakse. See võimaldab leida esmased uute muudatustega seotud vead kiiresti [6].
Rahvastikuregistri menetlustarkvara	IKT teenus, mis võimaldab vastavalt pädevusele ja õigustele kasutajal teha päringuid isikuandmete, dokumentide, kannete ja toimikute kohta.
Regressioonitestimine	Testimismetoodika, mille eesmärgiks on kontrollida, et lahenduse olemasolev, varasemalt testitud ja töötanud funktsionaalsus töötaks endiselt peale täienduste lisamist ja muudatuste, paranduste tegemist. Tavaliselt teostatakse regressioonitestimist olemasolevate testilugude kordamisega [7].
Scrum	Iteratiivne ja inkrementaalne agiilse tarkvara arendamise raamistik. Määratleb paindliku ja tervikliku toote arendamise

	strateegia, kus arendusmeeskond töötab ühise meeskonnana ühise eesmärgi nimel [15].
Selenium Webdriver	Automaattestide tööriist veebilehitsejaga suhtlemiseks.
SMIT	Siseministeeriumi infotehnoloogia- ja arenduskeskus.
Testilugu	Testitava nõude kirjeldus koos eeltingimuste, teostuse detailsete sammude ja oodatud tulemi kirjeldustega [9].
Testimisraamistik	Kontseptsioonide ja tavade kogum, mis pakub tuge tarkvara automatiseeritud testimiseks [32].

Sisukord

Jooniste loetelu

Tabelite loetelu

Sissejuhatus

Töö käsitleb automaattestide juurutamist e-rahvastikuregistri iseteenindusportaali arenduskeskkonnas elukohatoimingute näitel. Aastast 2019 on rahvastikuregistri e-teenused koondatud uude veebikeskkonda, kus on võimalik vaadata oma andmeid ja registreerida elukohta [1]. 2020 märtsis lisandus sünni registreerimise teenus ja sügiseks on plaanis täiendada e-teenuste valikut teiste rahvastikuregistri teenustega, näiteks tõendite ja andmete küsimise teenusega ja ruumi registreeritud isiku elukohaandmete muutmise teenusega.

SMITi ja Siseministeeriumi jaoks on oluline pakkuda Eesti elanikele kvaliteetset ja kasutajasõbralikku veebikeskkonda rahvastikuregistri e-teenuste kasutamiseks. Lubades kasutajale senisest oluliselt lihtsamat ja kaasaegsemat viisi riigiga suhelda, on Siseministeerium seadnud SMITile eesmärgiks vastav lahendus luua [1].

E-rahvastikuregistri iseteenindusportaali uue lahenduse arendamiseks ja haldamiseks kasutatakse agiilse tarkvaraarenduse mudelit. Agiilse tarkvaraarenduse üks põhimõtetest on tarkvara tarnimine nii tihti kui võimalik, sest suur infosüsteem vajab pidevat täiendamist ja parandamist [2]. Agiilse töökorralduse puhul on automaattestidel eriti tähtis roll arendusprotsessi toetamisel ja tarkvara kvaliteedi loomisel, sest automaattestide olemasolu muudab tarkvara testimise efektiivsemaks ja kiiremaks [12].

SMITi rahvastikuteenuste osakonnas puudub magistritöö kirjutamise ajahetkel kokkupuude automaattestidega ning kogu testimise töö toimub manuaalselt ja seetõttu on magistritöö eesmärk läbi viia ärianalüüs, et oleks võimalik valida parim viis automaattestide juurutamisega alustamiseks.

Esimeses peatükis antakse ülevaade valdkonnast – SMITist, rahvastikuregistrist, e-rahvastikuregistri iseteenindusportaalist ja testija rollist e-rahvastikuregistri iseteenindusportaali arendusprotsessis. Lisaks kirjeldab autor olemasolevat testimise protsessi ja sõnastab probleemi, mille uurimise ja lahendamisega magistritöö raames tegeletakse.

Teises peatükis püstitatakse magistritöö eesmärk ja tuuakse välja magistritöö skoop. Lisaks antakse ülevaade autori rollist arendusprojektis ja kirjeldatakse teiste arendusprojekti kaasatud osapoolte rolle.

Kolmandas peatükis kirjeldatakse e-rahvastikuregistri iseteeninduspordaali projektis kasutatavat arendusmetoodikat ja põhjendatakse analüüsiks valitud analüüsimetoodikaid.

Neljandas peatükis analüüsitakse põhilisi automaattestide lahenduse loomiseks vajalikke tööriistu: arenduskeskkondi, testimisraamistikke ja disainimustreid. Analüüsi käigus seatakse kriteeriumid, millele iga automatiseerimise tööriist peab vastama, ja võrdluse põhjal valitakse välja elukohatoimingute automatiseerimiseks kõige sobivam tööriistakomplekt.

Viiendas peatükis teostatakse testilugude valik esialgse automatiseeritud testide komplekti jaoks. Eeltööna viib autor läbi intervjuud huvitatud osapooltega ning seejärel, intervjuude tulemuste põhjal, toimub automaattestide nõuete prioritiseerimine. Järgmisena teostab autor põhiliste äriprotsesside modelleerimise BPMN notatsioonis. Enne automaattestide lahenduse realiseerimist toimub automatiseerimiseks valitud testilugude koostamine. Viienda osa tulemusena luuakse esialgne automatiseeritud testide komplekt, mis hõlmab elukohatoiminguid.

Kuues peatükk sisaldab automatiseeritud testide lahenduse kirjeldust. Kirjeldatakse loodud automaattestide raamistiku üldist struktuuri ning visualiseeritakse automatiseeritud testide lahendus. Automatiseeritud testide leheobjektide struktuuri väljendamiseks kasutab autor UML klassidiagrammi. Loodud lahendusele lisatakse ettepanekud raamistiku haldamise osas. Lisaks teeb autor täiendusi rahvastikuteenuste osakonna testimise protsessis seoses automaattestide kasutusse võtmisega.

Seitsmendas peatükis annab autor hinnangu magistritööga saavutatud eesmärkidele. Lisaks toob autor välja järgmised sammud automaattestide edasiarendamise ja kasutusse võtmise osas mujal SMITis.

1 Probleemipüstitus

Käesolevas peatükis antakse ülevaade valdkonnast – SMITist, rahvastikuregistrist, e-rahvastikuregistri iseteeninduspordaalist ja testija rollist e-rahvastikuregistri iseteeninduspordaali arendusprotsessis. Lisaks kirjeldab autor olemasolevat testimise protsessi ja sõnastab probleemi, mille uurimise ja lahendamisega magistritöö raames tegeletakse.

1.1 Valdkonna ülevaade

Siseministeeriumi infotehnoloogia- ja arenduskeskus (edaspidi SMIT) on Siseministeeriumi hallatav riigiasutus, mille põhitegevus on Siseministeeriumi valitsemisala ülesannete täitmiseks vajalike info- ja kommunikatsioonitehnoloogia teenuste arendamine ja haldamine [3].

Lapse sünni, abielu registreerimise või lahutamise, uue isikunimi saamise ja elukoha muutusega seotud andmed kantakse SMITi poolt arendatavasse ja hallatavasse rahvastikuregistrisse [4]. Rahvastikuregister on andmekogu, mis koondab Eesti kodanike, Eestis elukoha registreerinud Euroopa Liidu kodanike ja Eestis elamisloa või elamisõiguse saanud välismaalaste peamisi isikuandmeid [5].

Rahvastikuregistrisse kantud andmeid kasutavad nii riigi ja kohalike omavalitsuste asutused kui ka Eesti elanikud ise avalike ülesannete täitmiseks või asjaajamise lihtsustamiseks vastavalt kehtestatud seadustele ning piirangutele. Korrektsed andmed rahvastikuregistris annavad riigile mitmesugust teavet oma kohustuste paremaks täitmiseks [5].

Kui varem sai rahvastikuregistri e-teenuseid kasutada eesti.ee portaali kaudu, siis alates detsembrist 2019 on avatud uus veebikeskkond, kuhu SMIT tulevikku vaatavalt koondab kõik rahvastikutoimingud [1]. Magistritöö kirjutamise hetkel on e-rahvastikuregistri iseteeninduspordaalis võimalik registreerida elukohta, vaadata ja uuendada isikuandmeid.

1.2 Testija roll rahvastikuteenuste osakonnas

Rahvastikuteenuste osakonna testija roll on eelkõige tagada loodava rakenduse kvaliteet kogu tootmisprotsessi jooksul. Testija vastutab kogu projekti kvaliteedi eest, alustades kasutuslugude ülevaatusest ning testilugude koostamisest kuni regressioonitestimiseni.

Osakonna testimise peamisteks eesmärkideks on loodud tarkvara mittefunktsionaalsete ja funktsionaalsete nõuete kontroll, probleemide ning riskide tuvastamine, hindamine ning nende maandamine. Testija on kohustatud avastatud veast raporteerima, kirjeldades lühidalt ja selgelt selle olemust, esitama detailsed sammud vea taastekitamiseks ning võimalusel andma ülevaate soovitatavast lahendusest [22].

Lisaks osaleb testija tööde planeerimise ja ülevaate koosolekutel, et olla võimalikult kiiresti kursis sellega, mida ja miks tellija tegelikult vajab.

1.3 Testimise tegevused arendustsüklis

Arendustsüklis teostatakse esmase testimistegevusena hindamiseks valitud kasutajalugude kvaliteedi kontroll. Kui JIRA pilet (edaspidi pilet) on hindamiseks valmis, siis teeb testija sellest ülevaate, mille käigus tutvutakse valminud nõuete kirjeldusega, vastuvõtukriteeriumidega ning uuritakse välja täpsustused. Pärast küsimustele vastuse saamisest täiendab analüütik või tooteomanik pileteid ning vajadusel edastab need testijale veel kord üle vaatamiseks. Eesmärk on juba enne piletite arendamist tagada võimalike ebakõlade kõrvaldamine, et vähendada arenduse ümbertegemisele ja parandamisele kuluvat aega [22]. Sellega ei kaasne projektile täiendavat ajakulu, sest enne tööde hindamist on testija sunnitud piletitega igal juhul tutvuma, et testimisele võimalikult täpset hinnangut anda.

Järgmiseks tegevuseks on testimismahu hindamine. Pärast piletitega tutvumist on juba paika pandud võimalikud testimisstrateegiad ning hindamisel valitakse vastavad testimise vahendid. Testija annab hinnangu järgmistele testimistegevustele: planeerimine, testilugude koostamine, manuaalne testimine ning paranduste verifitseerimine.

Hinnatud piletite põhjal alustab testija testilugude koostamist. Testiloo eesmärk on dokumenteerida, et arendatud või muudetud funktsionaalsus on realiseeritud vastavalt piletis kirjeldatud nõuetele ja töötab korrektselt [22]. Tavaline testilugu

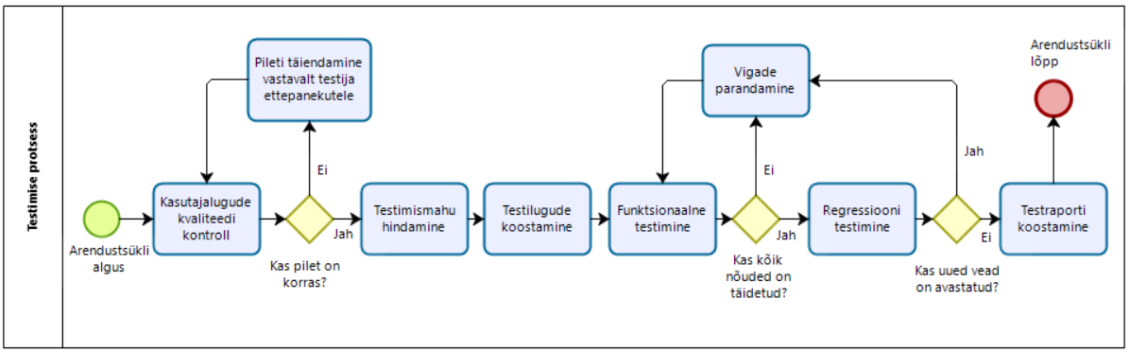
e-rahvastikuregistri iseteenindusportaali projektis koosneb pealkirjast, testimise eeltingimustest, testiloo sammude kirjeldusest, vastuvõtukriteeriumidest ja tegelikust tulemusest. Iga vastuvõtukriteeriumi kohta koostatakse vähemalt üks testilugu. Arvestatakse sellega, et iga testilugu peaks ära katma ka vähemalt ühe negatiivse stsenaariumi, mis tähendab, et kõik piletis kirjeldatud veateated peavad olema testilugudes kaetud. Autor lähtub sellest, et testilugu peab olema sedavõrd selge, et ka inimene, kes projekti ei tunne, saaks testimisega hakkama.

Kui esimesed piletid on testimiseks valmis, siis alustatakse funktsionaalse testimisega. Lisaks piletis kirjeldatud nõuetele pööratakse testimisel tähelepanu üldistele äriprotsessidele ning kasutajamugavusele. Liidestuse tekkimisel suunatakse fookus tehnilistele aspektidele [22]. Testid jooksutatakse vastavalt varem koostatud testilugudele. Testilugusid täidetakse JIRAs või lisatakse manusena testitava kasutajaloo juurde. Vigade raporteerimine käib samuti JIRAs. Kõik avastatud vead suunatakse võimalusel vastutavale arendajale, kes on testitava pileti funktsionaalsuse loonud. Kui selline suunamine ei ole võimalik, siis üritatakse hoida ühe ala vigu ühel ja samal arendajal [22] – et uus inimene ei peaks pikalt asjasse süvenema.

Kui kõik arendustsükli planeeritud tööd on arendatud ja testitud ning kõik avastatud vead on parandatud, siis alustatakse regressioonitestimisega. Regressioonitestimise eesmärk on eelkõige veenduda, et veebirakenduse uue versiooni integreerimine ei too endaga kaasa vigade tekkimist juba testitud koodis ning muutunud süsteem vastab endiselt nõuetele [8]. Magistritöö kirjutamise hetkel toimub kogu regressioonitestimise protsess manuaalselt.

Viimase testimistegevusena koostatakse testiraport, mis kirjeldab kõiki testimistegevusi viimase arendustsükli jooksul: kontrollitud kasutajalood, projekti seisu üldine kirjeldus, testija individuaalne hinnang testitud funktsionaalsusele ning soovitus tooteomanikule. Testiraport annab ülevaate ka kõikidest mõõdikutest, mille järgi projekti küpsust on hinnatud, ning märgatakse ära võimalikud probleemid ja riskid [22].

Allpool olev joonis illustreerib olemasolevat (AS-IS) testimise protsessi.



Joonis 1. Olemasolev (AS-IS) testimise protsess (Allikas: autori koostatud)

1.4 Probleemipüstitus

Iga arendatav toode või teenus on justkui projekt ning projekti olemuse järgi on sellel kolm olulist komponenti: aeg, raha ja kvaliteet. Need kolm komponenti on omavahel tihedalt seotud [10]. Sama mudel kehtib ka IT-arendusprojektide puhul.

Testide jooksutamise aja vähendamiseks, arendusprojekti kulude kokkuhoidmiseks ja kvaliteedi tõstmiseks peab testimise automatiseerimine olema projekti lahutamatu osa algusest peale [11].

SMITi rahvastikuteenuse osakonnas puudub magistritöö kirjutamise hetkel kokkupuude automaattestidega ning kogu testimise töö toimub manuaalselt. See on eriti kriitiline regressioonitestimise teostamisel, kuna iga uue funktsionaalsuse loomisega regressioonitestide arv suureneb ning sellega suureneb ka testija koormus ning pikeneb regressioonitestide täitmise aeg. Lisaks toob see e-rahvastikuregistri iseteenindusportaali projektile rahalist lisakulu, sest regressioonitestimise ajaks laenatakse tööressursse teistest rahvastikuteenuste osakonna projektidest.

Automaattestide juurutamisest on huvitatud nii rahvastikuteenuste arendusmeeskond ise kui ka tellijad.

Asutuse sees toimunud testijate ümarlaual selgus ka asjaolu, et SMITis on kasutusel palju erinevaid automaattestimisraamistikke, mida tihtipeale ei suudeta oma keerukuse tõttu edasi arendada ning automaattestide loomine on paljudes projektides peatatud. Seetõttu on vajadus luua SMITi sisemiseks kasutamiseks automatiseeritud testide näidisprojekti, mis oleks lihtsasti õpitav ja hästi dokumenteeritud, et ka teised SMITi testijad võiksid vajadusel oma projektides automaattestide juurutamisega alustada.

2 Töö eesmärk

Käesolevas peatükis püstitatakse magistritöö eesmärk ja tuuakse välja magistritöö skoop. Lisaks antakse ülevaade autori rollist arendusprojektis ja kirjeldatakse teiste arendusprojekti kaasatud osapoolte rolle.

2.1 Eesmärgi püstitus

SMITi ja Siseministeeriumi eesmärk on muuta e-teenused inimesele rahvastikuregistris paremini leitavaks ja kasutatavaks, pakkudes intuitiivset ja kasutajasõbralikku lahendust ning uue portaali avamisega teha riigiga suhtlemine veelgi lihtsamaks [1].

Seoses e-rahvastikuregistri uue iseteenindusportaali arendamisega on märkimisväärselt tõusnud vajadus automaattestide juurutamise järele, sest see aitab oluliselt kiirendada arendusprotsessi regressioonitestimise osas ja samas maksimaalselt vähendada erakorralisi tarkvarauuendusi, mis võivad olla põhjustatud arenduskeskkonnas avastamata vigadest.

Käeosoleva magistritöö eesmärk on läbi viia ärianalüüs ja luua esialgne automatiseeritud testide komplekt elukohatoimingutele, et oleks võimalik alustada automaattestide juurutamisega e-rahvastikuregistri arenduskeskkonnas ning sellega säästa meeskonna tööressursse, aega ja eelarvet.

Lisaks püstitab autor eesmärgi kasutada loodud automaattestide lahendust näitena teiste SMITi projektide automatiseerimiseks ning täiendada rahvastikuteenuste osakonna olemasolevat testimise protsessi automaattestidega seotud tegevustega.

2.2 Magistritöö skoop

Magistritöö skoopi kuulub eelkõige ärianalüüs, mille käigus uuritakse ja võrreldakse põhilisi automaattestimise tööriistu ning hinnatakse nende vajalikkust ning sobivust rahvastikuregistri teenuste testimise automatiseerimiseks. Lisaks kuulub magistritöö skoopi automaattestide osas nõuete kogumine ja prioritiseerimine ning e-rahvastikuregistri elukohatoimingutele esialgse automatiseeritud testide komplekti loomine, mida tulevikus oleks võimalik edasi arendada või kasutada näitena teiste SMITi

projektide automatiseerimiseks. Samuti sisaldab magistritöö skoop loodud lahenduse kirjeldust ja ettepanekuid automaattestide haldamise osas.

Magistritöö skoopi kuulub:

- automaattestimise tööriistade analüüs ja võrdlus;
- huvitatud osapoolte kaardistamine;
- intervjuud huvitatud osapooltega;
- automaattestide osas nõuete kogumine ja prioritiseerimine;
- elukohatoimingutega seotud peamiste äriprotsesside kirjeldamine ja modelleerimine BPMN notatsioonis;
- automatiseeritavate testilugude koostamine;
- e-rahvastikuregistri elukohatoimingutele esialgse automatiseeritud testide komplekti loomine;
- automaattestide leheobjektide klasside visualiseerimine UML klassidiagrammi abil;
- rahvastikuteenuste osakonna testimise protsessi täiendamine automaattestidega seotud tegevustega;
- autori ettepanekud loodud automaattestide haldamise osas.

Magistritöö skoopi ei kuulu:

- elukohatoimingute analüüs ja kasutajalugude kirjutamine;
- e-rahvastikuregistri iseteeninduspordaali arendusmetoodika ning programmeerimiskeele valik;
- automatiseerimise tööriistade valik veebilehitsejaga suhtlemiseks;
- automatiseeritud testide lahenduse sidumine pideva integratsiooni *(Continious Integration/CI)* vahenditega;
- automaatsete raportite lahenduse loomine;
- automaattestide loomine veebirakenduse mobiilsele versioonile;
- automatiseerimise edukuse mõõtmine ja hindamine erinevate mõõdikute abil.

2.3 Autori roll e-rahvastikuregistri iseteenindusportaali arendusprojektis

Magistritöö autor töötab SMITi rahvastikuteenuste osakonnas tarkvara testija ametikohal. Autori igapäevased tööülesanded testijana on kasutajalugude kvaliteedi kontroll, testimismahu hindamine, testilugude koostamine, funktsionaalne testimine, regressioonitestimine ja testimistulemuste analüüs. Lisaks püstitati autorile ülesanne luua Java programmeerimiskeeles esialgne automatiseeritud testide komplekt e-rahvastikuregistri iseteenindusportaali elukohatoimingutele.

Autoril on viieaastane töökogemus automaattestimise tööriistadega ja automaattestide juurutamisega nii era- kui ka avaliku sektori projektides.

Lisaks autorile on arendusprojekti kaasatud järgmistes rollides töötajad:

- tarkvaraarhitekt, kes vastutab tehnoloogia-alaste otsuste ja arendusvahendite valiku eest;
- tooteomanik, kes haldab ja viib ellu arendustegevuste plaani ning vastutab arendusmeeskonna töö eest;
- analüütik, kes analüüsib äriprotsesse ning tegeleb arenduseks vajalike spetsifikatsioonide ning nõuete lahtikirjutamisega;
- asutusesisesed ja -välised tarkvaraarendajad, kes tegelevad e-rahvastikuregistri portaali arendamisega;
- asutuseväline UX-disainer, kes analüüsib kasutajakogemust, disainib uusi lahendusi ja esitleb prototüüpi;
- nooremtestija, kes teostab manuaalset testimist ja tegeleb arendusprojektiga seotud dokumentatsiooni haldamisega.

3 Metoodikate ülevaade ja valik

Käesolevas peatükis kirjeldatakse e-rahvastikuregistri iseteenindusportaali projektis kasutatavat arendusmetoodikat ja põhjendatakse analüüsiks valitud analüüsimetoodikat.

3.1 Arendusmetoodika e-rahvastikuregistri iseteenindusportaali projektis

Metoodika on süstemaatiline viis millegi tegemiseks [13]. Alistair Cockburni definitsiooni järgi on tarkvara arendamise metoodika viis, kuidas tarkvara luuakse. Metoodika võib olla erineva detailsuse ja skoobiga. Detailsus näitab seda, kui põhjalikult ja täpselt on üldisi põhimõtteid ja väärtusi metoodikas kirjeldatud. Skoop näitab, kui suurt ulatust kogu arendustsüklist, rollidest ja tegevustest metoodika kirjeldab. Igal organisatsioonil on metoodika – see on viis, kuidas nad teevad oma tööd [14].

E-rahvastikuregistri uue iseteenindusportaali arendamiseks ja haldamiseks kasutatakse agiilse tarkvaraarenduse mudelit, kus tarkvara luuakse inkrementaalsetes arendustsüklites ning samm-sammult järgmistes etappides laiendatakse. Magistritöös käsitletavas projektis on agiilseks metoodikaks valitud Scrum – agiilse trakvara arendamise raamistik, mis põhineb iteratiivsetel ja inkrementaalsetel praktikatel [15].

E-rahvastikuregistri iseteenindusportaali projektis on arendustsükli pikkus üks nädal. Arendustsükli alguses toimub arendustööde planeerimise koosolek, kus vaadatakse üle JIRA piletites kirjeldatud ja töösse võetavad kasutajalood, mis peavad valmima vastava arendustsükli lõpuks. Seejärel toimub arendamiseks valitud tööde mahu hindamine, arendus ning testimine. Igal tööpäeval samal ajal toimub arendusmeeskonna lühike püstijalakoosolek, kus arutatakse eelmise tööpäeva saavutusi, tänaseid eesmärke ning uuritakse segavaid takistusi. Arendustsükli lõpuks korraldatakse arendatud tarkvara ülevaatus ning tagasivaatekoosolek, mille käigus arendusmeeskond hindab, milline tegevus läks hästi, milline ei läinud hästi ja mida saab järgmisel korral paremini teha. Seejärel alustatakse uue arendustsükliga.

3.2 Analüüsimetoodika valik

Analüüsida võib mida iganes – andmeid, olemeid, protsesse, sündmusi. Sõltuvalt sellest, mis on analüüsitavaks objektiks, rakendatakse vastavaid meetodeid. Ilma läbimõeldud lähenemiseta ei saavuta analüüsis just palju [17].

Analüüsi võib teostada mitmel tasemel – ärianalüüsi keskmes on protsessid ning antud laadi analüüs võib jääda kontseptuaalsele tasemele. Süsteemianalüüsi eesmärk on vaadata süsteemi sügavuti, kirjeldada nõudeid tehnilisemalt ja detailsemalt. Mõlema taseme analüüsid võimaldavad süsteemi uurida metoodiliselt ja aitavad seda lahti võtta komponentideks, mõistmaks iga osa tähtsust kogu süsteemi toimimises [17].

Käesolevas magistritöös keskendub autor ärianalüüsile, mille käigus uurib ja võrdleb põhilisi automaattestide tööriistu, kaardistab huvitatud osapooled ja viib läbi teemaintervjuud automaattestide lahenduse nõuete väljaselgitamiseks. Intervjuud toimuvad poolstruktureeritud vormis ehk teemad on ette antud, kuid küsimused pole eelnevalt täpselt sõnastatud ega järjestatud [18]. Intervjuude kokkuvõtted vormistatakse ja avaldatakse rahvastikuteenuste osakonna *Confluence Wiki* lehel. Valitud meetod nõuete kogumiseks annab autorile olulise sisendi automaattestide lahenduse realiseerimiseks ning aitab valida testilood esialgse automatiseeritud testide komplekti koostamiseks.

Intervjuude tulemuste põhjal toimub automaattestide nõuete prioritiseerimine. Selleks kasutab autor MoSCoW mudelit, mis aitab nõudeid prioritiseerida olulisuse seisukohalt ja jagada neid kategooriatesse [23]. Kogutud ja prioritiseeritud nõuded on aluseks elukohatoimingute automatiseerimiseks valitud testide komplekteerimisele. Järgmisena kirjeldab autor elukohatoimingutega seotud äriprotsesse. Peamiste elukohatoimingute äriprotsesside modelleerimiseks kasutatakse BPMNi (*Business Process Modelling and Notation*). Enne automaattestide lahenduse realiseerimist toimub automatiseerimiseks valitud testilugude koostamine.

Automaattestide testilugude leheobjektide klasside visualiseerimiseks kasutab autor UML (*Unified Modeling Language*) klassidiagrammi (*Class diagram*). See annab parima ülevaade loodud automaattestide struktuurist: klassidest ja kasutatud meetoditest.

4 Automatiseerimise tööriistade analüüs ja valik

Käesolevas peatükis uurib autor põhilisi automaattestide tööriistu, toob välja kriteeriumid, millele iga automatiseerimise tööriist peab vastama, ja võrdluse põhjal valib välja elukohatoimingute automatiseerimiseks kõige sobivama tööriistakomplekti. Viimases alapeatükis teeb autor elukohatoimingute automatiseerimiseks valitud tööriistade kohta kokkuvõtte.

Antud magistritöö kontekstis nimetab autor automatiseerimise tööriistadeks kogumit erinevatest automaattestide loomist toetavatest vahenditest ja tehnikatest, mis on abiks automaattestide lahenduse juurutamisel e-rahvastikuregistri iseteeninduspordaali projektis.

Töös käsitletakse ja võrreldakse järgmisi automatiseerimise tööriistu:

- arenduskeskkonnad automaattestide loomiseks Java programmeerimiskeeles – Eclipse IDE ja IntelliJ IDEA;
- testimisraamistikud – JUnit 5 ja TestNG;
- disainimustrid – Page Object Model (*POM)* ja Page Factory.

4.1 Arenduskeskkonnad automaattestide loomiseks Java programmeerimiskeeles

Käesolevas alapeatükis annab autor ülevaate järgmistest arenduskeskkondadest: Eclipse IDE, IntelliJ IDEA. Lisaks toob autor välja kriteeriumid, millele peab sobiv arenduskeskkond vastama, võrdleb neid ja põhjendab oma valikut.

4.1.1 Eclipse IDE

Eclipse IDE on IMBi poolt loodud vabavaraline arenduskeskkond, mis abistab arendajat koodi kirjutamisel ja vigade otsimisel. See võimaldab programmi käivitada, leida kompileerimisvigu, mugavalt koodis navigeerida ja koodi efektiivsemalt kirjutada [19].

Eclipse IDE on Eclipse'i avaliku litsentsi (*EPL*) v2 alusel täielikult avatud lähtekoodiga. Koodi haldab kogukonna juhitud mittetulundusühing (*The Eclipse Foundation*). Eclipse IDE pakub laiendatud pistikprogrammide ökosüsteemi ja paneb avatud lähtekoodiga

toetajad tundma, et nad kasutavad kogukonna jaoks kogukonna poolt välja töötatud tarkvara [20].

Mõned kasutajad määratlevad Eclipse'i rohkem kui pistikprogrammide kogumit, mitte eraldi IDEd, nii et arenduskeskkonna kohandamine vastavalt kasutaja konkreetsetele vajadustele võib algajale kasutajale osutada keeruliseks [20].

4.1.2 IntelliJ IDEA

IntelliJ on JetBrainsi poolt loodud Java arenduskeskkond, mis muudab koodi kirjutamise oluliselt mugavamaks, kui see lihtsalt Notepadi kasutades oleks. IntelliJ Community Edition keskendub Java ökosüsteemile, IntelliJ Ultimate Edition lisab võimalused ka veebirakenduste arendamiseks. IntelliJ peamised eelised on: levinud vigade ja probleemide jooksvalt välja toomine, muutujate ja funktsioonide nimede automaatne täitmine, kiire ja mugav koodis navigeerimine [19].

Kuigi Apache 2.0 litsents piirab mõnevõrra pistikprogrammide ja laienduste kasutamist, on IntelliJ IDEA eeliseks siiski see, et koodi haldab korporatsioon, mitte kogukonna juhitud mittetulundusühing [20].

2018. aasta lõpus avaldas Java Magazine 10 500 Java-arendaja uuringu tulemused. Selle uuringu kohaselt kasutab 45% Java arendajatest mõnda IntelliJ IDEA versiooni ja ainult 38% Eclipse'i. Võib öelda, et iga 10 Eclipse'i kasutaja kohta on 12 IntelliJ IDEA kasutajat [21].

Kuigi ühe IDE populaarsus teise ees ei tohiks olla peamiseks valikukriteeriumiks, võib ikkagi eeldada, et Java arendajate kogukonna suurus ja kasutajate arv võib mõjutada tootlikkust ja kasutusmugavust [20].

4.1.2.1 Arenduskeskkondade võrdlus

Tuginedes eelnevalt kogutud informatsioonile ja isiklikule töökogemusele Eclipse'i ja IntelliJ IDEA arenduskeskkondadega erinevate projektide automatiseerimiseks, toob autor välja järgmised nõuete punktid, millega peab e-rahvastikuregistri iseteeninduse portaali projekti automatiseerimiseks arenduskeskkonna valimisel arvestama:

- tarkvara maksumus;
- litsents;

- kasutatavus;
- pistikprogrammide ökosüsteem;
- kogukonna suurus ja kasutajate arv;
- tarkvara kasutamine on SMITis kooskõlastatud.

Kogutud informatsiooni põhjal koondab autor arenduskeskkondade Eclipse IDE ja IntelliJ IDEA võrreldud nõuete punktid tabelisse 1.

Tabel 1. Java arenduskeskkondade võrdlus (Allikas: autori koostatud).

Kriteerium	**Eclipse IDE**	**IntelliJ IDEA**
Maksumus	Vabavaraline	IntelliJ Community Edition – vabavaraline; IntelliJ Ultimate Edition – 499 eurot aastas kasutaja kohta
Litsents	EPL v2	Apache 2.0
Kasutatavus	Mõeldud pigem kogenud kasutajale	Sobib pigem algajale
Pistikprogrammide ökosüsteem	TestNG, JUnit 5 toetus	TestNG, JUnit 5 toetus
Kogukonna suurus ja kasutajate arv	The Eclipse Foundation kogukond; 38% kasutajat Java arendajatest vastavalt 2018 toimunud Java Magazine'i uuringule	IntelliJ IDEA kogukond; 45% kasutajat Java arendajatest vastavalt 2018 toimunud Java Magazine'i uuringule
Tarkvara kasutamine on SMITis kooskõlastatud	Jah	Jah

Kokkuvõtvalt saab välja tuua, et kuigi mõlemad arenduskeskkonnad toetavad kõiki populaarsemaid testimisraamistikke ja mõlema tarkvara kasutamine on SMITis kooskõlastatud, sobib rahvastikuteenuste osakonna testijatele kasutatavuse seisukohast siiski rohkem IntelliJ IDEA arenduskeskkond. Valiku tegemisel arvestas autor sellega, et e-rahvastikuregistri iseteeninduspordaali projekti on kaasatud ka nooremtestija, kes avaldas soovi tulevikus liituda automaattestide arendamisega, seega valitud arenduskeskkond peaks olema võimalikult lihtne ning hästi sobima algajale kasutajale. Lisaks kasutavad IntelliJ IDEA arenduskeskkonda ka e-rahvastikuregistri projekti arendajad, kelle kogemust võib kasutada arenduskeskkonna seadistamiseks projekti

koodi repositooriumiga. Rahvastikuteenuste osakonna juhataja toetas samuti IntelliJ Ultimate Edition versiooni valikut automaattestide arendamiseks.

4.2 Testimisraamistikud

Käesolevas alapeatükis annab autor ülevaate järgmistest automaattestimise raamistikest: JUnit 5, TestNG. Lisaks toob autor välja kriteeriumid, millele peab sobiv testimisraamistik vastama, võrdleb neid ja põhjendab oma valikut.

4.2.1 JUnit 5

JUnit 5 on avatud lähtekoodiga Java testimisraamistik, mida kasutatakse veebirakenduste testimise automatiseerimiseks [25].

Automaattestide koostamisel peab sageli enne testi täitmist täitma mõned konfiguratsiooni- või initsialiseerimisjuhised ja pärast testide lõpetamist ka tegema puhastuse [24].

JUnit pakub initsialiseerimist ja puhastamist kahel tasemel, enne ja pärast iga meetodit ja klassi. Meetodi taseme märkused on *@BeforeEach*, *@AfterEach* ja klassitasemel on *@BeforeAll* ja *@AfterAll* märkused [24].

JUnit 5 versioonis on nüüd saadaval uus funktsioon, mis võimaldab testjuhtumite integreerimist ja seejärel nende paralleelset käitamist testkomplektina [27]. Varem oli see funktsionaalsus saadaval vaid TestNG jaoks.

Kogukonna toetamise osas on JUnit'il pikk ja parem ajalugu ning ulatuslikum kasutajaskond, kuna see oli esimene laialdaselt kasutatav üksuste testimise raamistik [27].

4.2.2 TestNG

TestNG (*Test Next Generation*) on avatud lähtekoodiga Java testimisraamistik, mis on inspireeritud JUnit'ist ja NUnit'ist, kuid see pole JUnit'i laiendus ega täiendatud versioon [27]. TestNG oli spetsiaalselt loodud JUnit'i funktsionaalsuse piirangute ületamiseks [25].

TestNG ehitati paindlikumaks kui JUnit ja see võib hõlmata peaaegu kõiki tarkvara testimise kategooriaid: üksused (*unit*), integratsioon, E2E (*End-to-End*) [27].

Sarnaselt JUnit'iga pakub TestNG ka initsialiseerimist ja puhastamist meetodi ja klassi tasemel. Klassi tasemel kasutatakse *@BeforeClass* ja *@AfterClass* märkuseid, meetodi taseme märkused on *@BeforeMethod* ja *@AfterMethod* [24].

Lisaks pakub TestNG ka spetsiaalseid märkusi, näiteks *@BeforeSuite*, *@AfterSuite*, *@BeforeGroups*, *@AfterGroups* testide koosseisude ja rühmatasemete konfiguratsioonide jaoks [24], mis lubab jagada teste paindlikult rühmadesse ning välistada vajaduse midagi uuesti kompileerida, säästes sellega nii aega kui ka raha [27].

TestNG on QA (*Quality Assurance*) analüütikute eelistatud raamistik, kuna see võimaldab genereerida testiaruandeid nii HTML- kui ka XML-vormingus [32].

TestNG kasutajaskond võib olla väiksem, kuid ei saa öelda, et kogukonna tugi oleks seetõttu nõrgem [27].

4.2.2.1 Testimisraamistike võrdlus

Lähtudes eelnevalt kogutud informatsioonist ning oma varasemast töökogemusest JUnit ja TestNG testimisraamistikega, toob autor välja järgmised nõuded, millega peab automaattestimise raamistiku valimisel arvestama:

- testide konfiguratsioon;
- märkuste valik;
- testide käitamine testikomplektina;
- testide rühmitamine;
- kogukonna tugi.

Kogutud informatsiooni põhjal koondab autor testimisraamistike JUnit ja TestNG võrreldud nõuete punktid tabelisse 2.

Tabel 2. Testimisraamistike võrdlus (Allikas: autori koostatud)

Kriteerium	**JUnit 5**	**TestNG**
Testide konfiguratsioon	Klassi tasemel: *@BeforeAll* ja *@AfterAll*, meetodi tasemel: *@BeforeEach ja @AfterEach*	Klassi tasemel: *@BeforeClass* ja *@AfterClass*, meetodi tasemel: *@BeforeMethod* ja *@AfterMethod*

Kriteerium	JUnit 5	TestNG
Märkuste valik	Ei toeta spetsiaalseid märkusi koosseisude ja rühmatasemete konfiguratsioonide jaoks	Toetab spetsiaalseid märkusi koosseisude ja rühmatasemete konfiguratsioonide jaoks
Testide käitamine testikomplektina	Jah	Jah
Testide rühmitamine	Ei	Jah
Kogukonna tugi	GitHub, StackOverflow	GitHub, StackOverflow

JUnit ja TestNG on kahtlemata kaks kõige populaarsemat automaattestimise raamistikku Java ökosüsteemis, mida saab kasutada Selenium Webdriver'iga. Mõlemad on oma olemuselt ja käitumiselt sarnased testimisraamistikud, aga TestNG ületab JUnit'i piiranguid täiendavate funktsioonide ja spetsiaalsete märkustega, mida JUnit ei toeta, seega lähtudes projekti vajadusest käivitada automaatteste mõnedel juhtudel kindlas järjekorras ja sõltuvuses, osutus autori valik TestNG testimisraamistiku kasuks.

4.3 Disainimustrid

Käesolevas alapeatükis annab autor ülevaate järgmistest automaattestide disainimustritest: Page Object Model (*POM),* Page Factory. Lisaks toob autor välja kriteeriumid, millele peab sobiv disainimuster vastama, võrdleb neid ja põhjendab oma valikut.

4.3.1 Page Object Model (POM)

Page Object Model (*POM*) on disainismuster veebi kasutajaliidese elementide objektihoidla loomiseks. Selle mudeli kohaselt peaks rakenduse igal veebilehel olema vastav lehekülje klass. Selle klassi muutujad ja meetodid vastavad kõigile lehe veebi elementidele ja äriloogika toimingutele [29].

Page Object Model'i kontseptsioon seisneb selles, et kasutajaliideses tehtavad toimingud ja vood tuleks kinnitamisest lahutada. See muudab koodi puhtamaks ja hõlpsasti mõistetavaks. Objektihoidla on testjuhtumitest sõltumatu, seega saab sama objektihoidlat kasutada erinevatel eesmärkidel erinevate tööriistadega [29] või hoopis mõne muu projekti jaoks.

Lokaatorite määratlemiseks kasutatakse *By* käsku ja iga lehtobjekti peab individuaalselt lähtestama.

Lehekülјeobjekti mudelit saab kasutada mis tahes tüüpi raamistikus, näiteks modulaarses, andmepõhises, märksõnapõhises, hübriidses raamistikus jne [30].

POM kontseptsiooni eelisteks on koodi korduvkasutatavus, hooldatavus ja loetavus [30].

4.3.2 Page Factory

Page Factory on optimeeritud viis objektihoidla loomiseks POM kontseptsiooni järgi [29].

Veebilehed on esitatud klassidena ja lehe elemendid on määratletud klassis muutujatena, nii et kasutajate interaktsioone saab seejärel klassis meetoditena rakendada [29].

Leheobjektide määratlemiseks kasutatakse *FindBy* annotatsiooni ja kõik leheobjektid lähtestatakse, kasutades meetodit *initElements()* [30].

Üheks Page Factory suurimaks eeliseks on *AjaxElementLocatorFactory* klass. See töötab laisa laadimise (*lazy loading*) kontseptsiooni järgi, mis tähendab, et ajaveebide elemendi aegumine omistatakse *AjaxElementLocatorFactory* abil leheobjekti klassile. Seega, kui elemendiga tehakse toiming, algab selle nähtavuse ootamine alles sellest konkreetsest hetkest. Kui elementi antud ajaraamis ei leita, loobub testjuhtumi täitmine erandist [31].

4.3.2.1 Disainimustrite võrdlus

Järgmisena toob autor välja nõuete punktid, millega peab disainimustri valimisel arvestama:

- leheobjektide lähtestamise viis;
- lokaatorite määratlemine;
- laisa laadimise (*lazy loading)* kontseptsioon.

Kogutud informatsiooni põhjal koondab autor disainimustrite POM ja Page Factory võrreldud nõuete punktid tabelisse 3.

Tabel 3. Disainimustrite võrdlus (Allikas: autori koostatud)

Kriteerium	**Page Object Model (POM)**	**POM + Page Factory**
Leheobjektide lähtestamise viis	Iga lehtobjekti peab individuaalselt lähtestama	Kõik leheobjektid lähtestatakse, kasutades meetodit *initElements()*
Lokaatorite määratlemine	*By* käsu abil	*FindBy* annotatsiooni abil
Laisa laadimise kontseptsioon	Ei	Jah

Page Object Model'i (*POM*) toetamiseks otsustas autor kasutada Page Factory't, sest see on leheobjekti täiustatud laiendus. POM ja Page Factory koos kasutamine muudab automaattestide koostamise hõlpsamaks, võimaldades annotatsioonide abil lähtestada konkreetsed elemendid leheobjekti mudelis. Üks kord loodud *Page Object* klasse võib tulevikus uuesti kasutada teiste rahvastikuteenuste osakonna projektide jaoks.

4.4 Elukohatoimingute automatiseerimiseks valitud tööriistakomplekti kokkuvõte

Testide automatiseerimise edu seisneb suurel määral automatiseerimiseks sobivate tööriistade leidmises. Automatiseeritava projekti jaoks õige tööriistakomplekti kaardistamine on üks parimaid viise automaattestide juurutamise eesmärgi saavutamiseks. Mõeldes, milliseid automatiseerimise tööriistu valida, uuris ja analüüsis autor populaarsemaid arenduskeskkondi, testimisraamistikke ja disainimustreid ning seatud kriteeriumide põhjal tegi põhjendatud valiku järgmise tööriistakomplekti kasuks:

- IntelliJ Ultimate Edition'i arenduskeskkond;
- TestNG testimisraamistik;
- Page Factory disainismuster.

Kokkuvõtvalt saab välja tuua, et käesoleva magistritöö eesmärkide saavutamiseks loob autor Java programmeerimiskeeles e-rahvastikuregistri elukohatoimingutele esialgse automatiseeritud testide komplekti, kasutades selleks Selenium Webdriver'it koos TestNG testimisraamistikuga ja PageFactory disainimustriga. Koodi kirjutamine ja haldamine toimub IntelliJ Ultimate Edition'i arenduskeskkonnas.

5 Ärianalüüsi tulemused

Käesolevas peatükis kaardistab autor huvitatud osapooled ja viib läbi poolstruktureeritud intervjuud, mille tulemusena defineeritakse nõuded esialgsele automatiseeritud testide komplektile; seejärel toimub MoSCoW analüüsimeetodi abil nõuete prioritiseerimine. Järgmisena koostatakse elukohatoimingute peamiste äriprotsesside kirjeldus ning teostatakse üldiste äriprotsesside modelleerimine BPMN notatsioonis. Enne automaattestide lahenduse realiseerimist toimub automatiseerimiseks valitud testilugude koostamine. Viienda osa tulemusena luuakse esialgne automatiseeritud testide komplekt, mis hõlmab valitud elukohatoiminguid.

5.1 Huvitatud osapooled

Käesoleva magistritöö kontekstis on huvitatud osapooled e-rahvastikuregistri iseteeninduspordaali projektiga seotud inimesed (rühmad või üksikisikud), kellel on volitusi mõjutada või keda mõjutab planeeritav automaattestide lahendus.

Mindtools.com on oma veebiõpetuses [33] toonud välja huvitatud osapooltel põhineva lähenemise neli peamist eelist:

- projekti väljatöötamise varajases etapis kogutud arvamused võivad parandada projekti üldist kvaliteeti;
- huvitatud osapoolte toetuse saamine võib aidata võita rohkem ressursse, näiteks inimesi, aega või raha;
- suhtlemisel saab tagada, et huvitatud osapooled mõistavad täielikult planeeritud tegevusi ja projekti eeliseid;
- huvitatud osapooled saavad üldise ettekujutuse loodavast lahendusest.

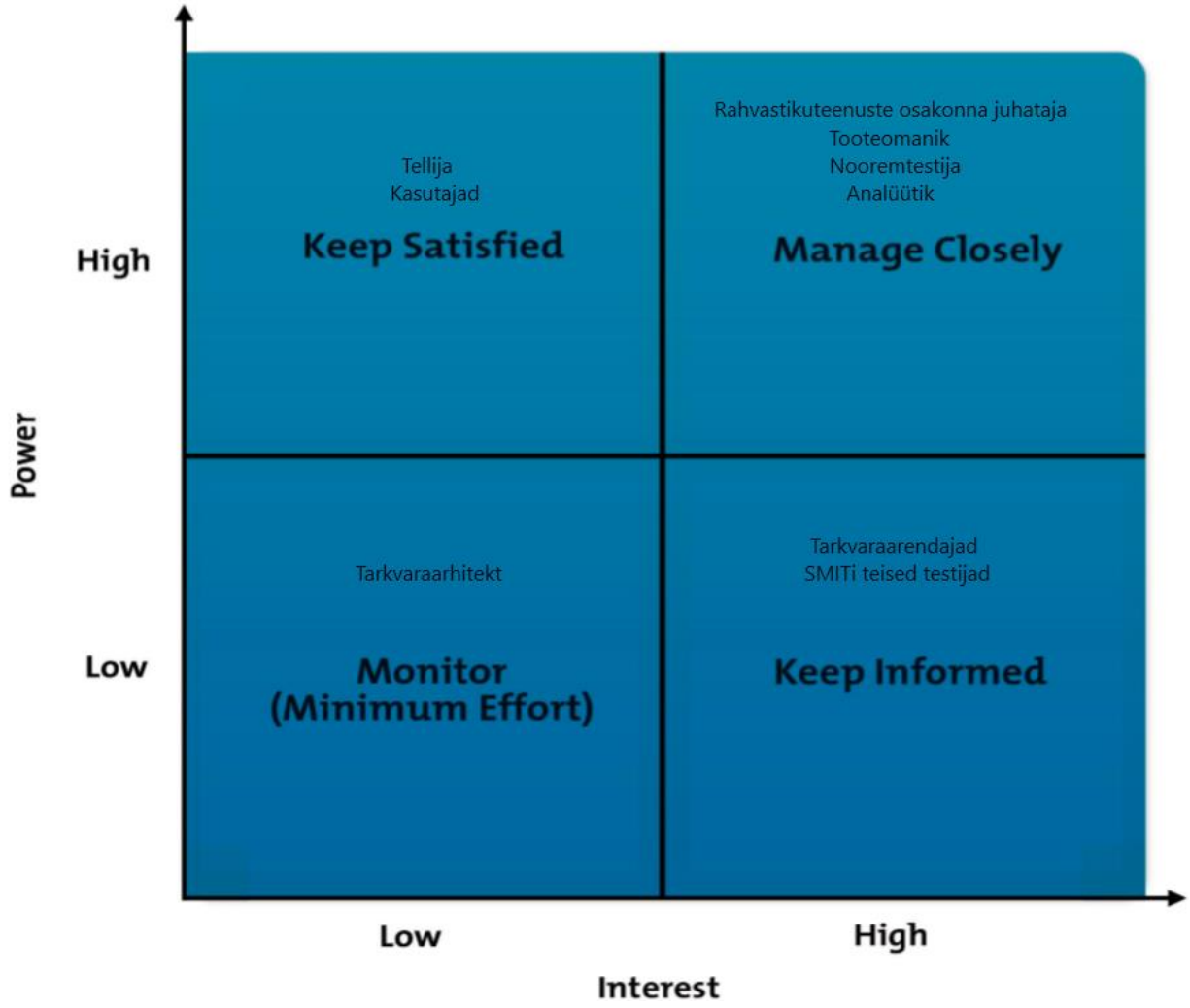
Huvitatud osapoolte analüüsis tuleb järgida kolme sammu. Esiteks määratakse kindlaks, kes on huvitatud osapooled. Järgmisena uuritakse välja nende huvi planeeritava lahenduse osas ja lõpuks luuakse arusaam kõige olulisematest huvitatud osapooltest, et saaks prognoosida, keda planeeritav lahendus kõige rohkem mõjutab [33].

Allolev tabel määratleb huvitatud osapooled, kellel on huvi rahvastikuteenuste osakonnas automaattestide eduka juurutamise osas:

Tabel 4. Huvitatud osapooled (Allikas: Autori koostatud)

Huvitatud osapool	**Huvi automaattestide lahenduse osas**
Rahvastikuteenuste osakonna juhataja	Vastutus e-rahvastikuregistri iseteenindusportaali projekti eesmärkide täitmise ja tulemuste eest.
Tooteomanik	Regressioonitestimisega seotud kulude vähendamine projektis.
Analüütik	Kiirem tagasiside muudetud funktsionaalsuse osas. Testandmete genereerimine automaattestide abil.
Tarkvaraarendaja(d)	Kindlustunne koodi muutmisel.
Tarkvaraarhitekt	Automaattestide pideva integratsiooni eelduse võimaldamine.
E-rahvastikuregisti iseteenindusportaali projekti nooremtestija	Võimalus testide automatiseerimine ära õppida.
SMITi teised testijad	Loodud automaattestide lahenduse kasutamine oma projektides.
Siseministeeriumi rahvastiku toimingute osakond ehk tellijad	Kvaliteedi tagamise lisategevuste ettevõtmine tarkvara arendamisel.
E-rahvastikuregisti iseteenindusportaali kasutaja(d)	Tarkvara kvaliteet ehk stabiilselt toimiv e-rahvastikuregistri iseteenindusportaal.

A.L. Mendelow [34] on toonud välja lihtsa viisi huvitatud osapoolte huvi ja mõju taseme kokkuvõtmiseks, soovitades kasutada spetsiaalse diagrammi huvitatud osapoolte prioritiseerimiseks:



Joonis 2. Kohandatud Mendelow' diagramm huvitatud osapoolte prioritiseerimiseks (Allikas: autori koostatud)

5.2 Intervjuud

Autor viis 2019. aasta detsembris läbi poolstruktureeritud intervjuud projekti peamiste huvitatud osapooltega. Peamised huvitatud osapooled said eelnevalt valitud, lähtudes mõju ja huvi prioritiseerimise põhimõttest. Vastavalt kohandatud Mendelow' diagrammile (vt. joonis 2. Kohandatud Mendelow' diagramm huvitatud osapoolte

prioritiseerimiseks) on automaattestide juurutamise lahenduse tugevamad toetajad järgmised osapooled:

- rahvastikuteenuste osakonna juhataja;
- tooteomanik;
- projekti nooremtestija;
- analüütik.

Intervjuud viidi läbi suuliselt, kohtudes iga huvitatud osapoolega eraldi. Intervjuude kohta vormistati kokkuvõte ja see avaldati rahvastikuteenuste osakonna *Confluence Wiki* lehel. Kohtumiste eesmärk oli saada sisend oodatud esialgse elukohatoimingute automatiseeritud testide komplekti lahenduse osas ning selleks said eeldefineeritud järgmised alamteemad:

- elukohatoimingute testimise automatiseerimise skoop;
- ärikriitilised testilood;
- testide detailsus ja tüüp;
- valideerimine;
- raporteerimine;
- veebilehetsejad testide käivitamiseks;
- võimalik esialgne automatiseeritud testide komplekt.

Intervjuude tulemuste põhjal teostati analüüs, millest selgusid nõuded esialgse elukohatoimingute automatiseeritud testide komplekti osas, millest annab autor ülevaate järgmises alapeatükis.

5.3 Nõuded esialgsele elukohatoimingute automatiseeritud testide komplektile

Autor jagas nõuded esialgsele elukohatoimingute automatiseeritud testide komplektile kahte rühma: nõuded automatiseeritud testidele äriprotsessi osas ja üldised nõuded automaattestide osas.

Peamiste huvitatud osapooltega läbi viidud poolstruktureeritud intervjuude tulemuste põhjal selgunud nõuded on koondatud tabelisse 5:

Tabel 5. Nõuded esialgsele elukohatoimingute automatiseeritud testide komplektile (Allikas: autori koostatud)

Tähis	**Nõuded automatiseeritud testidele äriprotsessi osas**
ÄN1	Esialgne automatiseeritud testide komplekt peab sisaldama vähemalt ühte põhivoo testi, mis on seotud elukohateate esitamise protsessiga
ÄN2	Esialgne automatiseeritud testide komplekt peab sisaldama vähemalt ühte alternatiivvoo testi, mis on seotud elukohateate esitamise protsessiga
ÄN3	Esialgne automatiseeritud testide komplekt peab sisaldama vähemalt ühte põhivoo testi, mis on seotud e-nõusoleku andmise protsessiga
ÄN4	Esialgne automatiseeritud testide komplekt peab sisaldama vähemalt ühte alternatiivvoo testi, mis on seotud e-nõusoleku andmise protsessiga
ÄN5	Esialgne automatiseeritud testide komplekt peab sisaldama vähemalt ühte kohustuslike andmete lisamise testi
ÄN6	Esialgne automatiseeritud testide komplekt peab sisaldama vähemalt ühte kolmanda isiku lisamise testi
Tähis	**Üldised nõuded automaattestide osas**
ÜN1	Automaatteste peab saama käivitada viimastes Google Chrome'i ja Mozilla Firefoxi veebilehitsejates
ÜN2	Automaattestide tulemuste põhjal peab saama genereerida raportit
ÜN3	Automaatiseerimisele kuuluvad ainult need kasutuslood, mis on lõpuni arendatud
ÜN4	Automatiseerimiseks valitud testilood peavad olema dokumenteeritud
ÜN5	Automaattestide koodi hoidmiseks tuleb kasutada põhiprojekti BitBucket repositooriumi

Tabelis 5 kirjeldatud nõuded on aluseks esialgse automatiseeritud testide komplekti planeerimisele.

5.4 Nõuete prioritiseerimine

Esialgse automaattestide komplekti planeerimisel on oluline luua selge arusaam huvitatud osapoolte nõudmistest ja nende prioriteedist. Nõuete prioritiseerimine aitab kõigil projekti kaasatud osapooltel mõista kõige olulisemaid nõudeid, mis järjekorras neid realiseerida ja millest võib loobuda, kui teatud nõude realiseerimine pole otstarbekas.

Nõuete prioritiseerimiseks kasutab autor MoSCoW analüüsimeetodit, mis aitab otsustada, millised nõuded on täitmiseks kõige tähtsamad, milliste nõuete täitmisega võib oodata ja millised nõuded võib hetkel täiesti välistada.

Alltoodud tabelisse on koondatud huvitatud osapoolte poolt prioritiseeritud nõuded:

Tabel 6. Prioritiseeritud nõuded MoSCoW mudeli järgi (Allikas: autori koostatud)

Ärinõue	***Must have***	***Should have***	***Could have***	***Won`t have***
ÄN1	X			
ÄN2				X
ÄN3	X			
ÄN4				X
ÄN5	X			
ÄN6	X			
ÜN1	X			
ÜN2				X
ÜN3		X		
ÜN4	X			
ÜN5			X	

Nõuete prioritiseerimisel lähtuti sellest, et esialgne automatiseeritud testide komplekt võiks sisaldada eelkõige elukohateate esitamise ja e-nõusoleku andmise põhivoo teste: ÄN1 ja ÄN3 (vt tabel 5. Nõuded esialgsele elukohatoimingute automatiseeritud testide komplektile). Kõikide alternatiivvoo testide osas jõuti kokkuleppele, et nende koostamine toimub hiljem läbi põhivoo testide automatiseerimise käigus loodud komponentide taaskasutamise, seega selliseid teste ei ole otstarbekas lisada esialgsele automatiseeritud testide komplektile.

Lisaks esialgsele automaattestide komplektile otsustati lisada ärikriitilised testilood, mis on seotud e-posti uuendamise funktsionaalsusega ja hooldusõiguse valideerimise äriloogikaga.

Automaattestide üldiste nõuete osas valiti täitmiseks välja järgmised: automatiseerimisele kuuluvad ainult need testilood, mille arendus on juba lõppenud ning automatiseeritud teste peab saama käivitada, kasutades Google Chrome'i ja Mozilla Firefoxi veebilehitsejaid. Kõikidele automatiseerimiseks valitud testilugudele luuakse testilugude mallid ning koodi hoidmiseks kasutatakse põhiprojekti BitBucket repositooriumi.

Automaatsete raportite genereerimist käesoleva magistritöö raames ei tehta.

5.5 Äriprotsesside kirjeldused

Tuginedes intervjuude käigus uuritud elukohatoimingute testimise automatiseerimise skoobile, koostas autor peamiste äriprotsesside kirjeldused ja diagrammid BPMN notatsioonis.

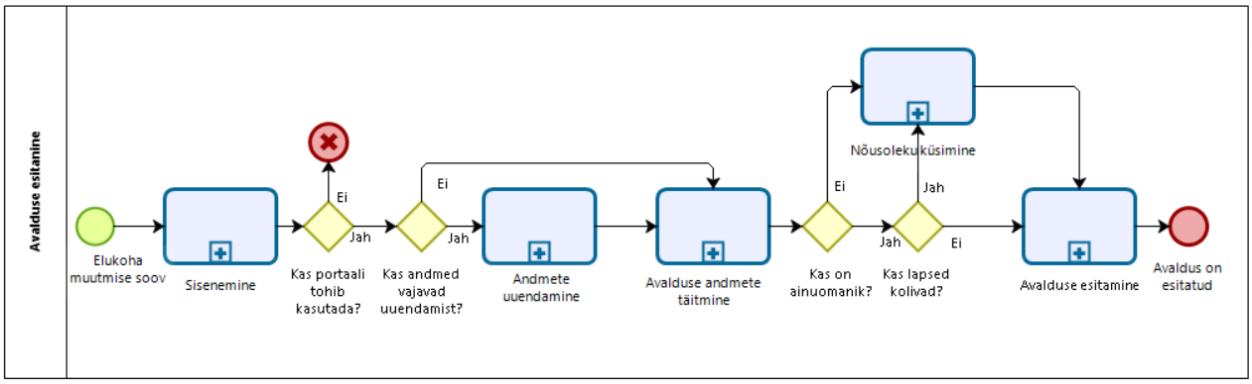
5.5.1 Üldprotsess

Elukoha registreerimise protsess algab elukohateate avalduse esitamisest. Kui avalduse esitamise käigus ilmneb nõusoleku küsimise kohustus, siis elukoha registreerimise protsess jätkub elukohateate e-nõusoleku andmisega.

Nõusoleku küsimise kohustus tekib siis, kui avaldaja on üürniku rollis või kui avaldusele on kaasatud isikud, kes ei kuulu avaldajaga seotud isikute hulka. Lisaks peab nõusolekut küsima, kui elukohta muutvate isikute nimekirjas on alaealised lapsed, kelle kohta kehtib ühine hooldusõigus.

Kui avalduse esitamisega ei kaasne nõusoleku küsimise kohustust, siis peale andmete kinnitamist suunatakse avaldus otse ametniku töölauale.

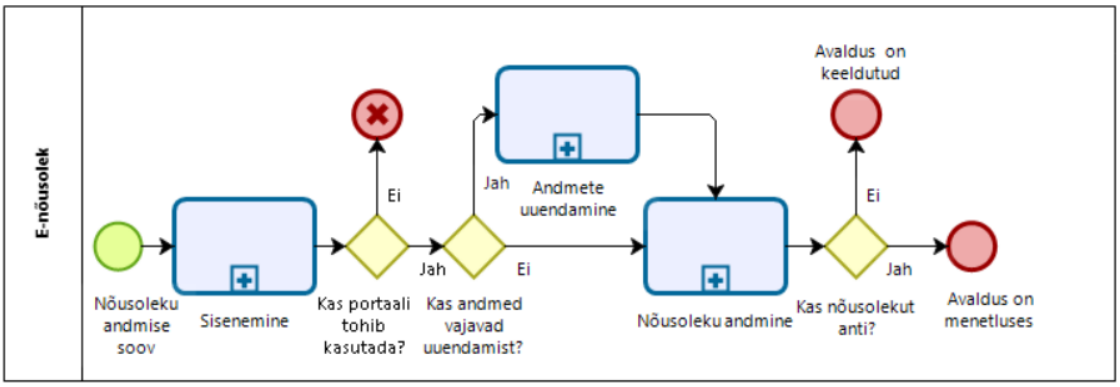
Joonis 3 illustreerib elukohateate avalduse esitamise üldprotsessi:



Joonis 3. Elukohateate avalduse esitamise üldine protsess (Allikas: autori koostatud)

Kui elukohateate esitamisega on tekkinud nõusoleku küsimise kohustus, siis esitatud avaldus jõuab ametniku töölauale ainult peale avalduse kinnitamist nõusoleku andja poolt.

Joonis 4 illustreerib e-nõusoleku andmise üldprotsessi:



Joonis 4. E-nõusoleku andmise üldine protsess (Allikas: autori koostatud).

Järgnevates alapeatükkides annab autor põhjalikuma ülevaate elukohateate esitamise ja e-nõusoleku andmise põhivoogudest.

5.5.2 Elukohateate esitamise põhivoog

Elukohateate esitamiseks peab kasutaja end eelnevalt autentima. Sisenemisel teostatakse kasutaja kohta rahvastikuregistri (edaspidi RR) päring, et tuvastada, kas isikul on lubatud e-rahvastikuregistri iseteeninduspportaali kasutada. Kui isik ei ole RR andmetel

teovõimeline, täisealine, elus või puudub rahvastikuregistris, siis e-rahvastikuregistri kasutamine ei ole isikule lubatud. Antud juhul kuvatakse kasutajale vastav veateade.

Järgmise RR päringuga kontrollitakse, kas kohustuslikud isikuandmed vajavad uuendamist. Kohustuslikud isikuandmed on e-post ja ütluspõhised andmed (rahvus, emakeel ja haridustase). Kui vastavad andmed puuduvad, siis suunatakse isik neid täitma. Uuendatud kontaktandmed saadetakse rahvastikuregistrisse.

Kui isikuandmed on korras, siis võib kasutaja alustada elukohateate esitamisega. Avalduse täitmine algab aadressi sisestamisest. Sisestatud aadressi kontrollitakse Maa-ametist vastava teenusega. Vastusest võetakse aadress komponentide kaupa välja ning lehe andmete salvestamisel saadetakse rahvastikuregistri menetlustarkvara poole. Kui sisestatud aadressi ei leita, siis kuvab portaal kasutajale veateate. Lisas 1 on toodud ekraanipilt elukohateate aadressi sisestamise sammust.

Teine samm on seotud isiku(te) valimisega, kelle elukohta soovitakse muuta. Isikud kuvatakse vastavalt rahvastikuregistrist saadetud päringu vastusele. Peale valiku tegemist ja andmete salvestamist saadetakse valitud isikute kohta info rahvastikuregistri menetlustarkvarasse. Lisas 2 on toodud ekraanipilt elukohateate isikute valimise sammust.

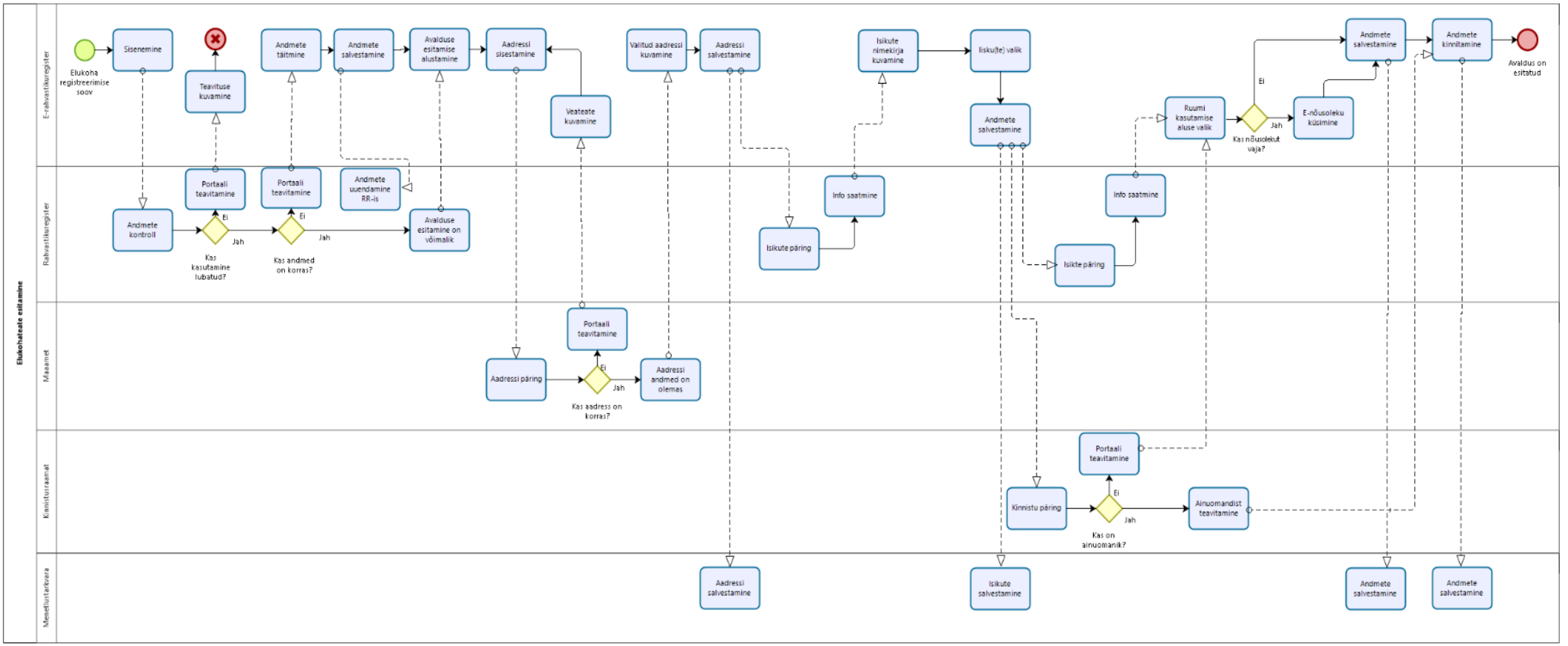
Järgmise sammuna teostatakse päring kinnistusraamatusse, millega kontrollitakse, kas avalduse esitaja on sisestatud uuel aadressil elukoha omanik või mitte. Kui ta ei ole omanik või ei ole ainuke omanik, siis peab kasutaja valima ruumi kasutamise õiguse. Lisas 3 on toodud ekraanipilt ruumi kasutamise õiguse valikutest.

Kui avaldaja on üürniku rollis, siis avalduse esitamiseks on vajalik omaniku nõusolek. Nõusoleku küsimiseks on kaks võimalust: e-rahvatikuregistri kaudu või lisada nõusolek failina.

Juhul kui avaldusele on kaasatud isikud, kes ei kuulu seotud isikute hulka, või alaealised lapsed, kelle kohta kehtib ühine hooldusõigus, siis on nõusoleku küsimine samuti vajalik.

Kui kõik kohustuslikud tegevused nõusolekute osas on tehtud, siis saadetakse andmed rahvastikuregistri menetlustarkvarasse ja isik suunatakse kinnitamise sammu juurde. Lisas 4 on toodud ekraanipilt elukohateate avalduse kokkuvõtvast sammust.

Kinnitamise samm on mõeldud eelnevalt sisestatud/valitud andmete kontrollimiseks. Kui andmed on korras, siis kasutaja võib avalduse esitamise kinnitada. Kinnitatud andmed saadetakse rahvastikuregistri menetlustarkvarasse.



Joonis 5. Elukohateate esitamise põhivoog (Allikas: autori koostatud)

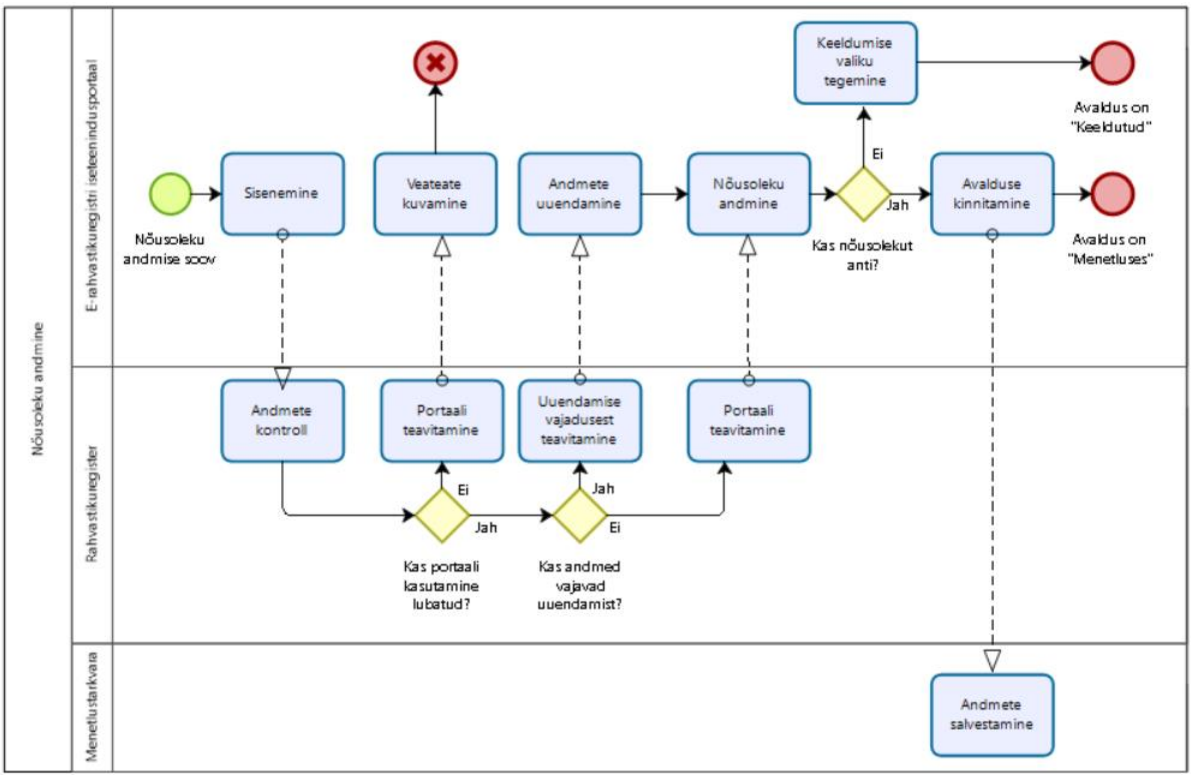
Joonis 5 illustreerib tegevusi ja kontrolle, mis teostatakse e-rahvastikuregistris elukohateate esitamise põhivoo käigus.

5.5.3 E-nõusoleku andmise põhivoog

E-nõusoleku andmiseks peab kasutaja end eelnevalt autentima. Sisenemisel teostatakse samasugused rahvastikuregistri päringud isiku ja isikuandmete kohta nagu elukohateate esitamisel. Töölaua lehele jõudmisel näeb isik kõiki temaga seotud avaldusi. Lisas 5 on toodud ekraanipilt e-rahvastikuregistri kasutaja töölauast.

Nõusoleku andja võib valida, kas ta kinnitab avalduse või keeldub nõusoleku andmisest. Kui kasutaja keeldub nõusoleku andmisest, siis avaldust ei edastata menetlustarkvarasse. Avalduse kinnitamisel salvestatakse nõusoleku andmise kuupäev ja edastatakse rahvastikuregistri menetlustarkvarasse. Lisas 6 on toodud ekraanipilt e-nõusoleku andmise vormist.

Joonis 6 illustreerib e-nõusoleku andmise protsessi:

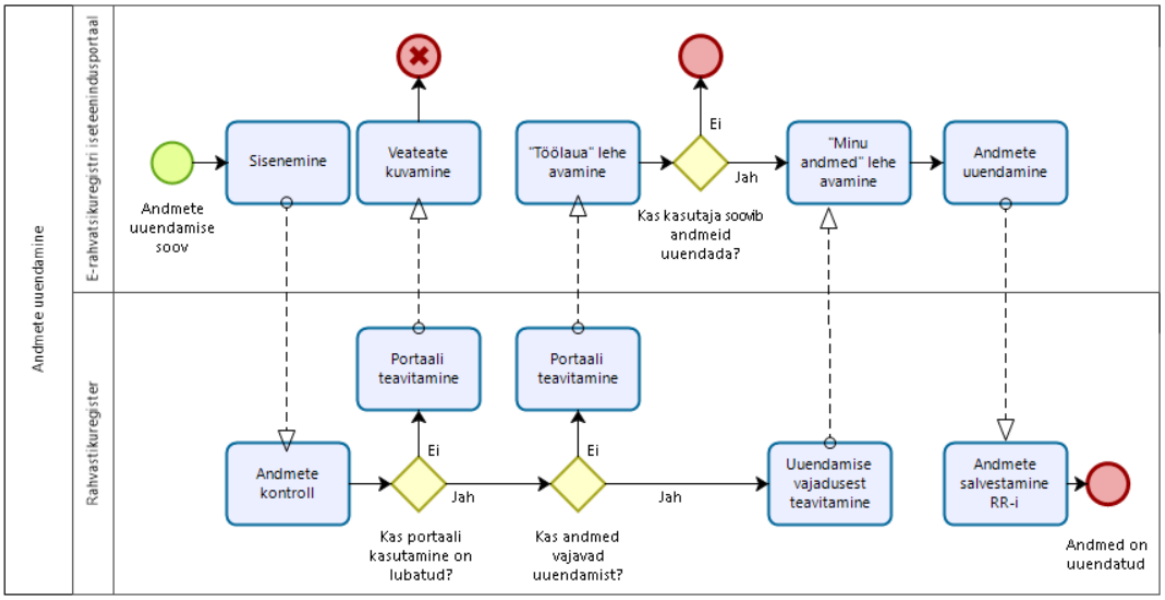


Joonis 6. E-nõusoleku andmise põhivoog (Allikas: autori koostatud)

5.5.4 Andmete uuendamise protsess

E-rahvastikuregistri iseteeninduspportaali sisenemisel kontrollitakse, kas isikuandmed vajavad uuendamist. Kontroll teostatakse rahvastikuregistri vastava päringuga. Kohustuslikud isikuandmed on e-post ja ütluspõhised andmed (rahvus, emakeel ja haridustase). Kui vastavate andmete lahtrid ei ole täidetud või kasutaja on andmed kinnitanud rohkem kui 60 päeva tagasi, siis suunatakse isik neid täitma. Kasutaja poolt lisatud kontaktandmed saadetakse rahvastikuregistrisse.

Joonis 7 illustreerib andmete uuendamise protsessi:

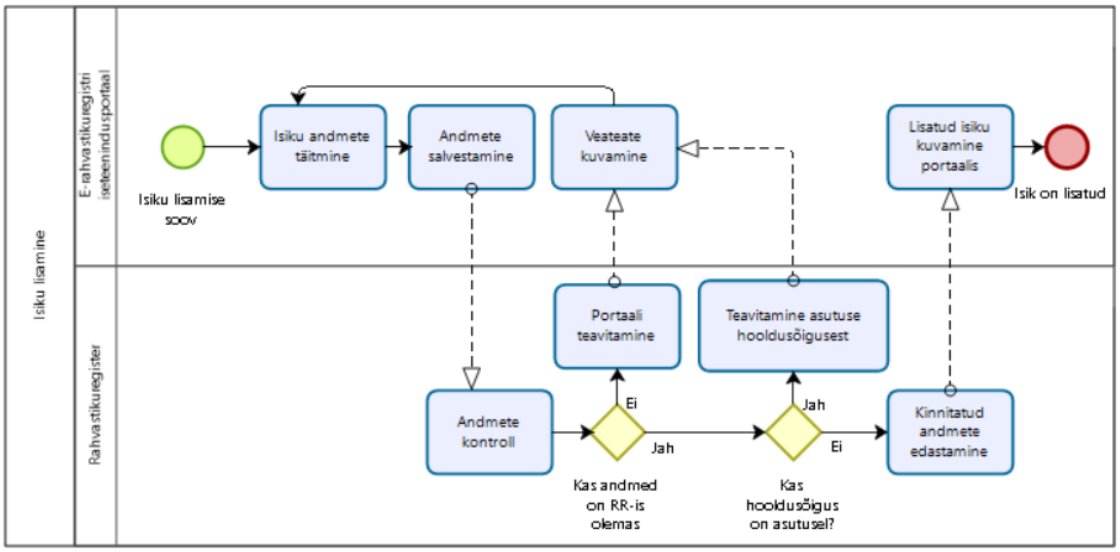


Joonis 7. Andmete uuendamise protsess (Allikas: autori koostatud)

5.5.5 Kolmanda isiku lisamise protsess

Avaldajal on võimalus lisada kolmandaid isikuid elukohta muutvate isikute nimekirja. Isiku käsitsi lisamisel tuleb sisestada lisatava andmed (eesnimi, perekonnanimi ja isikukood). Lisatud isiku salvestamisel toimub andmete valideerimine vastava rahvastikuregistri päringuga. Kui lisatava isiku hooldusõigus on asutusel või päring tagastab tühja vastuse, siis isiku elukoha muutmine ei ole võimalik. Kui päring veateadet ei anna, siis lisatakse isik elukohta muutvate isikute nimekirja.

Joonis 8 illustreerib kolmanda isiku lisamise protsessi:



Joonis 8. Kolmanda isiku lisamise protsess (Allikas: autori koostatud)

5.6 Esialgne automaatiseeritud testide komplekt

Tuginedes eelnevalt uuritud äriprotsessidele ning prioritiseeritud nõuetele elukohatoimingute testide automatiseerimise osas, koostas autor esialgse automatiseeritud testide komplekti, kuhu kuuluvad järgmised testilood:

- TC1: elukohateate avalduse esitamine üürniku rollis;
- TC2: e-nõusoleku andmine ruumi omaniku rollis;
- TC3: e-posti aadressi uuendamine;
- TC4: hooldusõiguste kontroll alaealise isiku lisamisel.

5.7 Testilugude mallid

Tõhusam ja jõulisem lähenemisviis ei ole mitte ainult modelleerida, kuidas kasutaja rakendusega suhtleb, vaid ka kirjeldada seda, mida kasutaja peab samm-sammult tegema soovitud tulemuse saavutamiseks.

Testilugude järgi testide loomine on kõige olulisem samm automaattestimise protsessis, sest ainult korrektselt loodud testid tagavad testitava tarkvaraarenduse kvaliteedi [28].

Järgmisena koostab autor testilugude mallid automatiseerimiseks valitud testidele.

5.7.1 TC1: elukohateate avalduse esitamine üürniku rollis

Tabelis 7 on kirjeldatud elukohateate avalduse esitamise testilugu, kus avaldaja on üürniku rollis. Avaldust esitades küsitakse omaniku nõusolekut uue elukoha registreerimiseks.

Tabel 7. Testilugu – TC1: elukohateate avalduse esitamine üürniku rollis (Allikas: autori koostatud)

ID ja nimetus	TC1: elukohateate avalduse esitamine üürniku rollis
Peamine aktor	Avaldaja üürniku rollis
Kirjeldus	Avaldaja üürniku rollis esitab e-rahvastikuregistri kaudu elukohateate, mille raames küsib omaniku nõusolekut uue elukoha registreerimiseks.
Eeltingimused	Avaldaja on teovõimeline, täisealine, elus ja rahvastikuregistris olemas. Avaldaja on oma andmed kinnitanud vähem kui 60 päeva tagasi.
Vastuvõtu kriteerium	Süsteem kuvab kinnituse elukohateate esitamise kohta.
Sammud	1. Avaldaja saabub e-rahvastikuregistri avalehele. 2. Avaldaja vajutab „Sisenen“ nuppu ja logib sisse. 3. Süsteem suunab avaldaja iseteenindusportaali töölauale. 4. Avaldaja valib vasakult menüüst „Elukoht“. 5. Süsteem kuvab elukoha registreerimise avalehe. 6. Avaldaja vajutab nuppu „Teatan uuest elukohast“. 7. Süsteem kuvab aadressi sisestamise vormi. 8. Avaldaja sisestab Eesti aadressi ja vajutab nuppu „Salvestan ja edasi“.

	9. Süsteem kuvab elukohta muutvate isikute nimekirja. Avaldaja on vaikimisi linnutatud. 10. Avaldaja vajutab nuppu „Salvestan ja edasi“. 11. Süsteem kuvab ruumi kasutamise õiguse valiku. 12. Avaldaja teeb valiku „Mul on omaniku nõusolek või olen üürnik“. 13. Süsteem avab modaalse akna nõusoleku küsimiseks. 14. Avaldaja teeb valiku „E-nõusoleku küsimine“. 15. Süsteem kuvab nõusoleku andja andmete sisestamiseks e-nõusoleku vormi. 16. Avaldaja sisestab ruumi omaniku eesnime, perekonnanime ja isikukoodi ning salvestab sisestatud andmed. 17. Süsteem kuvab lisatud omaniku andmed. 18. Avaldaja vajutab nuppu „Salvestan ja edasi“. 19. Süsteem kuvab avaldajale elukohateate kokkuvõtte. 20. Avaldaja sisestab e-posti ja vajutab nuppu „Kinnitan andmed“. 21. Süsteem kuvab modaalse akna avalduse kinnitamiseks. 22. Avaldaja vajutab „Kinnitan“. 23. Süsteeb kuvab kinnituse elukohateate esitamise kohta.

5.7.2 TC2: e-nõusoleku andmine ruumi omaniku rollis

Tabelis 8 on kirjeldatud e-nõusoleku andmise testilugu, kus nõusoleku andja ruumi omaniku rollis kinnitab avaldaja poolt esitatud elukohateate avalduse.

Tabel 8. Testilugu – TC2: e-nõusoleku andmine ruumi omaniku rollis (Allikas: autori koostatud)

ID ja nimetus	TC2: e-nõusoleku andmine ruumi omaniku rollis.
Peamine aktor	Nõusoleku andja ruumi omaniku rollis.
Kirjeldus	Nõusoleku andja ruumi omaniku rollis kinnitab avaldaja poolt esitatud elukohateate avalduse.

Eeltingimused	Nõusoleku andja on teovõimeline, täisealine, elus ja rahvastikuregistris olemas. Nõusoleku andja on oma andmed kinnitanud vähem kui 60 päeva tagasi.
Vastuvõtu kriteerium	Avaldus kaob ära „Kinnitamist vajavad avaldused“ plokist.
Sammud	1. Nõusoleku andja saabub e-rahvastikuregistri avalehele. 2. Nõusoleku andja vajutab „Sisenen“ nuppu ja logib sisse. 3. Süsteem suunab nõusoleku andja iseteeninduspportaali töölauale. 4. Süsteem kuvab töölaual „Kinnitamist vajavad avaldused“ ploki. 5. Nõusoleku andja avab „Nõusoleku ootel“ staatuses avalduse. 6. Süsteem avab modaalse akna, kus on kuvatud avalduse esitaja andmed, uue elukoha aadress ja nõusolekut vajavad isikud. 7. Nõusoleku andja kinnitab avalduse, vajutades nuppu „Kinnitan avalduse“. 8. Süsteem kuvab kinnituse elukohateate avalduse kinnitamise kohta. Avaldus kaob ära „Kinnitamist vajavad avaldused“ plokist.

5.7.3 TC3: e-posti aadressi uuendamine

Tabelis 9 on kirjeldatud e-posti uuendamise testilugu, kus e-rahvastikuregistri kasutaja uuendab oma andmeid, lisades uue e-posti aadressi.

Tabel 9. Testilugu – TC3: e-posti aadressi uuendamine (Allikas: autori koostatud)

ID ja nimetus	TC3: e-posti aadressi uuendamine
Peamine aktor	E-rahvastikuregistri kasutaja.

Kirjeldus	E-rahvastikuregistri kasutaja uuendab oma andmeid, lisades uue e-posti aadressi.
Eeltingimused	Kasutaja on teovõimeline, täisealine, elus ja rahvastikuregistris olemas. Kasutaja on oma andmed kinnitanud vähem kui 60 päeva tagasi.
Vastuvõtu kriteerium	Süsteem kuvab kinnituse andmete uuendamise kohta.
Sammud	1. Kasutaja saabub e-rahvastikuregistri avalehele. 2. Kasutaja vajutab „Sisenen“ nuppu ja logib sisse. 3. Süsteem suunab kasutaja iseteenindusportaali töölauale. 4. Kasutaja valib vasakult menüüst „Minu andmed“. 5. Süsteem kuvab kasutaja andmete lehe. 6. Kasutaja vajutab nuppu „Lisan e-posti“. 7. Süsteem avab e-posti lisamise vormi. 8. Kasutaja sisestab uue e-posti aadressi ja vajutab nuppu „Salvestan“. 9. Süsteem kuvab lisatud e-posti aadressi kontaktandmete plokis. 10. Kasutaja vajutab nuppu „Kinnitan andmed“. 11. Süsteem kuvab kinnituse andmete uuendamise kohta.

5.7.4 TC4: hooldusõiguse kontroll alaealise isiku lisamisel

Tabelis 10 on kirjeldatud kolmanda isiku lisamise testilugu, kus avaldaja soovib lisada elukohta muutvate isikute nimekirja alaealise isiku, kelle eestkostjaks on asutus.

Tabel 10. Testilugu – TC4: hooldusõiguse kontroll alaealise isiku lisamisel (Allikas: autori koostatud)

ID ja nimetus	TC4: hooldusõiguse kontroll alaealise isiku lisamisel.
Peamine aktor	Avaldaja.

Kirjeldus	Avaldaja lisab elukohta muutvate isikute nimekirja alaelise isiku, kelle eestkostjaks on asutus.
Eeltingimused	Avaldaja on teovõimeline, täisealine, elus ja rahvastikuregistris olemas. Avaldaja on oma andmed kinnitanud vähem kui 60 päeva tagasi.
Vastuvõtu kriteerium	Kuvatakse veateade „Isiku elukoha muutmine ei ole võimalik“.
Sammud	1. Avaldaja saabub e-rahvastikuregistri avalehele. 2. Avaldaja vajutab „Sisenen“ nuppu ja logib sisse. 3. Süsteem suunab kasutaja iseteenindusportaali töölauale. 4. Kasutaja valib vasakult menüüst „Elukoht“. 5. Süsteem kuvab elukoha registreerimise avalehe. 6. Avaldaja vajutab nuppu „Teatan uuest elukohast“. 7. Süsteem kuvab aadressi sisestamise vormi. 8. Avaldaja sisestab Eesti aadressi ja vajutab nuppu „Salvestan ja edasi“. 9. Süsteem kuvab elukohta muutvate isikute nimekirja. Avaldaja on vaikimisi linnutatud. 10. Avaldaja vajutab nuppu „Lisan isiku“. 11. Süsteem kuvab vormi lisatava isiku andmete sisestamiseks. 12. Avaldaja sisestab alaealise isiku, kelle eestkostjaks on asutus, andmed (eesnimi, perekonnanimi ja isikukood) ning vajutab nuppu „Salvestan“. 13. Süsteem kuvab veateate „Isiku elukoha muutmine ei ole võimalik“.

6 Automatiseeritud testide lahenduse kirjeldus

Käesolev peatükk sisaldab automatiseeritud testide lahenduse kirjeldust. Kirjeldatakse loodud automaattestide raamistiku üldist struktuuri ning visualiseeritakse automatiseeritud testide lahendus. Automatiseeritud testide leheobjektide struktuuri väljendamiseks kasutab autor UML klassidiagrammi. Loodud lahendusele lisatakse ettepanekud raamistiku haldamise osas. Lisaks teeb autor täiendusi rahvastikuteenuste osakonna testimise protsessis seoses automaattestide kasutussevõtmisega.

6.1 Testimise automatiseerimise raamistiku struktuur

Testimise automatiseerimise raamistiku loomise protsessi jagas autor kahte etappi: baasprojekti ja moodulite loomine ja seejärel koodi kirjutamine.

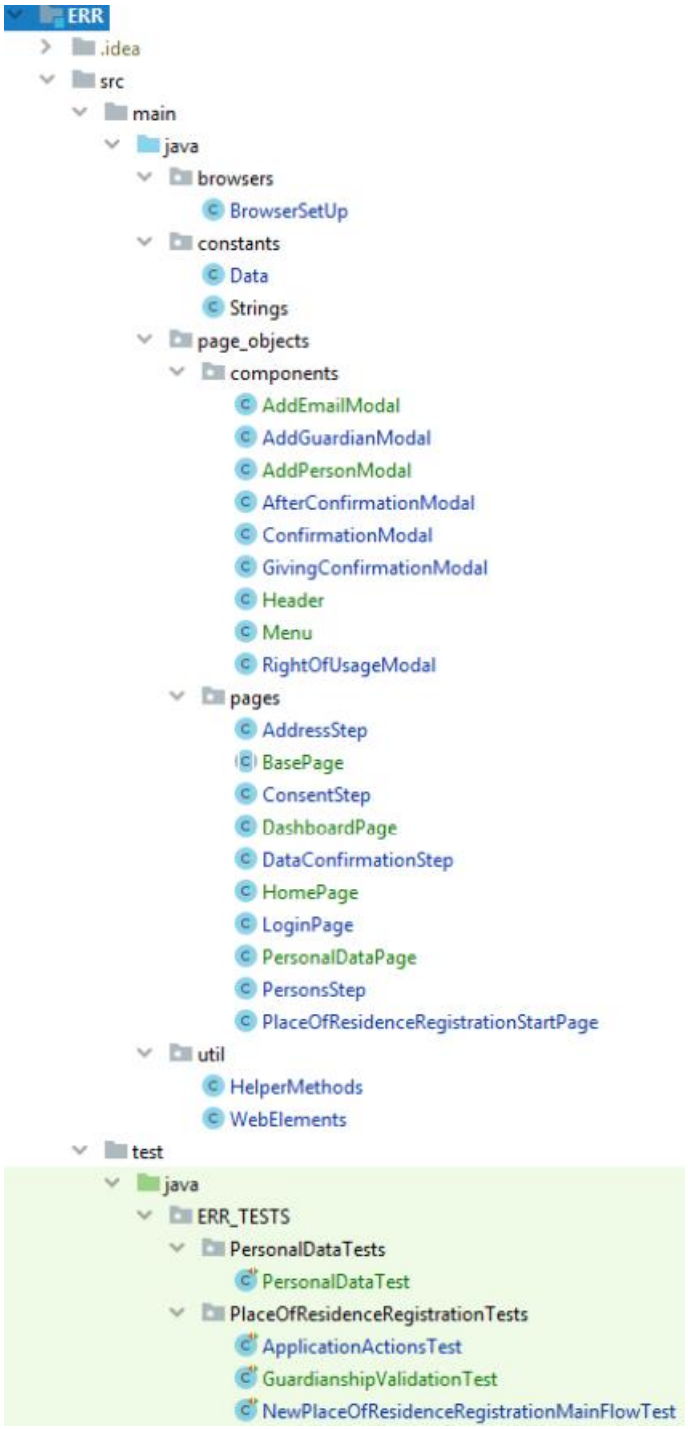
Automaattestide baasprojekti nimeks anti ERR (e-rahvastikuregister) ning projekti tüübiks valiti Maven.

Maveni projekt määratleti XML-failiga nimega pom.xml, mis sisaldab teavet projekti ja konfiguratsioonide kohta, mida Maven kasutab projekti ehitamiseks (näiteks sõltuvused, ehitamise kataloog, eesmärgid). Ülesande täitmisel loeb Maven pom.xml faili, saab vajaliku konfiguratsiooniteabe ja täidab eesmärgi [32]. ERRi projekti pom.xml on toodud lisas 7.

Testimise automatiseerimise raamistiku struktuuri korraldamiseks loodi järgmised moodulid:

- *Browsers* – moodul sisaldab BrowserSetUp klassi, mis on mõeldud draiveri tüübi ja omaduste määramiseks;
- *Constants* – moodul sisaldab klasse Data ja Strings, kuhu on koondatud konstantide kollektsioonid;
- *Page_objects* – moodul koosneb leheobjektide klassidest;
- *Util* – moodul koosneb HelpersMethodsi ja WebElementsi klassidest;
- *Tests* – moodul koosneb testide klassidest.

Joonis 9 illustreerib loodud testimise automatiseerimise raamistiku struktuuri:



Joonis 9. Testimise automatiseerimise raamistiku struktuur (Allikas: autori koostatud)

Järgmistes alapeatükkides lisab autor loodud moodulite lühikirjelduse.

6.1.1 Browsers

Vastavalt püstitatud nõuetele peab automatiseeritud teste saama käivitada, kasutades Google Chrome'i ja Mozilla Firefoxi veebilehitsejaid. Automaattestide käivitamiseks nõutud veebilehitsejates on saadaval järgmised draiverid: ChromeDriver ja GeckoDriver.

Nõude realiseerimiseks loodi Browsersi moodulis BrowserSetUp klass, mille kaudu määrati kasutatava draiveri tüüp ja tee, kuhu see draiver salvestati.

Allpool olev joonis illustreerib loodud BrowserSetUp klassi:

```
public class BrowserSetUp {

    private static WebDriver driver;

    public static WebDriver getDriver() {
        initDriver();
        return driver;
    }

    private static String browser;

    public static void assignChromeBrowser() { browser = "Chrome"; }

    public static void assignFireFoxBrowser() { browser = "FireFox"; }


    public static void initDriver() {
        if (driver == null) {

            /*Chrome */
            if (browser != "FireFox" )

            {
                if (getProperty("webdriver.chrome.driver") == null)
                    setProperty("webdriver.chrome.driver", "src/test/resources/browserBinaries/chromedriver.exe");

                /*Firefox*/

            } else {
                if (getProperty("webdriver.gecko.driver") == null)
                    setProperty("webdriver.gecko.driver", "src/test/resources/browserBinaries/geckodriver.exe");


            }

            /*Browser quit*/

            Runtime.getRuntime().addShutdownHook(run() → {
                    if (driver != null) {
                        driver.quit();
                    }
```

Joonis 10. BrowserSetUp klassi ekraanipilt (Allikas: autori koostatud)

6.1.2 Constants

Testiandmed võivad olla kahte tüüpi: fikseeritud või muutuvad. Constantsi moodul sisaldab Strings ja Data klasse, kuhu on koondatud fikseeritud testiandmed.

Stringsi klass sisaldab e-rahvastikuregistri arenduskeskkonna avalehe aadressi, hiljem lisatakse sinna muud fikseeritud testiandmed, näiteks kasutajanimed, kodifikaatorid jne.

Joonis 11 illustreerib Stringsi klassi:

```
package constants;

public class Strings {

    //URL

    public static String testPage = "https://url.for.rr.test.ee/";

}
```

Joonis 11. Stringsi klassi ekraanipilt (Allikas: autori koostatud)

Data klassi kasutab autor konstantide kogumi määratlemiseks. E-rahvastikuregistri projektis kasutas autor automaattestide koostamisel eelmääratud aadresside väärtuste loendit.

Joonis 12 illustreerib Data klassi:

```
public enum PersonalAddress {

    LASNAMAE("Virbi tn, Lasnamäe linnaosa, Tallinn, Harju maakond"),
    MUSTAMAE("Mustamäe tee, Mustamäe linnaosa, Tallinn, Harju maakond"),
    HAABERSTI("Vanakuu tn, Haabersti linnaosa, Tallinn, Harju maakond"),
    KRISTIINE("Käo tn , Kristiine linnaosa, Tallinn, Harju maakond");



    private final String PersonalAddress;

    private PersonalAddress (final String PersonalAddress) { this.PersonalAddress = PersonalAddress; }

    public static PersonalAddress  getRandom() {
        return values()[(int) (Math.random() * values().length)];

    }

    @Override
    public String toString() { return PersonalAddress; }
}
```

Joonis 12. Data klassi ekraanipilt (Allikas: autori koostatud)

6.1.3 Page_objects

Page_objects moodul kooseb Pages alammooduli klassidest ja Components alammooduli klassidest.

Pages alammoodul sisaldab BasePage klassi, kus hoitakse e-rahvastikuregistri automaattestide projekti leheobjektide ühiseid meetodeid. Kõik leheobjektide klassid laiendavad BasePage'i klassi.

Joonis 13 illustreerib e-rahvastikuregistri automaattestide projekti BasePage'i klassi:

```
package page_objects.pages;

import browsers.BrowserSetUp;
import page_objects.components.Header;
import page_objects.components.Menu;

public abstract class BasePage extends BrowserSetUp {

    private static final Menu sideBar = new Menu();//
    public Menu getSideBar() { return sideBar; }

    private static final Header header = new Header();//
    public Header getHeader() { return header; }


}
```

Joonis 13. BasePage'i klassi ekraanipilt (Allikas: autori koostatud)

Leheobjektide klassid on jagatud kahte gruppi: Pages ja Components. Pagesi klassidele vastavad e-rahvastikuregistri veebirakenduse veebilehtedel näiteks avaleht, töölaud jne. Componentsi klassidega tähistatakse lehe konkreetset osa, mis aitab kasutajal konkreetset toimingut täita, näiteks menüü, modaalaken jne

Iga leheobjektide klass koosneb veebielementide muutujatest ja veebielementide muutujatega seotud meetoditest. Meetoditeks on tavaliselt kasutaja interaktsioonid.

Enne veebielemendi muutujate kasutamist veebielemente sisaldavad leheobjektid lähtestatakse *initElements()* meetodi abil. Page Factory initsialiseerib kõik veebielementide muutujad viitega vastavale elemendile tegelikul veebilehel konfigureeritud lokaatorite põhjal @FindBy märkuse abil.

Joonised 14 ja 15 illustreerivad Pagesi ja Componentsi alammoodulite klasside näited:

```java
package page_objects.pages;


import util.WebElements;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;


public class LoginPage extends BasePage {

    @FindBy(id="dev-login")
    private WebElement loginInput;

    @FindBy(xpath = "//button[contains(text(), 'Login')]")
    private WebElement loginBtn;


    public LoginPage insertPersonalCode(String personalCode) {
        WebElements.waitForVisibility(loginInput);
        loginInput.clear();
        loginInput.sendKeys(personalCode);
        return this;
    }

    public DashboardPage clickLogin() {
        WebElements.waitForVisibility(loginBtn);
        WebElement element = loginBtn;
        JavascriptExecutor executor = (JavascriptExecutor) getDriver();
        executor.executeScript( s: "arguments[0].click();", element);
        return new DashboardPage();
    }



    public LoginPage() { PageFactory.initElements(getDriver(),  page: LoginPage.this); }
}
```

Joonis 14. Pages: LoginPage klassi ekraanipilt (Allikas: autori koostatud)

```java
package page_objects.components;

import page_objects.pages.BasePage;
import util.HelperMethods;
import page_objects.pages.DashboardPage;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;


public class GivingConfirmationModal extends BasePage {

    @FindBy(xpath = "//button[contains(text(), 'Kinnitan avalduse')]")
    private WebElement confirmBtn;

    public DashboardPage clickConfirmBtn() {
        HelperMethods.checkLoader();
        WebElement element = confirmBtn;
        JavascriptExecutor executor = (JavascriptExecutor) getDriver();
        executor.executeScript( s: "arguments[0].click();", element);
        return new DashboardPage();
    }

    public GivingConfirmationModal() { PageFactory.initElements(getDriver(),  page: GivingConfirmationModal.this); }
}
```

Joonis 15. Components: GivingConfirmationModal klassi ekraanipilt (Allikas: autori koostatud)

6.1.4 Util

Peaaegu igal Seleniumi abil automatiseeritud kasutajaliidese rakendusel on 80% samad meetodid, mis on loodud automaattestide lihtsustamiseks ja mida saab korduvalt kasutada [36]. Selliste meetodite ühes kohas hoidmiseks said loodud HelperMethodsi ja WebElementsi klassid Util moodulis.

E-rahvastikuregistri automaattestide projektis sisaldab HelpersMethodsi klass meetodeid, mille abil iga uue lehe laadimisel kontrollitakse, kas laadimisikoon kuvatakse, ning oodatakse, kuni leht täielikult laeb.

Joonis 16 illustreerib HelpersMethodsi klassi meetodid:

```
public static void waitForLoad() {
    WebDriverWait wait = new WebDriverWait(getDriver(), timeOutInSeconds: 15);
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//img[@src='/content/images/295.gif']")));
    wait.until(ExpectedConditions.invisibilityOfElementLocated(By.xpath("//img[@src='/content/images/295.gif']")));
}

public static void checkLoader() {
    try {
        WebElement loader = getDriver().findElement(By.xpath("//img[@src='/content/images/295.gif']"));
        if (loader.isDisplayed() && loader.isEnabled()) {
            HelperMethods.waitForLoad();
        }
    } catch (Exception e) {
        System.out.println("");

    }
}
```

Joonis 16. HelpersMethodsi klassi meetodid (Allikas: autori koostatud)

WebElementsi klass sisaldab meetodeid testilugude tulemuste valideerimiseks. Need meetodid aitavad mõista, kas testilugu on täidetud või mitte. Testilugu loetakse täidetuks, kui rakenduse tegelik tulemus langeb kokku eeldatava tulemusega.

Joonis 17 illustreerib näidet ühest valideerimise meetodist:

```
public static boolean doesExist(WebElement webElement) {
    try {
        return webElement.isDisplayed();
    } catch (NoSuchElementException e) {
        return false;
    }
}
```

Joonis 17. WebElementsi klassi valideerimise meetod (Allikas: autori koostatud)

6.1.5 Tests

Tests moodul sisaldab automatiseeritud testide klasse. Autor on jaganud automatiseeritud testid kahte gruppi. Automatiseeritud testide esimene grupp on seotud kasutaja andmete uuendamise protsessiga (*PersonalDataTests*) ja teine grupp on seotud elukoha registreerimise protsessiga (*PlaceOfResidenceRegistrationTests*). Igale testile on mõeldud eraldi klass, mis koosneb ühest põhivoo testist ja mida on hiljem võimalik täiendada alternatiivvoo testidega.

6.2 Automatiseeritud testid

Alljärgnevalt visualiseerib autor automatiseeritud testide lahenduse. Automatiseeritud testide leheobjektide struktuuri väljendamiseks kasutab autor UML klassidiagrammi.

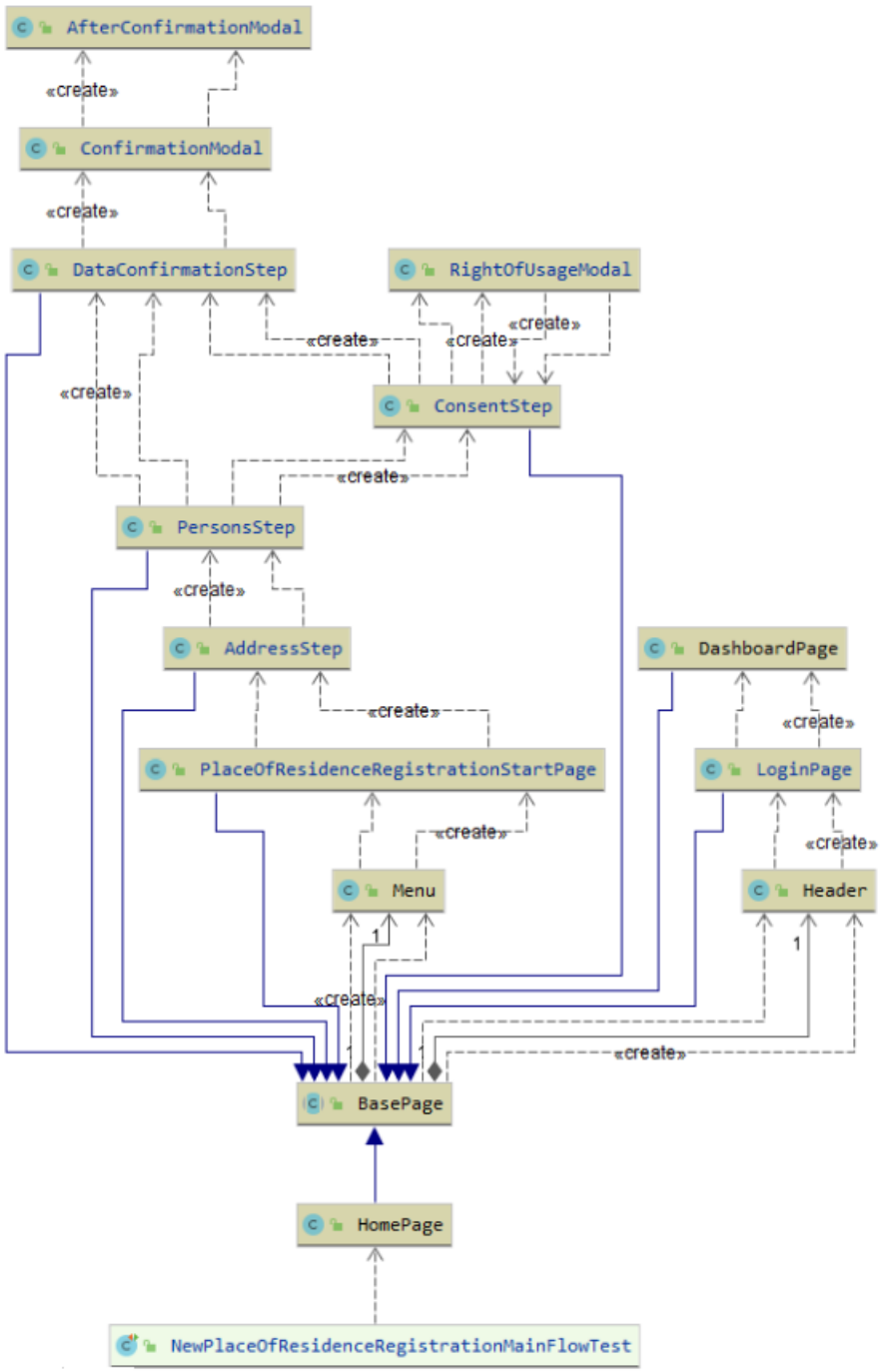
6.2.1 TC1: Elukohateate avalduse esitamine üürniku rollis

Joonis 18 illustreerib TC1 automatiseeritud testi. Test koosneb Pagesi ja Componentsi leheobjektide meetoditest. Testis kontrollitakse kinnituse kuvamist elukohateate esitamise kohta.

```
@Test
/*Isik üüriniku rollis küsib e-nõusolekut omanikult (RB2)*/
public void registrationOfANewResidentialAddressRB2Test() {
    goToLandingPage() HomePage
            .getHeader() Header
            .checkLogin() LoginPage
            .insertPersonalCode(Data.PersonalCode.TEST_PERSON_4.toString())
            .clickLogin() DashboardPage
            .getSideBar() Menu
            .clickPlaceOfResidenceIcon() PlaceOfResidenceRegistrationStartPage
            .clickNewHomeReportBtn() AddressStep
            .insertPersonalAddress(Data.PersonalAddress.LASNAMAE.toString())
            .insertApartmentNr("XX")
            .clickProceedPersonsStep() PersonsStep
            .clickProceedConsentStep() ConsentStep
            .selectRb2() RightOfUsageModal
            .selectTheWayOfAskingConsent(1)
            .setFirstName("Test1")
            .setLastName("Test2")
            .insertPersonalCode(Data.PersonalCode.TEST_PERSON_2.toString())
            .clickSavetBtn() ConsentStep
            .clickProceedConfirmationStep() DataConfirmationStep
            .clickAddEmail() AddEmailModal
            .setEmail()
            .saveEmail() DataConfirmationStep
            .clickConfirmBtn() ConfirmationModal
            .clickConfirmApplication() AfterConfirmationModal
            .confirmationTextIsDisplayed();

}
```

Joonis 18. Automatiseeritud test: TC1: elukohateate esitamine üürniku rollis (Allikas: autori koostatud)

Joonisel 19 on TC1 automatiseeritud testi leheobjektide klassid väljendatud klassidiagrammina. Joonisel ei ole kujutatud objektide atribuute.



Joonis 19. UML klassidiagramm: TC1: elukohateate esitamine üürniku rollis (Allikas: autori koostatud)

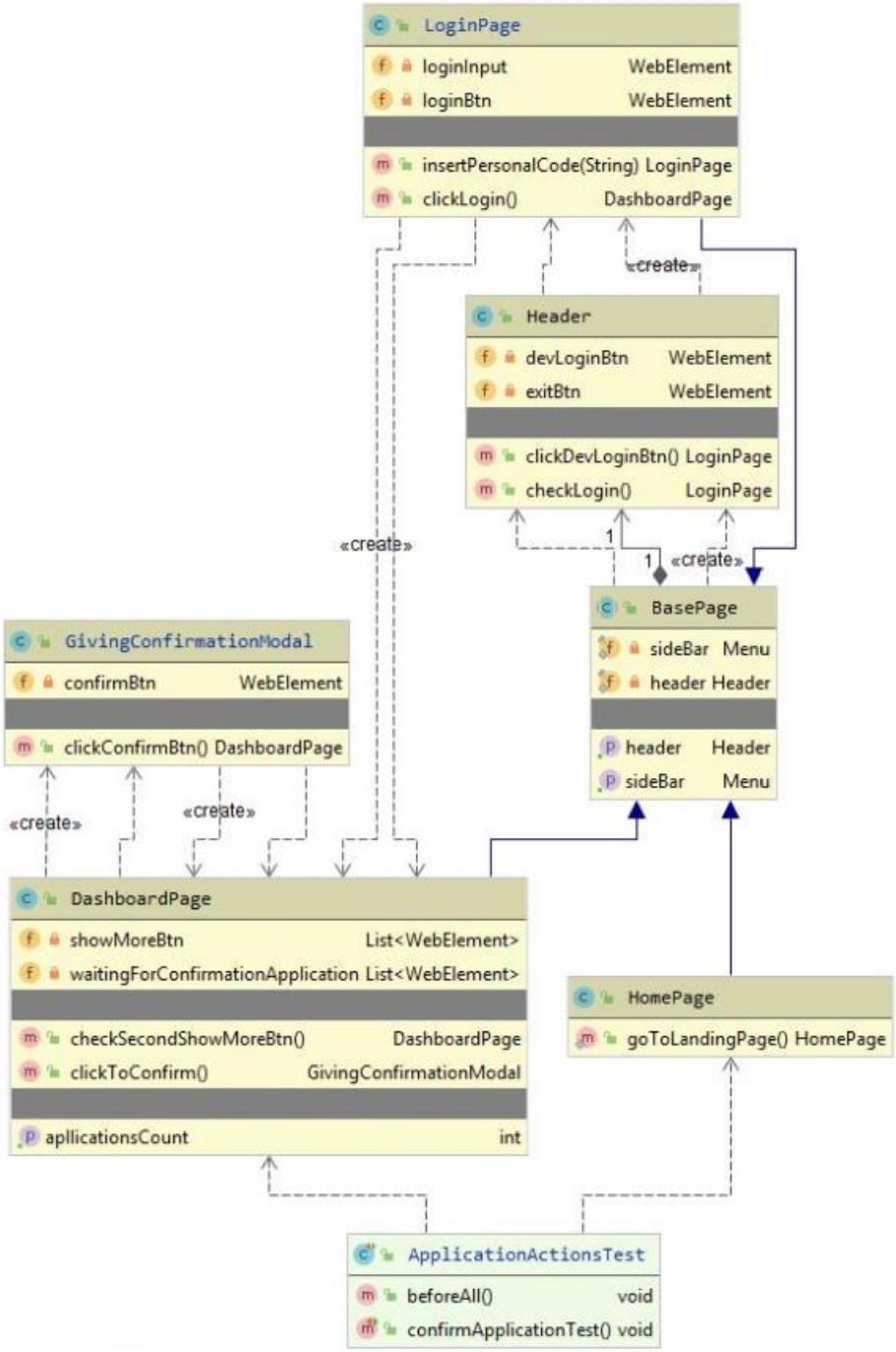
6.2.2 TC2: e-nõusoleku andmine ruumi omaniku rollis

Joonis 20 illustreerib TC2 automatiseeritud testi. Test koosneb Pagesi ja Componentsi leheobjektide meetoditest. Testis võrreldatakse kinnitamist vajavate avalduste arvu enne e-nõusoleku andmist ja peale e-nõusoleku andmist.

```
public class ApplicationActionsTest {

    @BeforeClass
    public void beforeAll() { BrowserSetUp.assignChromeBrowser(); }

    /* omanik annab elukoha registreerimiseks nõusolekut*/

    @Test
    public void confirmApplicationTest() {
        DashboardPage dashboardPage = goToLandingPage() HomePage
                .getHeader() Header
                .checkLogin() LoginPage
                .insertPersonalCode(Data.PersonalCode.TEST_PERSON_2.toString())
                .clickLogin();
        int initialIncompleteApplicationsCount = dashboardPage.checkSecondShowMoreBtn()
                .getApllicationsCount();
        dashboardPage = dashboardPage
                .clickToConfirm()
                .clickConfirmBtn();
        int newIncompleteApplicationsCount = dashboardPage.checkSecondShowMoreBtn()
                .getApllicationsCount();
        assertTrue( condition: initialIncompleteApplicationsCount - 1 == newIncompleteApplicationsCount);

    }


}
```

Joonis 20. Automatiseeritud test: TC2: e-nõusoleku andmine ruumi omaniku rollis (Allikas: autori koostatud)

Joonisel 21 on TC2 automatiseeritud testi leheobjektide klassid väljendatud klassidiagrammina. Lisaks leheobjektide klassidele on joonisel kujundatud ka veebielementide muutujad ja nendega seotud meetodid.



Joonis 21. UML klassidiagramm: TC2: e-nõusoleku andmine ruumi omaniku rollis (Allikas: autori koostatud)

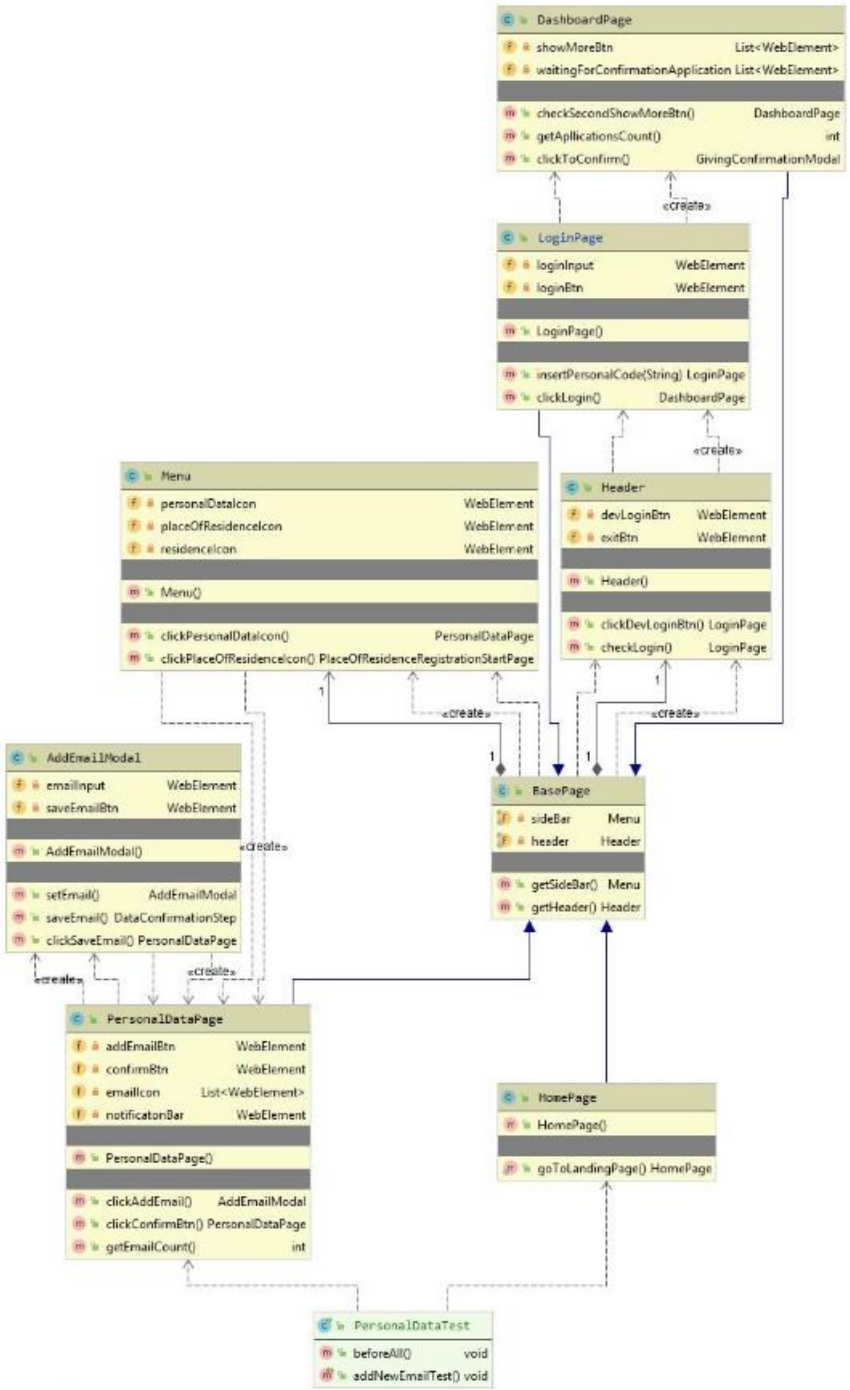
6.2.3 TC3: e-posti aadressi uuendamine

Joonis 22 illustreerib TC3 automatiseeritud testi. Test koosneb Pagesi ja Componentsi leheobjektide meetoditest. Testis võrreldatakse e-posti aadresside arvu enne uue e-posti aadressi lisamist ja peale e-posti aadressi lisamist. Lisaks kontrollitakse kinnituse kuvamist andmete uuendamise kohta.

```
public class PersonalDataTest {

    @BeforeClass
    public void beforeAll() { BrowserSetUp.assignChromeBrowser(); }


    /*uue e-maili lisamine*/

    @Test
    public void addNewEmailTest() {
      PersonalDataPage personalDataPage =  goToLandingPage() HomePage
                .getHeader() Header
                .checkLogin() LoginPage
                .insertPersonalCode(Data.PersonalCode.TEST_PERSON_1.toString())
                .clickLogin() DashboardPage
                .getSideBar() Menu
                .clickPersonalDataIcon();
              int initialEmailCount = personalDataPage.getEmailCount();
              personalDataPage = personalDataPage.clickAddEmail()
                .setEmail()
                .clickSaveEmail();
              int newEmailCount = personalDataPage.getEmailCount();
      assert (initialEmailCount + 1 == newEmailCount);
      personalDataPage = personalDataPage.clickConfirmBtn();

    }
```

Joonis 22. Automatiseeritud test: TC3: e-posti aadressi uuendamine (Allikas: autori koostatud)

Joonisel 23 on TC3 automatiseeritud testi leheobjektide klassid väljendatud klassidiagrammina. Lisaks leheobjeketide klassidele on joonisel kujutatud ka veebielementide muutujad ja nendega seotud meetodid.



Joonis 23. UML klassidiagramm: TC3: e-posti aadressi uuendamine (Allikas: autori koostatud)

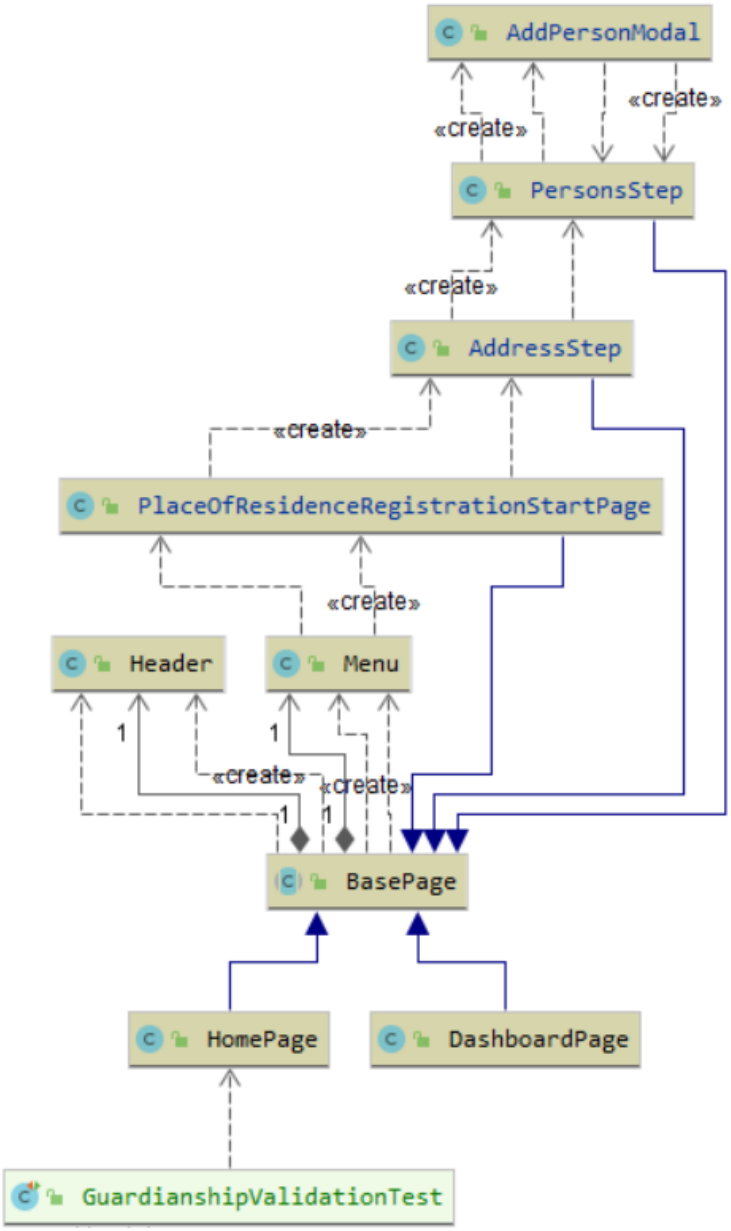
6.2.4 TC4: hooldusõiguse kontroll alaealise isiku lisamisel

Joonis 24 illustreerib TC4 automatiseeritud testi. Test koosneb Pagesi ja Componentsi leheobjektide meetoditest. Testis kontrollitakse hooldusõigust ning veateate kuvamist alaealise isiku lisamisel.

```
public class GuardianshipValidationTest {

    @BeforeClass
    public void beforeAll() { BrowserSetUp.assignFireFoxBrowser(); }


    @Test
    /*Asutus on lapse eestkostja*/

    public void personAuthorityGuardianshipTest() {
        goToLandingPage() HomePage
                .getHeader() Header
                .checkLogin() LoginPage
                .insertPersonalCode(Data.PersonalCode.TEST_PERSON_5.toString())
                .clickLogin() DashboardPage
                .getSideBar() Menu
                .clickPlaceOfResidenceIcon() PlaceOfResidenceRegistrationStartPage
                .clickNewHomeReportBtn() AddressStep
                .insertPersonalAddress(Data.PersonalAddress.LASNAMAE.toString())
                .insertApartmentNr("XX")
                .clickProceedPersonsStep() PersonsStep
                .clickAddPersonBtn() AddPersonModal
                .newPersonData()
                .validateGuardianship();
    }
}
```

Joonis 24. Automatiseeritud test: TC4: hooldusõiguse kontroll alaealise isiku lisamisel (Allikas: autori koostatud)

Joonisel 25 on TC4 automatiseeritud testi leheobjektide klassid väljendatud klassidiagrammina. Joonisel ei ole kujutatud objektide atribuute.



Joonis 25. UML klassidiagramm: TC4: hooldusõiguse kontroll alaealise isiku lisamisel (Allikas: autori koostatud)

6.3 Ettepanekud testimise automatiseerimise raamistiku haldamise osas

Arvatakse, et on väga raske välja tuua täpseid reegleid, mida peab järgima, et luua stabiilsete testidega hästi kavandatud ja hooldatav testimise automatiseerimise raamistik, sest igast reeglist on alati palju erandeid. Kuid hoolimata sellest ja tuginedes

automaattestimise parimatele praktikale [37], toob autor välja üldised põhimõtted, mida võiks testimise automatiseerimise raamistiku haldamisel järgida:

- Testiklasside loomisel tuleks kasutada klassinime, mis peegeldab vastava testi sisu ja millele järgneb sõna “test”, nt *GuardianshipValidationTest*, sest see aitab hoida fookust testi eesmärgil ning grupeerida sama tüüpi testid ühe testiklassi alla.
- Kõik testiklassid tuleks salvestada ainult testidele mõeldud kausta ja see peaks sisaldama ainult teste.
- Konfiguratsiooni seadistused, näiteks veebilehitseja valik, tuleks teha meetodiga *@BeforeClass* või mõne muu sarnase märkuste meetodiga enne testi või testide seeria täitmist.
- Kui veebiraksenduse leht on mahukas ning sisaldab liiga palju komponente, siis lahenduseks on nende komponentide jagamine väiksemateks klassideks, kus iga komponent tähistaks veebirakenduse lehe konkreetset osa, näiteks menüü, sisselogimisvorm jne.
- Navigeerimine lehtede või komponentide vahel peaks tagastama järgmise lehe leheobjekti.
- Tuleks vältida koodi dubleerimist ning veenduda enne uue meetodi loomist, et vastavas testiklassis puudub sarnane meetod, mis juba täidab samu ülesandeid.

6.4 Testimise protsessi täiendamine automaattestidega seotud tegevustega

Seoses automatiseeritud testide juurutamisega e-rahvastikuregistri arenduskeskkonnas täiendas autor rahvastikuteenuste osakonna testimise protsessi järgmiste tegevustega:

1. **Kasutajalugude valik automatiseerimiseks**

Kasutuslugude kvaliteedi kontrolli käigus teostatakse esialgne kasutuslugude valik testide automatiseerimiseks. Valiku tegemisel arvestatakse kasutuslugude olulisusega regressioonitestimise jaoks ning üritatakse eraldada kõrge riskiga kasutajalood, mis on seotud veebirakenduse kriitiliste osadega. Kasutajalugude valik automaattestide loomiseks kinnitatakse tooteomaniku poolt.

2. **Automaattestide arendusmahu hindamine**

Kui kasutajalugude valik testide automatiseerimiseks on teostatud, siis hinnatakse arendustööde hindamise koosolekul automaattestide arendusmahtu. Eraldi hinnatakse manuaalse testimisega seotud tegevusi ja automaattestide loomisega seotud tegevusi.

3. **Automaattestide kasutamine regressioonitestimisel**

Kõik automatiseeritud testid osalevad regressioonitestimisel. Eelkõige on plaanis täiendada automatiseeritud testidega elukohatoimingutega seotud regressioonitestide nimekirja, siis järgmise tegevusena luua automatiseeritud testid e-rahvastikuregistri teistele teenustele.

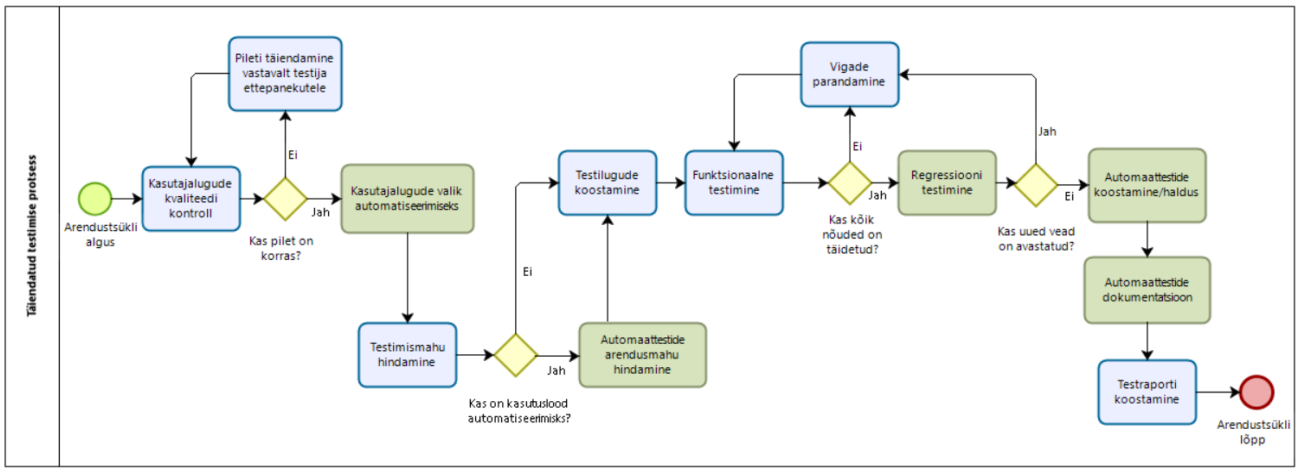
4. **Automaattestide koostamine ja haldamine**

Testide automatiseerimine on jooksev tegevus. Kui kasutuslugu on edukalt läbinud manuaalse testimise protsessi, siis alustatakse automaattestide koostamise protsessiga. Kui arenduse käigus on toimunud muudatused veebirakenduse automaattestidega kaetud funktsionaalsuse osas, siis toimub olemasolevate automaattestide täiendus vastavalt toimunud muudatustele.

5. **Automaattestide dokumentatsiooni loomine**

Kõik automatiseeritud testilugude mallid avaldatakse vastaval e-rahvastikuregistri *Confluence Wiki* lehel. Dokumenteeritud testilugu koosneb pealkirjast, testimise eeltingimustest, testiloo sammude kirjeldusest ning vastuvõtu kriteeriumist.

Allpool olev joonis illustreerib täiendatud (TO-BE) testimise protsessi, automaattestidega lisandunud tegevused on tähistatud rohelise värviga:



Joonis 26. Täiendatud (TO-BE) testimise protsess (Allikas: autori koostatud)

7 Järeldused

Käesolevas peatükis annab autor hinnangu magistritööga saavutatud eesmärkidele. Lisaks toob autor välja järgmised sammud automaattestimise edasiarendamise ja kasutussevõtmise osas mujal SMITis.

Töö käsitles automaattestide juurutamist e-rahvastikuregistri iseteeninduspordaali arenduskeskkonnas elukohatoimingute näitel. Magistritöö eesmärk oli läbi viia ärianalüüs ning luua esialgne automatiseeritud testide komplekt elukohatoimingutele, et oleks võimalik valida parim viis automaattestide juurutamisega alustamiseks ning sellega säästa arendusmeeskonna tööressursse, aega ja eelarvet. Magistritöö eesmärk on täidetud, automaattestide juurutamisega e-rahvastikuregistri arenduskeskkonnas on alustatud.

2020. aasta aprilliks olid kõik elukohatoimingutega seotud regressioonitestid kaetud automatiseeritud testidega ning järgnevalt teostatakse ettevalmistusi sünni registreerimise teenuse regressioonitestide automatiseerimiseks.

Vastavalt SMITi testijate ümarlaual avaldatud soovile pidi autor looma SMITi sisemiseks kasutamiseks testimisraamistiku näidisprojekti, mis oleks lihtsasti õpitav ja hästi dokumenteeritud, et ka teised SMITi testijad võiksid vajadusel oma projektides automaattestide juurutamisega alustada. Eesmärk on täidetud – automatiseeritud testide näidisprojekt, mis koosneb esialgsest elukohatoimingute automatiseeritud testide komplektist, on avaldatud SMITi testijatele mõeldud Bitbucketi repositooriumis ning sellele on lisatud autori koostatud tehniline juhend. Asutuse sees toimunud testijate ümarlaual autor on tutvustanud SMITi testijatele loodud automatiseeritud testide lahendust.

Lisaks püstitas autor eesmärgi täiendada rahvastikuteenuste osakonna olemasoleva testimise protsessi automaattestidega seotud tegevustega. Eesmärk on täidetud – testimise protsessi lisandusid järgmised tegevused: kasutuslugude valik automatiseerimiseks, automaattestide arendusmahu hindamine, automaattestide loomine ja kasutamine regressioonitestidena, automaattestide haldus ja dokumentatsiooni loomine.

Järgmisena toob autor välja automaattestimise edasiarenduse sammud, mis esialgsest skoobist välja jäid:

1. **Automatiseeritud testide lahenduse sidumine pideva integratsiooni *(Continious Integration/CI)* vahendiga**

Pideva integratsiooni ja tarnimise vahendiks e-rahvastikuregistri iseteenindusportaali projektis on Atlassian Bamboo keskkond, kus tarkvara ehitamine ning ühiktestid käivitatakse iga kord automaatselt, kui koodimuudatus Bitbucketi repositooriumisse jõuab. Loodud automatiseeritud testimise raamistik on plaanis siduda Bamboo keskkonnaga automaattestide käivitamiseks ning testraportite loomiseks.

2. **Automatiseeritud testide loomine e-rahvastikuregistri teistele e-teenustele**

Magistritöö kirjutamise hetkel on rahvastikuregistri iseteenindusportaalis võimalik registreerida elukohta, vaadata ja uuendada isikuandmeid. Märtsi lõpus lisandus ka sünni registreerimise võimalus. 2020. aasta jooksul täieneb e-teenuste valik paljude erinevate e-teenustega, näiteks rahvastikuregistrist tõendite saamisega, eluruumi registreeritud isikute ja surma andmete päringutega. Automaattestidega soovitakse katta kõik rahvastikuregistri iseteenindusportaali arendatud e-teenused 2020. aasta lõpuks.

3. **Automatiseerimise mõõdikute kasutamine**

Tulevikus on plaanis testimise automatiseerimise edukuse mõõtmiseks kasutada selliseid mõõdikuid nagu testide loomisele ja parandamisele kuluva ajakulu mõõtmine, automatiseeritavuse määra arvutamine, hooldatavus, usaldusväärtus, robustsus jne. Mõõdikute kasutamine aitab luua selge arusaama automaattestide tugevustest ja nõrkustest ning võimaldab keskenduda probleemsetele kohtadele automaattestide arendamisel.

4. **Nooremtestijate koolitamine**

Rahvastikuteenuste osakonnas on juba alustatud nooremtestija koolitamisega automaattestide koostamise osas. Selle tegevusega jätkatakse. Samuti on plaanis viia läbi testimise automatiseerimisega seotud koolitusi SMITi teistes osakondades, kus võib selline vajadus tekkida näiteks uue nooremtestija tööle võtmisel.

Kokkuvõte

SMITi ja Siseministeeriumi jaoks on oluline pakkuda Eesti elanikele kvaliteetset ja kasutajasõbralikku veebikeskkonda rahvastikuregistri e-teenuste kasutamiseks. Tarkvara kvaliteedi tagamisel on tähtis roll testide automatiseerimisel, sest see aitab oluliselt kiirendada arendusprotsessi regressioonitestimise osas ja samas maksimaalselt vähendada erakorralisi tarkvarauuendusi, mis võivad olla põhjustatud arenduskeskkonnas avastamata vigadest.

SMITi rahvastikuteenuste osakonnas puudus magistritöö kirjutamise hetkel kokkupuude automaattestidega ja seetõttu tekkis vajadus leida parima viis automaattestide juurutamisega alustamiseks.

Magistritöö esimeses peatükis andis autor ülevaate valdkonnast, rahvastikuteenuse osakonna olemasolevast testimise protsessist ja sõnastas magistritöö raames lahendatava probleemi.

Töö teises peatükis sai püstitatud magistritöö eesmärk – läbi viia ärianalüüs automaattestide juurutamise alustamiseks e-rahvastikuregistri arenduskeskkonnas. Autor tõi välja magistritöö skoobi, andis ülevaate enda rollist arendusprojektis ning lisas teiste arendusprojekti kaasatud osapoolte rollide kirjeldused.

Kolmandas peatükis kirjeldas autor e-rahvastikuregistri iseteeninduspordaali projekti arendusmetoodikat ja põhjendas analüüsiks valitud analüüsimetoodikaid.

Neljandas peatükis uuriti ja analüüsiti populaarsemaid arenduskeskkondi, testimisraamistikke ja disainimustreid ning seatud kriteeriumite põhjal tehti valik järgmise tööriistakomplekti kasuks:

- IntelliJ Ultimate Editioni arenduskeskkond;
- TestNG testimisraamistik;
- Page Factory disainimuster.

Töö viies peatükk hõlmas ärianalüüsi, mille raames kaardistati huvitatud osapooled ning viidi läbi poolstruktureeritud intervjuud. Intervjuude tulemuste põhjal koostati ärinõuded automatiseeritud testide lahenduse osas, mis hiljem prioritiseeriti. Lisaks teostati põhiliste

äriprotsesside modelleerimine BPMN notatsioonis, mis oli sisendiks testilugude koostamisele. Teostatud ärianalüüsi põhjal loodi e-rahvastikuregistri elukohatoimingutele automaatiseeritud testide komplekti kasutades kasutajaliidese automaattestimise tarkvara Selenium Webdriver koos TestNG testimisraamistikuga ja PageFactory disainimustriga. Esialgne automatiseeritud testide komplekt sisaldas järgmisi teste:

- TC1: elukohateate avalduse esitamine üürniku rollis;
- TC2: e-nõusoleku andmine ruumi omaniku rollis;
- TC3: e-posti aadressi uuendamine;
- TC4: hooldusõiguste kontroll alaealise isiku lisamisel.

Kuuendas peatükis kirjeldas autor automatiseeritud testide lahendust, mille käigus andis ülevaate loodud automaattestide raamistiku üldisest struktuurist ning visualiseeris automatiseeritud testide lahenduse. Automatiseeritud testide leheobjektide struktuuri väljendamiseks lõi autor UML klassidiagrammi. Loodud lahendusele lisati ettepanekud raamistiku haldamise osas. Seoses automaattestide kasutussevõtmisega täiendas autor rahvastikuteenuste osakonna testimise protsessi.

Seitsmendas peatükis andis autor hinnangu magistritööga saavutatud eesmärkidele ning tõi välja järgmised sammud automaattestide edasiarendamise ja kasutussevõtmise osas mujal SMITis.

Veebruaris 2020 alustati SMITis automatiseeritud testide lahenduse juurutamisega ning 2020 aprilliks olid kõik elukohatoimingutega seotud regressioonitestid automatiseeritud. Magistritöö raames loodud automatiseeritud testide komplektile lisati tehniline juhend ning näiteprojekti esitleti SMITi testijate ümarlaual. Rahvastikuteenuste osakonna testimise protsessi täiendati seoses automatiseeritud testide juurutamisega.

Magistritöös püstitatud probleem on lahendatud ja kõik magistritöös püstitatud eesmärgid on täidetud.

Kasutatud allikad

[1] SMIT avas uue rahvastikuregister.ee portaali. – *Siseministeeriumi infotehnoloogia- ja arenduskeskus*, 2019 [WWW] https://www.smit.ee/et/uudised/smit-avas-uue-rahvastikuregister-ee-portaali-70 (13. jaanuar 2020)

[2] Tarkvara pidev kasutuselevõtt tagab parema kvaliteedi ja rahulolu. – *Äripäev, ITuudised.ee*, 11. september 2018. [WWW] https://www.ituudised.ee/arvamused/2018/09/11/tarkvara-pidev-kasutuselevott-tagab-parema-kvaliteedi-ja-rahulolu (13.01.2020)

[3] Siseministeerium, valitsemisala asutused, 27. aprill 2016. [WWW] https://www.siseministeerium.ee/et/organisatsioon-kontaktid/valitsemisala-asutused (13.01.2020)

[4] SMIT, rahvastikuteenuste osakond [WWW] https://www.smit.ee/et/valdkonnad/arendusvaldkond/rahvastikuteenuste-osakond. (13.01.2020)

[5] Siseministeerium, rahvastikuregister, 21. november 2019. [WWW] https://www.siseministeerium.ee/et/eesmark-tegevused/rahvastikutoimingud/rahvastikuregister. (13.01 2020)

[6] Understanding the Bamboo CI Server. – *Atlassian* [WWW] https://confluence.atlassian.com/bamboo/understanding-the-bamboo-ci-server-289277285.html (16.02.2020)

[7] Glenford J. Myers, „The Art of Software Testing, Second Edition“, John Wiley & Sons, Inc., Hoboken, New Jersey, 2004

[8] Black Box Software Testing [WWW] http://www.testingeducation.org/k04/bbst11_2004.pdf (13.02.2019)

[9] Tarkvara testimist käsitlev juhendmaterjal. – *Majandus- ja Kommunikatsiooniministeerium*, 2006 [WWW] https://www.mkm.ee/sites/default/files/tarkvara_testimise_juhis_-_koopia.doc (15.01.2020)

[10] Wysocki K.R, „Effective Project Management. Traditional.Agile. Extreme, Seventh Edition“, John Wiley & Sons, Inc., 2014

[11] 5 Principles of Agile Testing and How Ranorex Fits In [WWW] https://www.ranorex.com/blog/5-agile-testing-principles/ (16.01.2020)

[12] When You Should Choose Manual vs. Automated Testing [WWW] https://www.utest.com/articles/when-you-should-choose-manual-vs-automated-testing (16.01.2020)

[13] Gezinus J. Hidding, Reinventing Methodology: Who Reads It and Why?, Communications of the ACM 40, No. 11, 102-109, 1997

[14] Alistair Cockburn, Agile Software Development, Addison-Wesley, 2001

[15] Agile Methodologies [WWW] https://www.blueprintsys.com/agile-development-101/agile-methodologies (16.02.2020)

[16] End-to-End (E2E) testing for an AngularJS with Protractor and gulp using IBM Cloud DevOps Delivery pipeline [WWW] https://developer.ibm.com/in/2018/08/16/end-end-testing-angularjs-protractor-gulp-usingibm-cloud-devops-delivery-pipeline/ (24.02.2020)

[17] Mida siis õieti analüüsida. – *Trinidad Wiseman* [WWW] https://blog.twn.ee/et/mida-siis-oieti-analuusida (16.02.2020)

[18] Robson, C., 2002. Real World Research: A Resource for Social Scientists and PractitionerResearchers. 2 toim. Oxford: Blackwell

[19] Objektorienteeritud programmeerimine. – *Tartu Ülikool,* 2017 [WWW] https://courses.cs.ut.ee/2017/OOP/fall/Main/IDEGuides (18.02.2020)

[20] IntelliJ IDEA vs Eclipse: Which Is Better for Beginners [WWW] https://blog.codota.com/intellij-idea-vs-eclipse/ (18.02.2020)

[21] IntelliJ IDEA Eclipses Eclipse for Java Programming [WWW] https://www.webucator.com/blog/2019/08/intellij-idea-eclipses-eclipse-for-java-programming/ (18.02.2020)

[22] Raamlepingust tulenevad testimise nõuded. – *Net Group,* 2018 – [WWW] http://nemo/Pages/Testijale (02.04.2019)

[23] DSDM Agile Project Framework Handbook. MoSCoW Prioritisation [WWW] https://www.agilebusiness.org/page/ProjectFramework_10_MoSCoWPrioritisation (19.02.2020)

[24] A Quick JUnit vs TestNG Comparison [WWW] https://www.baeldung.com/junit-vs-testng (18.02.2020)

[25] TestNG Vs JUnit: What's the Difference?[WWW] https://www.guru99.com/junit-vs-testng.html (18.02.2020)

[26] Apellatsioon, Frank. Testimine JUnitiga. Birmingham: Packt Publishing, 2015

[27] JUnit vs. TestNG: Choosing a Framework for Unit Testing [WWW] https://www.stickyminds.com/article/junit-vs-testng-choosing-framework-unit-testing (18.02.2020)

[28] Leotta, M., Clerissi, D., Ricca, F., Spadaro, C. Improving Test Suites Maintainability with the Page Object Pattern: An Industrial Case Study. Luxembourg, 2013

[29] Page Object Model (POM) & Page Factory: Selenium WebDriver Tutorial [WWW] https://www.guru99.com/page-object-model-pom-page-factory-in-selenium-ultimate-guide.html (19.02.2020)

[30] Page Object Model with Page Factory in Selenium – Complete Guide | Software Testing Material [WWW] https://www.softwaretestingmaterial.com/page-object-model/ (19.02.2020)

[31] Page Object Model (POM) with Page Factory in Selenium WebDriver [WWW] https://www.janbasktraining.com/blog/pom-with-page-factory-in-selenium-webdriver/ (19.02.2020)

[32] Automated Selenium Testing Framework Using TestNG & WebDriver [WWW] http://www.faichi.com/blog/automated-selenium-testing-framework-using-testng-webdriver (19.02.2020)

[33] Stakeholder Analysis. – *Mindtools.com* [WWW] https://www.mindtools.com/pages/article/newPPM_07.html (19.02.2020)

[34] Mendelow, A.L. Environmental Scanning – The Impact of the Stakeholder Concept, ICIS 1981

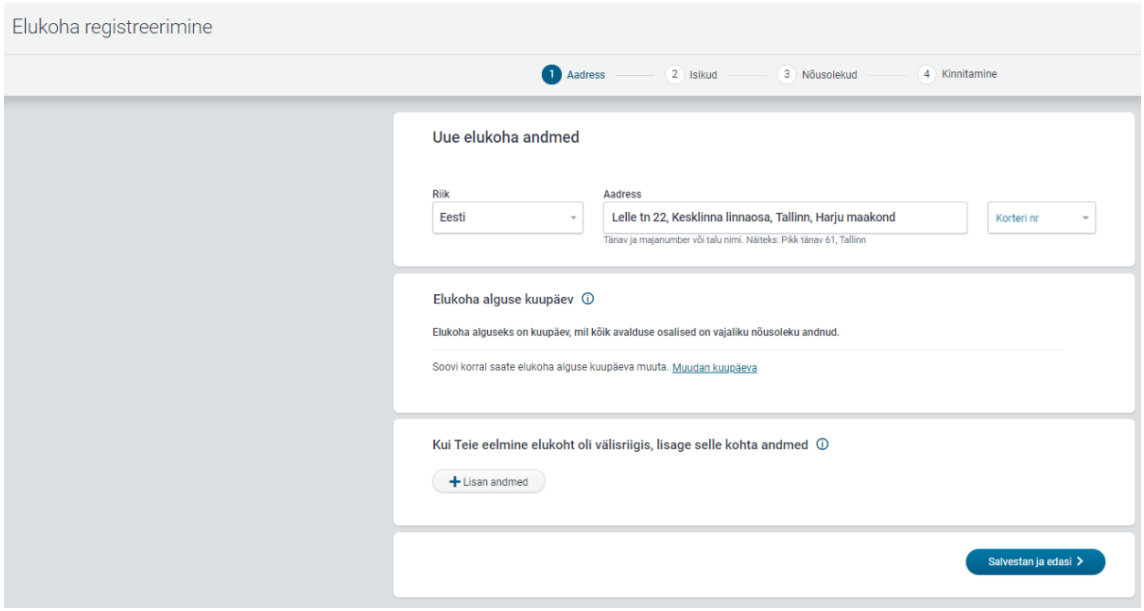
[35] Introduction to the POM [WWW] https://maven.apache.org/guides/introduction/introduction-to-the-pom.html (16.03.2020)

[36] UI Automation with Selenium Helper [WWW] https://medium.com/@jagdale0210/selenium-automation-with-selenium-helper-a421dcf40407 (16.03.2020)

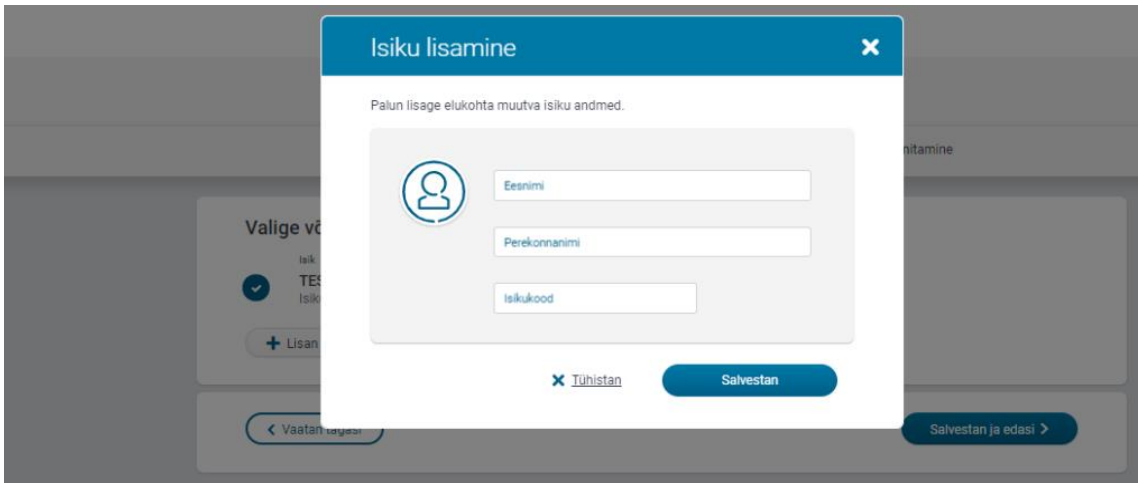
[37] Top 15 UI Test Automation Best Practices You Should Follow [WWW] https://www.blazemeter.com/blog/top-15-ui-test-automation-best-practices-you-should-follow/ (16.03.2020)

[38] D. Glegg ja R. Parker, Case Method Fast-Track: A Rad Approach, Boston: AddisonWesley Publishing Company, 1994

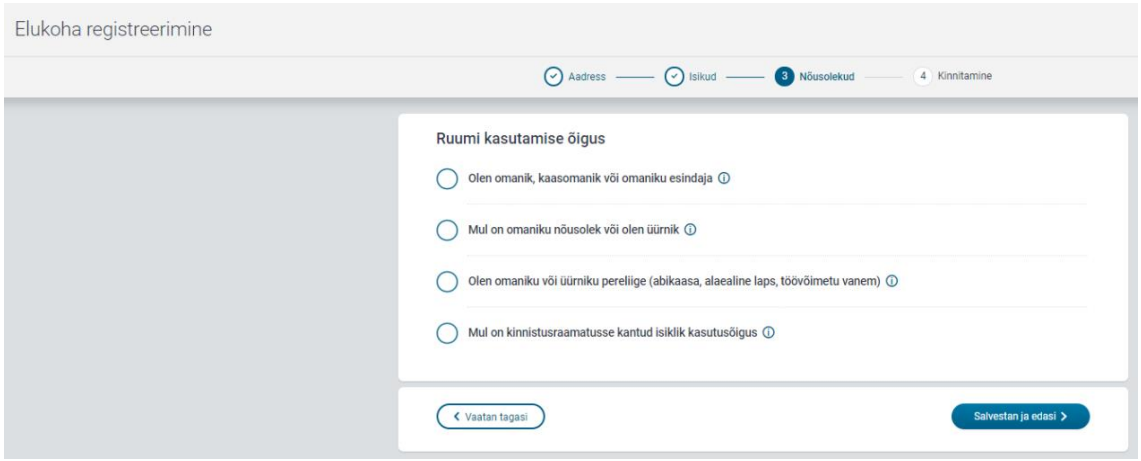
Lisa 1 Elukohateate avalduse aadressi sisestamise samm



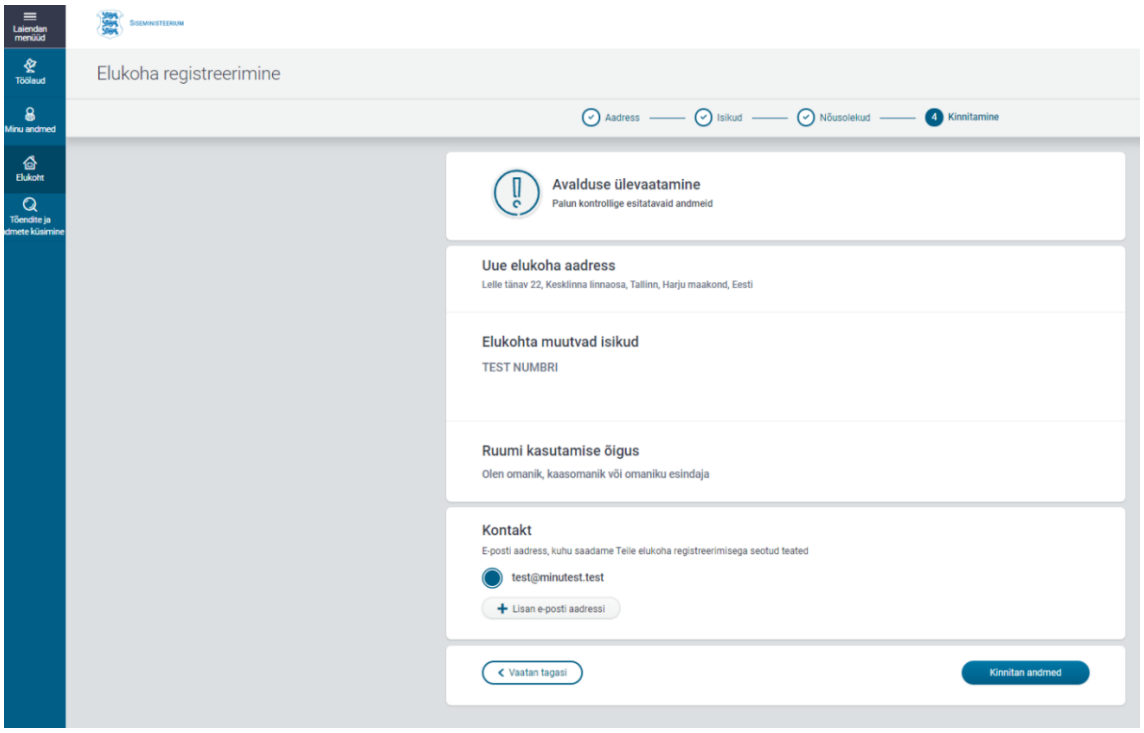
Lisa 2 Elukohateate avalduse isikute lisamise samm



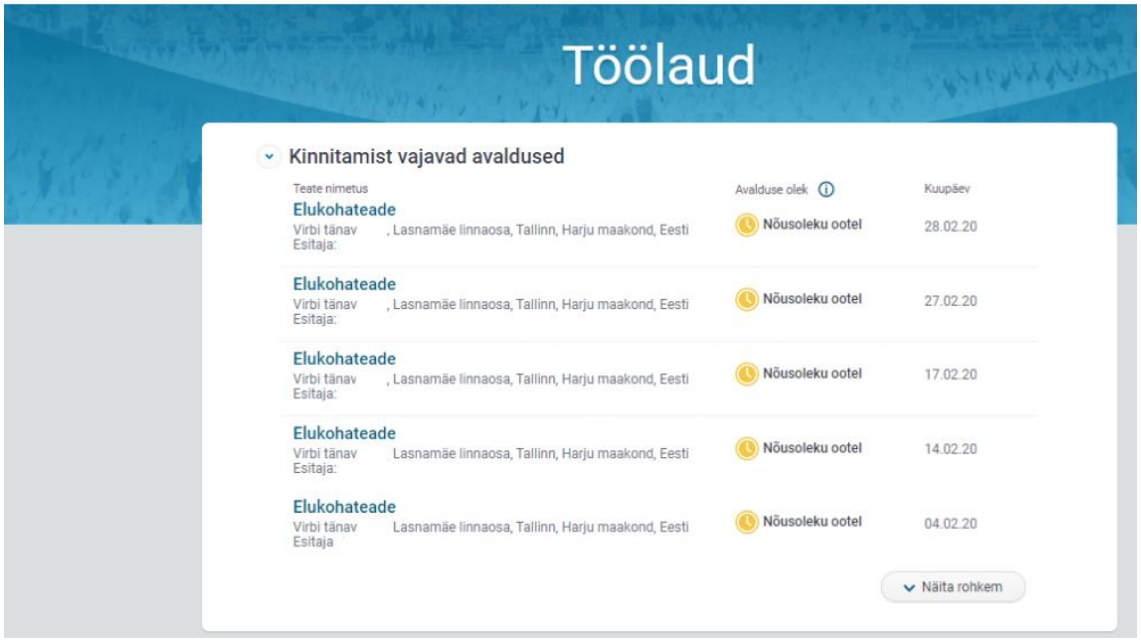
Lisa 3 Elukohateate ruumi kasutamise õiguse valik



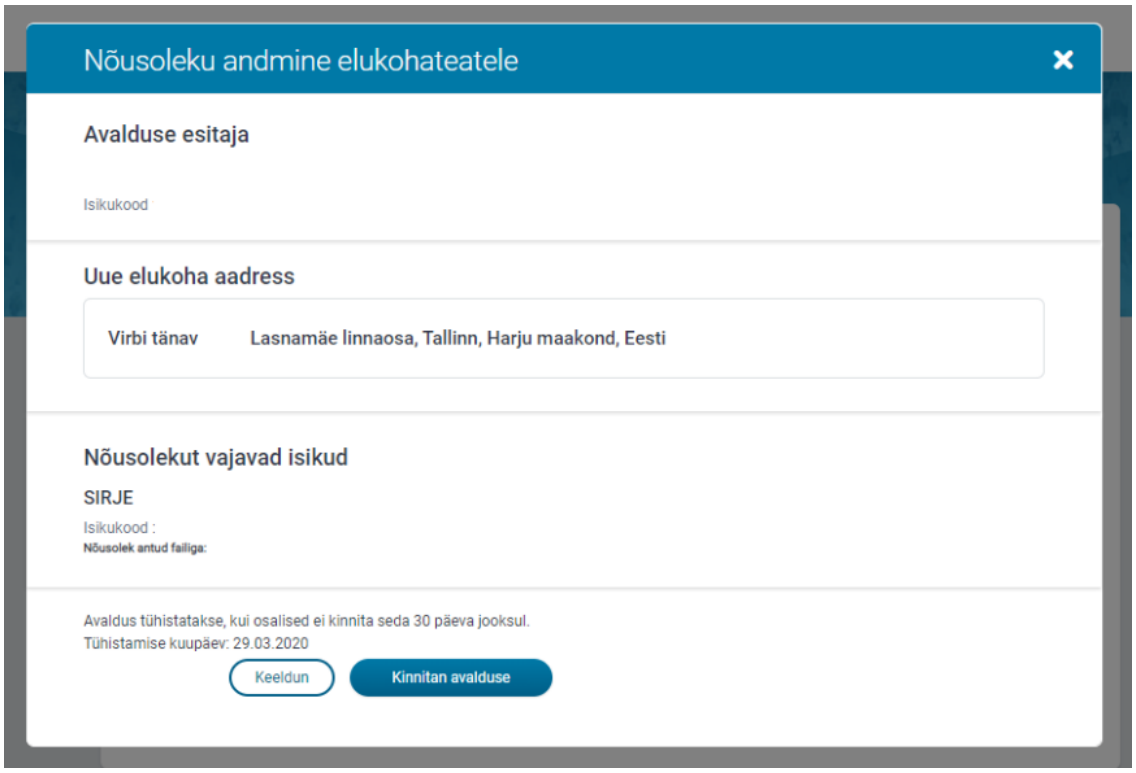
Lisa 4 Elukohateate avalduse kokkuvõtte samm



Lisa 5 E-rahvastikuregistri töölaud



Lisa 6 E-nõusoleku andmise vorm



Lisa 7 ERR pom.xml

```
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://maven.apache.org/POM/4.0.
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0
    <modelVersion>4.0.0</modelVersion>

    <groupId>ERR</groupId>
    <artifactId>selenium-tests</artifactId>
    <version>1.0-SNAPSHOT</version>

    <name>selenium-tests</name>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <testng.version>7.0.0</testng.version>
    </properties>
    <dependencies>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.13.0</version>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-server</artifactId>
        <version>3.13.0</version>
    </dependency>
    <dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
        <version>7.0.0</version>
        <scope>compile</scope>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <!--TestNG MVN run-->
                <version>3.0.0-M4</version>
            </plugin>
            <plugin>
                <artifactId>maven-clean-plugin</artifactId>
                <version>2.4.1</version>
                <configuration>
                    <filesets>
                        <fileset>
                            <directory>test-output</directory>
                            <followSymlinks>false</followSymlinks>
                        </fileset>
                    </filesets>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```