TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Oskar Ander Oja
155338IAIB

Codereview 3.0 programmeerimisülesannete tagasisidestamise keskkonna tarkvarapaketi loomine

Bakalaureusetöö

Juhendaja: Martin Rebane
MSc

Tallinn 2021

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: Oskar Ander Oja

19.05.2021

Annotatsioon

Töös käsitletakse programmi *Codereview* arendust, mis on programmeerimisainetes vajalikuks tööriistaks tudengite tööde hindamisel ning tagasiside andmisel. *Codereview* keskkond võimaldab keskenduda õppetöö käigus loodud koodide loetavusele, jälgida paremini ülesannete lahenduskäike ning anda tudengitele üksikasjalikku tagasisidet.

Codereview keskkonna arendusprojektiga alustas 2017. a. Maria Kert (loodi *Codereview* 1.0) ning seda jätkas 2019. a. Tiit Kuuskmäe (loodi *Codereview* 2.0). Käesoleva töö eesmärgiks oli *Codereview* platvormi töökindluse testimine praktilise õppe- ja õpetamistegevuse käigus, kuivõrd autor praktiseeris 2021. a. kevadsemestril õppejõu abina õppeaines „Tarkvaratehnika“.

Täienduste ning lisanduste tulemusena valmis tarkvarapakett *Codereview* 3.0, mis sisaldab uue lahendusena hindamislehte, samuti täiendavaid funktsionaalsusi, millest peamised on isoleeritud keskkonnas koodi jooksutamine ja automaatne tagasiside laialisaatmise süsteem.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 31 leheküljel, 5 peatükki, 13 joonist, 6 tabelit.

Abstract

Codereview 3.0 Reviewing Environment for Programming Exercises

In this thesis the subject was *Codereview* 2.0 program – an environment for lecturers of informatics to make the process of reviewing and grading programming exercises easier by automated routine tasks. This way the reviewers can focus more on analyzing code readability and structure. Also to help detect any misconceptions that students may have when programming is concerned.

This project is contiuous developement which was started by Maria Kert in 2017 (*Codereview* 1.0). The work was continued by Tiit Kuuskmäe in 2019 which resulted with Codereview 2.0.

In 2021 spring *Codereview* program was tested in the real studying environment. Goals for this work were to make the reviewing process smoother and to develop new features. Main topics were designing and building new frontend webpage „Review“ to make it more user-friendly, developing isolated code running environment and automated feedback delivery process.

In the developing process three new pages were added: review together with management and feedback page, numerous new features were added and whole *Codereview* program was also made more reliable and user-friendly. The outcome is called *Codereview* 3.0.

The thesis is in Estonian and contains 31 pages of text, 5 chapters, 13 figures, 6 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendustarkvara liides
Boolean	Andmetüüp, mis omab väärtusi *true* või *false*
Clipboard	Operatsioonisüsteemi ajutine andmesalvestuspuhver
CSS	*Cascading Style Sheets*, kujundamise märgistuskeel
Dind	*Docker in docker*
In-memory	Andmete hoidmiseks kasutatakse arvuti vahemälu
Integratsiooni testid	*Integration test*, komponentide vahelised testid
JVM	*Java Virtual Machine*, keskkond, mida Java kasutab programmide jooksutamiseks
Klientrakendus	*Frontend*, kasutajaliides
Mobx	Andmehaldus tehnoloogia klientrakenduses
MOSS	*Measure Of Software Similarity*, plagiaadituvastus-süsteem
REST	*Representational state transfer*, tarkvaraarhitektuuri laad
Sandbox	Isoleeritud keskkond
Serverrakendus	*Backend*, server
TDD	*Test driven developement*, testimisel põhinev arendus
UniId	Unikaalne ID, mis on igale tudengile antud ülikooli poolt
VCS	*Version Control System*, versioonihaldamise süsteem
Ühiktestid	*Unit-test*, testid väikseima ühiku kohta
XML	*Extensible Markup Language*, laiendatav märgistuskeel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Programmi koodi kirjutamisel on kaks olulist komponenti: funktsionaalsus ja loetavus. Koodi funktsionaalsust ehk mida kood teeb, kuidas teeb jms, saab kontrollida spetsiaalselt kirjutatud testide kaudu. Loetavust aga samamoodi kontrollida ei saa. See on osa, millele arendajad alati ei pööragi tähelepanu, kuid mängib suurt rolli üldises programmi töökindluses [1]. Martin C. Robert on raamatus „Clean Code“ seda võrrelnud automaailmas masinate hooldatavusega – see on osa, millesse autotootjad on panustanud, et autod oleksid lihtsamini hooldatavad [2]. Samalaadse hooldatava koodi loomist oleks hea võimalikult palju õpetada juba programmeerimisõppes.

Teiseks on programmeerimise juures vaja tegeleda ka tudengite valearusaamade (*misconceptions*) ümberlükkamisega [3]. See oleks palju lihtsam ja efektiivsem, kui sisutühja hinde asemel saab tudeng sisukat ja kiiret tagasisidet. Nii loetavuse kui ka valearusaamadega tegelemiseks pole veel välja töötatud eriti häid tööriistu. Seepärast on tagasiside andmine tudengitele olnud keeruline.

Ajendatuna nendest ning mitmetest teistest põhjustest on loodud *Codereview* keskkond. Eelnevalt on seda projekti lõputöö raames arendanud kaks inimest: Maria Kert teemal „Laiendatava arhitektuuriga hindamiskeskkond programmeerimisainetele“ ja Tiit Kuuskmäe teemal „Programmeerimisainete hindamiskeskkonna C*odereview.ee* 2.0 loomine“ [4] [5]. Käesoleva töö raames luuakse uus tarkvarapakett *Codereview* 3.0.

2021. a. kevadel kasutati Tarkvaratehnika õppeaines esimest korda *Codereview.ee* keskkonda, eesmärgiga hinnata ja anda tagasisidet iseseisvatele töödele. Praktilise töö käigus soovib autor luua uue hindamisliidese, mis oleks kasutajasõbralik ja lihtsalt hallatav. Lisaks võtta käsile tõsisemad puudused, mille kõrvaldamise järgselt keskkonda töökindlamaks muudetakse.

Niisiis on lõputööle seatud kaks eemärki:

- Luua hindamisliides, mis oleks kasutajasõbralik ja lihtsalt hallatav
- Toetada ning arendada *Codereview.ee* keskkonda, et sellega saaks õppeaine korrektselt läbi viidud

Rakenduse repositooriumide jaotus ning asukohad on toodud allolevas tabelis.

Tabel 1. Repositooriumide asukohad

Klientrakendus	https://github.com/tkuuskmae/codereview
Serverrakendus	https://gitlab.cs.ttu.ee/Martin.Rebane1/gradingreview

2 Taust ja probleemi analüüs

2.1 *Codereview* areng ja ülevaade olemasolevast

Koodi loetavuse kontrollimine on enamjaolt ajakulukas manuaalne tegevus ning senini ei ole leitud paremat lahendust. Esimene versioon *Codereview* platvormist valmis Maria Kerdil lõputööga 2017. aastal. Eesmärgiks oli luua automatiseeritud platvorm koodi disaini ja arhitektuuri hindamiseks [4]. Vaatamata mõningatele puudustele eesmärk ka saavutati. Edaspidi viitan Maria Kerdi tööle nimega *Codereview* 1.0.

Edasi asus projektiga tõsisemalt tegelema Tiit Kuuskmäe, kes lõputöö käigus seda edasi arendas. Keskkonnale loodi kaasaegse *ReactJS*-i raamistiku abil uus klientrakendus. Lisaks tehti *Codereview* 1.0 serverrakenduses refaktooring ja loodi juurde uusi funktsionaalsusi: kataloogipuu kuvamine, uus koodi versioonide ja kommentaaride haldamisteenus. Töö valmis 2019. a. suveks.

Lõputöö kaitsmisest minupoolse projekti ülevõtmiseni 2021. a. alguses oli Tiit Kuuskmäe jõudnud programmi nii palju edasi arendada, et toode oli valmis esimeseks reaalseks testimiseks õppeaine raames. Lõputöö kaitsmise järgsel ajal oli Tiit Kuuskmäe kirjutanud erinevaid ühik- ja integratsiooniteste nii serverrakenduses kui ka klientrakenduses. Dokumenteerimise osas oli lisatud *Swagger*. Täiendavalt oli klientrakendusse integreeritud *Mobx*-i keskne andmehaldussüsteem ja selle peale olid üleviidud kõik API-d. Edaspidi viitan Tiit Kuuskmäe tööle nimega *Codereview* 2.0.

2.2 *Codereview* ülesehitus

Järgnevalt kirjeldan lühidalt *Codereview* 1.0 ja 2.0 tööskeemi.

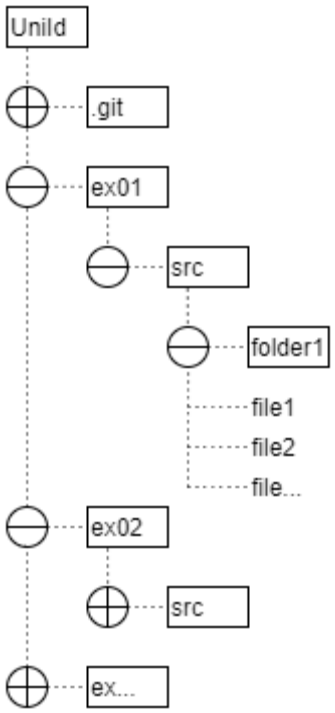
2.2.1 Serverrakendus

Codereview keskkonna serverrakendus on loodud, kasutades *Java Spring Boot* raamistikku. Lihtsustatult on kolm olemit: ülesanne (*Task*), harjutus (*StudentTask*) ja esitus (*Submission*), mille toel rakendus töötab. Lühidalt tõmmatakse serverisse tudengite

repositooriumid ning nende põhjal luuakse harjutuse olemid. Harjutus seob omavahel ära järgmise info: tudengi *UniId*, punktitulemus, ülesanne, asukoht serveris, esitus(ed) ja põhiesituse.

Edasi loob süsteemi loogika uue esituse versiooni. Esitus on üks-ühele seotud kindla *commit*-iga. Lisaks on esituse olemil testkeskkonna tulemused, *commit*-i täpne asukoht *GitLab*-is ning kontreetse esitusega seotud failide tagasiside ehk *FileReview*, mille kaudu on leitavad kõik kommentaarid.

Selleks, et rakendus saaks esitatud töid sisse lugeda, on vajalik alloleval pildil esitatud failistruktuur.



Joonis 1. Nõutud repositooriumi failistruktuur

Repositooriumide haldus failisüsteemis käib läbi *Git*-i. Selle abil saab repositooriumidega teha tegevusi nagu kloonimine (*clone*), uuendamine (*pull*) ja versioonide kontrollimine (*checkout*). *Git*-i ja failisüsteemiga suhtlemiseks on loodud erinevad *bash* skriptid, mida kaasaantud parameetritega käivitatakse. Peale *bash* skriptide on kasutusel ka lihtne *python*-is kirjutatud repositooriumide kloonimisprogramm. Esimesel korral kasutatakse seda kasutajate repositooriumide alla laadimiseks *Gitlab*-ist.

2.2.2 Klientrakendus

Klientrakendus valmis *Codereview* 2.0 projekti raames. See on loodud, kasutades tüübikindlat *Typescript* ja *ReactJS* raamistiku. Kujundamisel on kasutatud *Reactstrap* valmis komponente ja *Bootstrap* valmis CSS-i kujundust. Klientrakendusel on olemas 8 lehte, millest kolm – *Feedback*, *Setup* ja *Review* – sisaldavad keerulisemat äriloogikat.

Hindamisleht *Review* on tööde hindamise läbiviimiseks. Nähes kõiki tudengi esitatud faile saab nendele lisada reapõhiseid kommentaare, mida tudeng saab hiljem tagasiside lehelt lugeda.

Tagasiside leht ehk *Feedback* on mõeldud tudengitele tagasiside nägemiseks. Sellele pääseb ligi kindla lingiga, mille olemasolul näeb tudeng oma esitustele antud tagasisidet.

Järgmisena on välja toodud olulisemad *Codereview* keskkonna tehnilised parameetrid koos versioonidega.

Tabel 2. Serverrakenduses kasutatud tehnoloogiad ja nende versioonid

Parameeter	**Tehnoloogia**	**Versioon**
Programmeerimis keel	Java	Java 11
Java raamistik	Spring boot	2.3.1
Java build tool	Gradle	7.0
Andmebaasimootor	MariaDB	10.3.25
Failisüsteemi muutmine	Bash skriptid	-
Abiskriptid	Python	3.8.5
VCS	Git	2.31.1

Tabel 3. Klientrakenduse arenduses kasutatud põhilised tehnoloogiad ja nende versioonid

Parameeter	**Tehnoloogia**	**Versioon**
JS Raamistik	*ReactJS*	17.0.1
Tüübikindlus	*TypeScript*	4.1.3
Andmehaldus	*Mobx*	6.0.4

2.3 Õpetamispraktika Tarkvaratehnika õppejõu abina 2021 kevad

Lõputöö kirjutamise ajal 2021. a. kevadel avanes autoril võimalus sooritada Tarkvaratehnika aines õppejõu abina õpetamispraktika. See andis hea võimaluse näha esmakordselt kasutatavat *Codereview.ee* keskkonda otse produktsioonis. Nii oli võimalik rakendada tehtud töö tulemusi koheselt ning saada väärtuslikku tagasisidet. Lisaks sai koheselt reageerida erinevatele arendusvigadele ja kiirelt parandusi teha.

Täiendavalt mainin, et samal aastal oli märkimisväärselt suur registreerunud tudengite hulk – peaaegu 90. Varasemalt on see arv olnud enamasti kolmandiku võrra väiksem. Olukorda arvesse võttes panime juhendaja Martin Rebasega paika plaani, kuidas ainet läbi viies saaks paralleelselt toodet arendada ja võimalusel aine raames uuendusi jooksvalt testida.

2.4 Plaan programmi arenduseks

Õpetamispraktika algul selgusid veel mõned olulised puudujäägid, millega tuli arvestada ning lõplik plaan koostati nende lahendamise tähtsuse järjekorras. Järgnevalt toon välja punktid, mida arendusplaanis käsitlen:

- **Tagasiside saatmine tudengile:** ainuke võimalus tagasiside nägemiseks oli läbi kindla lingi ja ainuke lahendus lingi tudengitele toimetamiseks oli käsitsi e-maili teel. Kuna aine peale tuli kokku ligi 90 tudengit, siis oli vaja see lahendus kiiresti automatiseerida.

- **Hinnete ja tagasiside haldamine:** kui hinded on pandud, siis oleks liialt vaevanõudev töid ükshaaval välja saatmiseks või hinnete Moodle'sse kandmiseks üle käia, eriti kui tööde hulgad on suuremad. *Codereview* 2.0 versioonis selline lahendus puudus.

- **Hindamislehe (*Review*) kasutajamugavus:** *Codereview* 2.0 hindamislehe kujundus oli väga ebapraktiline. Enne Tarkvaratehnika aines rakendamist oli lehte testitud 2-3 test-tööga. Kui esimese kodutöö harjutused ligi 90 tudengi poolt sisse loeti, oli nende hindamine oluliselt raskendatud. Selle juures mängisid olulist rolli järgmised punktid:

    1. Kuna sisu ei mahtunud ekraanile ära, oli väga palju üles-alla kerimist.

2. Üleliigselt palju üksikuid ja korduvaid vajutusi, mille tegemata jätmisel süsteem enam korrektselt ei töötanud.

3. Puudusid visuaalsed viited, mis on hetkel valitud ehk aktiivsed elemendid. Tuli alati meelde jätta, keda hindad. Kuna *uniId*-d on tihtipeale lühikesed ja moodustavad arusaamatu tähtedejada, siis see tõi kaasa palju segadust ning pidi iga kord veenduma, et keegi kogemata vahele ei jäänud.

- **Plagiaadikontroll:** et tagada õpilaste vahel võrdsus ja õppeaine kvaliteet, siis plagiaadikontrolli tööriista olemasolu lihtsustab märgatavalt õppejõu tööd, kes ülesandeid üle vaatab ja neid hindab. Paraku oli *Codereview* 2.0 refatoorimise ajal moodul katki läinud, seega tol hetkel seda kasutada ei olnud võimalik.

- **Koodi testimine:** algne lahendus oli loodud juba *Codereview* 1.0 ajast – kasutati *Embeddable* lahendust, mis oli loodud Sven Kadaki poolt 2017. a. lõputöös „Embeddabl: veebilehte süstitav koodi jooksutamise platvorm“ [6]. Lähemalt uurides selgus, et kasutatav teenus töötas eraldiseisvalt ehk koodi jooksutamine ei toimunud *Codereview* serveris vaid kasutati *Embeddable* ressursse. Kuna *Embeddable*-i teenus enam saadaval ei olnud, siis koodi testimist vanal moel ei ole võimalik jätkata.

- **Taaskasutatavad kommentaarid:** Tarkvaratehnika õppeaines õpetamispraktikat tehes oli autoril võimalus hinnata tudengite iseseisvaid töid. Isegi kõigi abiliste vahel ära jagades tuli keskmiselt 18 tööd ülesande kohta ning juba selle koguse juures ilmnes, kui palju kasu oleks taaskasutatavatest kommentaaridest. Kui asuda keskkonda laialdasemalt programmeerimisainetes kasutama, siis kommentaaride taaskasutamine annaks kindlasti suure ajalise võidu.

- S**erverrakenduse turvamine:** Tänapäeva maailmas on infoturve üks suuremaid infosüsteemide murekohti ja *Codereview* rakendus ei ole erand. *Codereview* 2.0 versioonis oli turvalisust arendatud klientrakenduses. Serverrakendusel aga ei eksisteerinud ühtegi turvakihti.

  Probleemi teeb aktuaalseks see, et värskelt enne ülevõtmist oli *Codereview* 2.0 serverrakendusele paigaldatud *Swagger*-i API dokumenteerimismoodul ja kui õigest kohast otsida, siis on võimalik saada ligi kogu infole – mis URL-d olemas

on, nende nõutud parameetrid jne. Kuna kogu serverrakendus oli ilma turvakihita, sai igaüks neid päringuid teha.

Situatsioonist tulenevalt koostati kolmeosaline kavand. Esimesena võtab autor käsile turvamise ja tagasiside saatmise automatiseerimise. Edasi arendab välja uued leheküljed – hindamisleht ja haldusleht. Viimases järgus keskendub plagiaadikontrolli ja koodi test-jooksutamise lahenduse taastamisele.

2.5 Eesmärgid

Töö raames on paika pandud kaks eesmärki.

Esimene eesmärk on hindamislehe kasutajamugavuse parandamine. Tulenevalt vana lehe mainitud murekohtadest seadis autor eesmärgi täitmiseks kolm nõuet:

1. Üles-alla kerimine elimineerida
2. Hiirevajutuste arv peab vähenema võrreldes vana lahendusega
3. Kõik aktiivsed elemendid peavad olema eristatavad ülejäänud valikutest

Teine eesmärk on, et Tarkvaratehnika aine saaks kevadel 2021 korrektselt läbi viidud. Nõudeks on, et *Codereview* rakendus suudab täita kõiki ettenähtud ülesandeid ja pole vaja kasutusele võtta alternatiivseid meetodeid. Kuna platvormi pole varasemalt õppetöös testitud, siis siiamaani seda kindlalt väita ei saa.

2.6 Võrdlus sarnaste lahendustega

Algsed aspektid, mille alusel *Codereview* infosüsteem loodi, on endiselt väga aktuaalsed. Tiit Kuuskmäe on need oma töös välja toonud järgmiselt [5]:

1. *repode ükshaaval allalaadimine, avamine ja kompileerimine õppejõu arvutis on ajamahukas;*
2. *õppejõu koodiredaktoris läbivaadatud töödele tagasiside andmist muudab ebamugavamaks tõik, et üliõpilasele tuleks saata e-kiri, milles sisalduvate kommentaaride juures on viidatud kohtadele koodis, mis vajavad parandamist;*

3. *püüdes olukorda lahendada enimlevinud versioonihalduse süsteemide (Github, Bitbucket, GitLab jt) kaudu peaks õppejõud looma esmalt näidisprojekti, mille vastu saaksid üliõpilased teha pull-request-e (vastaselt ei teki loetelu erisustest, mida saaks kommenteerida); see tähendaks omakorda, et kaduma läheks väärtuslik õppetund, mille raames peaksid üliõpilased projekti ise algusest peale püsti panema;*

4. *git-il põhinevates versoonihalduse süsteemides on võimalik commit-i aega võltsida (sättides arvuti kella sobivale hetkele enne töö tähtaega), mis tähendab, et puudub võimalus tuvastada, kas üliõpilane esitas töö õigel ajal või mitte.*

Järgmisena toon välja, kuidas *Codereview* oma funktsionaalsustega nimetatud aspekte täidab:

- Automatiseeritud repositooriumide sisselugemine
- Koodi reapõhine kommenteerimine ja hindamine
- Lihtne ülevaade tudengite töödest ja tagasiside jagamine
- Plagiaadi kontrollimine
- Koodi jooksutamine testkeskkonnas
- Repositooriumide lokaalne haldus

Ärimaailmast võib näiteks tuua platvorme nagu *Phabricator* ja erinevad VCS-d. Näiteks on *Github*-s 2021 a. seisuga võimalik reapõhine kommenteerimine, kui kasutada lahendust, mida Tiit Kuuskmäe on kirjeldanud 3. punktis.

Ka koodi automatiseeritud jooksutamine isoleeritud keskkonnas on küllaltki piiratud valikutega. Leitud tooted – Programiz, Jdoodle, Onlinegdb – toetavad ainul lihtsat ühe-faililist programmi. Kahjuks ei leidnud nimetatud platvormide kohta avalikult ühtegi dokumentatsiooni, kus oleks pakutud automatiseeritud programmi jooksutamise teenust.

Nende põhjal võib öelda, et *Codereview* on ainuke omalaadne toode, mis toetab kõiki nõudeid. Ülaltoodud näidete puhul on tegemist siiski äriliste toodetega s.t. on mõeldud rohkem ärilahenduste arendamisele kui õppeaine läbiviimisele. Lisaks – kuna

Codereview on nii-öelda asutusesisene arendus, siis on soovi korral võimalik toodet edasi arendada.

2.7 Realiseerimise nõuded

Enne töö üleandmist oli Tiit Kuuskmäe jõudnud valmis kirjutatud mitmeid teste: *controller* ja *service* klassidele serverrakenduses ning API *endpoint*-idele klientrakenduses. Samuti oli refaktoorimise tulemusel kood korralikult struktureeritud. Autor seadis eesmärgiks järgida sama struktuuri ning uuele arendusele kirjutada korralikud koodi katvad testid. Lisaks, lähtudes Robert C. Martin raamatust Clean Code viidi sisse ka erinevaid koodi korrastusi [2].

3 Realisatsioon

Ülevõtmise hetkest on autor keskkonnale jätkusuutlikult uuendusi teinud – mai alguse seisuga kokku 320 *commit*-i, 157 klientrakendusele ja 163 serverrakendusele. Arendamisel kasutas autor TDD võtteid ning mõlemas repositooriumis arendati tööd eraldi arendaja harudes (*branch*). Uuenduste produktsiooni viimise puhul liideti (*merge*) kood arendaja harust põhiharusse (*master*). Nii oli uuenduste sisseviimine paremini hallatav.

Järgnevalt on välja toodud arendamisega seotud väljakutsed ning valminud tulemused koos lahenduskäikudega.

3.1 Uus hindamisliides klientrakenduses

Kasutuskõlblikkuse parandamise huvides vajas hindamisleht *Review* uut disaini.

3.1.1 Uue hindamisliidese nõuded

Uue lehe ehitamisel olid järgnevad lähtepunktid:

1. **Kompatksus:** luua kujundus, mis oleks vertikaalselt võimalikult kompaktne – s.t. võimalikult vähe põhjendamatut üles-alla kerimist ning horisontaalselt vaadatuna jääks piisavalt ruumi, et kõik koodiread ära mahuksid koos tudengilisti ja esituse failipuuga.

   Disaini kujundamisel jaotus vaade kaheks tasapinnaks: põhivaade ja abimodaal. Abimodaalile paigutusid ruumimahukad, kuid mitte otseselt vajalikud elemendid. Vajaduspõhisus selgus järgmise kahe küsimuse abil:

   - Kas info või funktsioon on vajalikud töö hindamisel?
   - Kas element on seotud ainult kasutajavaates hetkel avatud üliõpilase tööga?

   Juhul, kui üks neist andis vastuseks „EI“, siis paigutasin elemendi abimodaalile. Erandiks on koodi jooksutamise tulemuste liides, mis asub ka abimodaalil. Liidese

omapärast tulenevalt oli raske leida põhivaates kohta, kus saaks korrektselt kuvada programmi väljundit ja veateateid.

2. **Aktiivsed väljad:** kuivõrd arendustegevus algas puhtalt lehelt, ei olnud hetkeseisu kuvamist eriti keeruline rakendada tänu *ReactJS* raamistiku ja *ReactStrap* elemetidele sisseehitatud funktsionaalsustele.

3. **Sisseehitatud loogika:** kasutajamugavuse tõstmiseks oli eesmärgiks uude lahendusse integreerida võimalikult palju intuitiivset loogikat, mis tagab programmi automaatse toimimise kasutajapoolse sisendita. Järgmises kahes tabelis on esmalt kirjeldatud kasutaja tegevust ja uude lahendusse sisse ehitatud loogikat ning teises tabelis on hiirevajutuste koguarvu võrdlus vana ja uue lahenduse vahel.

Tabel 4. Kasutajamugavuse tõstmiseks sisseehitatud loogika

Kasutaja tegevus	**Sisseehitatud automaatne tegevus**
Uue üliõpilase valimine	Vaikeseadistusena valib rakendus aktiivseks esituseks põhiesituse ja laeb sisse selle esitusega seotud failipuu
Kopeerimisnupule vajutamine	Nupule vajutamise järel salvestatakse väljal olev tekst arvuti *clipboard* mälusse. See annab võimaluse kohe teksti kleepida soovitud asukohta kopeerimisprotseduurita
Tekstivälja loomine kommenteeritavale reale vajutamisel	Kui kommenteerimiseks on uus tekstiväli loodud, siis muudetakse see väli aktiivseks, et saaks kohe kirjutamisega alustada
Ülesannete vahel liikumine	Kui hindamisel on vaja liikuda erinevate ülesannete vahel, et vaadata hinnet, kommentaare vms, jätab rakendus meelde erinevate ülesannete vahelehtedel lahti olnud tudengi andmed, esituse ja faili. Nii ei pea iga kord uuesti valikuid tegema
Otsing	Rakendatud tudengite listi ja failipuu juures. Otsida saab *uniId* või faili nime järgi
Viimaste ja lemmik-kommentaaride modaali avamine	Mõlemad kommentaaride väljad asuvad samas modaalis ja vastavale nupule vajutades avaneb kas viimaste või lemmik-kommentaaride vaade

Kasutaja tegevus	**Sisseehitatud automaatne tegevus**
Uute kommentaaride salvestamine	Kommentaare on võimalik salvestada kahest kohast: otse koodiridade juurest ja viimaste kommentaaride vaatest. Lisaks tuleb teksti sisu automaatselt kaasa, ilma eraldi *copy-paste* protseduurita. See lahendus võimaldab salvestada valmiskirjutatud kommentaare kiirelt ja lihtsalt

Tabel 5. Kasutajamugavuse tõstmiseks sisseehitatud loogika võrdlus vana lahendusega

Kasutaja tegevus	**Hiirevajutused kokku**	
	Vana lahendus	**Uus lahendus**
Uue üliõpilase valimine	4	1
Kopeerimisnupule vajutamine	Puudub	1
Tekstivälja loomine kommenteeritavale reale vajutamisel	2[1]	1
Ülesannete vahel liikumine[2]	5	1
Otsing	puudub	1
Viimaste ja lemmik-kommentaaride avamine	puudub	1
Kommentaari salvestamine lemmikuks	puudub	1 või 2[3]

Üles-alla kerimine praktiliselt elimineeriti. Seda tuleb siiski ette tudengilistis ja juhul, kui koodifailis on ridasid keskmisest rohkem.

Kirjeldatud loogikate väljaarendus oli kohati küllaltki ajakulukas tegevus, mida aitas leevendada *Mobx*-i keskne andmehaldussüsteemi olemasolu.

[1] Sisaldab tekstivälja akiveerimist.

[2] Mõeldud on korduvat valikut kui tudeng, sooritus ja fail jäävad samaks

[3] Kui otsida viimaste kommentaaride hulgast ja sealt salvestada

3.1.2 Taaskasutatavad kommentaarid

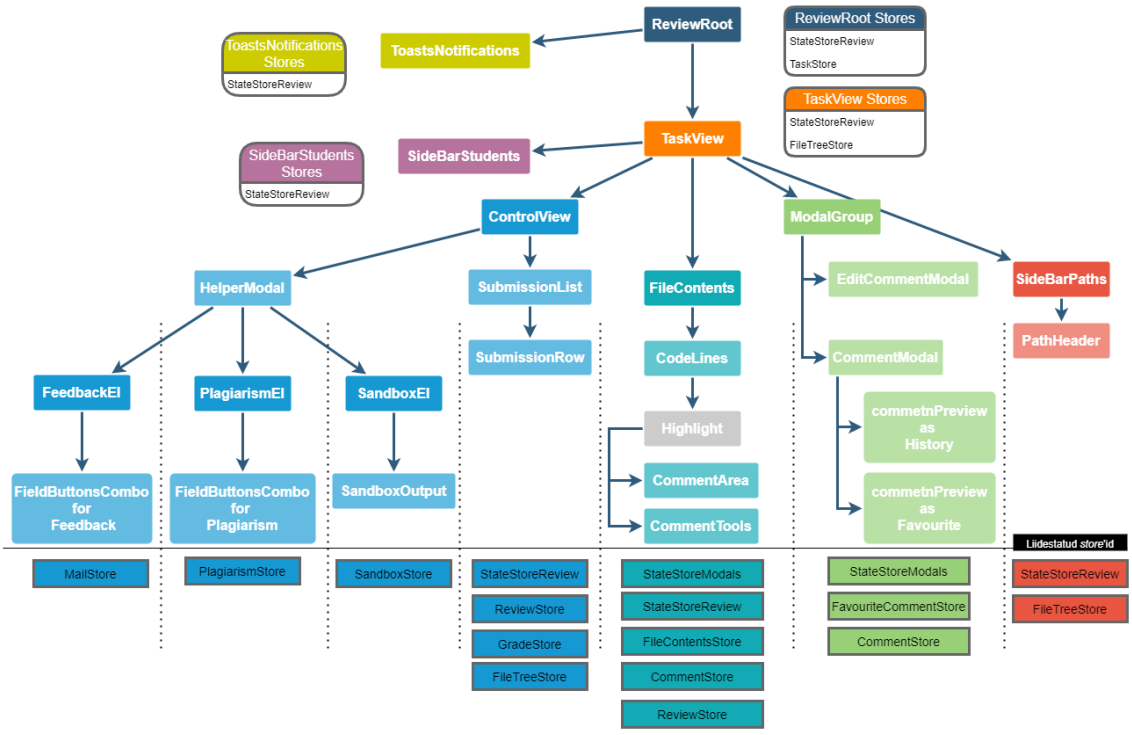
Uue disaini arendamisega paralleelselt loodi juurde taaskasutatavate kommentaaride funktsionaalsus. Rakendatavad nõuded olid järgmised:

- Lemmik-kommentaaride haldus
- Viimaste kirjutatud kommentaaride nägemine

Kuna ühe faili kohta võib olla mitu kommentaari, siis oli ebaefektiivne luua iga välja juurde eraldi staatiline väli. Seepärast lahendati kommentaarid modaali tehnoloogiaga, see võimaldas mobiilsemat lähenemist.

3.1.3 Tulemus

Tulemusena loodi täiesti uue arhitektuurilise põhja ja unikaalse disainiga hindamisleht. Leht on jaotatud 5 süstemaatiliseks osaks. Kokku on valmis tehtud 23 erinevat moodulit, sealhulgas ka kolm modaali.

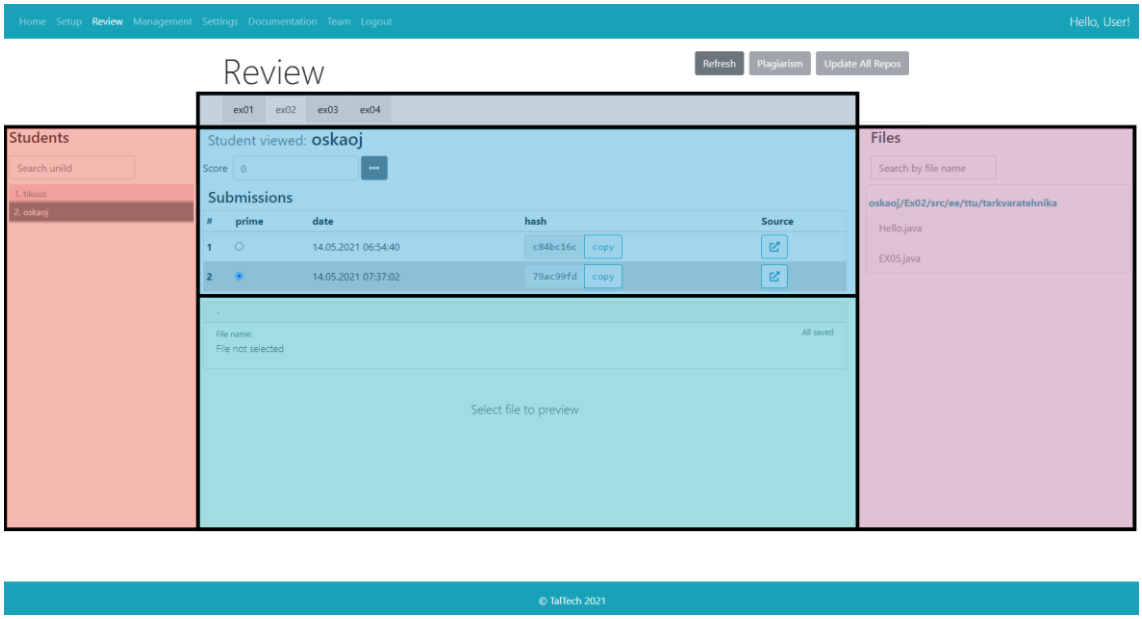


Joonis 2. Hindamisliidese komponentide struktuur. Värvid tähistavad komponentide paiknemist (vt. Joonis 3)

Mäluhalduseks on loodud kaks hoidlat ehk *Store*-i: *StateStoreReview* ja *StateStoreModals*. *StateStoreReview* hoidla hoiab kogu hindamislehega seotud infot. *StateStoreModals* hoidla on loodud modaalide kinni/lahti staatuse hoidmiseks. Modaalide staatuste viimine hoidlasse oli vajalik, et modaalide avamise ja sulgemise kontrollimine rakenduse erinevatest moodulitest oleks lihtne ning kiire.

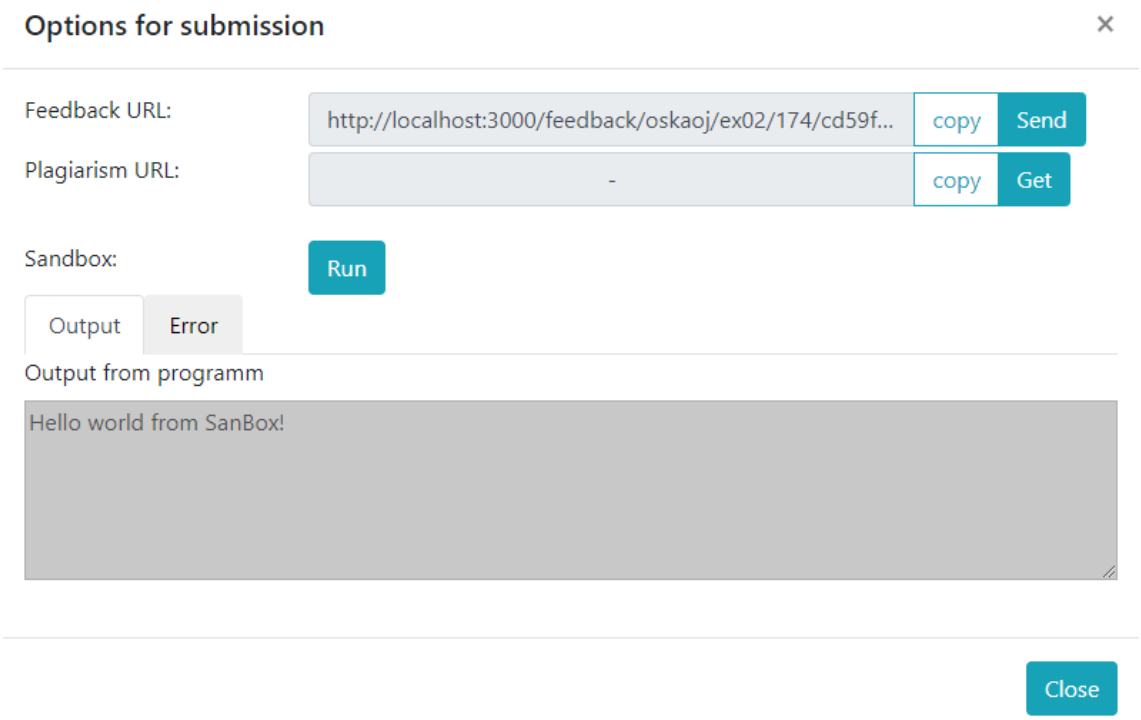
Järgmisel joonisel on kuvatud hindamislehe põhivaade. Komponendid ning funktsionaalsused on järgmised:

1. **Juhtpaneel:** siin saab määrata töö hinnet, avada abimodaali (vt. Joonis 4) ning valida tudengi erinevate esituste vahel
2. **Faili vaade:** kuvab avatud faili sisu (vt. Joonis 5)
3. **Tudengilist:** välja toodud kõik tudengid, kelle töid saab vaadata, sisaldab otsingut *uniId* järgi
4. **Failipuu:** konkreetse esitusega seotud failipuu, millelt saab faile avada, sisaldab otsingut failinime järgi
5. **Vaate kontroll:** saab valida ülesannete vahel, mis on süsteemi poolt sisse loetud



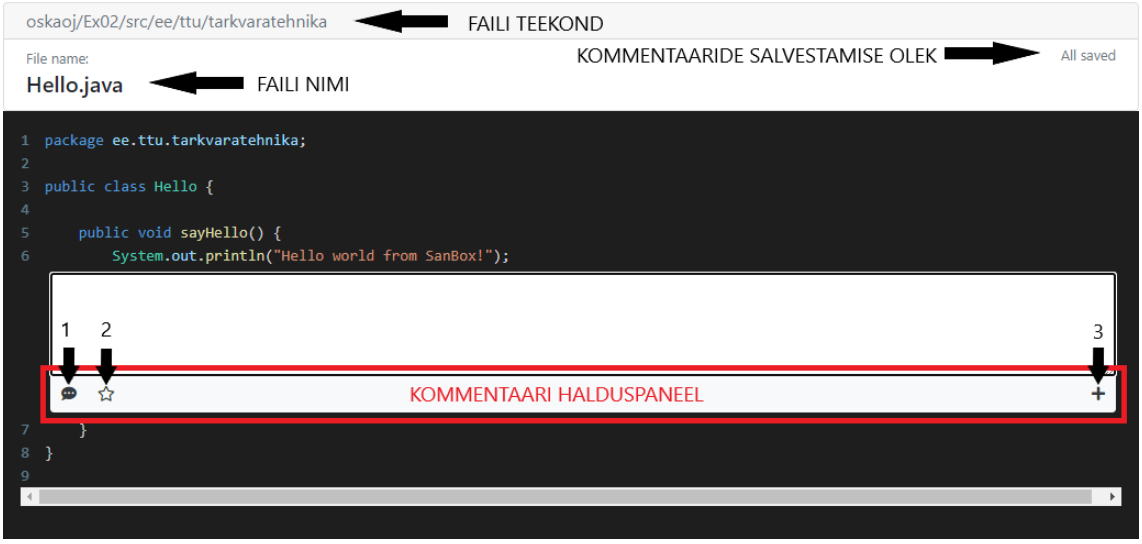
Joonis 3. Kuvatõmmis hindamislehe põhivaatest

Järgmine joonis on abimodaalist, kuhu on paigutatud tagasiside link, plagiaadianalüüsi tulemuste link ja testkeskkonna ehk *sandbox*-i tulemused.



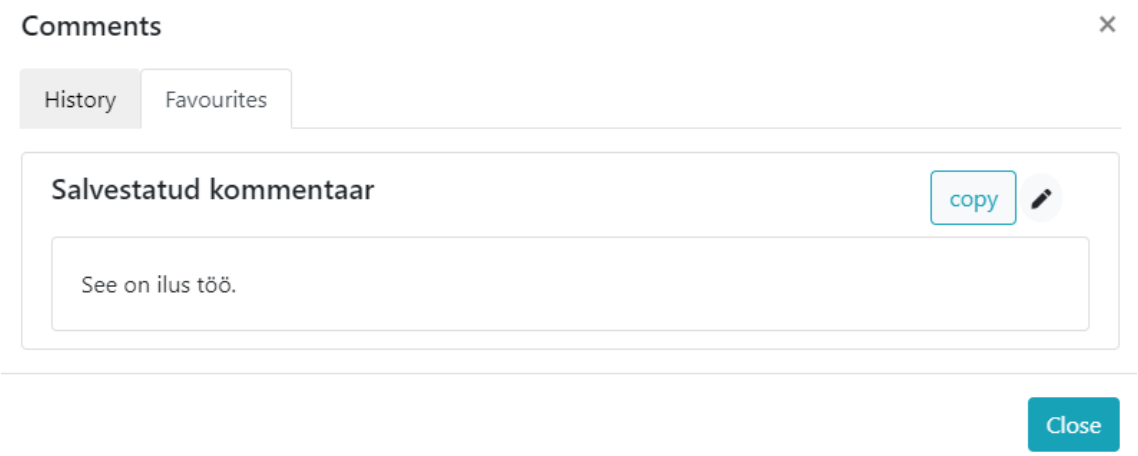
Joonis 4. Kuvatõmmis abimodaalist

Joonisel 5 on kuvatud vaade hinnatava faili sisust, kommentaari kast ja kommentaari halduspaneel.



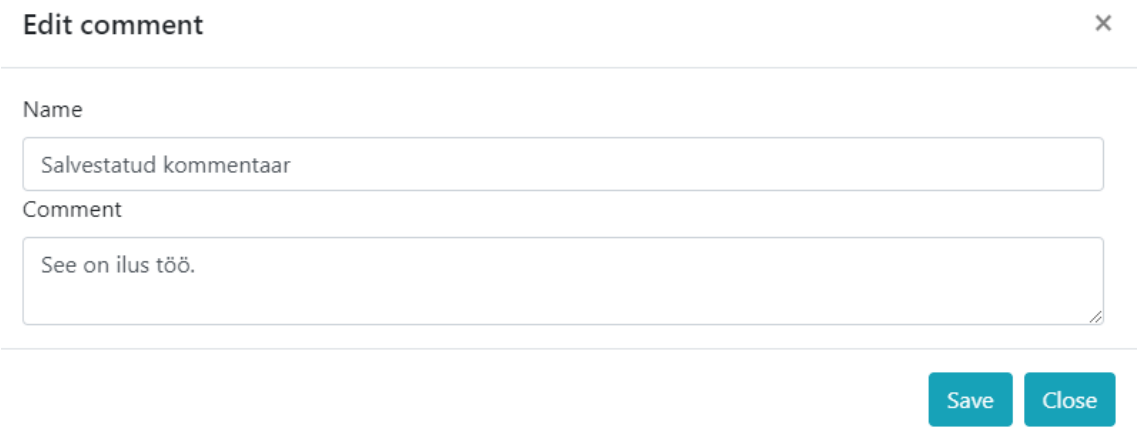
Joonis 5. Kuvatõmmis hindamislehe faili sisu vaatest. 1 – viimased kommentaarid, 2 – lemmik-kommentaarid ja 3 – salvesta kommetaar lemmikutesse.

Kommentaaride modaal on toodud välja Joonis 6. Võimalik on valida nii lemmikute kui viimaste kirjutatud kommentaaride kuva vahel.



Joonis 6. Kuvatõmmis kommentaaride vaate modaalist

Joonisel 7 on toodud kommentaaride muutmismodaal, sama põhja on kasutatud ka kommentaaride esimesel salvestamisel, modaali pealkiri on selle tegevuse puhul muudetud väärtuseks: *Save to favourites*.



Joonis 7. Kuvatõmmis lemmik-kommentaari lisamis-/muutmismodaalist

3.1.4 Uue hindamisliidese analüüs

Käesolevas osas tehakse kokkuvõte tulemustest, kus võrreldakse uue hindamislehe nõuete täituvust.

Üles-alla kerimine: üles-alla kerimine praktiliselt elimineeriti. Kuigi seda tuleb siiski ette, siis ei ole see kuidagi sõltuv lehe ülesehitusest, seega võib selle nõude lugeda täidetuks.

Hiirevajutused: Hiirevajutusi lugedes on arvesse võetud järgmised tegevused (väärtused on välja toodud Tabel 5):

- uue üliõpilase valimine
- tekstivälja loomine kommenteeritavale reale vajutamisel

Tegevused nagu faili avamine ning ülesannete vahel liikumine ei ole sisse arvestatud, sest neid tegevusi oli raske hinnata ja täpseid numbreid ei olnud kahjuks võimalik võtta.

Arvutuste kaudu kirjutatakse esitue kohta keskmiselt 2 kommentaari. Tulemused näitavad, et uuel hindamislehel tuleb teha keskmiselt kaks korda vähem hiirevajutusi, selle tulemusel võib nõude lugeda täidetuks. Hiirevajutuste analüüsiks on tulemuse andmed on välja toodud Tabel 6. Kommentaaride ja hinnatud tööde summad on võetud 2021 aasta kevadsemestri andmebaasist.

Tabel 6. Uue ja vana hindamislehe võrdluse tulemused

Kommentaare kirjutati kokku	338[1]
Hinnatud esitusi kokku	144
Keskmiselt töö kohta kirjutatud kommentaare	~2
Hiirevajutusi vanas versioonis	8
Hiirevajutusi uues versioonis	3
Vana ja uue versiooni vahe kordades	2,6

[1] Ei sisalda tühjasid kommentaare

Aktiivsed väljad: Kõik aktiivsed elemendid on eristatavad ja nõue on täidetud.

Analüüsi põhjal saab väita, et kasutajasõbralikuma hindamislehe loomise eesmärk on täidetud.

3.2 Uue testimiskeskkona (*Sandbox*) loomine

Testimiskeskkond annab võimaluse lihtsalt kontrollida, et tudengi poolt esitatud kood ei sisaldaks kompileerimis- või jooksutamis- ehk *runtime* vigu ning töötab ootuspäraselt. Nagu juba mainitud, siis *Codereview* 1.0 rakenduses oli kasutatud *Embeddable* poolt koodi jooksutamise teenust. Kuna *Embeddable* enam aktiivse rankendusena ei eksisteeri, siis seda lahendust kasutada ei saa. Seetõttu otsustas autor osaliselt integreerida *Embeddable* rakenduse funktsionaalsuse *Codereview* platvormi.

3.2.1 Tehnoloogiate valik

Testimiskeskkonna integreerimisel kasutati järgnevaid tehnoloogiaid:

- ***Docker*[1]:** Originaalselt töötas *Embeddable* keskkond *Linux Containers*[2] (LXC) konteinerite abil, kuid *Codereview* rakenduses otsustati üle minna *Docker* konteinerite lahendusele. *Docker* on rakenduste jooksutamiseks loodud automatiseeritud süsteem, mis kasutab ära LXC funktsionaalust [7]. *Docker*-i kasuks otsustati järgmistel põhjustel:
    - Populaarsus ja laialdane tugi teeb lihtsamaks abi otsimise võimalike probleemide puhul
    - Mobiilsem ja lihtsam haldus, võrreldes LXC konteineritega
    - Jätkusuutlikum lahendus

[1] https://docs.docker.com

[2] https://linuxcontainers.org

- ***Redis*[1]:** kiire ja multifunktsionaalne *in-memory* andmebaas [8]**.** Üks selle toodetest on *pub/sub*, mis aitab erinevatel programmidel suhelda. Kuivõrd *Codereview* võiks üle minna *Docker*-i tehnoloogiale, siis *Redis* tundus autori meelest kõige tulevikukindlam lahendus. Alternatiivsete lahendustena olid olemas *Docker in docker* (DinD), konteinerite käivitamine läbi *socket*-i ja *Rootless Docker* [9] [10]. Kõigi puhul tekkis autoril palju küsimusi, millele vastust ei leitud. Seetõttu *Redis'e pub/sub* kanali kasuks otsustati järgmistel põhjustel:
    - Tulevikus võimaldab *Redis Codereview* keskkonda lihtsamini *Docker* konteineritele üle minna ilma, et peaks testimiskeskkonna lahendust uuesti ümber tegema
    - Lahendust *Redis'e* toel oli kasutatud ka *Embeddable* süsteemis
- ***Python*[2]:** lihtne ja funktsionaalne kõrgtasemeline programmeerimiskeel. Kuna oli otsustatud kasutada *Redis*'t, siis oli vaja luua lihtne rakendus, mis oleks võimeline ühenduma *Redis'e pub/sub* liidesega ja käivitama käsureal *Docker* konteinereid. *Python*-i kasuks otsustati järgmistel põhjustel:
    - Lihtne ning funktsionaalne keel käsurea käskude kirjutamiseks
    - Autoril on olemas varasem kogemus *Python*-i keelega
    - Eelnevalt oli seda kasutatud ka *Embeddable* süsteemis

3.2.2 *Docker*-i jooksutamine

Docker konteineri jooksutamine käib *Docker-manager* mini-programmi kaudu. Käivitamisel antakse lugemisõigustega (*read-only*) kaasa tudengi koodi juurkaust ja käsud, mida konteiner täitma hakkab. Kuna käske on mitu, siis on need kokku aheldatud, et konteiner saaks neid järjest täita, ilma et peaks tööd katkestama.

Embeddable põhifunktsionaalsus oli vaid üksikute failide jooksutamine, seega *Java* koodi käivitamise käsud olid lihtsad. Tarkvaratehnika aines on programmid aga keerulisemad,

[1] https://redis.io

[2] https://www.python.org

esitused koosnevad tihtipeale mitmest failist. Sellest tulenevalt oli vaja modifitseerida nii kompileerimise kui ka jooksutamise käsku. Järgnevalt on lahti seletatud, mida uued käsud teevad.

Kompileerimise käsk ja kaasaantud parameetrite selgitused:

```
javac -d [buildLocation] @[sourcesFile]
```

- **Asukoht (-d *directory*):** sellega on määratud, kuhu kausta masinkoodiks kompileeritud failid salvestatakse. Selleks on staatiline asukoht (*buildLocation*), mis asub *Docker* konteineris. Asukoha määramine oli vajalik kahel põhjustel:

    1. Kui asukoht määrata *Docker*-i failisüsteemi, siis konteineri hävitamisel eemaldatakse protsessi käigus loodud failid automaatselt

    2. Kompileeritud failide jooksutamisel on vaja teada ainult nende asukohta ega pea muretsema, kui tudeng on teekonda modifitseerinud

- **Algfailid (*@[sourcesFile]*):** viitab failile, milles on kirjas kõik „ .java “-lõpuga failid, mis tuleb kompileerida

Kompileeritud koodi jooksutamise käsk ja kaasaantud parameetrite selgitused:

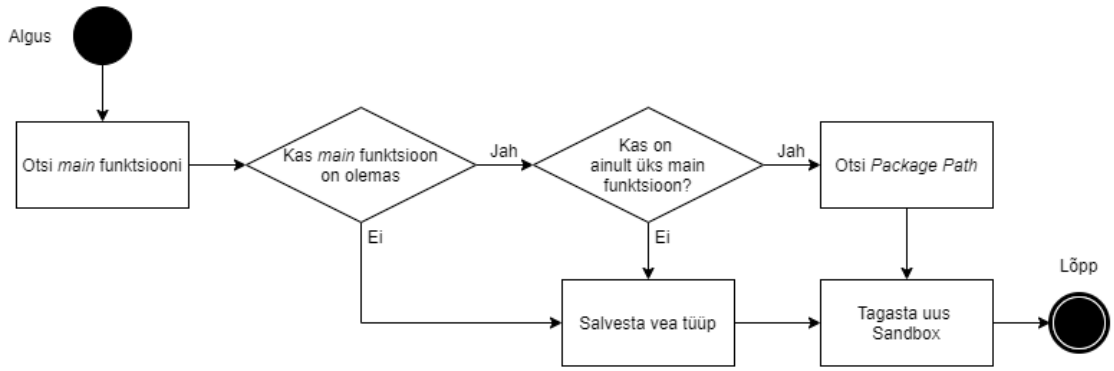
```
java -cp [buildLocation] [mainPackagePath]
```

- **Klassi asukoht (-cp *classpath*):** määrab ära kompileeritud failide asukoha, milleks on sama staatiline asukoht – *buildLocation*, mis sai kompileerimisel määratud

- Klass (*class*): teisisõnu paketi teekond failini (*mainPackagePath*), kus *Java* rakenduse *main*-nimeline funktsioon asub

3.2.3 Tulemus

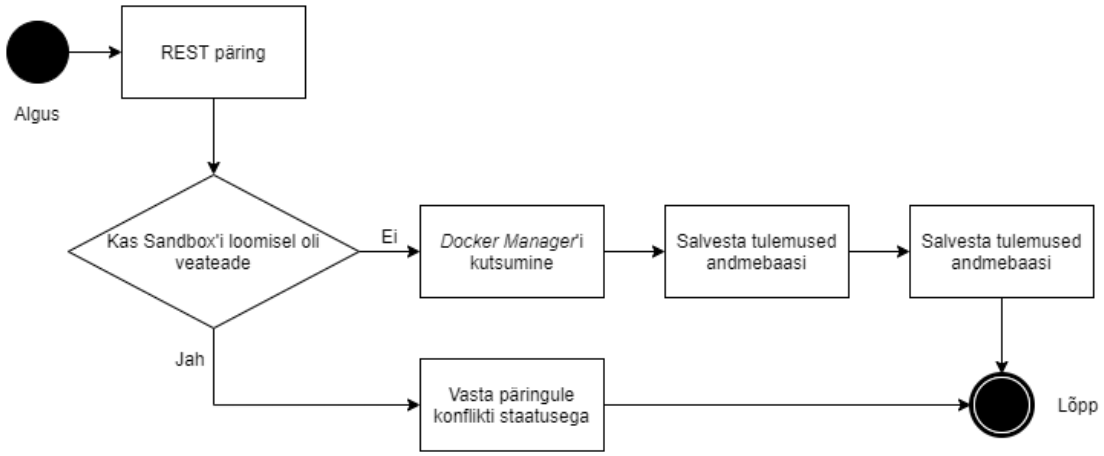
Lähteandmeid testkeskkonnas jooksutamiseks hoitakse *Sandbox*-i olemis, mis loodakse tööde sisselugemisel. Selleks on oluline, et lähteandmed oleksid korrektselt sisse loetud. Programmi sujuvamaks töötamiseks tehti *Sandbox* loomie protsessile erinevaid kohandusi – loodi veakontrollid, parandati algandmetest lähteandmete kättesaamist jms. Joonisel 8 on täiendatud *Sandbox* olemi loomise protsess:



Joonis 8. *Sandbox* olemi loomise protsess

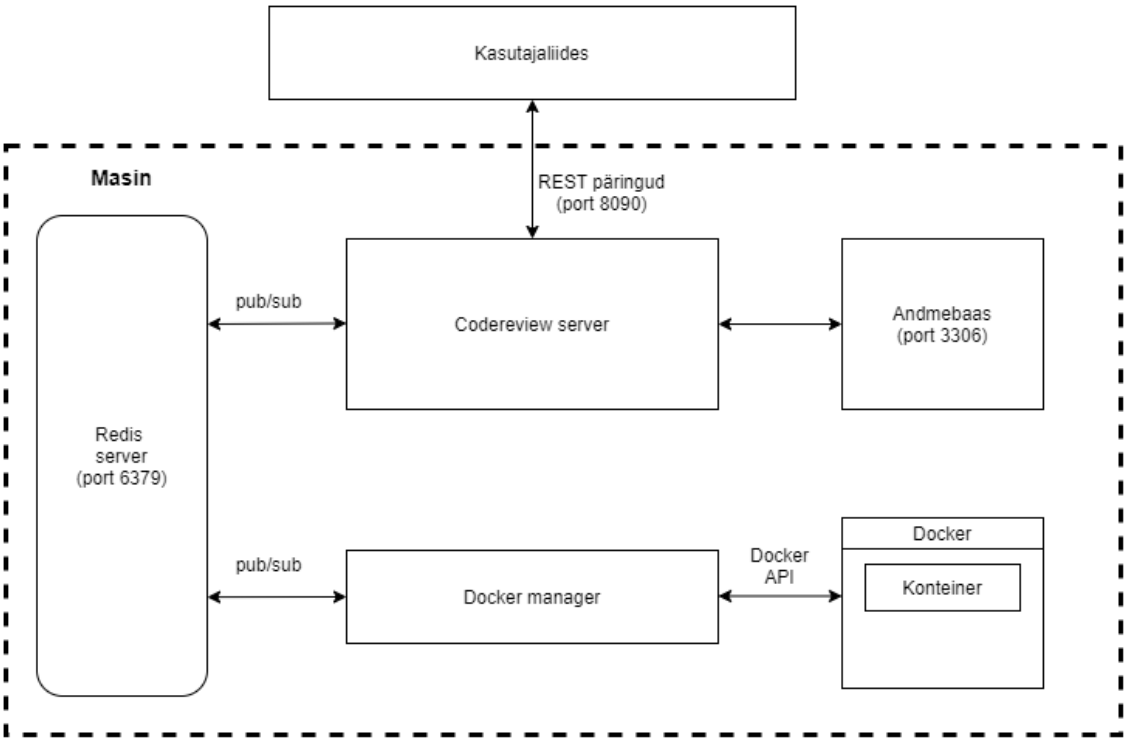
Docker-manager programm on ühenduses *Redis* serveriga, samamoodi on ennast ära ühendanud *Codereview* server. *Codereview* server saadab *Redis*-e kaudu *Docker-manager* programmile jooksutamise käsu koos vajalikke juhistega. Seejärel käivitab *Docker-manager* uue *Docker* konteineri koos vastavate käskudega. Saadud tulemused liiguvad sama teed pidi tagasi, mis *Codereview* server andmebaasi salvestab.

Joonisel 9 on näidatud lihtsustatud skeem, kuidas *Sandbox* teenus töötab.



Joonis 9. *Sandbox* teenuse kasutus

Kasutajal on võimalik protsessi käivitada klientrakendusest, hindamislehe abimodaalilt „Run“ nupust (vt. Joonis 4). Samas kohas on hiljem nähtavad programmikoodi jooksutamise tulemused. Kogu süsteemi arhitektuur on välja toodud Joonisel 10.



Joonis 10. *Sandbox-i* arhitektuur koos *Codereview* serveriga

3.3 Täiendavad arendused

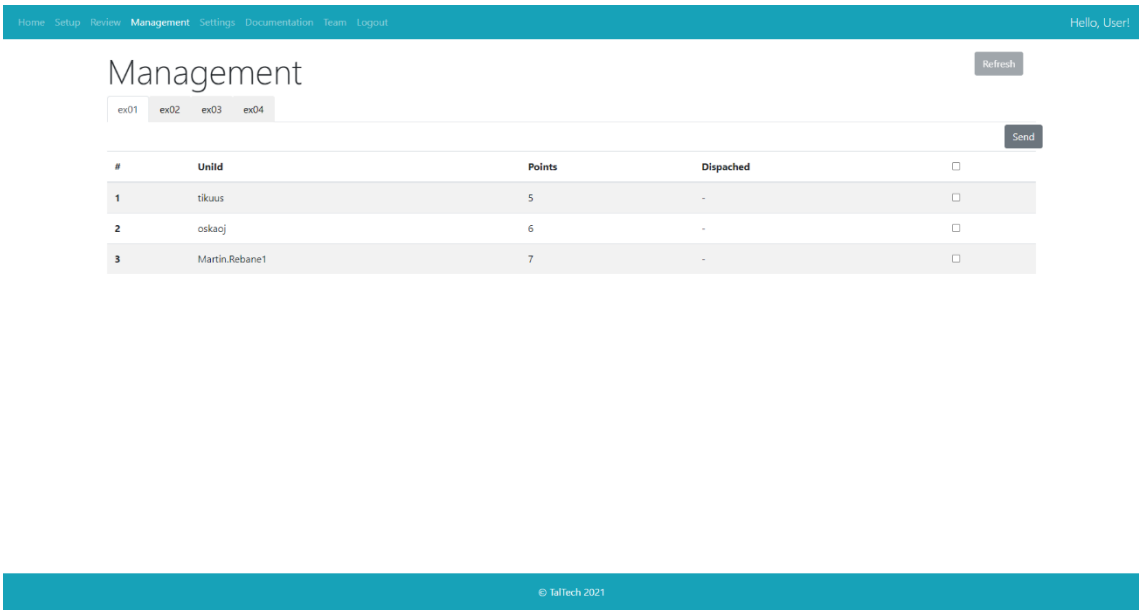
Järgnevalt põgus ülevaade lõputöö käigus valminud arendustest.

3.3.1 Uue halduslehe (*Management*) loomine

Codereview 2.0 versioonis oli arendus nimega *Distribution* loodud vaid kontseptina, kuid kasutada seda ei saanud. Seepärast sai tagasiside linke tudengitele toimetada ainult käsitsi.

Arenduse käigus loodud leht sisaldab juurdeehitatud tagasiside laialisaatmise funktsionaalsust ehk võimaldab välja saata tagasiside linke kõikidele või valitud tudengitele ühe nupuvajutusega.

Uus leht on üles ehitatud, kasutades sarnast kujundust, mis hindamislehel. Lehte disainides oli autoril eesmärk teha võimalikult ülevaatlik ja lihtne lahendus. Haldusleht on jaotatud vahelehtedeks ülesannete järgi ning iga vaheleht kuvab kõikide tudengite töid, väljapandud hindeid ja kuupäeva, millal on tagasiside link saadetud. Joonisel 11 on kuvatud halduslehe vaade.

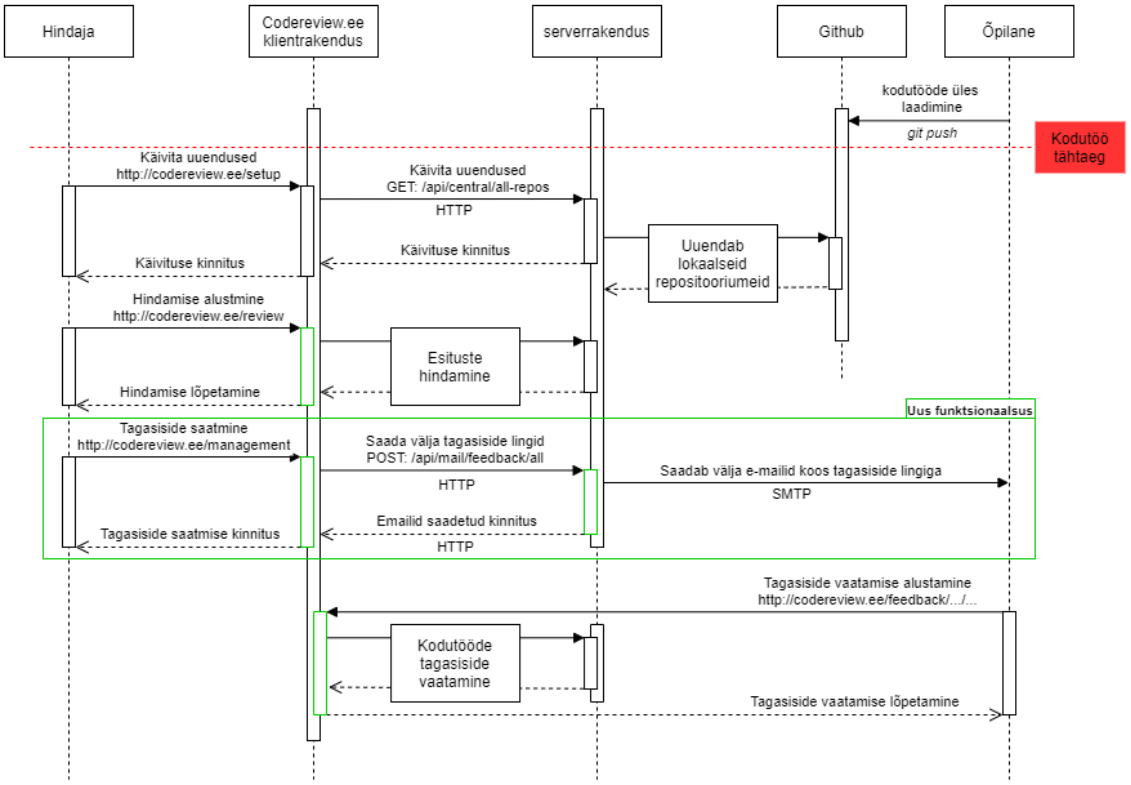


Joonis 11. Kuvatõmmis halduslehe vaatest

Et tagasiside linkide käsitsi väljasaatmisest loobuda, oli serverrakenduses vaja liidestada e-maili saatmise funktsionaalsus. E-maili mooduli valimisel otsustas autor *Spring boot* raamistiku pakutava *spring-boot-starter-mail*[1] lisamooduli kasuks, sest see oli lihtsalt liidestatav ning polnud vaja täiendavaid ettevalmistusi.

Juurdelisatud funktsionaalsus võimaldab nüüd hindamisprotsessi läbi viia nii, et kõik rutiinsed tegevused – tööde sissselaadimine ja jooksutamine, kommentaaride salvestamine reapõhise viitega ning tagasiside väljasaatmine – on automatiseeritud ja hindajal on lihtsam keskenduda parema tagasiside andmisele. Joonisel 12 on välja toodud hindamisprotsess tervikuna ning rohelise värviga on näidatud liidesed, mis on lõputöö käigus juurde ehitatud või ümber tehtud.

[1] https://docs.spring.io/spring-boot/docs/2.1.18.RELEASE/reference/html/boot-features-email.html

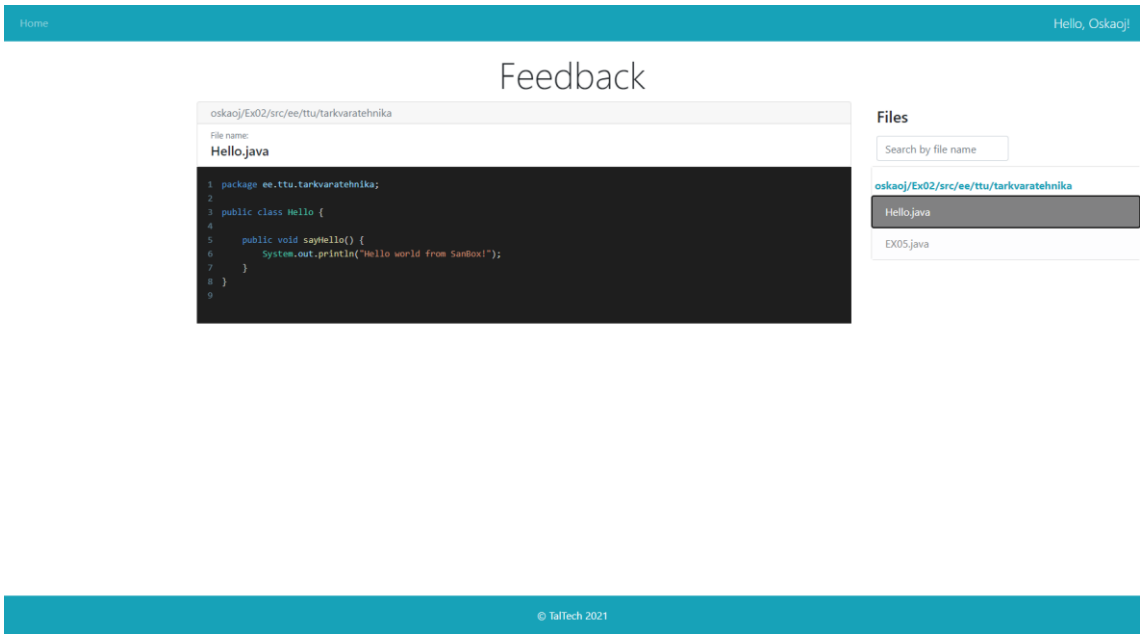


Joonis 12. Tudengi hindamise protsess. Rohelisega on näidatud uued liidesed.

3.3.2 Uus vaade tagasiside lehele

Uue kujundusega leht loodi ka tagasiside vormile, kus tudeng näeb oma esitusele antud kommentaare. Praktika käigus andsid tagasisidet ka tudengid:

- Ei teatud, kus kohast tagasisidet saab vaadata
- Ei õnnestunud kommentaaride laadimine. Probleemi lahendusest on lähemalt juttu alapeatükis 3.4 „Olemasoleva koodi korrastused – Olemasolevate kommentaaride laadimine klientrakenduses“.



Joonis 13. Kuvatõmmis tagasiside vaatest

3.3.3 Logimine

Serverrakenduses hakati rakendama logimist, mis salvestus faili, selle abil sai lihtsamini logisid analüüsida, et vigu tuvastada. Esmalt proovis autor integreerida Log4j2 logimise raamistikku kuna lisaks juhendaja esialgsele soovitusele näis, et paljude testide tulemuste näitel oli see kiirem ja andis paremaid tulemusi, võrreldes konkureeriva lahendusega LOGBack [11].

Log4j2 raamistikuga ei olnud süsteem siiski ühilduv ilma XML faili seadistuseta, seetõttu otsustati alternatiivse variandi *LOGBack* raamistiku kasuks. See variant töötas ja kõik seadistused oli võimalik teha rakenduse kontstantide failis (application.properties).

3.3.4 Kasutaja teavitamine *pop-up* sõnumitega

Klientrakenduses loodi kasutajale tagasiside andmiseks eraldi pop-up süsteem. Nende haldamine käib samuti läbi *Mobx*-i keskse andmehaldussüsteemi hoidla. Tänu sellele pääseb hoidlale hõlpsalt ligi igast moodulist ja saab kasutajat vajaliku infoga kursis hoida.

3.3.5 Plagiaadikontroll

Pärast vea eemaldamist töötas teenus probleemideta, mistõttu kasutatai edasi sama lahendust, mis versioonis *Codereivew* 1.0 ehk plagiaadi avastamise süsteem MOSS[1]. Täiendavaid parandusi tehti ka kontrollitavate failide ettevalmistatava osa kopeerimisfunktsioonile.

3.3.6 Serverrakenduse turvamine

Turvamisel võeti eesmärgiks piirata autoriseerimata ligipääs *Swagger* dokumentatsiooni lehele. Kasutati samuti *Spring boot* raamistiku pakutava *spring-boot-starter-security*[2] lisamoodulit selle ühilduvuse ja lihtsuse pärast.

3.4 Olemasoleva koodi korrastused

Nii klientrakenduses kui serverrakenduses tehti täiendusi ka olemasolevale koodile, mis ei olnud otseselt lõputöö skoobis, kuid olid edaspidise töö huvides vajalikud. Järgnevalt ülevaade olulisematest muudatustest:

- **Olemasolevate kommentaaride laadimine klientrakenduses:** tekkis olukordi, kus 50% tõenäosusega klientrakenduse poolt serverrakendusele tehtud täiendava info päringud olemasolevate kommentaaride laadimiseks vales järjekorras tagasi tulid. Selle tagajärjel tekkis olukord, kus päring, mis kommentaaride sisu küsis, oli vigaselt koostatud ja serverrakendusest vastati veateatega. Kokkuvõttes jäi klientrakenduses kommentaaride mitte ilmumisel mulje, nagu ühtegi kommentaari pole lisatud. See põhjustas segadust tudengite seas, kes tööle antud tagasisidet vaatasid.

  Lahendusena tehti päringud üksteisest sõltuvaks. See garanteeris olukorra, et enne kommentaare küsima ei hakata, kui vajaliku infoga päringud pole tagasi tulnud.

[1] https://theory.stanford.edu/~aiken/moss

[2] https://docs.spring.io/spring-security/site/docs/5.1.6.RELEASE/reference/html/get-spring-security.html

- **Esituste identifitseerimine:** *Codereview* 2.0 versioonis eristati esitusi viimase vastava *commit*-i räsiga. Sama räsi kasutati hinnatud esituse otsimisel, kui tudeng asub talle saadetud lingi kaudu saadud kommentaare vaatama. Antud lahendus aga ei olnud 100% töökindel.

  Testimise käigus avastas autor, et andmebaasi võib tekkida mitu sisendit, mis on identse räsiga. Probleem seisnes selles, et tudengi repositoorium sisaldab erinevaid ülesandeid ja kõik nad on lõpuks seotud sama *commit*-i räsiga. Piisab, kui tudeng teeb ühe väikse täheparanduse mõnes oma vanemas kodutöös ja laeb muudatuse koos uue kodutööga üles. Kuna süsteem võrdleb sisse tulnud infot esituse haaval, siis tulemusena avastatakse ja luuakse kaks uut esitust, mis on sama, uue *commit*-i räsiga, kuid seotud erineva ülesandega.

  Probleemi lahenduseks loodi juurde uus *hash_key* väli. Selle välja väärtuse genereerib *Git* konkreetse esituse kausta alamkaustade ja failide struktuuri põhjal. See lähenemine tagab räsi unikaalsuse ja privaatsuse aspektid. Sarnaselt eelmisele lahendusele ei saa tudengid linki modifitseerides minna võõraid töid vaatama.

  Esituse spetsiifilise räsi saamiseks loodi uus *bash* skript: *generatehash.sh*.

- **Põhiesituse restruktureerimine:** Põhiesituse valimisel oli *Codereview* 2.0 versioonis loodud lahendus, et igal esitusel oli *boolean* väli nimega *isPrime*. Antud lahendusel oli kaks peamist probleemi:

  1. Võib tekkida olukord, kui tudengi kodutööl eksisteerib rohkem kui üks esitus ja mõlemal on märge, et nad on põhiesitused. See kaotab märke tähenduse täielikult. Lisaks pole eelnimetatud vea vältimiseks ühtegi lihtsat lahendust.

  2. Põhiesituse otsingut on keeruline realiseerida, mis omakorda kahjustab rakenduse jõudlust.

  Tulemusena struktuuri muudeti ja harjutuse (*studentTask*) objektile loodi väli *primeSubmission*, mis hoiab endas põhiesituseks seatud olemi id-d.

- **Lisatud UTF-8 toetus:** JVM faili kodeeringu vaikesaedistus on *US-ASCII*. See tekitas probleeme tööde sisselugemisel, mille eestikeelsed failinimed sisaldasid täpitähti. Vale kodeering tähendas seda, et täpitähed muudeti JVM-i poolt loetamatuks sümbolite jadaks ning failisüsteemist otsimisel ei leitud soovitud faili üles. Õige kodeering, mis täpitähti toetab, on UTF-8. Probleemi lahendamisel oli vaja seadistada süsteemi muutujaid (*environment variable*), et serverrakenduse käivitumisel loeks JVM sisse õiged seadistused.

- **Koodi refaktooring:** lähtudes Robert C. Martini raamatust „Clean Code“ viidi läbi erinevaid refaktoorimisi [2]. Olulisemad muudatused olid spetsiifilisemate koodiosade klassidesse eraldamisel. Loodi kaks uut faili: *RepoHandler.java* ja *SandboxHandlerJava.java*. Esimene haldab repositooriumide failistruktuuri sisselugemist, teine fail tegeleb kodutööde sisselugemisel vajalike lähteandmete hankimisega testkeskkonna jooksutamiseks.

- **Esituste versiooni kontrollimine:** *Codereview* 2.0 versioonis valminud *Git*-i põhine versiooni kontrollimine on väga hea lahendus, kuid liiga ressursimahukas. Originaalselt oli see lahendatud nii, et iga operatsioon failisüsteemiga – s.t klientrakendusest iga faili avamine või failipuu pärimine – teostas konstantselt kaks operatsiooni:

    1. Failisüsteemi seisu muutmine esituse versioonile vastavaks (*checkout [versiooni räsi]*)

    2. Seisu seadmine tagasi algolekusse (*checkout master*)

    Autori arvates toodab aga antud lahendus põhjendamatut koormust ainuüksi andmekandjale ning ka süsteemile üldiselt, kui keskkonna kasutus peaks suurenema. Uue lahendusena jäeti alles esimene operatsioon – selle tulemusena muudetakse failisüsteemis seisu ainult siis, kui hetkeversioon erineb soovitud esituse versioonist. Teist operatsiooni täidetakse alles siis, kui otsitakse muudatusi, mis uuendamise käigus tulla võisid.

- **Serveri ümberkolimine:** Ühe töö osana viidi läbi *Codereview.ee* serveri kolimine. Nüüd jookseb server Ubuntu 20.04 operatsioonisüsteemi peal, mis uuendati versioonilt Ubuntu 14.04. Uuendamise põhjus oli tehniline ning samuti

tingitud asjaolust, et Ubuntu 14.04 ametlik avalik tugi[1] lõppes aprill 2019 [12]. Uuendamise käigus sai autor väga hea ettekujutuse *Codereview* tarkvaralistest nõuetest ning lisaks oli mugav teha mõningaid koondamisi failistruktuuris.

[1] End of Standard Support

4 Rakenduse edasiarendus

Codereview.ee keskkond tõestas, et rakendust on võimalik programmeerimise õppeainetes hindamistööriistana kasutada esialgsel kujul ning uus *Codreview* 3.0 tegi kasutamise veel lihtsamaks. Järgnevalt on autor välja toonud mõned mõtteid, kuidas saaks rakendust edasi arendada. Kindlasti sõltub kõik sellest kuidas keskkond tulevikus kasutust leiab, kuid siin on mõned ettepanekud:

- Testimiskeskkond on siiski väga uus ja paljuski testimata lahendus, see vajaks kindlasti täiendamist. Lisaks töökindlusele suurendamisele tasuks luua tugi keeltele – nt: *Python, C, Kotlin* – ning automaattestide jooksutamise funktsionaalsus. Sellisena leiaks keskkond kasutust ka ainetes, mis ei ole *Java* keeles.

- Kuna *Codereview* on õppejõududele hindamistööriistaks, siis oleks hea funktsioon hinnete automaatne üleslaadimine, näiteks Moodle'sse. Kindlasti teeks kasutamise mugavamaks ka see, kui oleks ülikooli keskse *uniId*-ga autentimissüsteem, siis ei pea õppejõududele looma eraldi kontosid.

- Plagiaadikontrolli tulemusi oleks edaspidi mõistlik hakata hoidma lokaalselt, sest piiratud ressursist tulenevalt hoiab teenuse pakkuja tulemusi oma süsteemis maksimaalselt kaks nädalat.

- Repositooriumi massilise kloonimise asemel oleks parem lahendus teha tudengite tööde sisselugemine *hook*-ide põhiseks s.t. et *VCS* nagu näiteks *Gitlab* teavitab *Codereview* rakendust, kui tudeng on oma kodutöö üles laadinud ning seejärel alustab süsteem kohaliku repositooriumi uuendamist. Selle abil on uuendusest võimalik teada saada koheselt ning pole vaja sisse seadistada uuendamist mingi kindla intervalli järel.

- Põnev ja uuenduslik laiendus võiks olla kodutööde masinõppel põhineva hindamise välja arendamine. Selles valdkonnas on viimaste aastate jooksul väga palju arengut olnud ning selle kasutusele võtmine muudaks tagasisde andmise praktiliselt hetkeliseks [13]. Näiteid väljatöötatud algoritmidest [14]. Muuseas saaks tudengi koode ja tulemusi kasutada erinevate analüüside tegemiseks.

5 Kokkuvõte

Lõputööl oli kaks eesmärki. Esimeseks eesmärgiks oli luua uus hindamisliides, mis oleks kasutajamugavam võrreldes vanema versiooniga. Teiseks oli näha kas esmakordselt kasutusele võetud *Codereview.ee* keskkond on praktiliseks õppetööks võimeline.

Lõputöö tulemusena on uus hindamisliides nüüd kasutajasõbralikum – seda kasutades on tudengi tööde hindamine palju lihtsam ja käepärasem. Samuti õnnestus 2021. a. Tarkvatatehnika aine läbiviimine – sai näha *Codereview* platvormi plusse ja miinuseid, parandades ettetulevaid arendusvigu muuta kogu süsteemi sujuvamaks ning luua uusi funktsionaalsusi, mis olid puudu või erinevate arendusetappide käigus katki läinud. Mõlemad eesmärgid said täidetud. Samas töö käigus ülesanne täpsustus ning lõpuks ületas lisandunud ja täidetud eesmärkide keerukus ja maht tunduvalt algseid eesmärke.

Palju tähelepanu pöörati kasutajamugavuse parandamisele, luues erinevaid pisidetaile hindamistöö tegevuse hõlbustmiseks. Tulemusena loodi klientrakendusele kolm uut veebilehte: hindamine, haldus ja tagasiside. Serverrakendusse loodud funktsionaalsustest suuremad on isoleeritud koodi jooksutamine ning automatiseeritud e-mailide saatmine. Samuti tõsteti rakenduse töökindlust koodivigade ehk *bug*-de parandamisega.

Autorile oli projekt heaks väljakutseks arendada infosüsteemi, mis leiab koheselt praktilist kasutust. Tegemisi oli igast valdkonnast – alustades klient- kui ka serverrakenduse arendamise ja kasutajavaate disainimisega ning lõpetades süsteemi administreerimise ja serveri ümberkolimisega. Valdavas osas oli autoril võimalik tugineda varasematele kogemustele, suuremaks väljakutseks osutusid *Docker* konteinerid, logimine ning testide kirjutamine.

Kokkuvõtteks autor kinnitab, et *Codereview* platvorm on suurepärane tööriist õppetegevuse tagasiside korraldamisel. Loodetavasti leiab keskkond ka tulevikus kasutamist ning väärib täiendavat edasiarendamist.

Kasutatud kirjandus

[1] S. Scalabrino, M. Linares-Vásquez, R. Oliveto, D. Poshyvanyk, „A comprehensive model for code readability“, juuni 2018. [Võrgumaterjal] Saadaval: https://onlinelibrary.wiley.com/doi/pdf/10.1002/smr.1958, [Kasutatud: mai 2021].

[2] Robert C. Martin, „Clean Code: A Handbook of Agile Software Craftmanship“, Pearson Education, 2009

[3] Y. Qian, J. Lehman, „Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review“, oktoober 2017. [Võrgumaterjal] Saadaval: https://www.researchgate.net/publication/320679287_Students'_Misconceptions_and_Other_Difficulties_in_Introductory_Programming_A_Literature_Review, [Kasutatud Mai 2021].

[4] M. Kert, „Laiendatava arhitektuuriga hindamiskeskkond programmeerimisainetele“, Tallinn: Taltech 2019. [Bakalaureusetöö] Saadaval: https://digikogu.taltech.ee/et/Item/332f4ee7-680b-4f12-b39a-7d9b37f57994

[5] T. Kuuskmäe, „Programmeerimisainete hindamiskeskkonna Codereview.ee 2.0 loomine“, Tallinn: Taltech 2019. [Diplomitöö]

[6] S. Kadak, „Embeddabl: veebilehte süstitav koodi jooksutamise platvorm“, Tallinn: Taltech 2017. [Bakalaureusetöö] Saadaval: https://digikogu.taltech.ee/et/Item/17ee2dd0-cb48-4c85-a539-92d99ecc69ca

[7] „LXC vs Docker: Why Docker is Better“, 30. aprill 2021, [Võrgumaterjal] Saadaval: https://www.upguard.com/blog/docker-vs-lxc, [Kasutatud: aprill 2021].

[8] P. Gogia, „Redis: What and Why?“, veebruar 2018. [Võrgumaterjal] Saadaval: https://codeburst.io/redis-what-and-why-d52b6829813, [Kasutatud: märts 2021].

[9] J. Petazzoni, „Using Docker-in-Docker for your CI or testing environment? Think twice.“, september 2015, uuendatud . [Võrgumaterjal] Saadaval:, [Kasutatud:].

[10] R. Ramblings, „Exploring Rootless Docker“, detsember 2020. [Võrgumaterjal] Saadaval: https://raesene.github.io/blog/2020/12/19/rootless_docker/, [Kasutatud: mai 2021].

[11] A. Newman, „Benchmarking Java logging frameworks“, 25 oktoober 2017. [Võrgumaterjal] Saadaval: https://www.loggly.com/blog/benchmarking-java-logging-frameworks, [Kasutatud: veebruar 2021].

[12] „Ubuntu: List of releases“. [Võrgumaterjal] Saadaval: https://wiki.ubuntu.com/Releases, [Kasutatud: mai 2021].

[13] M. Vujošević-Janičić, F. Marić, „Regression Verification for Automated Evaluation of Students Programs“, 2020. [Võrgumaterjal] Saadaval: http://www.doiserbia.nb.rs/img/doi/1820-0214/2020/1820-02141900019V.pdf, [Kasutatud: mai 2021].

[14] S. Srikant, V. Aggarwal, „A System to Grade Computer Programming Skills using Machine Learning“, august 2014. [Võrgumaterjal] Saadaval: https://dl.acm.org/doi/pdf/10.1145/2623330.2623377, [Kasutatud: mai 2021].

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

Mina, Oskar Ander Oja

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Codereview 3.0 programmeerimisülesannete tagasisidestamise keskkonna tarkvarapaketi loomine“, mille juhendaja on Martin Rebane
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

19.05.2021

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