TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kauri Kinks 213081IAIB
Artjom Gussev 205962IAIB
Raul Akerman 213093IAIB

LÕPUTÖÖDE VEEBIRAKENDUSE "PROTSESSOR" ÜMBEREHITAMINE

Bakalaureusetöö

Juhendaja: Ago Luberg
PhD

Tellija: Juhan-Peep Ernits
PhD

Tallinn 2024

Abstract

Rebuilding the Thesis Web Application "Protsessor"

The aim of this Bachelor's thesis is to rebuild the current "Protsessor" web application. The "Protsessor" web application is a system that is meant to provide a unified platform for finding and proposing thesis topics for students and teachers alike. The goal of the system is to make it as easy and trouble-free as possible for students and teachers to connect for Batchelor's and Master's theses.[1, 2]The goal of the rebuilding of the system is to achieve the ability to analyze and overview the status of students with their theses, add new or missing functionalities, achieve a more user-friendly interface, and to enable the thesis registration, approval, and submission process to be altered as simply as possible. This is the third thesis dealing with the development of this system, and the system requirements have also been analyzed in a Master's thesis [3], so it is necessary to consider previous successes and failures. The system requirements analysis (chapter 4) is also thoroughly discussed with the project client and supervisor. The website needs to be completely rebuilt, as the "Camunda" process engine used by previous teams did not allow for achieving the desired functionality, making the project's development and management unnecessarily complex, which has led to the conclusion that the project must be redone from scratch.

The thesis is written in Estonian and is 33 pages long, including 8 chapters, 13 figures.

Annotatsioon

Selle bakalaureusetöö eesmärgiks on ümber ehitada praegune ”Protsessor” veebirakendus [1, 2], saavutades võimekus analüüsida ja saada ülevaade tudengite seisust lõputöödega, lisada puuduvaid ja uusi funktsionaalsuseid, saavutada kasutajasõbralikum liides, ning võimaldada võimalikult lihtsalt muuta lõputööde registreerimise, nende kinnitamise ja esitamise protsessi. Tegemist on kolmanda lõputööga, mis tegeleb selle süsteemi arendusega ja süsteemi nõudmisi on ka analüüsitud magistritöös [3], seega tuleb arvesse võtta ka eelmised õnnestumised ja ebaõnnestumised. Süsteemi nõuete analüüs [4] on põhjalikult läbi arutatud ka projekti tellija ning juhendajaga. Veebileht tuleb täies mahus ümberehitada, kuna eelmiste tiimide kasutatud ”Camunda” protsessimootoriga ei olnud võimalik saavutada soovitud funktsionaalsust ning see muutis projekti arenduse ja haldamise kasutult väga keeruliseks, mistõttu on jõutud järeldusele, et tuleb projekt nullist ümber teha.

Lõputöö on kirjutatud Eesti keeles ning sisaldab teksti 33 leheküljel, 8 peatükki, 13 joonist.

Lühendite ja mõistete sõnastik

AD	(Azure) Active Directory
BPMN	Business Process Modelling Notation, Äriprotsesside modelleerimiskeel
CI/CD	Continuous Integration and Continuous Development, pidev integreerimine ja pidev arendus
NPM	Node Package Management, Javascripti paketihaldur
ORM	Objekitdevaheliste suhtete kaardistaja (ingl. k Object Relational Mapper)
QOL	Elu hõlbustav või parandav(ingl. k Quality of life)
SCRUM	Tarkvaraarenduse metoodika
SQL	Structured Query Language, standard keel kasutusel paljudes andmebaasides

Sisukord

Jooniste loetelu

1. Sissejuhatus

Tallinna Tehnikaülikoolis puutuvad lõpuks kõik tudengid kokku sellise aeganõudva, vägagi olulise ja mõnel ka stressirohke ülesandega nagu leida omale sobiv lõputöö teema mis mõjutab kas ja kuidas tundeng oma õpingud lõpetab. Traditsiooniliselt peab tudeng oma potentsiaalse juhendaja ja sobiliku lõputöö teema iseseisvalt leidma. Erinevate teaduskondade vahel varieeruvad lõputööde leidmise ja tegemise protsessid, mis teeb ühtse ja tõhusa keskkonna loomise vajalikuks. Keskkond peab olema arusaadav, võimalikult paindlik ja kasutajalõbralik nii tudengitele kui ka õppejõududele.

Praegune Protsessor rakendus oli loodud IT-teaduskonna tudengite poolt, lõputööde raames, ülal esitatud vajaduste rahuldamiseks, kuid selle kasutajaliides ja funktsionaalsus vajavad ümber tegemist. Eelmiste Protsessor arendusetiimide kasutatud Camunda protsessimootoriga ei olnud võimalik saavutada soovitud funktsionaalsust ning see muudab projekti arenduse ja haldamise kasutult keeruliseks.

Autorite eesmärgiks on ümber ehitada praegune Protsessor veebirakendus, saavutades sellega kasutajasõbralikum liides, võimekus analüüsida ja anda ülevaadet tudengite hetkeseisust ning võimaldada võimalikult lihtsalt hallata lõputööde registreerimise, nende kinnitamise ja esitamise protsessi. Autorid on pühendunud sellele, et hoida Protsessor projekti sellises korras, mis võimaldaks järgmistel arendustiimidel hõlpsasti koodist aru saada ja süsteemi laiendada. Autorite prioriteediks on ka luua selge ja arusaadav dokumentatsioon, mis katab nii olemasolevaid funktsioone, süsteemi arhitektuuri, taga- ja esirakenduse käimapanekut ja struktuuri ning juhiseid uute funktsionaalsuste implementeerimiseks.

2. Protsessori ajalugu

Käesolevas peatükis käsitletakse kahte eelmist Protsessori teemalist lõputööd ja hiljutisi Protsessoriga sarnaseid või seotud lõputöid, nende eesmärke ja tulemusi.

2.1 Esimene Protsessori lõputöö

Esimene Protsessori lõputöö valmis aastal 2022 kevadsemestri lõpuks bakalaureuse lõputöö raames. Selle eesmärk oli luua veebirakendus projektide haldamiseks, läbimisprotsessi jälgimiseks ja nende teemade leidmiseks, haldamiseks ning neile kandideerimiseks. Selle lõputöö suureks osaks oli protsessimootori kasutamine, kuna see võimaldaks defineerida erinevate õppekavade lõputöö leidmise ja tegemise protsessi BPMN skeemiga kui ka kasutajate jälgimist skeemi erinevates etappides. Protsessimootoriks valiti Camunda[4], mis on loodud Activiti[5] põhjal ja oli tollel ajal avatud lähtekoodiga. Selle lõputöö autorite sõnul said seatud eesmärgid täidetud ja nähti loodud rakenduses kasvupotentsiaali. [2]

2.2 Teine protsessori lõputöö

Teine Protsessori lõputöö sai valmis aastal 2023 kevadsemestri lõpuks, samuti bakalaureuse lõputöö raames. Lõputöö eesmärk oli edasi arendada lõputööde teemade rakendust Protsessor [1]. Selle lõputöö raames analüüsiti rakenduse kasutajakogemust, saadud tulemuste põhjal tehti mitmeid olulisi muudatusi, milleks olid kasutajaliidese uuendamine ja uue õigustepõhise rollisüsteemi loomine. Lõputöö käigus täiendati statistikat tudengite ja juhendajate kohta ning lisati parem teavituste süsteem. Eelmises lõputöös [2] kasutatud Camunda lisas palju keerukust rakenduse arendamises autorite sõnul. Lõputöö käigus analüüsisid autorid, kas Camunda kasutamine on õigustatud ning kas selle kasutamist peaks jätkama. Analüüsi tulemusena otsustati Camundaga edasi jätkata.

2.3 Muud sarnased lõputööd

Tehtud on ka lõputöid, mis on kas seotud Protsessoriga või sarnanevad sellega.

2.3.1 Tallinna Tehnikaülikooli lõputööde teemade haldamise ja juhendamiskokkulepete sõlmimise valdkonna äri- ja nõuete analüüs veebirakenduse Protsessor edasiarendamise toetamiseks

Selle magistritöö eesmärk oli viia läbi Tallinna Tehnikaülikooli bakalaureuse- ja magistritaseme lõputööde teemade haldamise, pakkumise, leidmise valdkonna äri- ja nõuete analüüs veebirakenduse Protsessor edasiarendamise toetamiseks. Analüüsi tulemusena valmisid 13 ärireeglit ja 20 funktsionaalset nõudmist mida saaks otse üle anda Protsessori arendajatele. [3]

2.3.2 Lõputöö teemade kosjasobitaja infosüsteem TalTech ülikoolile

Selle magistritöö eesmärgiks oli lihtsustada lõputöö teema valimise ja juhendaja leidmise protsessi TalTech-i õpilaste jaoks. Töö valmis paralleelselt Protsessor lõputööga aastal 2023 ja oli sarnase tulemusega. Seda tööd eristas süsteemi võime arvutada tudengi ja juhendaja sobivusprotsenti, mis aitaks leida tudeng-juhendaja paare. Töö käigus arendatav süsteem valmis vaid osaliselt ning seda kasutusele ei võetud. [6]

2.3.3 Lõputööde haldamise süsteemi projekteerimine, realisatsioon ja juurutamine Tallinna Tehnikaülikooli informaatikainstituudis DHS Amphora baasil

Selle magistritöö raames prooviti luua sarnaste eesmärkidega süsteem aastal 2015 DHS Amphora baasil, töö on kirjutatud eesti keeles. DHS Aphora on aga tasuline ja töö kirjutamise ajal ei olnud veel kasutuses Microsoft Azure AD keskkond ülikooli tundengite ja töötajate majandamiseks. Töö saavutas oma eesmärgid ja oli kasutuses aastal 2015. a kaitsmisperioodil lõputööde haldamiseks informaatikainstituudis. Süsteemi kõikide nõuete katmiseks oli aga vaja edasiarendusi, edasiarendustest informatsiooni ei leitud. [7]

2.3.4 Veebilahendus lõputööde kavandite loomiseks

Selle bakalaureusetöö eesmärgiks oli lahendada tundengite poolt pakutavate lõputöö teemade ebasobivuse probleemi kavandades selleks süsteemi, mis küsib tudengitelt nende teema otstarbekuse kohta ning seeläbi sunnib tudengeid tegema probleemile taustauuringut. Süsteemist valmis prototüüp, milles oli funktsionaalsus osaliselt implementeeritud. Süsteemi kasutusele ei võetud. [8]

2.3.5 Lõputööde teemade haldamise rakendus

Selle magistritöö eesmärk oli viia üliõpilased ning juhendajad omavahel paremini kokku, kui oli antud hetkel võimalik ning anda koht lõputööde hõlpsaks majandamiseks. Lõputöö eesmärk ja funktsionaalsed nõudmised on sarnased Protsessori omadega. Töö saavutas eesmärgid ja põhifunktsionaalsuse. Järgmine samm süsteemi jaoks oleks olnud laialdasem testimine. Süsteemi kasutusele ei võetud. [9]

3. Eelmise süsteemi puudused

Selles peatükis käsitletakse ja analüüsitakse eelmise süsteemi puuduseid ja kasutatud tehnoloogiaid mille tulemusena tekkis vajadus uue rakenduse loomiseks.

3.1 Camunda

Camunda eesmärk Protsessor-is oli võimaldada süsteemil käituda erinevate mustrite alusel kasutades BPMN skeeme ja läbi nende hõlpsasti muuta süsteemi käitumist. Kasutajate seisu jälgimine skeemide erinevates etappides oleks ka automatiseeritud. Süsteemi mugandamine selliste skeemide kasutamiseks on aga keeruline ja skeemide endi loomine osutus liigselt keerukaks. Esinesid paljud raskesti jälgitavad bugid ja süsteemi arendus osutus hoopis keerulisemaks, mitte lihtsamaks nagu oli algne nägemus ja protsessimootori kasutusele võtmise motivaatoriks. Camunda protsessimootoris ei ole klasside vahel kindlat hierarhiat ja seost, mistõttu ka vigade ilmnemisel on väga raske leida, miks või kus viga võib olla, või mis selle vea põhjustas. Samuti on saadaval olev dokumentatsioon küllaltki kasin, mis samuti raskendab selle süsteemi iseseisvat kasutamist, ning Camunda't pakkuv firma on rohkem kesendunud pakkuma Camnuda't kui teenust. Protsessori teise lõputöö Camunda analüüsis oli küll jõutud otsusele sellega jätkata [1], kuid juhendajaga nõu pidades tuli välja, et sellele otusele oli jõutud, kuna Camunda eemaldamine oleks tähendanud sisuliselt uue süsteemi loomist ja senini tehtud töö oleks läinud luhta.

3.1.1 Camunda alternatiivid

Käesoleva töö raames kaaluti samuti Camunda või mõne muu alternatiivse protsessimootori kasutamist. Leitud lahendused olid aga tasulised (sealhulgas Camunda uus variant Camunda 8) või ebasobivad meie kasutusjuhuks. Kaalutud alternatiivid:

- Camunda 8 - Muutus versiooniga 8 tasuliseks
- Huginn - Mõeldud süsteemivälisteks protsessideks (nt. Twitterisse automaatpostitamine, mõne muu saidi webscrapimine)
- Flowable - Vähese dokumentatsiooniga tasuta varjant.

Vaadatud sai ka palju teisi alternatiive kuid need märgiti vähese dokumentatsiooni või aktiivse arenduse lõpu/hangumise pärast ebasobivaks.

3.1.2 Protsessimootori vajalikkus

Süsteeminõuete kompileerimise järel tuldi järeldusele, et parimaks ja kõige paindlikumaks lahenduseks sobib tavaline veebirakendus ilma protsessimootorita, mis majandaks süsteemi allosasid. Võttes arvesse eelmise tiimi raskusi Camunda kasutamise ja arendamisega ning rakenduse hetkeseisu, otsustati luua uus süsteem tavapärasema veebriakenduse näol. See otsus andis ka autoritele vabaduse kasutada kõige otstarbekamaid teeke ja raamistikke.

3.2 Veebirakenduse seis

2024. aasta alguses töös olevas Camunda põhises rakenduses esinesid paljud vead ja põhifunktsionaalsused ei töötanud, mis tähendas sisulist mittekasutatavust.

3.2.1 Koodibaasi seis

Käesoleva lõputöö raames prooviti ka algselt parandada olemasolevat rakendust, et taastada põhifunktsionaalsus. See osutus aga liigselt ajakulukaks, kuna koodibaas oli keerulise ülesehitusega ja esinesid paljud funktsioonid ja klassid, mis ei olnud ilmselgel kasutusel.

3.2.2 Dokumentatsioon

Vana rakenduse dokumentatsioon oli jaotatud mitme erineva GitLabi projekti peale ja tihti osutus arusaamatuks, milline info oli aegunud ja milline asjakohane. Oli loodud ka veebikeskkond, kust peaks kogu dokumentatsiooni kätte saama, kuid see oli vigane ja paljude tähtsate kategooriate peale vajutades vahtis vastu tühi leht. See raskendas oluliselt rakenduse parandamist ja kergemaks osutus koodis sobramine, et leida otsitud funktsionaalsust või tekitada mõnest protsessist ettekujutus.

4. Süsteemi kirjeldus ja nõuded

Süsteemi ülesehitamisel on üheks tähtsaimaks etapiks panna paika nõuded ja funktsionaalsused, mida rakenduse eesmärgipäraseks tööks vaja oleks. Seda etappi hõlbustas autoritel eelneva magistritöö [3] olemasolu ning omapoolne analüüs. Eelmainitud magistritöös pandi selgelt paika nõudmised, mida Protsessori keskkond peaks võimaldama, millele omakorda lisandusid läbirääkimistel ilmnenud kliendi lisa soovid ja nõudmised.

4.1 Süsteemi kirjeldus

Käesoleva süsteemi üldine eesmärk on pakkuda lõputööde teemade leidmise ja väljapakkumise platvormi nii üliõpilastele kui ka õppejõududele, hõlbustades seeläbi lõputööde teemade otsimise ja väljapakkumise protsessi.

4.2 Süsteemi nõuded

Kasutades eelmainitud lõputöö analüüsi ning autorite ja tellija vahelisi läbirääkimisi ning koosolekute pidamisi määrati keskkonnale Protsessor järgnevad süsteemi ja funktsionaalsuse nõuded:

1. **Teemade haldus:**
   - 1.1. Tudengina lõputöö teema lisamine.
   - 1.2. Juhendajana lõputöö teema lisamine.
   - 1.3. Oma esitatud teema muutmine ja kustutamine.
   - 1.4. Teemale kandideerimine üksikisikuna.
   - 1.5. Teemale kandideerimine grupina.
   - 1.6. Teemale kandideerimise tagasi võtmine.
   - 1.7. Oma loodud teema avamine või sulgemine kandideerimiseks
   - 1.8. Oma loodud teemale kandideerimiste vastuvõtmine ja tagasilükkamine.
   - 1.9. Oma loodud teema info kopeerimine teise või samasse projekti.
   - 1.10. Oma loodud teema arhiveerimine.
   - 1.11. Tudengina juhendajale juhendamise kutse esitamine.
   - 1.12. Teemale kaasjuhendaja määramine ja eemaldamine.
   - 1.13. Kaasjuhendamise kutse vastuvõtmine või tagasilükkamine.
   - 1.14. Juhendaja lisamine läbi emaili kutse.
   - 1.15. Kliendi lisamine teemale

2. **Kasutaja haldus:**
    2.1. Oma kasutajaga seotud informatsiooni muutmine
3. **Gruppide haldus:**
    3.1. Võimalus luua uusi gruppe
    3.2. Kutsuda inimesi oma loodud gruppi
    3.3. Grupiga liitumise kutse vastuvõtmine või tagasilükkamine
4. **Projektide haldus:**
    4.1. Projektide loomine
    4.2. Projektide muutmine
5. **Süsteemi nõuded:**
    5.1. Teemade filtreerimine vastavalt looja rollile, looja nimele, grupi suurusele ja teema nimele
    5.2. Võimalikult kasutajasõbralik ja intuitiivne
    5.3. Vaade iga projekti jaoks, kus on kajastatud tudengite ja juhendajate hetkeseis
    5.4. Võimalikult paindlik projektipõhiste õiguste jagamise süsteem
6. **Admin õigused:**
    6.1. Õigus kõike näha ja muuta.
    6.2. Admin paneel vaade (ainult adminile kuuluvate funktsionaalsuste haldamiseks).
    6.3. Õigus lisada AD gruppe valgesse nimekrija.
    6.4. Õigus muuta kasutajate ülesüsteemilist rolli (anda teistele kasutajatele admin roll ja seda eemaldada).
    6.5. Õigus luua uusi projekte.
    6.6. Õigus näha süsteemi logisid

Praktilise lahenduse loomisel lähtusid autorid neist nõuetest ning ehitasid paindliku ja kasutajasõbraliku lahenduse nende nõuete ja süsteemide funktsionaalsuste rahuldamiseks.

5. Raamistike valik

Raamistike ja teekide valimisel ei olnud autorid kohustatud valima või kasutama kindlaid tellija või juhendaja pool määratud raamistikke. Seeläbi said kõik kasutusel olevad raamistikud ning teegid valitud puhtalt nende võimekuse, mugavuse ja jätkusuutlikkuse kaalutlustel. Selline lähenemine raamistike ja teekide valikul aitas autoritel vältida eelneva Protsessor keskkonna arendamisel, just tehnoloogiate kasutamise õigustamata jätmisel tehtud vigu, kus valiti ebaoptimaalseid lahendusi.

5.1 Kaalutud alternatiivsed raamistikud

5.1.1 ASP .NET Core/C#

ASP .NET Core on Microsofti poolt loodud veebiarenduse raamistik. Selle suurim tugevus on selles, et see on loodud Microsofti poolt ja seega on selle kasutamisel kättesaadavad paljud teegid, mis teevad liidestamise microsofti süsteemidega lihtsaks. See oleks kasuks tulnud TalTechi poolt kasutusel oleva Azure AD süsteemiga integreerimisel. ASP .NET võimaldab kasutada C#-i, F#-i ja Visual Basicut, käesolevas rakendues oleks kasutusel olnud C#. Tulevaste arendajate vaatepunktist ei oleks see ilmselt kõige tuttavam keel, kuid C# on vägagi populaarne ja leiab laia tööalast kasutust.

5.1.2 Spring/Java

Meeskond kaalus uuesti kasutada Spring raamistikku, mis leidis kasutust ka vanas Protsessor rakenduses. Spring on Java baasil tagarakenduse loomiseks mõeldud raamistik. Springi potentsiaalsel valikul oli autoritel ideeks, et oleks võimalik ära kasutada lahendusi ja koodi vanast rakendusest, kuid see mõte jäeti küllaltki kiiresti kõrvale. Probleem seisnes selles, et kasulikku koodi, mida oleks saanud vanast rakendusest üle võtta, oleks olnud küllaltki väike hulk ning see kood omakorda oleks võinud kaasa tuua samu vigu, mida tegid eelnevad arendustiimid. Kasuliku koodi hulka vähendas ka märgatavalt eelneva Protsessor rakenduses kasutusel olnud Camunda, mille jaoks kirjutatud koodi ei oleks olnud praktiline ümber kirjutada omaette seisva rakenudse näol.

5.1.3 Django/Python

Üheks parimaks kandidaadiks osutus arendusmeeskonnale ka Django raamistik, mis on kirjutatud Pythoni keeles. Kuna Pythonit õpetatakse paljudel infotehnoloogia õppekavadel, oleks Django olnud ka tuleviku arendajate mõttes sobilik valik. Autorid otsustasid siiski vähese kogemuse ja küllaltki järsu õppimiskõvera[10] pärast, mida oleks vaja Django heaks arenduseks, seda raamistikku mitte valida.

5.2 Tagarakenduse raamistik ja teegid

5.2.1 Ruby on Rails

Ruby on Rails (edaspidi Rails) on Rubys kirjutatud *full-stack* veebirakenduse arendamise raamistik[11]. Rails ilmus esmalt aastal 2004 ja on üks populaarsemaid veebiarenduse raamistikke[12]. Rails-i loojad on asetanud rõhku järgmistele põhimõtetele Rails-i arendamise käigus[13]:

- Optimize for programmer happiness - Proovida rahuldada arendaja subjektiivseid (ja kohati ebaaloogilisi) soove. Ruby, ja Rails laiemalt, lubab koodisiseselt kasutada kohati ebaefektiivset süntaksi, heaks näiteks saab sellest tuua olukorra kus arendaja soovib saada massiivi kolmandat liiget, tüüpiliselt kutsutakse selleks käsklust "massiiv[2]", kuid rails lubab ka kasutada "massiiv.third" käsklust, puhtalt kuna arendajal võib olla selleks soov.
- Convention over Configuration - Eelistada loogilist süsteemi ülesehitust, näiteks kui on mudel "Õpilane", siis on ka tõenäoliselt tabel "Õpilased" ja õpilasele viitava välisvõtme korral selle nimi "Õpilase_id", selline loogiline eeldamine vähendab kirjutatava koodi mahtu ja enese kordamist arenduse jooksul
- The menu is omakase - Lihtsustada arendaja jaoks valikud milliseid teeke valida ja kasutada, eelistada ühte universaalsemat tööriista mis on kõiges hea üle mitme rangelt spetialiseeritud tööriista.
- No one paradigm - Võimaldada arendajal järgida erinevaid süsteemide ülesehitamise põhimõtteid ja teda mitte piirata.
- Exalt beautiful code - Väärtustada inimkeelele lähedast ja lihtsasti arusaadavat koodi.
- Provide sharp knives - Lubada arendajal teha muutuseid, mis muidu oleksid keelatud, näiteks on lubatud sisseehitatud funktsioonide muutmine, näiteks kui "mõni string".capitalize tagastab tavaliselt "Mõni string", siis saab seda funktsiooni muuta vastavalt arendaja soovile.
- Value integrated systems - Väärtusta süsteeme, mis katavad kõiki vajadusi

- Progress over stability - Ebastabiilne areng on väärtuslikum kui kindel stagneerumine.
- Push up a big tent - Mitte keelata erinevate süsteemide arendust, mis võivad rikkuda algseid põhimõtteid.

5.2.2 Miks Ruby on Rails?

Railsi valimisel olid suurimad tegurid:

- Kiire arendus - Rails võimaldab vähese koodiga saavutada palju ja seeläbi võimaldadab saavutada rohkemat piiratud ajaga lõputööd tehes. Railsil on palju QOL funktisoone, mis arendajat aitavad, sellest hea näide on "rails g" käsklus, mille abil saab arendaja öelda, mida ta luua tahab, ja Rails paigutab faili õigesse kohta ise ja seab ka loogilise klassinime.
- Kindel ülesehitus - Eelnevalt mainitud Convention over Configuration printsiibile soosib Rails loogilisi nimetusi *controller*'itele, mudelitele, klassidele jne. See kehtib ka nendega seonduvatele failidele. Samuti ootab Rails ka loogilist failipaigutust, see tähendab, et amatöör arendaja nagu autorid on ei saa teha liigselt keerulist ja ebavajalikku omaloomingut süsteemi komponentide sidumisel ning paigutamisel.
- Loetavus - Ruby on loodud üritades matkida inimkeelt ja olla seeläbi loetav ka tavainimesele, see hõlbustab edasiste arendajate sisseelamist projekti.
- ActiveRecord ORM - ActiveRecord on Railsi sisseehitatud suhtlusvahend andmebaasi ja tagarakenduse vahel. ActiveRecord lubab luua mudelitevahelisi seoseid ja loogikat kiiresti ja loogiliselt, näiteks kui objekt "Student" kuulub objektile "Role" *many-to-one* seose alusel, saab vastavatele mudelitele märkida "belongs_to :role" ja "has_many :users" ja seeläbi saab kätte seotud objekte. ActiveRecordi roll rakenduses on teha arendaja eest ära andmebaasipäringud ja esitleda neid mugavamalt kui puhas SQL.
- Lihtne ümber kirjutada - Railsi kasutades on lihtne ja kiire muuta andmebaasiskeemi või programmi loogikat.

5.2.3 Railis suurim puudus arendajale

Railsi suurim puudus on tema endi sisse ehitatud mugavused. Kuna Railsis teeb arendaja eest palju tööd ise ära ei pruugi alati koodivea leidmine olla nii otsekohene kui ta oleks mõnes keeles, kus arendaja määrab ise kõik seosed. Selle leevendamiseks on Railsis võimsad silumise võimalused, näiteks käesoleva süsteemi arendamisel olid peamiselt kasutuses Railsi konsool ja ByeBug teek. Mõlemas on võimalik teha nii andmebaasipäringuid kui ka

jooksutada otse koodi.

5.2.4 Meie Railsi kasutus

Käesolevas süsteemis kasutame Railsi puhtalt tagarakendusena. Seega on tema tööülesanneteks andmebaasiga suhtlus, kasutaja andmete saamine AzureAD-st, autentimine ja vastavate andmete edastamine esirakendusele.

5.2.5 Sorbet

Sorbet on Rubyle loodud tüübikontrollija, mille peamiseks eesmärgiks on anda arendajale IDE-laadseid omadusi nagu:

- Koodi automaattäitmine ja kontroll (Vaata jooniseid 1 ja 2)

```
def dogs
  current_user.dogs
end
```

Joonis 1. *Vigane funktsioon ilma sorbetita.*

```
def dogs
  current_user.dogs    Method `dogs` does not exist on `User`
end
```

Joonis 2. *Vigane funktsioon sorbetiga.*

- Redaktorisisene dokumentatsioon
- Funktsioonide sisendite ja väljundite määramine (Joonis 3)

```
sig { returns(T::Boolean) }
def admin?
  'is_admin' if role&.name == 'admin'
end    Expected `T::Boolean` but found `T.nilable(String)`
```

Joonis 3. *Funktsiooni väljundi määramine ja kontroll.*

See teeb arenduse käigus arendaja elu palju lihtsamaks ja tagab, et tüübivead esinevad harva. See aitab palju, kui programeerija on Ruby-ga ebakindel või on seda alles õppimas, ja säästab palju aega, kuna vead leitakse üles enne kompileerimist.[14]

5.2.6 Pundit

Pundit on teek, mille eesmärk on tegeleda süsteemi autorisatsiooniloogikaga ja anda arendajale hõlpsasti kasutatavaid tööriistu, et tagada oma rakenduse turvalisus. Seda teeb ta Policy failide näol, need failid lubavad arendajal kontrollida, kas soovitud tegevus on lubatud vaadates kes tegevust soovib teha ja kas tal on õigus sihtobjekti või objektiklassi peal teha operatsioone. See teegi kasutamine tagab, et autorisatsiooniloogika on ühtne ja kergesti jälgitav. [15]

5.2.7 Rubocop

Rubocop on linter ja koodi vormindaja, mille eesmärk on proovida tagada puhta koodi kirjutamist ja teavitada arendajat halbadest otsustest/disainist. Rubocop on väga paindlik ja tema käitumist saab igati muuta, et tagada projekti juures asjakohasus. Käesolevas süsteemis on kasutusel laiendus rubocop-rails, mis on Railsi arendamiseks mõeldud rubocopi versioon.[16] [17] Näide rubocopi veateatest (Vaata joonis 4):

```
def admin?
  if role&.name == 'admin'    Style/IfWithBooleanLiteralBranches: Remove redundant `if` with boolean literal branches.
    true
  else
    false
  end
end
```

Joonis 4. *Rubocopi veateade.*

5.2.8 Annotate

Annotate'i (aka AnnotateModelsi) eesmärk on genereerida vastavatesse failidesse andmebaasimudeli kirjeldust, et arendaja elu mugavamaks teha. Annotate'ile saab ette anda just millistele failidele ja kui spetsiifiliselt peaks ta lisama mudeli kirjeldusi. [18] (Vaata joonis 5)

```
# == Schema Information
#
# Table name: roles
#
#  id         :uuid             not null, primary key
#  name       :string           not null
#  created_at :datetime         not null
#  updated_at :datetime         not null
```

Joonis 5. *Annotatsiooni rolli mudelile.*

5.2.9 Pagy

Pagy on kõige populaarsem paginatsiooniga tegelev teek Rubyle ja toetab enamusi paginatsiooni vorme. Pagy võimaldab kiiret ja kerget paginatsiooni kõigis Ruby rakendustes olenemata andmebaasitüübist või raamistikust. [19]

5.2.10 RSpec

RSpec on rakenduse testimiseks kasutatav avatud lähtekoodiga teek, mis võimaldab tagarakenduse hõlpsat testimist. RSpec-iga saab luua testimiseks suvaliselt genereeritud andmeid, et tagada piirijuhtude leidmist ja lahendamist. RSpec-i esimene versioon tuli välja aastal 2007 ja teda arendatakse tänaseni. [20]

5.2.11 Audited

Audited on laiendus Railsis kasutusel olevale ActiveRecord ORM-ile, selle eesmärk on logida muutuseid andmebaasis olevatele andmetele, märkides seejuures muudetud välju, muudatuse teinud kasutaja ja millise tegevuse käigus muutus toimus. [21]

5.3 Esirakenduse raamistikud

5.3.1 React

React on üks levinumaid JavaScripti raamistikke, mille eesmärk on luua kasutajaliideseid paindlikul ja tõhusal viisil. See võimaldab arendajatel ehitada suuri veebirakendusi, mis võimaldavad andmeid muuta ja värskendada ilma veebilehte uuesti laadimata. Eelmised tiimid kasutasid Vue.js raamistikku oma Protsessor versioonides, kuid käesolevas rakenduses on kasutatud React. Peamine põhjus Reacti valikul on see, et TalTech Digital Styleguide arendustiim on loonud React komponentide paketi, mis tulevad juba TalTechi CSS klassidega, mis annavad neile professionaalse välimuse ning see tõstab arenduse kiirust märgatavalt. [22]

5.3.2 TalTech Digital Styleguide

TalTech Digital Styleguide [23] on React Bootstrapil põhinev React komponentide paket, mille peamiseks eesmärgiks on lihtsustada arenduse protsessi ning standardiseerida stiilid, et säiliksid traditsioonilised TalTech stiilid, tekstid, värvid ning proportsioonid. TalTech Digital Styleguide on väga uus toode ning selle lõputöö raames on see esimest korda kasutusele võetud. Selle kasutamisel esinesid mitmed plussid ja miinused.

Styleguide kasutamise plussid

- Järjepidevus ja ühtsus: Styleguide aitas hoida projekti visuaalset järjepidevust, mis on oluline suuremate tiimide ja projektide puhul.
- Kiirem arendusprotsess: Eeltöödeldud stiili komponendid kiirendasid arendusprotsessi, kuna me ei pidanud iga detaili käsitsi nullist looma.
- Lihtsam koostöö: Styleguide'i olemasolu lihtsustab tulevikus uute arendajate kaasamist projekti, kuna stiilid olid juba ette defineeritud.

Styleguide kasutamise miinused

- Paindlikkuse piiramine: Kuigi Styleguide tagas järjepidevuse, piiras see mõnikord ka loovust ja kohandamist, kuna kõik visuaalsed elemendid pidid vastama kindlatele reeglitele. Näiteks algse Figma disaini prototüübi järgi nupp, mis vahetab keelt, peaks olema päise komponendi peal, et see oleks kohe kasutajale paremini nähtav, kuid nupp oli stiilijuhendi alsusel ette nähtud profiili rippmenüüsse.
- Algusfaasi probleemid: Kuna tegemist oli uue tootega, nõudis selle kasutusele võtmine alguses täiendavat aega ja ressursse, et mõista kõiki selle omadusi ja võimalusi.

Vaatamata dokumentatsioonile, mõnede komponentide kasutus ei olnud kohe näidiselt arusaadav reaalses projektis ning näidised olid kohati liiga abstraktsed. Mõned komponendid, mis olid Figma disainis [24] välja toodud, puudusid komponentide paketis. Selle pärast loodi mõningad komponendid autorid ise nullist.

- Tehnilised probleemid: Uute toodete puhul võib esineda tehnilisi kitsaskohti või vigu, mis võivad arendusprotsessi aeglustada või keerulisemaks muuta. Üks suurimaid väljakutseid oli integreerida TalTech Digital Styleguide CI/CD jaoks: TalTech Digital Styleguide'i kasutamisel ilmnes probleem, et ei olnud olemas häid näiteid või juhendeid selle kohta, kuidas integreerida styleguide'i CI/CD protsessidesse või Dockeri failidesse. See raskendas automaatsete arendus- ja kasutuselevõtuprotsesside seadistamist. Styleguide'i kasutamine nõudis ka sisselogimist, mis tõi kaasa täiendavaid keerukusi, eriti automatiseeritud keskkondades, kus sisselogimisprotsessi haldamine võib olla keeruline ja aeganõudev.
- Kontaktide puudus: Kuna TalTech Digital Styleguide'i dokumentatsioonilehel puudusid kontaktandmed, pidime arendustiimi poole pöörduma meie juhendaja kaudu. Lisaks ei olnud dokumentatsioonis kohta, kus oleks võimalik esitada küsimusi või tagasisidet, mis raskendas suhtlemist arendajatega ja piiras vahetut tagasiside andmist toote parendamiseks. See tõi kaasa väljakutseid, eriti probleemide kiire lahendamise ja toote parendamise protsessis, kus otsekontakt arendajatega oleks olnud hädavajalik.

5.3.3 i18next

i18next [25] on populaarne JavaScripti teek veebi-, mobiili- ja töölauarakenduste rahvusvahelistamiseks, see ühildub mitmete Front-End raamistikega nagu React, Angular ja Vue. Seda töörista kasutasid ka eelnevad meeskonnad. i18n paistab silma mitmete eeliste poolest võrreldes teiste rahvusvahelistamise teekidega, nagu FormatJS ja Polyglot, muutes selle paljude projektide jaoks eelistatumaks valikuks tänu ulatuslike pluginate kogule, kogukonna toele ning paindlikkusele.

5.3.4 Redux

Projektis oli oluline kasutada staatusehaldurit (*State manager*), et hoida kasutajainfot globaalselt, mis võimaldab komponentidel jagada ja hallata ühist seisundit tõhusalt ning hallata sisselogimist. Redux [26] on üks populaarsemaid staatusehaldusvahendeid, eriti Reacti projektide puhul, ja on laialdaselt tunnustatud kui hea valik [27]. Redux on valitud järgmistel põhjustel:

- Laiendatavus: Redux annab ruumi tulevikus laiendamiseks. Tänu oma modulaarsele ja skaleeritavale arhitektuurile on Redux hea valik projektidele, mis võivad ajas kasvada või nõuda täiendavaid funktsioone.
- Redux DevTools: Üks kõige märkimisväärsemaid vahendeid, mida Redux pakub, on Redux DevTools, mis on väga kasulik silumiseks. See tööriist võimaldab arendajatel jälgida rakenduse seisundi muutusi reaalajas, tuvastada ja lahendada vigu ning isegi testida uusi funktsioone ilma olemasolevat seisundit muutmata.

5.3.5 Typescript

React projekti arenduses on võimalik kasutada JavaScript või TypeScript[28]. TypeScript on Microsofti poolt arendatud programmeerimiskeel, mis laiendab JavaScripti, lisades tüübikontrolli. Rakenduses on kasutatud TypeScript, sest see parandab koodi loetavust ja avastab vigu juba arenduse ajal, vähendades käivitusaegseid probleeme. TypeScript sobib suurtele projektidele, tagades koodi hooldatavuse ja skaleeritavuse tänu tüübikontrollile.

5.3.6 SCSS

SCSS [29] on CSS-i laiendus, mis võimaldab kasutada muutujaid, SCSS "mixin" [30] ja muid funktsioone, muutes CSS-i paindlikumaks ja hooldatavamaks."Mixin" funktsioonid on eriti kasulikud, sest need võimaldavad defineerida taaskasutatavaid CSS-i reeglite kogumeid. SCSS-i on projektis kasutusele võetud kuna see pakub struktureeritumat ja modulaarset lähenemist veebilehtede kujundamisele. SCSS-i kasutamine aitab vähendada koodi korduvust, lihtsustada koodi haldamist ja kiirendada arendusprotsessi, kuna muudatused on lihtsamini hallatavad ja rakendatavad [31].

5.3.7 Vite

Vite [32] on kiire ja kaasaegne veebiarenduse tööriistakomplekt, mis pakub kiiret moodulite laadimist ja tõhusat arendusvoogu. See on eriti kasulik React ja TypeScripti projektides, kuna Vite kiirendab arendustegevust tänu rakenduste kiirele käivitumisele. Vite ehitusprotsessis jätetakse TypeScripti tüübikontroll vahele, mis võimaldab kiiremaid taaskompilatsioone. Lisaks toetab Vite kuuma laadimist, mis tähendab, et muudatused koodis kajastuvad kohe brauseris ilma lehe taaslaadimiseta. Vite integreerub hästi kaasaegsete töövoogudega ning võimaldab kohandatavaid konfiguratsioone, mis muudab CI/CD seadistamise protsessi lihtsamaks ja tõhusamaks. See vähendab ehitusprotsesside aega ning hõlbustab koodi pidevat juurutamist ja arendust [33].

5.4 Andmebaas

5.4.1 Postgres

Valitud sai just PostgreSQL, kuna see on võimas, avatud lähtekoodiga objekt-relatsiooniline andmebaasi süsteem, millel on üle 35 aasta aktiivset arendust, mis on talle taganud tugeva maine usaldusväärsuse, funktsionaalsuse ja jõudluse osas[34], mistõttu on PostgreSQL leidnud väga laialdast kasutust tarkvaraarenduses. Samuti oli eelneva projekti raames ka kasutatud PostgreSQL-i ning autorite töö raames ei olnud erilisi andmebaasi funktsionaalsuse nõudeid, mis oleksid andmebaasi valikut kuidagi kitsendanud. PostgreSQL on laialdasalet kasutuses ka TalTech informaatika õppekaval ning oli seetõttu autoritele juba tuttav.

5.5 Muud teegid

5.5.1 Docker

Docker on tehnoloogia, mis lihtsustab programmide ehitamist, paigaldamist ja jooksutamist kasutades konteinereid [35]. Konteinerid on kergekaalulised, iseseisvad ja käivitatavad paketid, mis sisaldavad kõike tarkvara tööks vajalikku (rakenduse koodi, süsteemi tööriistu, teeke ja raamistikke). Docker pakub tööriistu ja platvormi nende konteinerite tõhusaks haldamiseks[36].

Docker elimineerib probleemid, mis võivad tekkida proovides jooksutada programme erinevates keskkondades, tagades selle, et arendatav programm jookseb valutult kõigi arendajate masinates. Tänu Dockeri suurtele eelistele leiab see ka väga laialdast kasutust ning on vundamendiks pea igale veebipõhisele projektile. Dockeri mugavuse ja väga laialdase kasutuse tõttu on see ka vundamendiks selle projekti eesmärgipärases töös[36].

5.5.2 Nginx

Ngix on avatud lähtekoodiga tarkvara, mille eesmärk on veebipäringute suunamine, manageerimine, salvestamine ja üldisem haldamine. 2018. aastal avaldatud Dockeri kasutuse statistikas toodi välja, et Nginx oli kõige tihemini kasutuses olev tehnolooogia olles kasutusel üle 35 protsendis Dockerit kasutavates firmades. [37]

Nginx on rakenduses kasutusel kui *reverse-proxy*. Nginx-i ülesanne on suunata sissetulevaid päringuid eesrakendusele ja tagarakendusele vastavalt. Süsteemis on kasutusel 2

proxy't, üks mis suunab cs.taltech.ee domeenilt meie rakendust jooksutavale virtuaalmasinale ja teine mis suunab sellel virtuaalmasinal taga- või eesrakendusele vaadates sissetuleva päringu URL-i.

6. Meie lahendus

6.1 Tööjaotus

Tööjaotus teostati vastavalt esirakenduse ja tagarakenduse arenduse ajalistele kuludele. Autorid leidsid et kõige mõistlikum oleks jaotada meeskond 2 liikmeliseks esirakenduse tiimiks ja 1 liikmeliseks tagarakenduse tiimiks. Autorite tööjaotus sai paika pandud järgmiselt:

- Kauri Kinks - esirakenduse arendus
- Artjom Gussev - esirakenduse arendus
- Raul Akerman - tagarakenduse arendus

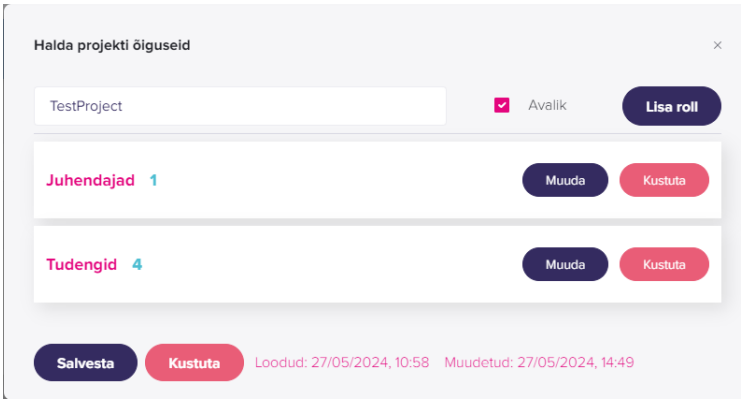
Samuti leidsid rakendust erinevad elemendid SCRUM ja agiilse arenduse metoodikatest. Autorid teostasid iganädalasi koosolekuid tellijaga, et saada tagasisidet, näidata rakenduse arengut, lisada uusi funktsionaalsusi või muuta olemasolevaid vastavalt kliendi soovidele ja ettepanekutele. Samuti teostasid tiimiliikmed üksteise koodi arvustusi, et hoida koodi kvaliteeti kõrgel ning vältida vigade ja ebaoptimaalsete lahenduste laekumist arendus-serverisse ja repositooriumitesse. Lisaks eelistasid autorid ka staatiliselt planeeritud tiimisiseste koosolekute asemel dünaamilist suhtlust igapäevase arendus käigus läbi erinevate suhtluskanalite , tõstes seeläbi arenduskiirust ja tiimiliikmete arusaama kaasliikmete tegevustest ja probleemidest. Semestri alguses pandi paika ka arendusplaan ja ajajoon ning seati sellele vastavad eesmärgid mida autorid töö käigus saavutada püüdsid.[38] [39]

6.2 Rakenduse ülesehitus ja struktuur

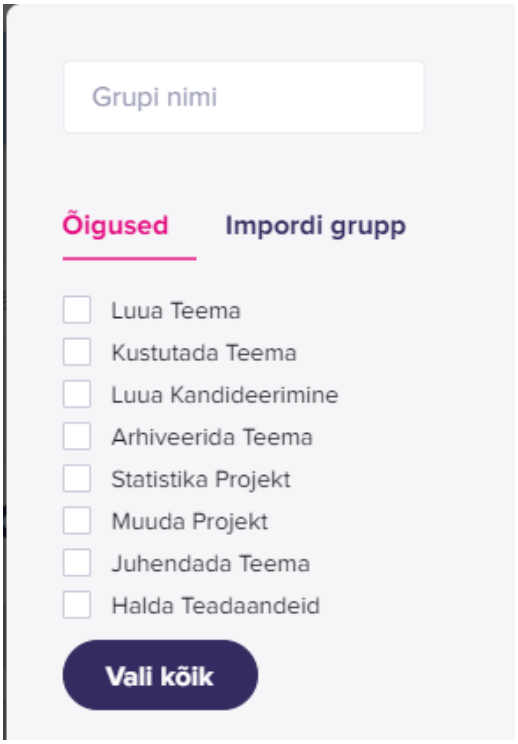
Loodud rakendus on loomult CRUD operatsioone täitev veebirakendus, kus esirakendus saadab päringuid tagarakenduse pihta ja tagarakendus loeb ja kirjutab andmebaasi ning edastab esirakendusele vastavat infot.

Üheks peamiseks põhjuseks, mille pärast sai kogu rakendus uuesti ehitatud, on luua süsteem, mis võimaldaks paindlikult õiguseid jagada, mida ei suudetud eelmise rakenduses Camundaga teostada. Loodud rakenduses on erinevad operatsioonid, mida mainiti nõuete peatükis 4, koondatud õigusteks (vaata joonis 7), mida saab läbi projekti halduse vaate (vaata joonis 6) erinevatele kasutaja gruppidele määrata. See funktsionaalsus võimaldab projekti muutmise õigusega kasutajal luua iga projekti jaoks vastavalt vajadusele vastavate

õigustega projektisiseseid rolle ning neile määratud õiguseid soovikorral jooksu pealt muuta ning kasutajaid lisada või eemaldada. Selline lähenemine õiguste määramiseks on märgatavalt pandlikum ja robustsem kui vanas süteemis kasutusel kindlate õigustega süsteemi rollid, mis määrasid õiguseid (siinkohal on mõeldud ligipääsu funktsionaalsustele projektisiselest, mitte projektile) üle kogu veebilehe mitte projektipõhiselt.



Joonis 6. *Pilt õiguste halduse kasutajaliidesest.*



Joonis 7. *Pilt õiguste rolli halduse kasutajaliidesest.*

Loodud süsteemis on ka kasutusel ülesüsteemilised rollid ("admin" ja "user") mida kasutatakse "admin" süsteemirolliga kasutajatele lisa funktsionaalsuste kuvamiseks "admin paneel" kasutajaliidese näol.

Antud süsteemi raames loodi ka statistika, süsteemi logide ja teavituste funktsionaalsused, mis eelneva süsteemi puhul ei eksisteerinud. Statistika vaade on saadaval kõigile, kellele on

määratud vastava projekti raames "stats Project" õigus. Statistika vaates antakse kasutajale kõigi projekti alla lisatud kasutajate hetkeseisu ülevaade ning samuti kuvatakse infot projekti all juhendajateks määratud kasutajate, loodud ja juhendamisel olevate teemade arvu. Süsteemi logide vaates antakse "admin" süteemirolliga kasutajatele ülevaade kõigest, mis on läbi päringute süsteemis toimunud, näiteks "John Doe lisas teema", "John Doe esitas kandideerimise teemale "Teema 1"". Nagu eelmainitud oli sai lisatud ka teavituste süsteem mille abil on võimalik panna üles teavitusi nii kodu lehele kui ka projektisiseselt.

Samuti implementeeriti autorite lahenduses ka Azure Active Directory (edaspidi AD) gruppide "*white list*" lisamise funktsionaalsus, mis leevendab eelnevas süsteemis andmebaasi tekkinud tohutute gruppide hulka lubades süsteemil automaatselt luua ja hallata ainult neid AD gruppe, mis selles nimekirjas juba olemas on. Antud lahenduse puhul on "admin" õigustega kasutajal võimalik lisada AD gruppe nimepidi valgesse nimekrja, ning kui süsteemi siseneb kasutaja vastavast AD grupist, lisatakse ta automaatselt vastavasse gruppi andmebaasis. See hõlbustab projekti halduritel kasutajate lisamist erinevatesse projektipõhistesse rollidesse. Kui importida AD grupp vastava rolli alla (joonis 7, "impordi grupp"), siis kasutaja lisandumisel süsteemi on tal automaatselt ligipääs ja õigused vastavate projektide nägemiseks ja kasutamiseks, kuhu AD grupp, mille alla ta kuulub, lisatud on.

6.3 Tagarakendus

Päringu kättesaamisel kontrollib tagarakendus algselt, kas kasutaja on süsteemi sisse logitud ja kellega süsteemisiseselt tegu on. Järgmisena kontrollitakse, kas kasutajal on õigus soovitud objektile või objektiklassile ligi saada ja objekti soovi korral muuta. Kasutaja õigust soovitud operatsiooni tegemiseks kontrollitakse Punditi poolt pakutava *policy* failide abil. Õiguse olemasolu korral saadetakse objekt või objektiklass läbi temale vastava Serializer klassi, mis määrab, milliseid välju ja mis moel mudeli kohta esirakendusele tagastatakse. Sisselogimise valideerimine käib läbi veebilehitsejasse salvestatud session_id, mis saadetakse veebilehitsejale, kui kasutaja edukalt sisse logib, süsteemisiseselt hoitakse session_id ja kasutaja seost Redises.

6.3.1 Autentimine

Kasutajate autentimine toimub läbi TalTechi Azure'i. Sisselogimisel suunatakse kasutaja kindlale URL-ile, kus kasutaja peab kinnitama oma soovi süsteemi sisse logida ja seeläbi anda rakendusele ligipääs kindlatele andmetele. Seejärel saadab Azure tagarakendusele tokeni, millega tagarakendus seeläbi saab pärida kasutaja andmeid ja luua süsteemisisene

kasutaja. Seeläbi on saavutatud olukord, kus TalTech meiliga kasutajad saavad kasutada käesolevat rakendust uut kontot loomata ja selles ei hoita paroole või muud kriitilist infot.

6.4 Esirakendus

Esirakendus järgib järgmisi põhimõtteid:

1. **Sisselogimine** Kogu rakenduse loogika algab kasutaja autentimisest. Kui kasutaja ei ole sisse loginud, näeb ta ainult teemasid ja projekte mis on määratud avalikkusele nähtavaks. Autentimise nupu vajutamisel suunatakse kasutaja vastava OAuth 2.0 sisselogimise URL-iga autentimisteenusesse.
2. **Veebilehtede suunamise kaitse**: Kõikede veebilehtede suunamised on rakenduses kaitstud, et isikud, kellel ei ole õigusi teatud lehte näha, ei pääseks sellele juurde. Kasutajaid suunatakse vastavalt nende rollile määratud vaadetele 8. Nii kasutajainfo kui ka rollid salvestatakse Reduxi seisundihaldusesse, mis võimaldab neid hõlpsalt kogu rakenduse ulatuses hallata. Kui määratud suunamise link ei leita, kuvatakse kasutajale vastav sõnum. Rakenduses on (’/’) route asendatud (’/home’) URL-iga, kuna see aitas vältida proxy-serveri suunamise probleemi, mis ilmus serveris [6.5.1].
   - Avalikud suunamised: Avaleht (’/home’) on kõigile ligipääsetav, kuid selle sisu on õigustega kaitsud. Ülejaanud veebilehed on kaitstud vastavalt kasutaja rollile.
   - Admini (’admin’) roll: Adminil on vaikimisi ligipääs kõigile vaadetele. Veebilehte ’/admin’ saab näha ainult kasutaja ’admin’ rolliga.
   - Kasutaja (’user’) roll: Tavalised kasutajad saavad ligipääsu teatud lehtedele ’/account’, ’/groups’, ’/thesis’ ja ’/theses’, nende lehtede sisu sõltub projektihalduri poolt antud õigustest.
3. **Teemade kuvamine staatuste järgi:** Teemad kuvatakse staatuste järgi. Teema looja näeb alati oma teemat projekti teemade listis, administraator näeb kõiki teemasid sõltumata nende staatusest. Ülejäänud kasutajad ei näe teemat, kui selle staatus on ’archived’, ’draft’ või ’closed’.
4. **Üldine loogika:** Kui kasutajal puudub õigus teatud vaateid näha või tegevusi teha (näiteks teema vaatamine projektis, nende loomine või kandideerimine), siis ta ei pääse nendele tegevusele või sisule ligi.
5. **Mobiili versioon:** Rakenduse disain on loodud kohandatuks tänu Bootstrap [40], Flexbox [41] ja SCSS [29] kombinatsioonile. See tagab, et kujundus kohandub sujuvalt erinevate ekraanisuurustega, pakkudes kasutajatele optimaalset kasutuskogemust nii lauaarvutis kui ka mobiiliseadmetes. Bootstrapi paindlik ruudustikusüsteem ja Flexbox võimaldavad elementidel automaatselt ümber paigutuda, samal ajal kui

```
<BrowserRouter basename={getBaseName()}>
  <Routes>
    <Route path="/" element={<Navigate replace to="/home" />} />
    <Route path="/home" index element={<HomePage />} />

    <Route element={<ProtectedRoute allowedRoles={['admin']} />}>
      <Route path="/project-stats" element={<StatsPage />} />
      <Route path="/admin" element={<AdminPage />} />
    </Route>

    <Route element={<ProtectedRoute allowedRoles={['admin', 'user']} />}>
      <Route path="/groups" element={<GroupsPage />} />
      <Route path="/thesis" element={<ThesisPage />} />
      <Route path="/theses" element={<ThesesPage />} />
      <Route path="/account" element={<AccountPage />} />
    </Route>

    <Route path="*" element={<div>404 PAGE NOT FOUND</div>} />
  </Routes>
</BrowserRouter>
```

Joonis 8. Rakenduse suunamiste kontrolli kood

SCSS kohandused annavad lisakontrolli ja täpsust iga vaate suuruse jaoks.

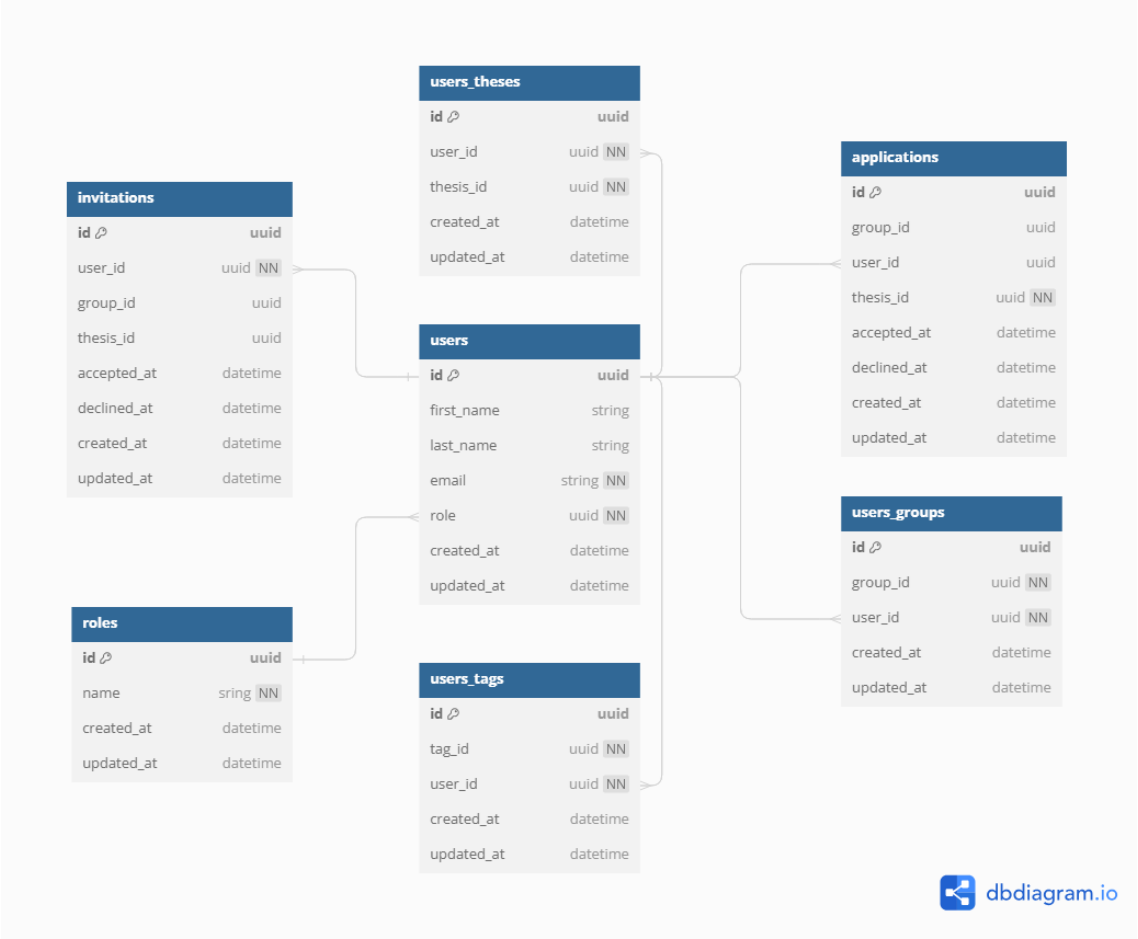
6.4.1 Andmebaas

Andmebaasis on 22 tabelit, millest 8 seovad mudeleid omavahel ja 14 hoiavad sisulisi andmeid.

Andmebaasi koguskeem (Joonis 9)

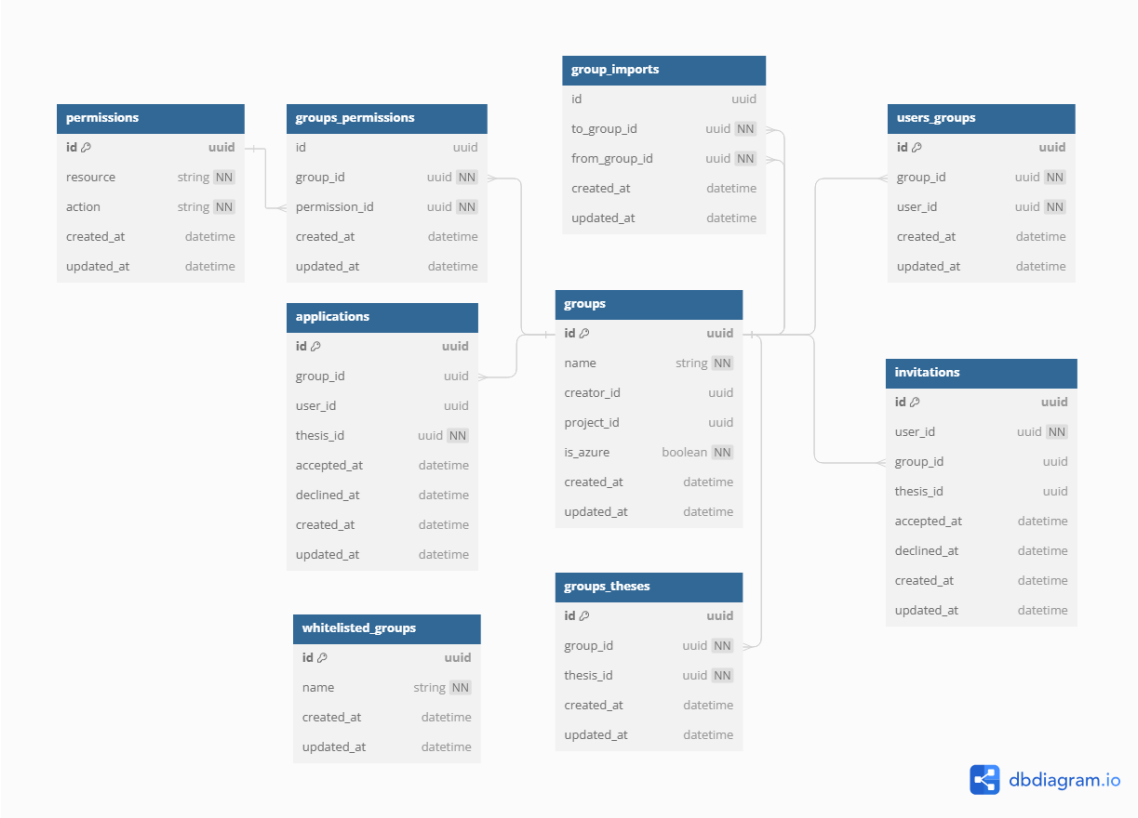
Joonis 9. *Andmebaasiskeem.*

Andmebaasi kasutajate alamosa (Joonis 10)



Joonis 10. *Andmebaasi kasutajate alamosa.*

Andmebaasi gruppide alamosa (Joonis 11)

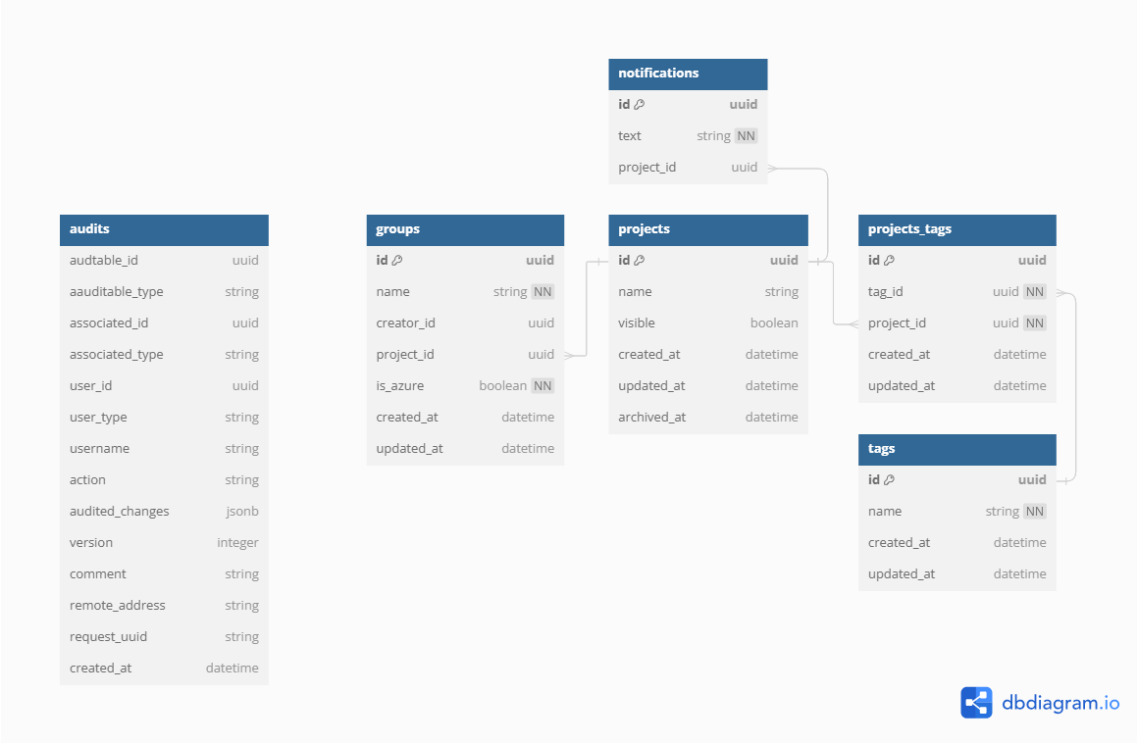


Joonis 11. *Andmebaasi gruppide alamosa.*

Andmebaasi teemade alamosa (Joonis 12)

Joonis 12. *Andmebaasi teemade alamosa.*

Andmebaasi projektide alamosa koos logimise tabeliga (Joonis 13)



Joonis 13. *Andmebaasi projektide alamosa.*

6.4.2 CI/CD, GitLab, runner

GitLab on veebipõhine DevOpsi elutsükli tööriist, mis pakub Git-repositooriumi halduri, wikit, probleemide jälgimist ja CI/CD torustiku funktsioone. GitLab kasutab avatud lähtekoodiga litsentsi. Kogu see süsteem võimaldab meeskondadel teha koostööd koodi kirjutamisel, hallata repositooriume, jälgida probleeme ning automatiseerida tarkvaraarendusprotsesse.

GitLab Runnerid on agendid, mis käivitavad CI/CD konfiguratsioonis määratud CI/CD-töid. Need võivad töötada erinevatel platvormidel ja käivitada töid isoleeritud keskkondades. Runnerid on konfigureeritud suhtlema GitLabiga, et saada töid, neid käivitada ning nende tulemustest tagasisidet anda.

CI/CD automatiseerib koodimuudatuste testimise ja kasutuselevõtu protsessi. Kui muudatused on arendaja poolt esitatud GitLabi repositooriumisse, käivitatakse CI/CD torustik, mis käivitab automaatselt testid, ehitades rakenduse uue versiooni seeläbi integreerides sisse tulevaid muudatusi serveris jooksvale rakendusele automaatselt, lähtudes määratud reeglitest ja konfiguratsioonidest (neid juhised hoitakse .gitlab-ci.yml nimelistes failides).

Kõigil, kellel on võimalik kasutada TalTechi GitLab võrku, on ligipääs TalTechi pakutavatele runneritele, mistõttu ei valmistanud autoritel CI/CD torustiku paigaldamine ja tööle saamine erilisi probleeme ning teha jäi vaid juhiste loomine runnerile .gitlab-ci.yml faili näol, mille alusel rakenduse ehitamine serveris töötab.

6.5 Suurimad esinenud vead ja probleemid

6.5.1 Alamdomeeni puudumine

Süsteemi arendamisel oli pidevaks pinnuks alamdomeeni puudumine. Kuna rakenduse asukoht ei ole väljendatud kui cs.protsessor.taltech.ee vms, vaid hoopis cs.taltech.ee/meie-suunamine/, oli probleemiks esi- kui ka tagarakenduses õige suunamine. Mõlemad raamistikud eeldavad subdomeeni olemasolu ja tahavad enda alamkausta panna esimeseks pärast .ee lõpulist domeeni, seda tehes aga unustatakse /meie-suunamine/ vahele panna. See probleem küll leidis lahenduse, kuid selle peale kulunud aeg oleks võinud millegi sisulise väljatöötamise peale minna. Sarnane probleem oli ka eelmistel Protsessori arendajatel. [2]

6.5.2 Taltech Styleguide integreerimine CI/CD protsessi

Taltech Styleguide'i versiooni 10.0.5 integreerimine esirakenduse CI/CD protsessi osutus keeruliseks, kuna see nõuab autentimist TalTechi NPM registris. Sisselogimisprotsess on iteratiivne, nõudes esmalt kasutajanime ja seejärel parooli sisestamist. Alguses võttis kaua aega, et selgitada välja õiged väärtused NPM registri sisselogimiseks, kuna dokumentatsioonist või arendustiimist ei olnud neid andmeid saadaval. Need väärtused saadi lõpuks juhendajalt, kelle läbi on arendustiimiga kontakti hoitud. Probleemiks osutus ka erinev sisselogimisprotsess lokaalse arenduskeskkonna ja Docker-keskkonna vahel, Docker-keskkonna seadistamisel puudus piisavalt detailne dokumentatsioon. Lisaks nõudis registrisse sisselogimine spetsiaalset *–auth-type=legacy* parameetrit, mida dokumentatsioonis ei mainitud.

On olemas kaks levinud viisi, kuidas tavaliselt kasutatakse privaatseid NPM registreid CI/CD protsessis: npm-cli-login [42] ja NPM autentimis-token [43]. Styleguide'i dokumentatsioon pakkus kasutada NPM autentimis-token, kuid sellega esines palju probleeme ja seda ei saanud edukalt kasutada. Dokumentatsioonis pakutud sammud ei olnud piisavalt detailsed. Dokumentatsioonis oli ainult info selle kohta, mida panna .npmrc faili ning link NPM dokumentatsioonile, mis kirjeldab, kuidas saada auth token pärast edukat sisselogimist. Kuid puudus info Dockerfile'ide ja GitLab CI failide seadistamisest.

Autorid otsustasid kasutada npm-cli-login, kuna see nõudis vähem seadistamisi, oli sarnane NPM registri seadistamisega lokaalselt ehk kasutatakse sarnaseid põhimõtted nii lokaalses kui ka Docker-keskkonnas ning kasutab samu keskkonnamuutujaid, mis lihtsustab CI/CD protsessi seadistamist ja vähendab käsitsi konfiguratsiooni vajadust.

7. Tulemused

Lõpptulemusena valmis rakendus, mis vastab etteantud nõuetele peatükis 4 (väljaarvatud e-mail süteem). Loodud süsteem pakub paindlikumaid võimalusi projektide haldamiseks ja ülesehitamiseks ning annab reaalse ülevaate projektide ja nende kasutajate hetkeseisudest. Elimineeriti pea kõik eelneva projektiga tehtud probleemid ja sai loodud põhjalik dokumentatsioon [44], mis hõlbustab edasiarendajate tööd märgatavalt ja peaks tagama selle, et projekti arendusel ei tohiks tekkida enam vajadust projektiga uuesti alustada. Ainuke funktsionaalsus, mis jäi ajapuuduse tõttu implementeerimata, puudutab juhendaja lisamist teemale läbi emaili kutse saatmise. Kogu süsteemi arendamise käigus peetud koosolekud ja tagasiside võeti töös arvesse ning tehti selle alusel vajalikke muudatusi või lisasid. Rakendus on juba eesmärgipärases kasutuses ja läbi neti kättesaadav.

7.1 Edasised arengud

Funktsionaalsused edasiseks arendustööks:

- emaili teel juhendaja kutsumine
- "inbox" süsteem kõigi kutsete koondamiseks
- läbi emaili kasutajate teavitamine (kutsete esitamisel, kutsete saamisel, kutse staatuse muutumisel ja muudel laadsetel tegevustel)
- võimalik teha süsteem tiimikaaslaste leidmiseks (analoogselt teemade otsimisega antud süsteemis)

7.2 Ettepanekud

- **Tagasiside Taltech Styleguide arendustiimile:** Tagasiside Taltech Styleguide arendustiimile keskendub parandustele, mis muudaksid arenduskogemuse paremaks. Esiteks, mõned vead takistavad mõne komponendi tõhusat kasutamist, eriti nende puhul, mille juures puudub selge dokumentatsioon või lihtsad näidised. Tagasiside siseldab ka dokumentatsiooni soovitusi, mis aitab paremini tunda komponente ning lisada puuduv info, näiteks paketi integreerimist CI/CD protsessi [6.5.2]. Samuti, pakkusime kaaluda npm-cli-login meetod CI/CD osas, kuna see tundub lihtsam, intuitiivsem ning vajab vähem seadistamist. Tagasiside dokument asub GitLab repositooriumis [45].

8. Kokkuvõte

Käesolev töö saavutas suures osas seatud eesmärgid. Loodi paindlik ja robustne süsteem, mis võimaldab tudengitel kui ka õppejõududel välja pakkuda lõputöö teemasid ning omale sobivaid teemasid leida. Süsteem võimaldab tööriistu, mida kasutades saavad projektide haldurid määrata täpselt, milliseid õiguseid anda kasutajatele projekti tasandil läbi projektipõhiste rollide. Rakenduse kasutajad saavad luua gruppe ja teemale kandideerides on võimalik teha seda nii grupina kui ka eraldiseisva üksikkasutajana. Teema väljapakkuv kasutaja ja teema juhendajad saavad otsustada, milliseid kandideerimisi vastu võtta ja milliseid tagasi lükata. Igal teemal on võimalik muuta tema seisundit, et märkida, mis on töö seis ja seeläbi kandideerimist lubada või takistada. TalTechi Azure'ist tulevate gruppide alusel ja projektipõhiste rollidega on võimalik kindlustada, et ka kasutajad, kes projekti loomisel süsteemist puudusid, määratakse sinna automaatselt kasutaja sisselogimisel süsteemi mugavdades seeläbi projekti haldamise protsessi. Rakendus kasutab ametlikku TalTechi stiili ja seeläbi on kindlustatud ühilduv ja korrektne väljanägemine ning sarnasus teiste TalTechi veebirakendustega. Rakendus kasutab kohanduvat disaini, mis tähendab, et elementide asukoht on õige paigutusega, töötab ja on proportsionaalne kõikidel seadmetel, millel on võimalus kasutada veebilehitsejat. Rakenduse koodibaas ja dokumentatsioon on heas seisus, mis võimaldab tulevastel arendajatel kiiresti end rakendusega kurssi viia ja arendustööd jätkata.

Kasutatud kirjandus

[1] Dmitri Voronoi Nikita Birjukovs Aleksandr Rudoi. *Lõputööde keskkonna Protsessor kasutatavuse täiendused*. [Accessed: 01-05-2024]. URL: https://digikogu.taltech.ee/et/Item/6aa3762e-2009-4093-b826-bbac1ac1871f.

[2] Kaspar Ustav Sedrik Suurmets Mikk Järvis. *Lõputööde ja projektide haldamise infosüsteem Protsessor*. [Accessed: 01-05-2024]. URL: https://digikogu.taltech.ee/et/Item/ec611fb8-4a08-4b9b-8d58-ef4b75cdd0e7.

[3] Jevgeni Serkin Rasmus Juurik. *Tallinna Tehnikaülikooli lõputööde teemade haldamise ja juhendamiskokkulepete sõlmimise valdkonna äri- ja nõuete analüüs veebirakenduse Protsessor edasiarendamise toetamiseks*. [Accessed: 01-05-2024]. URL: https://digikogu.taltech.ee/et/Item/2a5faec8-379a-4db9-8dc3-3b65232c04e1.

[4] *Camunda*. [Accessed: 18-05-2024]. URL: https://camunda.com/.

[5] *Activiti*. [Accessed: 18-05-2024]. URL: https://www.activiti.org/.

[6] Richardas Keršis. *Lõputöö teemade kosjasobitaja infosüsteem TalTech ülikoolile*. [Accessed: 01-05-2024]. URL: https://digikogu.taltech.ee/et/Item/a38663d0-e703-439e-8084-3749118ed11a.

[7] Anneli Karu. *Lõputööde haldamise süsteemi projekteerimine, realisatsioon ja juurutamine Tallinna Tehnikaülikooli informaatikainstituudis DHS Amphora baasil*. [Accessed: 01-05-2024]. URL: https://digikogu.taltech.ee/et/Item/be4ba776-7ca2-46c6-834b-26710b591db7.

[8] Aleksandr Kezerev. *Veebilahendus lõputööde kavandite loomiseks*. [Accessed: 01-05-2024]. URL: https://digikogu.taltech.ee/et/Item/ab3b17af-b35d-4c46-b757-49ff4fd8b5ff.

[9] Risto Ruuben. *Lõputööde teemade haldamise rakendus*. [Accessed: 01-05-2024]. URL: https://digikogu.taltech.ee/et/Item/ec74957c-49b4-4254-a1fa-87688a1f04d2.

[10] *Ruby on Rails vs Django: Which is Right for You?* [Accessed: 27-04-2024]. URL: https://www.bairesdev.com/blog/ruby-on-rails-vs-django/.

[11] *Compress the complexity of modern web apps.* [Accessed: 01-05-2024]. URL: https://rubyonrails.org/.

[12] *Most used web frameworks among developers worldwide, as of 2023*. [Accessed: 01-05-2024]. URL: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/.

[13] *The Rails Doctrine.* [Accessed: 01-05-2024]. URL: https://rubyonrails.org/doctrine.

[14] *Sorbet*. [Accessed: 04-05-2024]. URL: https://sorbet.org/.

[15] *Pundit*. [Accessed: 04-05-2024]. URL: https://github.com/varvet/pundit.

[16] *RuboCop*. [Accessed: 04-05-2024]. URL: https://github.com/rubocop/rubocop.

[17] *RuboCop Rails*. [Accessed: 04-05-2024]. URL: https://github.com/rubocop/rubocop-rails.

[18] *Annotate (aka AnnotateModels)*. [Accessed: 04-05-2024]. URL: https://github.com/ctran/annotate_models.

[19] *Pagy*. [Accessed: 04-05-2024]. URL: https://github.com/ddnexus/pagy.

[20] *RSpec*. [Accessed: 04-05-2024]. URL: https://rspec.info/.

[21] *Audited*. [Accessed: 04-05-2024]. URL: https://github.com/collectiveidea/audited.

[22] *React A JavaScript library for building user interfaces*. [Accessed: 02-05-2024]. URL: https://legacy.reactjs.org/.

[23] Tallinna Tehnika Ülikool. *Taltech styleguide*. [Accessed: 01-05-2024]. URL: https://portal-dev.ttu.ee/styleguide.

[24] Tallinna Tehnika Ülikool. *Styleguide design prototype*. [Accessed: 02-05-2024]. URL: https://www.figma.com/file/G4eaZg8SwP2BDLp2BiC6yq/Design%3A-Styleguide?type=design&node-id=7190-18081&mode=design&t=6Ddp2PrrbduOlq7K-0.

[25] *i18next: what it is, why you should use it*. [Accessed: 02-05-2024]. URL: https://poeditor.com/blog/i18next/.

[26] *Redux*. [Accessed: 03-05-2024]. URL: https://redux.js.org/.

[27] *A Comprehensive Comparison of React State Management Libraries*. [Accessed: 02-05-2024]. URL: https://weber-stephen.medium.com/a-comprehensive-comparison-of-react-state-management-libraries-550a0e84c441.

[28] Nihar Raval. *TypeScript vs JavaScript: Know The Difference?* [Accessed: 03-05-2024]. URL: https://radixweb.com/blog/typescript-vs-javascript.

[29] *Scss*. [Accessed: 03-05-2024]. URL: https://sass-lang.com/.

[30] *@mixin and @include*. [Accessed: 18-05-2024]. URL: https://sass-lang.com/documentation/at-rules/mixin/.

[31] Nikhil Dhiman. *The Benefits of Using SCSS Over CSS*. [Accessed: 03-05-2024]. URL: https://www.kanakinfosystems.com/blog/benefits-of-using-scss-over-css#:~:text=In%20conclusion%2C%20SCSS%20brings%20a,%2C%20viable%2C%20and%20proficient%20stylesheets..

[32] *Vite Next Generation Frontend Tooling*. [Accessed: 04-05-2024]. URL: https://vitejs.dev/.

[33] DOMINIK BISIAKOWSKI. *Why we use Vite instead of Create React App*. [Accessed: 04-05-2024]. URL: https://makimo.com/blog/why-we-use-vite-instead-of-create-react-app/.

[34] *New to PostgreSQL?* [Accessed: 04-05-2024]. URL: https://www.postgresql.org/.

[35] *What is Docker?* [Accessed: 04-05-2024]. URL: https://www.docker.com/.

[36] *Use containers to Build, Share and Run your applications*. [Accessed: 04-05-2024]. URL: https://www.docker.com/resources/what-container/.

[37] *8 surprising facts about real Docker adoption*. [Accessed: 04-05-2024]. URL: https://www.datadoghq.com/docker-adoption/.

[38] Zainab Aftab. *What Is Scrum Development Strategy Its 5 Stages?* [Accessed: 02-05-2024]. URL: https://hapy.co/journal/what-is-scrum/.

[39] Nithin Kumar Peratla. *What is Agile Strategy? Implementation, Frameworks, Benefits*. [Accessed: 02-05-2024]. URL: https://www.knowledgehut.com/blog/agile/agile-strategy.

[40] *Bootstrap Layout*. [Accessed: 04-05-2024]. URL: https://getbootstrap.com/docs/5.0/forms/layout/.

[41] Chris Coyier. *A Complete Guide to Flexbox*. [Accessed: 04-05-2024]. URL: https://css-tricks.com/snippets/css/a-guide-to-flexbox/.

[42] *npm-cli-login*. [Accessed: 25-05-2024]. URL: https://www.npmjs.com/package/npm-cli-login.

[43] *Creating and viewing access tokens*. [Accessed: 25-05-2024]. URL: https://docs.npmjs.com/creating-and-viewing-access-tokens.

[44] *Dokumentatsioon*. [Accessed: 27-05-2024]. URL: https://gitlab.cs.ttu.ee/protsessor/protsessor-documentation/-/wikis/Home.

[45] *Feedback to Taltech StyleGuide team*. [Accessed: 18-05-2024]. URL: https://gitlab.cs.ttu.ee/protsessor/protsessor-documentation/-/wikis/Feedback-to-Taltech-StyleGuide-team.

Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis[1]

I Kauri Kinks, Raul Akerman, Artjom Gussev

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Rebuilding the Thesis Web Application "Protsessor"”, supervised by Ago Luberg and Juhan-Peep Ernits
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons’ intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

06.06.2024

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student’s application for restriction on access to the graduation thesis that has been signed by the school’s dean, except in case of the university’s right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.