TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
TTÜ IT KOLLEDŽ

Frederick Liin 174932IDAR

RAKENDUSTE UUENDAMISE PROTSESSI OPTIMEERIMINE MUUSEUMIDE INFOSÜSTEEMI NÄITEL

diplomitöö

Juhendaja: Toomas Lepikult
Dotsent

Kaasjuhendaja: Roland Kaur
Rakenduskõrgharidus

Tallinn 2018

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: Frederick Liin

10.01.2018

Annotatsioon

Käesolevas diplomitöös käsitletud probleemiks on aeganõudev rakenduste uuendamise protsess, mis sõltub peale infosüsteemi halduri ka asutuse arendaja ja süsteemiadministraatori vaba aja olemasolust. Diplomitöö eesmärgiks on optimeerida rakenduste uuendamise protsessi. Selleks uurib autor erinevaid viise protsessi sammudes tehtavate tegevuste automatiseerimiseks. Töös keskendutakse Java rakenduste uuendamise protsessi optimeerimisele. Näitena kirjeldatakse ning optimeeritakse Muuseumide Infosüsteemi uuendamise protsessi.

Kasutusele võetud lahendusega eemaldati vajadus pöörduda asutuse arendaja poole kelle ülesandeks protsessis oli lähtekoodi kompileerimine. Tänu sellele toimib kogu protsess sujuvamalt ning kiiremalt, kuna vastutavaid osapooli on vähem ning vajalikud failid paigalduseks on automaatselt olemas kohe peale koodimuudatuse lisamist repositooriumisse.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 23 leheküljel, 6 peatükki, 3 joonist, 6 tabelit.

Abstract

Optimizing the Process of Updating Applications by the Example of the Museum Information System

This diploma thesis focuses on the optimization of a time consuming process of updating applications in the Centre of Registers and Information Systems. The problem that was faced within this thesis was that the process of updating applications takes too much time and because of that, it demotivates application managers and hinders their work. The main focus of this thesis is on Java applications. The update process of the Museum Information System is used as the example.

The goal of this diploma thesis is to optimize the process of updating applications. To achieve the goal background information was presented. In the part of the analysis, the author examines ways of automating certain parts of the process. In the practical part of this thesis, the author installs and configures the solution picked in the part of the analysis.

The goal of this thesis was met. A part of the process was automated which in turn means that there is no more need for the in-house developer to compile the code. Thanks to that, the process as a whole now consumes less time. The files needed to deploy the application are now automatically compiled after each commit to the repository by the outsourced developer.

The thesis is in Estonian and contains 23 pages of text, 6 chapters, 3 figures, 6 tables.

Lühendite ja mõistete sõnastik

GitHub	Suurim ühisarenduslike Git-versioonihaldusega IT-projektide majutuse veebiteenus
HTTP	(„hüperteksti edastuse protokoll“) veebi aluseks olev hüpermeediumile suunatud standardne rakenduskihi protokoll, mis määrab sõnumite vormingu ja edastuse ning veebiserverite ja brauserite käitumise
Kompileerima	kõrgkeeles väljendatud programmi või programmiosa transleerima programmiks, mis on väljendatud vahekeeles, assemblerkeeles või masinakeeles
MuIS	Muuseumide Infosüsteem
Pidevintegratsioon	*Continuous integration* - ekstreemprogrammeerimisse kuuluv arendusmeetod (alates 1991) integratsiooniprobleemide kiireks väljaselgitamiseks komponentide tööeksemplaride sagedate (mitu korda päevas) automatiseeritud koosteoperatsioonidega
Pidevvalmidus	*Continuous delivery* - pidevintegratsiooni edasiarendus, tarkvaraarenduse metoodika, mille järgi luuakse tarkvara lühikeste tsüklitena nii, et seda saab igal ajal (iga tehtud ja testitud muudatuse järel) väljastada
Pistikprogramm	*Plugin* - hõlpsalt paigaldatav olemasoleva tarkvarakomponendi võimalusi laiendav lisandprogramm
Pääsuloend	*Access control list* - loetelu kasutajatest, programmidest ja/või protsessidest ning neile kinnistatud pääsuõiguste andmetest
RIK	Registrite ja Infosüsteemide Keskus
Servlet	Väike serveril töötav programm, tavaliselt Java-rakend veebiserveri keskkonnas
Veebihaak	*Webhook* – kasutaja määratletav HTTP-tagasikutse

Sisukord

Jooniste loetelu

Tabelite loetelu

Sissejuhatus

Registrite ja Infosüsteemide Keskuses, edaspidi RIK, on erinevate rakenduste kiire arendusprotsess ülimalt oluline. Probleem on selles, et hetkel on RIK-is rakendustele uuenduste paigaldamine ajakulukas protsess, sest see sõltub mitme osapoole vaba aja olemasolust. See omakorda aeglustab rakenduste arendusprotsessi, demotiveerib infosüsteemide haldureid ning aeglustab nende tööd.

Probleemi lahendamiseks uurib autor erinevaid viise rakenduste uuendamise protsessis tehtavate sammude automatiseerimiseks. Seejärel paigaldab ning seadistab autor sobiva lahenduse testkeskkonnas.

Kuna lõputöö autori tööülesandeks RIK-is on hallata Java rakendust, milleks on Muuseumide Infosüsteem, siis käsitletakse antud diplomitöös Java rakenduste uuendamise protsessi optimeerimist. Eelnevast tulenevalt on käesoleva diplomitöö eesmärgiks optimeerida Java rakenduste uuendamise protsessi RIK-is niimoodi, et sellele kulutatav aeg väheneks.

Töö esimeses peatükis tutvustab autor olemasolevat olukorda ning kirjeldab hetkel kasutatavat Java rakenduste uuendamise protsessi. Järgnevalt uurib autor erinevaid, hetkel laialdaselt kasutatavaid, lahendusi, mis aitaksid uuendamise protsessi optimeerida. Lõpuks paigaldab ning seadistab autor sobiva lahenduse testkeskkonnas. Kõikideks tõlgeteks ja mõistete seletuseks on kasutatud Standardipõhist tarkvaratehnika sõnastikku ning Andmekaitse ja infoturbe leksikoni [1], [2].

Diplomitöö lahendusest saavad kasu eelkõige:

- **Infosüsteemi haldurid**, kelle töö muutub sujuvamaks, sest osa hallatava rakenduse uuendamise protsessi sammudest on automatiseeritud. See tähendab, et uuendamise protsess võtab vähem aega ning haldurid saavad paremini süveneda uuenduste testimise peale.

- **Asutuse arendajad**, kelle töökoormus väheneb, sest kaob vajadus lähtekoodi kompileerida.
- **Registrite ja Infosüsteemide Keskus**, kuna töökoormuse vähenedes ja töö sujuvamaks muutudes kasvab töötajate rahulolu.

Diplomitöö lähtetingimused:

- Lahendus peab olema vabavaraline – RIK soovib, et probleemi lahendamine ei tekitaks asutusele otseseid lisakulusid. RIK on riigiasutus ning tulenevalt riigireformist soovitakse avaliku sektori kulutusi vähendada või vähemalt samal tasemel hoida [3] [4].
- Lahendus peab olema piisavalt lihtsasti kasutatav – Protsessi optimeerimise rakendamine ei tohi tähendada arendajatele, halduritele ega süsteemiadministraatoritele täiendavat lisakoormust, mis selle kasutuselevõttu ära ei õigustaks.
- Lahendus peab olema liidestatav olemasolevate süsteemidega – Lahendus peab ühilduma RIK-is kasutatava serveritarkvaraga [5].

Lõputöö ülesehitus on jaotatud järgnevalt:

- Olemasolev olukord – Hetkel kasutusel oleva protsessi lühikirjeldus ning taust.
- Analüüs – Probleemi analüüs selle aktuaalsuse seisukohast, olemasoleva protsessi kitsaskohad, lahenduste variandid ning administraatorite küsitlemine.
- Praktiline osa – Testkeskkonna seadistamine, lahenduse paigaldamine ja seadistamine.
- Ülevaade tulemustest – Diplomitöö tulemuse analüüs ning järeldused.
- Edasised tegevused – Võimalike edasiste tegevuste analüüs.
- Kokkuvõte

1 Olemasolev olukord

RIK on Justiitsministeeriumi haldusala asutus, mis arendab ja haldab riigile ning kodanikele väga olulisi registreid ja infosüsteeme [6]. Lõputöö autor töötab RIK-is infosüsteemi haldurina ning lõputöö teema kujunes välja töötamise käigus ette tulnud probleemist. Probleemiks on liialt aeganõudev rakenduste uuendamise protsess, mis sõltub peale halduri ka asutusesisese arendaja ning süsteemiadministraatori vaba aja olemasolust. Kuna asutuse arendaja põhitööülesandeks on lähtekoodi kirjutamine erinevates arendusprojektides, siis võib põhitöö kõrvalt mõne teise projekti kompileerimine olla vähem prioriteetsem. Sellest tulenevalt võib juhtuda, et asutuse arendaja saab kompileerimisega alles mõne tunni pärast tegeleda. Süsteemiadministraator võib samuti olla hõivatud mõne muu prioriteetse tööga nagu näiteks tõrgete eemaldamisega mõnest tähtsast infosüsteemist.

Diplomitöö autor on Muuseumide Infosüsteemi, edaspidi MuIS, haldur. MuIS on veebipõhine töökeskkond, mis on mõeldud muuseumikogude haldamiseks ja riigivara üle arvestuse pidamiseks ning võimaldab teha muuseumides leiduva informatsiooni kättesaadavaks nii erialaspetsialistile kui kõigile teistele huvilistele [7], [8]. MuIS, nagu paljud teised rakendused, on pidevalt täienev ja vajab lisaarendusi ning olemasolevate vigade parandusi. MuIS on kirjutatud kasutades Java programmeerimiskeelt. RIK kasutab Java rakenduste arendamiseks väliseid arendajaid, sest ettevõttes puudub Java arendaja. Java on platvormist sõltumatu programmeerimiskeel, mis vajab kompileerimist baitkoodi (ingl k *bytecode*) [9]. RIK-is nõue, et väliste arendajate poolt tehtud tööd antakse üle koodi kujul koodirepositooriumisse. See lisab uuendamise protsessile juurde ühe sammu, milleks on koodi kompileerimine. Sellest tulenevalt on Java rakenduste uuendamise protsess RIK-is ajakulukam võrreldes teistes programmeerimiskeeltes kirjutatud rakendustega.

Rakenduste uuendamise protsessi kiirus on RIK-is väga oluline. Üheks põhjuseks on see, et väliste arendajatega koostööd tehes on hankelepingutega paika pandud kindlad kohustused, mida mõlemad osapooled peavad täitma kindla ajaperioodi vältel. Näiteks

kui väline arendaja on tööd üle andnud, siis on RIK-il õigus üleantud tööd üle vaadata 20 tööpäeva jooksul. Juhul, kui RIK leiab, et töö ei vasta lepingu tingimustele, on RIK kui tellija kohustatud teavitama arendajat töös avastatud puudustest. Arendaja ehk töö täitja on kohustatud puudused kõrvaldama 5 tööpäeva jooksul, kui pooled ei ole kokku leppinud täiendavat tähtaega. Puuduste kõrvaldamise kulud kannab arendaja. Kui RIK on tööd vastu võtnud, siis algab koheselt garantiiperiood. Detailsem info tööde üleandmise ja vastuvõtmise ning garantiiperioodi kohta, mille autor sai oma talituse juhilt, asub vastavalt **Lisa 1** ja **Lisa 2**. Lisaks on rakenduste uuendamise protsessi kiirus RIK-is väga oluline, sest RIK vastutab mitmete riigile oluliste registrite eest [6]. Sellest tulenevalt on töö seisma jäämise ning andmete terviklikkuse säilimise nimel ülimalt oluline kiire reageerimine ning probleemi lahendamine, et vältida suuri kahjusid.

1.1 Uuendamise protsess praegu

Järgnevalt kirjeldab autor MuIS-i uuendamise protsessi. Protsessi sisendiks on vajalik välise arendaja poolt üle antud koodimuudatus. Rollid protsessis ja nende tegevused on välja toodud tabelis **Tabel 1**.

Tabel 1. Rollid hetkel kasutatavas protsessis ja nende tegevused.

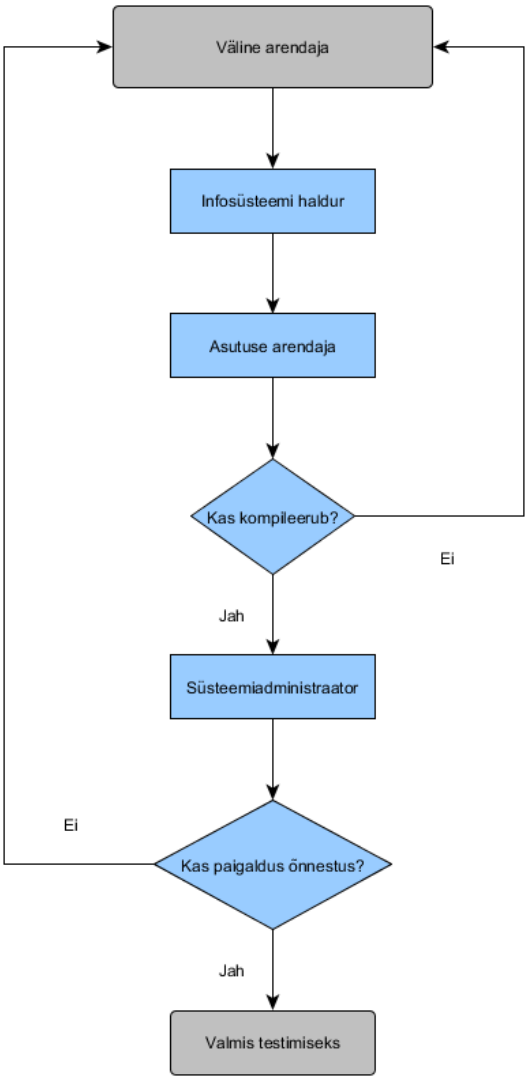
Roll	**Tegevus**
Infosüsteemi haldur	Koordineerib rakenduse arendusprotsessi ja tegeleb tarkvara testimisega
Asutusesisene arendaja	Kompileerib lähtekoodi
Süsteemiadministraator	Paigaldab uuenduse

Hetkel koosneb uuendamise protsess kolmest põhisammust. Protsessist ülevaate saamiseks on loodud töövoog joonisel **Joonis 1**. Järgnevalt seletab autor lahti uuendamise protsessi sammudes tehtavad tegevused:

1. Infosüsteemi haldur alustab uuendamise protsessi saates uuendatud/muudetud koodi asutusesisesele arendajale kompileerimiseks. Kui kood ei kompileeru, siis suunatakse see tagasi välisele arendajale parandamiseks.
2. Kui kood on kompileeritud, siis teavitatakse sellest infosüsteemi haldurit ning seejärel edastatakse paigalduspakk süsteemiadministraatorile kes uuendust testkeskkonda paigaldama hakkab. Kui uuenduse paigaldamise järgselt selgub, et

rakendus ei tööta korrektselt, siis kogutakse kokku logifailid ning probleem suunatakse välisele arendajale lahendamiseks.

3. Kui uuendus on paigaldatud testkeskkonda, siis võib lugeda uuendamise protsessi lõppenuks ning haldur saab hakata testima kas tellitud muudatused vastavad nõuetele.



Joonis 1. Uuendamise protsess hetkel.

1.1.1 Hetkel kuluv aeg

Töö autor fikseeris ühe kuu jooksul MuIS-i uuendamiseks kuluva aja alates hetkest kui asutuse arendajale oli edastatud kiri, millega oli kaasas viide kompileerimist vajavale koodimuudatusele, kuni hetkeni millal uuendus oli valmis halduri poolt testimiseks testkeskkonnas. Järgnevas tabelis **Tabel 2** on kirjas MuIS-i töötajakeskkonna uuendamise peale kulunud aeg. **Algus** tähistab aega, millal haldur saatis välja kirja asutuse arendajale,

et kood kompileeritaks ning **lõpp** tähistab aega, millal haldurini jõudis kinnitus administraatori poolt, et uuendus on paigaldatud ning valmis testimiseks testkeskkonnas. **Kompileeritud** tähistab aega, millal asutuse arendaja saatis kinnituse, et paigalduspakk on valmis administraatorile edastamiseks.

Tabel 2. Hetkel MuIS-i uuendamise peale kuluv aeg.

Versioon	**Algus**	**Kompileeritud**	**Lõpp**	**Kokku**
0.9.58	23.10.2017 11:36	23.10.2017 13:16	23.10.2017 14:03	2 h 27 min
0.9.58.2	27.10.2017 09:22	27.10.2017 09:49	27.10.2017 11:10	1 h 48 min
0.9.59.1	13.11.2017 10:49	13.11.2017 11:07	13.11.2017 13:28	2 h 39 min
0.9.60.1	13.11.2017 14:12	13.11.2017 14:44	13.11.2017 15:28	1 h 16 min
0.9.60.2	16.11.2017 14:25	17.11.2017 08:51	17.11.2017 10:10	3 h 45 min
0.9.62.1	21.11.2017 14:34	21.11.2017 15:17	22.11.2017 13:18	6 h 44 min

Eelmainitud tabelist on näha, et keskmiselt kestab MuIS-i uuenduste paigaldamise protsess ühe kuu jooksul 18 tundi ja 39 minutit, millest kompileerimise ootamise peale kulub 6 tundi ja 15 minutit ning paigaldamise ootamise peale 12 tundi ja 24 minutit.

2 Analüüs

Selles peatükis uurib töö autor erinevaid viise, mis aitaksid uuendamise protsessi optimeerida. Selleks uurib autor võimalusi uuendamise protsessis tehtavate sammude automatiseerimiseks. Veel seletab autor lahti automatiseerimise olulisuse ning toob välja automatiseerimise kasutegurid. Lisaks seletab autor, mida hõlmab endas pidevintegratsioon (ingl k *continuous integration*) ning pidevvalmidus (ingl k *continuous delivery*). Lõpuks, toetudes lõputöö lähtetingimustele ning läbi viidud küsitlusele, teeb autor valiku sobivate tarkvaralahenduste seast.

2.1 Automatiseerimise olulisus

Suurem osa ettevõtteid otsivad aktiivselt viise kuidas vähendada oma kulusid. Automatiseerimine aitab kokku hoida mitmel viisil. Automatiseerimine suurendab tootlust, vähendab vigade tegemise arvu ning aitab vähendada kaasatava personali arvu, mis omakorda vähendab tööjõukulusid. Automatiseerides töövoogusid eemaldatakse vajadus manuaalse töö vastu [10]. Tihti täidetakse automatiseeritud tööülesanne kiiremini kui sama tööülesannet täita manuaalselt. Lisaks on kindel, et tööülesande täitmisel järgitakse alati ette seatud reegleid [11]. See parandab tootlust ja vabastab töötajate aega, tänu millele saab nende oskusi ning teadmisi rakendada mõne tähtsama ülesande või projekti raames. Paljud IT toimingud sisaldavad tüütuid ja korduvaid tegevusi, mis omakorda suurendavad tõenäosust vigade tekkimiseks. Kui sellistel puhkudel eemaldada inimfaktor, siis kaovad ka kulukad vead [10]. Ettevõte nimega Plan B on läbi viinud uuringu, et analüüsida võtmetegureid, mis põhjustavad suuremahulisi IT intsidente ja teenuste rikkeid väikestes ja keskmise suurusega ettevõtetes. Uuringu tulemused näitavad, et inimlik viga põhjustas 47 protsenti intsidentidest [12].

2.2 Kompileerimise automatiseerimise võimalused

Nagu eelmainitud, siis võib kompileerimine võtta keskmisest kauem aega kui asutuse arendaja on hõivatud mõne muu, prioriteetsema tööga. Kui kompileerimine toimuks

automaatselt peale koodimuudatuse repositooriumisse saatmist, siis kaoks täielikult ära vajadus kaasata asutuse arendajat lähtekoodi kompileerimises, mis oluliselt kiirendaks uuendamise protsessi. Toetudes andmetele tabelis **Tabel 2** võtaks uuendamise protsess kuu aja jooksul keskmiselt 6 tundi ja 15 minutit vähem aega kui kompileerimine toimuks automaatselt.

Uurides võimalusi lähtekoodi kompileerimise automatiseerimiseks leidis töö autor, et kompileerimist saab automatiseerida kasutades selleks näiteks kompileerimistööriista, mis kasutab koosteskripte (ingl k *build script*). Koosteskriptidega saab automatiseerida lihtsad korduvad tegevused ning salvestada kompileerimiseks vajalikud suvandid [13], [14], [15]. Teine võimalus, millega on võimalik kompileerimise protsessi täielikult automatiseerida, oleks koosteserveri (ingl k *build server*) ehk pidevintegratsiooni võimaldava tarkvara kasutamine [16].

Et teada saada kuidas hetkel kompileerimine toimub küsis autor MuIS-i kompileerimise eest vastutavalt asutuse arendajalt, millist tööriista ta MuIS-i töötajakeskkonna kompileerimiseks kasutab. Kompileerimiseks kasutab arendaja Apache Ant nimelist tööriista. Apache Ant on Java teek ning käsurea tööriist, mille põhikasutusala on Java rakenduste kompileerimine. Apache Ant kasutab kompileerimiseks koostefaili (ingl k *buildfile*), millega on ette seatud reeglid, mille järgi kompileerimine toimub [17]. MuIS-i puhul asub koostefail koodirepositooriumis ning selle korrektsuse eest vastutab väline arendaja. Koostefailiga on juba kompileerimist lihtsustatud, kuid asutuse arendaja peab ikka koodi manuaalselt repositooriumist alla laadima ning seejärel kompileerimistööriista käivitama.

Koosteskriptidega on kompileerimise protsessi juba lihtsustatud, kuid pidevintegratsiooni võimaldava tarkvara kasutusele võtmisega saaks protsessi täielikult automatiseerida. Sellest tulenevalt otsustas autor, et kasutusele tuleks võtta pidevintegratsiooni võimaldav tarkvara. Järgnevas alapeatükis kirjeldab autor, mida antud lahendus endast täpselt kujutab.

2.2.1 Pidevintegratsioon

Arendajad, kes praktiseerivad pidevintegratsiooni, mestivad (ingl k *merge*) enda tehtud koodimuudatusi nii tihti kui võimalik peaharusse (ingl k *main branch*). Arendaja tehtud muudatused valideeritakse, tekitades uuest koodist redaktsioon, mille vastu käivitatakse

automaattestid. Niimoodi tehes välditakse integratsiooni põrgut (ingl k *integration hell*), mis tavaliselt juhtub kui oodatakse väljastamise päevani (ingl k *release day*), et oma kood mestida väljalaskeharusse. Pidevintegratsioon paneb rõhu testimise automatiseerimisele, et kontrollida kas rakendus töötab korrektselt peale muudatuste mestimist peaharusse [18].

Pidevintegratsiooni lahendus sobib autori arvates väga hästi, sest võttes arvesse RIK-i nõuet, et väliste arendajate poolt tehtud tööd antakse üle koodi kujul, siis selle lahendusega jääb nõue kehtima, kuid samal ajal toimub peale igat koodimuudatust automaatne koodi kompileerimine, tänu millele on alati olemas juba paigaldamiseks vajalikud failid. Antud lahenduse plussiks on veel see, et kui kood ei kompileeru, siis on võimalik seadistada enamus pidevintegratsiooni võimaldavat tarkvara nii, et arendajat teavitatakse probleemist automaatselt [19], [20], [21].

2.3 Paigaldamise automatiseerimise võimalused

Kui uuenduse paigaldamine testkeskkonda toimuks automaatselt peale lähtekoodi kompileerimist, siis kaoks ära vajadus kaasata süsteemiadministraatorit uuenduste paigaldamises, mis oluliselt kiirendaks uuendamise protsessi. Toetudes andmetele tabelis **Tabel 2** võtaks uuendamise protsess kuu aja jooksul keskmiselt 12 tundi ja 24 minutit vähem aega kui uuenduste paigaldus testkeskkonda toimuks automaatselt.

Uurides võimalusi paigaldamise automatiseerimiseks leidis autor, et lisaks automaatsele lähtekoodi kompileerimisele pakuvad enamus pidevintegratsiooni võimaldavad lahendused ka pidevvalmiduse funktsionaalsust, mis on mõeldud paigaldamise protsessi täielikuks automatiseerimiseks [22]. Järgnevas alapeatükis seletab autor, mida antud lahendus endast täpsemalt kujutab.

2.3.1 Pidevvalmidus

Pidevvalmidus on pidevintegratsiooni edasiarendus, mis teeb kindlaks, et uued muudatused lähtekoodis on valmis kiirelt klientidele väljastamiseks jätkusuutlikul viisil. See tähendab, et peale kompileerimise automatiseerimise on ka väljastamise protsess automatiseeritud ning rakendust saab evitada toodangkeskkonda nii vara kui võimalik. Nii on kindel, et väljastatakse väiksemaid pakke kust on kergem, probleemide korral, vigu leida ning parandada [18].

2.3.2 Pidevvalmiduse sobivus RIK-is

Rakendustele uuenduste paigaldamise protsess koosneb tihti mitmest erinevast tegevusest. Erinevalt kompileerimisest võivad uuenduste paigaldamise protsessis tehtavad tegevused erineda sõltuvalt uuenduse mahust ja sisust. Et teada saada kas rakenduste paigaldamise puhul on hetkel otstarbekas midagi muuta viis lõputöö autor RIK-i süsteemiadministraatorite seas läbi küsitluse, millele vastas 9 administraatorit. Küsimustiku vastused asuvad **Lisa 3**. Järgnevas tabelis **Tabel 3** loetleb autor küsitluses esitatud küsimused ning seejärel annab ülevaate küsitluse tulemustest.

Tabel 3. Küsimused RIK-i süsteemiadministraatoritele.

Küsimuse number	**Küsimus**
Küsimus nr 1	Mitut erinevat infosüsteemi/rakendust administreerid?
Küsimus nr 2	Mis tüüpi rakendustega on tegu (nt. java, python, php)?
Küsimus nr 3	Kui kaua võtab keskmiselt uue versiooni paigaldamine?
Küsimus nr 4	Kui rahul oled praeguse uuenduste paigaldamise protsessiga skaalal 1 - 10?
Küsimus nr 5	Kas oled kuulnud pidevintegratsioonist ja -valmidusest?
Küsimus nr 6	Kui vastasid eelmisele küsimusele JAH, siis märgi kas oled kuulnud nendest või kasutanud järgmisi pidevintegratsiooni ja -valmidust võimaldavaid tööriistu (Jenkins, Bamboo, TeamCity, Travis CI, GoCD)?
Küsimus nr 7	Kas oleksid huvitatud sellise lahenduse kasutamisest RIK-is? Kui JAH, siis kirjuta millist eelistaksid (ei pea olema eelmises küsimuses loetletud lahendused). Kui EI, siis palun põhjenda lühidalt.

Küsitluse tulemustest selgub, et ühe administraatori poolt hallatavate rakenduste arv ulatub kuuest kuni neljakümneni. Kuus administraatorit üheksast tegelevad Java rakendustega. Enamus RIK-i administraatoreid on rahul praeguse uuendamise protsessiga. Kõik peale ühe vastanud administraatoritest on kuulnud pidevintegratsioonist ja –valmidusest. Kõige tuntum automatiseerimislahendus RIK-i administraatorite seas on

Jenkins. Enamus RIK-i administraatoritest on kuulnud pidevintegratsiooni ja –valmiduse lahendusest, kuid enamus ei pea vajalikuks selle lahenduse kasutusele võtmist. Veel loeb põhjendustest välja, et enamus tegevusi on administraatoritel juba skriptide abil automatiseeritud.

Lähtudes RIK-i süsteemiadministraatorite seas läbi viidud küsitluse tulemustest ning lõputöö lähtetingimusest, et lahendus ei tohiks tekitada osapooltele lisakoormust otsustas lõputöö autor, et pidevvalmiduse kasutusele võtmine ei ole hetkel vajalik. See tähendab, et paigaldamise automatiseerimist käesolevas diplomitöös enam ei käsitleta.

2.4 Metoodika

Autor on valinud diplomitöös esitatud probleemi lahendamiseks välja 5 pidevintegratsiooni võimaldavat tarkvara. Tabelis **Tabel 4** kirjeldab autor nõudeid, millele valitav tarkvara peab vastama.

Tabel 4. Nõuded tarkvarale.

Nr	**Nõue**
Nõude kirjeldus	
1	**Lahendus peab olema vabavaraline.**
RIK soovib, et probleemi lahendamine ei tekitaks asutusele otseseid lisakulusid.	
2	**Lahendus peab olema ühilduv RIK-is kasutatava serveritarkvaraga.**
Lahendus peab ühilduma RIK-is kasutatava serveritarkvaraga [5].	
3	**Lahendus peab olema piisavalt lihtsasti kasutatav.**
Protsessi optimeerimise rakendamine ei tohi tähendada arendajatele, halduritele ega süsteemiadministraatoritele täiendavat lisakoormust, mis selle kasutuselevõttu ära ei õigustaks.	
4	**Lahendus peab sobima täitma seatud eesmärki.**
Tarkvara peab võimaldama kompileerimise protsessi täielikku automatiseerimist.	

Tabelis **Tabel 4** välja toodud nõuded on kombinatsioon autori poolt seatud nõuetest valitavale tarkvarale ning RIK-is järgitavatest parimatest praktikatest.

2.5 Pidevintegratsiooni võimaldavad tarkvarad

Selles alapeatükis kirjeldab autor pidevintegratsiooni võimaldavaid tarkvarasid, mis sobiksid lahendama lõputöö lähteülesannet. Toetudes materjalile, mis käsitleb enimkasutatavaid pidevintegratsiooni võimaldavaid lahendusi on autor valinud viis tuntud toodet mille seast teha valik. Nendeks on: Jenkins, Bamboo, TeamCity, Travis CI ning GoCD [23], [24], [25].

2.5.1 Jenkins

Jenkins on avatud lähtekoodiga pidevintegratsiooni server, mis on kirjutatud kasutades Java programmeerimiskeelt. Jenkins on platvormist sõltumatu ning pakub konfigureerimist läbi graafilise kasutajaliidese kui ka läbi konsooli käskude. Jenkinsi teeb väga paindlikuks võimalus tema lisaomadusi laiendada läbi pistikprogrammide (ingl k *plugin*). Jenkinsi pistikprogrammide nimekiri on väga ulatuslik. Jenkins on väljastatud MIT litsentsi alt, mis tähendab, et ta on tasuta kasutatav ning levitatav [23], [26], [27].

2.5.2 Bamboo

Bamboo on Atlassiani poolt pakutav pidevintegratsiooni lahendus. Bambood saab tasuta proovida 30 päeva jooksul ja peale seda saab teha valiku kahe väikestele ning kasvavatele tiimidele mõeldud paketi vahel. Olles välja töötatud Atlassiani poolt, on Bambool JIRA ja Bitbucketi tugi ning lisaks on võimalus lihtsalt importida oma Jenkinsi konfiguratsioonid [23], [28].

2.5.3 TeamCity

TeamCity on intelligentne pidevintegratsiooni lahendus igas suuruses tarkvara keskkondadele [24]. TeamCity pakub kõiki oma funktsioone tarkvara tasuta versioonis, kuid on limiteeritud kahekümnele konfiguratsioonile ja kolmele koosteprogrammile (ingl k *build agent*). Kui on vaja kasutada rohkem konfiguratsioone ning koosteprogramme, siis nende eest tuleb maksta [23], [29].

2.5.4 Travis CI

Travis CI on pidevintegratsiooni platvorm, mis automatiseerib rakenduste paigaldamise ning tarkvara testimise protsessid. Travis CI on loodud platvormina, mis integreerub kasutaja GitHub-i projektiga, et koodi testimine toimuks koodimuudatustega käsikäes [24]. Travis CI on tasuta kõikidele avatud lähtekoodiga projektidele ning teistele

projektidele on tasuta esimesed 100 konfiguratsiooni [23]. Travis CI-d kasutab mitu suurt veebipõhist firmat nagu Facebook, Mozilla ja Twitter [25], [30].

2.5.5 GoCD

GoCD on avatud lähtekoodiga pidevintegratsiooni lahendus, mis on ideaalne testimistsüklite automatiseerimiseks ja sujuvamaks muutmiseks. GoCD teeb teiste lahenduste kõrval eriliseks konveierite kontseptsioon, mis teeb keeruliste töövoogude modelleerimise lihtsaks. GoCD on tasuta allalaetav, kuid kasutajatugi ning mõned pistikprogrammid on tasulised [23], [24], [31].

2.6 Valiku tegemine

Võttes aluseks alapeatükis 2.4 toodud nõuded on loodud järgnev tabel **Tabel 5** illustreerimaks tarkvara valikute sobivust. Selle tabeli abil teeb autor valiku.

Tabel 5. Tarkvara nõuetele vastavus.

Nõue (vt Tabel 4)	**Jenkins**	**Bamboo**	**TeamCity**	**Travis CI**	**GoCD**
Nõue 1	Jah	Ei	Ei. Tasuta on esimesed 20 konfiguratsiooni ning kolm koosteprogrammi.	Ei. Tasuta avatud lähtekoodiga projektidele. Teistele projektidele on tasuta esimesed 100 konfiguratsiooni.	Jah, aga kasutajatugi on tasuline. Lisaks on mõned pistikprogram-mid tasulised.
Nõue 2	Jah	Jah	Jah	Jah	Jah
Nõue 3	Jah	Jah	Jah	Jah	Jah
Nõue 4	Jah	Jah	Jah	Jah	Jah

Eelmainitud tabelist on näha, et Bamboo ei ole vabavaraline ning seetõttu ei vasta ta ühele lõputöö lähtetingimusele, milleks on see, et lahendus peab olema vabavaraline. Sellest tulenevalt otsustas autor Bambood mitte valida. TeamCity on piiratud kahekümnele konfiguratsioonile ning kolmele koosteprogrammile. Võttes arvesse, et RIK-i hallata on suur hulk registreid ja infosüsteeme, mille arv asutuse laienedes tulevikus kasvab, siis otsustas autor, et TeamCity ei ole sobiv lahendus. Travis CI on tasuta avatud lähtekoodiga

projektidele. Teistele projektidele on tasuta esimesed 100 konfiguratsiooni. Kuna RIK-i hallata on suur hulk riigile olulisi infosüsteeme, mis ei ole avatud lähtekoodiga, siis otsustas autor, et Travis CI ei ole sobiv lahendus. GoCD on tasuta kasutatav, kuid kasutajatugi on tasuline. Lisaks ei ole kõik GoCD pistikprogrammid tasuta. Neid piiranguid arvesse võttes otsustas autor GoCD lahendust mitte valida. Jenkins on juhtiv avatud lähtekoodiga pidevintegratsiooni lahendus ning sobib ideaalselt lahendama lõputöös esile toodud probleemi [32]. Valdavas osas pidevintegratsiooni puudutavaid allikaid on Jenkinsi järjestanud just esimesele kohale [23], [24], [25]. Jenkinsi eeliseks on veel see, et ta on väga tuntud lahendus suure kasutajaskonnaga, mis tähendab, et tihti lisatakse uusi funktsionaalsusi ning parandusi, mis lahendavad aktuaalseid probleeme [33]. Võttes aluseks Jenkinsi vastavuse autori poolt kirjeldatud nõuetele alapeatükis Metoodika ning süsteemiadministraatorite seas läbi viidud küsitluse tulemused, otsustas autor valida Jenkinsi.

3 Praktiline teostus

Selles peatükis kirjeldab diplomitöö autor testkeskkonna seadistamist ning lõputöö analüüsi osas valitud tarkvara paigaldamist ja seadistamist. Testkeskkond loodi käesoleva diplomitöö tarbeks RIK-i süsteemiadministraatori poolt. Testkeskkond asub RIK-i sisevõrgus ning serveri IP aadress on 10.1.210.53. Lisaks on serverisse süsteemiadministraatori poolt paigaldatud operatsioonisüsteem CentOS 7 [34]. CentOS operatsioonisüsteem on üks põhilistest kasutusel olevatest Linux distributsioonidest RIK-is [5]. Lisaks on testimise eesmärgil autori poolt loodud privaatne koodirepositoorium kuhu on lisatud MuIS-i töötajakeskkonna lähtekood.

Loodud lahendus automatiseerib täielikult ühe uuendamise protsessi sammu ehk lähtekoodi kompileerimise. See lihtsustab haldurite tööd, vähendab kogu protsessile kulutatavat aega ning vähendab RIK-i arendajate töökoormust.

3.1 Testkeskkonna seadistamine

Testkeskkonda on paigaldatud Apache Tomcat, edaspidi Tomcat, ning Tomcati tööks vajalik Java arenduskomplekt. Tomcat on Java rakendusserver, mille funktsionaalsuste hulka kuulub Java servlettide kasutamine ja HTTP protokolli tugi [35]. Lisaks on Tomcat üks põhilistest rakendusserveritest, mida RIK-is kasutatakse [5]. Serverisse on loodud eraldi kasutaja ning grupp nimega *tomcat*, et Tomcat töötaks eraldi kasutaja alt. Tomcat kasutab vaikeseadistusena porti 8080. Lisaks on seadistatud vajaminevad õigused vastavatele kataloogidele. Sisestatud käsud antud tegevuste läbiviimiseks asuvad **Lisa 4**.

3.2 Jenkinsi paigaldamine ning seadistamine

Jenkinsi paigaldamiseks on alla laetud lõputöö kirjutamise hetkel uusim versioon Jenkins tarkvarast. Kuna Jenkins vajab pistikprogrammide alla laadimiseks ligipääsu välisele võrgule ning RIK-i sisevõrk kasutab välise võrguga suhtluseks vaheserverit (ingl k *proxy server*), siis on Jenkinsi seadetes konfigureeritud vaheserveri sätted. Vaheserveri põhiliseks ülesandeks on pakkuda võrguteenust, mille abil saaks klient teha kaudseid

ühendusi võrguteenustega, mis asuvad teistes serverites [36]. Kuvatõmmis vaheserveri sätete konfigureerimisest asub **Lisa 5**.

Et Jenkins pääseks ligi kompileerimist vajavale lähtekoodile, mis asub autori poolt loodud repositooriumis, on Jenkinsile paigaldatud GIT pistikprogramm. See pistikprogramm laeb alla ning võimaldab Jenkinsil kasutada GIT versioonihaldustarkvara [37]. GIT on vabavaraline versioonihaldustarkvara, mis loodi algselt Linus Torvaldsi poolt Linuxi kerneli arenduse tarbeks [38], [39]. Kui serverisse on eelnevalt paigaldatud eraldi GIT pakk, siis ei ole GIT pistikprogrammi vaja paigaldada, vaid kasutada saab juba olemasolevat GIT installatsiooni. Selleks tuleb Jenkinsi seadetes määrata serveris asuva GIT installatsiooni asukoht [37].

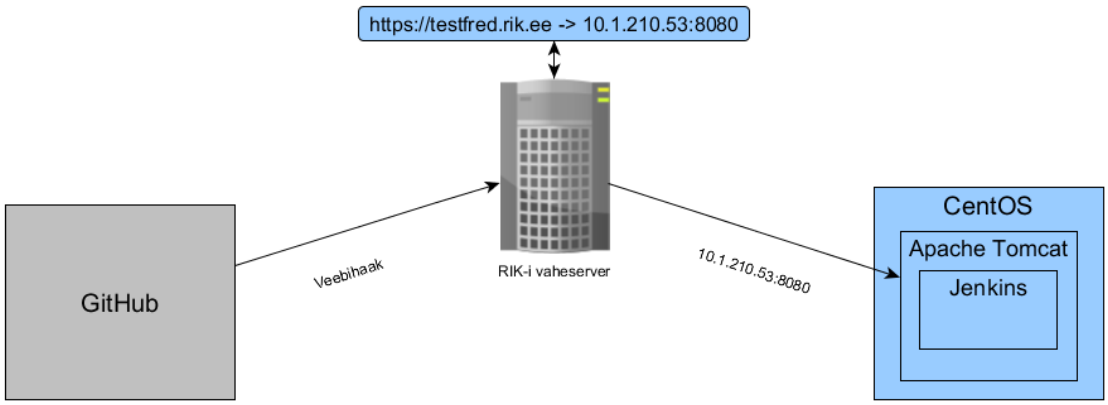
Et Jenkinsit saaks seadistada lähtekoodi kompileerima on Jenkinsile paigaldatud ka Ant pistikprogramm. Ant pistikprogrammiga lisandub Jenkinsile Apache Ant tugi. See funktsionaalsus oli varem osa Jenkinsi installatsioonist, kuid alates versioonist 1.431 see eraldati ning loodi pistikprogrammina [40]. Kuvatõmmis pistikprogrammide paigaldamisest asub **Lisa 6**. Kui serverisse on eelnevalt juba paigaldatud Apache Ant, siis ei ole Ant pistikprogrammi vaja paigaldada, vaid kasutada saab juba olemasolevat Ant installatsiooni. Selleks tuleb Jenkinsi seadetes määrata serveris asuva Ant installatsiooni asukoht [40].

3.3 Uue projekti loomine Jenkinsis

Järgnevalt kirjeldab autor uue projekti loomist Jenkinsis. Projekti üldistes sätetes on märgitud, et tegu on GitHub-i projektiga ning seadistatud on repositooriumi aadress. Järgmisena on määratud lähtekoodi haldamise (ingl k *source code management*) seadistuse tüübiks Git ning määratud on repositooriumi aadress. Kuna tegu on privaatse repositooriumiga, siis on autori poolt seadistatud kasutajakonto, mida Jenkins kasutab repositooriumisse sisselogimiseks. Lisaks on seadistatud kompileerimise sektsioonis Apache Ant versioon, mida kompileerimisel kasutada. Antud alapeatükis tehtavaid tegevusi illustreerivad kuvatõmmised asuvad **Lisa 7**.

3.4 Veebihaagi seadistamine

Et kompileerimine toimuks automaatselt peale koodi lisamist repositooriumisse on seadistatud selleks vajalik veebihaak (ingl k *webhook*). Veebihaak on teavitus mingist kindlast sündmusest [41]. Veebihaagi seadistamiseks on Jenkinsile paigaldatud GitHub pistikprogramm [42]. Et veebihaak toimiks, peab Jenkins olema ligipääsetav välisest võrgust. Selle jaoks tegi vajaliku nimelahenduse ning vaheserveri seadistused RIK-i süsteemiadministraator. Kasutades pääsuloendit (ingl k *access control list*) on ligipääs antud GitHub-i poolt kasutatavatele IP aadressidele [43]. Kuvatõmmised veebihaagi lingi seadistamisest GitHub-is ning konsooli väljundist peale kompileerimise projekti automaatset käivitumist asuvad vastavalt **Lisa 8** ja **Lisa 9**. Veebihaagi saatmist Jenkinsi installatsioonile on illustreeritud joonisel **Joonis 2**.



Joonis 2. Veebihaagi saatmine GitHub-ist Jenkinsi installatsioonile.

4 Ülevaade tulemustest

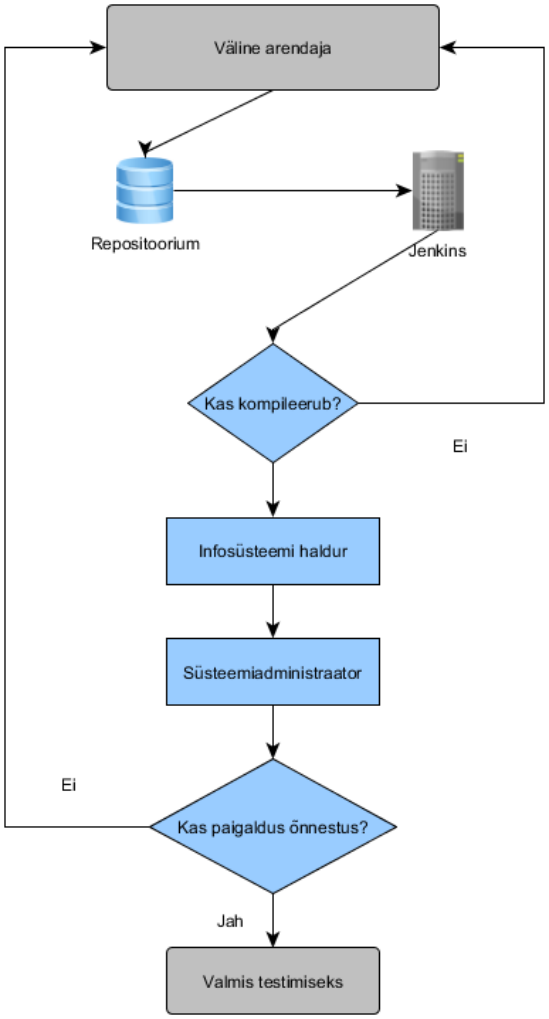
Järgnevalt kirjeldab autor MuIS-i uuendamise protsessi peale Jenkinsi kasutusele võtmist. Protsess eeldab, et välise arendaja poolt on üle antud koodimuudatus. Rollid protsessis ja nende tegevused on välja toodud tabelis **Tabel 6**.

Tabel 6. Rollid protsessis ja nende tegevused peale Jenkinsi kasutusele võtmist.

Roll	**Tegevus**
Infosüsteemi haldur	Koordineerib rakenduse arendusprotsessi ja tegeleb tarkvara testimisega
Süsteemiadministraator	Paigaldab uuenduse

Peale Jenkinsi kasutusele võtmist on üks uuendamise protsessi tegevus, milleks on lähtekoodi kompileerimine, automatiseeritud. Kui väline arendaja uuendab lähtekoodi repositooriumis, siis selle peale saadetakse teavitus töö praktilises osas seadistatud Jenkinsi installatsioonile. Kui teavitus jõuab kohale, siis käivitub Jenkinsis automaatselt autori poolt seadistatud projekt, mis laeb alla ning kompileerib uuendatud lähtekoodi. Sellest tulenevalt on kohe olemas paigalduspakk, mille haldur saab edastada süsteemiadministraatorile. Protsessist ülevaate saamiseks on loodud töövoog joonisel **Joonis 3**. Järgnevalt seletab autor lahti uuendamise protsessi sammudes tehtavad tegevused peale Jenkinsi kasutusele võtmist:

1. Infosüsteemi haldur edastab Jenkinsi poolt kompileeritud paigalduspaki süsteemiadministraatorile, kes uuendust testkeskkonda paigaldama hakkab. Kui uuenduse paigaldamise järgselt selgub, et rakendus ei hakka tööle, siis kogutakse kokku logifailid ning probleem suunatakse välisele arendajale lahendamiseks.
2. Kui uuendus on paigaldatud testkeskkonda, siis võib lugeda uuendamise protsessi lõppenuks ning haldur saab hakata testima kas tellitud muudatused vastavad nõuetele.



Joonis 3. Uuendamise protsess peale Jenkinsi kasutusele võtmist.

4.1 Võrdlus lähteolukorraga ja järeldused

Diplomitöö tulemusena ei ole enam vajadust oodata, et asutuse arendaja uuendatud lähtekoodi kompileeriks. Jenkinsi kasutusele võtmisega on see osa protsessist nüüd täielikult automatiseeritud. Võttes aluseks informatsiooni, mis on esitatud peatükis Olemasolev olukord, siis võib öelda, et diplomitöö tulemusena toimub Java rakenduste uuendamise protsess kiiremini ning sujuvamalt kui see toimus lähteolukorras. Sellest tulenevalt on lõputöö sissejuhatuses seatud eesmärk, milleks on rakenduste uuendamise protsessi optimeerimine, täidetud.

Nüüd, peale koodimuudatuse laekumist väliselt arendajalt, saavad infosüsteemide haldurid kiiremini tegeleda tarkvara testimisega. See aitab infosüsteemide haldurite tööd muuta sujuvamaks, mis omakorda tõstab RIK-i töötajate rahulolu. Lisaks toimub tänu

sellele Java rakenduste arendusprotsess kiiremini, mis on kasulik nii infosüsteemide omanikele kui ka RIK-ile.

Veel on diplomitöö tulemusena vähendatud asutuse arendajate töökoormust. Enam ei pea asutuse arendajad oma põhitöö kõrvalt teiste projektide kompileerimisega tegelema. See aitab kaasa RIK-i poolt arendatavate rakenduste kiirele ning kvaliteetsele arendusele.

5 Edasised tegevused

Edasisteks tegevusteks on autor planeerinud loodud lahenduse peenhäälestamise. Seadistada tuleks teavitused, mis saadetakse infosüsteemi haldurile ja välisele arendajale kui uuendatud koodi kompileerimisega on probleeme. Veel tuleks seadistada skriptid või leida mõni teine viis kuidas kompileeritud faile automaatselt kopeerida etteantud sihtkohtadesse. See teeks kompileeritud failidele ligipääsu veel mugavamaks.

Samuti plaanib autor seadistada Jenkinsit kasutades automaatse paigaldamise testkeskkonda. Tuleks uurida milliseid Jenkinsi pistikprogramme kõige kasulikum kasutada oleks ning milliseid automaatteste rakendada saaks. Selleks saab autor kasutada juba diplomitöö käigus loodud testkeskkonda. Eeltestkeskkonna mõte oleks leida viis protsessi edasiseks optimeerimiseks, mis ei tekitaks süsteemiadministraatoritele lisa töökoormust, vaid muudaks ka nende tööd mugavamaks.

6 Kokkuvõte

Käesoleva diplomitöö eesmärgiks oli optimeerida Java rakenduste uuendamise protsessi Registrite ja Infosüsteemide keskuses. Kuna diplomitöö autori tööülesandeks RIK-is on hallata Java rakendust, milleks on Muuseumide Infosüsteem, siis kasutati näitena MuIS-i töötajakeskkonna uuendamise protsessi.

Diplomitöös lahendatavaks probleemiks oli aeganõudev rakenduste uuendamise protsess, mis aeglustab rakenduste arendusprotsessi, demotiveerib infosüsteemide haldureid ning aeglustab nende tööd. Protsess sõltub peale halduri ka asutuse arendaja ning süsteemiadministraatori vaba aja olemasolust.

Probleemi lahendamiseks kirjeldas töö autor olemasolevat olukorda ning tõi näitena MuIS-i uuendamise peale kuluva aja. Seejärel analüüsis autor erinevaid viise uuendamise protsessis tehtavate sammude automatiseerimiseks. Taustinfona kirjeldas autor automatiseerimise olulisust ning tõi välja selle kasutegurid. Järgnevalt uuris autor võimalusi lähtekoodi kompileerimise automatiseerimiseks ning otsustas, et selleks sobib pidevintegratsiooni lahendus. Peale selle uuris töö autor ka rakenduste paigaldamise automatiseerimise võimalust. Selleks kirjeldas autor pidevvalmiduse lahendust ning viis läbi küsitluse RIK-i süsteemiadministraatorite seas, et teada saada kas paigaldamise protsessis on mõistlik midagi muuta. Toetudes läbiviidud küsitluse tulemustele otsustas töö autor, et paigaldamise automatiseerimist ei käsitleta. Järgnevalt kirjeldas autor viite erinevat pidevintegratsiooni lahendust, mille seast valis Jenkinsi.

Töö praktilises osas seadistas autor testkeskkonna ning paigaldas sinna Jenkinsi. Kasutades Jenkinsit seadistas töö autor lahenduse, millest tulenevalt toimub uuendatud lähtekoodi alla laadimine ning kompileerimine automaatselt peale seda kui väline arendaja on uuendatud koodi repositooriumisse lisanud.

Diplomitöö tulemusena toimub uuendamise protsess kiiremini ning sujuvamalt kui see toimus lähteolukorras. Enam ei ole vajadust oodata, et asutuse arendaja uuendatud

lähtekoodi kompileeriks. Sellest tulenevalt on lõputöö sissejuhatuses seatud eesmärk, milleks on rakenduste uuendamise protsessi optimeerimine, täidetud.

Kasutatud kirjandus

[1] „STANDARDIPÕHINE TARKVARATEHNIKA SÕNASTIK,“ Cybernetica AS, 2011-2017. [Võrgumaterjal]. URL: http://stats.cyber.ee. [Kasutatud 05 11 2017].

[2] „ANDMEKAITSE JA INFOTURBE LEKSIKON,“ Cybernetica AS, 2011 - 2017. [Võrgumaterjal]. URL: http://akit.cyber.ee/. [Kasutatud 05 11 2017].

[3] „Vabariigi Valitsuse riigireformi kava perioodiks jaanuar 2017 kuni märts 2019,“ Rahandusministeerium, [Võrgumaterjal]. URL: https://www.rahandusministeerium.ee/sites/default/files/riigireformi_tegevuskava_11_5_2017.pdf. [Kasutatud 22 11 2017].

[4] „Riigireform,“ Eesti Vabariigi Valitsus, [Võrgumaterjal]. URL: https://www.valitsus.ee/et/riigireform-tegevuskavas. [Kasutatud 22 11 2017].

[5] „IT-profiil,“ Registrite ja Infosüsteemide Keskus, 11 03 2013. [Võrgumaterjal]. URL: http://www.rik.ee/et/asutusest/it-profiil. [Kasutatud 29 12 2017].

[6] „Asutusest,“ Registrite ja Infosüsteemide Keskus, [Võrgumaterjal]. URL: http://www.rik.ee/et/asutusest. [Kasutatud 03 11 2017].

[7] „Muuseumide infosüsteem MuIS,“ Kultuuriministeerium, [Võrgumaterjal]. URL: http://www.kul.ee/et/tegevused/muuseumid/muuseumide-infosusteem-muis. [Kasutatud 04 01 2018].

[8] „Eesti Muuseumide Veebivärav,“ Kultuuriministeerium, [Võrgumaterjal]. URL: https://www.muis.ee/. [Kasutatud 04 01 2018].

[9] D. J. S. Carroll, „How Java Works,“ [Võrgumaterjal]. URL: http://www.cs.cmu.edu/~jcarroll/15-100-s05/supps/basics/history.html. [Kasutatud 05 01 2018].

[10] „5 Ways Automation Benefits IT Operations,“ [Võrgumaterjal]. URL: https://ayehu.com/5-ways-automation-benefits-operations/. [Kasutatud 12 12 2017].

[11] T. Kitchens, „Automating Software Development Processes,“ [Võrgumaterjal]. URL: https://www.developerdotstar.com/mag/articles/automate_software_process.html. [Kasutatud 12 12 2017].

[12] „Human error is the biggest cause of IT disasters: survey,“ Continuity Central, [Võrgumaterjal]. URL: http://www.continuitycentral.com/news06032.html. [Kasutatud 04 01 2018].

[13] „Using Apache Ant,“ [Võrgumaterjal]. URL: https://ant.apache.org/manual/using.html. [Kasutatud 09 01 2018].

[14] „Chapter 16. Build Script Basics,“ Gradle Inc., 2018. [Võrgumaterjal]. URL: https://docs.gradle.org/current/userguide/tutorial_using_tasks.html. [Kasutatud 09 01 2018].

[15] „POM Reference,“ [Võrgumaterjal]. URL: https://maven.apache.org/pom.html#What_is_the_POM. [Kasutatud 09 01 2018].

[16] J. Enos, „Automated Builds: The Key to Consistency,“ [Võrgumaterjal]. URL: https://www.infoq.com/articles/Automated-Builds. [Kasutatud 28 12 2017].

[17] „Apache Ant™,“ The Apache Software Foundation, 1999-2017. [Võrgumaterjal]. URL: http://ant.apache.org/. [Kasutatud 26 12 2017].

[18] „Continuous integration vs. continuous delivery vs. continuous deployment,“ Atlassian, [Võrgumaterjal]. URL: https://www.atlassian.com/continuous-delivery/ci-vs-ci-vs-cd. [Kasutatud 07 11 2017].

[19] „Continuous Integration,“ [Võrgumaterjal]. URL: https://www.jetbrains.com/teamcity/features/continuous_integration.html. [Kasutatud 28 12 2017].

[20] „Managing Notifications,“ [Võrgumaterjal]. URL: https://documentation.codeship.com/general/account/notifications/. [Kasutatud 28 12 2017].

[21] „Configuring Build Notifications,“ [Võrgumaterjal]. URL: https://docs.travis-ci.com/user/notifications/. [Kasutatud 28 12 2017].

[22] „What is Continuous Delivery?,“ [Võrgumaterjal]. URL: https://continuousdelivery.com/. [Kasutatud 15 12 2017].

[23] „Top 8 Continuous Integration Tools,“ [Võrgumaterjal]. URL: https://dzone.com/articles/top-8-continuous-integration-tools. [Kasutatud 05 11 2017].

[24] „Top Continuous Integration Tools: 51 Tools to Streamline Your Development Process, Boost Quality, and Enhance Accuracy,“ [Võrgumaterjal]. URL: https://stackify.com/top-continuous-integration-tools/. [Kasutatud 05 11 2017].

[25] „Continuous Integration Tools – 10 Leading Solutions,“ [Võrgumaterjal]. URL: https://www.keycdn.com/blog/continuous-integration-tools/. [Kasutatud 18 11 2017].

[26] „About Jenkins,“ [Võrgumaterjal]. URL: https://www.cloudbees.com/jenkins/about. [Kasutatud 04 11 2017].

[27] „Jenkins,“ [Võrgumaterjal]. URL: https://jenkins-ci.org/. [Kasutatud 30 12 2017].

[28] „Build, test, deploy,“ Atlassian, [Võrgumaterjal]. URL: https://www.atlassian.com/software/bamboo. [Kasutatud 12 11 2017].

[29] „TeamCity,“ JetBrains, [Võrgumaterjal]. URL: https://www.jetbrains.com/teamcity/. [Kasutatud 12 11 2017].

[30] „Test and Deploy with Confidence,“ [Võrgumaterjal]. URL: https://travis-ci.org/. [Kasutatud 13 11 2017].

[31] „SIMPLIFY CONTINUOUS DELIVERY,“ ThoughtWorks Inc, [Võrgumaterjal]. URL: https://www.gocd.org/. [Kasutatud 15 12 2017].

[32] „Jenkins vs. Other Open Source Continuous Integration Servers,“ [Võrgumaterjal]. URL: https://www.blazemeter.com/blog/jenkins-vs-other-open-source-continuous-integration-servers. [Kasutatud 29 12 2017].

[33] J. Giménez, „What Is Jenkins and Why Should You Be Using It?,“ Bugfender, 21 11 2017. [Võrgumaterjal]. URL: https://bugfender.com/blog/what-is-jenkins-and-why-should-you-be-using-it/. [Kasutatud 06 01 2018].

[34] „CentOS Linux,“ The CentOS Project, 2017. [Võrgumaterjal]. URL: https://www.centos.org/about/. [Kasutatud 29 12 2017].

[35] „Apache Tomcat,“ [Võrgumaterjal]. URL: http://kuutorvaja.eenet.ee/wiki/Apache_Tomcat. [Kasutatud 29 12 2017].

[36] „Proxy Server,“ [Võrgumaterjal]. URL: http://www.proxyserverprivacy.com/proxy-server.shtml. [Kasutatud 29 12 2017].

[37] „Git Plugin,“ [Võrgumaterjal]. URL: https://wiki.jenkins.io/display/JENKINS/Git+Plugin. [Kasutatud 29 12 2017].

[38] „GIT,“ [Võrgumaterjal]. URL: http://kuutorvaja.eenet.ee/wiki/GIT. [Kasutatud 29 12 2017].

[39] „Git,“ [Võrgumaterjal]. URL: https://git-scm.com/. [Kasutatud 29 12 2017].

[40] „Ant Plugin,“ [Võrgumaterjal]. URL: https://wiki.jenkins.io/display/JENKINS/Ant+Plugin. [Kasutatud 29 12 2017].

[41] „What is a WebHook?,“ [Võrgumaterjal]. URL: https://webhooks.pbworks.com/w/page/13385124/FrontPage. [Kasutatud 01 01 2018].

[42] „GitHub Plugin,“ [Võrgumaterjal]. URL: https://wiki.jenkins.io/display/JENKINS/GitHub+Plugin. [Kasutatud 01 01 2018].

[43] „About GitHub's IP addresses,“ GitHub Inc, 2017. [Võrgumaterjal]. URL: https://help.github.com/articles/about-github-s-ip-addresses/. [Kasutatud 02 01 2018].

Lisa 1 – Üleandmine ja vastuvõtmine

Saatja: Robin Lõo
Saatmisaeg: esmaspäev, 13. november 2017 10:44
Adressaat: Frederick Liin
Teema: Vastuvõtmise testimine

Järeltegevuse lipp: Järeltegevus
Olekulipp: Lipuga märgitud

1. Üleandmine ja vastuvõtmine

1.1. Töö või Töö etapi valmimise järgselt annab Täitja selle Tellijale üle vastuvõtmiseks.

1.2. Täitja peab Tellijat Töö või Töö etapi üle andmise viivitusest või viivitusse sattumise ohust ja põhjustest koheselt kirjalikku taasesitamist võimaldavas vormis informeerima. Kui Täitja viivituse põhjustab Tellija, on Täitjal õigus nõuda mõistlikku ajapikendust ja põhjendatud lisakulude hüvitamist.

1.3. Töö või Töö etapi üleandmise kohta koostab Täitja üleandmise akti, milles näidatakse muuhulgas ära üleandmise kuupäev, teostatud Töö, osutatud teenuste, hoolduse ja tarnitud seadmete, telekommunikatsioonide ja tarkvara detailiseeritud nimekiri ning vajaduse korral neis esinevad puudused. Iga Töö etapi üleandmisel koostab ja esitab Täitja antud etapi kohta koostatud dokumentatsiooni, vastavalt dokumentatsiooniplaanile või Tellija poolt tehnilises kirjelduses esitatud nõudmistele.

1.4. Töö või Töö etapi üleandmine Tellijale ei ole käsitatav selle vastuvõtmisena Tellija poolt.

1.5. Pärast Töö etapi üleandmist on Tellijal õigus Töö üle vaadata 10 tööpäeva jooksul. Pärast Töö üleandmist on Tellijal õigus Töö üle vaadata 20 tööpäeva jooksul.

1.6. Juhul, kui Tellija leiab, et Töö ei vasta Lepingu tingimustele, on Tellija kohustatud teavitama Täitjat Töös avastatud puudustest, keeldumisest Tööd enne puuduste kõrvaldamist vastu võtta ja kirjeldama Töö puudused. Täitja on kohustatud puudused kõrvaldama 5 tööpäeva jooksul, kui pooled ei ole kokku leppinud mõnda muud tähtaega. Töö puuduste kõrvaldamise kulud kannab Täitja.

1.7. Enne töö üleandmist viib Täitja Töö nõuetelevastavuse kindlakstegemiseks läbi testid.

1.8. Täitja esitab Tellijale kõik tema poolt läbiviidud testide tulemusena valminud dokumentide koopiad.

1.9. Enne töö vastuvõtmist viib Tellija Töö nõuetelevastavuse kindlakstegemiseks läbi testid.

1.10. Kui Töö või mistahes Töö osa ei läbi teste, viiakse otsekohe pärast seda, kui Täitja on teinud vajalikud korrektuurid testide edukaks läbiviimiseks, läbi kordustestid samadel tingimustel.

1.11. Tellija nõudmisel viib kordustestid läbi Täitja.

1.12. Parandatud Töö üleandmine toimub nagu esmakordsel üleandmisel. Tellijal on õigus parandatud Töö üle vaadata 10 tööpäeva jooksul. Parandatud Töös puuduste ilmnemisel on Täitja kohustatud nimetatud kõrvaldama punktis 11.6 toodud tingimustel.

1.13. Puudustega üle antud Tööd ei loeta tähtaegselt üleantuks ning Tellijal on õigus nõuda sellise lepingurikkumise korral Täitjalt leppetrahvi Lepingus sätestatud korras ja määrades või rakendada muid õiguskaitsevahendeid.

1.14. Vastuvõtmiseks valmis Töö peab vastama Lepingus sätestatud tingimustele. Töö etapi vastuvõtmine Tellija poolt ei tingi iseenesest ega kohusta Töö kui terviku vastuvõtmist Tellija poolt juhul, kui Töö ei vasta tingimustele.

1.15. Töö vastuvõtmise kohta koostab Tellija vastuvõtuakti, milles näidatakse muuhulgas ära vastuvõtmise kuupäev, teostatud Töö, osutatud teenuste, hoolduse ja tarnitud seadmete, telekommunikatsioonide ja tarkvara detailiseeritud nimekiri.

1.16. Töö või Töö etapp loetakse vastuvõetuks vastuvõtuakti allkirjastamisest või toodangukeskkonnas kasutusele võtmisest. Juhul, kui Tellija on võtnud puuduseid sisaldava Töö või Töö etapi toodangukeskkonnas kasutusele, loetakse vastuvõetuks ainult nõuetekohaselt teostatud tööd ja Täitjal on õigus nimetatute osas esitada arve. Taolises olukorras esitab Tellija Täitjale Töös või Töö etapis esinevate puuduste nimekirja ning puuduste kõrvaldamise tähtaja, kas enne Töö või Töö etapi toodangukeskkonnas kasutusele võtmist või vahetult pärast selle toimumist.

1.17. Mistahes seadmete, telekommunikatsioonide, tarkvara ja teenuste osa valmimisel võib Tellija anda välja vastuvõtuakti vastava osa kohta. Selline vastuvõtmine ei mõjuta Täitja kohustust täita kõiki Lepingus ettenähtud kohustusi.

1.18. Täitja vastutab Töö juhusliku hävimise või kahjustumise eest kuni Töö vastuvõtmiseni Tellija poolt.

Lisa 2 – Garantiitingimused

Saatja: Robin Lõo
Saatmisaeg: esmaspäev, 13. november 2017 10:30
Adressaat: Frederick Liin
Teema: Standard garantii tingimused

Järeltegevuse lipp: Järeltegevus
Olekulipp: Lipuga märgitud

1. Garantii

1.1. Täitja annab Tööle 12 (kaheteist) kuulise töövõtugarantii. Garantiiperiood algab Tööde kogumina vastuvõtmisest.

1.2. Garantiiperioodil ilmnevad puudused kõrvaldab Täitja omal kulul, v.a punktis 1.6 toodud juhtumitel. Kui ilmnenud puudused ei ole garantii korras kõrvaldatavad, esitab Täitja Tellijale põhjendused kirjalikku taasesitamist võimaldavas vormis, kuid mitte hiljem kui järgmisel tööpäeval pärast sellest asjaolust teada saamist.

1.3. Kui ei ole kokku lepitud teisiti, kõrvaldab Täitja puudused 10 tööpäeva jooksul puudusest teada saamisest.

1.4. Juhul kui garantiiperioodil ilmnenud puudused muudavad osa või kõik seadmed, telekommunikatsioonid ja/või tarkvara kasutamiskõlbmatuks, varustab Täitja Tellijat nõutud tasemel toimimise garanteerivate lisa- või asendusosade ja muu vajalikuga omal kulul.

1.5. Tellija informeerib Täitjat puuduse iseloomust ja ulatusest otsekohe selle ilmnemisel. Täitja on kohustatud eemaldama ilmnenud puudused vastavalt lepingus toodule või Tellija poolt määratud ajal.

1.6. Garantiiga ei ole hõlmatud:

1.6.1. puudused, mille tekkimise eest vastutab Tellija;

1.6.2. diagnostikaks kulunud aeg, juhul kui algselt garantii juhtumine registreeritud juhtumi raames tuvastatakse, et tegu ei ole garantiilise juhtumiga. Vastav töö kuulub eraldi tasustamisele Lepingu tunnihinna alusel.

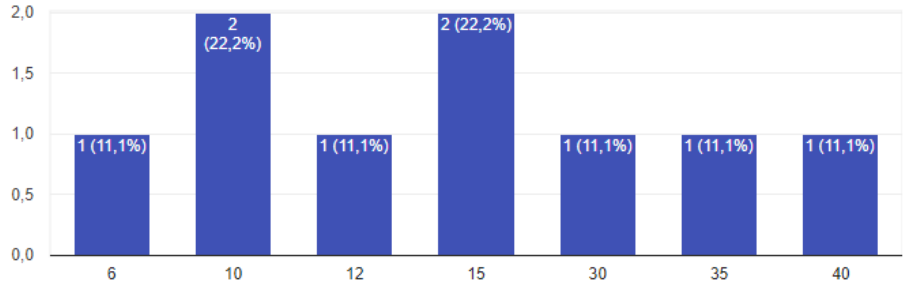
Garantii kaotab kehtivuse kui Täitjaga kooskõlastamata on muudetud või muudetakse lähtekoodi, v.a juhul kui Tellija suudab eristada lähtekoodis tehtavaid muudatusi.

Lisa 3 – RIK-i süsteemiadministraatorite seas läbi viidud küsitlus

- **Küsimus nr 1**

Mitut erinevat infosüsteemi/rakendust administreerid?

9 vastust



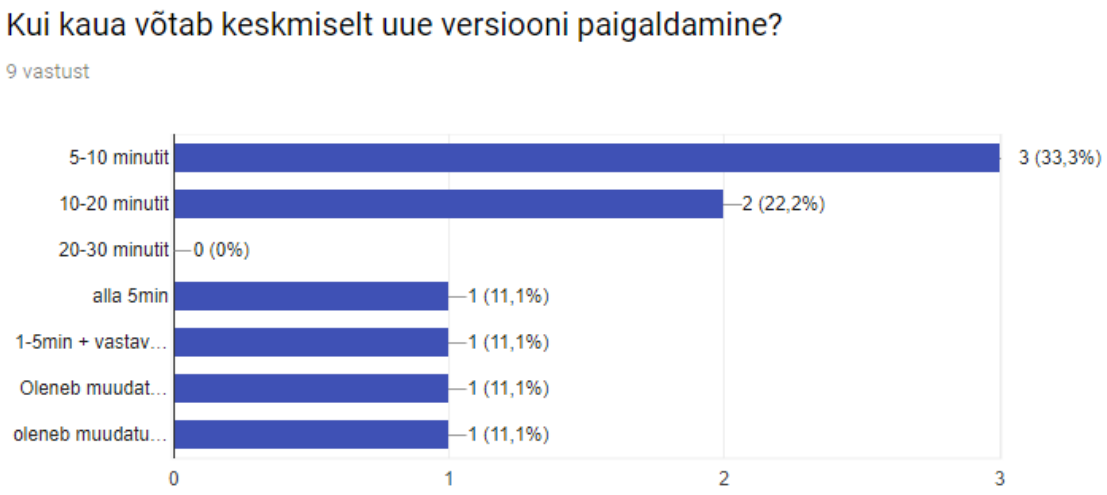
- **Küsimus nr 2**

Mis tüüpi rakendustega on tegu (java, python, php)?

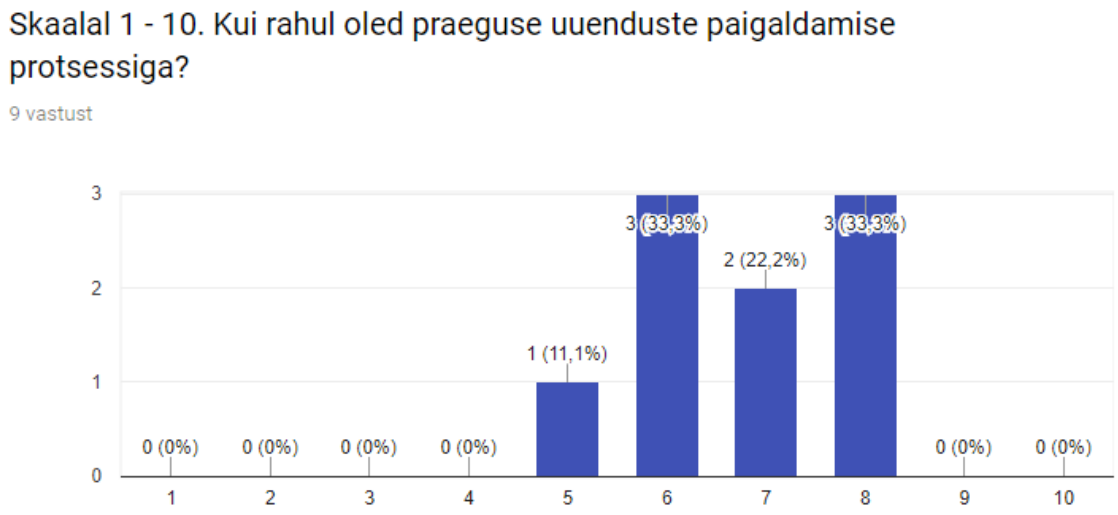
9 vastust

.net
iis, .net, Exchange
net, php, java, c
python,java
Java, python, LAMP
java, php
.net, python, php
java, python, c
.net, java, c

- **Küsimus nr 3**



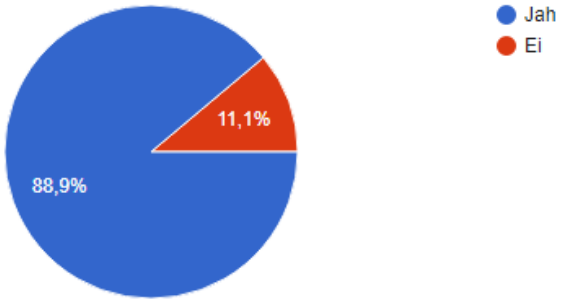
- **Küsimus nr 4**



- **Küsimus nr 5**

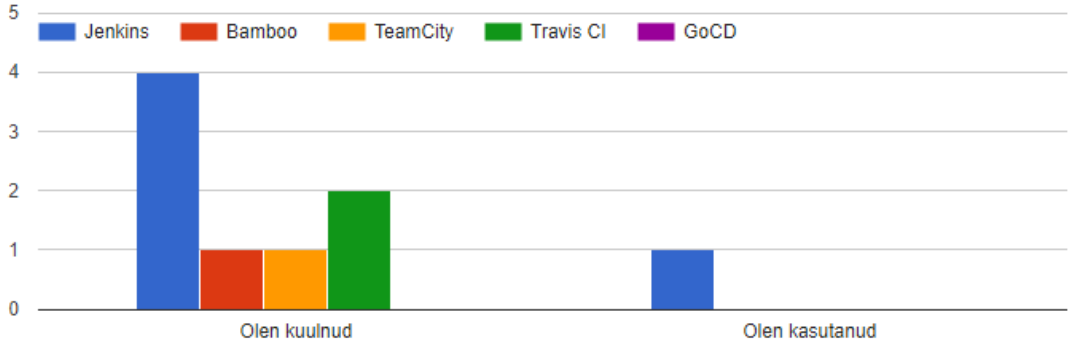
  Kas oled kuulnud pidevintegratsioonist ja -valmidusest? (Continuous integration and delivery)

  9 vastust



- **Küsimus nr 6**

  Kui vastasid eelmisele küsimusele JAH, siis märgi kas oled kuulnud nendest või kasutanud järgmisi pidevintegratsiooni ja -valmidust võimaldavaid tööriistu?



- **Küsimus nr 7**

Kas oleksid huvitatud sellise lahenduse kasutamisest RIK-is? Kui JAH, siis kirjuta millist eelistaksid (ei pea olema eelmises küsimuses loetletud lahendused). Kui EI, siis palun põhjenda lühidalt.

8 vastust

Ei, kuna ei tea millest täpselt jutt.
Ei, puuduv info.
ei ole vajadust. oma ajsad on kõik suht singleclick scripted, lisariskijubina vahele lisamine peale üüratu käimajooksmine lisab ka ohufaktori, et see keerab käru ja inimesed peavad veelrohkem kabetama
JAH, Jenkins
Ei, sest ei näe hetkel vajadust.
Ei, kuna ei tea millega tegu.
Ei, kuna enamuste korduvate tegevuste jaoks on skriptid juba olemas.
Võibolla jah, kui teaksin asjast rohkem. Hetkel arvan, et ei ole vajadust.

Lisa 4 – Java ning Apache Tomcati paigaldamine

Java paigaldamine ja kontroll:

```
yum install java-1.8.0-openjdk.x86_64
java -version
```

Apache Tomcatile eraldi kasutaja ning grupi loomine:

```
sudo groupadd tomcat
sudo mkdir /opt/tomcat
sudo useradd -s /bin/nologin -g tomcat -d /opt/tomcat tomcat
```

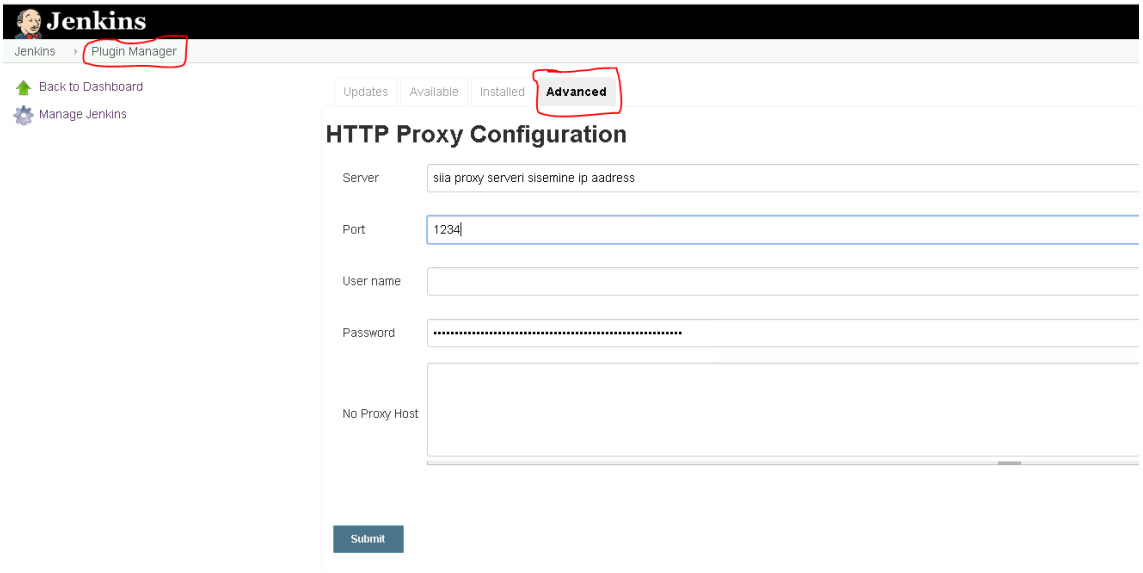
Apache Tomcati alla laadimine ning lahti pakkimine:

```
cd ~
wget http://www-us.apache.org/dist/tomcat/tomcat-8/v8.0.33/bin/apache-tomcat-
8.0.33.tar.gz
sudo tar -zxvf apache-tomcat-8.0.33.tar.gz -C /opt/tomcat --strip-
components=1
```

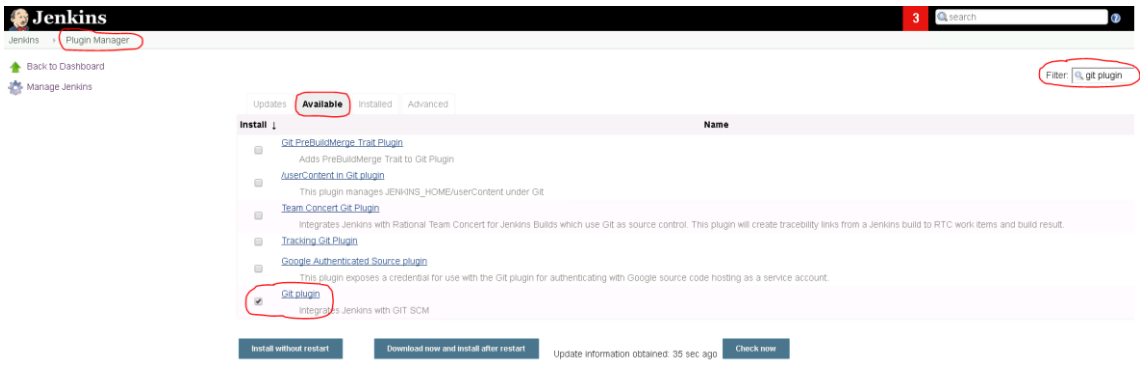
Apache Tomcati kataloogidele õiguste määramine:

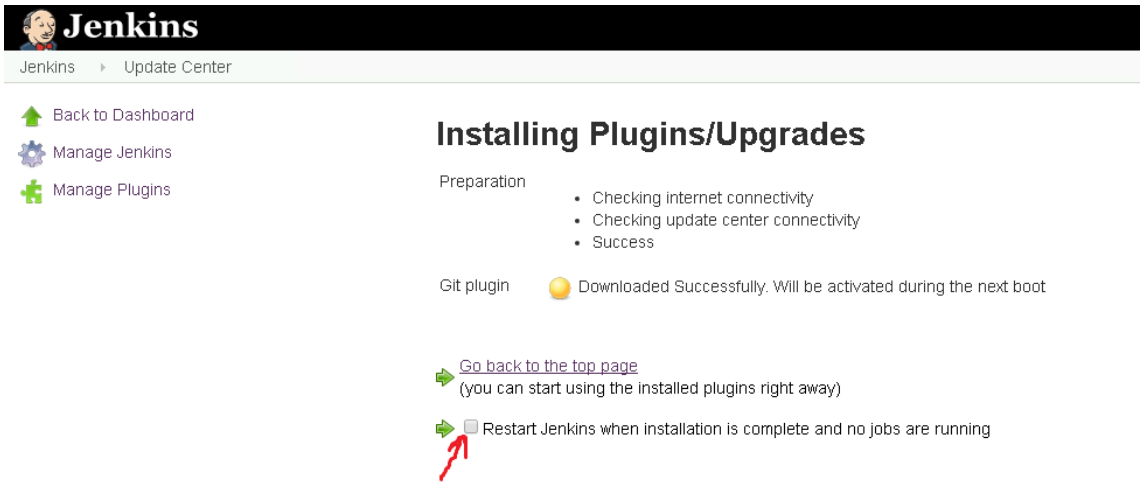
```
cd /opt/tomcat
sudo chgrp -R tomcat conf
sudo chmod g+rwx conf
sudo chmod g+r conf/*
sudo chown -R tomcat logs/ temp/ webapps/ work/

sudo chgrp -R tomcat bin
sudo chgrp -R tomcat lib
sudo chmod g+rwx bin
sudo chmod g+r bin/*
```

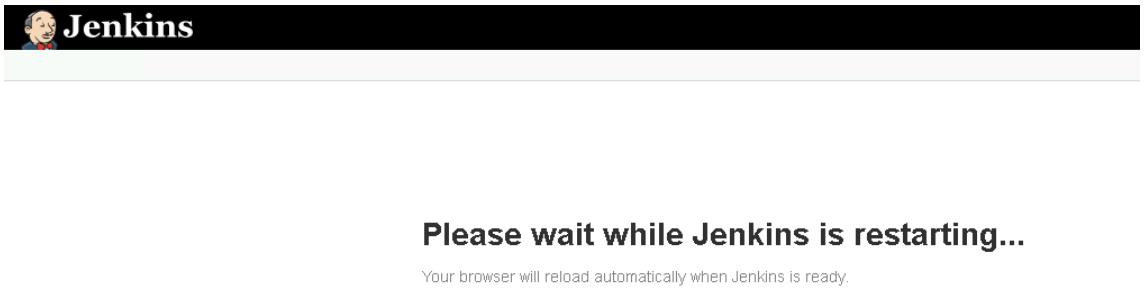
Lisa 5 – Vaheserveri sätted Jenkinsis



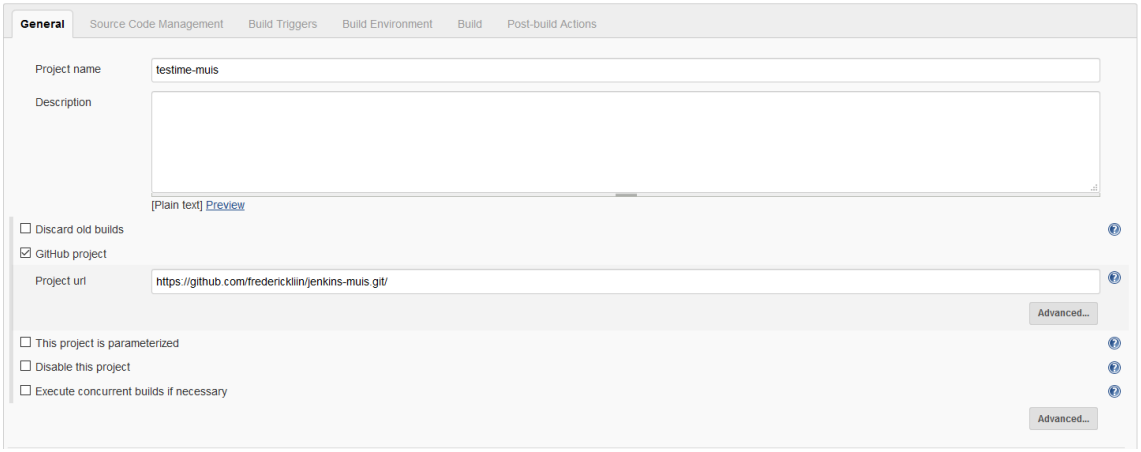
Lisa 6 – Pistikprogrammide paigaldamine Jenkinsis

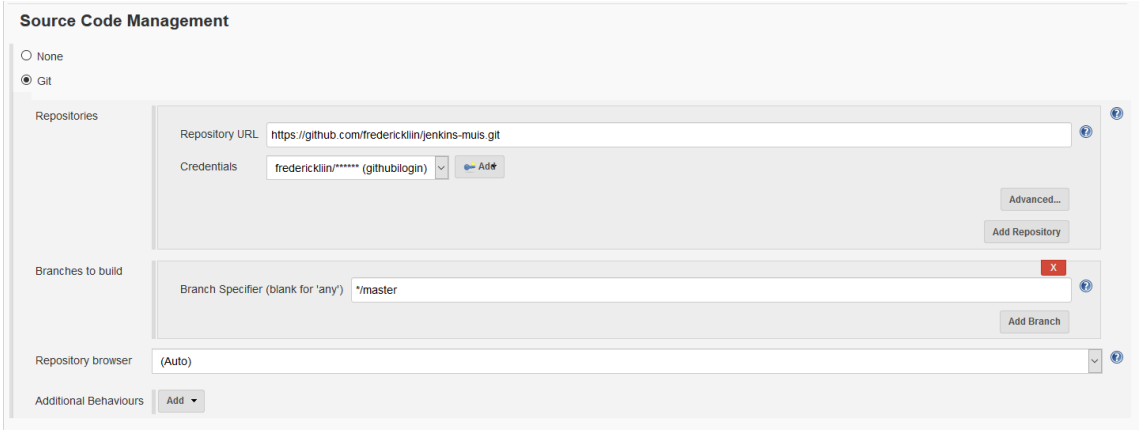


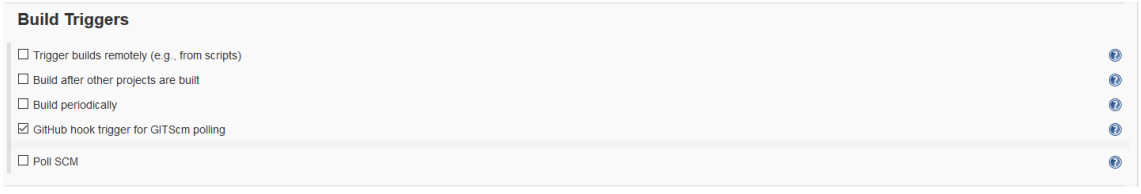




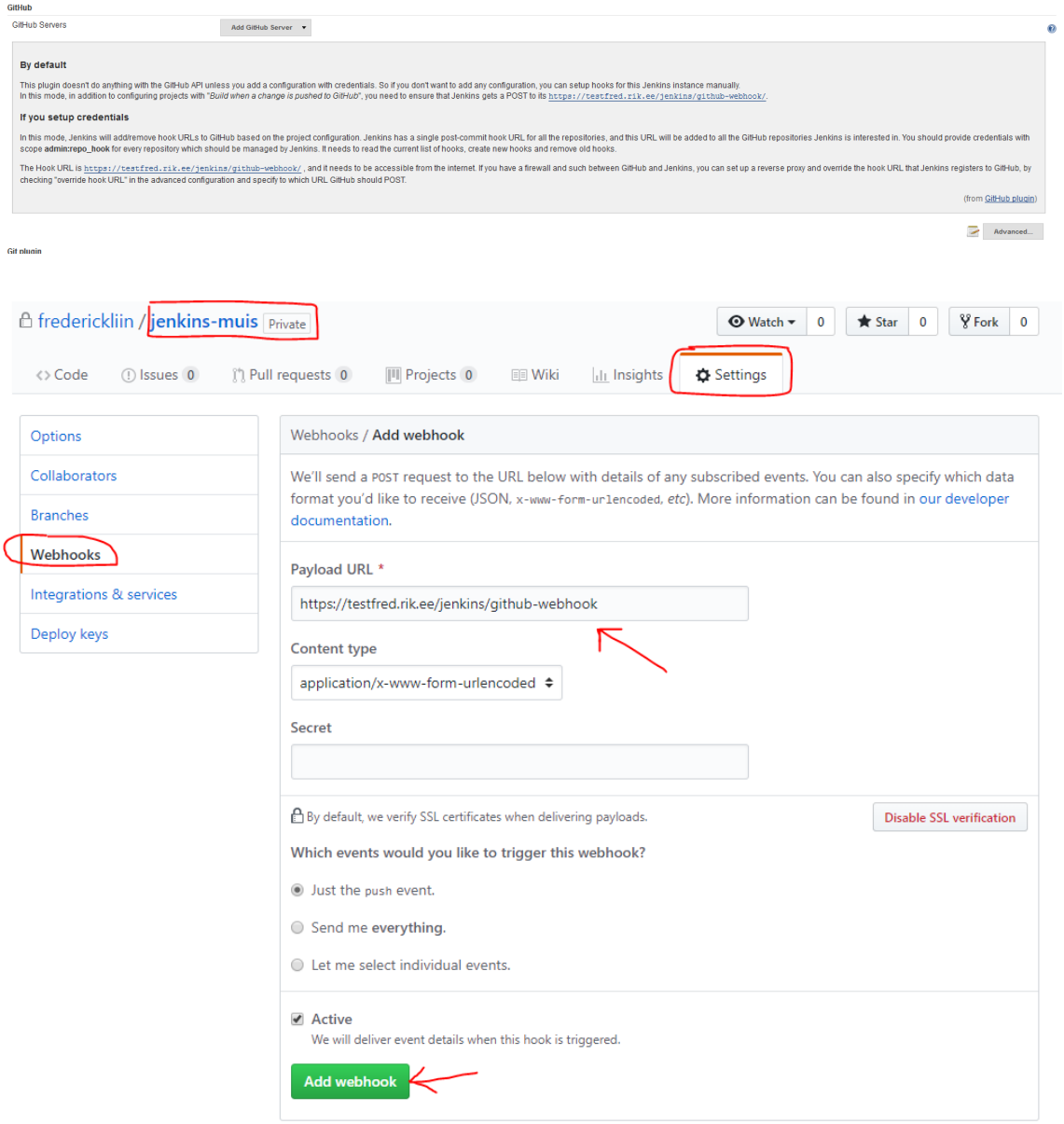
Lisa 7 – Uue projekti loomine Jenkinsis







Lisa 8 – Veebihaagi seadistus GitHubis



Lisa 9 – Konsooli väljund peale projekti automaatset käivitumist

Console Output

```
Started by GitHub push by frederickliin
Building in workspace /opt/tomcat/.jenkins/workspace/testime-muis
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/frederickliin/jenkins-muis.git # timeout=10
Fetching upstream changes from https://github.com/frederickliin/jenkins-muis.git
 > git --version # timeout=10
using GIT_ASKPASS to set credentials githubilogin
Setting http proxy: cache.just.sise:3128
 > git fetch --tags --progress https://github.com/frederickliin/jenkins-muis.git +refs/heads/*:refs/remotes/origin/*
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 4badd368fb1b963d20ceee6ad8aa894a13c70e2b (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 4badd368fb1b963d20ceee6ad8aa894a13c70e2b
Commit message: "Create 4"
 > git rev-list --no-walk 1abaaaf18dbc1d16d7ef6b2db691c7e83ef9fffa # timeout=10
[testime-muis] $ /opt/tomcat/.jenkins/tools/hudson.tasks.Ant_AntInstallation/ant/bin/ant -file muis
Buildfile: /opt/tomcat/.jenkins/workspace/testime-muis/muis/build.xml

clean:
   [delete] Deleting directory /opt/tomcat/.jenkins/workspace/testime-muis/muis/exploded
   [delete] Deleting: /opt/tomcat/.jenkins/workspace/testime-muis/muis/muis.war

copy:
    [mkdir] Created dir: /opt/tomcat/.jenkins/workspace/testime-muis/muis/exploded
    [mkdir] Created dir: /opt/tomcat/.jenkins/workspace/testime-muis/muis/exploded/WEB-INF/classes
    [mkdir] Created dir: /opt/tomcat/.jenkins/workspace/testime-muis/muis/exploded/WEB-INF/lib
     [copy] Copying 1125 files to /opt/tomcat/.jenkins/workspace/testime-muis/muis/exploded
     [copy] Copying 77 files to /opt/tomcat/.jenkins/workspace/testime-muis/muis/exploded/WEB-INF/lib
     [copy] Copying 5 files to /opt/tomcat/.jenkins/workspace/testime-muis/muis/exploded/WEB-INF/classes
     [copy] Copying 1 file to /opt/tomcat/.jenkins/workspace/testime-muis/muis/exploded/WEB-INF/classes
     [copy] Copying 1 file to /opt/tomcat/.jenkins/workspace/testime-muis/muis/exploded/WEB-INF/classes
     [copy] Copying 7 files to /opt/tomcat/.jenkins/workspace/testime-muis/muis/exploded/WEB-INF/classes/ee/medisoft/muis/hibernate/hbm
     [copy] Copying 5 files to /opt/tomcat/.jenkins/workspace/testime-muis/muis/exploded/WEB-INF/classes/ee/medisoft/muis/hibernate/entity

compile:
    [javac] Compiling 1330 source files to /opt/tomcat/.jenkins/workspace/testime-muis/muis/exploded/WEB-INF/classes
    [javac] warning: [options] bootstrap class path not set in conjunction with -source 1.7
    [javac] Note: Some input files use or override a deprecated API.
    [javac] Note: Recompile with -Xlint:deprecation for details.
    [javac] Note: Some input files use unchecked or unsafe operations.
    [javac] Note: Recompile with -Xlint:unchecked for details.
    [javac] 1 warning
    [javac] Creating empty /opt/tomcat/.jenkins/workspace/testime-muis/muis/exploded/WEB-INF/classes/ee/kul/muis/version/package-info.class

war:
      [war] Building war: /opt/tomcat/.jenkins/workspace/testime-muis/muis/muis.war

BUILD SUCCESSFUL
Total time: 18 seconds
Finished: SUCCESS
```