TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Sedrik Suurmets 193297IAIB
Mikk Järvis 185656IAIB
Kaspar Ustav 193929IAIB

Lõputööde ja projektide haldamise infosüsteem Protsessor

Bakalaureusetöö

Juhendaja: Ago Luberg
PhD

Tallinn 2022

Autorideklaratsioon

Kinnitame, et oleme 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.

Autorid: Sedrik Suurmets, Mikk Järvis, Kaspar Ustav

30.05.2022

Annotatsioon

Selle bakalaureusetöö eesmärgiks on luua veebirakendus projektide haldamiseks, läbimisprotsessi jälgimiseks ja nende alamteemade leidmiseks, haldamiseks ning neile kandideerimiseks. Lõputöö käigus keskenduti Tallinna Tehnikaülikooli informaatika bakalaureuse ja magistri õppekavade lõpetamisprotsessi kasutusjuhtudele.

Meeskonnatööna valminud veebirakendus koosneb eraldi esi- ja tagarakendusest. Rakenduses sooritatavad kasutajapoolsed tegevused on defineeritud protsessiskeemidena ning nende käivitamiseks ja haldamiseks on kasutusel protsessimootor. Protsessimootori valikul analüüsiti selle vajalikkust ning erinevaid lahendusi. Rakendust kasutati 2021/2022 õppeaasta informaatika bakalaureuse õppekava lõputöö teemade leidmisel.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 40 leheküljel, 8 peatükki, 12 joonist, 1 tabelit.

Abstract

Thesis and Project Management Information System Protsessor

The purpose of this bachelor's thesis is to create a web application for managing projects, monitoring user project completion, and managing, finding and applying to project subtopics. The functionality is based on the Informatics bachelor's and master's study programmes' graduation process use cases.

The web application consists of separate front- and backend applications. Actions performed by users through the application are defined as business process diagrams. To start and manage these actions and diagrams, a process engine is used. To choose a process engine, its necessity and different options were analysed. The application was used to find thesis topics for the Informatics bachelor's study programme in the 2021/2022 academic year.

The thesis is in Estonian and contains 40 pages of text, 8 chapters, 12 figures, 1 table.

Lühendite ja mõistete sõnastik

AD	*(Azure) Active Directory*
API	*Application Programming Interface,* Rakendustarkvara liides
BPMN	*Business Process Modelling Notation*, Äriprotsesside modelleerimiskeel
CI/CD	*Continuous Integration and Continuous Development*, Pidevinegratsioon ja pidevvalmidus
CORS	*Cross-Origin Resource Sharing*
Dockeri konteiner	*Docker container,* eelpakitud rakenduse sõltuvusi ja rakendust sisaldav tarkvaratükk
Dockeri tõmmis	*Docker image,* Dockeri konteineri loomiseks kasutatavaid masinloetavaid käske sisaldav fail
IDE	*Integrated Development Environment*, Integreeritud programmeerimiskeskkond
JPA	*Java Persistance API*
JSON	*JavaScript Object Notation,* JavaScripti alamhulgal põhinev andmevahetusvorming
JWT	*JSON Web token*, standard andmete loomiseks koos digitaalallkirja ja krüpteerimisega
MVP	*Minimum Viable Product*, toote algkuju, mis rahuldab kliendi esialgseid vajadusi
NPM	*Node Package Management*, Javascripti paketihaldur
proxy	Server, mis on vahelüliks kliendi ja loodud süsteemi vahel
REST	*Representational State Transfer,* Arhitektuuri stiil
SPA	*Single Page Application,* Üheleheline veebirakendus
SQL	*Structured Query Language*, Standardiseeritud andmebaasi päringukeel
SSH	*Secure Shell Protocol,* Võrguprotokoll
YAML	*YAML Ain't Markup Language*, inimeste ja programmide poolt lihtsasti loetav keel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Lõputöö leidmine on midagi, millega iga lõpetav tudeng Tallinna Tehnikaülikoolis silmitsi seisab. Erinevate teaduskondade vahel on lõputöö leidmise ja tegemise protsess erinev, mis teeb ühise süsteemi loomise, kuhu oleks koondatud kõikide teaduskondade lõputööd ja rahuldaks kõiki kasutusjuhte, keeruliseks.

Lõputööde teemade leidmiseks puudub Tallinna Tehnikaülikooli informaatika tudengitel keskkond. Teemad lisatakse juhendajate poolt ühte tekstidokumenti ning tudeng peab leidma sealt sobiva teema või teemat otsima suheldes otse potentsiaalse juhendajaga. Samuti puudub õppekavajuhtidel ning kaitsmiskomisjonil võimalus saada ülevaadet tudengi teema leidmise protsessist või sellega seonduvatest raskustest, et saaks võimalikult kiiresti välja pakkuda potentsiaalseid lahendusi.

Praegune lahendus tekitab lisatööd tudengitele ning juhendajatele. Tudeng peab potentsiaalsete juhendajatega eraldi ühendust võtma. Selle tõttu on ka juhendajatel raskendatud teemast huvitunud tudengite haldamine. Juhendajad peavad haldama tekstidokumenti, kus on raske teemasid filtreerida, ning neil puudub lihtne ülevaade sellest, kes on nende teemade vastu huvi üles näidanud.

Tudengitele ja õppejõududele lõputöö teemade leidmise ja haldamise protsessi lihtsustamiseks loome veebirakenduse. See peab võimaldama teemade esitamist juhendajate poolt, teemadele kandideerimist, teemade väljapakkumist tudengite poolt ja teemade sidumist tudengite ning juhendajatega. Juhendajatel, lõputööde komisjonil ja muudel selleks määratud isikutel peab olema võimalus näha ülevaadet tudengitest, näiteks saavad nad võimalikult kiirelt teada, kui mõni tudeng on jäänud ilma teemata. Neid õiguseid peab olema võimalik kasutajatele lisada või eemaldada. Lisaks lõputööde protsessile peab infosüsteem olema võimalikult paindlik, et see arvestaks ka erinevate kursustega seotud teemade esitamist ja teemadele tudengite leidmist.

2 Projekti kirjeldus

Selles peatükis esitatakse projekti kirjeldus, lähtudes kasutusjuhtudest, mida loodav süsteem peab rahuldama. Analüüsitakse ka sarnaseid olemasolevaid lahendusi. Lisaks esitatakse põhjused protsessimootori kasutuselevõtuks ning analüüsitakse erinevaid protsessimootoreid.

2.1 Kasutusjuhud

Rakenduse arendamisel võtsime aluseks Tallinna Tehnikaülikooli informaatika bakalaureuse ja informaatika magistri õppekavade lõpetamise kasutusjuhud. Lisaks arvestasime, et loodav infosüsteem peaks olema piisavalt paindlik, et rahuldada hiljem ka teiste õppekavade ning teaduskondade kasutusjuhte. Töö käigus keskenduti peamiselt teema esitamise, leidmise ja kinnitamise kasutusjuhtudele.

Teema esitatakse üldjuhul juhendaja poolt, kes peab saama rakenduses teema esitada. Arvestada tuleb ka juhtudega, kus teema esitaja ei ole ise teema juhendaja, ega kuidagi muud moodi teemaga seotud. Teemal peab olema võimalik määrata mitu juhendajat – ka süsteemi väliseid juhendajaid. Süsteemis peab saama esitada teemasid nii üksinda tegemiseks kui ka tiimiga tegemiseks.

Teema leidmiseks peab olema tudengitel võimalus leida omale sobiv teema juhendajate poolt välja pakutud teemadest. Juhendajate poolt välja pakutud teemasid peab nägema avalikult, ehk ilma sisse logimata. Samuti peab tudengitel olema võimalik juhendajale teema ise välja pakkuda. Tudengi välja pakutud teema peab juhendaja süsteemis kinnitama.

2.2 Olemasolevad lahendused

Enne meie loodud rakendust oli kasutusel Microsoft Wordi tekstifail, kuhu oli ligipääs kõigile läbi Tallinna Tehnikaülikooli Uni-ID. Sinna said juhendajad lisada enda teemasid, soovi korral kasutades dokumendi alguses olevat tekstipõhja, mis aga ei olnud kasutajasõbralik. Märksõna järgi otsimine toimus läbi brauseri/rakenduse märksõna otsingu. Kandideerimise protsess oli lihtsalt soovitud lõputöö teema leidmisel juhendajaga ühenduse võtmine, enamasti meili teel ning see protsess pikenes kui ilmnes, et antud teemale olid juba tudengid olemas.

Juhendajaga otse suhtlemine on hea alternatiiv, sest tal võib olla teemasid, mida veel välja reklaamitud ei ole. Lisaks saad välja pakkuda endapoolse või pikema suhtluse korral leida tema abiga kellegi kolmanda pakutud teema. Meie rakendus ei päästa juhendajaga suhtlemisest, aga kiirendab oma teema pakkumise ja saadaval olevatele teemadele kandideerimise protsesse.

Osade õppekavade puhul on lõputööde teemad nähtaval Tallinna Tehnikaülikooli kodulehel. Kuna teemad on kodulehel tavalises tekstivaates, siis puudub võimalus tudengitel neid lihtsasti filtreerida. Lõputööde protsessi haldajatel ei ole kodulehel võimalust näha ülevaadet sellest, kuhu teemadele tudengid on üritanud kandideerida ning mis teemadele nad on vastu võetud. Meie rakendus võimaldab teemasid lisada ja muuta nii juhendajatel endil kui ka näiteks õppekava juhtidel või teistel isikutel, kellele selleks on õigus antud. Teemasid saab filtreerida erinevate parameetrite põhjal ning vastava õigusega kasutajatel on võimalik vaadata teemadega seotud tudengite, juhendajate ja kandideerimiste ülevaadet.

2.3 Protsessimootori analüüs

Kuna rakendus peab arvestama võimalikult paljude teaduskondade kasutusjuhtudega, siis tekkis vajadus luua rakendus sellisel viisil, et hiljem oleks uute kasutusjuhtude implementeerimine kiire ja lihtne. Leidsime, et üks võimalus on defineerida erinevad

tegevused rakenduses protsessiskeemidena – mida tellijad meile ka soovitasid. Järgnevalt on esitatud kolme protsessimootori analüüs.

2.3.1 Activiti

Activiti [1] on Javas kirjutatud protsessimootor, mille esimese versiooni Activiti 5.0 andis välja Alfresco nime kandev ettevõte aastal 2010. See oli esimene BPMN 2.0 spetsifikatsiooni järgiv protsessimootor, mis turule tuli. BPMN 2.0 tähendas, et protsessi skeemides on nüüd ka protsesside täitmise semantika, mida varem saavutati BPMN 1.0 ja BPEL (Business Process Execution Language) 1.0 abil. [2]

Activiti pakub lisaks protsessimootorile ka protsesside modelleerimiseks ja haldamiseks veebipõhist keskkonda. Võrreldes konkurentidega on Activiti aeglasemalt edasi arenenud ning konkurendid pakuvad autorite hinnangul mugavamat keskkonda protsesside modelleerimiseks ja haldamiseks. Samuti pakuvad konkurendid tuge suuremale hulgale BPMN 2.0 komponentidele.

2.3.2 Flowable

Flowable [3] sai alguse aastal 2016, kui grupp endiseid Activiti protsessimootori loojaid otsustasid Activiti projektist taanduda ning luua Activiti repositooriumist uue haru [4], millest arenes välja protsessimootor Flowable BPM (Business Process Management). [5]

Flowable pakub lisaks protsessimootorile erinevaid tööriistu, mis aitavad protsesside modelleerimisel (Flowable Modeler) ja nende haldamisel (Flowable IDM, Flowable Task). Eelnimetatud tööriistad tulevad kaasa Flowable *core engine'*iga, mis on pakett erinevatest Flowable teenustest ja API-dest. Töö autorid ei leidnud võimalust äriprotsesside modelleerimiseks loodud Flowable Modeler tööriista eraldiseisvana, ilma tervet Flowable *core engine*'t käima panemata käima panna.

2.3.3 Camunda

Camunda [6] on avatud lähtekoodiga Activiti põhjal loodud protsessimootor. Camunda protsessimootor sai alguse 2013 aastal, kui Camunda nime kandev konsultatsioonifirma

lõi Activiti koodihoidla põhjal uue haru [7], kuhu nad hakkasid arendama oma enda protsessimootorit Camunda. [8]

Camunda protsessimootori loojad suutsid luua paremini hallatava koodibaasi ja pakkuda tuge peaaegu kõigi BPMN 2.0 standardisse kuuluvate komponentide kohta. Samuti tekkis tugi juba alustatud protsessi instantside muutmiseks ja migreerimiseks uuematele protsessi versioonidele. [9]

Protsessimootor on ehitatud Java keeles ning on olemas täielik Spring Boot tugi, mida autorid rakenduse loomisel kasutasid. Camunda pakub ka protsesside, kasutajate ja gruppide haldamiseks veebipõhist töölauda Camunda Cockpit [10], mille olemasolu pidasime protsessimootori valikul tähtsaks. Samuti on loodud eraldi töölauarakendus Camunda Modeler [11], mida kasutatakse protsessiskeemide loomiseks ja millega genereeritud BPMN skeemid sisaldavad Camunda protsessimootori tööks vajalikke komponente.

Rakenduse loomisel valisimegi protsessimootoriks Camunda. Leidsime, et Camunda protsessimootor on võrreldes alternatiividega parema toe ja dokumentatsiooniga ning annab parema ülevaate protsessidest. Samuti osutus valikul tähtsaks eraldiseisev Camunda Modeler rakendus.

3 Projekti disain

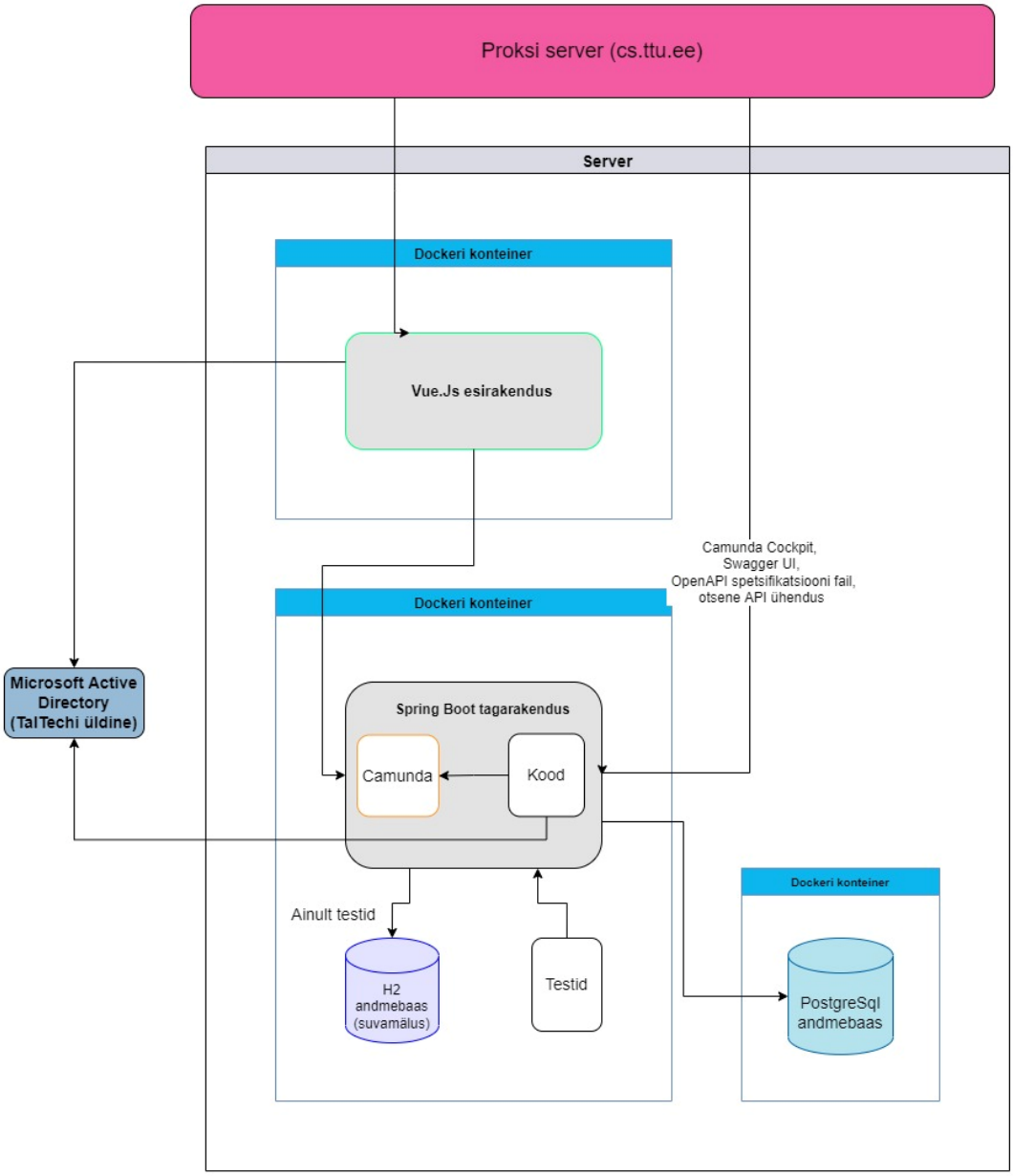
Rakendus koosneb kolmest suuremast osast: Spring Boot tagarakendus koos Camunda platvormi liidestusega, andmebaas ja Vue.Js SPA esirakendus. Iga osa jaoks on olemas serveri peal jooksev produktsiooni- ning arendusversioon, mis evitatakse automaatselt kasutades GitLabi CI/CD tehnoloogiat.

3.1 Arhitektuur

Esirakendusel, tagarakendusel ja andmebaasil on kolm eraldi keskkonda: produktsioon, arendus ja lokaalne arendus. Veebirakendus tervikuna kasutab samas keskkonnas asetsevat esirakendust, tagarakendust ja andmebaasi. Produktsioonikeskkond on kasutajatele avalik versioon rakendusest, arenduskeskkond on võimalikult sarnane produktsioonikeskkonnaga ning võimaldab arendajatel katsetada lisatud funktsionaalsust, lokaalses arenduskeskkonnas toimub aktiivne koodi kirjutamine.

Produktsiooni- ja arenduskeskkond asuvad ühes serveris ja kõik rakenduse osad jooksevad eraldi Dockeri konteinerites. Mõlemad keskkonnad on üksteisest sõltumatud ja nendes sisalduvad andmed tulenevad eraldi andmebaasidest. Serverile saab väljastpoolt päringuid teha ainult läbi proksi (ingl *proxy)* või SSH ühenduse. Avalik ühendus rakendusega luuakse läbi `cs.ttu.ee` proksi. Antud keskkonnad kasutavad kasutajate autentimiseks Tallinna Tehnikaülikooli üldist identiteedi- ja juurdepääsuhaldusteenust Azure Active Directory. Eraldiseisev produktsiooni- ja arenduskeskkond annab võimaluse arenduskeskkonnas muudatusi katsetada ning vigu avastada enne nende produktsioonikeskkonda lisamist.

Joonis 1 esitab produktsioonikeskkonna arhitektuuri skeemi. Antud joonisel välja toodud arhitektuur kehtib ka arenduskeskkonna puhul.



Joonis 1. Produktsioonikeskkonna arhitektuuri skeem.

Lokaalne arenduskeskkond käivitatakse iga arendaja arvutis ning eeldab esirakenduse ja tagarakenduse koodi alla laadimist. Samuti on vajalik lokaalselt töötav PostgreSQL instants. Teoreetiliselt on esirakendusel võimalik lokaalselt luua ühendus ka serveril jooksva tagarakenduse ning andmebaasiga. Kuna serveriga ühenduse saamiseks on vaja luua SSH tunnel või muuta CORS-i seadeid oleme lihtsuse mõttes otsustanud kasutada

lokaalselt täiesti eraldiseisvat rakendust. Samuti on eeliseks, et soovi korral saab esirakendust arendades koheselt testida võimalikke vajalikke tagarakenduse muudatusi. Kasutajate autentimiseks kasutab lokaalne keskkond eraldi Azure Active Directoryt, kuhu saame vajadusel lisada kasutajaid või muuta autentimisega seonduvat infot. Lisas 2 on esitatud lokaalse arenduskeskkonna arhitektuuri skeem.

3.2 Esirakendus

Otsustasime esirakenduse ehitada kasutades SPA põhimõtet. SPA on veebirakenduse implementatsioon, mis uuendab lehti komponentide kaupa JavaScripti koodi abil, vastupidiselt klassikalisele implementatsioonile, kus igale lehele vastab terve dokument. See tagab parema kasutajamugavuse, sest lehed saavad kiiremini laetud ja rakendus tundub ühtse tervikuna, sarnaselt töölauarakendusetele. Esirakendus oli koheselt plaanis ehitada kasutades mõnda üldtuntud esirakenduse kirjutamise raamistikku, mis enamasti juba vaikimisi toimivad SPA põhimõtetel ning see on ka üks põhjustest, miks esirakendus on SPA.

Esirakendus on üles ehitatud kasutades Node.js-il baseeruvat Vue.Js [12] (Vue) raamistikku. Lehed ja komponendid on Vue Options API-l põhinevad ning neid on lihtne organiseerida ning erinevatel lehtedel taaskasutada. Lehtede vaheline navigeerimine toimib vue-router teegi abil, mis võimaldab komponentide HTML koodi sisse lisada vue-routeri linke. Antud lingid ei uuenda tervet lehekülge vaid vastavalt SPA printsiibile uuendavad vajalikke komponente JavaScripti abil.

Algselt kasutasime esirakenduse kirjutamiseks JavaScripti, kuid nii tagarakenduse kui ka esirakenduse kasvades tegime koodi parema loetavuse, lihtsuse ja veakindluse tagamiseks otsuse kasutada TypeScripti. TypeScript [13] on JavaScriptil baseeruv, tugevalt tüübitud programmeerimiskeel. TypeScripti kood kompileeritakse ümber JavaScripti koodiks ning tänu sellele on seda võimalik käivitada igal pool, kus on JavaScripti tugi. Tugev tüüpimine ja lisasüntaks tagab võrreldes JavaScriptiga parema liidestatuse IDE-dega. Kompilaator ning IDE liidestatus aitab märgata ja ennetada koodis tekkivaid vigu.

3.2.1 Liidese disain

Liidese ülesehitamiseks kasutasime Element Plus komponentide teeki ning vastavalt vajadustele kohandatud CSS-i. Disainielemendid ja komponendid on suuremas osas kõik Element Plusi valmiskomponendid, mida on vajadusel vähesel määral muudetud. Isekirjutatud CSS-i kasutasime põhiliselt komponentide joondamiseks ning paigutamiseks.

Lähtusime süsteemi oleku nähtavuse kasutajaliidese heuristikast [14]. Kasutajatel peaks olema kohene tagasiside nende sooritatud tegevustele. See hõlmab endas näiteks vormide valideerimist täitmise ajal, teavitusi või märguandeid andmete salvestamise õnnestumise või ebaõnnestumise kohta ja visuaalseid vihjeid kui toimub andmete laadimine.

Kuna rakendus oli reaalselt kasutuses informaatika eriala 2022. aasta kevadiste bakalaureusetööde teemade leidmiseks, siis oli meil võimalik saada kasutajatelt tagasisidet nii liidese kui ka rakenduse funktsionaalsuse kohta. Vastavalt sellele lisasime uut ning parandasime olemasolevat disaini ja funktsionaalsust.

3.2.2 Autentimine

Rakendus on seotud Tallinna Tehnikaülikooli üldise Azure Active Directoryga ning kasutajad saavad süsteemi sisse logida Uni-ID kontoga. Esirakendus loob ühenduse Azure autentimisteenusega, kust kaudu kasutaja saab sisestada sisselogimiseks vajalikud andmed. Autentimisteenusest vastuseks saadud JWT salvestatakse brauseri mälusse ning antakse läbi autoriseerimise päise (ingl *authorization request header*) kaasa kõigile päringutele, mis esirakendusest tagarakendusse saadetakse. Selle eluiga on 1 tund, aga lisaks tuleb kaasa nö värskendustooken, millega saab uuendada autentimise tookenit automaatselt. Hetkel uuendatakse autentimise tookenit 6 minutit enne eluea lõppu.

3.2.3 Ühendus tagarakendusega

Päringud tagarakendusse saadetakse HTTP protokolli abil. Tagarakendusega ühenduseks vajalik serveri klient genereeritakse automaatselt kasutades OpenAPI Generatorit [15] ning tagarakenduse OpenAPI 3 [16] spetsifikatsiooni faili. Klient genereeritakse

TypeScripti keeles ning see baseerub Axios HTTP serveri kliendil. Lisaks kliendile genereeritakse ka TypeScripti tüübid, mida kasutatakse tagarakenduse API otspunktide sisenditena või väljunditena. Genereerimine vähendab kirjutatava koodi hulka ja tagab selle, et tüübid ja API meetodid on esi- ja tagarakenduses omavahel vastavuses.

Genereerimiseks on esirakenduses defineeritud skriptid, mida jooksutatakse automaatselt rakenduse ehitamisel (ingl *build*) ning mida on võimalik ka käsitsi käivitada. Võrreldes vajaliku koodi ise kirjutamisega on genereerimine palju paindlikum. Kuna ehitasime esi- ja tagarakendust samaaegselt, siis oleks kõigi tüüpide ja URL-ide lisamine ning muutmine olnud väga ajakulukas ning suurendanud vigade tekkimise ohtu.

3.2.4 Mitmekeelsuse tugi

Esirakendust on võimalik kasutada eesti ja inglise keeles, mis on oluline selle tõttu, et mitmed Tallinna Tehnikaülikooli õppejõud ja tudengid on rahvusvahelise päritoluga ning seetõttu ei pruugi osata eesti keelt. Mitmekeelsuse võimaldamiseks kasutame Vue i18n teeki. Tõlked on ära määratud eesti ja inglise keelele eraldi JSON formaadiga failides. See võimaldab võtmetele vastavateks väärtusteks määrata nii teksti kui ka alamobjekti. Lauseid, fraase või sõnu saab koodis kasutada vastavalt tõlkefailides määratud JSON-i võtmetele. Defineerida saab ka keelepõhiseid numbrite ja kuupäevade formaate. Iga tõlkefaili JSON elementide võtmed peavad omavahel kattuma, et koodis oleks võimalik olenemata kasutaja poolt valitud keelest neile ühte moodi refereerida.

Rakenduse stiilimiseks kasutatav Element Plus raamistik on samuti mitmekeelsuse toega ning on sünkroonis meie rakenduses seadistatud keele valimise loogikaga. See tähendab, et Element Plusi valmiskomponentides sisalduvad tekstid on vastavalt kasutaja keelevalikule meie rakenduses samas keeles. Näiteks tabelis, kus andmed puuduvad, kuvatakse sellekohane tekst kasutaja valitud keeles.

3.2.5 Äriprotsesside kasutajategevuste vormid

Tagarakenduse äriprotsesside skeemides on defineeritud kasutajapoolsed tegevused, mis põhiliselt kujutavad endast vormide täitmist. Antud vormid on defineeritud HTML vormingus ning saadetakse esirakendusse koos teiste tegevusega seotud andmetega.

Oleme üritanud esirakenduses tegevuste kuvamise muuta võimalikult dünaamiliseks. Tänu sellele ei ole iga uue äriprotsessi skeemi lisatava kasutajapoolse tegevuse jaoks vaja esirakenduses luua uut vormi, vaid saab näidata tagarakendusest kaasa tulevat HTML vormingus vormi ning sisestatud andmed tagarakendusele edastada. Staatiline HTML vorm ühendatakse Vues defineeritud HTML elementide sisse kasutades `v-html` funktsionaalsust.

Enamus olemasolevad tegevuste vormid oleme esirakenduses implementeerinud käsitsi, st mitte kasutades tagarakenduses olemasolevaid HTML faile, Vue ja Element Plus abil, et meil oleks parem kontroll kasutaja sisendite üle. See võimaldab näiteks sisendit koheselt peale sisestamist valideerida, vajaduseta andmeid selleks tagarakendusse saata. Vue põhiselt vormi ülesehitamine võimaldas sellele lisada mitmekeelsuse toe. Samuti oleme osad väiksemad kasutajategevused sel viisil saanud muuta üheks suuremaks vormiks. Element Plus pakub palju paindlikumat ja laiemat sisenditüüpide valikut kui äriprotsessi skeemid ning selle tõttu on vorme olnud võimalik muuta kasutajale arusaadavamaks ning visuaalselt ilusamaks. Samuti on mitmekeelsuse tugi Element Plusi sisseehitatud.

3.3 Tagarakendus

Meie tagarakendus on üles ehitatud Spring Booti [17] peale, mis on selle arendajate poolt seadistatud Spring raamistik, kuhu on nende poolt lisatud terve hulk vajalikke teeke. Tehnoloogia valik tulenes sellest, et see on Java-põhine ning me oleme erinevatel kursustel sellega kokku puutunud, seega oli meil juba varasem kogemus nii rakenduste loomisel kui ka evitamisel. Lisaks on rakendusse integreeritud Camunda Platform. Testid on kirjutatud kasutades Junit 5, Mockitot ja Spring Booti sisseehitatud funktsionaalsust. Sõltuvuste haldamiseks, rakenduse loomiseks ja jooksutamiseks on kasutatud Gradle'it.

Üks väga vajalik integratsioon on veebiserver Apache Tomcat, mis pakub Java-põhise rakendusega suhtlust läbi HTTP ning tänu sellele saame toetuda REST põhimõtetele. Mainimist väärt on ka Lombok, mis aitab vähendada tüüpsisu (ingl *boilerplate*) tänu annotatsioonidele, mis on täpselt sama mustri järgi nagu Spring ja Hibernate.

3.3.1 Ühendus andmebaasiga

Andmebaasiga suhtleb Hibernate, mis on Spring Booti sisseehitatud JPA implementatsioon, mis on vahelüli Java objektide ja relatsioonilise andmebaasi vahel. Kaardistades Java objektid ja nendega seotud toimingud SQL tabeli ridadeks ja lauseteks. Loomulikult on see vastastikune suhtlus. Spring JPA lisab funktsionaalsust Hibernate'ile ning ei vaja liialt koodi kirjutamist, sest sisaldab valmis kirjutatud reegleid.

Meie endi loodud JPA mudelite pärimine on väga lihtne, kasutame seotud andmehoidla (ingl *repository*) liidest, millel on lihtsamad päringud sisseehitatud. Meil on väga palju kasutusel ka endi kirjutatud päringuid, mida saab kasutada Spring JPA annotatsioonidega. Ühtlasi on võimalik neid kirjutada puhtas SQL-is kui ka JPA endi päringukeeles – JPQL-is. Viimane on eriti mugav, sest kasutab Javale omast OOP-i stiili.

Suureks abiks on ka JPA-sse integreeritud lehekülgedena päringule vastamine (ingl *pagination*). Oluline on just sellepärast, et esirakenduses käib projektide ja teemade kuvamine samal põhimõttel, et vältida liigset andmebaasi ning serveri koormust.

3.3.2 Camunda

Põhilised kasutajapoolsed tegevused käivitavad rakenduses protsessi, mis on kirjeldatud BPMN skeemidena ja mille käivitab ning haldab Camunda protsessimootor. Prosesside modelleerimiseks kasutame Camunda Modeler graafilist liidest. Camunda Modeleri eelis alternatiivide ees on selle liidestus Camunda protsessimootoriga. Skeemil saab viidata otse Java klassidele ja muutujatele, mis teeb tööprotsessi kiiremaks ja lihtsamaks.

Kasutajapoolsete tegevuste vormide koostamiseks kasutame Camunda Modeleri pistikprogrammi „UserTask Generated Form Preview and Embedded Form Generator" [18], mille abil Camunda Modeleris koostatud vormid genereeritakse automaatselt HTML vormiks. Neid genereeritud vorme saavad kasutajad täita nii Camunda veebiliideses kui ka läbi meie loodud eesrakenduse.

3.3.3 Koodi muster

Kõik HTTP otspunktid on kirjeldatud kontrollerinimeliste Java klassidega, mis suhtlevad esirakenduse või mistahes vajaliku osapoolega. Nad saadavad ja võtavad info vastu ning suhtlevad rakenduse siseste teenustega (ingl *services*). Neis toimub andmete töötlemine (otse või teiste teenuste abil), kontrollimine ja vajadusel uute andmete andmehoidlatest pärimine. Andmehoidla on liides, millele laiendatakse Spring JPA andmehoidla liidest, mis ühtlasi seob varasemalt loodud mudelid andmebaasi tabelitega.

Eraldiseisvalt on meil veel Camunda protsessid, millele igale vastab oma bpmn faililaiendiga XML-kirje ning vajadusel on lisaks Java klassid ja HTML vormid. Java klassidele laiendatakse Camunda loodud liidest, mis lubab seda XML-s välja kutsuda ning tegeleb teatud tegevuste automatiseerimisega (*service task*), kaasa arvatud otse tagarakendusega suhtluseks.

3.3.4 Autentimine

Kasutajate autentimiseks kasutame Microsoft Azure Active Directoryt, millel põhineb Tallinna Tehnikaülikooli Uni-ID süsteem. Active Directory põhineb OAuth 2 protokollil. Microsofti serveriga suhtluseks on kasutusel Azure Spring Boot Active Directory teek.

Azure kasutaja identifikaatori põhjal salvestatakse meie rakenduse andmebaasi sama identifikaatoriga kasutaja. Lokaalse Camunda kasutajaga on võimalik paindlikult andmeid siduda, näiteks gruppidesse kuuluvust määrata.

Produktsiooni- ja arenduskeskkonnad kasutavad Tallinna Tehnikaülikooli üldist Azure Active Directory süsteemi, mis võimaldab Uni-ID omanikel ennast meie rakenduses autentida. Lokaalne arenduskeskkond on seadistatud kasutama eraldi AD süsteemi, mille üle meie omame kontrolli. See tagab võimaluse vabalt luua ja muuta kasutajaid ning testida seadistusi.

3.3.5 Õiguste haldus

Rakenduses on kasutusel gruppide põhine õiguste süsteem. Gruppidesse kuuluvused määravad ära erinevad õigused süsteemis. Õiguste halduse oleme püüdnud võimalikult

paindlikuks ja lihtsaks teha. Gruppide põhine õiguste süsteem võimaldab grupile anda rakenduse siseseid õiguseid. Kasutaja õiguste kontrollimisel saab kontrollida tema gruppidesse kuuluvust. Gruppide liikmeid on võimalik lihtsasti lisada ja eemaldada, mis tagab õiguste määramise paindlikkuse ja dünaamilisuse. Samuti vähendab see rakenduse koodi ja andmebaasi keerukust, sest igale kasutajale ei ole vaja eraldi õiguseid määrata.

Tervet infosüsteemi puudutavad õigused on võimalik seadistada administratiivõigustega kasutajal Camunda Admin paneelil. Sisse logimisel saavad kasutajad esialgsed õigused Azure AD gruppide põhjal.

Projekti loojal on võimalik rakenduses projektisiseseid õiguseid hallata otse esirakenduse kaudu projekti seadete vaates. Järgnevalt on esitatud õigused, mida kasutajal on võimalik projekti seadetes kasutajatele määrata:

- `VIEW` - Õigus projekti ja avalike teemad nägemiseks
- `EDIT` - Õigus projekti seadete ja sisu muutmiseks
- `SUBMIT` - Õigus teemasid esitada
- `PROPOSAL_SUBMIT` - Õigus teema välja pakkuda, mille peab kinnitama juhendaja
- `PROPOSAL_VIEW` - Õigus välja pakutud teemasid näha.
- `APPLY` - Õigus avalikele teemadele kandideerimiseks
- `PROPOSAL_APPLY` - Õigus välja pakutud teemadele juhendajaks kandideerimiseks.

Teema raames on samuti erinevatel kasutajatel erinevad õigused. Avalikke teemasid saavad näha kõik kasutajad (ka sisse logimata kasutajad). Teema muutmise õigused antakse teema esitajale ja juhendajatele. Kui teemale on tudengid kinnitatud, siis antakse neile teemaliikme õigused (ingl *member access*), mis võimaldab tudengitel ligipääsu oma teemale sõltumata teema nähtavusest.

3.3.6 OpenAPI dokumentatsioon

API meetodite ja nende poolt kasutatavate andmemudelite dokumentatsioon genereeritakse automaatselt läbi Springdoc OpenAPI teegi. Antud teek suudab lugeda meetodite sisendeid ja väljundeid ning on võimalik spetsiifiliselt ära defineerida HTTP tagastuskoodide põhjuseid ja tähendusi. Samuti on võimalik meetodeid ja andmemudeleid kirjeldada inimloetavas keeles. Enamus antud meetoditest on kirjeldatud Javadoc kommentaaride abil, ning lisateek Springdoc OpenAPI Javadoc suudab kommentaaride põhjal dokumentatsiooni lisada kirjeldusi. Koodi kirjeldada on võimalik kasutades ka Springdoci sisseehitatud annotatsioone.

Dokumentatsioon on saadaval nii Swagger kasutajaliidese kujul, mis võimaldab ka meetodeid lihtsasti testida, kui ka spetsifikatsiooni failidena JSON või YAML vormingus. Just failipõhine genereeritud dokumentatsioon on kasutusel meie esirakenduses. Tagarakenduse API-ga ühenduseks vajaminev kood ning andmemudelite tüübid genereeritakse automaatselt just OpenAPI dokumentatsiooni põhjal.

3.3.7 Emailide saatmine

Rakendus on seadistatud võimalusega välja saata emaile läbi Spring Boot Starter Mail teegi. Saatmiseks on kasutusel Tallinna Tehnikaülikooli Onyx SMTP server.

Emailide saatmise põhieesmärk on teavitada kasutajaid, kui mõne teise kasutaja sooritatud tegevus nõuab nendepoolset edasist tegevust või omab mõju nende varasemalt sooritatud tegevustele. Email saadetakse näiteks teema juhendajatele, kui nende teemale on laekunud kandideerimine, ja kandideerijatele, kui nende kandideerimisele on vastatud.

3.4 Andmebaas

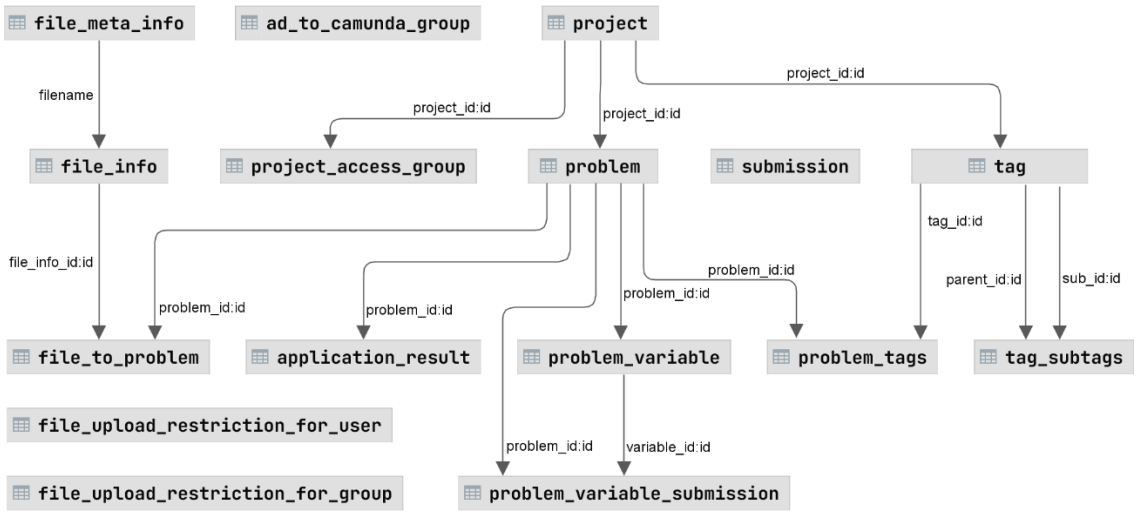
Nii arendamisel kui ka produktsioonis on kasutusel PostgreSQL läbi Spring JPA. PostgreSQL sai valitud, sest on objekt-relatsiooniline andmebaasisüsteem, mis toetab tugevat tüüpimist. See kasutab laiendatud SQL-i keelt, mis lisab turvalisust ja andmebaasi skaleeruvuse võimekust tõstvat lisafunktsionaalsust. Andmebaasis on koos nii meie endi

loodud (vt Joonis 2) kui ka Camunda toimimiseks vajalikud tabelid. Serveri peal jooksevad arenduse ja produktsiooni andmebaasid Dockeri konteinerites, mis salvestavad andmed serveri peale kasutades Docker köiteid (ingl *volume*).

Automaattestid jooksevad läbi H2 andmebaasi, mis käivitatakse muutmälus. See sisaldab ühilduvuskihte erinevate andmebaasidega, sealhulgas PostgreSQL-iga, mis võimaldab meil kasutada spetsiifilist SQL süntaksit. Testide käivitamisel loetakse tühja andmebaasi näidisandmed, mis on kirja pandud SQL formaadis. Suvamälus jooksvat andmebaasi on lihtne käivitada automatiseeritud pidevintegratsiooni töövoogudes, mis rakenduse teste jooksutavad.

Tänu Liquibasele saab muuta tabeleid automaatselt ilma andmebaasi taaskäivitamata. Sobib suurepäraselt produktsioonis veergude, kitsenduste jms lisamiseks või muutmiseks. Seadistamiseks on eraldi YAML-fail ning meil on kasutusel SQL süntaksiga failid muudatuste jaoks. Viimaste täideviimist hoitakse meeles andmebaasi loodud tabelis, et vältida probleeme kas mitmekordse või puuduliku jooksutamisega.

Joonis 2 esitab tagarakenduse andmebaasi skeemi meie loodud tabelitele (Camunda tabeleid on ligi 50 ja need ei ole seotud meie looduga).



Joonis 2. Andmebaasi skeem.

3.5 CI/CD

Rakenduse automaatseks ehitamiseks, testimiseks ja paigaldamiseks on loodud taga- ja esirakenduses pidevintegratsiooni töövood (CI/CD *pipeline*'id). Pidevintegratsiooni töövood on loodud, et automatiseerida tegevusi, mida kasutaja peaks vastasel juhul rakenduse testimisel ja paigaldamisel ise tegema.

CI/CD implementeerimiseks kasutame GitLab CI/CD tööriista. Tööriist võimaldab luua erinevaid koodihoidla põhiseid töövoogusid, seadistada väliseid keskkondi skriptide jooksutamiseks (GitLab Runner), hoida tundlikke andmeid keskkonnamuutujates (CI/CD variables) ja erinevate keskkondade ülevaadet ning haldust, kuhu rakendus paigaldatakse (Environments). [19]

3.5.1 Tagarakendus

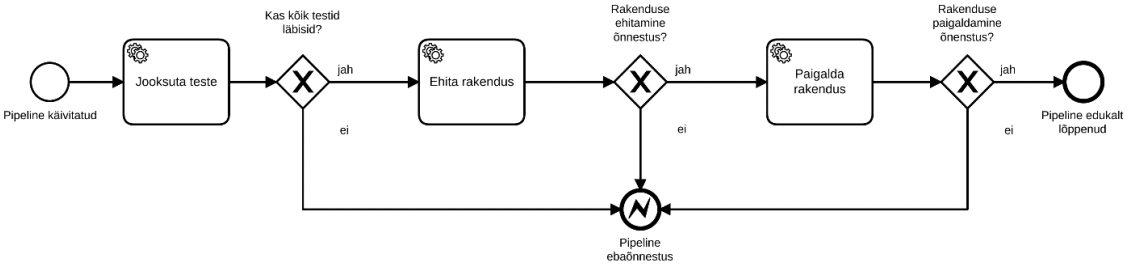
Tagarakenduse GitLabi salves on loodud gitlab-ci.yml skriptifail, kus on defineeritud testimise, ehitamise ja paigaldamise faasid. Iga kehtestus (ingl *commit*) `develop` või `master` harusse käivitab töövoo, mis koosneb kõikidest defineeritud CI/CD protsessidest. Testid käivitatakse igal kehtestustusel igas harus.

Testimise käigus jooksutatakse kõik rakendusele loodud ühiktestid ja API testid, ning kui mõni test läbi ei lähe, siis on töövoog ebaõnnestunud ning järgmiste protsesside juurde edasi pole võimalik liikuda. Kui mingis harus testid läbi ei lähe, siis seda haru pole ka võimalik teiste harudega mestida (ingl *merge*).

Rakenduse ehitamise käigus luuakse Dockeri tõmmis, mis laetakse üles GitLabi Container Registrysse [20].

Rakenduse paigaldamise käigus tõmmatakse GitLab Container Registryst Dockeri tõmmis GitLab Runnerisse, mille järel rakendus konteineriseeritakse ning konteiner pannakse serveri peal jooksma koos vajalike köidetega (ingl *volume*). Paralleelselt pannakse ka serveri peal jooksma `postgres` konteiner vajalike köidetega, mille peal jookseb rakenduse andmebaas.

Joonis 3 esitab tagarakenduse salve CI/CD töövoo skeemi.



Joonis 3. Tagarakenduse CI/CD töövoo skeem.

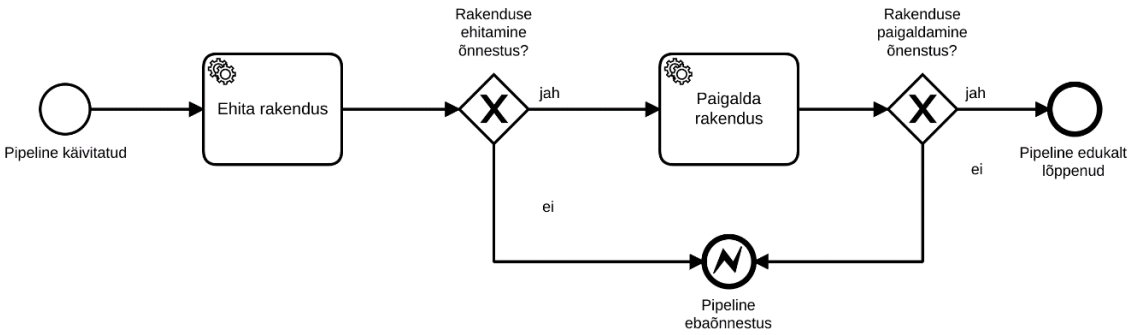
3.5.2 Esirakendus

Esirakendus GitLabi salves on loodud gitlab-ci.yml skriptifail, kus on defineeritud ehitamise ja paigaldamise faasid. Iga kehtestus `develop` või `master` harusse käivitab töövoo, mis koosneb kõikidest defineeritud CI/CD faasidest.

Rakenduse ehitamise käigus luuakse Dockeri tõmmis, mis laetakse üles GitLab Container Registrysse.

Rakenduse paigaldamise käigus tõmmatakse GitLab Container Registryst Dockeri tõmmis GitLab Runneri-sse, mille järel rakendus konteineriseeritakse ning konteiner pannakse serveri peal jooksma.

Järgnevalt on esitatud eesrakenduse salve CI/CD töövoo skeem



Joonis 4. Esirakenduse CI/CD töövoo skeem.

4 Tööprotsess

Meil oli igal koosolekul arutelu mida, kas ja kuidas teha. Eelkõige lähtusime kasutajate tagasisidest ja kliendi soovidest. Iga uue probleemi või funktsionaalsuse panime kirja eraldi GitLabi piletina (ingl *issue*). Tiimi koosolekutel arutlesime funktsionaalsuse implementeerimise üle ning üle nädala koostasime 2-nädalase tähtpunkti (ingl *milestone*) ja jagasime piletid ära.

Esirakendusel puuduvad automaattestid, aga koodi ülevaatus toimub nii GitLabis kui ka suuremate muutuste puhul jooksutame koodi enda arvutis. Tagarakendusel on üldiselt iga funktsiooni jaoks kirjutatud testid ja koodi üle vaatamine toimub sarnaselt esirakendusele. Testid said kirjutatud enda arendatud funktsionaalsusele paralleelselt.

4.1 Tööjaotus

Rakenduse disainimisel ja kirjutamisel ei olnud tiimiliikmetele kindlalt ära määratud, mis valdkonnaga nad tegelema peavad. Kõik said tegeleda nii andmebaasi, taga- ja esirakenduse ning Camunda protsesside disainimise ja kirjutamisega. Selle tõttu oli igal tiimiliikmel võimalik õppida erinevaid tehnoloogiad ja tarkvaraarenduse printsiipe ning vajadusel võimalik teiste liikmete käest nõu küsida.

4.1.1 Suvi 2021

Alustasime kaheliikmelise tiimina praktika vormis. Esimene pool kulus uute tehnoloogiate õppimisele. Suve lõpuks olid olemas projekti ja teema esmased andmemudelid, algversioon teemade lisamisest ja sirvimisest. Puudus teemade ja projektide õiguste halduse süsteem, mida oleks võimalik dünaamiliselt kontrollida.

4.1.2 Sügissemester 2021

Jätkasime tarkvaraarenduse projekti aine raames. Tiimi kolmanda liikmena liitus Sedrik. Eesmärk oli rakendus saada kasutajatele minimaalse funktsionaalsusega kasutatavaks. Minimaalseks funktsionaalsuseks pidasime lõputöö teemade lisamist ning muutmist, nende sirvimist ja grupipõhist kandideerimist. Samuti oli vaja hallata kasutajate õiguseid, kes saavad teemasid või projekte lisada ja muuta.

Panustasime ka koodikvaliteedi parandamisele. Tagarakenduses hakkasime kirjutama teste, esirakenduse viisime JavaScripti keelelt üle TypeScriptile, tööülesandeid planeerisime läbimõeldumalt ja koordineeritult. Enne iga uue koodi muutuse lisamist koodihoidla põhiharudesse testis tiimikaaslane funktsionaalsuse käsitsi üle ning vaatas üle kirjutatud koodi. Need sisse viidud muutused jätkusid bakalaureusetöö kirjutamise lõpuni.

4.1.3 Kevadsemester 2022

Kevadsemestri alguses läks rakendus kasutusse informaatika eriala bakalaureusetööde teemade leidmiseks. Algselt tuli tegeleda palju vigade silumisega, mis reaalsete kasutajatega välja tulid. Tänu kasutajate tagasisidele oli võimalik analüüsida ning planeerida uue funktsionaalsuse lisamist.

Suuremad eesmärgid olid näiteks: kandideerimise protsessi dünaamiliseks muutmine, projektipõhine kasutajate ja teemade juhendajate statistika vaade, õppekavade ja rollide määramine läbi AD gruppide, failide üleslaadimise võimaldamine.

4.2 Töövahendid

Tabel 1 esitab töövahendid, mida arendamisel töö autorid kasutasid.

Tabel 1. Töövahendid.

IntelliJ Idea	IDE, algselt arendatud Java-põhiste keelte jaoks, aga saab kasutada nii taga-, esirakenduse kui ka andmebaasi jaoks
Webstorm	IDE, spetsialiseeritud veebi arendamise jaoks
Camunda Modeler	Graafiline tööriist Camunda BPMN skeemide loomiseks.
GitLab	Töö- ja ajahaldus tööriist, koodihoidla
Putty	Kaugühenduse tööriist. Kasutasime ühenduse loomiseks serveriga, kus rakendus paikneb.
Docker	Virtualiseerimise programm, mis lubab tarkvara jooksutada konteinerites.
Postman	Platvorm API arendamiseks ja testimiseks.
Swagger UI	Graafiline interaktiivne tagarakenduse dokumentatsioon, läbi mille saab teha päringuid tagarakenduse API otspunktide vastu.

5 Funktsionaalsus

See peatükk kirjeldab valminud rakenduse põhifunktsionaalsusi. Lahti on seletatud kasutajate, gruppide, teemade ja projektide haldus ning loomine. Kirjeldatakse ka tudengi ja juhendaja poolset kandideerimisprotsessi. Lisaks on esitatud kasutaja gruppi lisamise ja kandideerimisel käivituvate protsesside protsessiskeemid.

5.1 Kasutajad

Tallinna Tehnikaülikooli kasutajakontoga esimest korda süsteemi sisse logides luuakse rakenduses Camunda kasutaja, salvestades kasutaja ees- ja perekonnanime, meiliaadressi ning Uni-ID. See võimaldab Camunda protsesse otse kasutajale määrata ja gruppidesse lisada [21]. Esialgsed kasutaja andmed saadakse TalTech Azure AD-st. Azure AD kasutaja rollide põhjal lisatakse kasutaja ka rakenduses õiguste gruppidesse. Azure AD rollide põhjal eristame näiteks õpetajad ja tudengid ning jagame tudengid eraldi õppekavade kaupa gruppidesse.

Igal kasutajal on esirakenduses avalik profiilivaade. Profiilivaates näeb kasutaja ees- ja perekonnanime, meiliaadressi ning kirjeldust. Kõiki eelnimetatud andmeid saab sisse loginud kasutaja enda profiilivaates muuta. Enda profiilivaates näeb sisse loginud kasutaja gruppe, projekte ja teemasid, mille liige ta on. Samuti näeb kasutaja oma aktiivseid ning varasemaid kandideerimisi. Teise kasutaja profiilivaates näeb sisse logitud kasutaja teise kasutaja neid teemasid, gruppe ja projekte, mille liikmed mõlemad on.

5.2 Grupid

Gruppide jaoks kasutame rakenduses Camunda gruppide implementatsiooni. Camunda gruppe on võimalik luua ja hallata Camunda Admin paneelis [22] ning meie loodud esirakenduses. Gruppe saab luua läbi meie esirakenduse. Esirakenduses loodud grupid on

kõik tüübiga `USER_GROUP`. Kõik grupid, kuhu kasutaja kuulub, on nähtavad esirakenduse navigeerimisribal gruppide valikus.

5.2.1 Grupi tüübid

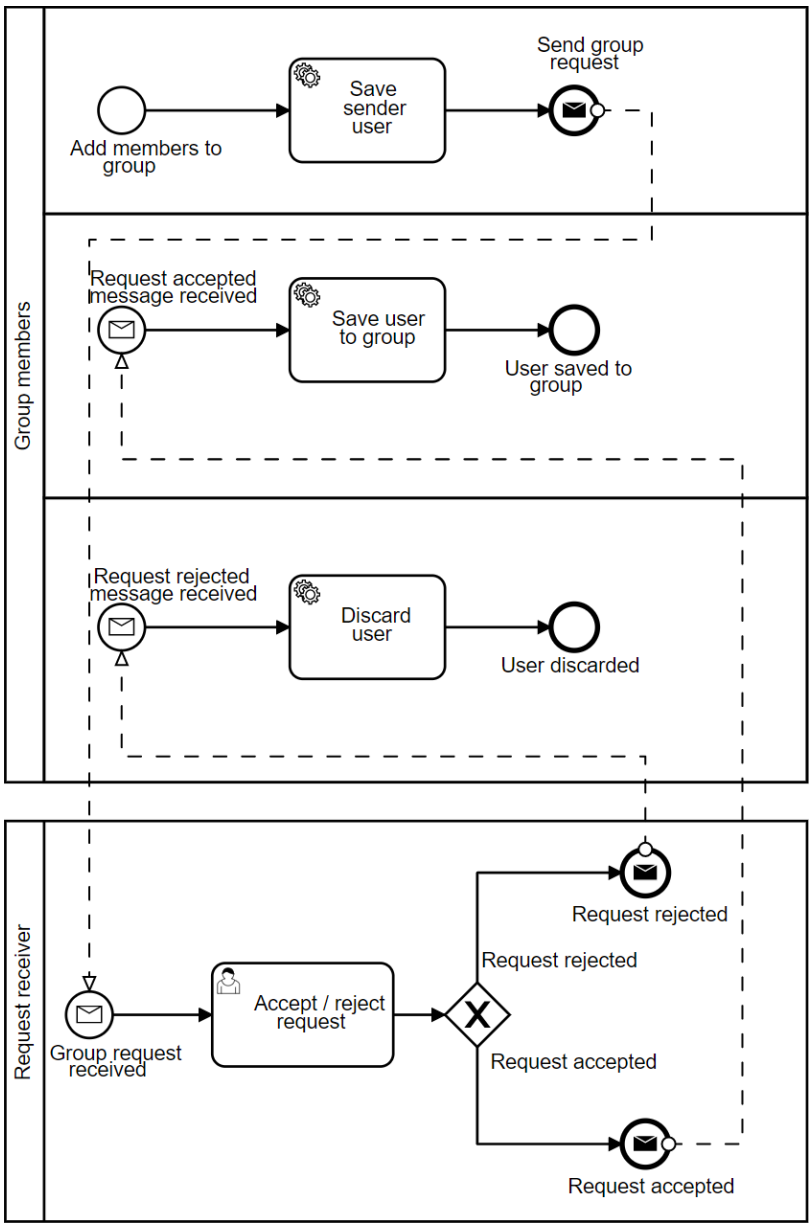
Järgnevalt on esitatud erinevad grupi tüübid ja nende omapärad, mis rakenduses kasutusel on:

- `AUTH_ROLE` - seda tüüpi grupid on rakenduses kasutusel õiguste haldamiseks. `AUTH_ROLE` grupid luuakse automaatselt ning neid ei saa kasutaja ise luua. Autenditud kasutaja saab need grupid omale külge Spring Security `GrantedAuthority` tüüpidena ning neid saab kasutada tagarakenduses `preAuthorize` annotatsioonides.
- `USER_GROUP` – Seda tüüpi on kõik grupid, mida kasutajad saavad läbi esirakenduse luua. Seda tüüpi gruppe saab kasutada teemadele kandideerimiseks.
- `PROBLEM_SUPERVISORS` - seda tüüpi grupid luuakse rakenduses automaatselt ning kasutaja ei saa neid ise luua. Grupp seotakse teema identifikaatori järgi teemaga ning kõigil selle grupi liikmetel on grupiga seotud teemal eriõigused ja juhendaja roll.
- `PROJECT_USERS` - seda tüüpi grupid luuakse rakenduses automaatselt ning kasutaja ei saa neid ise luua. Grupp seotakse mingi konkreetse projektiga. Kõigil selle grupi liikmetel on automaatselt projekti nägemise õigused, ehk nad on projekti liikmed.
- `PROBLEM_ASSIGNED_USERS` - seda tüüpi grupid luuakse rakenduses automaatselt ning kasutaja ei saa neid ise luua. Grupp seotakse mingi konkreetse teemaga. Grupi liikmeks saab läbi teemale kandideerimise või juhendaja poolse otse lisamise. Selle grupi liikmetel on seotud teemaga teema nägemise õigused sõltumata teema nähtavusest.
- `STUDY_ROLE` – seda tüüpi grupid luuakse rakenduses automaatselt ning grupi liikmed määratakse Azure AD-s määratud õppekavadesse kuuluvuse järgi.

5.2.2 Grupi liikmete haldus

Läbi esirakenduse ei saa liikmeid gruppidesse otse lisada, vaid kasutajale, keda soovitakse lisada, saadetakse kutse. Kutse on kasutajal võimalik kas vastu võtta või tagasi lükata. Kutse vastu võtmisel saab kasutaja grupi liikmeks. Kutse tagasi lükkamisel tühistatakse kutse ning kasutajat ei lisata gruppi.

Joonis 5 esitab BPMN skeemi, mis käivitub kasutaja gruppi lisamisel:



Joonis 5. Kasutaja gruppi lisamise BPMN skeem.

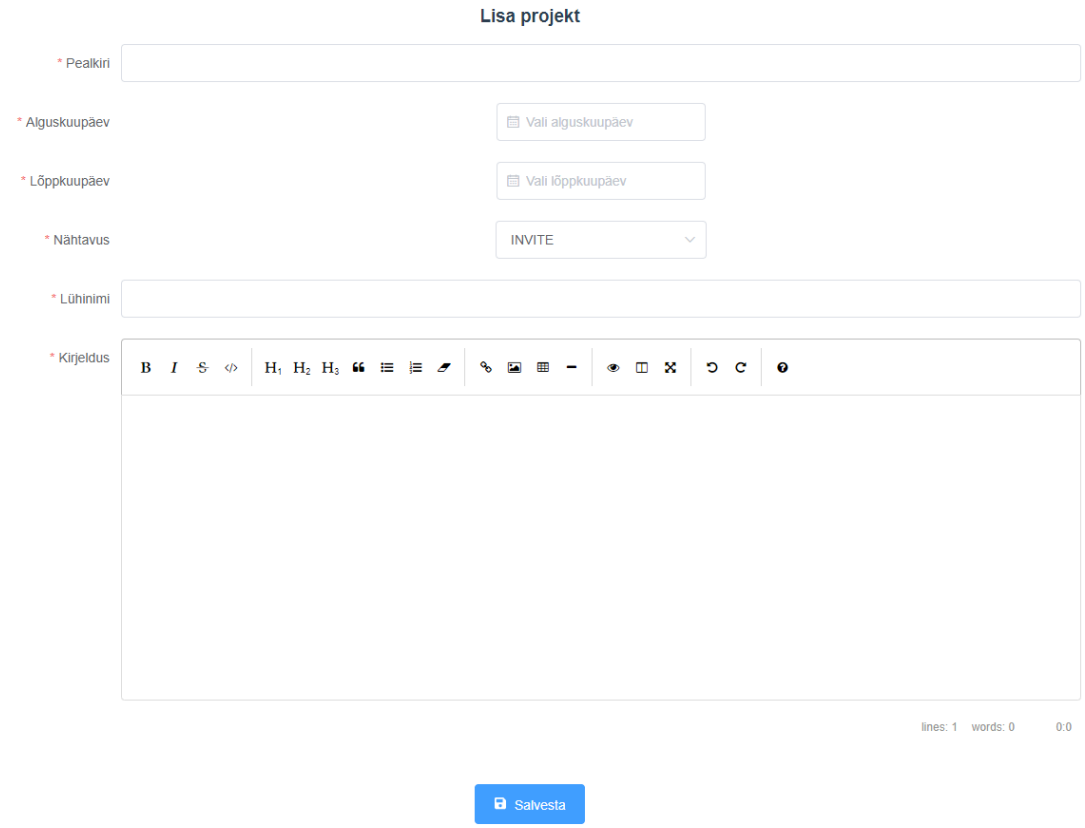
5.3 Projektid

Rakenduses saavad kasutajad luua ja hallata projekte. Lihtsustatul öeldes on projektid meie rakenduses teemade kogumid, kuhu kasutajad saavad teemasid esitada. Iga projekti jaoks loob süsteem automaatselt projektiga seotud kasutajate grupi, mille liikmetel on projekti nägemise õigus.

5.3.1 Projektide lisamine

Rakenduses saavad projekti loomise õigusega kasutajad lisada luua uusi projekte. Projekte saab lisada läbi meie esirakenduse või Camunda Tasklist paneeli kaudu käivitades vastava Camunda protsessi. Projekti loomisel määratakse projekti nähtavus, pealkiri, algus- ja lõppkuupäev, nähtavus, lühinimi ja kirjeldus. Kirjelduse väli toetab Markdown formaati.

Joonis 6 esitab esirakenduse projekti loomise vormi.



Joonis 6. Projekti loomise vorm.

5.3.2 Projekti seaded

Projekti muutmisõigustega kasutajatel on võimalik muuta projekti seadeid projekti seadete lehel esirakenduses või Camunda Tasklist paneelis, käivitades vastavaid Camunda protsesse. Projekti seadetes on eraldi sektsioonid projekti sisu, projekti liikmete, projekti õiguste ja projekti siltide muutmiseks.

Projekti seaded oli loodud eesmärgiga luua kasutaja vajaduspõhiseid projekte. Seadete abil saab lisada projekti tegevusi ja täpsustada, kes neid tegevusi käivitada saab. Veel saab seadetest määrata projekti nähtavust, kutsuda kasutajaid projektiga liituma, luua silte ja arhiveerida projekt.

Milliseid tegevusi projekti raames millised kasutajad teha saavad, määrab projekti õiguste seadistus. Projekti õiguste seadetes on võimalik erinevatele gruppidele anda projekti raames erinevad õigused. Gruppide valik koosneb `AUTH_ROLE`, `STUDY_ROLE` ja `USER_GROUP` tüüpi gruppidest. `USER_GROUP` tüüpi grupi valikuks peab kasutaja ise vastava grupi liige olema.

5.3.3 Projekti statistika

Projekti ülevaateks on loodud projekti statistika leht. Lõputööde projekti näite puhul on lõputöö protsessi haldamisega seotud kasutajatel suur huvi iga tudengi teema kohta, juhendajate teemadest ning juhendatavate tudengite arvust juhendaja kohta. Projekti statistika leht koondab tähtsamad andmed ühte kohta ning annab kasutajale hea ülevaate otse rakenduses tabelina või allalaetava `csv` failina.

Igal projektil on oma statistika leht, millele pääsevad ligi kõik projekti muutmisõigustega kasutajad. Statistika lehel näeb kokkuvõtet kõigi projekti liikmete kohta või juhendajate kohta eraldi. Nii juhendate kui ka projekti liikmete statistikat on võimalik samalt lehel alla laadida `csv` failina.

5.4 Teemad

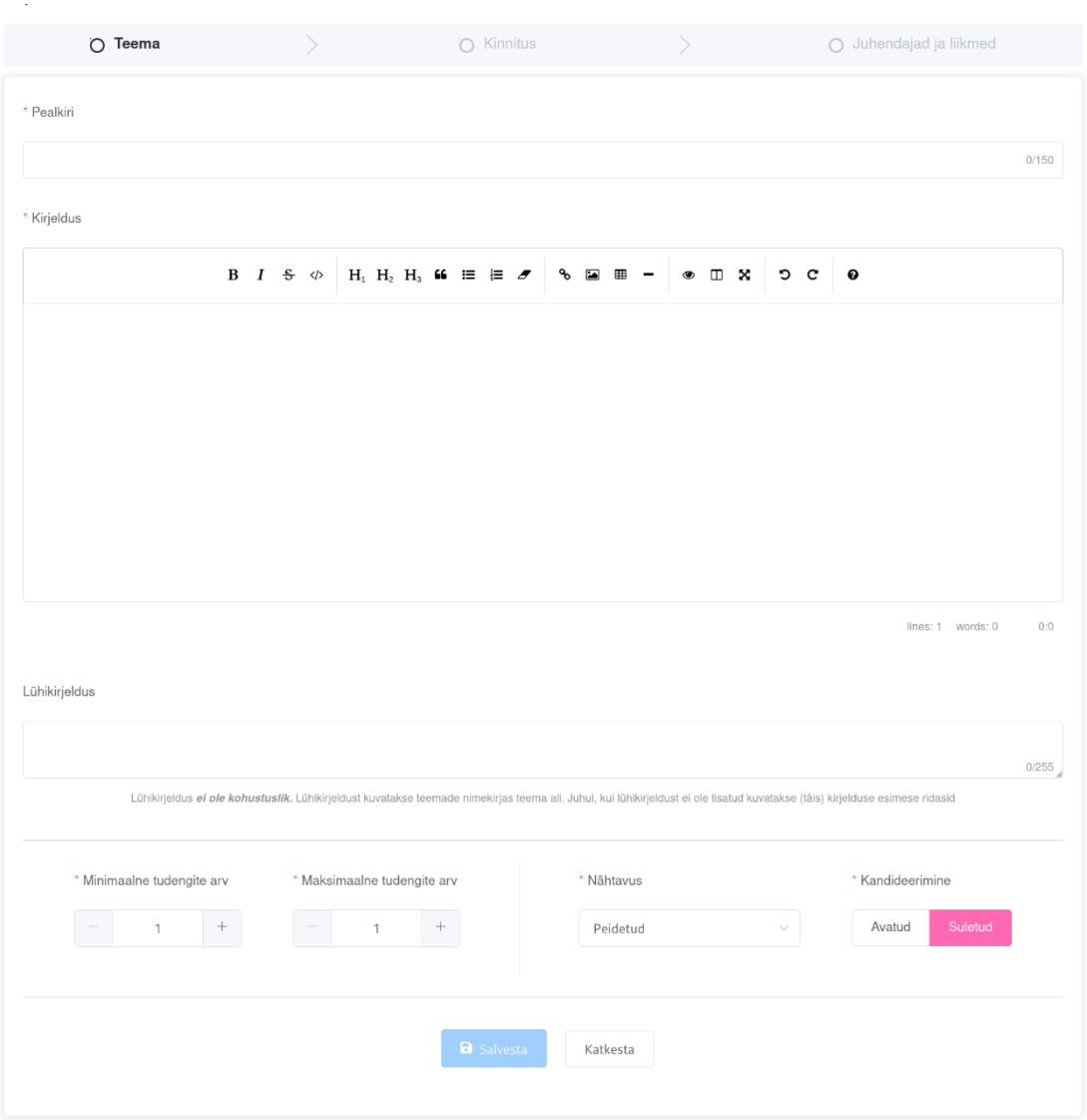
Rakenduses saavad vastava õigustega kasutajad projekti sees esitada ning välja pakkuda teemasid. Mõlemaid tegevusi saab täita läbi meie esirakenduse kui ka Camunda Tasklist paneeli käivitades vastavaid Camunda protsesse.

5.4.1 Avaliku teema lisamine

Teema lisamisel sisestab kasutaja teema pealkirja, kirjelduse Markdown formaadis, lühikirjelduse, minimaalse ja maksimaalse lubatud tudengite arvu, nähtavuse ja kandideerimise avatuse. Kui teema andmed on sisestatud on võimalus teemale kohe juhendajaid ja liikmeid lisada.

Teema lisamise õnnestumise korral salvestatakse teema ning seotakse teema projektiga, kuhu see lisati. Nähtavuse “Avalik” korral muutub teema projekti avalike teemade lehel teistele kasutajatele nähtavaks. Enda varasemalt lisatud teemasid saab kasutaja teise projekti üle viia ilma andmeid uuesti sisestamata.

Joonis 7 esitab esirakenduse teema lisamise vormi.



Joonis 7. Teema lisamise vorm.

5.4.2 Teemade välja pakkumine

Teema välja pakkumine tähendab sellise teema esitamist, mille juures esitaja soovib ise olla töö tegija ning millel juhendaja veel puudub. Teema välja pakkumisel sisestab kasutaja teema pealkirja, kirjelduse Markdown formaadis ning lühikirjelduse.

Teema välja pakkumise õnnestumise korral ei ole teema kõigile projekti liikmetele koheselt nähtav, kuna teemal puudub veel juhendaja. Kui teemale on juhendaja leitud,

siis saab juhendaja teema kinnitada, pärast mida muutub teema avalikuks ja teistele projekti liikmetele nähtavaks.

5.4.3 Teema seaded

Teema seadetes saavad teema muutmisõigustega kasutajad muuta teema pealkirja, kirjeldust, lühikirjeldust, minimaalset ja maksimaalset lubatud tudengite arvu, nähtavust, kandideerimise avatust, teema juhendajaid ning teemale määratud tudengeid. Lisaks saab teemale määrata vastava projekti all loodud silte, mis on eelkõige mõeldud teemade otsimise lihtsustamiseks. Muutmisõigustega kasutajad on teema juhendajad ning avaliku teema esitaja.

Piiratud muutmisõigustega kasutajad ei saa muuta teema juhendajaid ning teemale määratud tudengeid. Piiratud muutmisõigus on teema välja pakkunud kasutajatel.

5.5 Teemale kandideerimine

Teemale kandideerimine tähendab kasutajapoolset soovi olla kas välja pakutud teema puhul teema juhendaja või avaliku teema puhul teemale määratud tudeng. Kandideerimiste arvul piirangut ei ole ning kasutajal võib olla aktiivseid kandideerimisi samaaegselt mitmeid.

5.5.1 Avalikule teemale tudengiks kandideerimine

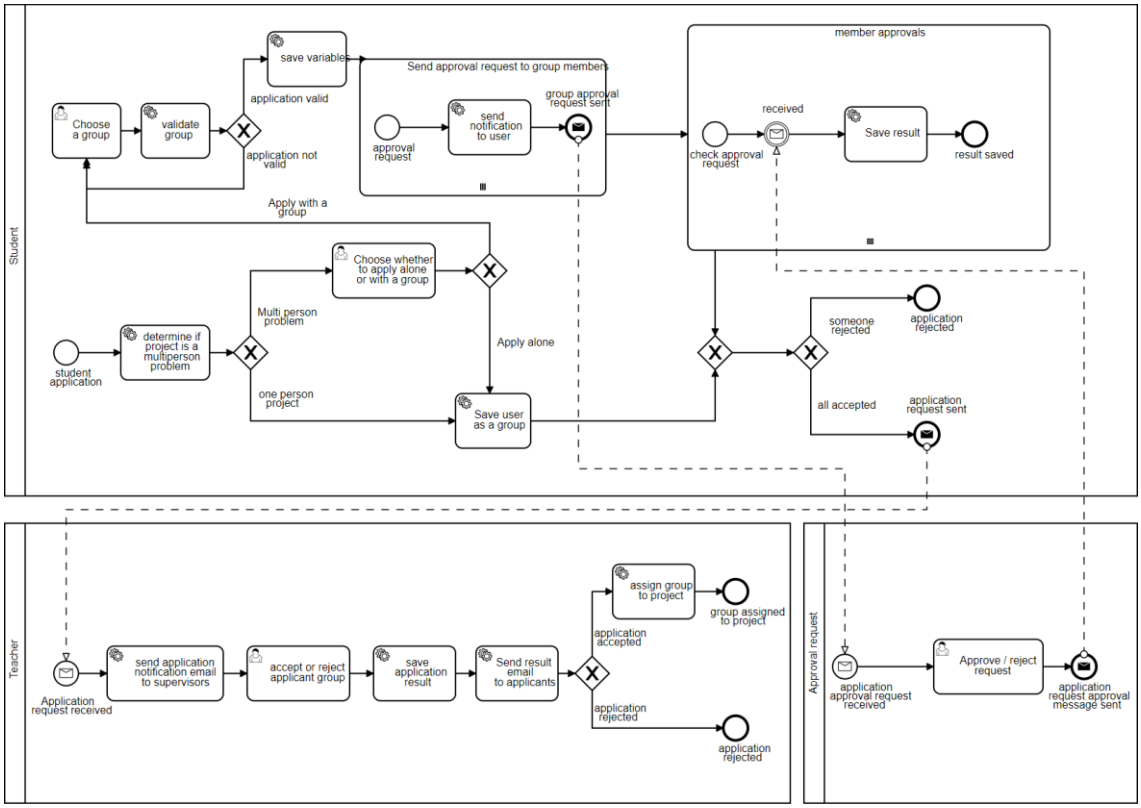
Avalikule teemale saab kandideerida esirakenduse kaudu teema lehelt või käivitades Camunda Tasklist paneelil vastavad Camunda protsessi. Teemale saab kandideerida üksi või koos tiimiga.

Üksinda kandideerimise korral saadetakse juhendajatele meili kaudu teavitus ning kandideerimine tehakse juhendajatele nähtavaks nii teema lehel, projekti lehel kui ka esilehel. Kandideerimise esitajal on võimalik igal ajahetkel enne juhendajapoolset kinnitamist kandideerimine tagasi võtta. Juhendajal on võimalik kandideerimine kas vastu võtta või tagasi lükata. Vastu võtmise või tagasi lükkamise korral on juhendajal võimalik lisada vastavasse tekstilahtrisse ka enda poolne selgitus. Tagasi lükkamise

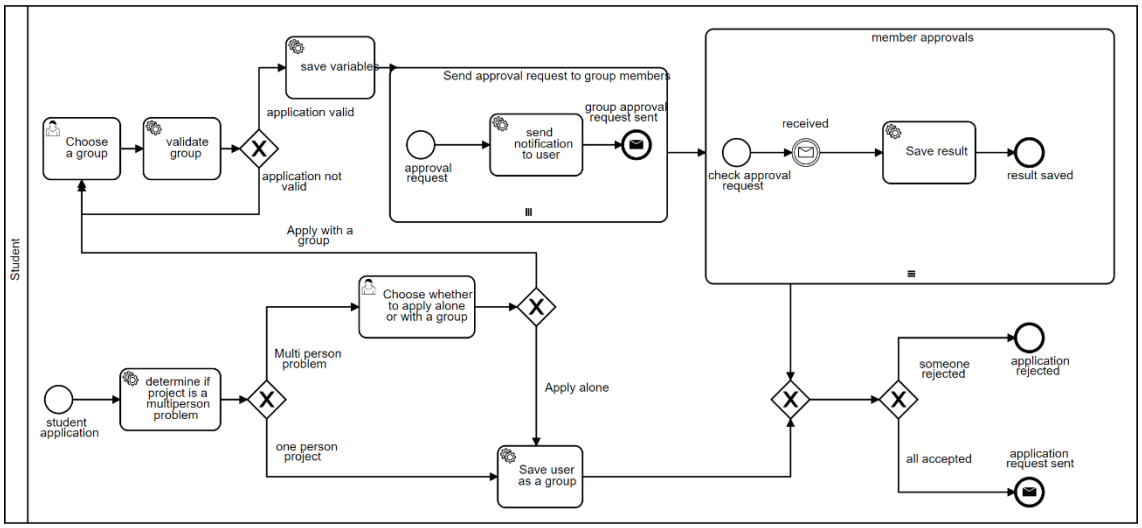
korral tühistatakse kandideerimine. Vastu võtmise korral lisatakse kandideerimise esitaja teemale määratud tudengiks.

Tiimiga kandideerimise korral peab kasutaja kandideerimise vormis sisestama grupi, mille liikmed määravad ära tiimi, kellega teemale kandideeritakse. Enne kui juhendaja kandideerimisest teada saab, peavad kõik tiimi liikmed veel omalt poolt kandideerimise kinnitama. Kui kasvõi üks tiimiliige kandideerimist vastu ei võta, siis kandideerimine tühistatakse. Kui kõik grupi liikmed on kandideerimise kinnitanud, siis saadetakse kandideerimine juhendajatele kinnitamiseks. Juhendajad näevad kandideerimisvormil kõiki tiimi liikmeid ja võimalust kandideerimine vastu võtta või tagasi lükata. Kandideerimise vastu võtmise korral määratakse teemale kõik tiimi liikmed.

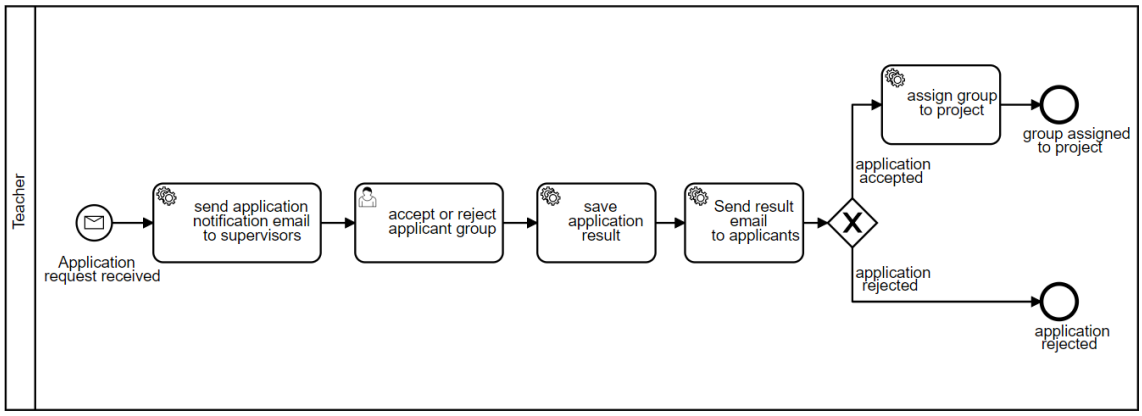
Joonis 8 esitab täieliku avalikule teemale kandideerimise BPMN skeemi. Joonis 9,10 ja 11 esitavad avalikule teemale kandideerimise BMPN skeemi tegutsejate kaupa.



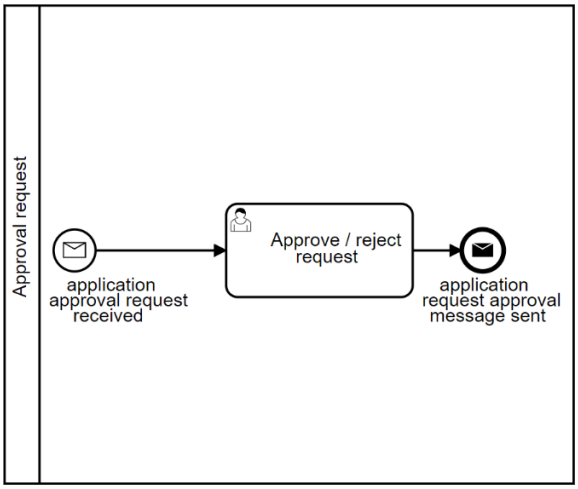
Joonis 8. Avalikule teemale kandideerimise BPMN skeem.



Joonis 9. Avalikule teemale kandideerimise alustaja tegevused.



Joonis 10. Avalikule teemale kandideerimise juhendajapoolsed tegevused.



Joonis 11. Avalikule teemale kandideeriva grupi liikmete tegevused.

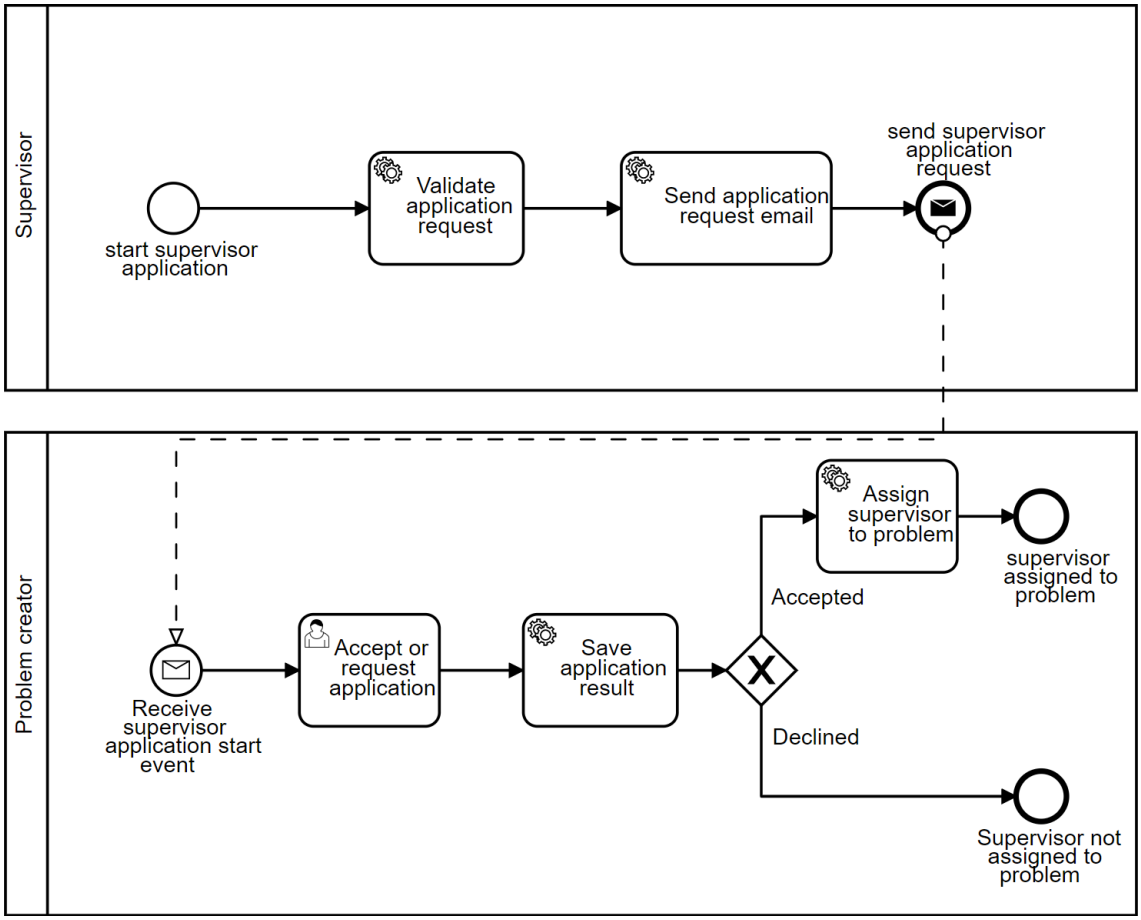
5.5.2 Välja pakutud teemale juhendajaks kandideerimine

Välja pakutud teemale saab juhendajaks kandideerida, kui teema on tehtud nähtavaks ning projekti seadetes on antud kasutajale vastav õigus.

Teemale saab juhendajaks kandideerida esirakenduse kaudu välja pakutud teema lehelt või käivitades Camunda Tasklist paneelil vastavad Camunda protsessi. Juhendajaks saab teemale kandideerida ainult üksinda.

Kandideerimise esitamisel saadetakse teema välja pakkujale meili kaudu teavitus ning kandideerimine tehakse kasutajale nähtavaks nii teema lehel kui ka esilehel. Kandideerimise kätte saajal on võimalik kandideerimine vastu võtta või tagasi lükata. Vastu võtmise või tagasi lükkamise korral on juhendajal võimalik lisada vastavasse tekstilahtrisse ka enda poolne selgitus. Kandideerimise tagasi lükkamise korral tühistatakse kandideerimine. Vastuvõtmise korral määratakse kandideerimise esitaja teemale juhendajaks.

Joonis 12 esitab välja pakutud teemale juhendajaks kandideerimise BPMN skeemi.



Joonis 12. Välja pakutud teemale juhendajaks kandideerimise BPMN skeem.

6 Valideerimine

Rakenduse kirjutamise alguses anti meile ette nõuded, mida terviklik lõputööde infosüsteem peaks sisaldama. Iganädalastel koosolekutel juhendajaga arutasime olemasolevat funktsionaalsust ning vajaminevat või teostatavat lisatavat funktsionaalsust. Nende põhjal lõime endale realistliku tegevuskava (ingl *roadmap*), mis oli jaotatud ühe- kuni kahekuulisteks alamperioodideks.

Koosolekutel juhendajaga, harvemini ka klientidega, vaatasime üle alates eelnevast koosolekust valmis saanud funktsionaalsused. Juhendaja või klient katsetas need enda poolt üle ning andis meile tagasisidet, mille põhjal oli võimalik teha parandusi või lisada midagi uut juurde.

Kord kuus vaatasime üle ka tegevuskavas kirjas olevad ülesanded ning kontrollisime, kas need on vastavalt kirjeldusele implementeeritud. Peale väiksemate alamülesannete valmimist vaatasid ka teised tiimikaaslased need üle ning kontrollisid nõutele vastavust. Selle põhjal oli võimalik sisse viia muudatusi enne, kui kood jõudis arenduse või produktsiooni koodiharusse.

Tagarakenduse alamülesannete funktsionaalsust valideerisime ka automaattestide abil. Testid kirjutasime vastavalt alamülesande kirjeldusele ja nõutele. Lisaks sellele testisime ka erinevaid piirjuhte (ingl *edge case*).

Meie rakendus oli kasutusel 2021/2022 õppeaasta informaatika bakalaureuse lõputööde leidmisel. Kokku esitati lõputööde projekti 84 teemat ning 53 tudengit oli projektis määratud mõne töö tegijaks. Saamaks konstruktiivset tagasisidet meie loodud rakenduse kohta koostasime rakenduse kasutajatele küsitluse vormid ja tagasisidevormi, mille vastuseid on analüüsitud.

6.1 Rakenduse tagasiside

Lisasime Microsoft Formsi küsitluse tekkinud probleemide või küsimuste jaoks, mis oli ligipääsetav esirakenduse jaluses. Kokku saime 16 vastust, millest enamus olid veateadetega seotud. Vastuse ja serveri logi järgi oli võimalik üles leida täpne murekoht ning probleem lahendada.

6.2 Tagasiside küsitlus

Küsitlus oli jagatud kaheks – üks tudengitele, teine õppejõududele. Tudengite küsitlusele vastas 13 ning õppejõudude küsitlusele 4 inimest.

Tudengite käest tahtsime põhiliselt teada kui paljud on meie rakenduse kaudu omale teema leidnud, üldist tagasisidet ning viise, kuidas kandideerimise protsessi paremaks teha. 8 tudengit 13st vastasid, et leidsid meie rakenduse kaudu omale lõputöö ning kõik nad olid enne kandideerimist juhendajaga ise ühendust võtnud. 3 tudengit vastanutest leidsid just läbi meie rakenduse omale lõputöö. 7 tudengit oleks soovinud läbi meie rakenduse ka omale tiimi leida, sealhulgas 3 neist arvas, et teemale kandideerijaid võiks näha avalikult. Küsisime tudengite käest ka millist funktsionaalsust nad meie rakenduses näha sooviksid. Näiteks sooviti teemade filtreerimiseks ja otsimiseks siltide süsteemi, mis sai ka hiljem implementeeritud.

Õppejõudude käest soovisime tagasisidet teemade lisamise ning kandideerimise kohta. Samuti tahtsime paremini mõista, kuidas seni on vastajad oma teemadele tudengeid leidnud. Kõik 4 vastajat lisasid ise oma teema(d) meie rakendusse ning 3 vastanut ütles, et teema lisamine oli pigem arusaadav ning 1 vastas, et teema lisamine oli täiesti arusaadav. Õppejõude, kelle teemadele kandideeriti läbi meie rakenduse ilma, et õppejõuga oleks enne ühendust võetud, oli 1 vastanutest.

7 Järeldused ja edasine arendus

See peatükk analüüsib tehtud tööd ning esitab autorite poolseid soovitusi rakenduse edasiseks arenduseks. Osad edasiseks arenduseks plaanitud soovituslikud funktsioonid ja muudatused on kirja pandud ka GitLabis piletitena.

7.1 Äriprotsesside modelleerimine

Üks peamiseid põhjuseid protsessimootori kasutamisele võtmisel oli erinevate õppekavade lõputöö leidmise ja tegemise protsessi selge defineerimine BPMN skeemi abil ning hiljem ülevaade kasutajate seisust selles protsessis. Loodud rakenduses aga puuduvad skeemid, mis kirjeldaks ühes suures skeemis ära lõputöö protsessi alates teema leidmisest kuni lõpetamiseni.

Suured skeemid tükeldasime väiksemateks skeemideks, näiteks käsitleme teemale kandideerimist täiesti eraldi protsessi ja skeemina. Siiski järgivad kõik loodud BPMN skeemid selgelt BPMN 2.0 standardit ning nende hiljem kokku panemine üheks suureks skeemiks ei tekita suuri takistusi.

Mitme väiksena protsessiskeemi kasuks otsustasime ka veel sellepärast, et kui kasutajad saavad luua loodud rakenduses ise uusi projekte, siis seaks see projekti seadistamisele teatud piirid ette. Kui me oleks loonud süsteemi, kus projektil on kohe algselt seotud mõne suurema protsessiskeemiga, siis olukord, kus kasutaja talle sobivat protsessiskeemi ei leia, tähendab arendajate poole pöördumist ja arendajate poolt spetsiaalselt sellele kasutajale loodud protsessiskeemi koostamist. Väiksemate protsesside abil on võimalik kasutajatel projektis ise määrata, milliseid protsesse saab keegi projekti raames käivitada ja täita.

Mida rohkem protsesse kasutajatel võimalik käivitada on, seda keerukamaks muutub kasutajatel uute projektide seadistamine. Sedasi tekib vajadus projekti esialgset seadistust

lihtsustada. Töö autorid näevad üheks rakenduse edasiarendamise võimaluseks projekti tüübi malle, mida valides tehakse esialgne projekti seadistus kasutaja eest ära. Sedasi ei pea näiteks iga aasta uut lõputööde projekti luues seda uuesti seadistama, vaid vastav lõputööde projekti mall on juba defineeritud ja kasutajad saavad selle põhjal uue projekti luua.

7.2 Teenuste tükeldamine

Meie tagarakendus on ehitatud ühe Camunda protsessimootori peale ühte rakendusse suureks monoliidiks. See on loonud olukorra, kus ühtegi API päringut ei saa teha ilma, et kõik Camunda teenused samuti jookseksid ning tulevikus loodavad uued teenused peavad kõik ehitama selle sama protsessimootori peale. Erinevate teenuste Camunda protsesse võiks olla võimalik hallata ühest eraldiseisvast teenustest.

7.3 Autentimine

Autentimine ei tööta sajaprotsendiliselt nii nagu ta võiks. Hetkel valitakse automaatselt viimasena sisse loginud kasutaja, aga mitme sisselogitud kasutaja korral peaks tulema ette valik kellena tahad jätkata. JWT uuendamine võiks ka paremini töötada, hetkel kontrollitakse iga 5 minuti tagant, kas see aegub 6 minuti jooksul.

Tihti välja logides ei suunata Microsofti lehelt rakendusele. Lisaks aegajalt tuleb ette olukordi, kus sisse logides küsib autentimisteenus hoopis, millise kasutajaga tahetakse välja logida. Harva võib ka juhtuda, et peale sisse logimist, mingil põhjusel rakendus ei saa täit URL-i, s.t saab `cs.ttu.ee/services/protsessor` asemel `cs.ttu.ee` ning seetõttu suunatakse kasutaja hoopis Tarkvarateaduse Instituudi TalTechi lehele.

7.4 Andmete hoiustamine

Meil otseselt pole implementeeritud andmebaasi varundamist. Mõned korrad sai kasutatud `pg_dump` (PostgreSQL-i meetod andmete varundamiseks faili) enne kui mingi

suurem andmebaasi muudatus sai tehtud. Plaan oli kasutusele võtta automaatne varundamine paar korda nädalas.

Produktsioonis tuli paar korda ette, et rakenduse taaskäivitamisel Spring Boot ei suutnud andmebaasi tabelis kas uut veergu või vaikimisi väärtusi lisada ega muuta. Teinekord oli vaja jooksvalt ka lisada andmebaasi kitsendusi, aga neid pidi igakord käsitsi lisama nii lokaalselt, arenduskeskkonnas kui ka produktsioonis. Liquibase varasem lisamine aidanuks vältida neid olukordi.

7.5 Testid

Tagarakenduse testidesse testandmete lisamine käib hetkel suures osas läbi käsitsi sisestatud staatiliste SQL *insert* (SQL andmete sisestamise funktsioon) lausete. Igale tabelile on loodud eraldi SQL fail, kust andmed testide käivitamisel sisse loetakse. Kuna testide jaoks kasutatav H2 andmebaas on seadistatud käivituma suvamälus, siis andmed seal ei säili ning puudub ligipääs andmebaasi konsoolile, läbi mille saaks olemasolevaid andmeid lisada või muuta. Kui lisada andmebaasi tabelile uus veerg, mille väärtus on näiteks nõutud, tuleb selle tabeli andmete failis igale olemasolevale reale uue veeru väärtus käsitsi lisada. Seda saaks lahendada, kui testimise andmebaas muuta failipõhiseks. Siis tekiks ligipääs andmebaasi konsoolile ja oleks võimalik ühe SQL päringuga muuta mitut rida korraga.

Esirakendust testisime ainult käsitsi. Arendaja ise, ning koodi ülevaatusel tiimikaaslane, katsetas uue funktsionaalsuse läbi ning üritas läbi proovida võimalikult palju piirjuhte. Selline teguviis enamjaolt toimis hästi, kuid puudus kindlus, et tulevikus, kui olemasolevat koodijuppi muuta, eelnev funktsionaalsus säilib. Enamus probleemid, mis sellega seoses tekkisid olid lihtsasti lahendatavad. Kuna tulevikus koodimaht kasvab, on testimine kindlasti üks oluline viis tagada rakenduse toimimine ettenähtud viisil.

7.6 Andmete küsimine

Esirakendus küsib REST API päringute kaudu tagarakenduselt pidevalt andmeid. Tihti juhtub nii, et juba varem küsitud saadud andmed küsitakse kasutaja poolt uuesti, tehes ebavajalikke päringuid tagarakendusele, mis tähendab aga kasutajale aeglasemat lehte ja tagarakendusele lisakoormust.

Loodud esirakenduses liiguvad komponentide vahel ühes komponendis küsitud andmed järgmisesse komponenti, juhul kui viimane neid andmeid vajab. See on võimaldanud osasid andmeid taaskasutada. Komponentide vaheline andmete edasi kandmine ei kõrvalda aga siiski kõiki ebavajalikke API päringuid. Lehe uuesti laadimisel või eelmise komponendi juurde tagasi minnes küsitakse ikka andmeid uuesti.

Edasisel arendusel tuleks kaaluda andmete vahemällu salvestamine võimalusi, et ebavajalikke API päringuid minimeerida. Vahemälus andmete kasutamine kestaks mingi määratud aja või kuni mingi kindla kasutajapoolse tegevuseni.

7.7 Tiimikaaslaste leidmine

Informaatika erialal olid paljud 2022. aasta bakalaureusetöö teemad tiimiprojektid. Kasutajate tagasisidest tuli välja, et nad sooviksid lisaks teema leidmisele läbi meie rakenduse leida ka tiimi. Projekti- või teemavaates peaks olema sektsioon, kuhu tiimi otsijad saavad panna üles sellekohase teadaande ja luua grupi. Gruppi peaks olema võimalik teistel tiimiotsijatel kandideerida, või võiks olla võimalus teadaande esitajaga ühendust võtta. Lihtsam lahendus oleks selles vaates lasta tiimiotsijatel kirja panna lihtsalt väike lühikirjeldus ja kontaktandmed, et teised tiimiotsijad saaksid ühendust võtta meie rakenduse väliselt.

7.8 Teemadele kandideerimine

Praegune teemale kandideerimise lahendus võimaldab juhendajatel muuta kandideerimise avatust ja oodatud tiimi suurust. Kandideerida saab kas üksi või grupiga, kui oodatud tiimi suurus seda lubab. Sellele lisaks võiks saada piirata kandideerimise

ajaperioodi kuupäevade põhjal. Antud kitsenduse võiks saada lisada tervele projektile ühiselt, kui ka iga teema juurde eraldi. Juhendajatel võiks olla võimalus nõuda kandidaadilt enne kandideerimise alustamist küsimustele vastamist, ning näha vastuste ülevaadet enne, kui langetatakse otsus kandideerimise vastu võtmise või tagasilükkamise kohta. Hetkel saavad juhendajad lisada kandideerimise vastusele endapoolse sõnumi, ka kandideerijatel võiks olla võimalus lisada kandideerimisel endapoolne sõnum, mida juhendajad näevad.

8 Kokkuvõte

Antud bakalaureusetöö eesmärgiks oli luua veebirakendus projektide haldamiseks. Loodav rakendus pidi olema piisavalt paindlik, et kasutajad saaksid projekti seadistada enda vajaduste järgi. Eesmärgiks oli ka kasutada rakendust 2021/2022 õppeaasta informaatika bakalaureuse õppekava lõputööde esitamisel ja leidmisel.

Olulist rolli valminud veebirakenduses omas protsessimootor. Protsessimootori kasutuselevõtt tegi esialgu arendusprotsessi aeglasemaks, sest tegemist oli meeskonna jaoks uue tehnoloogiaga. Valminud rakenduses näeme siiski, et protsessimootori kasutamine aitab rakenduses keerukamaid protsesse paremini hallata ning uue funktsionaalsuse lisamine on paindlikum. Protsessiskeemid aitasid ka tiimil keskenduda konkreetsele eelnevalt paika pandud kasutusjuhule, mis vähendasid mestimisel konflikte koodis ja tegid koodi ülevaatusel protsessi paremini jälgitavaks ning mõistvamaks.

Seatud eesmärgid said täidetud. Oleksime muidugi tahtnud MVP varem valmis saada. Kindlasti leidub asju, mida me hiljem oleks teistmoodi teinud, aga üldjoontes oleme loodud süsteemiga rahul.

Näeme sellel rakendusel head kasvupotentsiaali ja loodame, et rakenduse arendus jätkub ka peale meie tiimi lõpetamist. Eriti tore oleks seda keskkonda näha mõne aasta pärast kasutuses ning laiema kasutuspinnaga kui lõputööd ja IT-teaduskond.

Kasutatud kirjandus

[1] Activiti, [Võrgumaterjal]. Available: https://www.activiti.org/. [Kasutatud 06 05 2022].

[2] J. Long, „InfoQ,“ [Võrgumaterjal]. Available: https://www.infoq.com/news/2010/12/activiti_5_released/. [Kasutatud 03 05 2022].

[3] Flowable, „Flowable open source,“ [Võrgumaterjal]. Available: https://www.flowable.com/open-source. [Kasutatud 26 04 2022].

[4] „Github flowable-engine,“ [Võrgumaterjal]. Available: https://github.com/flowable/flowable-engine. [Kasutatud 26 04 2022].

[5] P. Holmes-Higgin, „Flowable and Activiti: What the Fork?!,“ 12 10 2016. [Võrgumaterjal]. Available: https://www.flowable.com/blog/flowable-and-activiti-what-the-fork. [Kasutatud 26 04 2022].

[6] „Camunda,“ [Võrgumaterjal]. Available: https://camunda.com/. [Kasutatud 26 04 2022].

[7] „Github camunda-bpm-platform,“ [Võrgumaterjal]. Available: https://github.com/camunda/camunda-bpm-platform. [Kasutatud 26 04 2022].

[8] C. Humble, „ Camunda Forks Alfresco Activiti,“ 21 03 2013. [Võrgumaterjal]. Available: https://www.infoq.com/news/2013/03/Camunda-Forks-Activiti/. [Kasutatud 26 04 2022].

[9] D. Meyer, „Camunda Engine Evolution since Activiti Fork,“ 19 10 2016. [Võrgumaterjal]. Available: https://camunda.com/blog/2016/10/camunda-engine-since-activiti-fork/. [Kasutatud 26 04 2022].

[10] Camunda, „Camunda Cockpit,“ [Võrgumaterjal]. Available: https://camunda.com/platform-7/cockpit/. [Kasutatud 26 04 2022].

[11] Camunda, „Camunda Modeler,“ [Võrgumaterjal]. Available: https://camunda.com/download/modeler/. [Kasutatud 26 04 2022].

[12] E. You, „Vue.Js,“ [Võrgumaterjal]. Available: https://vuejs.org/. [Kasutatud 04 2022].

[13] Microsoft, "TypeScript," [Online]. Available: https://www.typescriptlang.org/. [Accessed 04 2022].

[14] A. Harley, „Visibility of System Status (Usability Heuristic #1),“ 03 06 2018. [Võrgumaterjal]. Available: https://www.nngroup.com/articles/visibility-system-status/. [Kasutatud 04 2022].

[15] OpenAPI-Generator Contributors, "OpenAPI Generator," 2022. [Online]. Available: https://openapi-generator.tech/. [Accessed 04 2022].

[16] OpenAPI Initiative, „OpenAPI Specification v3.1.0,“ 15 02 2021. [Võrgumaterjal]. Available: https://spec.openapis.org/oas/latest.html. [Kasutatud 04 2022].

[17] „Spring Boot,“ [Võrgumaterjal]. Available: https://spring.io/projects/spring-boot. [Kasutatud 30 04 2022].

[18] Camunda, „Github - UserTask Generated Form Preview and Embedded Form Generator,“ [Võrgumaterjal]. Available: https://github.com/camunda-consulting/camunda-7-code-examples/tree/master/snippets/camunda-modeler-plugins/camunda-modeler-plugin-usertask-generatedform-preview. [Kasutatud 26 04 2022].

[19] Gitlab, „Gitlab CI documentation,“ [Võrgumaterjal]. Available: https://docs.gitlab.com/ee/ci/ . [Kasutatud 03 05 2022].

[20] Gitlab, „Gitlab,“ [Võrgumaterjal]. Available: https://docs.gitlab.com/ee/user/packages/container_registry/.. [Kasutatud 05 03 2022].

[21] „User Management,“ [Võrgumaterjal]. Available: https://docs.camunda.org/manual/7.16/webapps/admin/user-management/. [Kasutatud 02 05 2022].

[22] Camunda, „Camunda documentation,“ [Võrgumaterjal]. Available: https://docs.camunda.org/manual/7.16/webapps/admin/group-management. [Kasutatud 03 05 2022].

[23] „Camunda enterprise,“ Camunda, [Võrgumaterjal]. Available: https://camunda.com/enterprise/. [Kasutatud 26 04 2022].

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

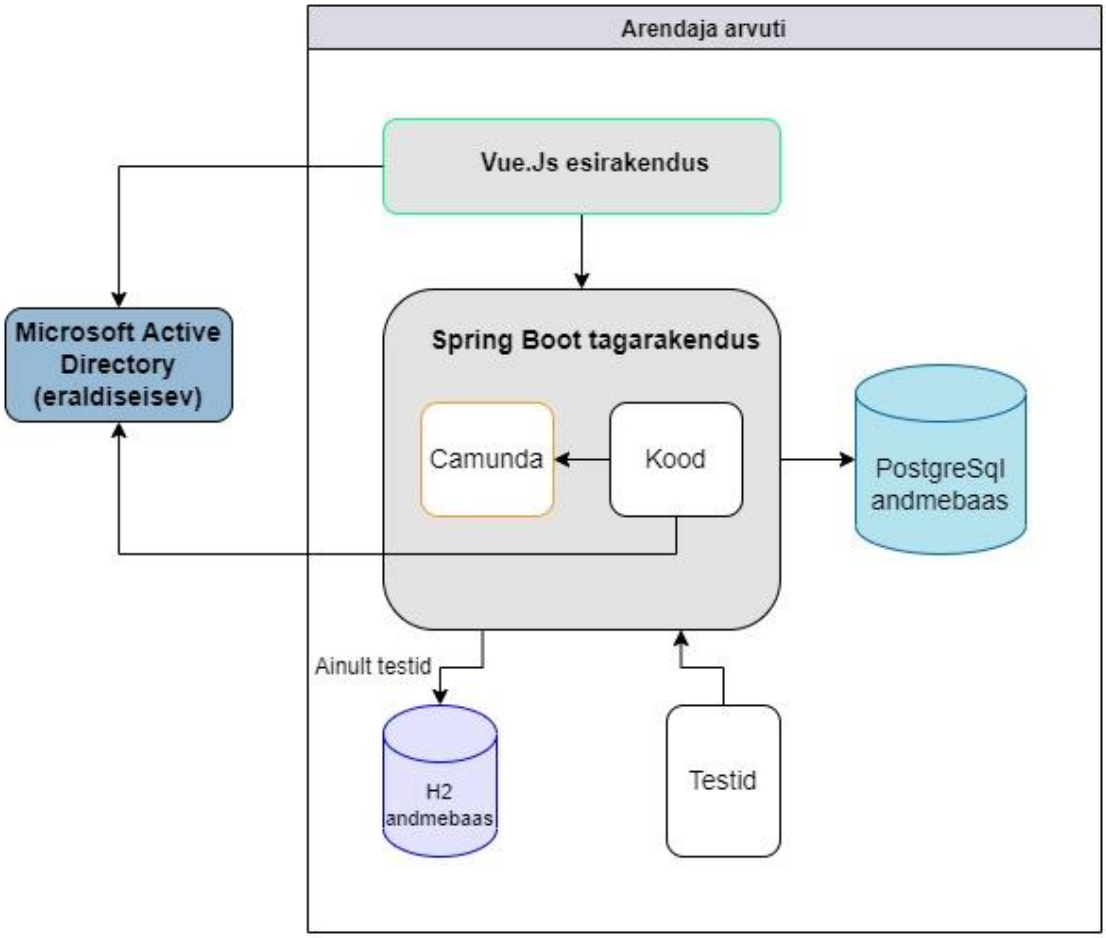
Meie, Sedrik Suurmets, Mikk Järvis ja Kaspar Ustav

1. Anname Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Projektide halduse infosüsteem Protsessor", mille juhendaja on Ago Luberg
   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. Oleme teadlikud, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitame, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

30.05.2022

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.

Lisa 2 – Lokaalse arenduskeskkonna arhitektuuri skeem



Joonis 2- 1. Lokaalse arenduskeskkonna arhitektuuri skeem.