TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Marek Lahk 185897IAIB
Oliver Laidma 185672IAIB
Kauri Riivik 186048IAIB

Väikeostude keskkond

Bakalaureusetöö

Juhendaja: Priit Järv
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: Marek Lahk, Oliver Laidma, Kauri Riivik

30.05.2022

Annotatsioon

Antud bakalaureusetöö eesmärgiks oli luua veebirakendus (edaspidi rakendus) TalTechi töötajate (edaspidi ülikooli töötaja või töötaja) jaoks ja kolmandate osapoolte jaoks (edaspidi partner) kes osalevad minikonkurssides ning väikeostudes, et lihtsustada praegust protsessi. Esirakendus loodi Reactiga [1], tagarakendus Laraveli [2], ja MySQL-ga.

Käesolev dokument annab ülevaate tehtud arendusest, tööriistadest, projekti haldusest ja tuleviku arenduste plaanidest.

Hetkel käib minikonkursside ja väikeostude protsess e-kiri teel, mingit eraldi süsteemi selle jaoks ei ole. Näiteks uue minikonkurssi puhul peab ülikooli töötaja käsitsi partneritele saatma kirjad, et uus minikonkurss on algatatud.

E-kirja saatmisega on kaks peamist puudust – esiteks nõuab see palju manuaalset tööd. Tekkinud andmed tuleb käsitsi teisaldada teistesse süsteemidesse edasi töötlemiseks. Teiseks, vead on kerged tulema. E-kirja rakendused ei kontrolli konkursi sisu loogilisust.

Arendatava rakenduse eesmärgiks on luua keskkond, mille kaudu minikonkursside ja väikeostude protsess oleks, nii ülikooli töötajatele, kui ka partneritele kiire, lihtne ja kasutajasõbralik.

Selle arenduse tulemusena oleme saanud valmis rakenduse, kus saavad nii ülikooli töötajad kui ka kolmandad osapooled mugavalt ja intuitiivselt minikonkursse ja väikeoste hallata/luua/muuta/kustutada.

Lõputöö on kirjutatud Eesti keeles ning sisaldab teksti 37 leheküljel, 11 peatükki, 11 joonist, 0 tabelit.

Abstract
Small procurement system

The objective of this bachelor's thesis was to create an appication for TalTech's employees and third party participants who will participate in procurements and small purchases. The frontend is made using React [1] , backend is made with Laravel [2] and MySQL.

This document will give an overview of the development process, tools used, project management and future developments for said application.

Currently, the process of handling procurements is done through emails. The main problem is that the emails have to be manually sent out. For example, when creating a new procurement an employee has to manually send out the emails.

There are two main issues with this approach. Firstly, it requires considerable amount of work by hand. Data generated by the process needs to be manually inserted into other university systems. Secondly, this approach is error-prone. Email client does not check for logic errors in the email content.

Our application simplifies and makes the whole process less prone to errors through a user-friendly user interface and a backend system.

With this development, we have completed an application, where both employees of the university and third party participants can easiliy and intuitively create, view, add, delete procurements and small purchases.

The thesis is in Estonian and contains 37 pages of text, 11 chapters, 11 figures, 0 tables.

Lühendite ja mõistete sõnastik

Tagarakendus	Rakenduse tagaloogika, kõik mis taustal toimub ja ei ole lõpp kasutajale nähtav
Esirakendus	Rakenduse kasutajaliidese kiht, see, mida lõppkasutaja näeb
Java	Tagarakenduse programmeerimise keel
SpringBoot	Tagarakenduse programmeerimise raamistik
PHP	*Personal Home Page*, tagarakenduse programmeerimise keel
Laravel	Tagarakenduse programmeerimise raamistik
React	Esirakenduse programmeerimise raamistik
TypeScript	Esirakenduse programmeerimise keel
MySQL	Relatsioonilise andmebaasi haldamise süsteem
Clockify	Rakendus, mis võimaldab tööle kulutatud aega kirja panna
Git	Versioonihalduse tarkvara
Gitlab	Projektihalduse süsteem
HTML	*Hyper Text Markup Language*, keel, millega brauser ehitab valmis veebilehed
API	*Application programmable interface,* võimaldab suhtluse vähemalt kahe rakenduse vahel
Endpoint	Aadress, läbi mille saab tagarakendusega suhelda
Swagger	API dokumenteerimise tööriist
Minikonkurss	Raamlepingu alusel asja ostmine, teenuse või ehitustöö tellimine
Väikeost	Asja ostmine, teenuse või ehitustöö tellimine
Töötaja	Tallinna Tehnikaülikooli töötaja
Partner	Minikonkurssi või väikeostu pakkuja
CI/CD	*Continuous integration / continuous delivery* – pidev integratsioon / pidevvalmidus

Sisukord

1 Sissejuhatus 9
2 Ülesandepüstitus 10
2.1 Funktsionaalsused 10
3 Projekti kirjeldus 12
3.1 Esialgse rakenduse kirjeldus 12
3.2 Lõputöö sisu 13
4 Projekti haldus 14
4.1 Koodi haldamine 14
4.2 Arenduse haldamine 14
4.3 Ajaarvestamine 15
5 Projekti disain 16
5.1 Kasutatud tehnoloogiad 17
5.1.1 React 18
5.1.2 TypeScript 18
5.1.3 PHP 19
5.1.4 Laravel 19
5.1.5 MySQL 19
5.1.6 Git 20
5.1.7 GitLab 20
5.1.8 Swagger 20
5.1.9 NGINX 20
5.1.10 Gitlab runner 21
5.1.11 Docker 21
6 Projekti sisu 22
6.1 Tagarakenduse struktuur 22
6.2 Esirakenduse struktuur 23
6.3 Swagger 25
6.4 Andmebaas 26

6.5 Kasutaja tuvastamine 26
6.5.1 Töötajate tuvastamine 26
6.5.2 Partnerite tuvastamine 27
6.6 Kasutaja õigused 27
6.7 Minikonkursside ja väikeostude staatuste haldus 27
6.8 Töötaja voog 29
6.9 Partneri voog 31
6.10 Dokumentide süsteem 32
7 Testimine 33
7.1 Tagarakendus 33
7.2 Esirakendus 33
8 Valideerimine 34
9 Tulemused 35
10 Kommentaarid 37
10.1 Esirakendus 37
10.2 Tagarakendus 37
10.3 Andmebaas 37
10.4 Koodi haldus 38
10.5 Projekti haldus 38
10.6 Swagger 38
11 Kokkuvõte 39
11.1 Edasised arendused 39
Kasutatud kirjandus 41
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 42

Jooniste loetelu

Joonis 1. Rakenduse voog 17
Joonis 2. Tagarakenduse voog TalTechi serveris 22
Joonis 3. Tagarakenduse sisemine voog 23
Joonis 4. Esirakenduse jaotus 24
Joonis 5. Esirakenduse üldisem skeem 25
Joonis 6. Swaggeri minikonkurssi vaade 25
Joonis 7. Minikonkursside ja väikeostude staatuste diagramm 28
Joonis 8. Töötaja voog 29
Joonis 9. Partneri voog 31

1 Sissejuhatus

Tallinna Tehnikaülikool kasutab oma minikonkursside ja väikeostude haldamiseks e-kirju. Kogu tegevus, alates minikonkurssi või väikeostu loomisest kuni selle lõppemiseni, käib tervenisti e-kirja vahendusel. Selline lahendus on raskesti hallatav ja on palju võimalusi vigu teha.

Probleemile lahenduseks on luua rakendus, mis võimaldaks läbi mugava kasutajaliidese minikonkursside ja väikeostude protsessi töötajatel hallata. Lisaks töötajate rakendusele, on vaja ka rakendust kolmandatele osapooltele, ehk partneritele, kes saavad läbi süsteemi esitada pakkumusi ning esitada küsimusi.

Eelnevalt kirjeldatud süsteemi arendus sai alguse Tarkvaratehnika aine raames [3]. Algne tagarakendus loodi kasutades Java SpringBooti [4]. Seda ei jõutud täielikult valmis ning ei olnud reaalsesse kasutusse võtmiseks sobiv.

Antud lõputöö raames tehti muudatusi esialgses projektis, et see oleks võimalikul valmis reaalseks kasutuseks ülikooli infosüsteemis. Selleks kirjutati kogu tagarakendus ümber, kasutades PHP-d ja raamistiku Laraveli ning liidestati Tarkvaratehnika aines valminud esirakendusega. See esirakendus järgis TalTech Styleguide-t [5]. Vastavalt kliendi soovidele täpsustati ja muudeti mõlema projekti osa nõudeid. Vastavad muudatused viidi ka rakendustes ellu.

2 Ülesandepüstitus

Ülikool kasutab oma minikonkursside ja väikeostude haldamiseks e-kirju, mis tekitab palju manuaalset tööd. Näiteks peab töötaja hanke jooksul tekkinud andmed käsitsi TalTech sisestama. Lisaks on praeguses süsteemis palju veaohtlike kohti – ühele partnerile jääb e-kiri saatmata või e-kirja sisus olevad kuupäevad on minevikus.

Ülikool soovis käsitsi e-kirjade saatmise asemel kasutada mugavat rakendust, mis võimaldaks läbi teha kogu äriloogika mis on minikonkursside ja väikeostude protsessidega seotud ning võimaldaks ka kolmandatel osapooltel nendest protsessidest osa võtta lihtsamalt ja mugavamalt.

Lisaks oli vaja lisada õiguste süsteem, dokumentide süsteem, endpoindid, mis kataks kogu äriloogika ning osa tagarakendusest testidega ära katta. Ülesandeks oli valmis teha esi- ja tagarakendus, mis kommunikeerivad omavahel. Mõlemad peavad asuma ülikooli poolt antud serveris. Enne rakenduse lõpliku üleandmist peaks ka kasutajatestimine ja valideerimine tehtud olema, veendumaks, et lõplik rakendus vastab tellija poolt püstitatud nõuetele.

2.1 Funktsionaalsused

- Rakendus asub aadressil *https://minihanked-thesis.taltech.ee/*
- Ülikooli töötaja saab oma kasutajaga sisse logida
- Süsteem kontrollib kasutaja õiguseid
- Töötaja saab hallata partnerit (luua, muuta, kustutada)
- Töötaja saab raamlepingut hallata (luua, muuta, kustutada)
- Töötaja saab hallata minikonkurssi/väikeostu (luua, muuta, kustutada, partnereid lisada)

- Partner saab hallata oma pakkumisi minikonkurssidele/väikeostudele, kuhu ta on osalema valitud (luua, muuta, kustutada)
- Partnerile ja töötajale saadetakse automaatselt e-kirjad ja süsteemisisesed teavitused
- Töötaja saab enda dokumente hallata (üles ja alla laadida)
- Partner saab enda dokumente hallata (üles ja alla laadida)
- Partner saab küsimusi minikonkurssi/väikeostu kohta küsida
- Töötaja saab küsimustele minikonkurssi/väikeostu kohta vastata
- Töötaja saab saata teavitusi
- Töötaja saab vaadata statistika vaadet

3 Projekti kirjeldus

Väikeostude süsteemi projektil on olnud mitu iteratsiooni. Kliendi sõnul on enne Tarkvaratehnika aine raames loodu projekti üritatud mitme teise projekti raames sama ülesannet lahendada. Nende käigus tehtud töö ei olnud sobiv või ei jõutud piisavalt kaugele ning projektid jäid seisma.

Nendele ebaõnnestunud projektidele järgneski Tarkvaratehnika projekt. Klient jäi tulemusega piisavalt rahule, et seda edasi arendada. Selle töö edasiarendus ongi praegune lõputöö.

3.1 Esialgse rakenduse kirjeldus

Tarkvaratehnika aine raames valmis esi- ja tagarakendus. Tagarakendus oli kirjutatud Java SpringBootiga, äriloogika jäi paika panemata või realiseerimata. Näiteks jäi läbi arutamata kasutaja õiguste süsteem ja realiseerimata audit vaade (uue nimega statistika vaade). Lisaks oli vaid väike osa rakendusest kaetud automaattestidega. Sisse oli jäänud ka palju vigu. Eelkõige oli IT osakonna soov et konkreetne rakendus oleks kirjutatud PHP-s mitte Javas, kuna enamus ülikooli sisestest süsteemidest on just PHP peal kirjutatud.

Tarkvaratehnika aine raames sai alustatud ka esirakendusega. Ülikool jagas pärast Framer prototüübi valmimist Taltech Styleguide, mis on ülikooli rakenduste stiili põhi. Paika said peamised põhiobjektid, mis olid töötaja voogude jaoks olulised. Seejärel sai teostatud lihtsamad vormi komponendid ja tabel vaated, et põhiobjekte rakendusse tekitada. Viimaks valmis algne hanke teostamise voog ja sinna juurde partneri poolne pakkumuse teostamise voog. Esirakendus kasutajaliides jäi viimistlemata, vaated ning vormid polnud väga kasutajasõbralikud. Suur osa andmetest oli kasutajaliideses staatilised. Lisaks jäid paljud kliendi poolt nõutud täiendused sisse viimata aja puudumise tõttu.

3.2 Lõputöö sisu

Arenduse käigus pidi valmima eraldi taga -ja esirakendus. Mõlemad pidid ülikooli serveris püsti olema ning avalikult nähtavad. Enamus äriloogilistest nõuetest sai kirja tarkvaratehnika aine raames. Paljud nõuded aga täpsustusid, äriloogika loomisele aitas klient kaasa nõuete paika panemisele. Paljudele mudelitele tulid juurde uued väljad, täpsustusid piirangud. Auditi vaate sisu muutus ja sellega seos sai ka uue nime – statistika vaade. Statistika vaatele tehti mitu erinevat kasutajaliidese prototüüpi, et klient saaks hinnata nende sobivust. Väga suur muudatus oli ka hanke oleku süsteemi muutus - täpsustusid olekude definitsioonid ning tulid juurde uued lõppolekud. Paika sai pandud täpsed kasutajaõigused.

Lõpprakenduses pidi olema kaks eraldi vaadet – ülikooli töötaja vaade ning partneri vaade. Nii töötaja kui ka partneri vaade eeldab täpset äriloogika jälgimist. Töötaja vaade eeldab Azure AD kasutaja olemasolu ülikooli süsteemis. Sisse logimine käib läbi selle kasutaja, liidestus Azure AD kasutajatega peab olemas olema. Lisaks peab olema automaatne e-kirjade saatmine teatud tingimustel (näiteks, kui partneri pakkumine võeti vastu, siis partnerile saadetakse automaatselt e-kiri). Veel peab rakendusel olema failide üles- ja allalaadimise funktsionaalsus, nii töötajatele kui partneritele. Et rakendus õigesti funktsioneeriks, pidi olema ka õiguste süsteem.

4 Projekti haldus

Enne kui koodi kirjutama hakati, pandi täpselt paika, milline projekti haldus välja näeks. Ilma adekvaatse projekti halduse strateegiata on keeruline ka edukalt projekt lõpuni arendada. Sai paika pandud nii omavahel sprintide planeerimise koosolekutel, tellija ja juhendajaga toimuvate koosolekute ajad.

4.1 Koodi haldamine

Koodi hoidmine TalTechi siseses gitlabis (gl.ttu.ee). Gitlab võimaldab projekte (esi- ja tagarakendust) hallata – tekitasime iga äriloogika osa (Edaspidi osa, näited osadest: *employee, partner, procurement participant*) jaoks eraldi *milestone*, mis andis hea ülevaate kui kaugel mingi osa on. Iga *milestone* alla sai pandud eraldi *issued* ehk ülesanded, mis olid kindla osa väiksemad arendused. Konkreetsed ülesanded lisati pärast äriloogika arutamist kliendiga. Kui midagi valmis sai, tekkis kohe hea ülevaade kui kaugel arendatav rakendus on.

Projekti koodihalduses sai kasutatud versiooni halduse tööriista Git. See võimaldab arenduse jaoks erinevaid harusid luua, ning kui arendus valmis sai siis muudatused pea harusse viia. Vajadusel saab ka muudatusi tagasi võtta.

Sai kasutatud ka põhimõtet et kui mingi väike arendus sai tehtud siis tehti *merge request*, mille vaatas teine tiimiliige üle, mõttega, et kui midagi jäi kahe silma vahele või midagi saaks paremini teha siis see muudatus ellu viia, mille tulemusena rakenduse töökindlus suurenes. See metoodika erines esirakenduses kuna esirakendust kirjutati üksi.

4.2 Arenduse haldamine

Töö käis sprintides, iga nädala kolmapäeval algas uus sprint. Uue sprindi algul vaatasime tehtud tööd üle, kommenteerisime neid, vajadusel parandasime. Lisaks planeerisime ka uue nädala sprindi, mida keegi teeb selle jooksul. Dokumenteerisime ka sprinte. Lisaks omavahelistele sprintidele, toimus iga nädal koosolek tellija ja juhendajaga, kus panime

paika uued eesmärgid, täpsustasime olemasolevaid ning rääkisime ja näitasime mis tehtud, mis tegemisel ja arutasime küsimusi, mis arenduse käigus tekkisid.

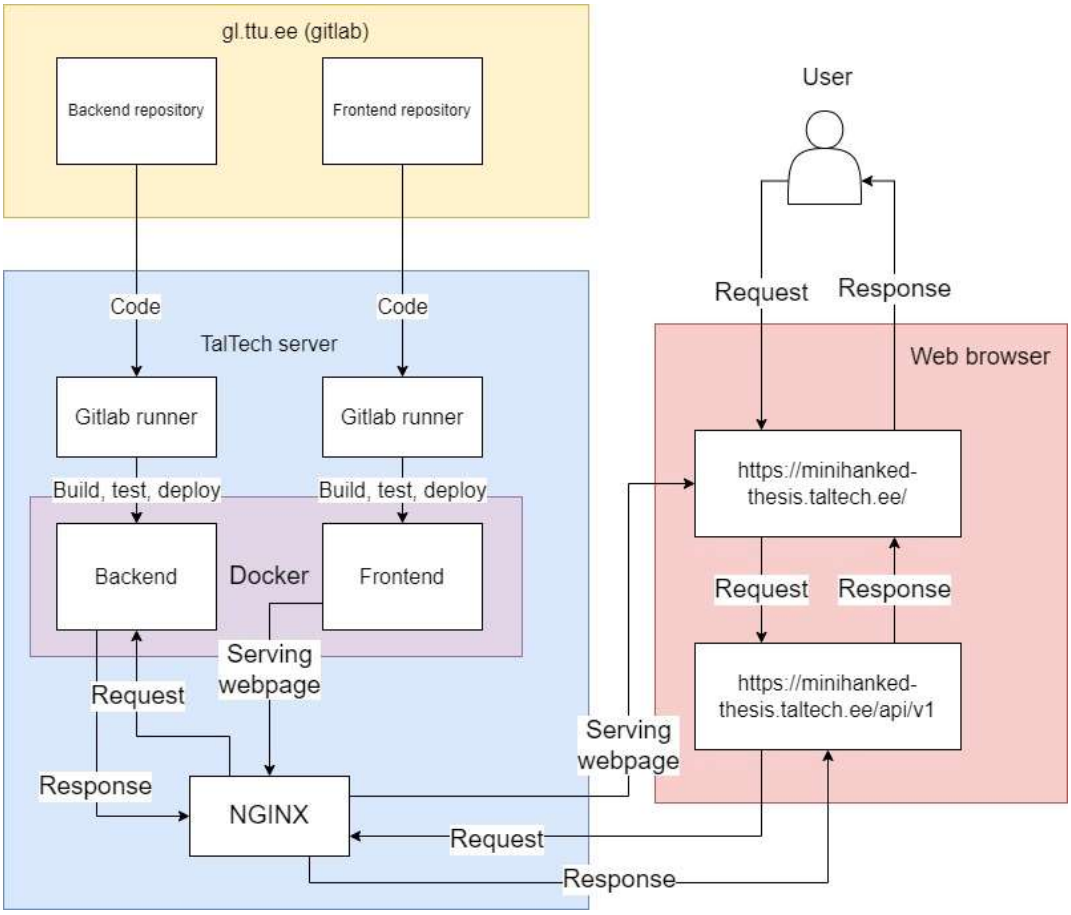
4.3 Ajaarvestamine

Aega sai mõõdetud kasutades Clockify ajahaldus programmi. Aega võtsime mitte ainult koodi kirjutamisel vaid ka äriloogika läbimõtlemisel, koosolekutel, omavahelistel aruteludel. Aja võtmisega lähtusime sellest, kas tegu on koodi kirjutamisega või mitte – kui jah, siis vastavalt kas tagarakenduse alla arvestasime aega või esirakenduse alla. Kui tegu ei olnud koodi kirjutamisega, siis panime aja koosolekute alla.

5 Projekti disain

Rakendus koosneb esi- ja tagarakendusest. Mõlemad rakenduse lähtekood asub ülikooli sisemises gitlabis ehk gl.ttu.ee-s. Ülikooli serveris töötavad nii taga- kui esirakenduse CI/CD süsteem, NGINX ja docker. Docker konteinerites töötavad taga- ja esirakendus.

Esirakendus on Reacti peal kirjutatud, kasutades TypeScripti. Tagarakendus ehk kõik mis taustal toimub on kirjutatud PHP raamistiku Laraveli peal. Laravel toetab ka *server side renderingut* [6] kuid meie rakendus kasutab seda ainult API-na. Nii taga- kui ka esirakenduse päringud käivad läbi NGINX-i (joonis 1). Esirakendus serveerib veebilehti (joonis 1), pärast kasutaja toimingut pärib andmeid vastu tagarakenduse API-d (joonis 1) ja tagarakendus saadab vastuse (joonis 1). Andmebaasina on kasutusel MySQL Andmebaas asub tagarakenduse sees. API dokumenteerimiseks kasutasime Swaggerit. Ülikooli server jooksutab taga- ja esirakenduse jaoks eraldi Gitlab runnerit (joonis 1), mis ehitab, testib ja lõpuks paneb koodi üles (*build*, *test*, *deploy*).



Joonis 1. Rakenduse voog

5.1 Kasutatud tehnoloogiad

Meie süsteem kasutab erinevaid komponente. Allpool toome välja kõige tähtsamad tehnoloogiad. Lisaks mainime muid lahendusi ning miks me otsustasime just valitud tehnoloogia kasuks.

- React – Valitud tarkvaratehnikas
- TypeScript – Valitud tarkvaratehnikas
- PHP – Ülikooli poolne nõue
- Laravel – Ülikooli poolne nõue
- MySQL – Ülikooli poolne nõue
- Git – Omal valikul

- GitLab – Omal valikul
- Swagger – Omal valikul
- NGINX – Omal valikul
- Gitlab runner – Omal valikul
- Docker – Omal valikul

5.1.1 React

Reacti kasutasime esirakenduse arendamise jaoks. React on esirakenduse arendusraamistik, mis toetab TypeScripti ning JavaScripti. React on väga populaarne esirakenduse raamistik. Algselt on see mõeldud suurtele ettevõtetele, kus toimuvad suuremahulised projektid ja teeb lihtsaks veebilehe kerge muutmise, uute vaadete lisamise, toetades suuresti komponentide põhist loogikat ehk iga väikene osa veebilehel on kindel komponent. Neid saab taaskasutada ja muuta. Komponentidega on lihtsam ümber käia, lihtsam uusi vaateid luua kui need tükid juba olemas on. Kõik see tähendab, et esirakenduse arendus on kiire ja mugav. Sai valitud just nende põhjuste pärast Reacti. Alternatiivideks oleks Vue ja Angular. Populaarsuse koha pealt Vue ja React on suhteliselt sama populaarsed, samuti kogukonna toetuse poolest [7]. Angulari õppimiskõver on võrreldes Reactiga natuke suurem, üldiselt on Angular parem suuremate rakenduste kirjutamiseks [7].

5.1.2 TypeScript

TypeScripti kasutasime Reacti raamistiku programmeerimiskeelena. TypeScript on üks populaarsemaid esirakenduse programmeerimise keeli, mis on loodud Microsofti poolt ning on avatud lähtekoodiga. TypeScript üritab lahendada JavaScripti probleemi, kus puuduvad sisemiste andmestruktuuride deklareerimise võimalused ehk lisab tüübid. Sisuliselt ilma tüüpideta programmeerimine on lihtsam aga mahukamate projektide juures suurendab vigade ohtu. Nende eeliste tõttu valisime TypeScripti mitte JavaScripti.

5.1.3 PHP

PHP-d kasutasime Laraveli raamistiku programmeerimiskeelena. PHP on programmeerimise keel, mis on avatud lähtekoodiga. See toetab *server-side renderingut*, mis lühidalt tähendab et server konverteerib HTML failid otse kuvatavateks lehekülgedeks. PHP on selle poolest natuke eriline tagarakenduse programmeerimise keel, kuna tegelikult temaga saaks ehitada valmis kogu rakenduse ehk tal on tugi ka esirakenduse poole pealt. Meie oma rakenduses seda funktsionaalsust ei kasutanud. Valisime PHP kuna kliendi sooviks oli just et rakendus PHP-d kasutaks kuna ülikooli enda süsteemid on kõik PHP-s kirjutatud. Alternatiivsed valikud oleks olnud Java, Golang.

5.1.4 Laravel

Laraveli kasutasime tagarakenduse raamistikuna. Laravel on avatud lähtekoodiga PHP raamistik. Laraveli põhimõte on sama nagu PHP-l aga tal on sisse ehitatud rohkem funktsionaalsust näiteks autentimine, e-kirjade saatimise süsteem jne. Lisaks on tal nagu PHP-l võimekus otse esirakendust tagarakendusse kirjutada. Aga nagu PHP puhul juba mainitud, siis meie rakendus kasutab PHP-d ja sh ka Laraveli ainult tagarakendusena. Laravel teeb päringute valideerimise väga lihtsaks, sellesse sisse ehitatud e-kirjade süsteem on lihtne ja intuitiivne ning see liidestub hästi andmebaasidega. Alternatiivideks oleks olnud Symfony, Phalcon. Mõlemad on Laravelist vähem populaarsed. Symfoni koodi struktuur on veidi pikkem, Laravel kasutab selle vältimiseks *magic-methode* ja *traite*. Symfoni on mõeldud suuremate projektide jaoks, mida kasutab väga suur hulk kliente [8]. Võrreldes Phalconiga on Laravel aeglasem aga selle eest Laraveli kirjutada on lihtsam ning kogukond on palju suurem [9].

5.1.5 MySQL

MySQL-i kasutasime tagarakenduse andmebaasina. MySQL on üks laialdaselt kasutatav relatsiooniline andmebaasisüsteem, mis on avatud lähtekoodiga. Valitud sai see just sellepärast et see on väga populaarne ja kiire [10] ning ülikooli sisesed süsteemid kasutavad just MySQL-i. Alternatiivideks oleks olnud PostgreSQL, SQLite, mõlemad head valikud.

5.1.6 Git

Git on väga populaarne avatud lähtekoodiga versioonihaldussüsteem, mida antud arenduses kasutasime. Git teeb võimalikuks kasvõi väga suure projekti halduse, mis võimaldab sh ka uusi harusid luua, tagasi muudatusi võtta, kahte haru kokku viia jpm. Alternatiivideks olid CVS ja SVN aga GitLab toetab ainult Git-i.

5.1.7 GitLab

GitLab on avatud lähtekoodiga projektihaldusesüsteem, mis lihtsustab suurte ja ka väikeste projektide haldamise. Sellele on sisse ehitatud palju funktsionaalsusi, näiteks Giti tugi, *issued, milestoned.* GitLab sai valitud kuna ülikool kasutab enda süsteemide lähtekoodide hoiustamiseks GitLab-is. Headeks alternatiivideks oleks olnud GitHub ja BitBucket.

5.1.8 Swagger

Swagger on avatud lähtekoodiga töörist API dokumenteerimiseks katsetamiseks. Antud rakendus kasutab OAS3 spetsifikatsiooni. Swaggeri abil saab luua kasutaja liidese API päringute dokumenteerimiseks ja testimiseks. Selle kaudu on võimalik kirjeldada parameetreid ning and muud infot tagarakenduse kohta.

Alternatiivideks on Postman ning Apigee. Postmaniga võrreldes on Swaggeri püstipanek keerulisem, kuid see eest API testimine Swaggeriga on palju kiirem ja lihtsam. Lisaks toetab Swagger paremini ka õigusi, Postmanil seda võimekust pole [11]

Apigeel on sisse ehitatud rohkem võimalusi, näiteks statistika näitamine, krüpteerimine jne, Swagger on selle koha pealt lihtsam. Lisaks on Apigee loomisel mõeldud rohkem pilvelahenduste peale [12]. Meie Apigee lisa funktsionaalsusi ei vajanud, sellepärast valisime Swaggeri.

5.1.9 NGINX

NGINX on vabavaraline tarkvara, mille abil saab veebilehti serveerida, veebi liiklust suunata, reverse-proxy-t [13] tekitada jpm. Meie kasutame seda reverse-proxy loomiseks ning veebilehtede serveerimise jaoks. Lisaks tegeleb ka NGINX SSL sertifikaatidega ning võimaldab meil krüpteerita kasutajaliidese päringud ning vastused. Alternatiivideks

oli Apache. Apache puhul on tegemist protsessipõhise arhitektuuriga ehk iga uus päring tähendab uut lõime [14]. NGINX suudab tänu oma sündmustepõhisele arhitektuurile ühe lõimega mitut päringut korraga töödelda [14]. Meile sobis sündmustepõhine arhitektuur rohkem.

5.1.10 Gitlab runner

Gitlab runner on CI/CD tarkvara. See süsteem on liidestatud koodivaramuga ning koodimuudatuse korral kontrollitakse muudatusi automaattestidega. Sellega saame veenduda, et varamus olev kood on töökorras, enne, kui koodi jõuab lõppkasutajani. Kui kõik vajalikud koodi muudatused on sisse viidud ja testitud, tõstame muudatused varamu põhiharusse, millega käivitatakse koodimuudatuste sisse viimine lõppkasutaja teenusesse.

Alternatiiviks on Jenkins. Peamine põhjus, miks me valisime Gitlab runneri oli selle parem liidestatus Gitlabiga. See tuleneb asjaolust, et Gitlab runnerit arendab sama firma, kes arendab Gitlabi.

5.1.11 Docker

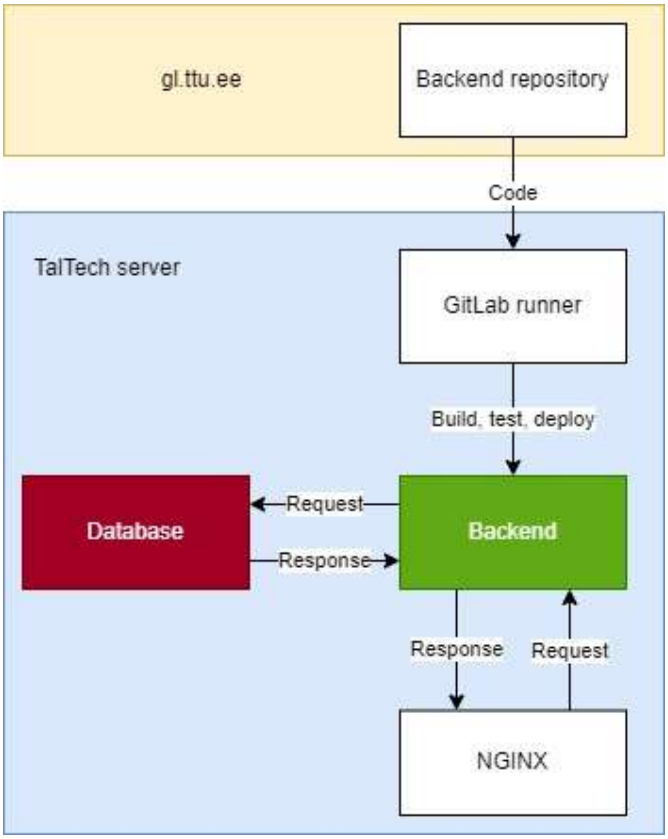
Docker on konteineriseerimis (virtualiseerimis) tarkvara. Meie rakendus kasutab nelja konteinerit - ühte tagarakenduse, ühte andmebaasi ja kahte esirakenduste jaoks. See lahendus lihtsustab meie süsteemi ülesseadmist ja haldamist. Konteineri sees töötav operatsiooni süsteem ja abiteenused ei sõltu serveri baasoperatsioonisüsteemist. See teeb tarkvaraliste sõltuvuste haldamise palju lihtsamaks. Lisaks sellel saab süsteemi, konteineri malli olemasolul, käivitada ühe käsuga.

Alternatiiviks on Kubernetes. Sarnaselt Dockeriga on tegemist konteineriseerimis tarkavaraga. Kubernetese eeliseks on selle võimekus hallata suurt hulla konteinereid. Otsustasime kasutada Dockerit, kuna meie süsteem ei nõua skaleeritavust ja suurt kogust konteinereid. Valisime tööriista, mis sobitus kõige paremini meie nõuetega.

6 Projekti sisu

Järgnevalt kirjeldame kõige olulisemad komponendid nii taga- kui esirakendusest. Räägime taga- ja esirakenduse struktuurist, Swaggerist, andmebaasist, süsteemi kasutajate tuvastamisest (nii töötaja kui partner), õiguste süsteemist, minikonkursside ja väikeostude staatuse haldusest, töötaja ja partnerite voogudest ning dokumentide süsteemist.

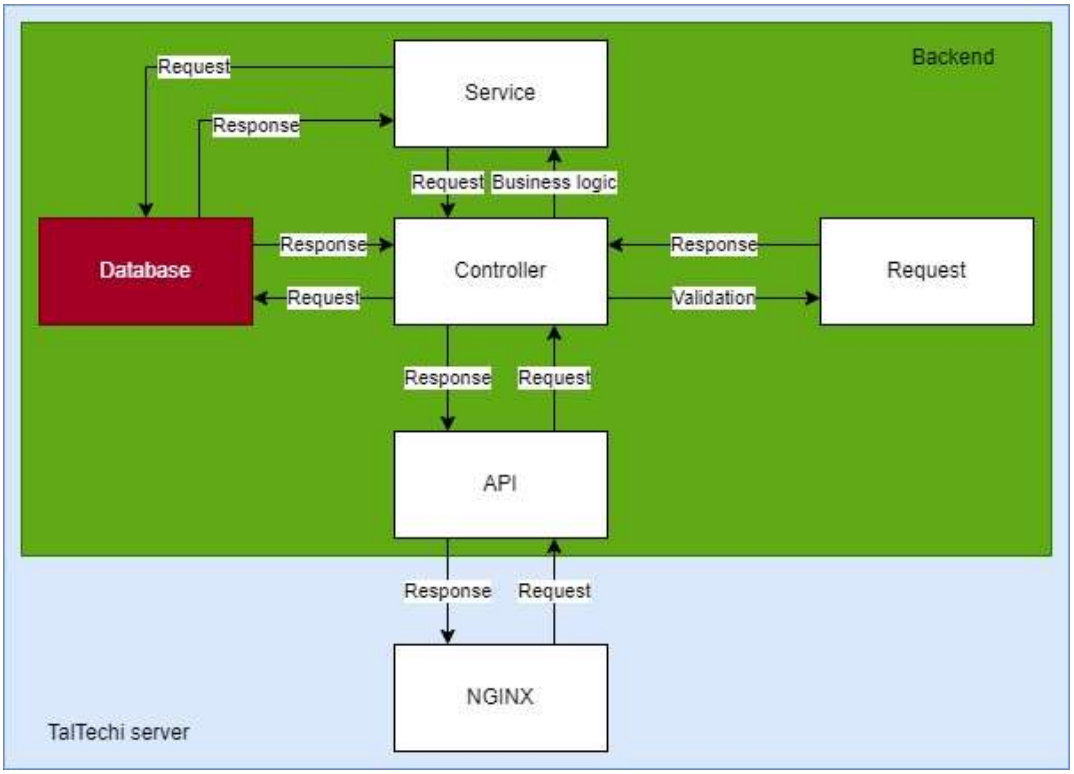
6.1 Tagarakenduse struktuur



Joonis 2. Tagarakenduse voog TalTechi serveris

Tagarakenduse kood tuleb ülikooli sisesest GitLabist ehk gl.ttu.ee alt. GitLab runner ehitab rakenduse valmis, paneb ühik- ja intergratsioonitestid käima. Kui testid õnnestusid, siis uuendatakse tagarakenduses asuvat koodi (deploy) vastavalt muudatustele.

Päringud tulevad esirakendusest läbi NGINX-i. Vajadusel küsib tagarakendus andmebaasi käest andmeid ehk teeb päringu, andmebaas omakorda vastab. Lõpuks saadab tagarakendus vastuse NGINX-le, mis omakorda suunab selle esirakendusele.



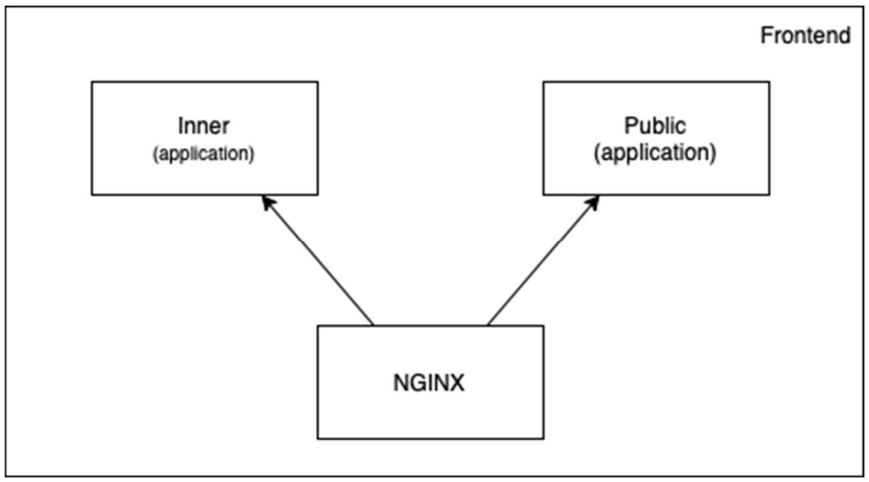
Joonis 3. Tagarakenduse sisemine voog

Sisemiselt saadab NGINX päringu tagarakenduse API pihta, mis suunab selle edasi kontrollerisse. Kontroller saadab üldiselt esimese asjana selle päringu valideerijasse, mis tagastab vastuse. Kui vastus tuli et vigane päring, siis saadab kontroller kohe API-le vastuse, mis omakorda saadab selle NGINX-le edasi. Kui vastus tuli positiivne, siis kontroller teeb päringuga teatud toiminguid, suurema äriloogika korral annab *service*-le ehk teenusele mingi osa äriloogikast ise ära teha. Nii kontroller kui ka teenus võivad vajadusel andmebaasi käest lisa andmeid pärida, andmebaas vastab nendele. Lõpuks kui kogu äriloogika on ära tehtud, saadab kontroller API-le vastuse, mis edastab selle NGINX-le.

6.2 Esirakenduse struktuur

Esirakendus on jagatud kaheks eraldiseisvaks rakenduseks, kujutatud joonisel 4. *Inner* ehk töötajatele mõeldud konkursside haldamise rakendus ja *public* ehk partneritele

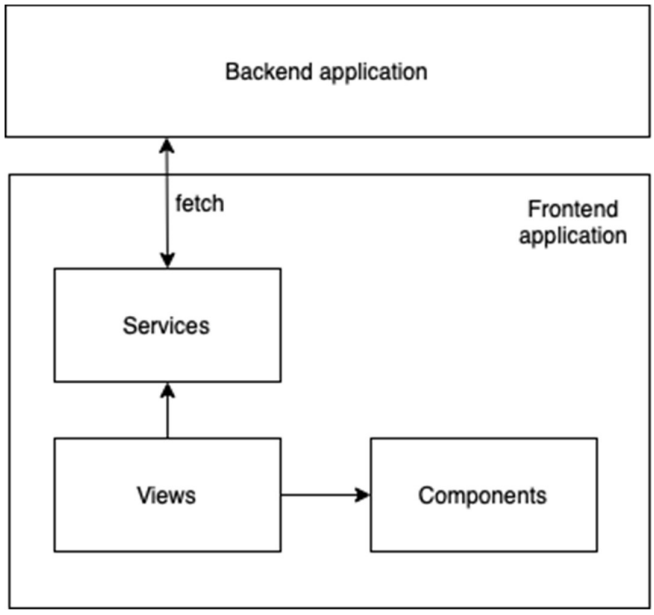
mõeldud pakkumuste tegemiste rakendus. Erinevaid rakendusi jagab NGINX vastavalt aadressile.



Joonis 4. Esirakenduse jaotus

Rakenduste eraldamine toob kaasa mitmeid eeliseid. Esiteks on lehe laadimine kiirem. *Public* rakendus on lihtsa ehitusega ja vähe komponente, samas *Inner* rakendus hõlmab endas kogu haldamise poolt, kus on sisselogimine, statistikad, jpm. keerulisi komponente. Teiseks on turvalisus. *Inner* rakendusega ei tule kaasa kõiki API aadresse ja äriloogikat. Viimaks on rakenduse komponendid ja funktsionaalsused loogilisemalt jagatud ja võõrale inimesele kergemini mõistetav.

Lisaks on rakendust jagatud eraldi kihtideks. Lihtsaim kihiline esitusviis on vaated, teenused ja komponendid, mida illustreerib joonis 5. Vaated kombineerivad päringud ja komponendid. Rakenduse juurde kuulub palju teisi tähtsaid elemente, milleks on klasside kirjeldused, stiilid ja muud kasulikud funktsioonid. Kuid kõigi kuvamine muudaks skeemi keeruliseks ja raskesti mõistetavaks.



Joonis 5. Esirakenduse üldisem skeem

6.3 Swagger

Joonis 6. Swaggeri minikonkurssi vaade

Et tagarakendusele päringuid teha, on vaja mingit aadressi. Aadressi alguse osa ehk antud rakenduse puhul https://minihanked-thesis.taltech.ee/ ei ole tegelikult tagarakenduse pool vaid esirakenduse pool. Kui aga sinna lisada api/v1/ siis suunatakse päringud otse tagarakendusse. Lihtsamaks päringute tegemiseks on lisatud tagarakendusse Swaggeriga dokumentatsiooni, mis võimaldab dokumenteerida kõiki päringuid ja lihtsustab arendust. Swaggeri aadress on https://minihanked-thesis.taltech.ee/api/v1/docs ning https://minihanked-thesis.taltech.ee/api/v1/public/docs. Swaggeri eeliseks on see, et kogu kasutajaliides genereeritakse PHP kommentaaride põhjal – see eemaldab vajaduse ise kasutajaliidest kirjutada.

6.4 Andmebaas

Andmebaasina kasutatakse MySQL-i, mis integreerub Laraveliga väga lihtsalt ja hästi. Andmebaasi loomise jaoks kasutame migreerimise skripte, mis rakenduse valmis ehitamisel käima pannakse ja mis sisestavad õigel kujul andmebaasi kõik tabelid, vaated, trigerid jne. Testimise jaoks lisasime andmebaasi *seederid*, mis lisavad andmebaasi testandmed.

6.5 Kasutaja tuvastamine

Süsteem peab teadma, milline kasutaja päringu serverisse saatis. Selleks peab kasutaja tuvastama. Rakenduses on kaks erinevad viisi kasutaja tuvastamiseks - üks töötajate, teine partnerite jaoks.

6.5.1 Töötajate tuvastamine

Kuna ülikool kasutab kasutaja andmete hoiustamiseks ja tuvastamiseks Azure AD süsteemi, siis pidi ka meie süsteem liidestuma sellega. Azure AD kasutab Oauth2 standardit. See standard lubab rakendusel kasutada sisse logimiseks teise teenuse pakkuja kasutajatuvastus süsteemi. Selle kaudu saab kätte töötaja andmed, mille salvestame süsteemisiseseks kasutamiseks andmebaasi.

Uue päringu saabudes kontrollib süsteem päringus olevat sessiooni küpsist. Otsitav küpsis on nimega *small_procurements_session.* Selle leidmisel kontrollitakse selle vastavust tagarakenduses salvestatud väärtusega. Juhul, kui kõik kontrollid õnnestuvad, lubatakse kasutaja ligi ressursile, mida ta päris.

Küpsise puudumise või vale väärtuse korral tagastab tagarakendus veateate ning kasutajat ei lubata edasi. Selle peale suunab esirakendus kasutaja ümber sisselogimis vaatele.

Küpsise aegumise korral suudab süsteem siiski kasutaja tuvastada. Sisse logides salvestati andmebaasi kasutaja andmed ning tuvastus- ja värskendustoken. Rakendus, kasutades samu tokeneid, üritab neid läbi Azure AD värskendada. Kui värskendamine õnnestub lubatakse kasutaja süsteemi. Ebaõnnestumisel peab kasutaja uuesti Azure AD-sse sisse logima.

6.5.2 Partnerite tuvastamine

Rakenduse üks eesmärkidest on kasutaja mugavus. Nõudes partnerilt konto loomist teeks protsessi tülikamaks ja aega nõudvamaks. Selle vältimiseks määratakse igale konkursis osalevale partnerile unikaalse ligipääsu võti, millega ta saab ligi pakkumuste leheküljele.

Minikonkurssi alguses saadetakse igale osalejale email. Selles sisaldub pakkumuste lehe link tema unikaalse võtmega. Pakkumuste lehel tehtavate päringutega annab kasutajaliides tagarakendusele päises kaasa võtme, mille alusel suudab tagarakendus partneri tuvastada.

6.6 Kasutaja õigused

Pärast kasutaja tuvastamist on vaja kontrollida, et kasutajal on lubatud ligi pääseda päritud ressursile. Selleks otstarbeks lõime süsteemi nende haldamiseks ja kontrollimiseks.

Rakenduses on kindel hulk eeldefineeritud privileege, mida saab kasutajatele ja gruppidele jagada. Need õigused võivad käia kindla objekti kohta, näiteks õigus muuta raamlepingut id-ga 5. Kuid saab anda ka õiguse kõigi objektide kohta, näiteks õigus muuta kõiki raamlepinguid. Süsteemis defineerime ära, millised privileegidest peab kasutajal olema, et tema päringut lubataks täita.

Enamus kasutajate õiguseid tulevad nende gruppidest, kuhu nad kuuluvad. Näiteks kõik töötajad saavad õiguse süsteemile ligi pääseda. Hanketalitus osakonna liikmed saavad õigused hallata kõiki hankeid ning luua uusi raamlepinguid.

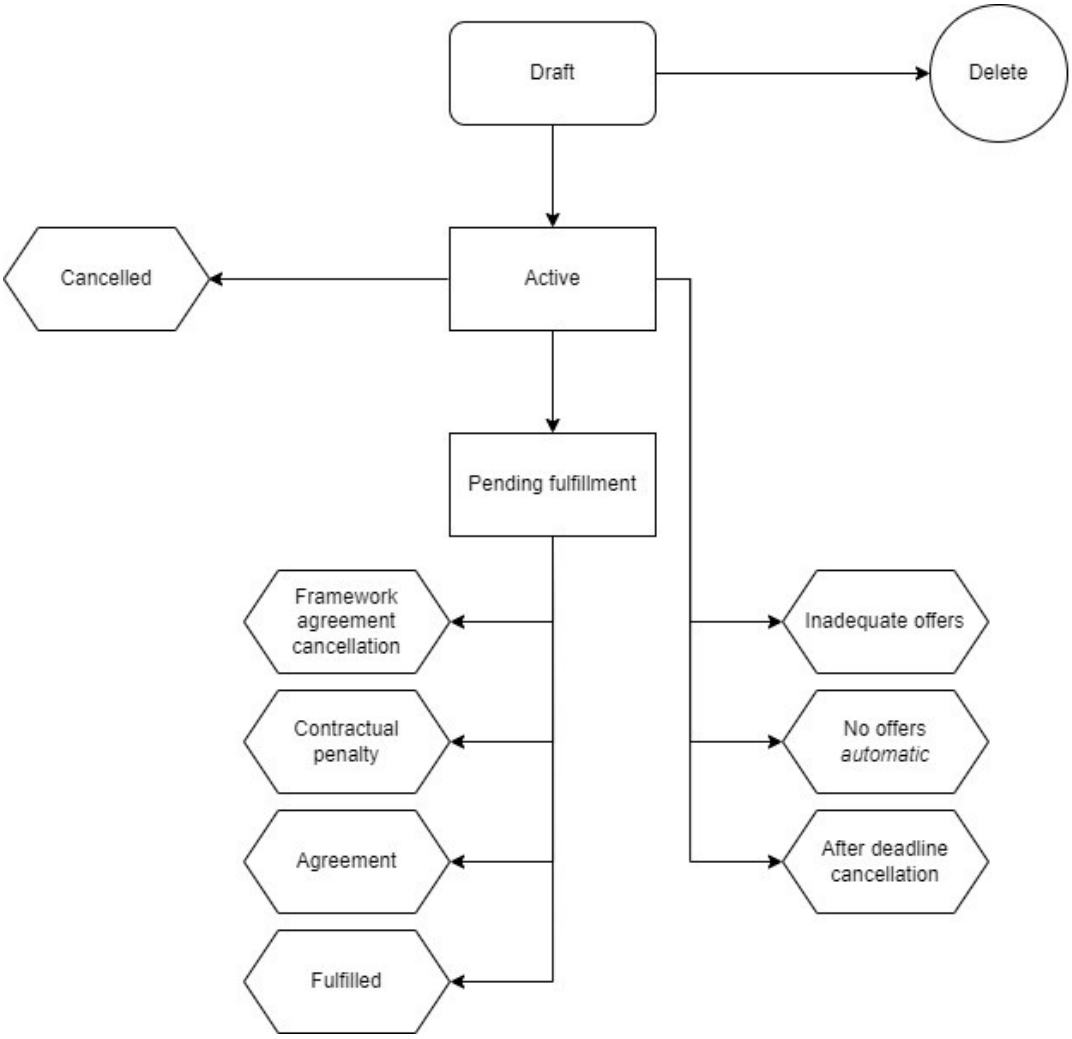
Selle lõputöö raames ei jõutud realiseerida tervet õiguste süsteemi. Suur osa sellest jääb edasiseks arenduseks.

6.7 Minikonkursside ja väikeostude staatuste haldus

Kaks olulist osa olid minikonkursid ja väikeostud, ning kõik mis nendega seotud. Nendel on mitu võimalikku staatust, kogu äriloogika peab täpselt paigas olema ning töötama nii, nagu peab.

Et minikonkursse või väikeoste eristada, olenevalt sellest, kui kaugel elutsükklis mingi konkreetne minikonkurss või väikeost on, nõuab staatuste süsteemi ja haldust. Kõiki

seisundeid saab näha jooniselt 7. Algseisundiks on *draft* ehk mustand. Eduka minikonkurssi või väikeostu seisunditeks on valida nelja staatuse vahel – *framework agreement cancellation, contractual penalty, agreement, fulfilled*. Mitte eduka minikonkurssi või väikeostu puhul saab valida kolme lõppstaatuse vahel – *inadequate offers, no offers* ja *after deadline cancellation*. Lisaks on veel eraldi lõppstaatusteks *delete* ja *cancelled*. Vahestaatusteks on *active* ehk aktiivne ja *pending fulfillment* ehk täitmisel.

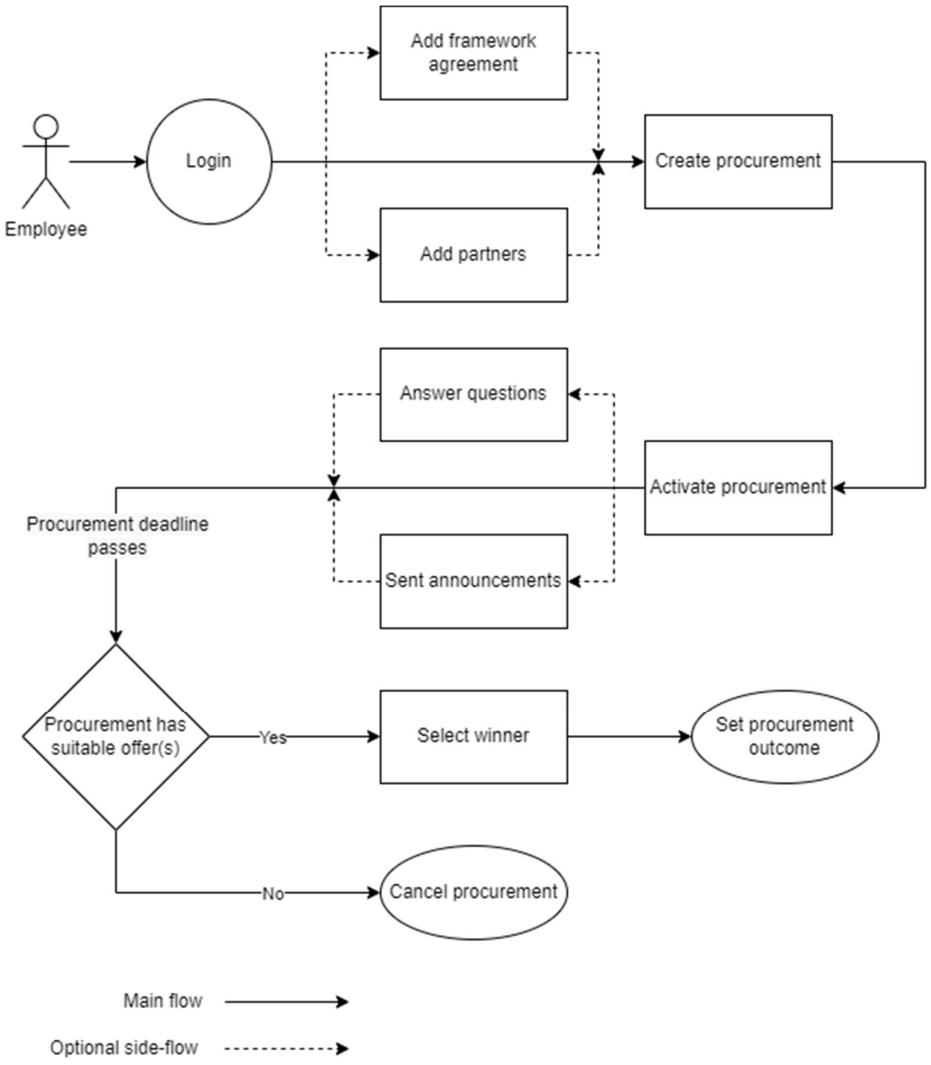


Joonis 7. Minikonkursside ja väikeostude staatuste diagramm

Ülikooli töötaja saab minikonkurssi või väikeostu muuta, sealhulgas selle staatust. Staatuse võimaliku muutmist illustreerib joonis 7. *Draft* ehk mustandi saab ära kustutada (*delete*) või aktiveerida (*active*). Kui leiti, et aktiivne minikonkurss või väikeost ei sobinud, siis saab selle ära tühistada ehk see läheb staatusesse *cancelled* (joonis 3). Edasi lähevad valikud kaheks – kas võitja valiti või mitte. Kui valiti siis saab minikonkurss või väikeost minna staatusesse *pending fulfillment* ehk täitmisel. Edasi saab minna nelja lõpp

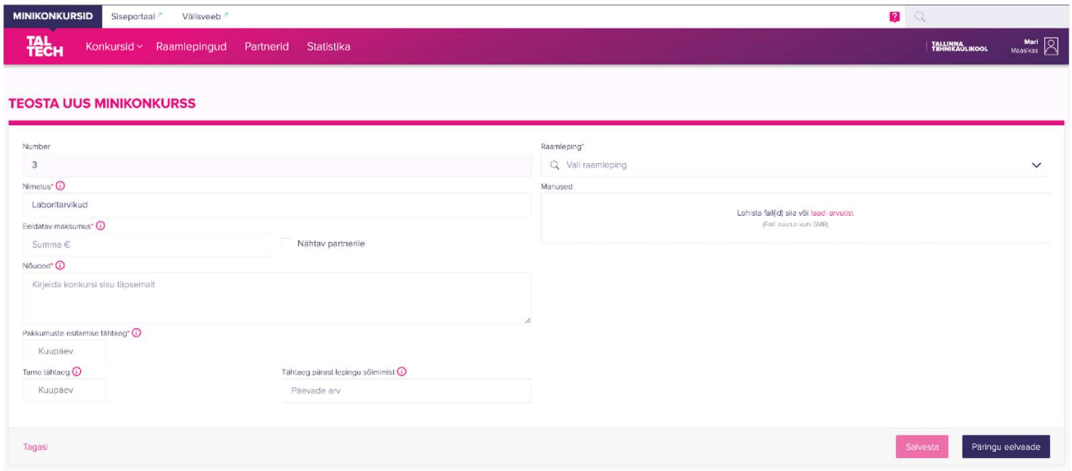
staatusesse. Esimene on *framework agreement cancellation* ehk raamleping mille alusel konkreetne minikonkurss või väikeost põhines, tühistati. Teiseks on *contractual penalty* ehk lepingu rikkumine, täitja saab trahvi. Kolmandaks on *agreement* ehk toimus vale arusaam, aga jõuti poolte kokkuleppele. Kui võitjat ei valitud siis saab minikonkurss või väikeost minna kolme lõpp staatusesse. *Inadequate offers* ehk mitte ühtegi sobivat pakkumist ei saabunud tähtaja möödudes, *no offers*, ehk ei tulnud ühtegi pakkumust tähtaja saabudes, sellele üleminek toimub automaatselt. Viimaks *after deadline cancellation* ehk minikonkurss või väikeost tühistati pärast tähtaja saabumist töötaja poolt.

6.8 Töötaja voog



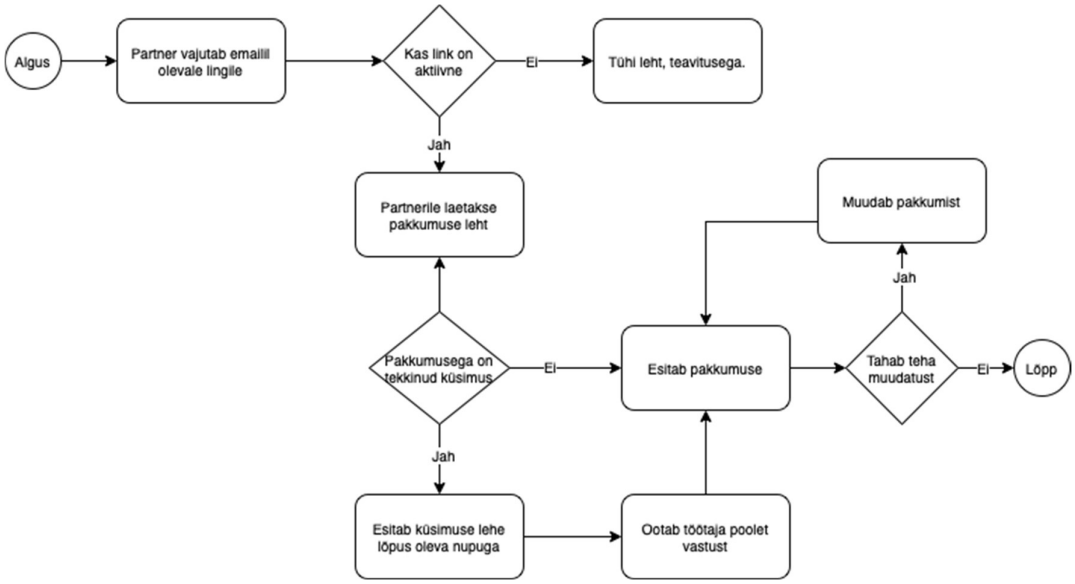
Joonis 8. Töötaja voog

Töötajal on kaks peamist tegevust – partneri lisamine ning minihanke või väikeostu lisamine. Minihange erineb väikeostust selle poolest, et tal on raamleping. Rääkides töötaja voost, toome näitena minihanke teekonna, mida illustreerib joonis 8. Voog hakkab töötaja sisse logimisega, peale seda saab vajadusel lisada raamlepingu või partnereid. Peale minihanke loomist, vorm on kujutatud joonisel 9, saab minihanke aktiveerida. Hanke aktiveerimisel saavad sellega seotud partnerit automaatselt teavituse nii süsteemis kui ka kirja e-kirja teel. Pärast hanke aktiveerimist, saab töötaja partneritel tekkinud küsimustele vastata. Kui hanke tähtaeg on möödas, siis peab töötaja otsustama kas minihankel on sobivad pakkumused või mitte. Kui ei ole, siis töötaja tühistab hanke. Kui jah, siis hange märgitakse täitmisel staatusesse.



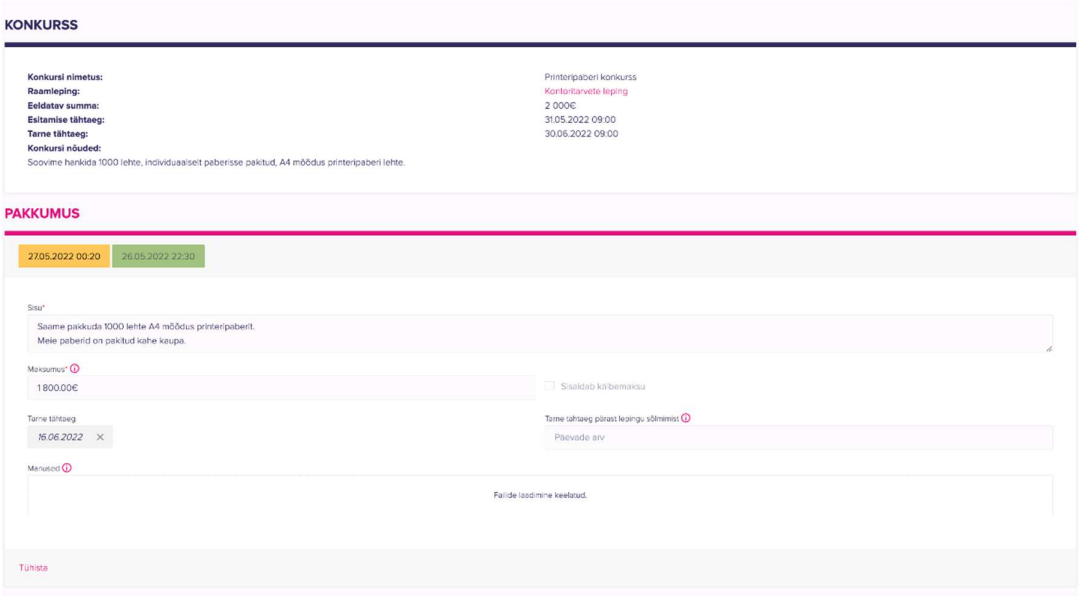
Joonis 9. Minikonkurssi loomise vaade

6.9 Partneri voog



Joonis 10. Partneri voog

Partneri voog hakkab sellest et rakendus saadab automaatselt partnerile e-kirja minikonkurssi või väikeostu kohta. Kui tegu on vana lingiga, siis kuvatakse tühi leht ning antakse partnerile teada, et link oli vananenud. Kui link on endiselt aktiivne, siis kuvatakse partnerile kõik tema pakkumused selle konkreetse minikonkurssi/väikeostu kohta. Edasi saab partner küsimust esitada pakkumuse kohta küsimus, mida saab teha lehe lõpus oleva nupuga. Küsimusele saab vastata töötaja. Kui ei tekkinud küsimust või kui küsimusele vastati ära, siis edasi saab partner pakkumuse esitada, mis on kujutatud joonisel 11, minikonkursile või väikeostule. Pärast pakkumuse esitamist saab ka teha muudatust pakkumusele.



Joonis 11. Pakkumuse loomise vaade

6.10 Dokumentide süsteem

Dokumentide süsteemi jaoks kasutasime Laraveli sisse ehitatud failide süsteemi integratsiooni. Dokumendid salvestatakse vastavalt keskkonnamuutuja väärtusele otse failisüsteemi. Dokumendi metaandmed salvestatakse andmebaasi. Kõige olulisem nendest on unikaalne identifikaator, mille alusel salvestatakse iga dokument eraldi kausta. Nii välditakse sama nimega dokumentide konflikti. Piirangut dokumentide tüüpidel pole, maksimaalne dokumendi suurus, mida üles saab laadida on 20 MB.

7 Testimine

Testisime nii taga- kui esirakendust. Tagarakenduse puhul tähendas see nii manuaalset testimist kui ka testide kirjutamist. Esirakenduse puhul kasutasime erinevatel põhjustel ainult manuaalset testimist.

7.1 Tagarakendus

Tagarakenduse testimise jaoks kirjutasime ühik ja integratsiooni testid. Nende kirjutamisel üritasime kõiki võimalikke juhte ära katta. Ühik testid testivad mingit kindlat asja, nagu kindla teenuse kindlat meetodi. Integratisooni testid testivad mitut asja, näiteks minikonkurssi GET endpointi. Kood ei ole täielikult testidega kaetud, olulisemad äriloogika vood on aga ära kaetud. Lisaks testisime kõik endpoindid käsitsi läbi.

7.2 Esirakendus

Esirakenduses on ühikteste vähe. Kuna suur osa äriloogikast asub tagarakenduses sai esirakenduses ühiktestidega testitud vaid tähtsamad universaalsemad funktsioonid. Peamiselt manuaalne voogude testimine ja valideerimine.

8 Valideerimine

Valideerimine toimus kõigepealt omakeskis, siis andsime valideerimiseks ülikooli poolsetele esindajatele ja kõige lõpuks suuremale ülikooli töötajate seltskonnale. Valideerimise tagasiside andmine käis läbi ühiskasutuses oleva faili, mida rakenduse valideerija sai täita. Kuna ülikooli töötajad on just selle süsteemi kasutajad, siis pool rakenduse funktsionaalsuse valideerimisest on kaetud. Partnerite loogika valideerimist ei ole tehtud kuna tellija ei ole jõudnud veel täielikult valideerida. Valideerimise tulemused on toodud peatükkis 9.

Seniste valideerimis tulemuste põhjal võib eeldada, et rakendus täidab oma esialgseid eesmärke. Kuna praegused tulemused on suhteliselt algsed, ei saa seda veel nende põhjal täiesti kindlalt väita.

9 Tulemused

Lõputöö raames valmis tagarakenduse baasfunktsionaalsus. Üheks peamiseks baasfunktsionaalsuseks on endpoindid, mis võimaldavad nõutud äriloogikat realiseerida. Kokku tuli 37 endpointi ja päringut. Endpointide funktsionaalsuste täitmiseks sai loodud ka kontrollerid ja teenused. Kontrollereid tekkis 13, teenuseid 7. Erinevate olemite staatuste haldamiseks tekkis 8 tabelit andmebaasi, kokku tekkis 28 tabelit andmebaasi äriloogikaga seoses (sh ka staatuste tabelid).

Lisaks kuulub baasfunktsionaalsuste alla ka e-kirjade automaatne saatmine, ülikooli töötaja sisselogimine läbi Azure AD, kolmandate osapoolte kasutajate tugi, õiguste süsteem, dokumentide süsteem.

Tuleviku hooldusteks ja arendusteks lisasime ka Swaggeri dokumentatsiooni ning kasutajaliidese, et edasistel arendajatel oleks lihtsam süsteemi hallata, parandada, töös hoida. Koodi töökindluse mõttes on ühik- ja integratsioonitestid lisatud olulisematele äriloogika osadele, taga- ja esirakenduse päringuid aitab püsti seatud NGINX suunata, GitLabist näeb nii taga- kui esirakenduse arenduse kulgu, näeb täpselt kes mingit ülesannet tegi.

Esirakenduses uued ja uuendatud vaated. Võrreldes aine raames valmis saanud rakendusega, tuli algul palju refaktoreerida ja parandada. Üksikasjalikult sai üle käidud kõik vormid ja vaated. Kliendi üheks sooviks oli saada statistiline ülevaade toimunud hangetest ja iga hanke kohta saada samm-sammuline ülevaade. Selleks valmis eraldi statistika vaade, kus kasutades filtreid võimalik filtreerida ja otsida hankeid ja genereerida neist aruanded. Lisaks sellele sai tehtud lepingule ja hankele detailvaated, kus samuti eraldi aruandeid alla tõmmata saab.

Senine kliendi peamine tagasiside on olnud peamiselt erinevad pisivead, mis on liidestamisega tekkinud. Hetkel ootame põhjalikumat tagasisidet. Näiteks nupud ebamugavates kohtades, mõni validatsioon ei ole korrektne ja muud pisivead. Lisaks tulid ka paar väikest äriloogilist viga, aga üldpildis kogu äriloogika on olemas.

Hetkel on paar suuremat funktsionaalsust veel arendus- ja liidestusjärgus. Õiguste süsteemi toimimisloogika on valmis. Vaja on veel see konfigureerida ja parandada vead, mis välja tulevad. Viimaseks on veel õigetes kohtades vaja saata välja e-kirjad.

10 Kommentaarid

Üldiselt läks kogu arenduse protsess väga hästi, teatud probleemid tulid ikka ette, eriti kui tegemist oli keerukamate funktsionaalsustega nagu autentimine ja Swaggeri lisamine. Tänu varem paika pandud projekti haldusele sai kõik mured ja probleemid kiiresti lahendatud, kas omakeskis või kui tegemist oli äriloogilise probleemiga, siis koosolekutel tellijaga sai nendele probleemidele lahendus leitud.

10.1 Esirakendus

Esirakenduse arenduse koha pealt React võimaldas väga lihtsalt ja kiiresti kasutajasõbralikku kasutajaliidest kirjutada, igati õige valik. Sama käib ka TypeScripti kohta. Kasuks tuli ka TalTech Styleguide, mis pani stiilid hästi paika. Reacti peamisteks miinusteks on struktuuri hoidmine. On väga oluline teada, kuidas rakendust osadeks jagada.

10.2 Tagarakendus

Tagarakenduse koha pealt Laravel vajas natuke aega ja süvenemist, eriti Spring Booti pealt tulles. Aga kui põhi asjadele pihta sai, tundus, et arendus läks sama kiirelt kui Sprint Bootiga tarkvaratehnika aine raames. Teatud asju tegi Laravel isegi väga lihtsaks, nagu sinna sisse ehitatud e-kirjade süsteemi kasutamine. Ainuke erinevus võrreldes tarkvaratehnika ainega oli teside kirjutamine – neid sai palju rohkem ja põhjalikumalt kirjutatud. Tarkvaratehnika aine fookus oli rohkem mingi baasrakendus valmis saada, mida hiljem oleks hea jätkata, lõputöö fookus oli ikkagi töökindel rakendus valmis saada.

10.3 Andmebaas

MySQL-ga probleeme ei tekkinud, kõik olime seda varem kasutanud ning integratsioon Laraveliga oli väga hea.

10.4 Koodi haldus

Giti kasutamine oli väga hea valik, temaga sai lihtsalt muudatusi ellu viia, muudatusi tagasi võtta jne, igati tuli kasuks.

10.5 Projekti haldus

GitLabi kasutamine oli ka väga mugav, läbi kasutajasõbraliku kasutajaliidese sai uusi ülesandeid teha, vanasid kinni panna, *pull requeste* teha, *pull requeste* üle vaadata, *milestone* luua, ajapikku tekkis hea ülevaade projekti staatusest.

10.6 Swagger

Swaggeri kasutuselevõtt tuli ka plussiks. Vaatamata selle ülesseadmise keerukusele sai sellest väga palju kasu. Arenduse käigus oli lihtne ja kiire kontrollida, kas muudatused toimivad nii, nagu me eeldasime. Lisaks tekkis ülevaade sellest, mis päringuid on üldse võimalik saata – see aitas kaasa kahe poole liidestamisele.

11 Kokkuvõte

Lõputöö raames valmis rakendus, mis võimaldab läbi kasutajaliidese TalTechi töötajatel ja partneritel minikonkursse või väikeoste luua, muuta, kustutada, dokumente alla/üles laadida, küsimusi esitada minikonkurssitele või väikeostule küsimusi, vastata nendele samadele küsimustele. Partnerid saavad konkreetseltele minikonkurssitele/ostudele pakkumine esitada, töötajad saavad valida, kes konkreetse minikonkurssi või väikeostu võitis. Lisaks on e-maili kaudne teavitamine, näiteks kui partner võitis minikonkurssi siis teada teavitatakse e-maili teel sellest. Olemas on ka õiguste süsteem - et kõik süsteemi kasutajad saaks nii palju teha kui vaja aga mitte midagi mida ei tohiks teha saada. Loodud on kasutaja autentimine, töötajate puhul käib see läbi Azure AD kasutajate, kasutaja lisatakse ka andmebaasi (esimesel sisselogimisel). Tuleviku hooldusteks, arendusteks sai lisatud Swagger dokumentatsioon, et edasistel arendajatel oleks lihtsam süsteemi hallata, parandada, töös hoida. Koodi töökindluse mõttes on ühik- ja integratsioonitestid lisatud olulisematele äriloogika osadele, taga- ja esirakenduse päringuid aitab püsti seatud NGINX suunata, GitLabist näeb nii taga- kui esirakenduse arenduse kulgu ning kui kaugele lõpuks jõuti, lisaks näeb täpselt kes mingit ülesannet tegi.

11.1 Edasised arendused

Valminud rakenduses on puudusi, mis tuleb veel lahendada enne rakenduse kasutusele võttu. Tagarakenduses on vaja veel lisada äriloogilisi kontrolle. E-kirja saatmises on lahendatud konkursi kutse saatmine aga lisaks tuleb teostada muud hankega seotud e-kirjade saatmised. Õiguste süsteemi kontseptsioon sai analüüsitud ja teostatud esmased funktsionaalsused, kuid ajapuuduse tõttu veel lõpetamata. Endiselt pole teostatud täielik süsteemi auditimine, millega saab näha süsteemis tehtud muudatusi.

Esirakenduses on vaja liidestada õiguste manageerimine ja statistika vaated. Reguleerida töötaja vooge läbi õiguste. Statistika vaade sai esmase välimuse, kuid juurde tuleb veel lisada raportite genereerimine. Keel ei ole dünaamiliselt muudetav ja seetõttu tuleb sõnastusi parandada läbi koodi.

Süsteemina tuleb juurde luua lepingud konkursi võitjaga ja kõik sellega seonduv äriloogika, mida pole üldse kliendiga arutatud. Lisaks parandada pisivigu ja kirjutada juurde teste.

Kasutatud kirjandus

[1] „React“, 2022. [Võrgumaterjal]. Saadaval: https://reactjs.org/. [Kasutatud 08.05.2022].

[2] „Laravel“, 2022 [Võrgumaterjal]. Saadaval: https://laravel.com/. [Kasutatud 08.05.2022].

[3] Boriss Zahharov, Kristjan Variksoo, Marek Lahk, Jaspar Jaansoo, Kauri Riivik, Reiko Roopärg, Rait Kulbok, Margus Rätsep, Janar Velleste, „Tarkvaratehnika“, aine tarkvaratehnika ITI0208 raames, 2019.

[4] „Spring Boot“, 2022 [Võrgumaterjal]. Saadaval: https://spring.io/projects/spring-boot. [Kasutatud 08.05.2022].

[5] „TalTech Styleguide“ [Võrgumaterjal]. Saadaval: https://gl.ttu.ee/pfe/styleguide. [Kasutatud 08.05.2022].

[6] „Server-Side Rendering“ [Võrgumaterjal]. Saadava: https://www.heavy.ai/technical-glossary/server-side-rendering. [Kasutatud 08.05.2022].

[7] „Angular vs React vs Vue“, 2022 [Võrgumaterjal]. Saadaval: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/. [Kasutatud 08.05.2022].

[8] „Laravel vs Symfony“, 2019 [Võrgumaterjal]. Saadaval: https://asperbrothers.com/blog/laravel-vs-symfony/. [Kasutatud 08.05.2022].

[9] „Laravel vs Phalcon“, 2022 [Võrgumaterjal]. Saadaval: https://www.educba.com/phalcon-vs-laravel/. [Kasutatud 08.05.2022].

[10] „DB-Engines Ranking“, 2022 [Võrgumaterjal]. Saadaval: https://db-engines.com/en/ranking. [Kasutatud 08.05.2022].

[11] „Postman vs Swagger“, 2022 [Võrgumaterjal]. Saadaval: https://www.educba.com/postman-vs-swagger/. [Kasutatud 08.05.2022].

[12] „Apigee vs Swagger“, 2022 [Võrgumaterjal]. Saadaval: https://mindmajix.com/apigee-vs-swagger. [Kasutatud 08.05.2022].

[13] „What is a reverse proxy?“, 2022 [Võrgumaterjal]. Saadaval: https://www.cloudflare.com/en-gb/learning/cdn/glossary/reverse-proxy/. [Kasutatud 08.05.2022].

[14] „Apache vs NGINX“, 2021 [Võrgumaterjal]. Saadaval: https://serverguy.com/comparison/apache-vs-nginx/. [Kasutatud 08.05.2022].

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

Meie, Marek Lahk, Oliver Laidma ja Kauri Riivik

1. Anname Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Väikeostude keskkond" mille juhendaja on Priit Järv
   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 autoritele.
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.