TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Rasmus Juurik 193443IAIB
Jevgeni Serkin 221137IAIB

PROGRAMMEERIMISÜLESANNETE HALDAMISE REGISTRI AURORA ÕIGUSTE JA STATISTIKA SÜSTEEMI JUURDEARENDUS

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: Rasmus Juurik, Jevgeni Serkin

30.05.2022

Annotatsioon

Selle bakalaureusetöö eesmärgiks on kirjeldada programmeerimisülesannete haldamise registri Aurora edasiarendusi. Lõputöö tellijaks on Tallinna Tehnikaülikooli infotehnoloogia teaduskonna dekanaat ja informaatika programmijuht Ago Luberg.

Aurora on veebirakendus, mis on mõeldud TalTech tarkvara instituudi õppejõududele. Rakendus lahendab probleemi, kus õppejõud peab iga kursuse alguses õppeaine programmeerimisülesanded kopeerima eelmise kursuse GitLab salvest, uude salve. Aurora süsteemis on ülesanded ühekordselt koos statistikaga, mis annavad kasutajale hea ülevaate olemasolevatest ülesannetest.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 25 leheküljel, 8 peatükki, 8 joonist.

Abstract

Further Development of Permission and Statistics System for Programming Assignment Management Registry Auror

The purpose of this bachelor's thesis is to describe further developments of programming assignment management registry Aurora. The project was commissioned by Ago Luberg, who is the Dean's Office at School of Information Technologies and the Programme Director of Informatics in Tallinn University of Technology.

Aurora web application is designed for teachers at the Institute of Software of Taltech. The application tries to solve the problem where teachers have to copy programming assignments in the beginning of each course, from one repository to another. In the Aurora system, no duplicates of assignments exist and there are statistics linked with each assignment, which gives a good overview of all existing programming assignments.

The thesis is in Estonian and contains 25 pages of text, 8 chapters, 8 figures.

Lühendite ja mõistete sõnastik

API	*Application programming interface,* rakendusliides
BE	*Back-end,* tagarakendus
CRUD	*Create, Read, Update, Delete*: loo, loe, uuenda, kustuta.
CSS	*Cascading Style Sheets,* astmelised stiililehed
FE	*Front-end,* esirakendus
HTTP	*Hypertext Transfer Protocol*¸ hüperteksti edastusprotokoll
JPA	*Java Persistence API,* Java püsivus rakendusliides
JSON	*JavaScript Object Notation,* JavaScript objekti tähistus
JWT	*JSON Web Token,* JSON Veebimärk.
REST	*Representational State Transfer,* tarkvaraarhitektuuri laad
SASS	*Syntactically Awesome Style Sheets,* stiililehe keel
SPA	*Single-Page Application,* üheleherakendus
SQL	*Structured Query Language,* struktuurpäringukeel
UI	*User interface,* kasutajaliides
CI/CD	*Continuous Integration and Continuous Development,* pidev integratsioon ja pidev juurutamine/tarnimine

Sisukord

Jooniste loetelu

1 Sissejuhatus

Üks suur osa Tallinna Tehnikaülikooli infotehnoloogia teaduskonna erialal õppimisel on programmeerimisoskuste ning sellega seonduva omandamine. Programmeerimise õpetamiseks ja praktiseerimiseks kasutatakse ette antud programmeerimisülesandeid. Nende ülesannete haldamine võib erinevate keskkondade ja õppeainete raames minna tülikaks ning aeganõudvaks. Õppejõul puudub üldine ülevaatlik pilt kõikidest ülesannetest.

1.1 Taust ja probleem

Oskar Pihlaku poolt loodud bakalaureusetöö „Programmeerimisülesannete haldamise register Aurora“ [1] üritas lahendada mainitud probleemi. Aurora veebirakendus on mõeldud TalTechi õppejõududele kasutamiseks. Rakendus annab hea ülevaate kõikidest programmeerimisülesannetest ja võimaldab otsida märksõnade kaudu sobilikke ülesandeid. Rakendus suudab leida antud ülesannete hoidlast korduvad ülesanded, seega oleksid kõik ülesanded registris ühekordselt. Keskkond väldib olukorda, kus õppejõud peab kopeerima programmeerimisülesandeid iga semestri alguses.

2021. aasta kevadel Oskar Pihlaku lõputöö tulemusel loodud Aurora veebirakendus omab küll vajalikku funktsionaalsust, kuid palju tähtsaid osi ei suudetud projekti suure mahu tõttu valmis saada. Seetõttu ei ole leidnud rakendus ka tänaseni kasutust. Aurora veebirakendusest on puudu:

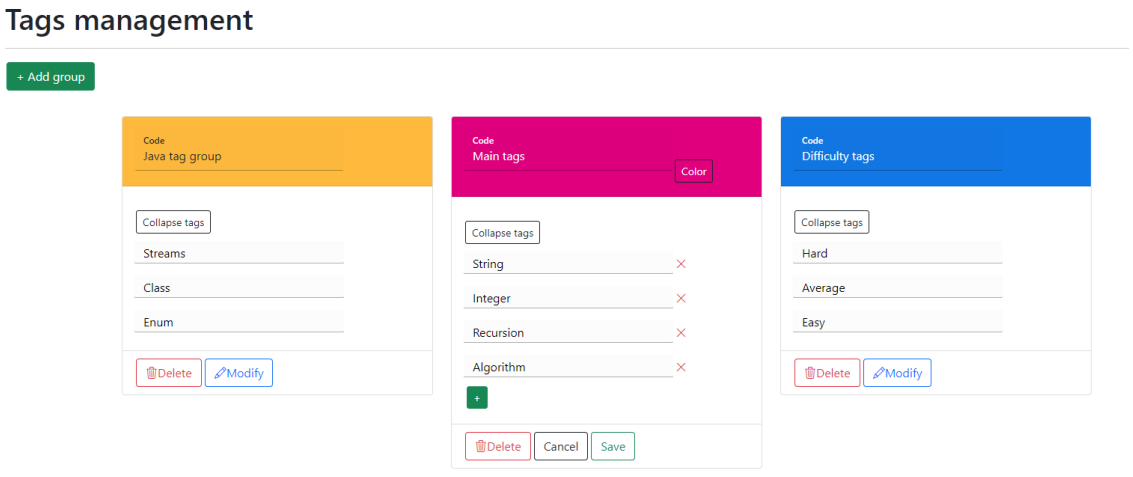
- Autentimine
- Programmeerimisülesannete statistika ülevaade
- Kasutajate õiguste haldamine

1.2 Olemasolev rakendus

Olemasolev Aurora veebirakendus võimaldas kasutajal hallata GitLabi salvesid. Rakendus kasutab salvede kättesaamiseks eraldi loodud Aurora GitLab kasutajat, millele on loodud *Personal Access Token*. Kui sellele kasutajale anda GitLab keskkonnas salve kloonimis õigus, muutub see Aurora registris nähtavaks ja tekib võimalus salv kloonida registri süsteemi. Rakendus võimaldab otsida salvesi nime järgi, uuendada salve (sünkroniseerida GitLabi keskkonnas olevate andmetega) eraldi, kui ka kõiki korraga, ja uuendada salve autori andmeid eraldi, mis sisuliselt pärib salvest lisainfot salves olevate ülesannete jaoks.

Juhul, kui süsteemis on programmeerimisülesandeid sisaldav salv, saab neid ülesandeid otsida märksõnade ja nime järgi. Otsingutulemused tulevad pagineeritud kujul.

Rakenduses on võimalus luua ülesannete jaoks märksõna silte ning sildi gruppe. Märksõnu saab siduda programmeerimisülesandega, et hiljem oleks võimalus otsida vajalikku ülesannet märksõna kaudu. Märksõna sildi gruppide eesmärk on grupeerida mitmeid märksõnu endaga, et oleks lihtsam leida sobilik märksõna ülesandega sidumiseks (Joonis 1. Märksõna gruppide haldamine.).



Joonis 1. Märksõna gruppide haldamine.

Õppejõud loovad igal aastal uue Gitlab salve algava kursuse jaoks. Kuna samu programmeerimiseülesandeid kasutatakse läbi mitme aasta, tekib olukord, kus üks ja sama ülesanne on mitmes erinevas salves. Aurora rakendus suudab tuvastada need korduvad ülesanded kasutades Levenshteini kauguse algoritmi [2].

1.3 Ülesande püstitus

Käesolev bakalaureusetöö on olemasoleva programmeerimisülesannete haldamise registri Aurora edasiarendus. Eesmärk on lisada puuduv funktsionaalsus ja vajadusel parandada olemasolev, et rakendus saaks olema kasutuskõlblik.

Esmatähtis on liidestada rakendusele autentimine. Autentimise puuduse tõttu ei saa antud süsteemi tarnida ega avalikult kasutada. Eksisteerib oht, et võõras isik siseneb keskkonda ning pääseb ligi vabalt andmetele, mis ei ole mõeldud avalikkusele. Autentimise puhul on plaanis kasutada OAuth 2.0 protokolli [3] läbi TalTech Azure Active Directory.

Aurora rakendus on mõeldud õppejõududele ja nende abilistele. Selleks, et abiõppejõud saaksid ka rakendust kasutada, oleks tarvis luua õiguste haldamise süsteem. Muul juhul pääseksid abiõppejõud ligi kõikidele andmetele ning neil jääks võimalus luua, modifitseerida, kui ka kustutada, süsteemis olevaid andmeid. Õiguste haldamise süsteem võimaldaks administraator õigustega kasutajal määrata teistele kasutajatele vajalikud ja sobilikud õigused.

Statistika puuduse tõttu puudub ülevaade programmeerimisülesannete keerukusest. Ei ole teada, kui palju tudengeid sai antud ülesandega hakkama. See raskendab õppejõul teha parim ülesande valik kursuse jaoks. Probleemi lahendamiseks loome API (*Application programming interface*) lõpp-punkti, kuhu TalTechi Arete teenus saadaks automaatselt statistilisi andmeid.

Kui rakenduses on olemas töötav autentimine, õiguste haldamise süsteem ning programmeerimisülesannetel on küljes statistika, siis tuleks üle vaadata ja kontrollida kogu funktsionaalsus, et ei eksisteeriks vigu. Töö lõpus antakse rakendus üle tellijale, kellelt saaks tagasisidet süsteemi parandamiseks.

2 Projekti kirjeldus

Projekti tegemisel lähtusime meeskonnaga üldtuntud tarkavaraarenduse printsiipidest, mis on ka kasutusel suurtes ettevõtetes.

2.1 Metoodika

Projekti raames kasutame meeskonnana agiilset tarkvara arendamise meetodit nimega ekstreemprogrammeerimine. Ekstreemprogrammeerimine on üks paljudest populaarsetest agiilsetest arendusmeetoditest. See on tõestatud olema väga edukas nii väikestes kui ka suurtes ettevõtetes. Ekstreemprogrammeerijad kommunikeerivad omavahel tihedalt. Nad hoiavad projekti disaini lihtsa ja puhtana [4].

Meeskonnaga ja juhendajaga/tellijaga kommunikeerimiseks, nii kirjalikult kui ka heli/pildi vahendusel, kasutasime suhtlustarkvara Discordi. Valisime Discordi, kuna kõik liikmed, sealhulgas ka juhendaja, olid varasemalt seda tasuta tarkvara kasutanud. Lisaks toetab Discord teksti vormindamise süsteemi Markdown [5], kus on võimalik kirjutada koodi juppe lihtsasti loetavate plokkidena. Seal me lõime kokku neli erinevat suhtluskanalit:

- **Aurora general**. Selles suhtluskanalis arutati peamiselt lõputöö korralduslike küsimuste ning kõige muu seonduva üle.
- **Aurora tehniline**. Tehnilises suhtluskanalis kirjutati tarkvaraarendus käigus tekkinud probleemide ja lahendust üle.
- **Aurora memos**. Aurora memos kanalis postitati pärast igat meeskonna koosolekut memo vormis kokkuvõte, mis koosolekul sai arutatud ja otsustatud.
- **Daily summary**. *Daily summary* ehk päevase kokkuvõtte vestluskanal loodi mõttega, et iga meeskonna liige annab seal teada oma päevase arendustöö tulemustest lühikese kokkuvõttena vabas vormis.

Koosolekuid pidasime nädalas kokku kolm tükki: kaks meeskonna vahel ja üks koos juhendajaga. Kõik koosolekud olid internetis Discord tarkvara vahendusel. Koosolekutel arutasime probleeme, tehtud tööd ning edasisi plaane. Discordis on võimalus jagada enda arvutiekraani teistele, mis võimaldab mugavalt näidata meeskonnaliikmele või juhendajale tehtud arendustööd.

Aurora rakenduse koodibaase hoiame Tallinna Tehnikaülikooli GitLabis juba varasemalt eksisteerivates salvedes. Kasutasime erinevaid GitLabi võimekusi, et hallata projekti iganädalaseid ülesandeid, progressi, ajakulu ja tähtaegasid. Jagasime ülesanded laiali erinevate tähtpunktide (*milestone*) vahel GitLabis, et saaksime luua ka ülesandeid, mida teeme millalgi tulevikus või millel on väiksem prioriteet.

Enne suurema tüki arendamist teostasime meeskonnaga, kui ka juhendajaga/tellijaga, põhjaliku analüüsi probleemi lahendamiseks. Analüüsi tehes on verbaalne suhtlus kiire ja mugav, kuid tihti ei osata vestluse käigus korrektselt ennast väljendada või puudub kohene arusaam ja idee. See tekitab arusaamatusi ja aeglustab protsessi. Sellepärast otsustasime suuremate osade arendamisel luua GitLab keskkonnas analüüsiks mõeldud ülesande pileti (*issue*) ning seal kirja panna kõik küsimused, plaanid ja skeemid. Nii on kogu informatsioon selle osaga dokumenteeritult ühes kohas, kus saab igal ajal vaadata, mis on planeeritud ja otsustatud.

Koodi kirjutamisel lähtusime puhta-koodi printsiipidest [6]. Kirjutasime ainult vajalikku koodi ning mitte midagi rohkem. Jälgisime koodi joondust ja selle loetavust. Ei korranud koodi lõike mitmes erinevas kohas ja võimalusel lisasime koodisiseseid kommentaare keerukamatele osadele, et oleks parem arusaam kirjutatud loogikast. Meie poolt kirjutatud kood on järjekindel ja lihtsasti mõistetav.

Koodi veakindluse tagamiseks kirjutasime nii tagarakenduse (*back-end*) kui ka esirakenduse (*front-end*) jaoks teste. Tagarakenduses lõime üksusteste (*unit testing*) tähtsametele, suuremat äriloogikat hõlmavatele, meetoditele. Esirakenduses lõime samuti üksusteste, kui ka otsast-lõpuni teste (*end-to-end test*), komponentide korrektse funktsionaalsuse tagamiseks.

2.2 Tööjaotus

Projekti tööülesanded jagasime üksteise vahel võimalikult võrdselt laiali vastavalt pädevustele ja praktilistele teadmistele. Kõigi liikmete rolliks oli arendada *full-stack* stiilis. Algse plaani järgi jagasime tööd järgnevalt:

1. Rasmus Juurik liidestab rakendusele autentimise.
2. Jevgeni Serkin vastutab ülesannete statistika eest.
3. Kolmas meeskonnaliige loob õiguste haldamise süsteemi.

Peamine eesmärk oleks see, et kõik liikmed oleksid tuttavad ja teadlikud tervest süsteemist. Rollid ei oleks rangelt fikseeritud.

Kahjuks teatud põhjustel ei saanud meie meeskonna kolmas liige projektis jätkata. Seetõttu pidime tööjaotust natukene muutma. Rasmus Juurik võttis lisaks autentimisele ka enda peale õiguste haldamise süsteemi. Jevgeni Serkini vastutuseks jäi statistika ja Aurora esirakenduse uue kujunduse peale viimine. Kasutajaliidese viimine uuele kujundusele ei olnud algselt plaanis, selle vajadus tekkis töö käigus.

2.3 Töökäik

Koosolekutel arutasime plaane ja vajalikke töid. Vajalikud arendustööd märkisime GitLab keskkonnas piletina (*issue*) ja määrasime töö ühele meeskonnaliikmele. Pileti loomisel valisime paljuütleva pealkirja, mis annaks kohe aimu probleemist/plaanist. Lisaks kirjeldasime pileti kirjelduses probleemi veelgi detailsemalt. Nii mõistab iga isik, kes piletit vaatab, probleemi sisu. Samuti lisasime iga pileti juurde märksõna silte. See andis aimu, millega on tegu ja mis staatus piletil hetkel on. Märksõnadega sildistatud koondusid kenasti piletite tahvlile, mis andis hea ülevaate planeeritud, käimasolevatest kui ka lõpetatud töödest.

Arendustöö pileti enda peale võtnud meeskonnaliige lõi enne töö alustamist uue haru. Uus haru põhines põhiharust (*main branch*). Arendaja tegi lokaalselt koodimuudatused ning postitas (*push code*) seejärel koodi eelnevalt loodud uude harusse. Koodi postitamisel viitas arendaja postitamissõnumis lahendatava pileti peale. Nii tekkis pileti tasemel GitLabis märge, et see koodi uuendus lahendas just selle pileti probleemi. Juhul,

kui kood GitLabi postitatud ning algne probleem koodi tasemel lahendatud, määras pileti volitaja (*assignee*) pileti ülevaatajaks (*reviewer*) teise meeskonnaliikme. Seega, ta lõi tõmbetaotluse (*pull request*). Teine meeskonnaliige vaatas tehtud koodimuudatused üle ning vajadusel jättis kommentaare, mida võiks parandada või selgitada. Kui parandused tehtud ja kõik korrektne, võis pileti jaoks loodud haru mestida (*merge*) põhi haruga.

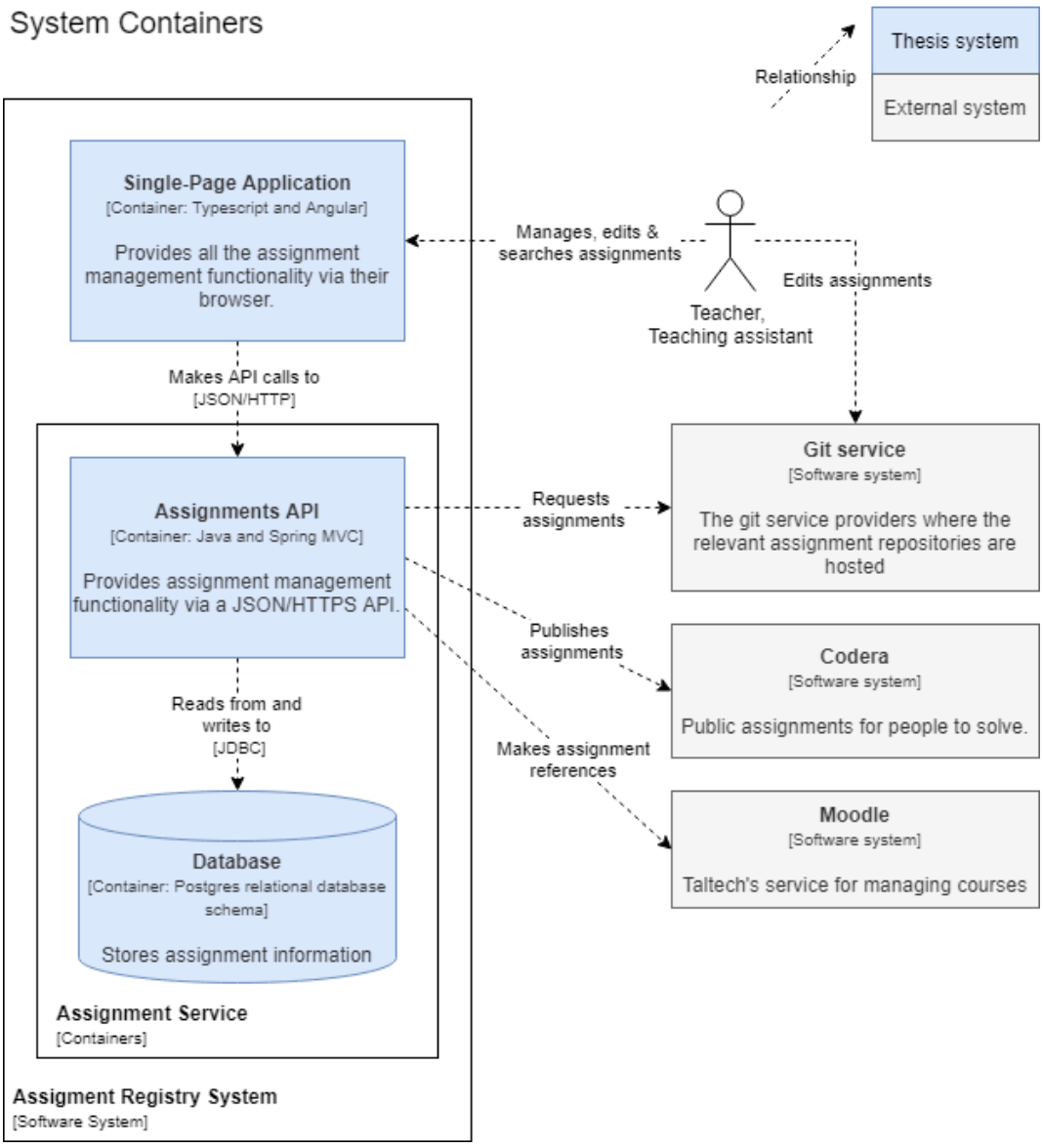
Kui arendustöö käigus tekkis küsimusi või arusaamatusi, olime julged küsima meeskonnaliikmelt või juhendajalt abi/nõu.

3 Projektidisain

Aurora terviklik süsteem hõlmab enda all kolme peamist osa: tagarakendus (rakendusliides), esirakendus (kasutajaliides) ja operatsioonid. Enamik tehnoloogilisi ja arhitektuurilisi valikuid oli tehtud eelneva autori poolt.

3.1 Projekti arhitektuur

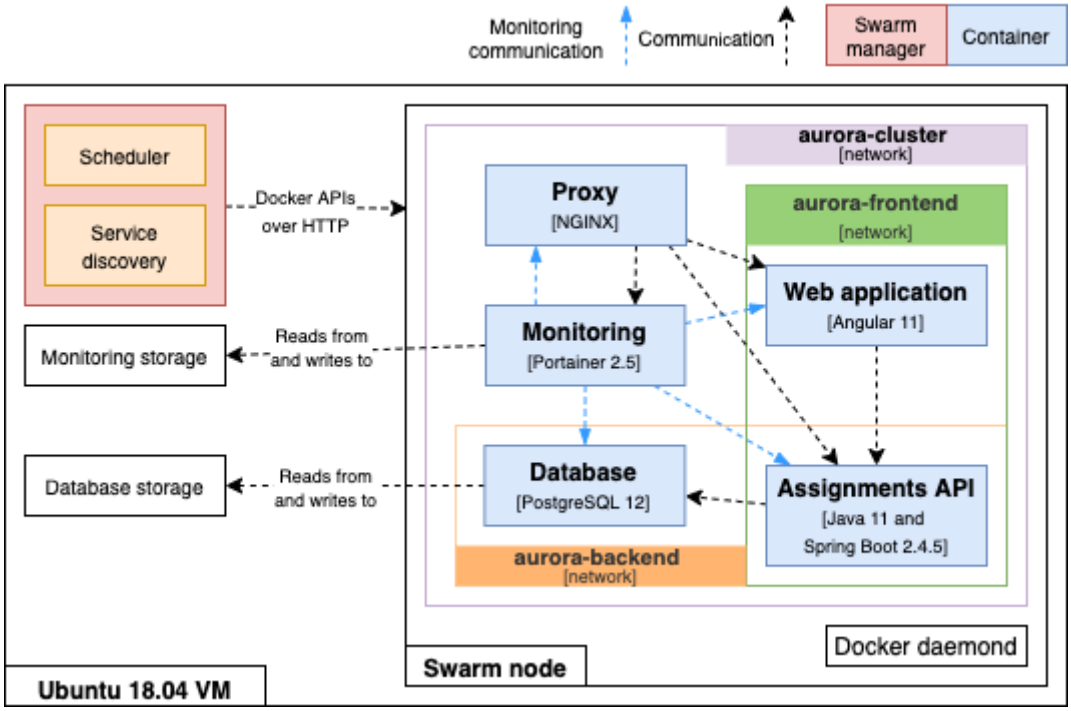
Aurora rakendus kasutab monoliitset lähenemist projekti arhitektuuril. Süsteemi konteinerid on kirjeldatud Joonis 2. Süsteemi konteinerite arhitektuur . See on tehtud seetõttu, et modelleerimis otsuseid ja kihtide piire on lihtsam parandada, kui selged domeeni piirid on paigas. Rakenduse komponendid on organiseeritud ja kokku pandud ülalt-alla neljakihilisse arhitektuurimustrisse. Kihid on iseseisvalt eraldatud süsteemi osad, mis käituvad ülejäänud süsteemiga kui väliste üksustena [1].



Joonis 2. Süsteemi konteinerite arhitektuur [1].

Tagarakenduse ja esirakenduse vaheline suhtles käib läbi REST (*Representational State Transfer*) API.

Aurora rakendus jookseb TalTechi serveri virtuaalmasinas Ubuntu, ühe sõlme klastri kujul (*single node cluster*) Docker Swarmis. Swarmi klaster on agnostiline, mis tähendab, et konteinereid ja võrke saab tarnida erinevatest projektidest. Loogikat kirjeldab paremini Joonis 3. Docker Swarm . [1].



Joonis 3. Docker Swarm [1].

3.2 Tagarakendus

Aurora tagarakendus on üles ehitatud Spring Boot raamistikule ning kirjutatud Java 14. versiooni programmeerimis keeles. Rakenduse turvalisuse tagamiseks on kasutusel Spring Security. Alampeatükkidena on välja toodud tähtsamad tehnoloogiad, mis on projekti tagarakenduses kasutusel.

3.2.1 Spring Boot

Spring Boot on Spring raamistiku lahendus, mis pakub terviklikku programmeerimis ja konfiguratsiooni mudelit tänapäevasele Java põhisele rakendusele [7].

3.2.2 Spring Security

Spring Security on võimekas ja laialdaselt kohandatav kasutaja autentimiseks ja autoriseerimiseks mõeldud raamistik. Spring Security on *de-facto* standard Spring põhiste rakenduste turvamiseks [8].

3.2.3 Gradle

Gradle on avatud lähtekoodiga rakenduse ehitamist automatiseeriv tööriist, mille paindlikusest piisab peaaegu igat tüüpi tarkvara koostamiseks. Gradle'il on hea võimekus,

mille tööriist saavutab jooksutades ainult neid ülesandeid, mille sisend või väljund muudetud sai, vältides sellega ebavajalikku töö tegemist [9].

3.2.4 Project Lombok

Project Lombok on teek, mis lihtsustab ja kiirendab arendustööd Java klassidega. Annotatsioonide abil saab ära jätta klassi muutujate *getter* & *setter* meetodid. Nii jääb kood lühem ja arusaadavam [10].

3.2.5 Swagger

Swagger on rakenduse API (*Application Programming Interface*) jaoks mõeldud tarkvara tööriist, mis teeb arendaja elu kergeks rakenduse lõpp-punkte testides. Swagger loeb tagarakenduse struktuuri ja loob selle põhjal selge kasutaja-sõbraliku ja interaktiivse API dokumentatsiooni lehekülje [11].

3.2.6 Mockito

Mockito on üksustestide kirjutamise raamistik, mis isoleerib osa testitava rakenduse süsteemist, imiteerides sõltuvuste käitumist. Raamistik võimaldab testida vaid soovitud meetodeid lihtsalt ja koodi-puhtalt [12].

3.3 Esirakendus

Esirakendus on üles ehitatud Angular 13 raamistikuga SPA (*Single-page application*) stiilis. Rakendus kasutab stiilimiseks Bootstrap 5 teeki ja SASS (*Syntactically Awesome Style Sheets*) stiililehe keelt. Alampeatükkidena on välja toodud tähtsamad tehnoloogiad, mis on projekti esirakenduses kasutusel.

3.3.1 Angular

See on võimas tööriist kasutajaliidese arendamiseks. Angulari raamistik pakub kõiki vajalikke koostisosasid, mis on vajalikud ühe korraliku SPA rakenduse arendamiseks [13].

3.3.2 Bootstrap

Bootstrap on stiilide teek, mis lihtsustab kasutajaliidese loomist, kuna pakub laia eelmääratud CSS (*Cascading Style Sheets*) klasside valikut, et arendajal oleks võimalus minutitega panna puhas kujundus püsti [14].

3.3.3 Ngx bootstrap

Ngx-bootstrap on teek, mis pakub taaskasutatavaid Bootstrap komponente, Angular komponentide kujul. Sinna kuuluvad sellised ettedefineeritud komponendid nagu modaal aknad, kuupäeva valik, vahelehed ja nii edasi [15].

3.3.4 TypeScript

TypeScript on programmeerimiskeel, mis võimaldab lisada staatilist tüübikirjeldust, mille abil saab esirakenduse arenduse käigus varem tuvastada vea olukorrad. TypeScript kompileerub ehitamise ajal JavaScriptiks [16].

3.3.5 SASS

SASS on CSSi laiendus stiililehe keel, mis muudab kasutajaliidese stiilimise hõlpsamaks. SASS lubab luua muutujaid, mille abil saab stiililehe koodi muuta lühemaks ja rohkem loetavamaks. Koodi kompileerimise käigus muudetakse SASSI kood CSSiks [17].

3.4 Andmebaas

Aurora rakendus kasutab PostgreSQL objekt-relatsioonilist andmebaasisüsteemi, mis põhineb SQL (*Structured Query Language*) andmebaasi päringukeelel [18].

Püsivad andmeobjektid on defineeritud tagarakenduses Java klassidena, mida nimetatakse olemiteks. Entiteet klassis märgitakse Java muutujad *Column* (veerg) annotatsiooniga iga andmebaasi tabeli veeru jaoks. Sel viisil oskab objektide relatsioonilise kaardistamise tööriist Hibernate teisendada andmebaasi tabeli read Java klassi objektiks [19].

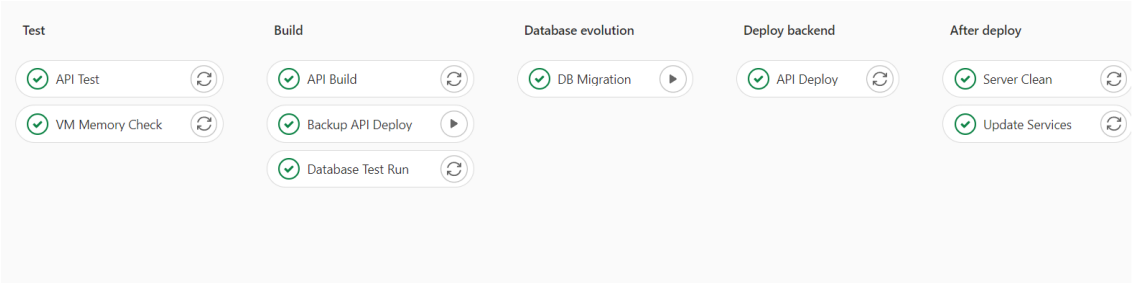
3.4.1 Liquibase

Liquibase on avatud lähtekoodiga andmebaasi skeemi muutmist haldav teek, mis muudab lihtsaks andmebaasi versioonide haldamise. Andmebaasi tabelite loomised, muudatused ja andmete lisamised on kirjeldatud SQL failides. Liquibase auditeerib kõik muudatused eraldi andmebaasi tabelisse, kuhu märgitakse kogu informatsioon muudatustest (autor, kuupäev, tabel) [20].

3.5 CI/CD

Pideva integreerimise ja pideva tarnimise praktika on kasutusel meie projektis. Iga koodimuudatuste tegemisel jooksutatakse konveieris (*pipeline*) olemasolevaid teste, et tagada süsteemi järjepidevus. Nii on pidevalt kontrollitud, et viimased muudatused ei teinud rakendust katki.

Konveieri tööotsasi (*jobs*) jooksutab GitLabi rakendus nimega GitLab Runner. Kõikidel rakenduse komponentidel (esirakendus, tagarakendus, operatsioonid) on olemas konveierid (Joonis 4. Tagarakenduse konveier..



Joonis 4. Tagarakenduse konveier.

3.5.1 Docker

Docker on avatud platvorm rakenduse arendamiseks, tarnimiseks ja jooksutamiseks. See võimaldab rakendusi jooksutada isoleeritud keskkonnas, konteineris. Konteinerisse lisatakse kõik vajalikud teegid ja failid, mida rakendus töötamiseks vajab [21].

4 Edasiarendus

Eelneva autori poolt loodud rakenduses oli puudujääke ja programmivigu, mis tuli töökorras rakenduse loomiseks kõrvaldada. Siin peatükis kirjeldame rakenduse peamisi ning kaalukamaid uusi funktsionaalsusi ja parandusi.

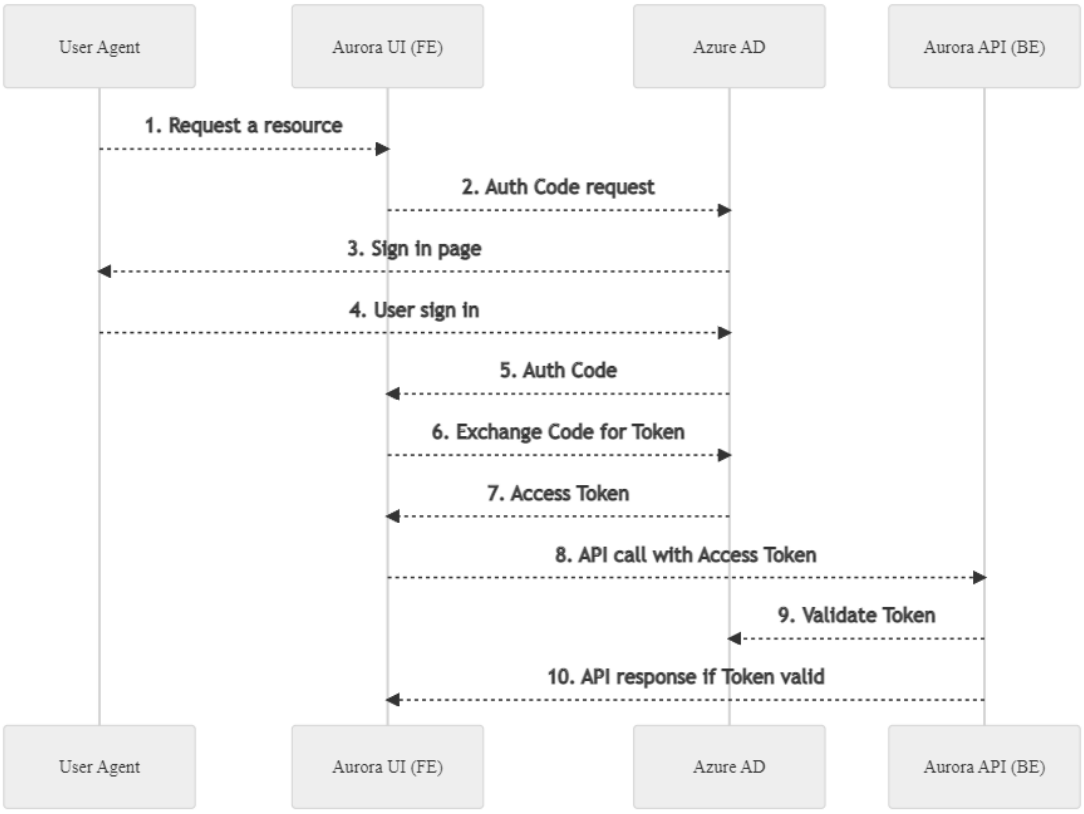
4.1 Autentimine

Autentimisel otsustasime kasutada Open Authorization 2.0 protokolli (OAuth 2.0). OAuth 2.0 on *de-facto* tööstuse standard autoriseerimiseks. Protokoll defineerib neli rolli: ressurssi omanik, klient, ressurssi server ja autoriseerimisserver [22]. OAuth rolle järgides, on meie Aurora kasutajaliides kliendi rollina. Ressursi omanikuks on kasutaja, autoriseerimisserveriks Azure Active Directory ja ressursi serveriks Aurora tagarakendus.

Autentimise puhul otsustasime võtta autoriseerimisserveriks, TalTechi süsteemides levinud, Azure Active Directory (Azure AD). Azure AD on pilvepõhine identiteedi -ja juurdepääsuhaldusteenus. Selle abil saab kasutada standarditel põhinevat lähenemist ühekordse sisselogimise implementeerimiseks oma rakendusele [23]. Kuna Aurora rakendus on mõeldud TalTechi töötajatele, siis tundus väga mõistlik kasutada TalTechi enda Azure AD teenust. Nii saavad autentida ainult ülikooliga seotud isikud, kellel on kehtiv Uni-ID kasutajakonto. Lisaks kasutades Azure AD teenust, puudub meil vajadus hoiustada kasutaja paroole ja muud infot meie andmebaasis, sest Azure AD vastutab selle eest ise.

Aurora rakenduse puhul kasutame OAuth autoriseerimiskoodi voogu (*Authorization Code flow*). Aurora esilehel sisselogimise nuppu vajutades suunab rakendus kasutaja TalTech Azure AD sisselogimislehele, küsides autoriseerimisserverilt (Azure AD-lt) eduka sisselogimise puhul autoriseerimiskoodi. Pärast koodi kättesaamist vahetatakse kood juurdepääsuloa (*Access token*) vastu, mis on JWT (*JSON Web Token*) kujul. Iga HTTP (Hypertext Transfer Protocol) päringuga Aurora tagarakenduse (API) poole, annab esirakendus päringu päisega kaasa Azure AD poolelt allkirjastatud kehtiva JWT.

Tagarakendus kontrollib JWT autentsust, pöördudes autoriseerimisserveri poole. Juhul, kui juurdepääsuluba on kehtiv ning korrektne, annab tagarakendus loa minna päritud lõpp-punkti meetodisse (Joonis 5. Autentimis voog.).



Joonis 5. Autentimis voog.

Esirakendus kasutab autoriseerimisserveri poole pöördumiseks, kui ka JWT värskendamiseks, teeki angular-oauth2-oidc [24]. Tagarakendus kasutab JWT valideerimiseks Spring Security teeki spring-security-oauth2-resource-server [25].

Meie süsteemis eraldi registreerimist ei eksisteeri. See tähendab, et kui kasutaja siseneb esimest korda Aurora veebirakendusse, saab kasutaja läbi Azure AD süsteemi sisse logida. Esirakendus saadab kasutaja info pärimiseks päringu tagarakendusse. Kasutaja info lisatakse andmebaasi juhul, kui tema infot seal ei ole.

4.2 Õiguste haldamine

Enne õiguste haldamise süsteemi arendamist lõime GitLab pileti, kuhu kirjutasime tellijale küsimusi, kuidas võiks olla süsteem lahendatud. Pärast tagasiside saamist, arutasime teostuse üle. Kuna rakenduse autentimine sai tehtud läbi TalTech Azure AD,

oleks olnud võimalus määrata kasutajatele rolle Azure ADs. Sel juhul oleksid kasutaja rollid kaasas kasutaja *Access Tokenis*. See lahendus oleks lihtsustanud meiepoolset arendustööd märkimisväärselt, kuid oleks muutnud Aurora administraatori elu keerulisemaks. Nimelt, tähendaks seesugune lahendus seda, et adminil peaks olema ligipääs TalTech Azure AD Aurora kataloogi ja Aurora kasutajate õiguste haldamine toimuks seal. Vältimaks rakenduse süsteemide eraldatust, otsustasime luua kogu õiguste haldamise Aurora rakenduse siseselt.

Õiguste haldamise süsteemi loomisel lähtusime kasutaja mugavusest ja laiaulatuslikust kohandatavusest. Paindlikkuse huvides otsustasime lisaks rollidele luua ka grupid.

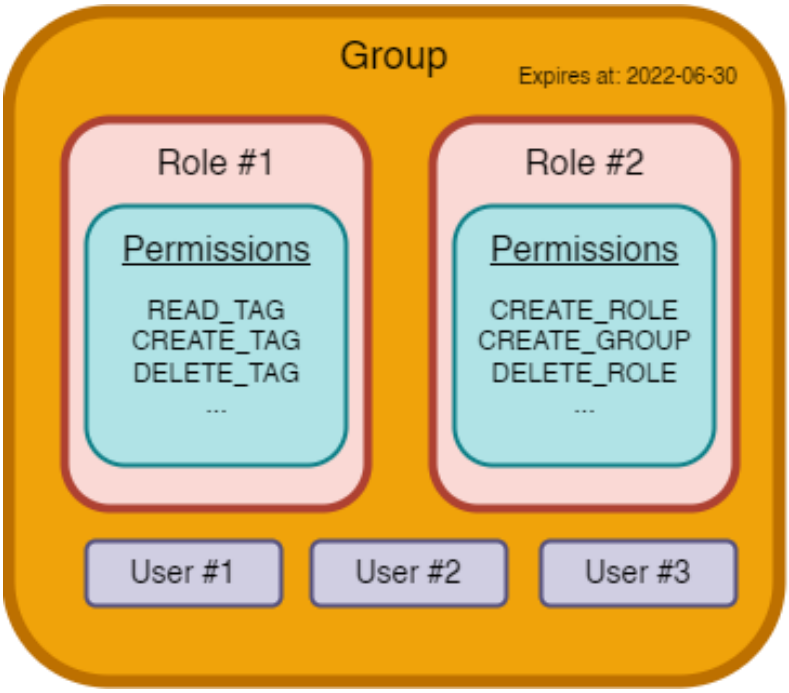
Loodud süsteemis saab Aurora superkasutaja määrata kasutajatele iga rakenduse siseste funktsionaalsuste osas CRUD (*create, read, update, delete*) õigusi eraldi (vt Lisa 6 - Õiguste haldamise süsteemi andmemudel).

Vajalike õigustega kasutaja saab luua uusi rolle. Rollile saab määrata nime ning valida rakendusse eelnevalt kirjutatud õiguste hulgast vajalikud õigused. Loodud rolle saab muuta või kustutada (vt Lisa 2 – Rollide haldamine).

Vajalike õigustega saab kasutaja luua uusi gruppe. Gruppi luues saab määrata:

- Grupi nime
- Aegumis kuupäeva
- Rollid
- Kasutajad

Rolle valitakse olemasolevate rollide seast, kasutajate puhul samuti. Kõik kasutajad, kes kuuluvad gruppi, saavad kõikide gruppi kuuluvate rollide õigused endale. Kasutajad ja rollid võivad kuuluda mitmesse gruppi. Pärast grupi aegumiskuupäeva, kaovad grupis olevatel kasutajatel selle grupi õigused - selle eesmärk on anda abiõppejõududele õigusi vaid teatud perioodiks, näiteks üheks semestriks (Joonis 6. Grupi olemust ja sõltuvusi kirjeldav joonis. (vt Lisa 3).



Joonis 6. Grupi olemust ja sõltuvusi kirjeldav joonis.

Kui kasutajal ei ole vajalikke õigusi mingisuguse tegevuse jaoks, on kasutajaliidese tasemel sellele kasutajale selle funktsiooni vaade peidetud või siis funktsionaalsust tegev nupp töövõimetus olekus (*disabled*).

Tagarakenduse poole pealt on iga API lõpp-punkt turvatud. Tagarakendus kontrollib enne meetodisse sisenemist, kas päringu teinud kasutajal on vajalik õigus olemas. See on saavutatud Spring Security annotatsiooniga *PreAuthorize*. Päringu teinud kasutaja kehtiva JWT seest võetakse informatsioon, mis seotakse andmebaasis oleva kasutajaga. Seejärel kogutakse selle kasutaja õigused kokku ning kontrollitakse, kas API lõpp-punkti poole pöördumise jaoks vajalik õigus on olemas.

4.3 Statistika

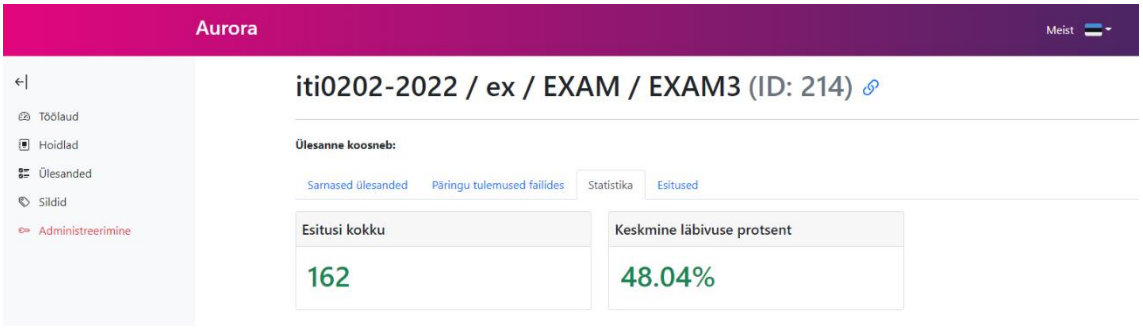
Statistika olemasolu programmeerimisülesannetel on üks tähtsamaid aspekte Aurora rakenduse juures. Õppejõud saavad selle info põhjal teha otsuseid, mis soodustavad õppeaine paremaks muutmist. Olgu see siis ülesande testide eemaldamine, testide lihtsustamine või ülesande täielik ümberkirjutamine.

Nagu ka teiste suurte täienduste puhul, nii ka statistika puhul oli loodud GitLab keskkonnas analüüsi eesmärgiga pilet (*issue*). Seal pidasime tiheda arutelu tellijaga, kus uurisime välja vajadused, nõuded ja ootused. Analüüsi raames oli selge, et mingit valmis

lahendust antud soovide jaoks ei eksisteeri, seega oli vaja luua uus lahendus Aurora süsteemis. Esialgne plaan oli võimaldada Aurora kasutajal sisestada statistika käsitsi agregeeritud kujul, aga hiljem antud vajadus jäi ära. Selgitasime välja, et tulev lahendus hakkab saama esitusi (*submission*) TalTech Arete teenuse poolt. Lõime uue vastava andmemudeli, (vt Joonis 16. Statistika esituste andmemudel.) mis põhines Arete teenuse andmemudelil ja varasemalt arutatud vajadustel, mida tõi välja tellija.

Selleks, et statistikat luua, on vaja andmeid. Andmete kogumine on meil automaatne. Andmete kättesaamine käib läbi TalTech Arete välisteenuse.

TalTech Arete on ülikooli enda automatiseeritud testimise teenus [26]. Sisuliselt, kui üliõpilane lahendab õppeaine raames programmeerimisülesannet ning esitab oma koodi GitLab keskkonda, käivitub Arete süsteemi põhivool, mis jooksutab esitatud koodi peal ülesande teste. Kui testid jooksutatud ja tulemus saadud, saadab Arete süsteem Aurora API lõpp-punkti saadud tulemused. Meie rakendus töötleb saadud andmeid ja seob need süsteemis oleva korrektse programmeerimisülesandega (Joonis 7. Ülesande esituste põhine statistika.. Arete poolt saadud andmetega saab kuvada detailset statistikat, näiteks kuvada ülesande testide läbimis mõõtu eraldi. Nii on näiteks näha, millised testid osutuvad ülesandel liiga rangeks.

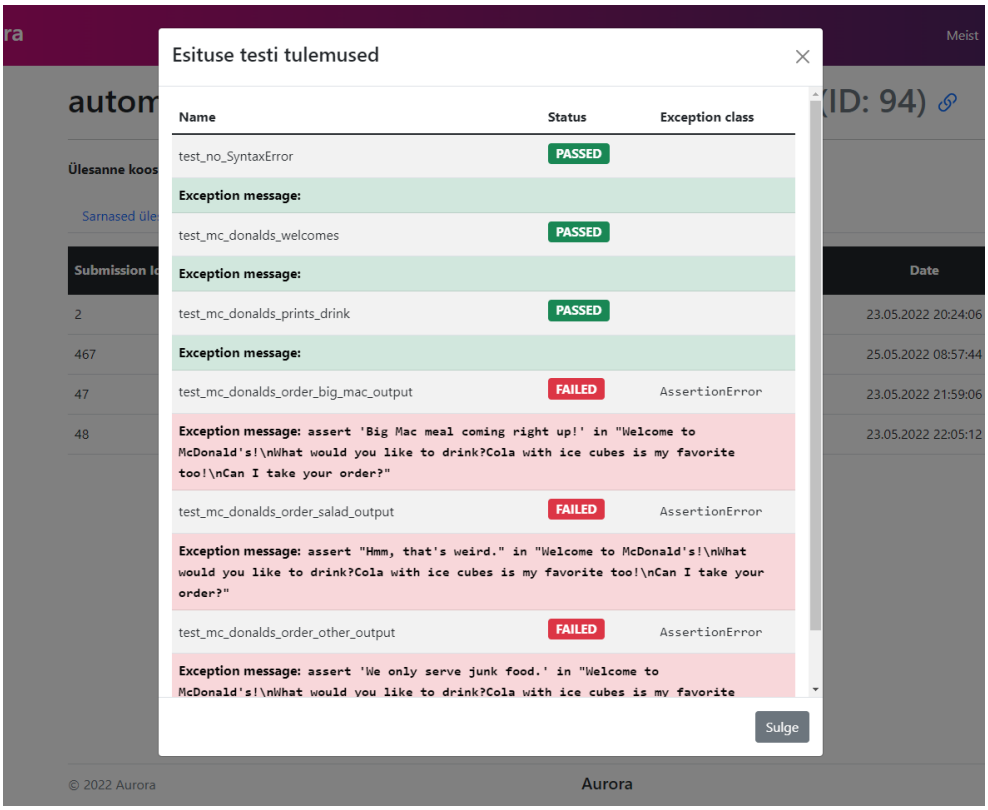


Joonis 7. Ülesande esituste põhine statistika.

Välispoolsete andmete korjamine Aurora süsteemis ei ole rangelt piiratud ainult TalTech Arete süsteemiga. Süsteem sai loodud viisil, mis võimaldab tulevikus saada andmeid ka muudest allikatest.

Süsteem arvestab ka esitustega, mida pole võimalik koheselt siduda ülesandega (ülesanne ei eksisteeri süsteemis). Nende jaoks oli tehtud autonoomne protsess, mis määratud vahemikuga kontrollib kõiki esitusi, millel puudub seos ülesandega. Antud protsessis on rakendatud sama äriloogika, mis ka algsel lõpp-punktil.

Ülesannete statistika on kättesaadav rakenduse kasutajale tervikuna. Statistikat on võimalik näha iga programmeerimisülesande küljes. Sealhulgas on võimalik näha kõiki ülesandega seonduvaid esitusi (*submission*). Iga esituse puhul on võimalik näha ka testide sooritus tulemusi (Joonis 8. Ülesande esitustega seotud testide tulemused..



Joonis 8. Ülesande esitustega seotud testide tulemused.

4.4 Uus esirakendus

Auroral oli eelnevalt olemas viisakas kasutajaliides ning meil ei olnud plaanis kulutada aega selle muutmiseks/uuendamiseks. Kuid ootamatult, esirakenduse arendustöö käigus, tekkis palju probleeme. Näiteks ei olnud võimalus uusi teeke installeerida ja rakendus ei tahtnud tööle minna serveris. Rohkesti panustasime aega nende probleemide lahendamiseks. Saime aimu, et asi võib olla esirakenduse vananenud aluses - mingil põhjusel, olid kasutusel väga vanad raamistiku versioonid. Nii vanad, et isegi neid versioone enam ei toetatud tarkvara välja lasknud arendajate poolt. Kartuses, et versioonide uuendamine toob kaasa uusi probleeme ning säilitab vanad, otsustasime hoopis luua uue esirakenduse nullist, kõige uuemate tehnoloogiatega, ja tõsta olemasolevad funktsionaalsused ükshaaval üle.

Uus esirakenduse sai loodud Angular 13 versioonile. Võtsime kasutusele uued modernsed tehnoloogiad:

- JavaScript testimis raamistik Jest [27].
- Rakenduse ehitus süsteem Nx [28].
- Ilusate komponentide loomis teek ngx-bootstrap [29].

Nende tulemusel ehitab ja käivitub rakendus märkimisväärselt kiiremini kui ta tegi seda enne.

Olemasoleva funktsionaalsuse tõstmisel vanalt uuele märkasime, et vanas esirakenduse koodibaasis eksisteerib palju koodi, mis ei ole kusagil kasutusel (surnud kood). Puhta-koodi printsiipe järgides eemaldasime ebavajaliku koodi ning refaktoreerisime vajalikud koodi osad rohkem loetavamaks. Lisaks kirjutasime komponentidele üksusteste. Puhtam ja loetavam kood muutis edasise arenduse protsessi kiiremaks.

Otsustasime minna ka uue kujunduse peale. Peamine idee kujunduse uuendamise osas oli muuta see visuaalselt ühtlasemaks teiste TalTech arendustega, et lõppkasutajal oleks ühtlasem vaade, kui ta liigub erinevate TalTech rakenduste vahel (Lisa 4 ja 5).

Lõpptulemusena oli uue esirakenduse loomine kasulik ja oluline otsus. Saime lahti ebavajalikust koodist, võtsime kasutusele uusimad tehnoloogilised lahendused ja kujundasime kasutajaliidest mugavamaks. Ja mis kõige tähtsam - likvideerisime eelnevad rakenduse käivitamisel tekkinud probleemid.

5 Tulemused

Projekti eesmärk oli muuta õppejõududele väärtust pakkuv programmeerimisülesannete haldamise register kasutuskõlblikuks. Algsest rakendusest oli puudu õiguste süsteem, ülevaatlik statistika ja hädavajalik autentimine. Samuti oli rakenduses pooleli jäänud funktsionaalsust ja mõningaid programmivigu (*bug*).

Lõputöö tulemusena rakendasime Aurora rakendusele autentimise läbi TalTech Azure Active Directory. Isikud, kes omavad kehtivat Uni-ID kontot saavad Aurorasse sisse logida läbi TalTech Microsofti kasutaja.

Loodud sai ka funktsioneeriv õiguste haldamise süsteem, mis võimaldab määrata administraatori õigustega kasutajal teistele kasutajatele spetsiifilise õigusi. Tegemist ei ole traditsioonilise süsteemiga, kus kasutajatele saab valida ettedefineeritud rolli. Aurora rakenduses saab uusi rolle luua, valides rollile õigusi eraldi. Nii on rakendust kasutaval administraatoril rohkem võimalusi hallata kasutajaid.

Implementeerisime automatiseeritud statistika kogumise. TalTech Arete süsteem saadab meie tagarakenduse pihta andmeid igakord, kui keegi esitab ülesande lahenduse GitLabi. Selle tulemusel on võimalik näha iga programmeerimisülesande küljes statistikat, mis annab aimu ülesande efektiivsusest rakenduse kasutajale.

Aurora kasutajaliides sai uue kujunduse ja uued tarkvaralised versioonid. Kuna olemasoleva esirakenduse raamistiku versioonid olid väga vanad (enam isegi ei toetatud), oli selle uuendamine suure tähtsusega. Versioonide tõttu tekkis palju probleeme uute teekide lisamisega ja rakendus käima saamisega. Uus esirakendus omab uusimad tehnoloogiaid ning omab puhtamat koodibaasi kui enne. Uue esirakenduse loomisel tõstsime olemasolevat koodi ümber. Selle käigus saime lahti kasutamata koodist ja üleliia keerulistest lahendustest, mille tulemuseks oli puhtam ja loogilisemalt struktureeritud koodibaas. Lisaks on Aurora kasutajaliidese kujundus rohkem TalTechi stiililaadi.

Sai parandatud või ümber tehtud ka juba olemasolevat funktsionaalsust. Näiteks parandasime vea, kus märksõna gruppide siseselt ei olnud võimalik kustutada sinna loodud märksõnu.

6 Kommentaarid

Autentimiseks valitud teek angular-oauth2-oidc [24] ei olnud kõige parim valik. Teegi mõistmiseks kulus palju aega ning teegi töötamiseks vajalik kood oli raskesti loetav ja mõistetav. Teegi dokumentatsioon oli puudulik. Lisaks põhjustas teek veebirakenduse esmasel laadimisel keskmiselt 10 sekundilise viivituse. Mis põhjusel viivitus tekib, jäi ebaselgeks.

Õige otsus oli esirakendus luua nullist uusimate tehnoloogiatega. See lahendas teekide versioneerimise probleemid ning vähendas rakenduse käivitus aega märgatavalt.

Tagantjärgi mõeldes sai õiguste haldamise süsteem tehtud liiga keeruline. Rollide osa oleks võinud ära jätta, ja erinevad õigused oleks saanud otse siduda grupiga. Keskendusime süsteemi loomisel liialt kohandatavus võimalustega. Loodud lahendus ei ole otseselt halb, kuid üldpildis oleks saanud lihtsamalt.

6.1 Tellija tagasiside

Kui ülesandepüstituse peamised punktid olid täidetud, andsime tellijale Ago Lubergile rakendust kasutada. Siin on tema tagasiside ja kommentaarid.

Tellija sõnul oli autentimise lisamine väga oluline aspekt. Ilma selleta ei olnud eelnevalt võimalik rakendust kasutada, ega serveris üleval hoida. Kõik õppeaine testid ja ülesandekirjeldused oleksid olnud avalikkusele kättesaadavad.

Märkimisväärselt oluline oli ka tellija arvates õiguste süsteemi olemasolu. Tema sõnul on väga tavaline, et õppeainete abiõppejõududeks on tudengid, kes samal ajal sooritavad mingit muud õppeainet, mis hõlmab enda all programmeerimisülesannete lahendamist. Seega võimalus piirata kasutajate õigusi tuleb väga kasuks. Eriti kui on nii paindlik võimalus seda teha: võimalus luua vajalikud rollid ja grupid ning need kombineerida.

Lisaks mainis Ago, et on väga hea, et esirakendus sai uue koodibaasi ja välimuse. Rakendus töötab kiiremini ning järjekindlamalt, lisaks näeb ka parem visuaalselt välja ja liides on kasutajasõbralikum.

Automaatne statistika olemasolu ülesannetel, oli Ago jaoks üks tähtsamaid osi rakenduse juures. Selle olemasolu tõttu ei pea ta enam iseseisvalt käsitsi statistikat korjama. Nii on tal parem ülevaade ülesannetest ning testidest.

Tellija Ago sõnul on rakendus pärast arendustöid kasutuskõlblik ja tuleval semestril proovib ta seda kasutada. Kindlasti jääb süsteemi osi, mis võiksid olla täiendatud või paremini viimistletud. Näiteks autentimiseks kasutusel olev teek tekitab rakenduse esmasel laadimisel viivituse, mis tema sõnul vajab parandamist.

7 Järeldused ja edasised sammud

Projekti raames sai eelnevalt loodud rakendus kasutuskõlblikuks. Töö käigus esines negatiivseid kui ka positiivseid aspekte.

7.1 Mis läks halvasti?

Järgnevalt on välja toodud projekti arendamise vältel tekkinud negatiivsed asjaolud.

- Kolmas meeskonnaliige lahkus projektist ajal, millal oli juba projekti maht kinnitatud. Seetõttu suurenes koormus allesjäänud liikmetel.
- Raskusi oli eelneva autori poolt loodud süsteemi mõistmisega koodi tasemel. Koodis esines palju keerukaid lahendusi, mille mõistmiseks kulus palju aega.
- Esirakenduse versiooniliste probleemide tõttu, tekkis meil ülesandepüstituse välist lisatööd, millega ei olnud arvestanud.
- Aurora rakenduse autentimisel on kasutusel Azure AD. Selleks, et autentimine teha läbi TalTech Azure AD kataloogi, oli vaja kontakteeruda vastava ülikooli töötajaga, kes seadistaks vajalikud sätted Aurora rakenduse jaoks. Töötaja poolne suhtlus oli kehv ning vajalike seadistustega läks kaua aega.

7.2 Mis läks hästi?

Järgnevalt on välja toodud projekti arendamise vältel tekkinud positiivsed asjaolud.

- Meeskonnaliikmete vahel oli tihe koostöö, mis soodustas pidevat süsteemi mõistmist kõigi poolt.
- Suutsime vähendatud koosseisuga täita meie algse ülesandepüstituse põhiprobleemid.

- Kasutasime modernseid tehnoloogilisi lahendusi, mille käigus omandasime uusi teadmisi ja tuletasime meelde vanu.

7.3 Edasised sammud

Nagu tarkvara arendamisel tavaks, ei saa tarkvara kunagi nii-öelda valmis. Samuti on lood ka meie lõputöö raames arendatud Aurora rakendusega.

Õiguste haldamise süsteem vajab lisa funktsionaalsust. Hetkel saab küll kasutajal ära keelata erinevad tegevused, kuid puudub programmeerimisülesannete nähtavuse piiramine. See tähendab, et praeguses versioonis näevad kõik kasutajad kõiki salvesid ning nende sees olevaid programmeerimisülesandeid.

Oleks vaja ka lisada mingisugune audit logi, kus oleks administraatoril võimalik näha kõiki muudatusi (kustutamised, modifitseeringud, lisamised) kasutaja nimeliselt ja kuupäevaliselt.

Programmeerimisülesannete külge võiks saada lisada, lisaks märksõna siltidele, ka kommentaare. Kommentaarid aitaksid lisada täiendavat informatsiooni ülesande kohta.

Alati saab optimeerida kasutajaliidest: lisades detailsemaid veateateid, uusi tööriistavihjeid (*tooltip*) ja rohkem kohandatavus võimalusi visuaalsuse parandamiseks. Näiteks saaks programmeerimisülesannete lahendamis andmete põhjal luua graafikuid ja diagramme, mis soodustaksid näha paremini statistikat.

8 Kokkuvõte

Projekti eesmärk oli muuta Oskar Pihlaku poolt loodud veebirakendus Aurora kasutuskõlblikuks. Rakenduses olid puudujäägid, mis ei võimaldanud tellijal seda kasutada. Autentimise ja õiguste haldamise süsteemi puuduse tõttu ei saanud rakendust hoida serveris avalikult.

Bakalaureusetöö raames lahendasime püstitatud ülesanded. Aurora veebirakendus on kasutuskõlbulik ning loodetavasti leiab see järgnevatel semestritel tellija Ago poolt kasutust. Vajadusel on rakenduse edasiarendus lihtsam meie poolt lisatud dokumentatsiooni ja puhtama koodi tõttu.

Projekti käik ei olnud lihtne. Üks meeskonnaliige lahkus poole semestri pealt ja see raskendas olusid märgatavalt. Lisaks suurenenud töömahule, oli probleeme esirakenduse tööle saamisega, mille tõttu tekkis palju lisatööd.

Suutsime küll peamised probleemid lahendatud, kuid kindlasti ei olnud see tulemus, mida lootsime. Jäime ajaliselt hätta ning ei suutnud tulemust piisavalt valideerida ja seepärast ka viimistleda.

Kasutatud kirjandus

[1] O. Pihlak, „Programmeerimisülesannete haldamise register Aurora,“ 2021.

[2] „Levenshtein algorithm,“ [Võrgumaterjal]. Available: https://www.cuelogic.com/blog/the-levenshtein-algorithm. [Kasutatud 30 May 2022].

[3] „OAuth 2.0 webpage,“ [Võrgumaterjal]. Available: https://oauth.net/2/. [Kasutatud 30 May 2022].

[4] „Extreme programming,“ [Võrgumaterjal]. Available: http://www.extremeprogramming.org/. [Kasutatud 30 May 2022].

[5] „Discord Markdown Text 101,“ Discord, [Võrgumaterjal]. Available: https://support.discord.com/hc/en-us/articles/210298617-Markdown-Text-101-Chat-Formatting-Bold-Italic-Underline-. [Kasutatud 30 May 2022].

[6] „Clean code principles,“ [Võrgumaterjal]. Available: https://x-team.com/blog/principles-clean-code/. [Kasutatud 30 May 2022].

[7] „Spring Boot,“ [Võrgumaterjal]. Available: https://spring.io/projects/spring-framework. [Kasutatud 30 May 2022].

[8] S. Security, „Spring Security,“ [Võrgumaterjal]. Available: https://spring.io/projects/spring-security. [Kasutatud 30 May 2022].

[9] „Gradle,“ [Võrgumaterjal]. Available: https://docs.gradle.org/current/userguide/what_is_gradle.html. [Kasutatud 30 May 2022].

[10] „Project Lombok,“ [Võrgumaterjal]. Available: https://projectlombok.org/. [Kasutatud 30 May 2022].

[11] „Swagger,“ [Võrgumaterjal]. Available: https://swagger.io/docs/specification/2-0/what-is-swagger/. [Kasutatud 30 May 2022].

[12] „Mockito,“ [Võrgumaterjal]. Available: https://site.mockito.org/. [Kasutatud 30 May 2022].

[13] „Angular,“ [Võrgumaterjal]. Available: https://angular.io/guide/what-is-angular. [Kasutatud 30 May 2022].

[14] „Bootstrap,“ [Võrgumaterjal]. Available: https://getbootstrap.com/. [Kasutatud 30 May 2022].

[15] „Ngx-bootstrap,“ [Võrgumaterjal]. Available: https://valor-software.com/ngx-bootstrap/#/. [Kasutatud 8 May 2022].

[16] „TypeScript,“ [Võrgumaterjal]. Available: https://www.typescriptlang.org/. [Kasutatud 30 May 2022].

[17] „SASS,“ [Võrgumaterjal]. Available: https://sass-lang.com/documentation. [Kasutatud 30 May 2022].

[18] „PostgreSQL,“ [Võrgumaterjal]. Available: https://www.postgresql.org/. [Kasutatud 30 May 2022].

[19] „Hibernate,“ [Võrgumaterjal]. Available: https://hibernate.org/orm/. [Kasutatud 30 May 2022].

[20] „Liquibase,“ [Võrgumaterjal]. Available: https://docs.liquibase.com/home.html. [Kasutatud 30 May 2022].

[21] „Docker,“ [Võrgumaterjal]. Available: https://www.docker.com/. [Kasutatud 30 May 2022].

[22] „OAuth 2.0,“ [Võrgumaterjal]. Available: https://auth0.com/intro-to-iam/what-is-oauth-2/#how-does-oauth-20-work. [Kasutatud 30 May 2022].

[23] „Azure Active Directory,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis. [Kasutatud 30 May 2022].

[24] M. Steyer, „angular-oauth2-oidc,“ [Võrgumaterjal]. Available: https://www.npmjs.com/package/angular-oauth2-oidc. [Kasutatud 30 May 2022].

[25] „Spring Security OAuth 2.0 Resource Server,“ [Võrgumaterjal]. Available: https://docs.spring.io/spring-security/reference/servlet/oauth2/resource-server/index.html. [Kasutatud 30 May 2022].

[26] E. Vompa, „TalTech Arete,“ [Võrgumaterjal]. Available: https://ained.pages.taltech.ee/it-doc/arete/. [Kasutatud 30 May 2022].

[27] „Jest,“ [Võrgumaterjal]. Available: https://jestjs.io/. [Kasutatud 30 May 2022].

[28] „Nx,“ [Võrgumaterjal]. Available: https://nx.dev/getting-started/nx-and-angular. [Kasutatud 30 May 2022].

[29] „Ngx Bootstrap,“ [Võrgumaterjal]. Available: https://valor-software.com/ngx-bootstrap. [Kasutatud 30 May 2022].

[30] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

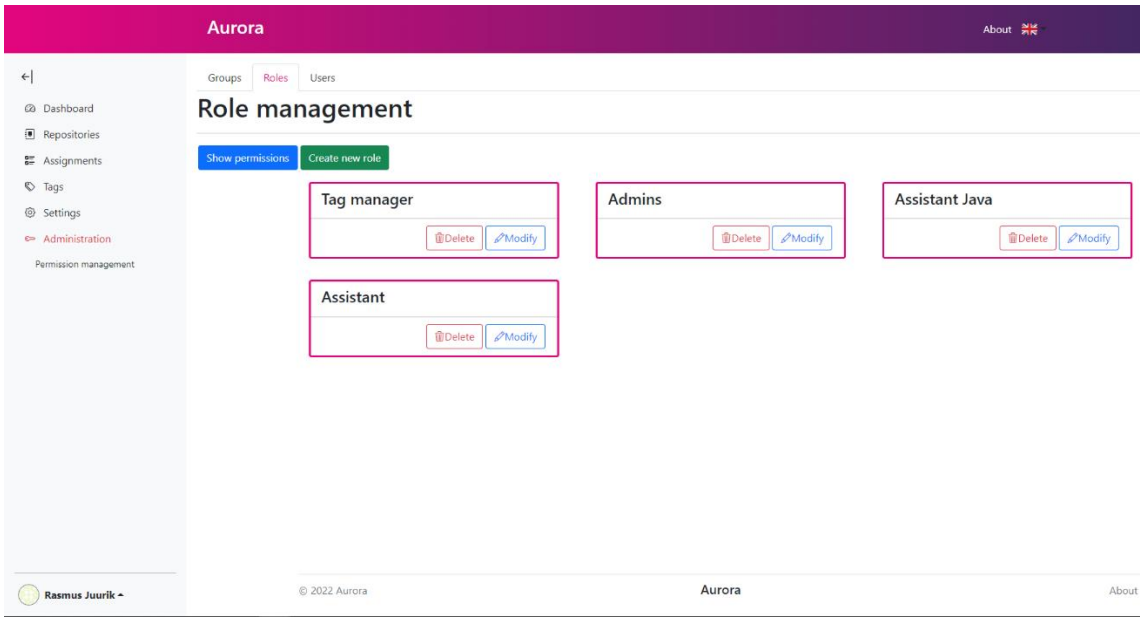
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Meie, Rasmus Juurik ja Jevgeni Serkin

1. Anname Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Programmeerimisülesannete haldamise registri Aurora õiguste ja statistika süsteemi juurdearendus“, 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 autoritele.
3. Kinnitame, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

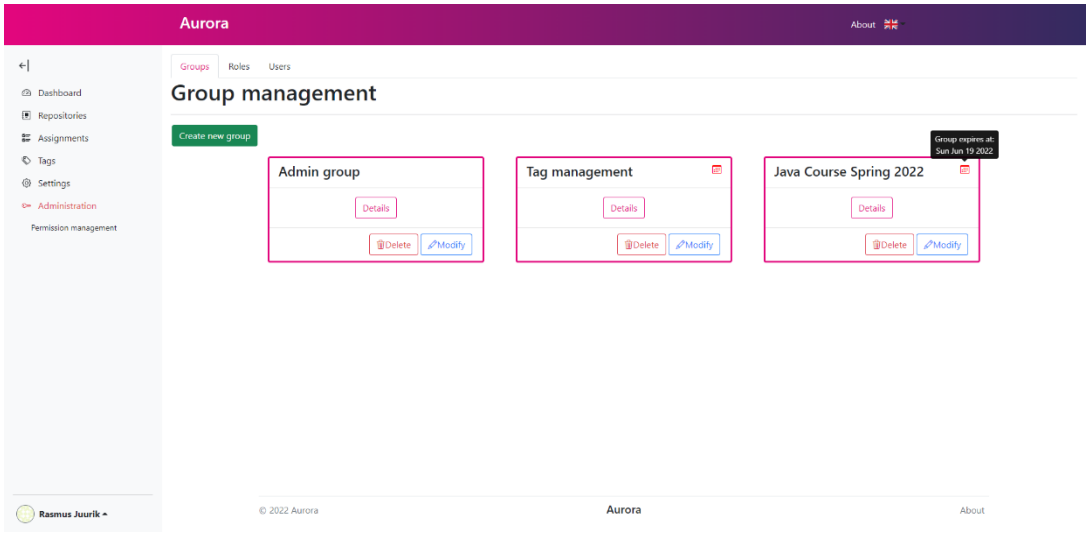
1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või 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 – Rollide haldamine



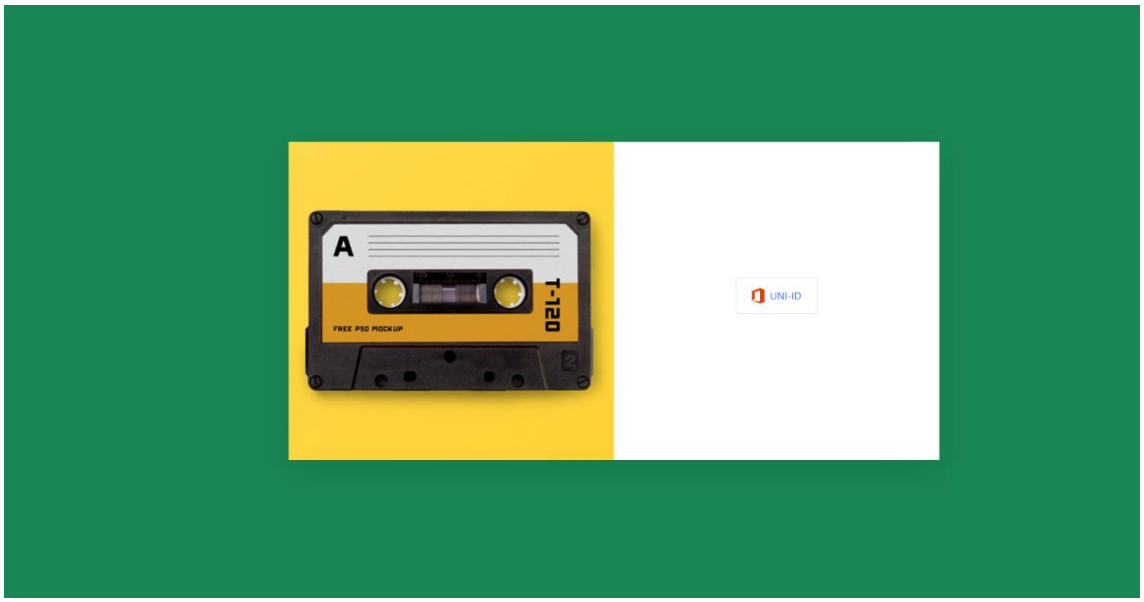
Joonis 9. Rollid haldamise vaade.

Lisa 3 – Gruppide haldamine

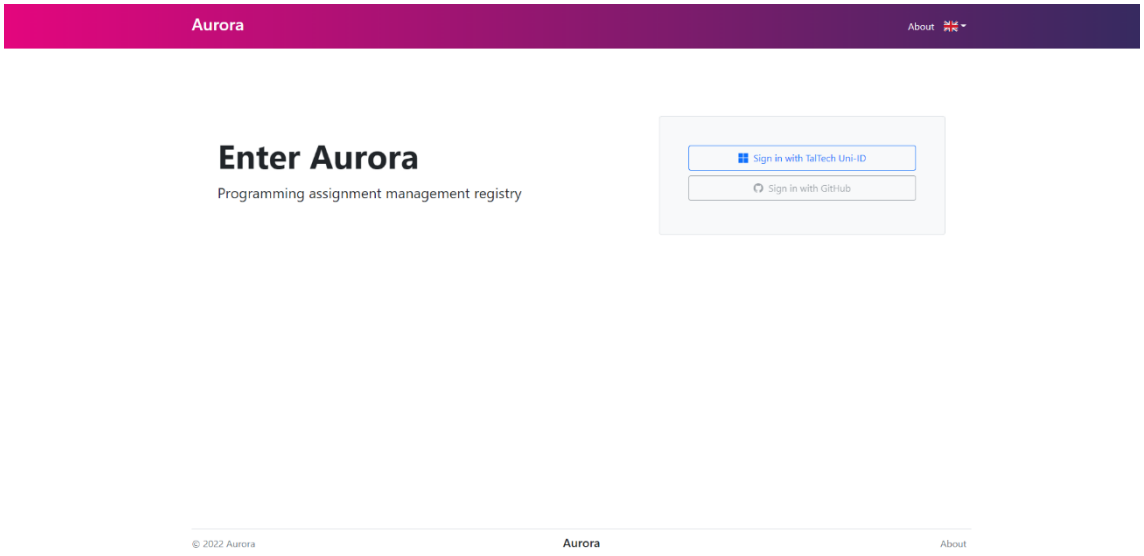


Joonis 10. Gruppide haldamise vaade.

Lisa 4 – Kasutajaliideste võrdlus, sisselogimine

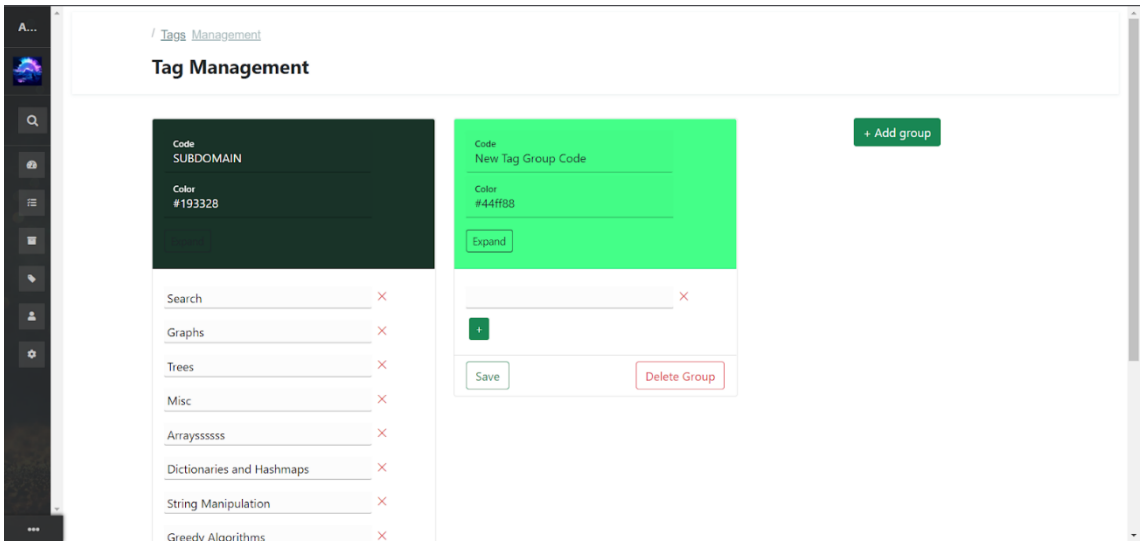


Joonis 11. Sisselogimise vaade vanal kujundusel.

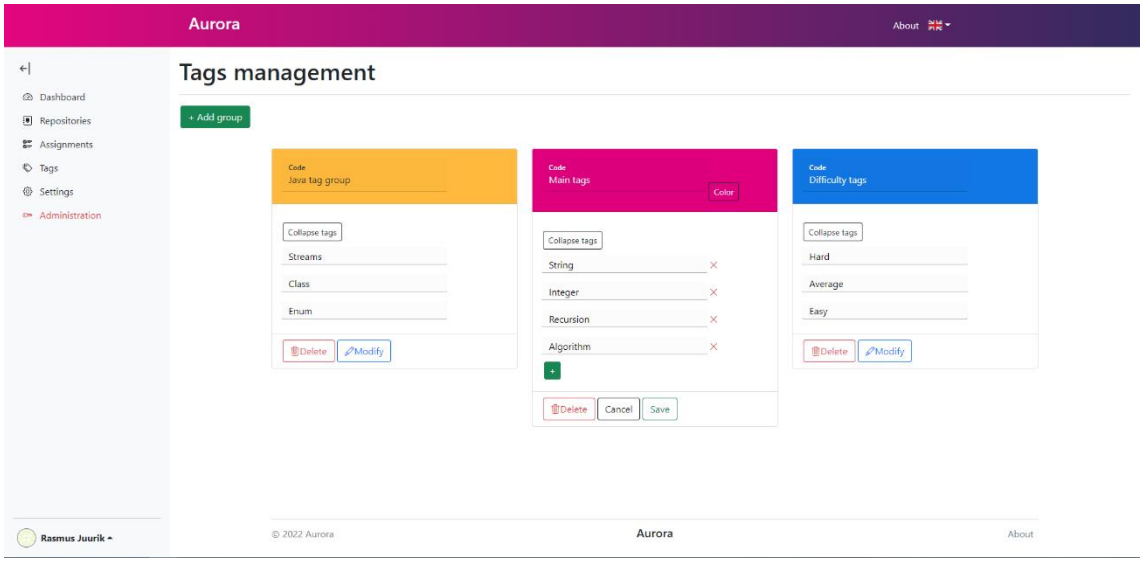


Joonis 12. Sisselogimise vaade uuel kujundusel.

Lisa 5 – Kasutajaliideste võrdlus, siltide haldamine

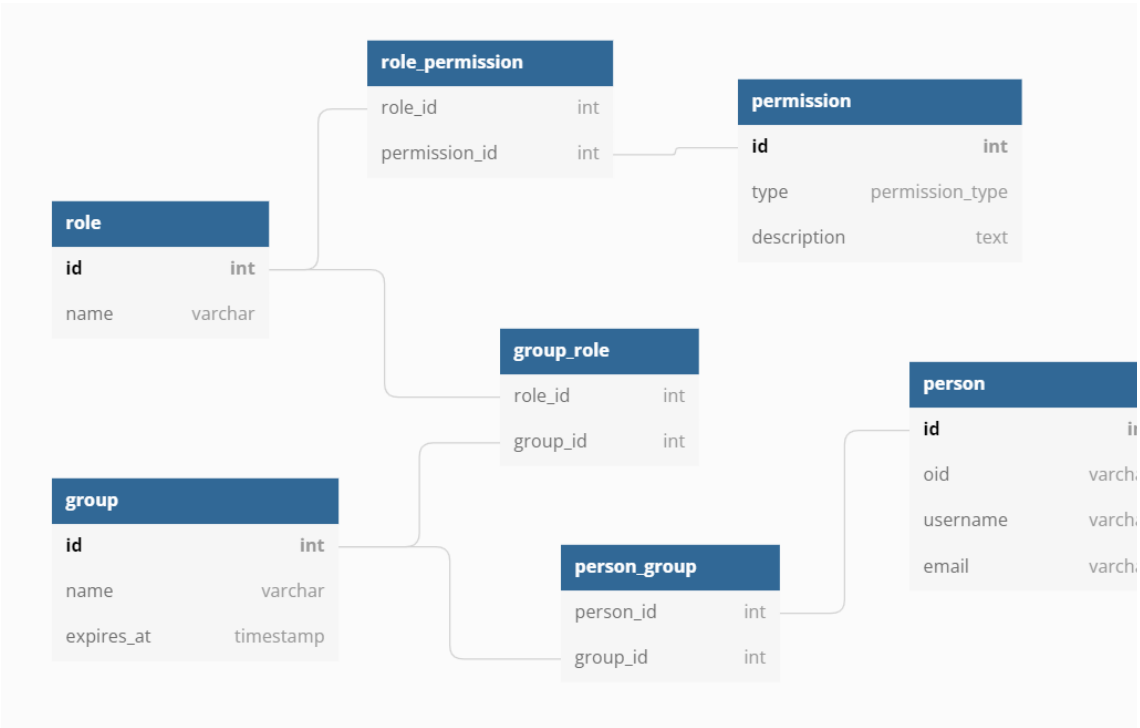


Joonis 13. Siltide haldamise vaade vanal kujundusel.



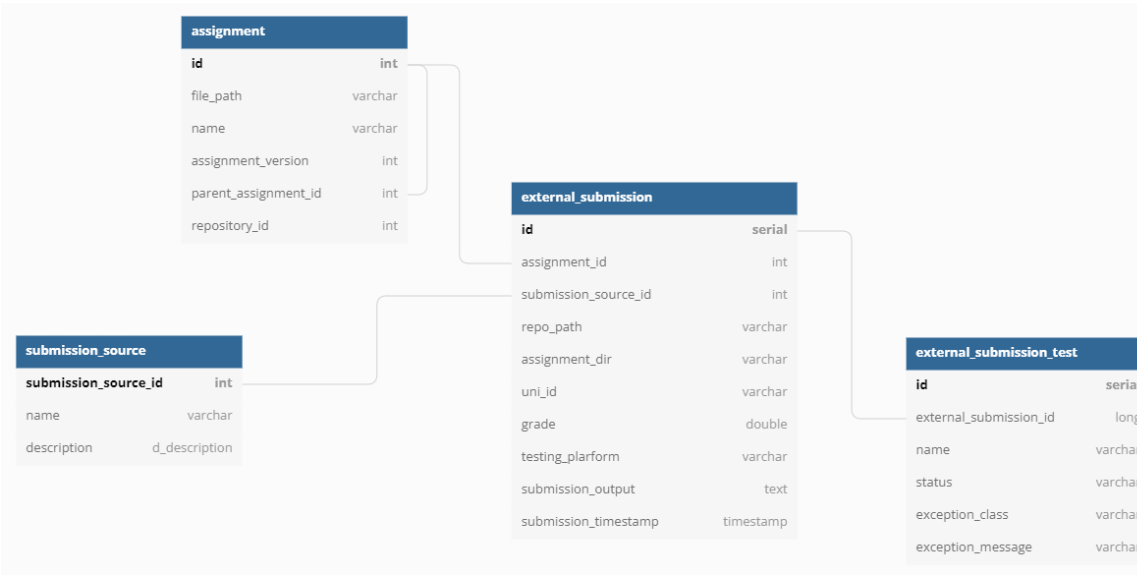
Joonis 14. Siltide haldamise vaade uuel kujundusel.

Lisa 6 – Õiguste haldamise süsteemi andmemudel



Joonis 15. Õiguste haldamise süsteemi andmemudel.

Lisa 7 – Statistika esituste andmemudel



Joonis 16. Statistika esituste andmemudel.