TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Mikk Õunmaa

Automaattestide loomine sessioonõppe ainele „Sissejuhatus andmebaasidesse“

Bakalaureusetöö (9 EAP)

Juhendaja: Piret Luik, PhD

Tartu 2019

Automaattestide loomine sessioonõppe ainele „Sissejuhatus andmebaasidesse“

Lühikokkuvõte:

Bakalaureusetöö eesmärk oli luua automaattestid Tartu Ülikooli sessioonõppe ainele „Sissejuhatus andmebaasidesse“. Töö tulemusena valmisid testid, mida on võimalik panna Moodle’isse ja testimisvahend, mis lihtsustab õppejõul tudengite andmebaasi kontrollimist. Esimeses peatükis tutvustatakse erinevaid veebis kasutatavaid iseseisvate tööde kontrollimise meetodeid ning kuidas on individuaalseid töid kontrollitud Tartu Ülikoolis. Teises peatükis kirjeldatakse ainet „Sissejuhatus andmebaasidesse“ ja selles tehtavaid iseseisvaid töid. Viimases peatükis selgitatakse, kuidas valmisid testid ja testimisvahend, kirjeldatakse esinenud probleeme ning tuuakse välja testide ja kontrollimisvahendi kasutamise eripärad. Samuti võrreldakse valminud programmi varasemate Tartu Ülikoolis valminud andmebaasi testimisvahenditega ja tuuakse välja edasi arendamise võimalused.

Võtmesõnad:

Andmebaasid, Java, automaattestid, MOOC

CERCS: P175 Informaatika, süsteemiteooria, S270 Pedagoogika ja didaktika

Creating automated tests for session-based course “Sissejuhatus andmebaasidesse”

Abstract:

The aim of this bachelor’s thesis was to create automated tests for session-based course “Sissejuhatus andmebaasidesse”, that is taught in the University of Tartu. As a result, tests, that can be configured in to the Moodle, and testing tool for tutors, were created. The first chapter describes different methods that are used to grade individual tasks in the web-based courses and how individual tasks have been checked in the University of Tartu. The second chapter introduces the course “Sissejuhatus andmebaasidesse” and explains the tasks, for which the tests were created. The final chapter gives insight to the implementation of automated tests and the testing tool, points out the problems during the creation process and teaches how to use the program and tests. In additon, compares the created program with database testing tools, that were created beforehand in the University of Tartu, and gives suggestions what to improve in the future.

Keywords:

Database, Java, automated testing, MOOC, massive open online courses

CERCS: P175 Informatics, systems theory, S270 Pedagogy and didactics

Sisukord

Sissejuhatus

Tartu Ülikoolis on informaatika erialale vastuvõetute arv viimastel aastatel suurenenud [1]. Õppekavast moodustavad suure osa kursused, kus tuleb programmeerida [2]. Kuna programmeerimise aineid on efektiivsem õppida lahendades praktilisi ülesandeid [3], siis on vaja, et õppejõud mõtleks välja ülesanded ja ka kontrolliks nende lahendusi. Koostatud ülesandeid on võimalik erinevatel õppeaastatel taaskasutada, kuid kontrollima peab iga aasta uusi töid, mis tähendab suurt koormust õppejõududele. Tööde automatiseeritud kontrollimine kiirendab lahenduste tagasisidestamist, kuid kõigi ainete puhul ei ole seda võimalust Tartu Ülikoolis kasutatud. Sellised on näiteks ained, milles käsitletakse andmebaaside kasutamist.

Tartu Ülikoolis on bakalaureusetöö raames varem valminud kaks programmi, mis keskenduvad „Andmebaasid“ aines kasutatavate andmebaaside testimisele [4, 5], kuid kumbki neist pole tööde kontrollimisel kasutust leidnud. Seetõttu kontrollitakse tudengite individuaalseid töid endiselt manuaalselt.

Käesoleva bakalaureusetöö eesmärk on luua automaattestid sessioonõppe ainele „Sissejuhatus andmebaasidesse“. Tulemusena valmivad testid, mida on võimalik panna Moodle'i keskkonda, et tudengid saaksid kiiremini oma iseseisvatele töödele tagasisidet. Samuti valmib testimisvahend, mis vähendab õppejõul andmebaasi kontrollimisele kuluvat aega.

Esimeses peatükis tutvustatakse meetodeid, mida on kasutatud veebipõhiste tööde kontrollimisel vaba juurdepääsuga e-kursuste (MOOC) näidetel. Samuti tutvustatakse Tartu Ülikoolis automaattestide kasutamist ja andmebaaside testimisega seotuid varasemaid Tartu Ülikooli bakalaureusetöid. Teises peatükis tutvustatakse ainet „Sissejuhatus andmebaasidesse“ ning aines tehtavaid iseseisvaid töid ja tuuakse välja, millistele individuaalsetele töödele tehakse automaattestid. Kolmandas peatükis kirjeldatakse bakalaureusetöö raames valminud programmi ning automaatteste ja nende tegemisel esinenud probleemid. Tuuakse välja, kuidas igat iseseisvat tööd kontrollitakse ja mis tingimused peavad kasutamiseks olema täidetud. Samuti võrreldakse valminud testimisvahendit Tartu Ülikoolis varem valminud andmebaaside testimisvahenditega ning esitatakse võimalusi, kuidas valminud programmi ja teste edasi arendada.

1 Veebipõhiste kursuste iseseisvate tööde hindamine

Ülikoolid pakuvad Internetis võimalust inimestel osaleda vaba juurdepääsuga e-kursustel. Tavaliselt MOOC-ide käigus ei toimu õppejõu ja tudengite kokkusaamisi, seetõttu kasutatakse eelkõige nendel kursustel veebipõhist hindamist, mida järgnevalt käsitletakse. Kontrollimaks, kas tudengid on nõutud materjali selgeks õppinud, antakse iseseisvaid töid. Kuna õppurite arv võib ulatuda tuhandetesse [6], siis tähendab, et on vaja kontrollida ka tuhandeid töid, mis tähendab suurt koormust õppejõududele. Seetõttu on MOOC-ides õppurite tööde kontrollimiseks kasutusel mitmeid meetodeid. Mõned viisid kursusel osalejate individuaalsete ülesannete hindamisel on lasta tudengil enda tööd või kaaslaste töid hinnata või töö tulemuse määrab arvuti [7].

1.1 Enda ja kaaslase töö hindamine

Enda või kaaslase töö hindamine on MOOC-ides kasutusel kirjalike ülesannete, nagu esseede, kontrollimiseks [7]. Ülesande lahendaja peab läbi lugema töökäsu ja sellele kirjutama vastuse. Kui see on valmis, siis antakse ette hindamiskriteeriumid (vt joonis 1.1), mille alusel punkte anda, ja töö, mida hinnata. Leidub uurimusi, mis väidavad, et oma töö hindamine pole adekvaatne meetod õppuri teadmisi kontrollida, sest õppur hindab oma tööd leebemalt kui kursusekaaslased [7].

	5 (Excellent)	4 (Very good)	3 (Good)	2 (Fair)	1 (Poor)
Writing	Without mistakes in spelling, punctuation, and grammar	Very few mistakes in spelling, punctuation, and grammar	Few mistakes in spelling, punctuation, and grammar	Some mistakes in spelling, punctuation, and grammar	Spelling, punctuation or grammar errors seriously hinder comprehensibility
Format and organization	The logical structure of the argument is totally clear. Main arguments, objections, and replies are very clearly indicated.	The logical structure of the argument is very clear. Main arguments, objections, and replies are very clearly indicated.	The logical structure of the argument is clear. Main arguments, objections, and replies clearly indicated.	The structure of argument is not clear, or main arguments, objections, and replies not clearly indicated.	The structure of argument cannot be made out, or it is missing completely.
Language and bibliographic	It uses a language that is totally appropriate, without errors in citing materials, either in in-text citations or reference list.	It uses a language that is quite appropriate, with minor errors in some references, either in in-text or reference list.	It uses a language that is appropriate with few errors in references.	It uses a language not quite appropriate, and it has got consistent errors either in in-text or reference list.	It uses a language clearly inappropriate and lacks understanding of proper citing procedures or lacks references altogether.
Argumentation	Concepts and argument clearly and accurately explained.	Concepts and argument mostly clearly and accurately explained.	Concepts and argument clear.	Concepts and argument not clearly and accurately explained.	Concepts and argument not included or misrepresented.

Joonis 1.1 Kirjaliku ülesande vastuse hindamiskriteeriumid MOOC-is [8].

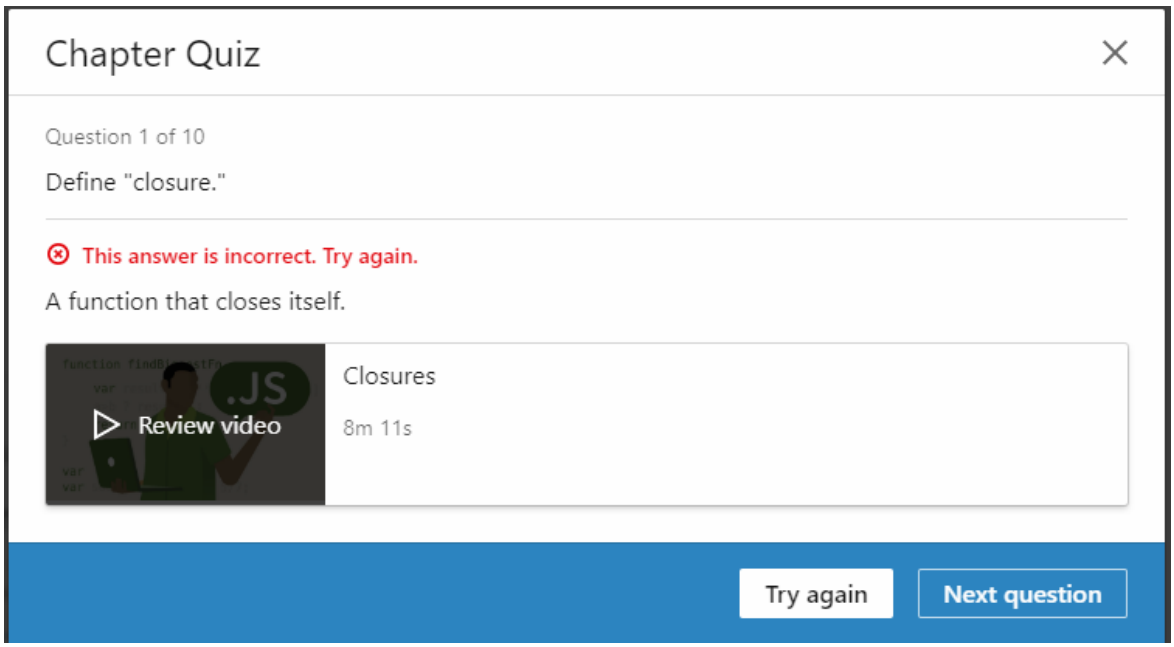
Kaaslase töö hindamise meetod on arenenud traditsioonilisest klassiruumi õpingutest, kus õpetaja läbiviimisel õpilased andsid üksteisele tagasisidet [9]. Samuti väidetakse, et klassiruumis kaaslasele tagasisidet andes mõjutab hindaja tagasisidet arvamus hinnatavast isikust.

MOOC-ides on see probleem lahendatud andes hindajale ette anonüümsed tööd. Samas on vaba juurdepääsuga e-kursuste peamine murekoht osalejate motivatsiooni puudus anda tagasisidet teistele osalejatele [9]. Samuti tuuakse välja, et võib tekkida olukord, kus hindajale on jäänud õpitavast valearusaam ja seega annab ebakorrektse tagasiside kaastudengile. Ka on probleemiks õpilaste usalduse puudus meetodi suhtes [9]. Hoolimata meetodi puudustest, on hindamiste vastavus, võrreldes ekspertide grupi hinnangutega, rahuldav ning hea tulemuse saamine kaaslastelt on korrelatsioonis kursuse eduka läbimisega [10].

1.2 Automaatne hindamine

Tehnoloogilised vahendid võimaldavad välja töötada automaatselt töötavaid kontrollimismehhanisme. Kui kaaslase töö hindamise korral, tuleb kõigepealt töö esitada ja alles teatud aja tagant saab enda hinde teada, siis arvuti abil automaatse hindamise korral saab tulemuse teada kohe peale esitamist. Levinumad viisid, kuidas arvutit hindamisel kasutada, on valikvastustega küsimuste vastuste kontrollimine ja automaatsete testide jooksutamine [11].

Valikvastustega küsimustega saab kontrollida tudengi värskelt omandatud teadmisi teema kohta ja anda otsekohe tagasisidet vastuste õigsusele. Tulemuste põhjal saab õppur teha järelduse, kas teema on piisavas mahus omandatud või tuleb materjali uuesti õppida. Selle meetodi probleem on, et pole võimalik kontrollida, kas õpilane oskab ka omandatud teadmisi kasutada ja ise funktsionaalseid ülesandeid lahendada [9]. Joonisel 1.2 on näha, et valikvastustega küsimusele on valesti vastatud ja näidatakse, millises videos küsimuse teemat õpetati, et õppur saaks soovi korral teema uuesti omandada.



Joonis 1.2 Peatüki lõpu valikvastustega testi küsimuse tagasiside.

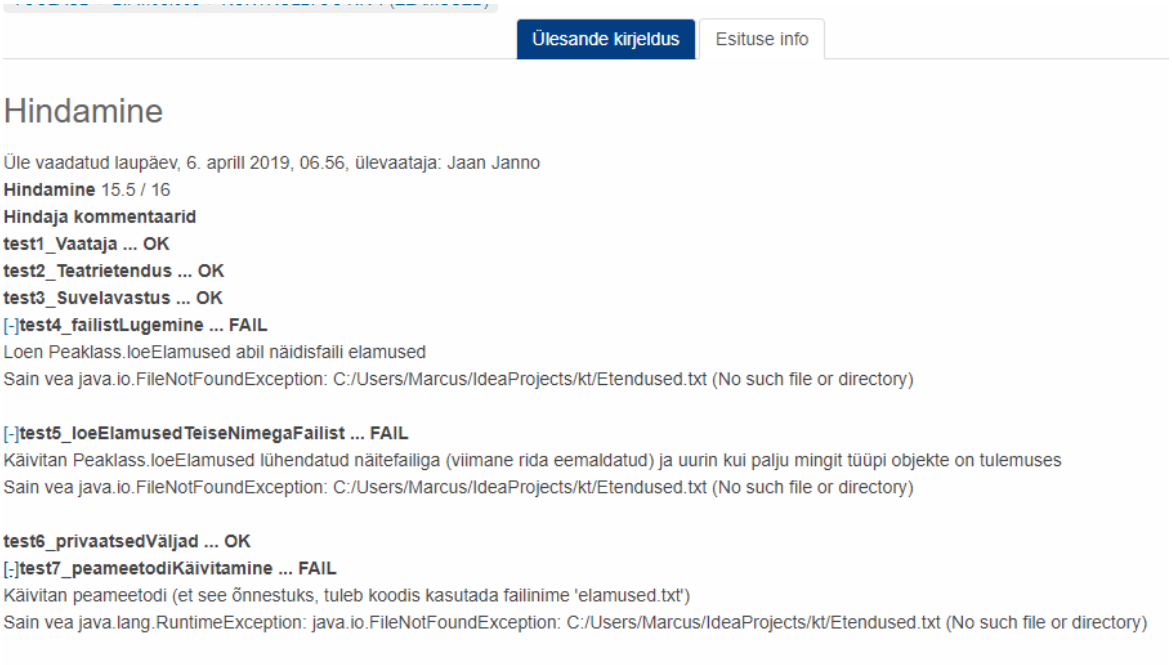
Arvutiteaduste ainete õpetamisel on olulise tähtsusega praktiliste ülesannete täitmine, sest ainult teoreetilise materjali omandamisest ei piisa õpitud teadmiste kinnistamiseks [3]. Paljude osalejatega ainete puhul tähendab see suurt kontrollitavate tööde arvu õppejõule, mistõttu on kasutusele võetud automaatsed testid [12]. Automaatsed testid võivad kontrollida esitatud töö funktsionaalsust või töö sarnasust olemasolevatele õigetele lahendustele [13].

Funktsionaalsuse kontrollimisel saab teada, kas tehtud programm vastab ülesandes kehtestatud nõuetele, kuid selle meetodi korral on võimalik, et tudeng arendab oma programmi nii, et see läbib just õppejõudude koostatud kontrolltestid, ent teiste sisendite korral ei saa programm hakkama. Samuti on puuduseks, et juhul kui koodis on mingi näpuviga, mille tõttu funktsionaalsus ei toimi, siis võetakse hindamisel palju punkte maha [13].

Töö sarnasuse kontrollimisel on programmile ette antud hulk lahendusi ja nende lahenduste tulemus ning tudeng saab oma tööle hinnangu samasuguste tööde tulemuste põhjal [13]. Koodi sarnasuse testimisel on funktsionaalsuse kontrollimisel esinenud probleemid väiksemad, sest etteantud õigete lahenduste hulka ei panda täpselt testide sisenditega töötavat koodi, mistõttu selline tudengi töö saab vähe punkte. Samuti on limiteeritud kogemata tehtud vigade mõju lõpphindele, sest ülejäänud kood sarnaneb suures osas kontrollhulga lahendustega ning seega saab kõrgema hinde kui funktsionaalsust kontrollides. Selle asemel on sarnasuse testimisel peamine probleem ebastandardseid meetodeid kasutavatele tudengitele, hoolimata lahenduse õigsusest, halvema hinnangu andmine, sest nende lahendus ei sarnane kontrollhulgas olevate lahendustega [13].

1.2.1 Automaattestid Tartu Ülikoolis

Tartu Ülikoolis on arvutiteaduste ainetes kasutatud tööde kontrollimisel funktsionaalsuse testimist. Tudeng laeb Moodle'isse üles oma töö, selle peal jooksutatakse testid ja testide tagasiside kuvatakse õppurile. Kui õppejõud on võimaldanud mitu korda kodutööd esitada, siis tudeng saab tagasiside abil vead parandada ja uuesti esitada oma töö. Ainete hulka, kus on kasutatud automaatteste, kuuluvad näiteks: „Programmeerimine" [14], „Objektorienteeritud programmeerimine" [15] (vt joonis 1.3) ning „Automaadid, keeled ja translaatorid" [16].



Joonis 1.3 Aine „Objektorienteeritud programmeerimine" kontrolltöö automaattestide tagasiside [15].

Tartu Ülikoolis on ainele „Andmebaasid" teinud bakalaureusetöö raames testimisvahendi Marten Siiber [4] ja Robert Sepp [5]. Mõlemad tõid oma töös välja, et edasi arenduse võimalus on testid panna Moodle'isse, kuid kahjuks seda pole tehtud ning kummagi programm pole kodutööde kontrollimisel kasutust leidnud. Mõlemad on kasutanud programmi kirjutamiseks Java keelt ning programmide töö põhimõte seisneb, et testandmed loetakse sisse, nende peal jooksutatakse tudengi tehtud funktsioone ja protseduure ning väljundit kontrollitakse oodatud väljundi suhtes.

Marten Siiberi [4] testimisvahend testib aines „Andmebaasid" kasutatavat andmebaasi Ope. Ta kasutas oma töös Java 7 versiooni, mis on nüüd iganenud ja Oracle[1] seda enam ei toeta

[1] Korporatsioon, mis tegeleb Java arendamisega

[17]. Andmebaasi testid on tehtud kasutades TestNG raamistikku. Tema programmi testimise protsess koosneb kahest etapist. Esimeses etapis kontrollitakse testitavas andmebaasis olevate objektide meta-andmete ühtimist oodatavate andmebaasi meta-andmetega. Teine etapp koosneb andmebaasis olevate andmete ja objektide funktsionaalsuste kontrollimisest. Moodle'isse ei õnnestunud valminud teste panna aegunud Java versiooni kasutamise tõttu.

Robert Seppa [5] testimisvahend testib „Andmebaasid" aines teist kasutatavat andmebaasi Edu. Võrreldes Marten Siiberi tööga, kasutas Sepp uuemat versiooni Javast, versioon 8. Samuti loobus ta väliste raamistikkude kasutamisest. Robert Seppa tehtud programmi probleem on testide tagasiside tehnilisus, mis raskendab leitud vea põhjusest arusaamist.

2 Aine „Sissejuhatus andmebaasidesse“

Tartu Ülikoolis on mitu ainet, kus õpetatakse andmebaaside loomist ja kasutamist. Nendest on kõige suurema osalejate arvuga aine „Andmebaasid“, mis kuulub informaatika, arvutitehnika ning matemaatilise statistika õppekavadesse. Kuna viimastel aastatel on suurenenud nendel õppekavadel õppivate tudengite arv [1], siis on ka suurenenud aines „Andmebaasid“ osalejate arv [18], mistõttu kulub õppejõul rohkem aega iseseisvate tööde hindamiseks. Kontrollimiseks kuluvat aega aitab vähendada automaattestide loomine.

Selle bakalaureusetöö raames keskenduti testide loomisele ainele „Sissejuhatus andmebaasidesse“. Kuigi „Sissejuhatus andmebaasidesse“ aines pole nii palju osalejaid kui „Andmebaasid“ ainel, siis on ka sellel ainel mitu erinevat rühma ning seda antakse nii päevases õppes ja sessioonõppes kui ka inglise keeles [19]. Võrreldes „Andmebaasid“ ainega, kus iga õpilane genereerib suvalised andmed oma andmebaasi, siis „Sissejuhatus andmebaasidesse“ kursusel on sessioonõppes aines osalejatel samad andmed [20]. See lihtsustab ja kiirendab testimise protsessi, sest pole vaja andmebaasi sisse laadida testandmeid.

Alljärgnev kirjeldus põhineb sessioonõppe kursuse „Sissejuhatus andmebaasidesse“ Moodle’i lehel [20]. Aines „Sissejuhatus andmebaasidesse“ õpetatakse osalejatele andmebaaside algteadmisi, nagu andmebaasi ja vaadete loomine ning päringute koostamine. Kursuse läbimiseks peab tudeng saama vähemalt pooled punktid loengu- ja praktikumipunktidest, oma projektist ning päringu koostamise eest ja saama üle poolte punktide arvestustestis. Töö kirjutamise ajal on automatiseeritud loenguküsimuste vastuste ja osa arvestustesti küsimuste hindamine. Töö tulemusena on võimalik automaatselt kontrollida ka osasid sessioonõppe iseseisvaid ülesandeid. Kuna sessioonõppes õpivad enamasti rakenduskõrgharidust omandavad tudengid, kes käivad ülikoolis kord kahe nädala tagant, on kontakt õppejõuga väike, seega annavad automaattestid neile kiirema tagasiside töö korrektsuse kohta.

Iseseisvad ülesanded jagunevad:

1. vabalt valitud tabeli loomine ja andmete sisestamine;
2. tabelite loomine ja andmete importimine;
3. tabelite vahele seoste loomine ja päringute tegemine;
4. keerulisemate päringute tegemine;
5. vaadete koostamine;

Esimeses individuaalses ülesandes tuleb tudengil paberi peale kavandada tabel ning välja mõelda millised ja mis kujul andmed sinna panna. Seejärel tuleb andmebaasis luua see tabel vastavalt kirjeldusele ning lisada vähemalt neli kirjet tabelisse.

Tabelite loomise ja andmete importimise ülesandes tuleb luua neli tabelit: Laulud, Lauljad, Voistlused ja Riigid. Seejärel tuleb tekstifailist importida andmed tabelitesse. Tabelite kirjeldused on saadaval praktikumi slaididel ja imporditavad andmed on alla laetavad Moodle'ist.

Kolmandas ülesandes tuleb luua tabelite vahele välisvõtmed. Kokku luuakse kuus välisvõtit, millest kaks tehakse praktikumis ning ülejäänud tuleb õppijatel ise teha. Samuti tuleb teha kaks andmebaasi päringut, kus tuleb kasutada tabelite vahele loodud seoseid.

Keeruliste päringute töö on mõeldud tudengitele, kes praktikumis ei käinud. Tuleb alla laadida Wordi dokument, kus on lünktekst, mille saab täita tehes päringuid andmebaasi. Kuigi töö kirjutamise ajal täidetakse lünkteksti faili, siis tulevikus on võimalik sama ülesanne teha Moodle'i testina, mis kontrollib automaatselt lünkadesse sisestatud vastuseid.

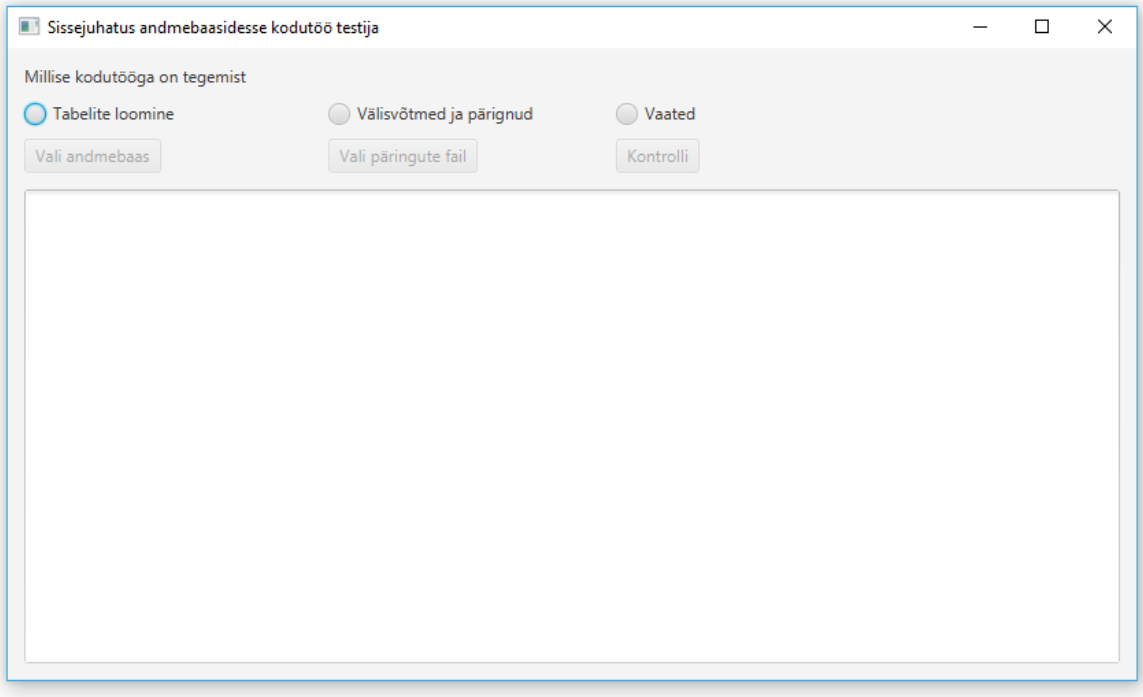
Viimane iseseisev töö on vaadete tegemine. Kokku tuleb teha neli vaadet, millest kolm peavad vastama etteantud kirjeldusele ja neljas võib olla vabalt valitud. Neljanda vaate ainus tingimus on, et see peab endas sisaldama andmeid kahest erinevast tabelist. Testimise lihtsustamiseks saab ka neljas vaade kindla kirjelduse.

Nendest ülesannetest on võimalik teha automaatsed testid teisele, kolmandale ja viiendale ülesandele. Esimene ülesanne jääb endiselt õppejõu kontrollida, sest selles loob andmebaasi tudeng vabalt valitud tabeli ning seda on raske automaatselt kontrollida. Neljandale ülesandele on võimalik luua automaatne kontroll, kasutades Moodle'is olemasolevaid vahendeid.

3 Automaattestimise vahendi tutvustus

Automaattestimise vahendi teostamisel analüüsiti 2018/2019 õppeaasta sügissemestril sessioonõppe aines „Sissejuhatus andmebaasidesse“ osalenud tudengite tehtud töid. Kokku registreeris ainele 43 inimest [19]. Programmi loomisel ja tagasiside koostamisel võeti arvesse tudengite töödes tehtud vigu.

Automaattestimise vahend ja testid sessioonõppe ainele „Sissejuhatus andmebaasidesse“ valmis kasutades programmeerimiskeelt Java ning selle versiooni 8. Kuigi Javast on ka olemas uuemad versioonid, siis otsustati kasutada versiooni 8, sest Moodle’is kasutatav programmeerimiskeel VPL ei ühildu Javaga ning Tartu Ülikoolis on välja töötatud eraldi raamistik, et jooksutada teste Java serveris, mille versioon on 8[2]. Raamistikust tulenevalt on nõue kasutada JUnitit testide tegemiseks[3]. Kuna tekkis probleeme testide Moodle’isse integreerimisega, siis valmis töö käigus peale testide ka graafilise liidesega programm (vt joonis 3.1), mida saab käivitada oma arvutis. Programmi loomisel kasutati Mavenit[4], sest töö autoril on selle kasutamisega kogemusi ning see lihtsustab käivitatava JAR-faili loomise protsessi.



Joonis 3.1 „Sissejuhatus andmebaasidesse“ iseseisvate tööde testimisvahendi graafiline liides.

[2] Meilivestlusest Maria Gaidukiga, 14. jaanuar 2019.
[3] Meilivestlusest Aivar Annamaaga, 17. aprill 2019.
[4] Deklaratiivne tarkvara ehitamise automatiseerimise rakendus.

Graafilise liidesega programm on mõeldud õppejõule, et kontrollida tudengi esitatud tööd kiiremini kui seda manuaalselt SQL Centralis[5] tehes. Moodle'ile tehtud testid on mõeldud tudengitele, et nad saaksid tagasisidet oma esitatud töö kohta ning soovi korral teha parandusi. Sihtgrupi ja kasutatava keskkonna erinevuste tõttu on testide ja testimisvahendi tööprotsessis ja kasutamises erinevusi.

Moodle'is kasutatavad testid on tehtud JUnitiga, mis seab testidele mõningad piirangud. Peamine puudus on, et teste ei saa taaskasutada, sest Moodle'i testid on ülesande spetsiifilised, ja kui muutub ülesanne, siis tuleb ka testid uuesti kirjutada. Samuti on Moodle'i testidel eripära, et kasutaja tööd kontrollitakse kuni esimese veani testis. Näiteks esimese kodutöö kontrollimisel, kui tudeng on teinud vea tabeli tulba tüübi määramisel, siis tagastatakse vastav teade, kuid järgmisi tabeli tulpi ei kontrollita. Graafilise liidesega programmi puhul neid piiranguid ei ole.

3.1 Esinenud probleemid

Testide ja testimisvahendi tegemisel esines takistusi, millest enamus sai lahendatud kokkuleppel juhendajaga. Peamised probleemid olid soov vältida muutuja liitmist otse andmebaasi päringusse, välisvõtmete kirjelduse ning tudengi vaadete saamine andmebaasist ja täpitähtede kuvamine valminud programmis. Samuti oli probleemiks valminud testide Moodle'isse panemine, mis ei õnnestunudki.

Muutujate otse andmebaasi päringusse liitmine on turvaoht, sest seeläbi on ründajal võimalus käivitada pahatahtlikke andmebaasi käsklusi. Kuigi valminud testide ja programmi korral on ründe tõenäosus väga väike, sest ründe saab teha ainult tudeng ning ta rikub sellega ainult enda esitatud andmebaasi, siis on hea tava vältida muutujate otse liitmist päringutesse [21]. Javas on selle vältimiseks klass *PreparedStatement*, kus väärtustatakse päringus muutujate väljad ükshaaval meetodi välja kutsumisega. Enamus andmebaasi päringuid on tehtud kasutades seda klassi, kuid osadel päringutel on muutuv osa tabelinimi ning selle hiljem lisamine pole lubatud. Seega sai nende päringute puhul kasutatud tabelinime otse päringu lausesse liitmist. Samuti on võimalik, et halva otstarbega andmebaasi käsklused esitatakse päringute töö failis, kuid päringute failis olevatele käsklustele ei ole lisa turvameetmeid rakendatud, sest ründeoht on väike.

[5] Rakendus, mis kuvab informatsiooni andmebaasis olevate objektide kohta.

Välisvõtmete loomisel täpsustatakse tabelid ja tulbad, mille vahel võti luuakse. Samuti määratakse tegevused, mida tehakse, kui andmed muutuvad või kustutatakse. Tegevused jaotuvad neljaks: keela muutus või kustutamine, mine kaasa muutuse või kustutamisega, määra NULL väärtus ja määra vaikeväärtus. Andmebaasis hoitakse iga välisvõtme tegevusi tabelis SYSTRIGGER, kuid kuna vaikimisi kasutatakse välisvõtmete puhul keela tegevust, siis tabelis pole selle tegevusega kirjeid. Seetõttu, kui küsitakse tudengi välisvõtmeid andmebaasist ning tegevus, mida tehakse andmete kustutamisel või muutmisel, puudub, siis pannakse tegevuseks keela.

Vaadete kodutöös tuli tudengil luua neli vaadet. Vaated asuvad andmebaasis tabelis SYSVIEW, kus on vaate nimi, vaate autor ja päring, millega vaade loodi. Nende andmete põhjal on raske otsustada, milline vaade vastab millisele nõutud kirjeldusele. Seega kooskõlastati juhendajaga, et kontrollimist lihtsustada, peavad tulevikus tudengid looma ettemääratud nimedega vaateid. Kaaluti ka andmebaasis tudengi tehtud vaadete arvu piiramist neljale, kuid siis on võimalus, et tudeng teeb vaated teises järjekorras kui on oodatud ja kontrollitakse valesid vaate päringuid omavahel.

Testimisprogrammile graafilise liidese tegemisel, tekkis probleem, kus kui tagasisides esines täpitähti, siis täpitähtede asemel kuvati imelikud sümbolid. Marten Siiberil oli oma töö tegemisel sama probleem olnud ja tema lahendus parandas täpitähed ka valminud programmis ära [4]. Tema parandus oli, et täpitähed esinevad kahe baidina, millest esimese väärtust ignoreeritakse ja teise väärtuse järgi otsustakse, mis täht kuvada.

Java testide Moodle'isse panemiseks on vaja serverile, kus testid käivitatakse, installeerida SQL Anywhere 17. Selleks, et programm installeerida, on vaja ligipääsu serverile. Peamine probleem oli Tartu Ülikoolist õige inimese leidmine, kes teab, kuidas Java automaattestid toimivad ning kellel on ligipääs testserveritele ja aega tegeleda SQL Anywhere 17 installeerimisega serverisse. Töö kirjutamise ajal pole veel õnnestunud installeerida SQL Anywhere 17 serverisse.

3.2 Iseseisvate tööde kontrollimine

Nii Moodle'i testide kui ka graafilise liidesega programmi puhul luuakse algul ühendus andmebaasiga. Kui see ei õnnestu, siis tagastatakse viga ja andmebaasi ei hakata testima. Üks põhjus, miks andmebaasiga ei õnnestu ühendust luua võib olla tudengi pandud vale kasutajanimi ja parool andmebaasile. Kuigi praktikumi slaididel on näidatud, et kasutajanimi panna „dba" ja parool „sql" [20], siis esines ikkagi andmebaase, kus kasutajanimi ja parool

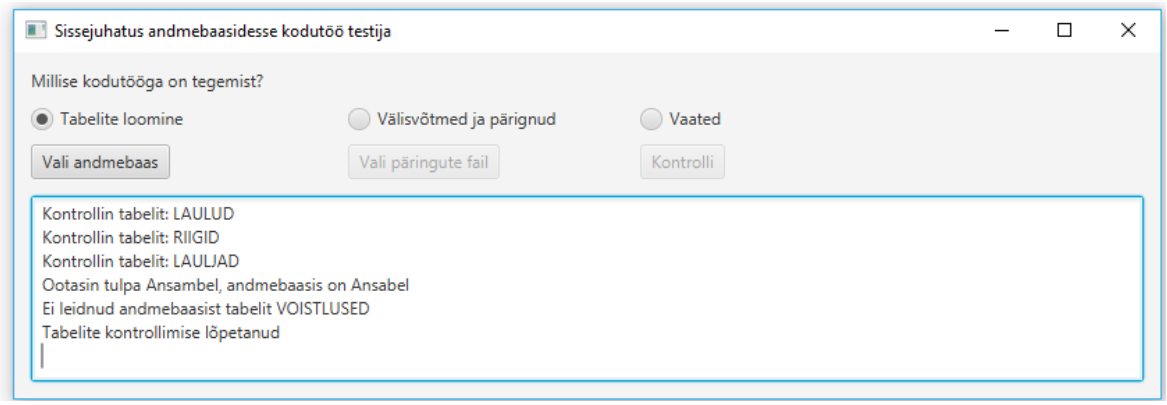
oli midagi muud. Pärast edukat ühenduse loomist, hakatakse kontrollima andmebaasis eelnevalt valitud ülesannet ning pärast kontrollimise lõppu suletakse ühendus andmebaasiga.

3.2.1 Tabelite loomine ja andmete importimine

Selles individuaalses töös tuli tudengitel luua tabelid ja nendesse andmed importida failidest. Peamised probleemid, mis tudengid teevad valesti selle ülesande lahendamisel, on tabelile või tulbale vale nime panemine või ei õnnestu importida andmeid tabelisse.

Lisas 1 on välja toodud töökäik, kuidas igat tabelit testitakse pärast andmebaasiga ühenduse loomist Moodle'i testides. Iga tabel kontrollitakse sõltumata teistest ehk kui üks tabel on vigane, siis kontrollitakse teisi tabeleid ikkagi. Juhul kui tudeng on teinud tabeli teise nimega, siis kuvatakse talle teade, et õige nimega tabelit ei leitud ning vale nimega tabelit ei kontrollita. Vigase nimega tulba korral on lubatud õige tulba nimega võrreldes ühe tähemärgi erinevus. Kuna tavalist väljundit pole võimalik kuvada tudengile raamistiku eripärade tõttu ning kirjaviga tulba nimes ei ole suur eksimus, siis sellekohast veateadet ei kuvata töö esitajale ning kontrollitakse tabeli tulpasid edasi. Andmete importimist tabelitesse kontrollitakse leides tabelis olevate ridade arvu ning vaadates kas see on võrdne oodatavate ridade arvuga. Ridade sisu ei kontrollita, kuna võib tekkida probleem andmebaasi või andmete faili kodeeringuga ja samuti kui mõni tulba tunnus või kitsendus on vale, siis ei lisata andmeid tabelisse ning see tuleb ridade arvu kontrollimisel välja.

Joonisel 3.2 on näha õppejõule kuvatav andmebaasi tagasiside, kui kontrollitakse tabelite loomise tööd. Kontrollitud andmebaasis on Lauljad tabeli tulbal vale nimi ning üks nõutav tabel on puudu.



Joonis 3.2 Näidis õppejõu programmi tagasisidest tabelite loomise töö kontrollimisel.

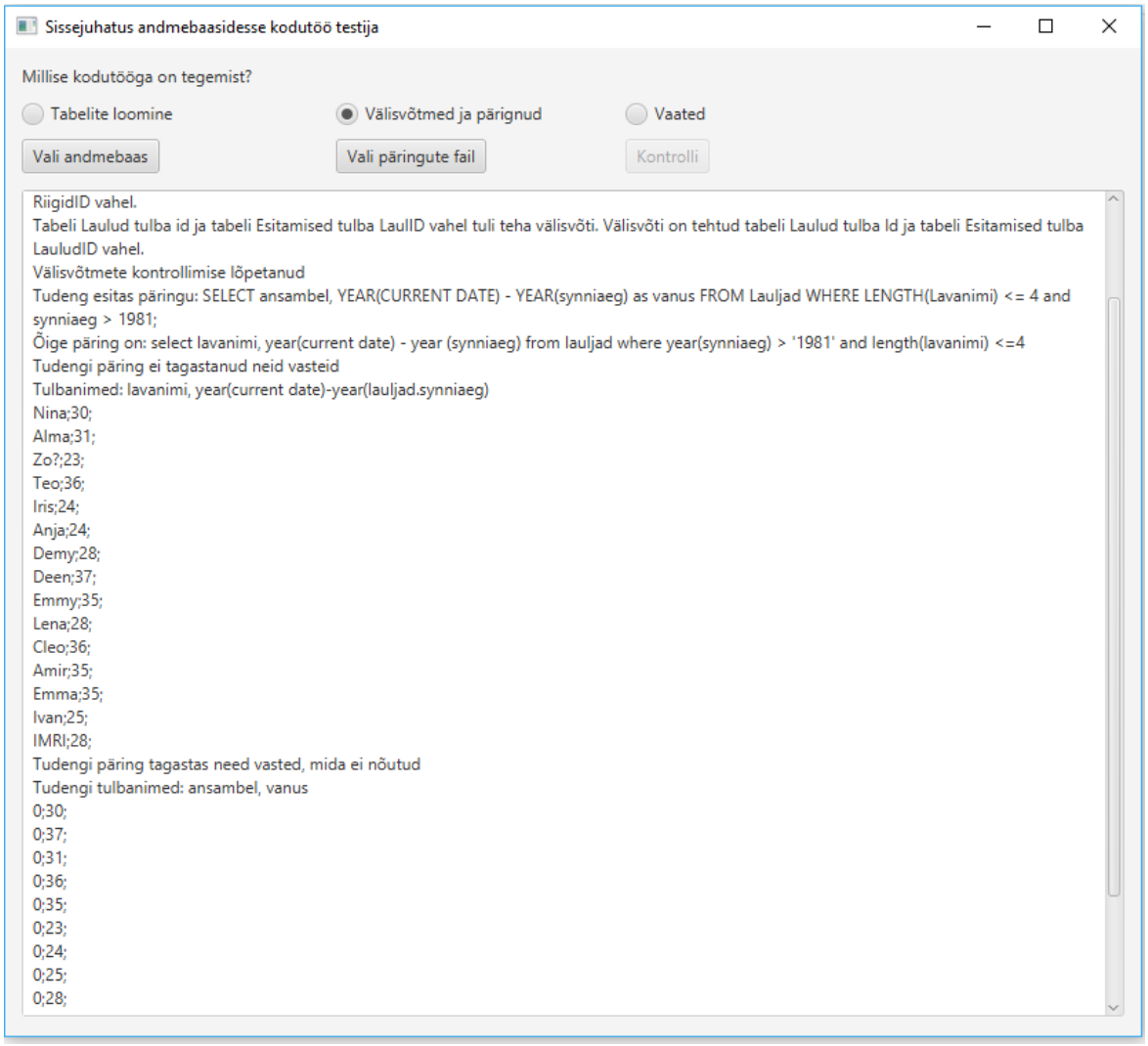
Lisas 2 on näidatud õppejõu programmis tabelite testimise protsess. Kui Moodle'is kontrollides testitakse kõiki tabeleid eraldi, siis õppejõule mõeldud programmis kontrollitakse tabeleid üksteise järel, kuid kui esineb viga, siis ei lõpetata tööd, vaid kuvatakse veatekst ning jätkatakse kontrollimist. Juhul kui tulba nimes on tehtud viga, siis kuvatakse tekst, et tulba nimi ei ole sama oodatava nimega.

3.2.2 Välisvõtmete loomine ja lihtsad päringud

Tuli luua kuus välisvõtit erinevate tabelite vahel ja koostada kaks ülesandes kirjeldatud päringut. Välisvõtmete tegemisel olid tavalised vead seoste valepidi loomine tabelite vahele ja seoste loomisel tabelis puuduva tulba nime kasutamine. Mõlema vea korral tekib tabelisse juurde uued tulbad nende nimedega, mis olid välisvõtme loomise käsus, kuid reaalset seost tabelite vahele ei teki. Samuti eksiti välisvõtmete tegevuste määramisega. Päringute loomisel tehti vigu päringu vastavusse seadmisel ülesandes kirjeldatud töökäsuga. Lihtsamad vead olid lisa tulpade kuvamine ja töökäsu ebatähelepanelik lugemine. Suurem viga oli töökäsus kirjeldatud nõuetele mitte vastav päring.

Lisas 3 on välja toodud Moodle'i testide käivitamise, välisvõtme ja päringu testimise protsess. Moodle'is testitakse välisvõtmeid sarnaselt tabelitega. Iga välisvõti kontrollitakse teistest eraldi. Andmebaasiga ühenduse loomisel küsitakse andmebaasist kõik tudengi tehtud välisvõtmed. Iga õige välisvõtme definitsiooniga on seotud üks test, mis otsib andmebaasist saadud võtmete seast, kas selliste tabelite vahel on välisvõti ja kui on, siis kontrollib, kas välisvõti on õigete tulpade vahel ja on õiged tegevused määratud. Samuti kontrollib, kas tabelite vahel on võti teistpidi loodud. Vea korral tagastatakse vastav veatekst. Päringute kontrollimisel jooksutatakse nii õiget päringut kui ka tudengi päringut andmebaasis. Kontrollitakse, kas mõlemal päringul on sama tulpade arv ja sama ridade arv. Samuti vaadatakse, kas ridade sisu on sama. Kui ridade sisu on erinev, siis kuvatakse „Vigane päring".

Joonisel 3.3 on näha tagasiside andmebaasile, milles on tabelite vaheline välisvõti tehtud valede tulpade vahel. Samuti on kontrollitud esitatud päringute faili ning selle testimisel leiti, et päringute failis olev päring ei tagasta samu vasteid õige päringuga.



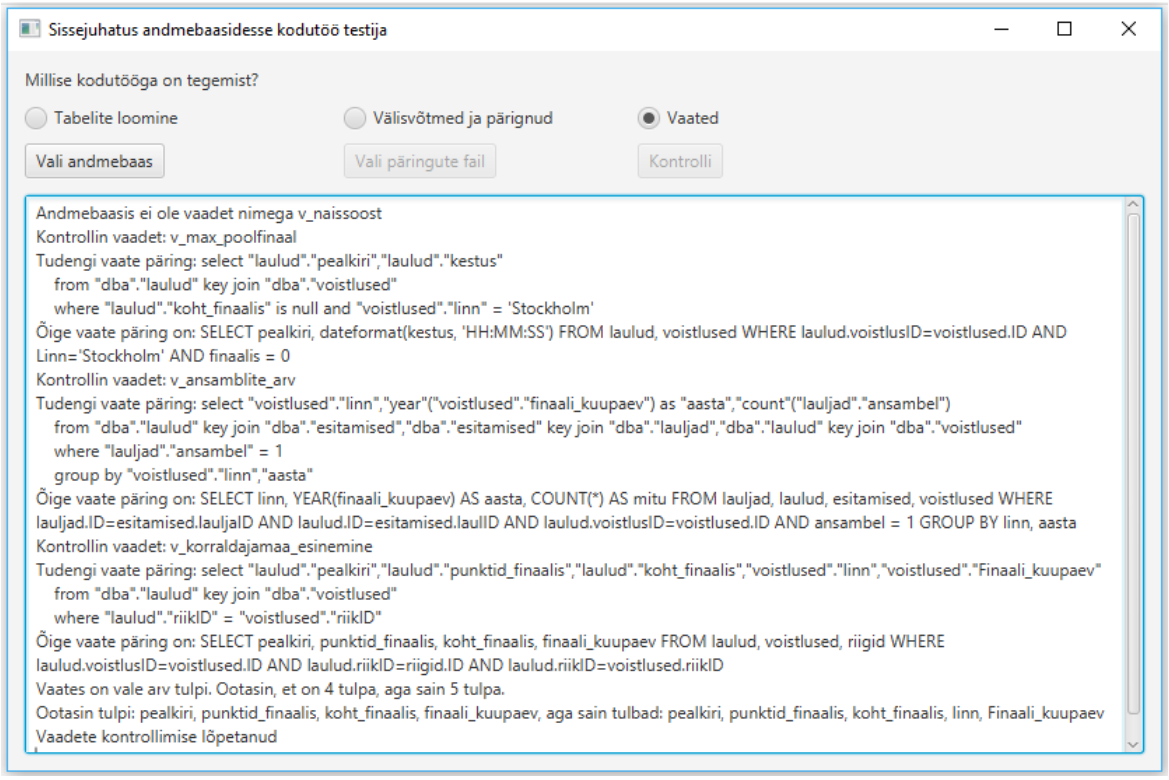
Joonis 3.3 Näidis tagasiside välisvõtmete ja päringute kontrollimisel õppejõu programmis.

Lisas 4 näidatakse, testimisvahendi andmebaasi kontrollimise protsessi, kui soovitakse testida välisvõtmete ja päringute õigsust. Õppejõule mõeldud programmi puhul loetakse välisvõtmete kirjeldused failist sisse. Seejärel leitakse andmebaasis tudengi tehtud välisvõtmed ning kontrollitakse, kas need on sama kirjeldusega nagu failist loetud võtmed. Erinevuse korral kuvatakse erinevus õppejõule ja jätkatakse teiste võtmete kontrollimist. Samuti teavitatakse, kui tudeng on teinud seose valepidi tabelite vahele. Päringute kontrollimisel loetakse õiged päringud failist ning võrreldakse nende tulemust tudengi päringute, mis saadakse samuti failist, tulemusega. Erinevus Moodle'is päringu testimisega, seisneb selles, et kui päringute vastused on erinevad, siis õppejõule näidatakse vasteid, mis saadi õige päringuga, kuid mida ei saanud tudengi päringuga ja vasteid, mis saadi tudengi päringuga, kuid mida ei saadud õige päringuga.

3.2.3 Vaadete loomine

Tudengitel tuli luua andmebaasis neli vaadet, millest kolm pidid vastama nõutud kirjeldusele ning neljas pidi olema tehtud vabalt valitud kahe tabeli vahel. Tavalised eksimused olid valede tulpade näitamine või valede tabelite kasutamine andmete saamiseks. Moodle'i testides kontrollitakse ainult kolme ülesandes defineeritud vaadet, kuna neljandat vaadet, mis on tudengi vabalt valitud andmetest, pole võimalik kontrollida, sest puudub õige vaade, mille vastu valideerida vaate sisu. Õppejõu programmis kontrollitakse nii palju vaateid kui on õigete vaadete failis.

Joonisel 3.4 on näha tagasiside tudengi vaadete tööle. Andmebaasist otsiti nelja vaadet, ühte vaadet ei olnud andmebaasis ning ühel vaatel oli vale arv tulpi. Ülejäänud vaated olid tehtud korrektselt.



Joonis 3.4 Vaadete kontrollimise tagasiside.

Lisades 5 ja 6 näidatud, kuidas kontrollitakse vaadete iseseisvat ülesannet Moodle'i testides ja õppejõu programmis. Mõlemas käib vaadete kontrollimine sarnaselt päringute kontrollimisega. Kontrollitakse, kas tulpade ja ridade arv on sama ning kui ei ole, siis kuvatakse vastav teade. Ridade sisu erinevuse korral kuvatakse Moodle'is „Vigane vaade", õppejõu programmi korral kuvatakse erinevus tudengi ja õige vaate päringu vahel.

3.3 Võrdlus eelnevate töödega

Valminud testimisvahendis esineb erinevusi nii Marten Siiberi [4] kui ka Robert Seppa [5] tehtud töödest. Kui nende testimisvahendid olid mõeldud „Andmebaasid“ aines olevate andmebaaside testimiseks, siis uus testimisprogramm on „Sissejuhatus andmebaasidesse“ aines kasutatav. Siiberi [3] ja Seppa [4] testimisvahendid töötasid põhimõttel, et laaditakse testandmed testitavasse andmebaasi ning siis kontrollitakse, kas saadakse oodatud tulemused. Valminud testimisvahendi puhul pole testandmeid vaja andmebaasi lisada ja seetõttu on testimine kiirem. Samuti testisid eelnevad programmid andmebaasi funktsioone ja trigereid, kuid kuna neid aines „Sissejuhatus andmebaasidesse“ ei käsitleta, siis neid ei testita uues programmis.er

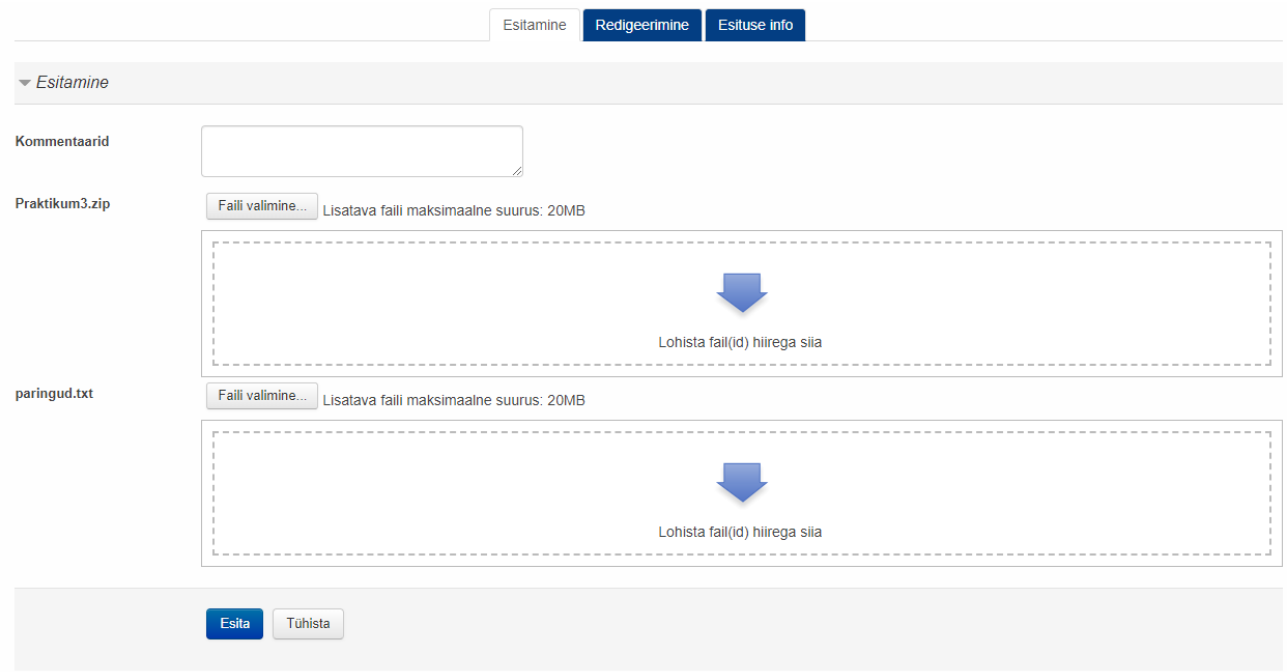
Nii Siiber [4] kui Sepp [5] kasutasid oma programmi tegemiseks Java programmeerimiskeelt ning graafilise liides tehti JavaFX abil. Samasid vahendeid kasutati ka uue programmi loomisel. Marten Siiber [3] kasutas testide tegemiseks TestNG raamistikku ja Robert Sepp [4] otsustas teha testimisvahendi ilma testimisraamistikuta. Uus testimisvahend on samuti ilma välise testimisraamistikuta, kuid testid, mis on mõeldud Moodle’isse panemiseks on tehtud JUnit raamistikuga. Andmebaasiga ühendumiseks on kõikidel programmidel vaja JDBC draiveri olemasolu.

3.4 Kasutusjuhend

Moodle’i testide ja lokaalselt jooksutava programmi kasutamine erineb üksteisest mõne nüanssi poolest. Mõlema edukaks töötamiseks peab arvutis olema installeeritud SQL Anywhere 17. Samuti on vaja, et on olemas oodatud tulemuste failid töökataloogis ja et on testitav andmebaas, millega ühendus luua.

3.4.1 Moodle’i testid

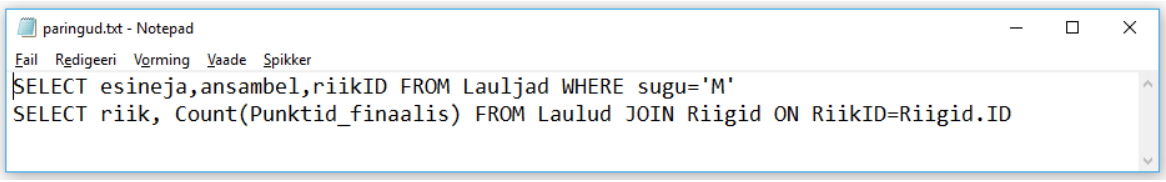
Tudengile avaneb töö esitamisel Moodle’is koht (vt joonis 3.5) , kuhu laadida oma töö. Tabelite loomise ja vaadete töö korral on vaja laadida üks kokku pakitud fail, mille nimi on *Praktikum<praktikumiNumber>.zip*, näiteks *Praktikum1.zip*. Selles failis peab olema nii andmebaasi fail kui ka andmebaasi logi fail. Välisvõtmete ja päringute töös peab lisaks kokku pakitud failile olema tekstifail *paringud.txt*, milles on nõutud päringud. Kui nõutud nimega faile ei esitada, siis saab tudeng veateate. Testid, mis kontrollivad tudengi tööd peavad olema Java serveris ning Moodle peab olema seadistatud neid käivitama. Tudengi töö esitamisel pakitakse tihendatud fail lahti testide töökataloogi ning käivitatakse testid.



Joonis 3.5 Tudengi vaade välisvõtmete ja päringute töö esitamisel.

Tabelite loomise ja andmete importimise ülesande puhul on vaja, et tabeli tulpade õige kirjeldus ja tabeli sisu oleks failides, mis asuvad töökataloogis. Tabeli tulpade kirjeldus peab olema failis *<Tabelinimi>ColumnDescriptions.txt*, näiteks *LauludColumnDescriptions.txt*. Tabeli sisu peab olema failis, mille nimi on *<Tabelinimi>.txt*, näiteks *Laulud.txt*.

Välisvõtmete loomise ja päringute tegemise kontrollimisel, pole vaja oodatavaid tulemusi failidesse panna, kuna need on koodi sisse kirjutatud. Juhul kui ülesanne muutub, siis tuleb muuta ka koodi kirjutatud oodatavaid tulemusi. Tudengite esitatud päringute fail (vt joonis 3.6) peab olema nimega *paringud.txt* ja selles peab olema kaks päringut eraldi ridadel üksteise järel.



Joonis 3.6 Tudengite esitatava päringute faili näidis.

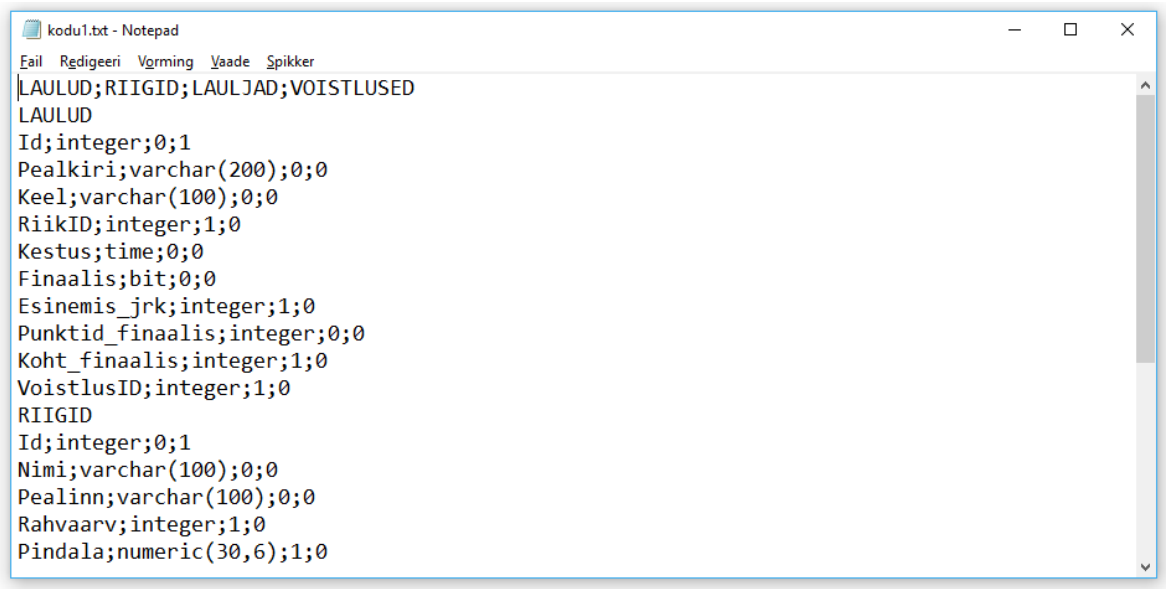
Vaadete kontrollimisel on päring, mis tagastab oodatud vaate sisu, koodi sisse kirjutatud. Samuti on olemas vaate nimi, mille järgi otsitakse vaadet andmebaasist. Juhul kui sellise nimega vaadet pole, siis selle vaate test ebaõnnestub.

3.4.2 Õppejõu programm

Programm on käivitatava JAR-faili kujul. Käivitatava programmi saamiseks lähtekoodist on vaja, et arvutis on installeeritud Maven ning tuleb käsureal lähtekoodi kataloogis jooksutada käsk „mvn clean compile assembly:single“. Käivitatav JAR-fail tekib kausta *target* nimega *database.tester-1.0-jar-with-dependencies.jar*.

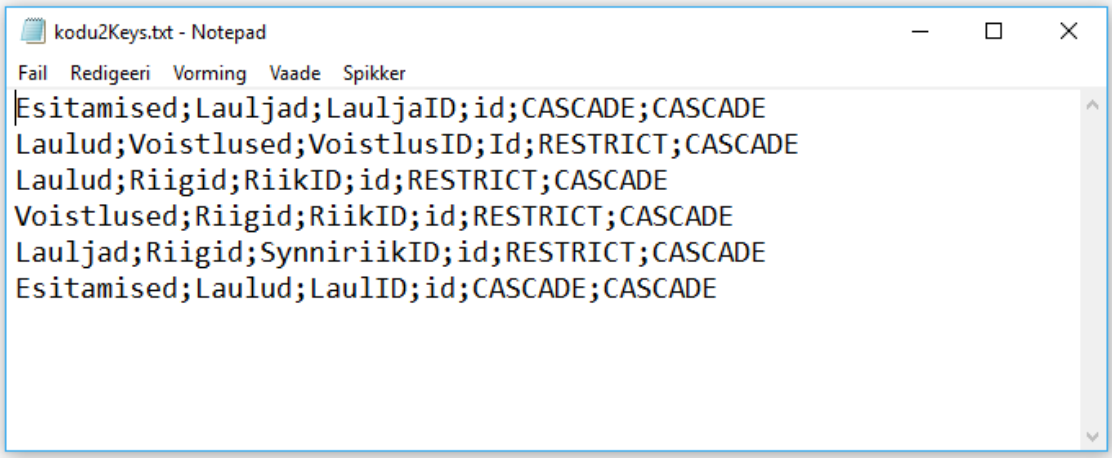
Iga kodutöö õiged vastused peavad olema kindla nimega failides, mis asuvad programmi töökataloogis. Käivitamisel kuvatakse programmi aken (vt joonis 3.1), kus saab valida, millist kodutööd kontrollitakse. Vastavalt tehtud valikule, muutuvad nupud vajutatavaks. Kui on valitud „Tabelite loomine“ või „Vaated“, siis saab vajutada „Vali andmebaas“ nupule ja kui andmebaas valitud, muutub aktiivseks „Kontrolli“ nupp. „Välisvõtmed ja päringud“ kontrollimise korral muutub aktiivseks ka „Vali päringute fail“ nupp. Kui andmebaasi ja päringute fail on valitud, saab vajutada „Kontrolli“ nupu peale.

Tabelite loomise ja andmete importimise kontrollimisel peab olema fail nimega *kodu1.txt* (vt joonis 3.7) ja failid <*Tabelinimi*>*.txt*, mis asuvad töökataloogis. Esimese faili esimesel real on kõik kontrollitavate tabelite nimed semikoolonitega eraldatult, sellele järgneb eraldi ridadel tabeli nimi ja selle tabeli tulpade kirjeldus. Tulpade kirjeldust on võimalik saada andmebaasist päringuga ning kopeerides päringu tulemuse faili. Iga tulba puhul peab tema tunnuste vahel olema semikoolon. Failides, mille nimi on kujul <*Tabelinimi*>*.txt*, on tabelitesse imporditud andmed.



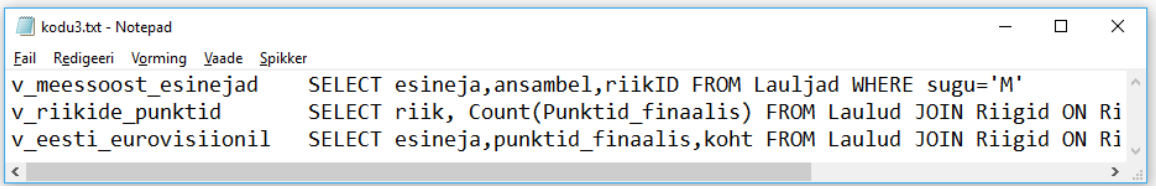
Joonis 3.7 Tabelite ja tulpade kirjeldus failis *kodu1.txt*.

Välisvõtmete kontrollimisel on vajalik faili *kodu2Keys.txt* (vt joonis 3.8) olemasolu programmi töökataloogis. Sellest failist loetakse õiged välisvõtmete kirjeldused, mis on eraldi ridadel. Iga välisvõtme puhul on vaja ülem- ja alamtabelit ning nendes omavahel seotavate tulpade nimesid. Samuti on välja toodud tegevused, mida tehakse kui andmeid kustutatakse või uuendatakse. Kõik need tunnused peavad omavahel olema eraldatud semikooloniga. Päringute kontrollimisel peavad õiged päringud olema failis *kodu2Queries.txt* ja tudengi kontrollitavad päringud eraldi tekstifailis. Mõlemale failile kehtivad samad vormistusreeglid nagu Moodle'i testide korral esitatavale *paringud.txt* (vt joonis 3.6) failile.



Joonis 3.8 Välisvõtmete kirjeldus failis *kodu2Keys.txt*.

Vaadete kontrollimisel peab iga vaate nimi ja vaate koostamisel kasutatav päring olema failis *kodu3.txt* (vt joonis 3.9) eraldi real ning nimi ja päring eraldatud tabulaatoriga.



Joonis 3.9 Vaate nimi ja vaate päring failis *kodu3.txt*.

Juhul kui vajalikke faile töökataloogis ei ole kuvatakse vastav informatsioon kasutajale. Samuti teavitatakse kasutajat ebasobiva faili vormingu korral ja juhul kui testimisel tuleb ootamatu viga.

3.5 Edasiarendus

Selle töö raames valminud programmi kui ka Moodle'i teste on võimalik edasi arendada. Töö kirjutamise hetkel ei ole Moodle'i testid Moodle'isse pandud, seega on üks edasi arendamise võimalus valminud testid teha kasutusvalmis Moodle'i keskkonnas. Samuti on võimalik teha paremaks nii vaadete kui ka päringute kontrollimisel antav tagasiside. Moodle'ile tehtud testides öeldakse ainult, kas päring või vaade on õige või vale, mis ei anna tudengile tagasisidet, milles ta päringu või vaate loomisel eksis. Veel tuleb uurida võimalusi, et muuta Moodle'i testid üldisemaks, et neid ei pea ülesande kirjelduse muutmisel uuesti kirjutama. Õppejõu programmis saab parandada tagasiside kujundust, et oleks lihtsam eristada kasulikku informatsiooni. Samuti on võimalik uurida, kas selle tööga valminud teste või programmi on võimalik kasutada aine „ Andmebaasid" iseseisvate tööde hindamiseks.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli luua automaattestid sessioonõppe aine „Sissejuhatus andmebaasidesse“ iseseisvatele töödele, et vähendada iseseisvate tööde kontrollimiseks kuluvat aega ning anda tudengitele kiiremini tagasisidet. Töö eesmärk sai täidetud, valmisid testid, mida on võimalik panna Moodle’isse, et need kontrolliksid tudengite esitatud andmebaase. Samuti valmis programm, mida õppejõud saab jooksutada oma arvutis, et kontrollida tudengi esitatud andmebaasi ja päringuid. Nii testidega kui ka testimisvahendiga on võimalik kontrollida, kas tudeng on andmebaasi õiged tabelid, välisvõtmed ja vaated loonud. Samuti saab kontrollida tudengi koostatud päringuid.

Testid ja testimisvahend valmisid kasutades programmeerimiskeele Java versiooni 8. Testide tegemiseks kasutati JUnit raamistikku. Testimisvahendi graafiline liides tehti JavaFX abil.

Edasiarendusena on võimalik valminud testid Moodle’isse panna, et need annaksid tudengite töödele tagasisidet. Samuti on võimalik teha programmi ja testide päringu ning vaadete ülesande tagasisidet paremaks, et oleks aru saada, milles viga seisnes.

Viidatud kirjandus

[1] T. Ülikool, „Tartu Ülikooli sisseastumise statistika,“ Tartu Ülikool, 2018. Saadaval: https://www.ut.ee/et/sisseastumine/statistika-0. [Kasutatud 4 aprill 2019].

[2] Tartu Ülikool, „ÕIS2: 2476 Informaatika (180 EAP),“ 2019. Saadaval: https://ois2.ut.ee/#/curricula/2476/version/2018/details. [Kasutatud 9 mai 2019].

[3] T. Staubitz, H. Klement, J. Renz, R. Teusner ja C. Meinel, „Towards practical programming exercises and automated assessment in Massive Open Online Courses,“ %1 *2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering* , 2015.

[4] M. Siiber, „Arvutiteaduse instituut - Lõputööderegister: Aine "Andmebaasid“ automaattestimise vahend,“ 2015. Saadaval: https://dspace.ut.ee/bitstream/handle/10062/56117/thesis.pdf?sequence=1&isAllowed=y. [Kasutatud 12 jaanuar 2019].

[5] R. Sepp, „Arvutiteaduse instituut - Lõputööderegister: Automaattestide loomine andmebaasile Edu,“ 2018. Saadaval: https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=61981&year=0. [Kasutatud 12 jaanuar 2019].

[6] D. Shah, „By The Numbers: MOOCs in 2018,“ Class Central, 11 detsember 2018. Saadaval: https://www.classcentral.com/report/mooc-stats-2018/. [Kasutatud 9 mai 2019].

[7] W. Admiraal, B. Huisman ja O. Pilli, „Assessment in Massive Open Online Courses,“ *Electronic Journal of e-Learning,* pp. 207-216, aprill 2015.

[8] C. García-Martínez, R. Cerezo, M. Bermúdez ja C. Romero, „Improving essay peer grading accuracy in massive open onlinecourses using personalized weights from student's engagementand performance,“ *Journal of Computer Assisted Learning,* kd. 35, nr 1, pp. 110-120, 2019.

[9] H. K. Suen, „Peer Assessment for Massive Open Online Courses (MOOCs),“ *International Review of Research in Open and Distance Learning,* pp. 312-327, juuli 2014.

[10] M. Formanek, M. C. Wenger, S. R. Buxner, C. D. Impey ja T. Sonam, „Insights about large-scale online peer assessment from an analysis of an astronomy MOOC,“ *Computers & Education,* pp. 243-262, oktoober 2017.

[11] N. Kalogeropoulos, I. Tzigounakis, E. A. Pavlatou ja A. G. Boudouvis, „Computer-based assessment of student performance in programing courses,“ *Computer Applications in Engineering Education,* pp. 671-683, detsember 2013.

[12] P. Duch ja T. Jaworski, „Dante - Automated Assessments Tool for Students' Programming Assignments,“ *11th International Conference on Human System Interaction (HSI),* pp. 162 - 168, 2018.

[13] A. Bey, P. Jermann ja P. Dillenbourg, „A Comparison between Two Automatic Assessment Approaches for Programming : An Empirical Study on MOOCs,“ *Journal of Educational Technology & Society,* pp. 259-272, 2018.

[14] Tartu Ülikool, „Moodle: Programmeerimine (LTAT.03.001),“ 2018. Saadaval: https://moodle.ut.ee/enrol/index.php?id=500. [Kasutatud 9 mai 2019].

[15] Tartu Ülikool, „Moodle: Objektorienteeritud programmeerimine (MTAT.03.130),“ Tartu Ülikool, 2017. Saadaval: https://moodle.ut.ee/course/view.php?id=142. [Kasutatud 8 mai 2019].

[16] Tartu Ülikool, „Moodle - Automaadid, keeled ja translaatorid (LTAT.03.006),“ 2019. Saadaval: https://moodle.ut.ee/enrol/index.php?id=2479. [Kasutatud 9 mai 2019].

[17] N. Vlatko, „JAXenter - News, Articles, Code.,“ JAXenter, 6 mai 2015. Saadaval: https://jaxenter.com/oracle-announces-java-7-end-of-life-116960.html. [Kasutatud 12 jaanuar 2019].

[18] Tartu Ülikool, „ÕIS2: LTAT.03.004 - Andmebaasid (6 EAP),“ Tartu Ülikool, 2019. Saadaval: https://ois2.ut.ee/#/courses/LTAT.03.004/version/lt-2019-spring-fulltime-c6/details. [Kasutatud 7 mai 2019].

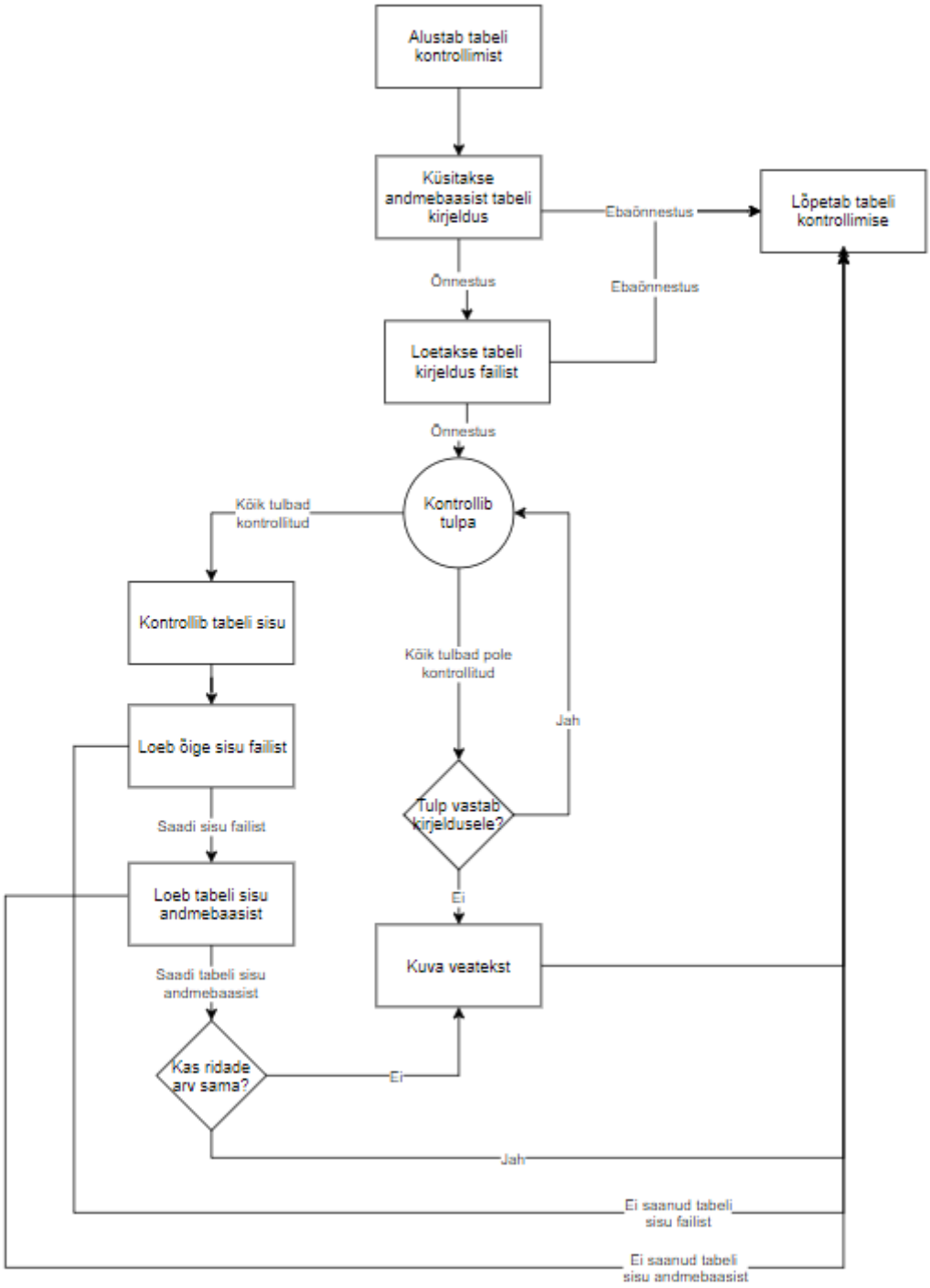
[19] Tartu Ülikool, „ÕIS2: MTAT.03.105 - Sissejuhatus andmebaasidesse (3 EAP),“ 2018. Saadaval: https://ois2.ut.ee/#/courses/MTAT.03.105/version/lt-2018-autumn-openuniv-c3/details. [Kasutatud 9 mai 2019].

[20] P. Luik ja L. Feklistova, „Moodle: SISSEJUHATUS ANDMEBAASIDESSE (MTAT.03.105), SÕ,“ 2018. Saadaval: https://moodle.ut.ee/course/view.php?id=4068. [Kasutatud 7 mai 2019].

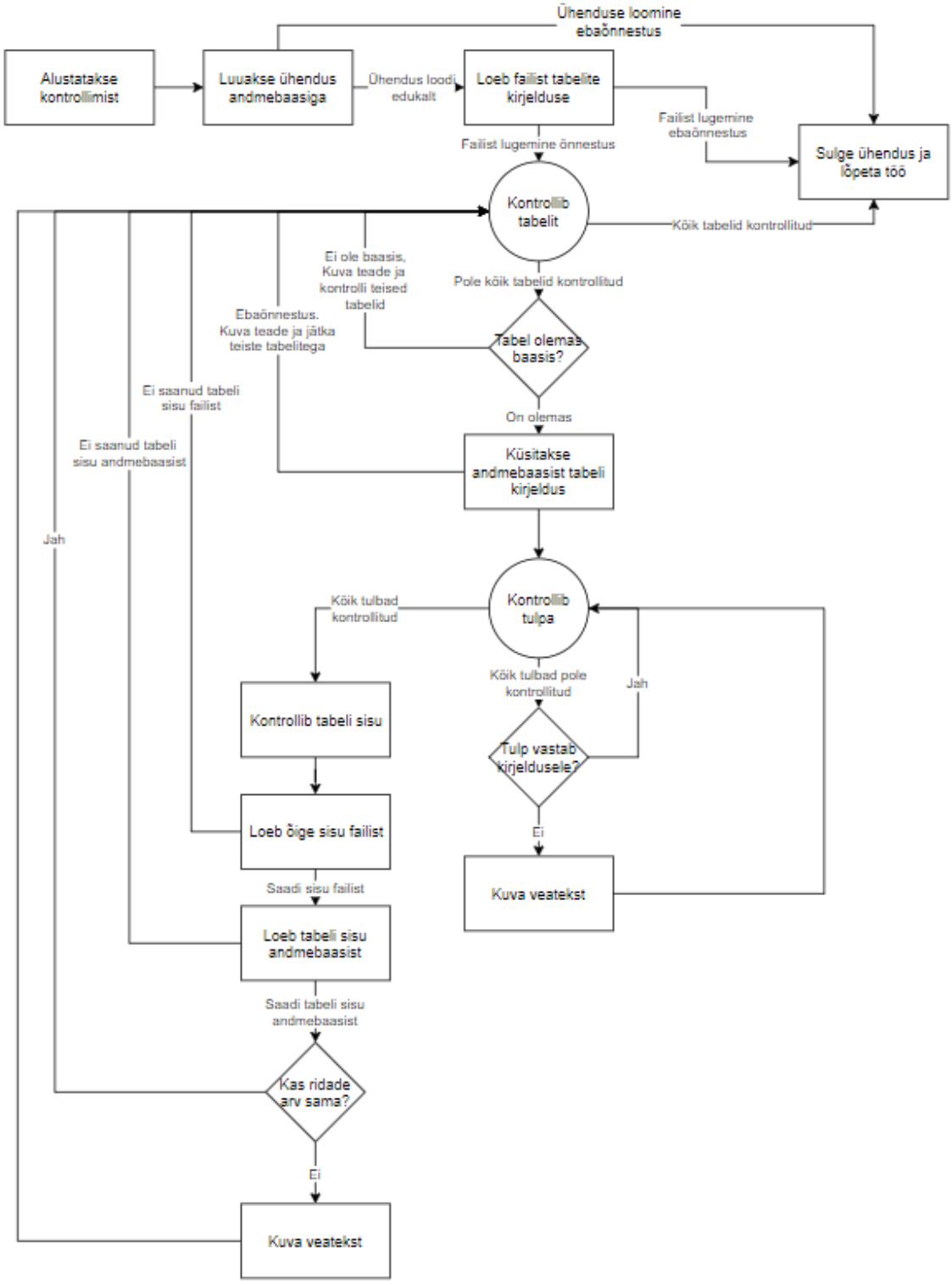
[21] Yatin, „Java Code Geeks,“ 25 juuli 2017. Saadaval: https://examples.javacodegeeks.com/enterprise-java/sql-enterprise-java/jdbc-best-practices-tutorial/. [Kasutatud 9 mai 2019].

Lisad

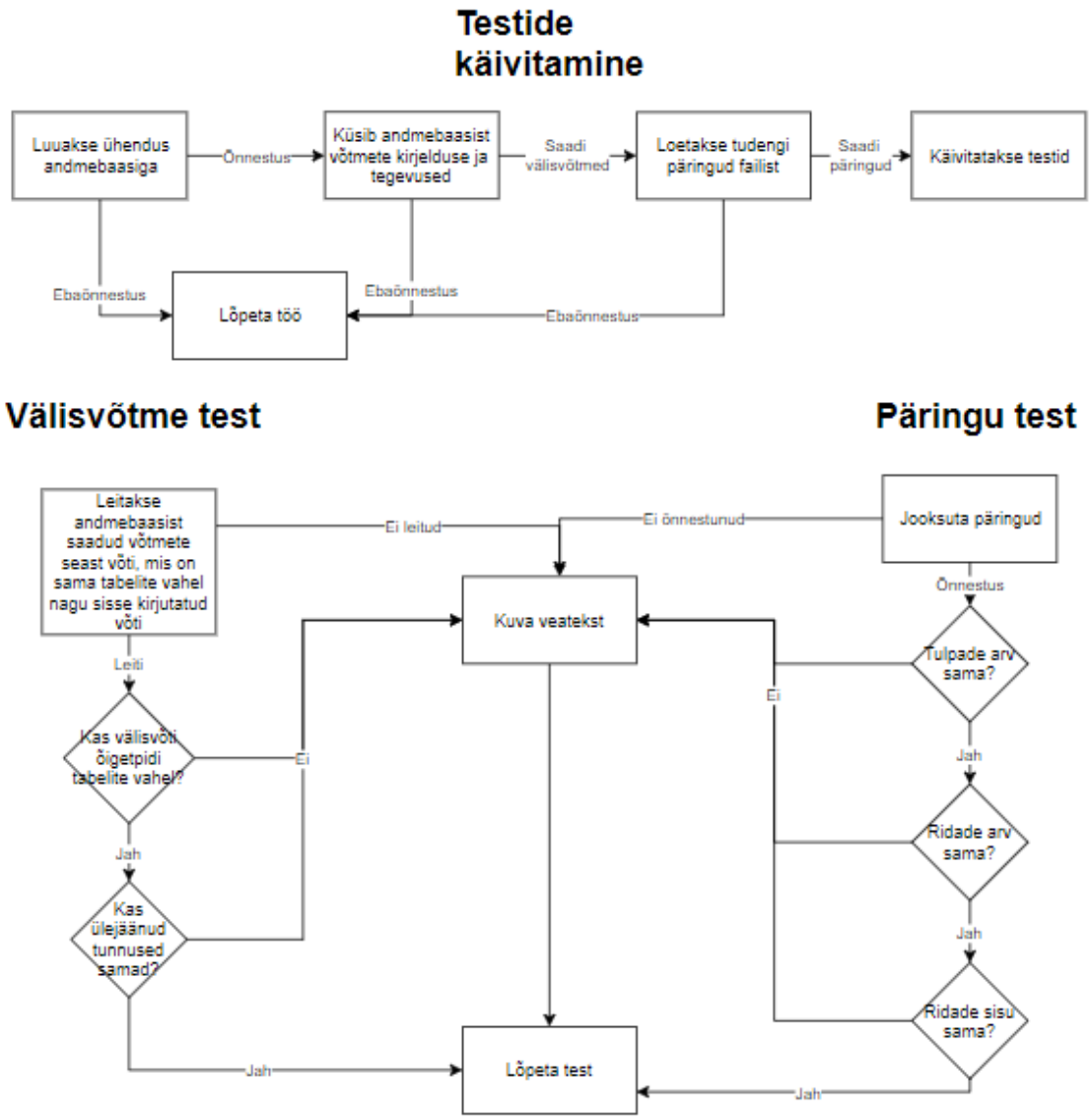
Lisa 1 Tabelite kontrollimise protsess Moodle'i testis



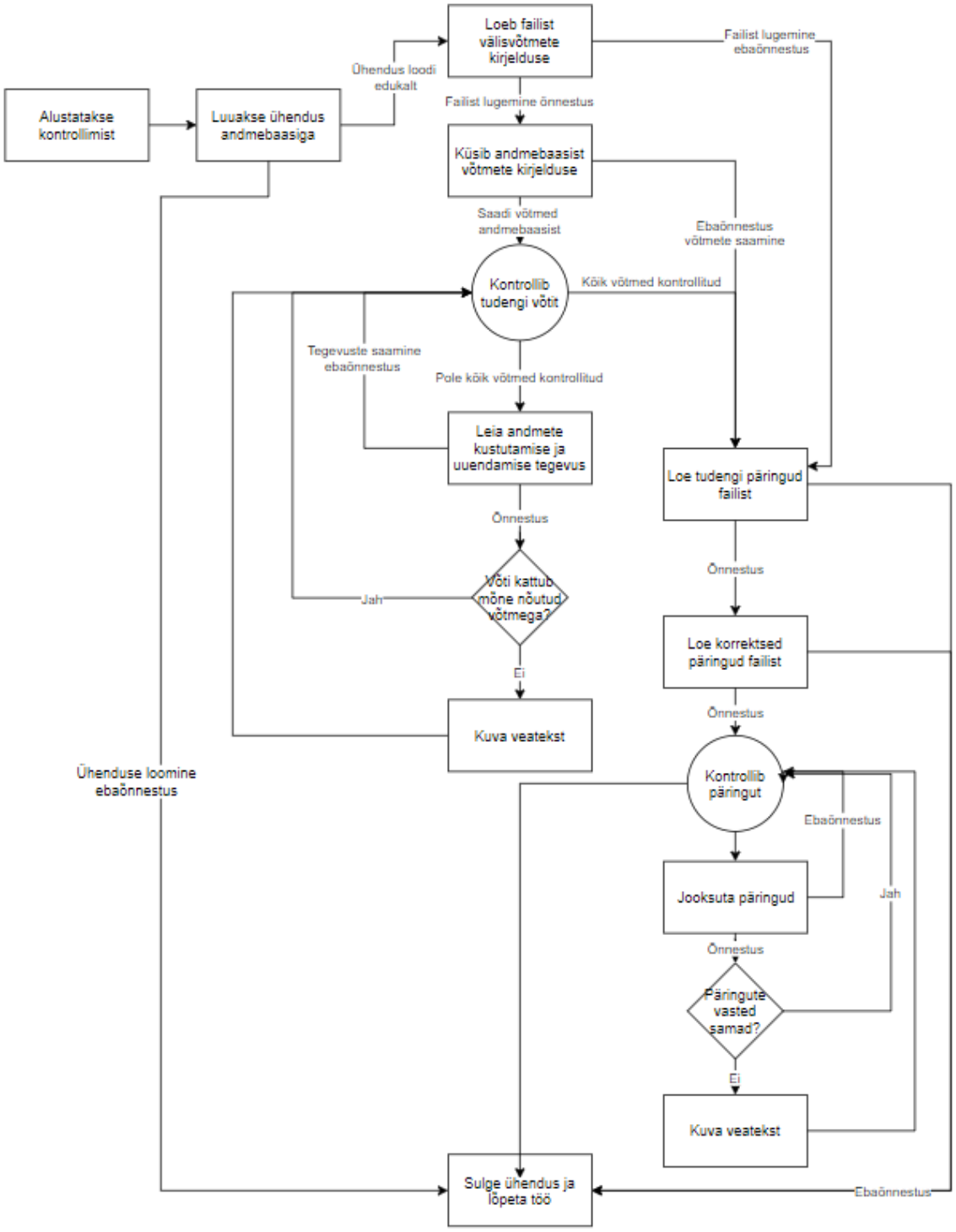
Lisa 2 Õppejõu programmis tabelite testimise tööprotsess



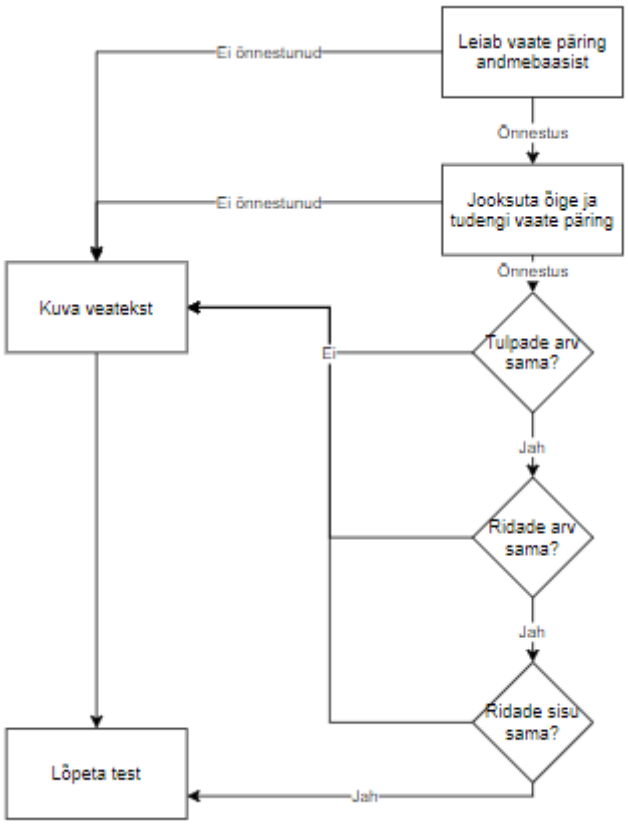
Lisa 3 Välisvõtmete ja päringute testimise protsess Moodle'i testides



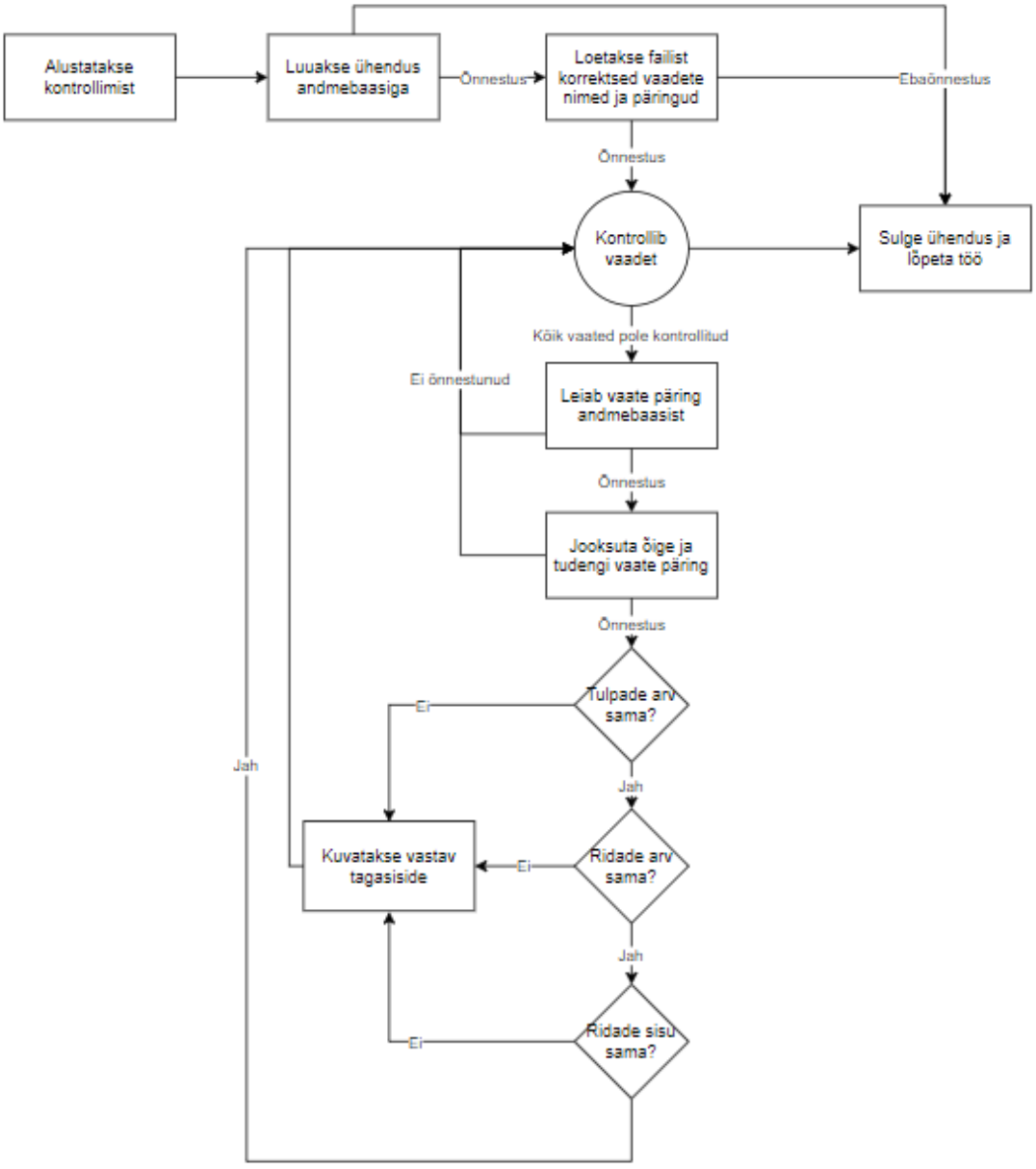
Lisa 4 Välisvõtmete ja päringute testimise protsess õppejõu programmis



Lisa 5 Vaadete kontrollimine Moodle'i testides



Lisa 6 Vaadete kontrollimine õppejõu programmis



Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Mikk Õunmaa**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

   Automaattestide loomine sessioonõppe ainele „Sissejuhatus andmebaasidesse“,

   mille juhendaja on Piret Luik,

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

   1. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

   2. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

   3. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Mikk Õunmaa

10.05.2019