TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Laura Anni 206577IAIB
Marcus Pruks 206539IAIB
Henri Helisma 193285IAIB

AUTOMATING COGNATE AND CREATING A NEW USER INTERFACE

Bachelor's Thesis

Supervisor: Ago Luberg
Ph D

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Laura Anni 206577IAIB
Marcus Pruks 206539IAIB
Henri Helisma 193285IAIB

COGNATE AUTOMATISEERIMINE JA UUE KASUTAJALIIDESE LOOMINE

Bakalaureusetöö

Juhendaja: Ago Luberg
Ph D

Tallinn 2023

Autorideklaratsioon

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Laura Anni, Marcus Pruks, Henri Helisma

22.05.2023

Abstract

As part of this thesis, further developments were made to a system developed in 2022 for managing team projects named Cognate. The system was created in an earlier thesis work called GitLab Project Metadata Analysis Ecosystem with the purpose of giving an overview of different teams functioning in a course. Cognate fetches data from GitLab, stores it and makes it available to the user through a web application user interface. The created system worked, but it had flaws and opportunities for further development.

During this thesis, to identify the bigger flaws and further development wishes, interviews were held with Teachers Assistants using Cognate and discussed with the client. As a result of further development a new user interface was made, some new features were added and most of the major flaws were fixed.

The new version of Cognate consists of a backend and a new frontend. The backend was written in Python using Django framework. The frontend was written in Typescript using Vue3 frontend framework.

The thesis is written in Estonian and is 33 pages long, including 7 chapters, 15 figures and 0 tables.

Annotatsioon

Cognate automatiseerimine ja uue kasutajaliidese loomine

Bakalaureusetöö raames on edasi arendatud 2022. aasta lõputöö, "GitLabi projektide metaandmete analüüsi ökosüsteem"[1], mille käigus loodi Cognate rakendus tiimiprojektide haldamiseks ja ülevaatuseks. Cognate pärib andmeid GitLabi serveritest, hoiustab neid ja teeb need läbi veebirakenduse kasutajale kättesaadavaks. Valminud süsteem töötas, kuid selles leidus vigu, puudujääke ja võimalikke edasiarenduse kohti.

Käesoleva töö raames on suuremate puuduste ja soovide tuvastamiseks teostatud intervjuusid süsteemi kasutanud abiõppejõududega ja vesteldud kliendiga. Edasiarenduse tulemusena on süsteemile valminud uus kasutajaliides, lisatud mitu soovitud funktsionaalsust ja parandatud suurem osa tuvastatud probleemidest.

Süsteemi uus versioon koosneb tagarakenduse edasiarendusest ja uuest kasutajaliidesest. Tagarakendus on kirjutatud Pythonis kasutades Django raamistikku. Esirakendus on kirjutatud Typescriptis kasutades Vue3 raamistikku.

Lõputöö on kirjutatud Eesti keeles ning sisaldab teksti 33 leheküljel, 7 peatükki, 15 joonist, 0 tabelit.

Lühendite ja mõistete sõnastik

API	Application Programming Interface
GitLab	tarkvaraarenduse koostööplatvorm.
IDE	Integreeritud programmeerimiskeskkond.
IntelliSense	*Intelligent code completion* koodi lõpetamise/kirjutamise abi.
JSON	*Javascript Object Notation*. Andmevahetusformaat, mis jaotab andmed võti väärtus paaridesse.
JavaScript	Objektorienteeritud programmeerimiskeel, mida kasutatakse peamiselt veebilehtede kirjutamisel.
PostqreSQL	Relatsiooniline andmebaasi haldamise süsteem
Python3	Üldotstarbeline interpreteeritav programmeerimiskeel.
REST	Tarkvaraarhitektuuri stiil, mis määrab veebirakendusele kindlad reeglid.
runner	GitLabi pideva integratsiooni ülesandeid jooksutav rakendus.
Tagarakendus	Teenus, mis serveerib andmeid kasutajaliidesele.
TypeScript	Microsofti arendatud vabavaraline programmeerimiskeel, mis kompileerub JavaScriptiks.
UX	*User Interface* ehk kasutajakogemus.
VSCode	Microsofti arendatud integreeritud programmeerimiskeskkond, mis toetab muuhulgas JavaScripti.
Vue3/Vue.js	Progrssiivne JavaScripti raamistik, millega luuaks ühelehelisi rakendusi.

Sisukord

Jooniste loetelu

1. Sissejuhatus

Eelneval õppeaastal (2022) valmis lõputöö, "GitLabi projektide metaandmete analüüsi ökosüsteem"[1], mille tulemusena loodi süsteem nimega Cognate. Rakenduse primaarne sihtgrupp on (abi)õppejõud, kes annavad projektiained, mis koosnevad üldiselt tudengite gruppidest, kellel on üks või mitu koodi hoidlat. Sellist tüüpi aine õppejõududel on tihti raske ülevaadet hoida sellest, kas kõik tiimid tegelevad semestri jooksul piisavalt oma projektiga, kas kõik tiimiliikmed panustavad võrdselt projektile ja kas projekti haldamine vastab nõuetele. Selle probleemi leevendamiseks tuleb kasuks Cognate, süsteemi põhiline funktsionaalsus seisneb TalTechi GitLabi projektide grupeerimises, GitLabi statistika visualiseerimises ja projektide hindamise võimekuses.

Kuigi suur osa tööst sai eelnevas lõputöös tehtud, siis tegelikult jäi ka palju tegemata. Selle bakalaureusetöö raames otsustasime arendada edasi Cognatet. Saime varakult kliendiga edasiarenduse plaanid läbi räägitud. Rõhk oli olemasoleva parendamisel ja tudengitele rakenduse kasutatavaks tegemine.

Erinevatel põhjustel otsustasime, et meil oleks vaja luua uus kasutajaliides, mis hoiab ka muude TalTechi toodetega sarnast stiili. Seda lihtsustas töötava tagarakenduse olemasolu, kuna andmete struktuur ja API lõpp-punktid olid juba teada. Selle võrra oli uue kasutajaliidese arendamine suhteliselt hea tempoga ja palju soovitud muudatusi sai jooksvalt rakendada. Arenduse käigus saime kliendilt ja projektiainete abiõppejõududelt korduvalt tagasisidet.

Tagarakenduse pooles jätsime olemasoleva koodi kasutusse, kuid täiendasime ja parandasime seda arenduse käigus tunduvalt, lahendasime jooksvalt selgunud probleeme ning lisasime uusi funktsioone.

Töös esitame rakenduse edasiarendusega seotud taustainfo, muudatuste kirjeldused, juurde arenduste kirjeldused, sihtgrupid, eesmärgid ja konkurendid. Anname detailse ülevaate arendusprotsessist ja selle eri etappidest. Lisaks anname ülevaate rakenduse testimisest, tagasisidest ja tulemustest ning pakume välja, milliseid edasiarendusi saaks rakenduses veel teostada.

2. Taust

Töö eesmärk on edasi arendada lõputöö "GitLabi projektide metaandmete analüüsi ökosüsteem"-i käigus valminud rakendust Cognate. Cognate on rakendus, kus õppejõud saavad luua gruppe (ehk enamasti aineid), gruppidesse lisada projekte ja projektidega siduda üks või mitu GitLabi koodihoidlat, kust leitakse ka projekti liikmed. Lisaks saab gruppidele seadistada tähtpunktid, mille põhjal kogutakse ja analüüsitakse erinevat *meta* infot GitLabist[2], mille alusel kuvatakse lõppkasutajale infograafikat. Analüüsi järgselt saab õppejõud hinnata iga tähtpunkti kohta projekti liikmete panust.

Edasiarenduse üheks eesmärgiks on teha hindamise protsess läbipaistvamaks projekti liikmetele, andes neile ülevaate enda kohta käivast statistikast ja hinnetest. Rakendus on kasutatav abiõppejõudude ja õppejõudude poolt, mistõttu peab saama administraator või õppejõud määrata teavitusi, mida kuvatakse gruppide või projektide loetelus. Kuna rakendus on suunatud TalTechile, siis peab rakendus ka välja nägema nagu TalTechi toode ja inimesed peavad saama sisse logida oma TalTechi kasutajaga.

2.1 Sihtgrupp

TalTechi IT erialadel on mitmeid aineid, milles moodustatakse tiime, kes peavad semestri jooksul läbima etappe ehk tähtpunkte ja hoidma oma tehtud tööd TalTechi GitLabis. Sellest lähtuvalt on rakenduse põhilised kasutajad projekti ainete läbiviijad ja nende abiõppejõud. Cognate eesmärk on lihtsustada õppejõudude tööd, andes neile arusaadaval kujul statistikat tiimide progressi kohta ja andes neile koha, kus nad saavad kergelt eelnevalt seadistatud punktide põhjal hinnata õpilasi.

Varem valminud lõputöös "GitLabi projektide metaandmete analüüsi ökosüsteem" on ka põhjalikult analüüsitud võimalikke Cognate alternatiive ja konkurente[1]. Aasta jooksul ei ole olemasolevad konkurendid toonud enda toodetele sisse suuremaid suunamuudatusi, mis kattuksid Cognatega. Samuti pole turule uusi konkurente juurde tulnud.

3. Nõuded edasiarendusele

Enne arenduse algust analüüsisime olemasolevat süsteemi ja suhtlesime kliendiga, et panna paika nõuded, mida edasi arendatud rakendus peaks täitma.

1. Uus kasutajaliides, millel on parem väljanägemine ja kasutajakogemus
    - Rakenduse stiil peab vastama TalTechi digitaalsele stiiliraamatule[3].
    - Kasutaja jaoks peab kasutusvoog olema loogiline.
    - Kasutaja jaoks peavad joonised ja graafikud olema arusaadavad.
    - Kasutaja jaoks peavad kõik täidetavad väljad ja nende vajadus olema arusaadav.
2. Sisse logimine läbi TalTechi kasutaja
    - Kasutaja peab saama sisse logida läbi olemasoleva TalTechi kasutaja.
    - Õpilane näeb oma TalTechi kasutajaga sisse logides projekte, kus ta on arendaja.
3. Projektide omavaheline võrdlemine
    - Õppejõud peab saama võrrelda tiimide vahelist projektide statistikat.
    - Võrdlusesse peab saama lisada mitu projekti.
    - Võrreldav info peab olema mõõdetav, et võrdlusest oleks kasu.
4. Teavituste haldamine
    - Administraator ja õppejõud peavad saama luua teavitusi.
    - Teavitustele peab saama lisada värvikoodi ja perioodi, millal seda kuvatakse.
    - Peab saama valida, mis vaadetes teavitust kuvatakse.
5. Lihtne automatiseerimine
    - Õppejõud saaks kergemini üles seadistada gruppe.
    - Õppejõul oleks kuvatud tagarakenduse poolt arvutatud soovituslik punktisumma.

4. Realisatsioon

Arenduse realisatsioon algas olemas oleva analüüsiga, millele järgnes nõuete analüüs ja tehnoloogiate valik.

Esimene versioon Cognatest koosnes eraldi kasutajaliidesest ja tagarakendusest. Kasutajaliidese jaoks oli kasutatud JavaScripti[1] Vue2[2] raamistikku ja tagarakenduse jaoks oli kasutatud Pythoni Django[3] raamistikku. Lisaks oli andmebaasiks kasutatud PostqreSQL[4].

Eelnevat koodibaasi analüüsides arutasime, kas oleks mõistlik teha terve projekt uuesti teiste tehnoloogiatega. Varakult otsustasime kasutajaliidese uuesti tegemise poolt. Selleks oli mitmeid põhjuseid. Nende hulgas:

1. Kasutajaliides oli loodud tehnoloogiale Vue2. Vue2 oli 2022 juulist alates nn LTS (*long-term support* ehk pikaajaline tugi) olekus ja 2023 aastaks lõpuks oleks tehnoloogia jõudnud EOL (*end of life* ehk elu lõpp)[4] seisu. Sellest lähtuvalt oleks juba ammu enne kasutajaliidese arendust pidanud olema teada, et Vue2 saab lähiajal aegunuks.
2. Eelnevast punktist lähtuvalt tähendab see ka seda, et kõik Vue2 raamistikku jaoks loodud teegid nagu Vuex[5] on samamoodi aegunud ja ei saa enam uuendusi.
3. Kliendiga arutades tekkis plaan teha kõik visuaalid TalTechi stiiliraamatu peale ümber, sest rakendus on põhiliselt suunatud TalTechi töötajatele. Kuna vana kasutajaliides oli ehitatud CSS raamistik Bootstrap[5] peale ja enamus komponente polnud tehtud taaskasutatavaks, siis terve kujunduse ümber tegemine oleks osutunud ajakulukamaks kui kujunduse uuesti tegemine.
4. Uuesti tegemine annab võimaluse luua parema struktuuriga koodibaasi ja kasutusele võtta abistavad tehnoloogiad nagu programmeerimiskeel TypeScript[6], mis kompileerub JavaScriptiks, ja koodi analüüsi tööriist ESlint[7].

Välja toodud punktide põhjal saaks argumenteerida, et neid muudatusi oleks võimalik ka teha olemas oleva koodibaasi peal, aga kuna sisuliselt iga fail oleks saanud ümber kirjutatud

[1] JavaScript `https://www.javascript.com/about`
[2] Vue2 `https://v2.vuejs.org/v2/guide/`
[3] Django `https://www.djangoproject.com/foundation/`
[4] PostqreSQL `https://www.postgresql.org/about/`
[5] Bootstrap `https://getbootstrap.com/`
[6] TypeScript `https://www.typescriptlang.org/`
[7] ESlint `https://eslint.org/`

ja iga teek oleks saanud asendatud (nt Pinia asendab Vuexi), siis polnud sellel erilist mõtet, sest üldkokkuvõttes oleks juba vana koodibaase Vue2 pealt Vue3 peale migreerimine olnud väga mahukas ettevõtmine enne kui muid muudatusi oleks saanud teha.

Lisaks mõtlesime ka asendada tagarakendus hoopis Java[8] Spring Boot[9] lahendusega, aga kuna uue kasutajaliidese arendus jõudis kõvasti ette ja selgus, et olemas oleval koodi ümberkirjutamine on ajakulukam kui algul oodatud, siis otsustasime, et lisanduv ajakulu oleks halb meie eesmärkide jaoks.

4.1 Tehnoloogilised valikud

Tehnoloogiliste valikute tegemisel lähtusime sellest, et soovisime luua jätkusuutliku tarkvara, mida oleks ka järgnevatel edasiarendajatel kergem täiendada. Selle tõttu on kõik valitud tehnoloogiad aktiivses arenduses, hea mainega ja põhjaliku dokumentatsiooniga.

4.1.1 Vue3

Uue kasutajaliidese jaoks oli mitu võimalikku JavasScripti raamistikku: React.js[10], Next.js[11] (React.js peale ehitatud raamistik), Vue3[12] ja Nuxt.js[13] (Vue peale ehitatud raamistik). React.js ja Next.js tundusid nagu hea valik kuna ühel tiimi liikmel on 2 aastat eelnevat töökogemust Reacti kirjutamisega, aga kuna Vue ja Reacti süntaksid on väga erinevad, siis ei oleks saanud vana lahendust kasutada viitena ja uue lahenduse tegemine oleks selle võrra pikkemaks läinud. Järgmisena tuli valida Vue3 ja Nuxt.js vahel. Nuxt.js suurim eelis Vue3 ees oleks, et saaks kasutada SSR (*server side rendering*), mis tähendaks, et kasutaja masin peaks tegema vähem tööd lehe kuvamiseks. Nuxt.js-le jäi saatuslikuks fakt, et kuna Vue3 oli suhteliselt uus, siis polnud ka Nuxti implementatsioon täielikult ümber kolinud. Esimene versioon Vue3 Nuxtist ehk Nuxt3-st tuli alles välja 2 kuud peale arenduse algust.[6]

4.1.2 Vue3 seotud valikud

Vues on mitu erinevat viisi koodi kirjutamisele. Ametlikult on põhilised 2 varianti Options Api ja Composition Api[7]. Options Api oli originaalne viis komponentide kirjutamiseks ja sisuliselt võttis sellise lähenemise, et igale komponendile sai edastada objekti, kus oli

[8]Java `https://www.java.com/en/`
[9]Spring Boot `https://spring.io/`
[10]React.js `https://react.dev/`
[11]Next.js `https://nextjs.org/`
[12]Vue3 `https://vuejs.org/`
[13]Nuxt.js `https://nuxtjs.org/`

defineeritud komponendi jaoks kasutatavad andmed ja funktsioonid. Seda oli kasutatud vanas rakenduses.

Uuem viis Vue komponentide kirjutamiseks on Composition Api. Tegemist on lähenemisega, mis on sarnasem tavalisele JavaScriptile, kus saab defineerida funktsioonid ja muutujad ja siis saab neid kasutada komponendi skoobis.

Uue kasutajaliidese jaoks otsustasime kasutada Composition Api kuna:

1. Seda on kergem organiseerida.
2. Seda on kergem lugeda.
3. Seda on kergem õppida.
4. Sellel on parem tugi TypeScript jaoks.

Koodi stiili ja tavade kvaliteedi tõstmiseks tuli ka paigaldada ESLint. ESlint on koodi analüüsimise tööriist, mis töötab tavaliselt koos koodiredaktoriga, et välja tuua ja parandada probleeme koodibaasis. Eelmises koodibaasi oli küll tehniliselt ESlint olemas, aga ei olnud õigesti seadistatud või ei töötanud mingil muul teadmatul põhjusel kuna koodibaasis leidus palju probleeme, mille peale tavaliselt hoiatused tuleks. Nt.

1. Paljud muutumatud muutujad olid deklareeritud kasutades *let* või *var* märksõna kuigi tavaliselt saaks *let* asendada *const* märksõnaga ja *var* kasutamine pole üldse soovitatud [8].
2. Enamasti oli kasutatud loetelude läbimiseks nn *for...in* meetodit, mis on keskmisel halvema jõudlusega kui *forEach* või tavaline *for loop*[9].
3. Leidus sõltuvuste tsükleid, kus kaks faili olid üksteisest sõltuvad.

Lisaks koodi kvaliteedile sai ESlint ka seadistatud koodi vormistamiseks tänu teegi laiendusele eslint-plugin-vue[14]. See tagab selle, et läbi koodibaasi hoitakse ühtset joont kui arendavad erinevad inimesed. Tihti kasutatakse vormistamiseks ka teisi teeke nagu Prettier[15], aga kuna ESlint töötas väga hästi ja seda on kergem erinevates arvutites seadistada[10], siis otsustasime, et sellest piisab.

Muu hulgas andis rakenduse uuesti kirjutamine võimaluse luua koodibaas, kus on kergem orienteeruda ka järgnevatel edasi arendajatel. Kuna tiimil oli raske navigeerida vana kasutajaliidese koodibaasis, sest failid olid suuresti paigutatud kahte kausta ja nimetamissüsteem oli ebaühtlane (Lisa 1), siis teadsime varakult, et peaks endale mingid kindlamad eeskirjad

[14]eslint-plugin-vue `https://eslint.vuejs.org/`
[15]Prettier `https://prettier.io/`

seadma. Eelnevalt oli kasutatud stiili, mida saaks nimetad lamedaks struktuuriks [11]. Lame struktuur võib olla hea väiksemate projektide haldamiseks, aga tavaliselt eeldab see ka, et komponentide nimetused viitavad sellele, kus neid kasutatakse. Nt, kui meil on vaate komponent Project.vue ja komponent ProjectDeveloper.vue, siis võib eeldada, et teine on kasutuses esimeses, aga tegelikult polnud rakendus nii struktureeritud.

Enda eesmärkide täitmiseks tundusid kõige paremad vueschool.io soovitused[12]. Lühidalt, jälgime seda põhimõtet, et igale vaate komponendile on oma kaust, kuhu grupeeritakse kõik komponendid, mida kasutatakse ainult selle vaate jaoks. Taaskasutatavad komponendid paiknevad *ui* kaustas ja lehe põhi struktuur asub *layout* kaustas (Lisa 2).

4.1.3 Andmete säilitamine ja haldamine kasutajaliideses

Modernsed kasutajaliidesed on väga keerulised ja dünaamilised ning vajavad tihti andmete säilitamiseks ja komponentide vahel info jagamiseks oleku haldamise võimekust. Kõige levinum näide infost, mida on vaja säilitada terve lehe peal, on kasutaja info, et leht teaks, kas praegune kasutaja on juba sisse loginud ja kui on, siis mis kasutajaga seotud info ta peaks erinevatele päringutele kaasa andma. Vanas kasutajaliideses oli kasutatud selle jaoks teeki nimega Vuex[16].

Vuex oli pikka aega Vue ametlik soovitatud oleku haldamise teek, mis tegi andmete salvestamise ja lugemise kergeks, aga selle edasi arendus on nüüdseks lõppenud. Arenduse lõpp tulenes sellest, et esile kerkis teine teek nimega Pinia[17], mis võttis Vuex koha endale Vue ametliku soovitusena. Vuex enda kodulehel on kirjas, et võimalusel oleks parem olemas olevad koodibaasid viia Pinia peale ja uuel koodibaasil kasutada Piniat mitte Vuex[5].

Muidu on mõlema teegi API väga sarnane, aga Pinia on sarnasem modernsele Vue3 API-le kui Vuex. Kuna Pinia on ametlikult soovitatud ja Vuex ei arendata enam edasi, siis oli see lõpus õige valik millega edasi liikuda.

Lisaks kasutaja andmete salvestamisele, kasutame Pinia võimekust ka, et salvestada päringute tulemusi, mis muutuvad harva, aga mida kutsutakse tihti välja. Andmete säilitamise pikkus oleneb eelkõige sellest kui tihti andmed võivad muutuda ehk see võib erineva iga päringu jaoks. Siiski pole mõtet neid korraga säilitada rohkem kui 2 tundi. Eeldatavasti kasutatakse rakendust kõige rohkem koolitunni jooksul (TalTechis on tavaliselt koolitunni pikkuseks 1.5 tundi) ja umbes pool tundi peale seda.

[16]Vuex `https://vuex.vuejs.org/`
[17]Pinia `https://pinia.vuejs.org/`

4.1.4 TypeScript

Nagu varasemate valikute põhjal on välja tulnud, siis otsustasime ka kasutada JavaScripti asemel TypeScripti. See valik tuli suuresti sellest, et seda tüüpi projekti arendatakse kindalt ka tulevikus edasi erinevate inimeste poolt. Selle koha pealt annab TypeScript eelised nagu:

1. Parem IntelliSense[13]. Mis annab koodile peale hõljudes info selle kohta, mis parameetreid miski kasutab, mis on oodatud sisendi ja väljundi kuju, mis sorti infot tagastatakse tagarakendusest jne.
2. Aitab kergemini vältida vigu andmete töötlemisel ja üldisel koodi kirjutamisel.
3. Tagab uutele arendajatele parema ülevaate sellest, kuidas koodibaas töötab.

4.1.5 Less

Kuna uus rakendus peab olema TalTech stiilis ja TalTechil pole ametlikku komponentide teeki Vue jaoks, siis kõige kergem lähenemine on stiil ise jäljendada TalTechi kujundusvaate ja TalTech *Digital Styleguide*[18], lahendusi. Ise komponentide stiilimine on tehtav läbi CSS-i või läbi CSS eeltöötlejate nagu Less[19] või Sass[20]. Kogemuse järgi võib öelda, et terve lehe stiilimine võib palju CSS-i tekitada ja selle tõttu otsustasime võtta kasutusse eeltöötleja Less. Less annab arendusele järgnevad eelised.

1. Laseb pesitseda selektoreid üksteise sees. Tänu sellele saab kirjutada kordades paremini struktureeritud ja loetavamat stiili.
2. Laseb deklareerida ja kasutada muutjaid, mis on kasulikud globaalse stiili hoidmiseks.
3. Laseb importida Less faile teiste Less failide sees.

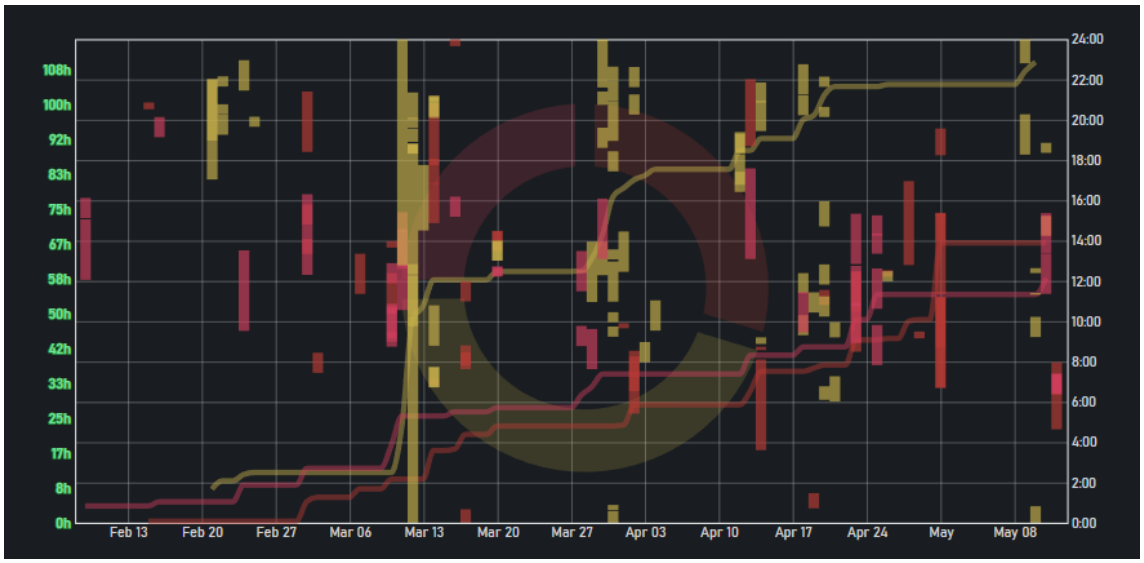
4.1.6 Apache ECharts

Üks suurimaid probleeme vanas kasutajaliideses oli andmete visualiseerimine. Jooniste loomiseks oli kasutatud teeki D3.js[21]. D3.js on väga madalal tasemel seadistatav ja pakub arendajale võimalusele luua just sellise joonise nagu ta soovib, aga sellest tuleneb ka palju probleeme. Kuna D3.js eesmärk pole valmis lahendusi anda, siis võib jooniste loomiseks

[18]TalTech Digital Styleguide `https://portal-dev.ttu.ee/styleguide/?path=/story/docs-intro--default`
[19]Less `https://lesscss.org/sa`
[20]Sass `https://sass-lang.com/`
[21]D3.js `https://d3js.org/`

minna mitmeid kümneid tunde. Eelnevas kasutajaliideses oli selliseid probleeme nagu.

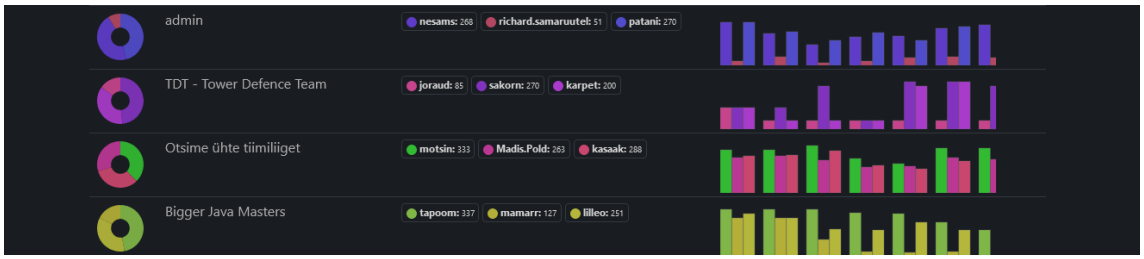
1. Joonistel puudusid nimed. (Joonis 1)
2. Joonistel puudusid ühikud ja info selle kohta mida mõni tulp või fraktsioon näitab. (Joonis 1)
3. Mitu erinevat joonist oli ühe kasti sisse mahutatud kuigi nad näitasid täitsa erinevat infot. (Joonis 1)



Joonis 1. Probleemne joonis vanas kasutajaliideses.

Uue kasutajaliidese jooniste jaoks otsustasime kasutada teeki Apache ECharts[22]. Valik sai tehtud kuna üks tiimi liikmetest oli kokku puutunud selle kasutamisega TalTechi aines Kasutajaliidesed (ITI0209)[14]. Tegemist on väga lihtsasti kasutatava teegiga, kuhu peab andma ette õigel kujul andmed ja joonise seadistused. Läbi seadistuste saab määrata nt, et näitaks info legendi (kus saab ka tulpasid või jooni vajadusel välja lülitada), info punktidele peale hõljudes näidatakse vihjet parema ülevaate jaoks ja palju muud. Enne mainitud aines sai ka ülevaate andmete visualiseerimise headest tavadest. Näiteks jooniste näitamine tumedal taustal[15] ja sektor- ning sõõrikdiagrammid[16] pole hea loetavuse jaoks (Joonis 2). Igal joonisel peab olema nimi ja telgedel peab olema arusaadav mis infot näidatakse (Joonis 3). Isegi peale hõljudes ei kuvanud need joonised infot selle kohta, mida nad kujutavad. Inimesele, kes ei kasuta seda rakendust iga päevaselt, oleks väga raske aru saada nendest. Uues kasutajaliideses on suur rõhk info kuvamisel arusaadaval viisil.

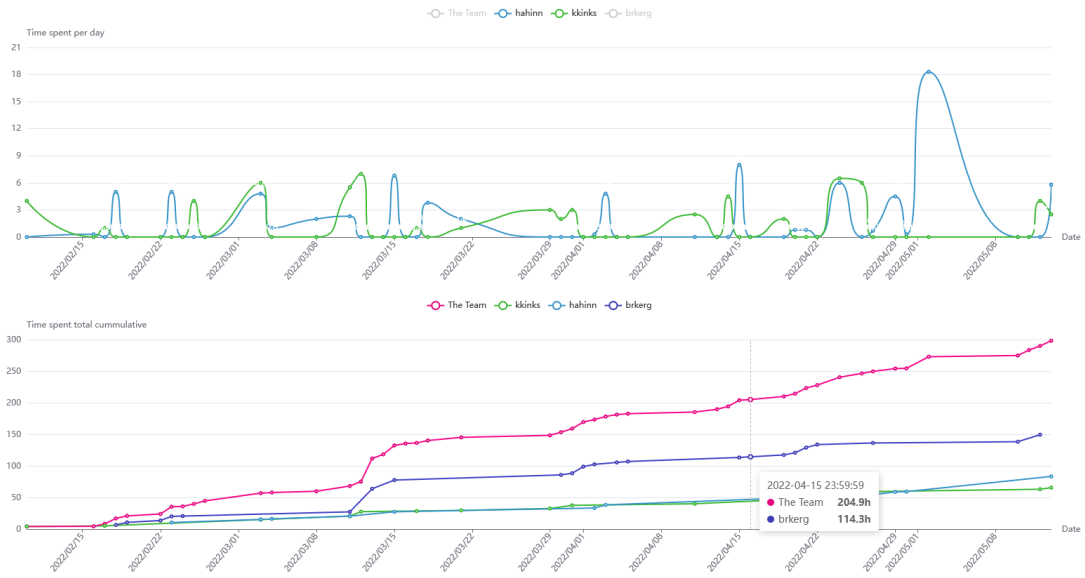
[22]Apache ECharts `https://echarts.apache.org/`



Joonis 2. Joonised ilma kriitilise infota tumedal taustal.

Joonis 3. Tulpdiagramm koos telgede nimetustega ja vihjega.

Legend ja vihjed annavad parema ülevaate joonisel toimuvast. Apache ECharts võimaldab ka välja lülita jooni kui on soovi ainult osade tiimi liikmete kohta infot vaadata (Joonis 4).



Joonis 4. Kaks joondiagrammi koos legendidega ja vihjega.

Info mida varem kuvati sektordiagrammidena kuvatakse nüüd tulpdiagrammidena, et vältida juhtumeid kus kahel isikul on määratud taoline värv ja neid on raske eristada (Joonis 5).

Joonis 5. Tulpdiagramm tiimi punktide jaotuse kohta.

4.1.7 i18n

Kuna suur osa uue kasutajaliidese arendusest pöörles ümber põhimõtte, et lihtsustada arendust endale ja ka järgmistele arendajatele, siis tekkis küsimus, et mida teha parema lokaliseerimise jaoks. Lokaliseerimist tuntakse arenduse kontekstis tavaliselt termini i18n all, mis on lühend ingliskeelsele sõnale *internationalization*, kus 18 on esimese i ja viimase n vahel olevate tähtede arv. Kui rääkida tõlkimisest kasutades mõnda i18n tehnoloogiat, siis see tähendab, et terve lehe saab ära tõlkida ühest keelest teise ilma, et peab koodibaasis kõik kohad manuaalselt ära muutma. Lisaks lihtsustab see mitme keele hoidmist lehel[17]. Tavaliselt i18n teegid töötavad nii, et kuskil vahendite kaustas saab määrata JSON formaadis failid, kus on erinevat võtmed ja tõlked ning koodis kasutatakse mõnda abifunktsiooni, et kutsuda välja võtmele vastav tõlge olenevalt selles, mis keel talle on antud hetkel määratud. Muuhulgas tähendab see ka seda, et kui lehel on palju taaskasutatavaid sõnu, siis nende muutmiseks piisab ühe rea muutmisest.

Antud projekti raames otsustasime kasutada vue-i18n teeki, et implementeerida tõlkesüsteem. Kogemuse järgi saab öelda, et erinevate raamistike erinevate i18n teegid töötavad võrdlemisi sarnaselt. Sellest lähtuvalt ei olnud palju tegureid, mis valikut suunaksid, aga on väärt märkimist, et tegemist on teegiga, mis valimise hetkel oli väga populaarne koos 1 miljoni alla laadimisega nädalas. Selle tõttu tundus tegemist olevat usaldusväärse valikuga.

4.1.8 Microsoft Graph API

Üks aspekt ühendamaks Cognatet teiste Tallinna Tehnikaülikoolis kasutusel olevate keskkondadega on võimalus sisselogimiseks kasutada kooli emaili. Selle võimaluse rakendamiseks on vaja saada informatsiooni kasutajate Microsofti kontode kohta. See on seetõttu, et kooli poolt loodud emailid on Microsofti kontod. Saamaks ligipääsu vaja minevale informatsioonile pidi kasutama Microsoft Graph API-d[23][18]. Microsoft Graph API on REST API, mille kaudu saab ligi Microsofti pilveteenuste ressurssidele. Ligipääsu saamiseks registreeriti rakendus Microsofti Azure Portalis[24].

4.2 Arendusprotsess

Enda arenduskeskkonnas tarkvara käivitamiseks on meie poolt soovitatud vahendid, mida me ise kasutasime, kuigi need saab asendada ka vabalt valitud endale sobilikega alternatividega.

[23]Microsoft Graph API `https://learn.microsoft.com/en-us/graph/`
[24]Microsoft Azure Portal `https://portal.azure.com/`

Lisaks on vaja paigaldada enda arenduskeskkonda erinevaid programme nii kasutajaliidese kui ka tagarakenduse jaoks.

4.2.1 Arendusvahendid

Kuna kasutajaliidese kirjutamisel on palju alternatiivseid koodiredaktoreid mida kasutada saab, siis rakendust saavad kõik arendada endale sobivas keskkonnas, aga antud töö raames panime põhi rõhu VSCode (Visual Studio Code)[25] kasutamisele. VSCode on vabavaraline koodiredaktor loodud tarkvara ettevõtte Microsoft poolt. See on laialt levinud ning omab sellised atribuute nagu:

- On väike ja võtab vähe arvuti ressursse.
- Omab palju inimeste poolt loodud laiendusi.
- On tasuta ja vabavara.
- Töötab erinevate operatsioonisüsteemide peal.

Eelmises peatükis sai mainitud, et kasutame Vue3, TypeScript ja Eslint. Ka nende jaoks leidub VSCode-l erinevaid laiendusi, mis aitavad teha arendusprotsessi tunduvalt mugavamaks. Selleks et oleks ka erinevate inimeste vahel kergem säilitada ühtsust arendusvahendite vahel, siis VSCode jaoks saame me luua projekti JSON faili, kus on märgitud ära soovitatud laiendused (Joonis 6). See tähendab seda, et kui keegi teeb projekti lahti esimest korda, siis talle soovitatakse kohe need laiendused, et ta saaks oma keskkonna ära seadistada.

Joonis 6. Näide failist, kus on määratud soovitatud laiendused.

Lisaks laienduste soovitustele lisasime ka projekti README.md faili muud vajalikud sammud, et oma keskkond püsti saada.

Kuigi VSCode toetab mitmeid keeli kaasaarvatud Python3, siis tiimisiseselt kasutame

[25]VSCode `https://code.visualstudio.com/`

tagarakenduse arenduse jaoks rakendust PyCharm[26], mis loodi tarkvara ettevõtte JetBrains poolt. PyCharm on spetsiifiliselt loodud Python keele jaoks ja selle tõttu on seda mugav kasutada ja seadistada.

Andmebaasi haldamiseks kasutame rakendust pgAdmin 4[27], mis on loodud PostgreSQL andmebaaside jaoks. Läbi rakenduse on kerge ühendust saada andmebaasiga, mis asub kuskil serveris ja vajadusel andmeid analüüsida ja muuta kiiremini kui üle käsurea.

4.2.2 Arenduskeskkonna seadistamine (tagarakendus)

Tagarakendus käivitamiseks on vaja arvutis seadistada järgnevad tehnoloogiad.

1. Python versioon 3.11
2. Django versioon 4.0 (Võimalik paigaldada läbi koodiredaktori)
3. Docker (windowsil Docker Desktop)[28]
4. Postgres dockerile versioon 14[29]

Kui need tehnoloogiad on paigaldatud, siis tuleb projekti kaust avada enda eelistatud koodiredaktoris või käsureal ja käivitada järgmise käsuga:

```
docker-compose up
```

4.2.3 Arenduskeskkonna seadistamine (kasutajaliides)

Kasutajaliidese käivitamiseks on vaja arvutis seadistada esialgu järgnevad tehnoloogiad.

1. Node.js versioon 18[30]
2. Vue CLI (vajab, et Node.js oleks paigaldatud)[31]

Kui need tehnoloogiad on paigaldatud, siis tuleb projekti kaust avada enda eelistatud koodiredaktoris või käsureal.

Kui projekt on avatud läbi VSCode, siis rakendus peaks teavitama, et on soovituslik

[26]PyCharm `https://www.jetbrains.com/pycharm/`
[27]pgAdmin 4 `https://www.pgadmin.org/faq/`
[28]docker `https://www.docker.com/`
[29]Postgres for docker `https://hub.docker.com/_/postgres`
[30]Node.js 18 `https://nodejs.org/en/download`
[31]Vue CLI `https://cli.vuejs.org/`

paigaldada järgnevad laiendused: ESlint, Volar, TypeScript Vue Plugin. Kui laiendused on paigutatud, siis peab käsureal jooksutama järgnevat käsku, et tõmmata kasutajaliidese ehitamiseks vajalikud teegid.

```
npm install
```

Et näha rakendus oma lõppkujul, tuleb algul ehitada projekti staatilised failid ja siis rakendus käivitada.

```
npm run build
npm run start
```

Arenduse jaoks tuleb ainult kasutajaliidese server, mis jälgib koodi muudatusi ja uuendab väljundid, käivitada järgneva käsuga.

```
npm run serve
```

Lisaks on projekti README.md[19] failis välja toodud täiendavad sammud, mida peab täitma, et automaatne vormindamine töötaks.

4.2.4 Pidev integratsioon (*CI/CD*)

Varasemast oli projekti arendamiseks kasutusel testimise server ja keskkonna kasutamiseks *live* server. Uue kasutajaliidese kasutusele võtmiseks oli tarvis mõlemas serveris peatada jooksvad kasutajaliidese jooksutajad ja asendada need uue kasutajaliidese jooksutajatega. Protsessi õppimiseks tegime seda esmalt test serveris ja kui seal kõik töötas, sai ka *live* serveris jooksev kasutajaliides ümber vahetatud.

Peale selle sai test serveris implementeeritud järgmised muudatused:

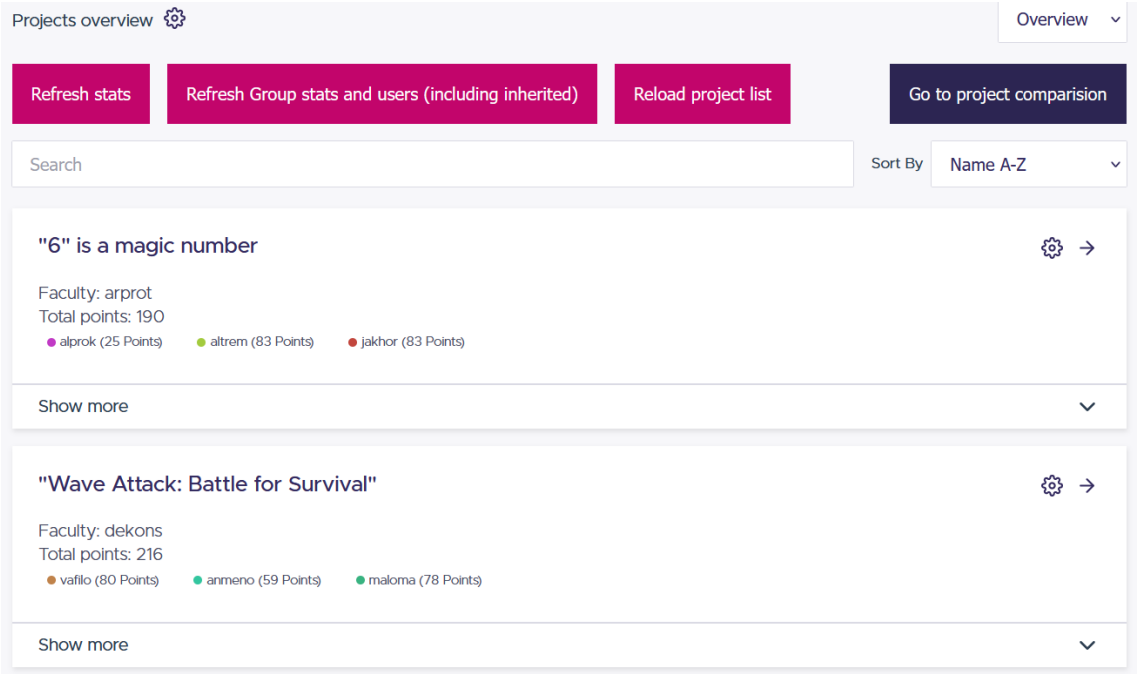
1. Lisasime serverile domeeninime kasutamise.
2. Lisasime serverile koodi aegunud dockeri failide automaatseks kustutamiseks - viimane aitas leevendada serveri mahu otsa saamise probleemi, mis eksisteerib tänu test serveri väiksemale mahule.

4.3 Juurdearendused

Lähtuvalt projekti nõuetest valmisid Cognate rakenduse jaoks ka täiesti uued funktsionaalsused, mida ei leidunud eelnevalt.

4.3.1 Projektide võrdlemise vaade

Uue kasutajaliidese arendusel selgus, et olemasolev grupi vaade ehk ainuke vaade, mis edastas mitme projekti infot korraga ei andnud head pilti projektide võrdlemiseks (Joonis 7). See kuvab iga projekti kohta ainult ühe graafiku ja selle nägemiseks tuleb vastav projekt lahti klõpsata.



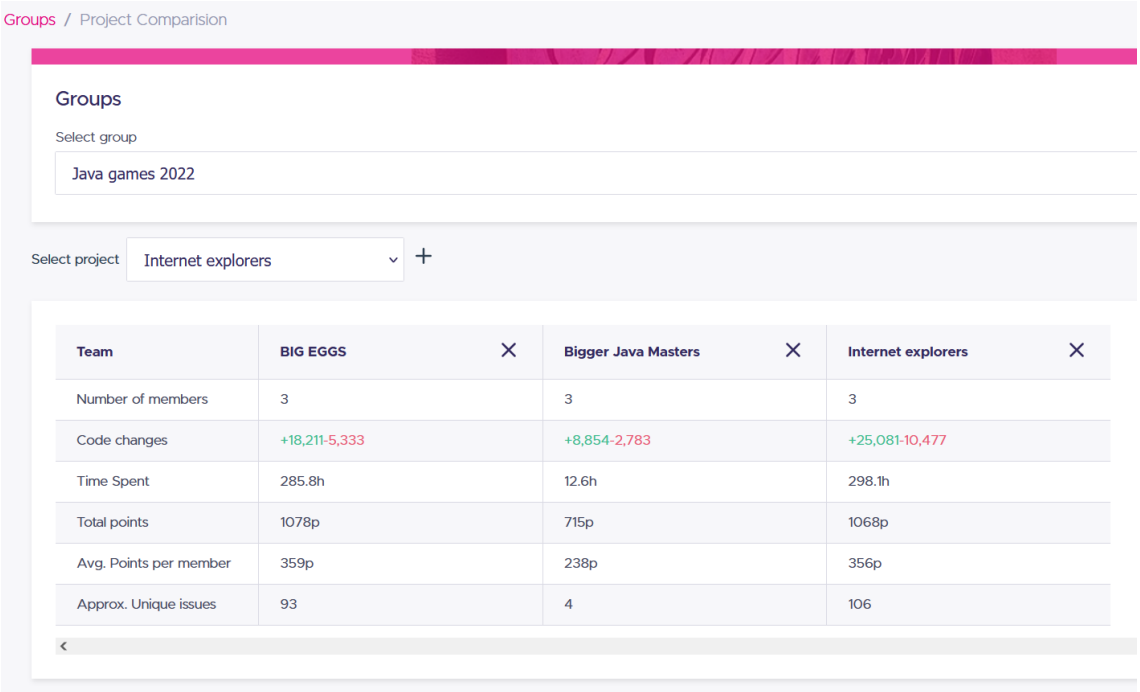
Joonis 7. Erinevate projektide info grupivaates.

Kuna antud vaade erinevate projektide erinevate andmete võrdlemiseks tõhus ei olnud otsustasime lisada uue vaate - projektide võrdlemise vaate. Uus vaade pidi võimaldama tabelis ja graafikutel kuvada üheaegselt mitme projekti andmed. Samuti pidi uus vaade andma kasutajale suurema kontrolli kuvatavate projektide valiku üle.

Seoses asjaoluga, et uus vaade kujutas endas puhtalt juba kättesaadava info alternatiivset kuvamist, osutus selle arendus puhtalt kasutajaliidese edasiarenduseks.

Valminud projektide võrdlemise vaates (Joonis 8) tuleb kasutajal kõigepealt valida temale kättesaadavatest gruppidest üks grupp, mille projekte ta hakkab võrdlema. Seejärel küsib

kasutajaliides tagarakenduselt vastava grupi info ning kasutajale avaneb võimalus lisada vaatesse erinevaid projekte ja neid ka vaatest eemaldada. Lisatud projektide kohta kuvatakse kõigepealt tabel erinevate arvuliste andmete võrdlemiseks. Sellele järgneb graaf, mis kuvab infot tähtpunktidel saadud punktisummade kohta. Kolmandana on vaates ajakulu võrdlus, millel on ka võimalik kuvatava aja perioodi muuta. Ajakulu võrdluses on kuvatud 2 graafikut - aeg kulutatud päeva kohta ja aeg kulutatud kumulatiivselt.

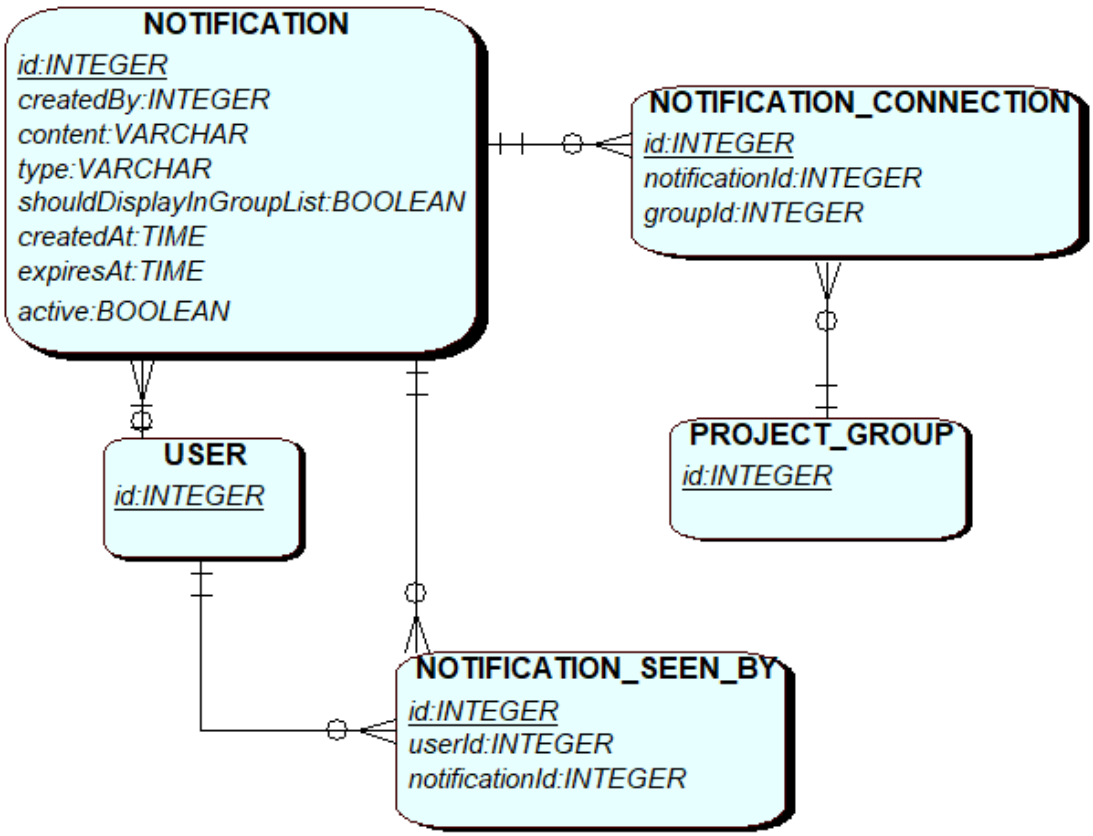


Joonis 8. Ekraanitõmmis uuest Projektide võrdlemise vaatest - tõmmisel on näha grupivalik, projektide lisamine ja esimene kuvatav tabel.

4.3.2 Teavituste süsteem

Arenduse käiku plaanides avaldas klient soovi uuele funktsionaalsusele, mis võimaldaks keskkonda kasutavatel õppejõududel ja abiõppejõududel jätta teavitusi teistele kasutajatele.

Kuna varasemalt selline süsteem puudus, siis tuli arendust alustada tagarakenduses. Kõigepealt lisasime andmebaasi tabelid, mis võimaldasid hoida infot teavituse kohta, sellega seotud gruppide kohta ja nimekirja kasutajatest, kes on teavitust näinud (Joonis 9).



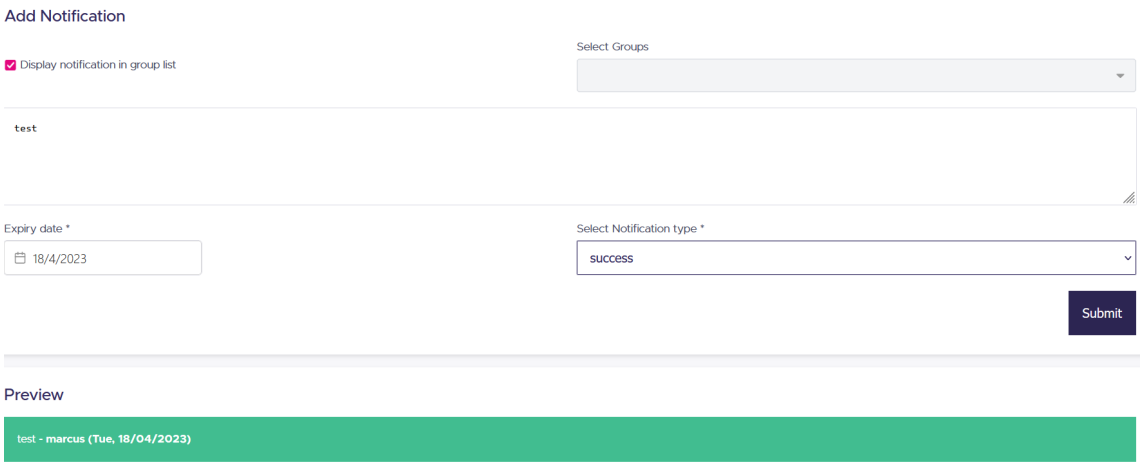
Joonis 9. ERD Skeem teavituse tabeli, sellega seotud uute tabelite ja olemasolevate tabelite ühendustega.

Peale tabelite loomist ja vastavate migratsioonide genereerimist lisasime tagarakendusele teavitusi saatvad ja sissetuleva info põhjal teavitusi loovad otspunktid. Sellega oli tagarakenduse teavituste süsteemi esimene versioon valmis ning asusime teavituste vaadet ja vastavate teavituste kuvamist lisama kasutajaliidesesse.

Kasutajaliideses kuvatakse teavitusi kahes kohas: gruppide nimekirjas kuvatakse lehe halduri poolt loodud teavitusi, nende puhul on atribuut *shouldDisplayInGroupList* tõene ja neid luues ei looda *NotificationConnection* olemeid; grupi vaates enne projektide nimekirja kuvatakse selle grupi haldaja poolt loodud ja selle grupiga ära seotud teavitusi, nende puhul on atribuut *shouldDisplayInGroupList* väär ja nende loomisel luuakse teavituse ja grupi omavaheliseks sidumiseks *NotificationConnection* olem. Esimesena saigi implementeeritud teavituste nimekirja kuvamine antud vaadetes.

Seejärel oli vaja lisada kasutajaliidesele vorm, mis võimaldaks uute teavituste loomist. Selleks sai arendatu teavituste vaade. Vaade koosneb uue teavituse lisamise vormist, loodava teavituse eelvaatest ja kasutaja poolt loodud teavituste nimekirjast (Joonis 10).

Teavituse lisamise vormis tuleb kõigepealt valida, kus teavitust kuvama hakatakse, seejärel sisestada teavituse tekst ja lõpuks valida teavituse arhiveerimise kuupäev ja teavituse tüüp. See vorm sai teisena välja arendatud, kuid puudus viis kontrollimaks, kas kasutaja on lehe haldur.



Joonis 10. Ekraanitõmmis teavituse lisamise vaatest ja loodava teavituse eelvaatest.

Viimaks lisasime vaatele nimekirja kasutaja poolt loodud teavitustest ja nende juurde võimaluse teavitus arhiveerida (Joonis 11). Teavituste nimekirjas kuvatakse ka nimekiri kasutajatest, kes on seda teavitust näinud.

Algselt implementeerisime teavituste süsteemi koos teavituste täieliku kustutamisega, aga lõpuks otsustasime selle ümber vahetada teavituste arhiveerimisega, kuna võib osutuda vajalikuks varem eksisteerinud teavitusi vaadata. Selleks lisasime teavitusele *active* tõeväärtuse atribuudi. Nüüdsest muudetakse teavituse see atribuut läbi kasutajaliidese kustutamise vääraks ja teavitus ja sellega seotud olemid jäävad andmebaasi alles.

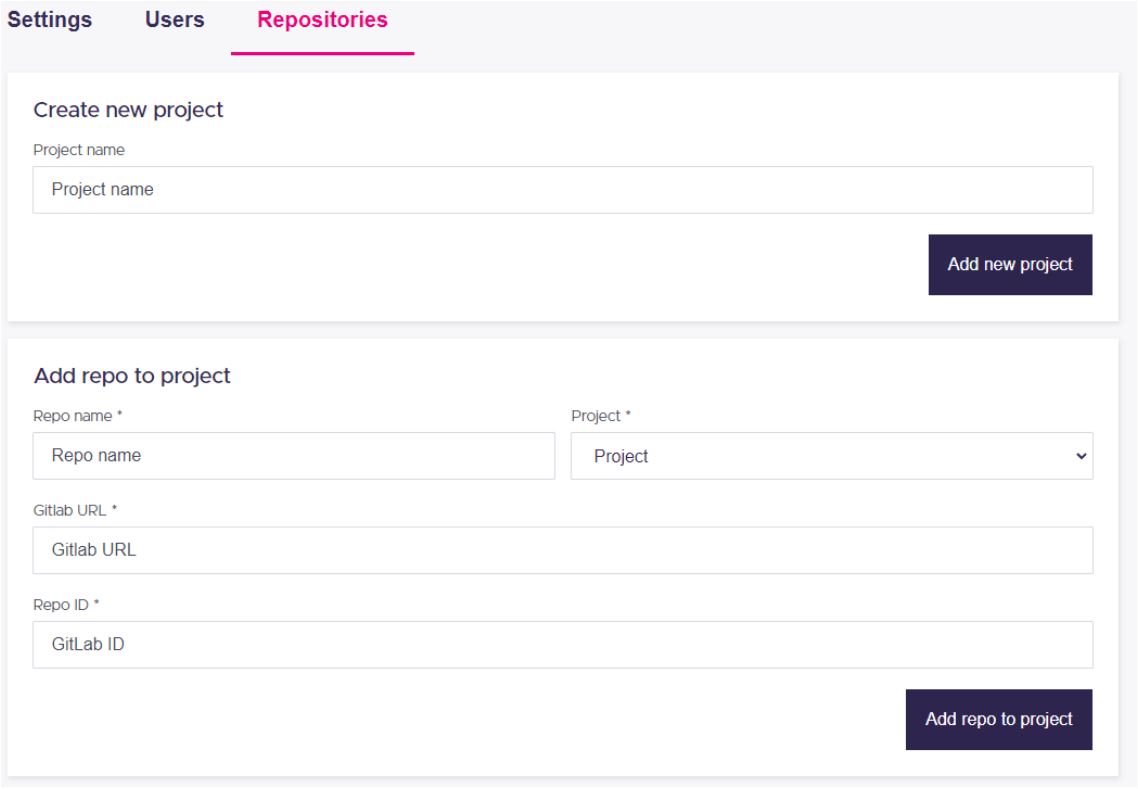
Joonis 11. Ekraanitõmmis kasutaja poolt loodud teavituste nimekirjast.

Teavituste süsteemi korrektseks töötamiseks oli sellele vaja lisada kontroll, kas kasutaja on lehe haldur. Siiani aga puudus funktsionaalsus lehe haldurite määramiseks või kontrollimiseks. Seega sai see juurde lisatud. Tagarakenduses on lehe haldurid implementeeritud

läbi Django sisseehitatud kasutaja gruppide funktsionaalsuse. Lehe haldurid kuuluvad gruppi nimega *Admin* ja sellesse gruppi kuulumist kontrollitakse lehe halduriks olemise kontrollimisel. Ka kasutajaliideses sai lisatud tõeväärtus muutuja halduriks olemise kohta salvestatavatele kasutaja andmetele. Lõpuks lisasime profiili vaatesse teise kasutaja halduriks tegemise võimaluse esialgse halduri poolt. Peale seda oli kõik vajalik olemas, et teavituste süsteem lõpetada ja vastav kontroll implementeerida.

4.3.3 Automatiseerimised

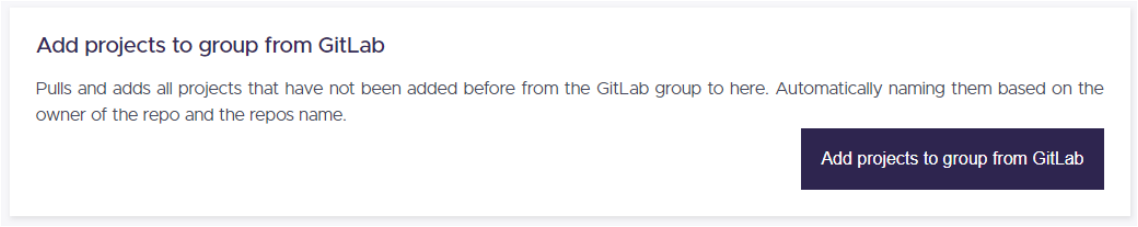
Rakenduse seadistamisel Ago Lubergi Tarkvaraarenduse projekti aines kasutamiseks anti abiõppejõudude poolt tagasisidet, et manuaalse seadistamise vajadus oli liiga suur. Juhendatavad tiimid lisati ükshaaval ning koodihoidlad lisati igale tiimile ühe kaupa (Joonis 12).



Joonis 12. Algne projektide ja nendega seotud koodihoidlate lisamise võimalus.

Eelnevad rakenduse tegijad olid arendanud tagarakenduses võimaluse automaatseks projektide loomiseks, kuid selle kasutamiseks polnud loodud võimalust kasutajaliideses. Ühe aine kontekstis loovad tudengite tiimid enda projektide jaoks koodihoidlad. Abiõppejõud saavad ligipääsu neile koodihoidlatele läbi GitLabis oleva grupi, millel on ligipääs kõikidele tudengite tiimide koodihoidlatele. Rakendus kasutab seda gruppi, et saada kõik sellega ühendatud koodihoidlad, nende alusel loob ta projektid, mille nimi tuleb GitLabis

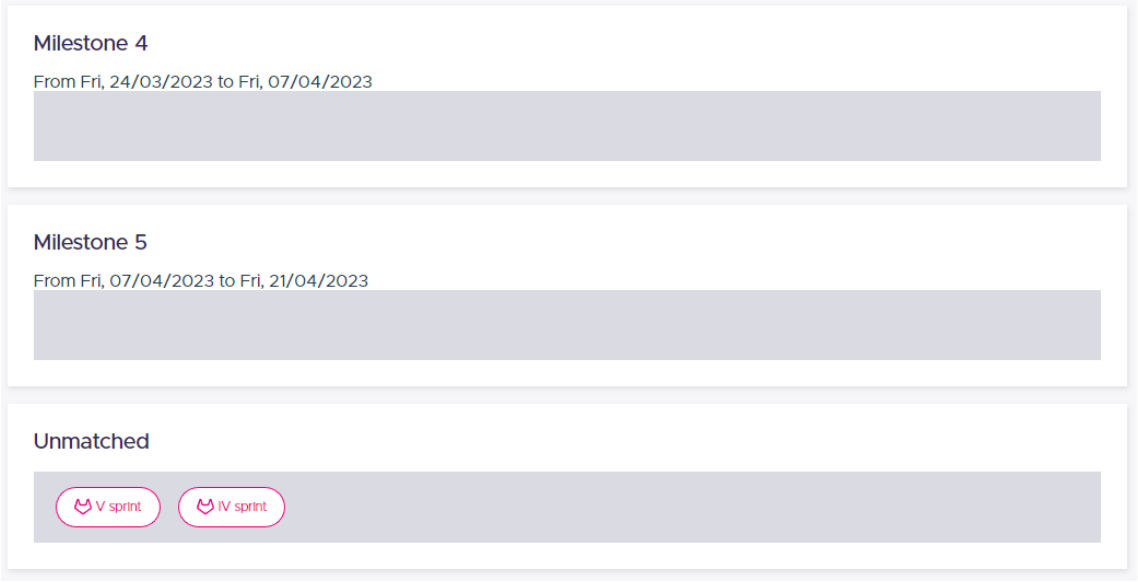
olevast projekti nimest ja projekti loojast, ning lisab projektile külge GitLabis oleva koodihoidla ja koodihoidlas olevad liikmed ehk suur osa tööst on lihtsustatud nuppu vajutusele (Joonis 13).



Joonis 13. Automatiseeritud projektide ja nendega seotud koodihoidlate lisamise võimalus.

Abiõppejõududel esines liigne manuaalse seadistamise probleem projektide andmete uuendamisel. Ülenädalaste *sprint*'ide lõpus pidid abiõppejõud igal juhendataval tiimil ühendama GitLabist saadud sprindid Cognate rakenduses kasutatavate tähtpunktidega (Joonis 14). Lahendusena sai loodud protsess, mis jookseb GitLabist andmete uuendamisel.

Grupis olevate andmete uuendamisel GitLabist rakendub programm uute *sprint*'ide korral ning arvutab kas on ajaliselt vastav tähtpunkt Cognates olemas. Kui GitLabist saadava sprindi algus ja lõpuaeg jääb vahemikku kaks päeva enne või kaks päeva pärast Cognates oleva tähtpunkti algus ja lõppaega, ühendatakse käesolev *sprint* tähtpunktiga ära.



Joonis 14. Ühendatama sprindid tiimi projektil Cognates.

Abiõppejõud avaldasid soovi hindamise automatiseerimiseks läbi automaatsete soovituste erinevatele hindamiskategooriatele. Eelnevad rakenduse arendajad olid arendanud tagarak-

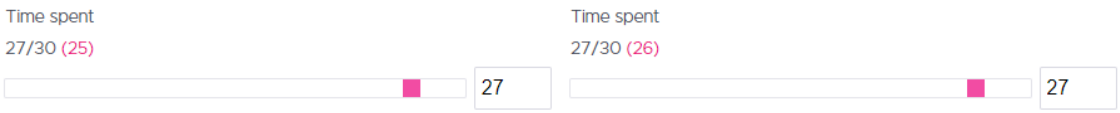
enduses võimaluse lihtsamate automaatsete hinnete soovitamiseks, kuid selle kasutamiseks polnud loodud visuaali kasutajaliideses.

Hinde automaatse arvutamise võimalused olid nende baasil:

- juhuslik (*random*)
- kulutatud aeg (*time spent*)
- koodiridu lisatud (*lines added*)
- ainulaadsete kehtestuse sõnumite suhe (*unique commit message ratio*)
- keskmine sõnade arv kehtestuse sõnumis (*average word count in commit message*)
- kehtestuste arv kasutajal (*commit amount for user*)
- keskmine sõnade arv kasutaja piletite kirjelduses (*average word count in issue description for user*)
- kasutajale määratud piletite arv (*issue amount for user*)

Uue automaatse arvutamise võimalusena sai lisatud planeerimine (*planning*). See arvestab määratud piletite suhet, piletite silte ja eeldatavat ajakulu piletitel. Hindamise alusena võeti tarkvaraarenduse projekti aines planeerimise hindamiskategooria ja selle selgitus.

Automaatse hindamise saab määrata hindamispuu sõlmele ning peab valima hindamise tüübi. Vastav sõlm on omakorda tähtpunkti all. Hinnatava sõlme soovitatavat hinnet saab vaadata vastava tähtpunkti vaatest (Joonis 15).



Joonis 15. Abiõppejõu poolt pandud punktid ja soovitatud punktisumma.

4.3.4 Ülikooli kasutajaga autentimine

Ülikooli emailiga sisselogimine lihtsustab Cognate kasutamist abiõppejõududel ja õppejõududel. Lihtsustus seisneb võimaluses sisse logida samasuguse e-posti aadressiga, mida kasutatakse ka teistes koolis kasutusel olevatel veebilehtedel. See vähendab vajadust meelde jätta erinevaid kasutajanime ja parooli kombinatsioone ning lihtsustab ka Cognates olevasse gruppi kutsumise süsteemi, kuna rakenduses ei ole enam vaja luua igale kasutajale eraldi ID-d, vaid saab kasutada juba olemasolevaid avalikke UniID-sid.

Vajutades Cognates sisselogimise nuppu, suunab programm kasutaja Microsoft OAuth

2.0 autentimise URL-ile, kus kasutaja saab Microsofti emailiga sisse logida (selle kaudu on loodud emailid kooli tudengitele ja õppejõududele). Edukal sisselogimisel saadetakse kasutaja tagasi Cognatesse koos autoriseerimiskoodiga, mis saadi Microsoft OAuth 2.0 autentimisel. See kood vahetatakse volitustõendi vastu. Lõpuks kasutatakse volitustõendi saamaks Microsoft Graph APIlt kasutaja email, läbi emaili kasutaja UniID ning ees- ka perekonnanimi. Seejärel logitakse kasutaja sisse vastavasse Cognate kasutajasse.

Selle abil on ka loodud vaade, mille abil on hinnatavatel tudengitel võimalus vaadata jooksvalt enda tulemusi ja hetkseisu projekti ainetes. UniID-ga sisse logides on võimalik valida enda projektide vaade, mis kontrollib, kas selle kasutaja nimega kasutaja on mingis projektis arendaja rolliga ja kuvab nimekirja nendest projektidest. Seejärel on võimalik projektile vajutades vaadata selle projekti vaadet peale tähtpunktide hindamise osa.

4.3.5 Suuremad parandused olemasolevale koodile

Kõige esimene probleem, mille me juba koodi kätte saamisel mõistsime, oli tagarakenduse käivitamine uuel masinal. Nimelt ei õnnestunud README failis antud juhistega projekti tööle saada. Uurimisel selgus, et probleem seisnes Django migratsioonides. Varasemalt tööle pandav migratsioon sõltus hilisemalt tööle pandud migratsioonist ja seetõttu ei läinud kõik migratsioonid tööle. Olukorra parandamiseks kirjutasime migratsiooni ümber mitte sõltuma hilisemast migratsioonist.

Kohe algusest oli teada, et olemasolev tagarakendus ei näita korrektselt GitLabis deklareeritud kulutatud aega. Nimelt ei arvestanud ta kustutatud ajaga. Selle parandamiseks lisasime tagarakendusele ka aja kustutamise sõnumitest *TimeSpent* objektide loomise, mis sisaldasid aja väärtusena negatiivset arvu. Saamaks teada kustutatud aja hulka kasutasime GitLabi piletitel esinevaid kommentaare, mis tekivad igal aja eemaldamise tegevusel. Erinevalt ajakulutuse kirjapanemise viisidest on aja kustutamise erinevatel viisidel ka erinevad sõnade järjestused kommentaarides, millega pidi arvestama. Kasutajaliideses ei tulnud nendega arvestamiseks midagi muuta.

Kommentaaride süsteem juba eksisteeris, kuid see ei olnud täielikult välja arendatud. Lisasime võimaluse kommentaare muuta ja kustutada. Selleks tuli tagarakenduses kirjutada vajalikud otspunktid ning kasutajaliidesesse lisada vastavad nupud. Kasutajaliides sai ka stiili poolest ümber tehtud, et ta sarnaneks rohkem TalTechi stiili juhistega.

Probleemne oli ka teistele kasutajatele läbi kasutajaliidese rollide andmine. Nimelt sai kasutajal olla sama roll mitu korda ja rolli kustutamisel kustusid kõik selle kasutaja rollid. Selle tõttu kadus kasutaja vastava grupi või projekti kasutajate loetelust ära ja teda sai

uuesti tagasi lisada ainult läbi tagarakenduse Django halduri vaate. Süüdlaseks osutus vigase loogikaga *if* lause, mille parandasime.

Kasutajaliideses tegime parandusi grupi hindamise konfigureerimise vaatele ehk hindamispuu loomise vaatele. Varasem versioon ei kuvanud loodavat puud õigesti, loodud sõlmede kustutamine ei töötanud ja peale uue sõlme loomist või kopeerimist ei ilmunud uus sõlm vaates enne lehe uuendamist. See kõik sai uues kasutajaliideses parandatud.

Kogu esitatav info rakenduses on saadud GitLabist, kuid teatud juhtumitel ei saadud kogu informatsiooni koodihoidlatest kätte. Asjaolu tulenes sellest, et GitLabist tõmmatav informatsioon arvestas ainult koodihoidlatega otseselt seotud tähtpunkte ja liikmeid, kuid ei arvestanud GitLabi grupi, mille alamprojekt see koodihoidla oli, tähtpunkte ja liikmeid. Selle jaoks sai muudetud olemasolevat otspunkti, mis tegeles GitLabist informatsiooni saamisega. Samuti sai kasutaja liidesesse loodud kaks valikut kasutajatele, kas nad soovisid informatsiooni uuendada koodihoidlate põhiselt või koodihoidlate ja nende ülemgruppide põhiselt.

Tarkvaraarenduse projekti aine käigus esines probleem liig-kasutajatega tiimide üksteise hindamiste käigus ehk Cognates projekti liikmete hulgas olid inimesed, kes ei kuulunud sellesse tiimi. Projekti ja tähtpunktide vaadetes infot kuvades olid esindatud kõik projektiga seotud kasutajad. Probleem tekkis juhtudel, kui GitLabi koodihoidlatesse lisati teise tiimi liikmeid projekti hindamiseks ning neile anti vähemalt *developer* tasemega õigus. Kuna *developer* tase on esimene, mis lubab kasutajatel liitmistaotlus teha on see Cognates määratud piiriks, kust alates arvestatakse neid kasutajaid kui liikmeid Cognate projektides. Probleemi lahendusena muudeti tagarakenduse otspunkte arvestamaks info tagastamisel vaid neid, kellel oli Cognates määratud *member* roll.

5. Tulemused ja valideerimine

Projekti edukus sõltub sellest, kui palju abiõppejõud ja õppejõud kasutavad seda ning kui rahul nad on selle kasutamise mugavusega. Selleks, et saada aru eelneva versiooni edasiarenduse võimalustest kasutati 2022. aasta erialatutvustuse aines (ITI0105) eelnevalt arendatud versiooni, millele oli tehtud muudatused, et seda tööle saada. Selle lõpus tehti kasutajaintervjuu ühe abiõppejõuga.

Edasiarenduse pädevuse kontrollimiseks sooritati laialt manuaalset testimist ja kasutati programmi 2023. aasta tarkvaraarenduse projekti aines (ITI0301). Aine alguses tehti kasutajaintervjuu sama abiõppejõuga ning aine jooksul küsiti tagasisidet kasutajatelt ehk abiõppejõududelt.

5.1 Kasutamine erialatutvustuse aines

Alates novembrist 2022 kasutati programmi varasemat versiooni hindamaks tudengeid erialatutvustuse aine viimase projekti raames. Rakendust kasutati iganädalaselt jälgimaks tudengite projektide kulgu, mille alusel pandi tudengitele hinded neljas kategoorias. Kursuse õppejõud sai programmi kasutada selleks, et saada ülevaadet projektide arengust ning abiõppejõudude hinnangutest.

Erialatutvustuse aine alguses peeti intervjuu abiõppejõuga, kes oli ka varasemalt olnud selles ja teistes ainetes abiõppejõud saamaks aru Cognate nõrkadest ja tugevatest külgedest enne aktiivsemat edasiarendust. Märkmed intervjuust on lisas 3.

Suurimad märkmed edasiarenduse võimalustest olid kasutajaliidese visuaali valdkonnas. Loetavus oli raskendatud heleda taustaga versioonis. Lahendamaks seda sai kasutajaliidese ümbertegemisel font muudetud ning tähtsam informatsioon nagu tiimide nimed esile toodud suurema fondiga. Abiõppejõududelt tagasiside nende muudatuste osas oli positiivne. Juhendatavaid tiime oli lihtsam üles leida, sest nimed said kiiremini loetud, ei pidanud pingutama, et väikest kirja lugeda. Samuti ei olnud enam liiga kontrastseid tekste, mida oli raske lugeda.

Graafikud olid segased, sest neil oli liiga palju infot korraga ning polnud tähistatud, mis informatsioon graafikul on. Ühe graafiku, kus oli kõik informatsioon edastatud, asemel sai loodud eraldiseisvad graafikud. Kõige positiivsema vastukaja sai aja kulutuse

joondiagramm, millelt oli lihtne välja lugeda, mis päevadel oldi aja kasutust kirja pandud ning kuidas töö aeg kulges. Samuti oli võimalik valida, kelle kohta informatsiooni tahetakse näha: üksikisik või tiim kokku. Eelnevalt kasutuses olnud sektordiagramm näitamaks aja jaotust tiimiliikmete vahel ei toodud üle, sest see informatsioon tuli paremini esile numbriliselt kirjutatuna tiimi ülevaates. Neutraalsem tagasiside tuli piletite graafiku ära võtmisest. Selle asemel sai loodud piletite ja nende aja kulutuste logi. Positiivsena toodi välja arusaadavus, et on võimalik neile peale vajutada, mille tõttu avanes GitLabis vastav pilet ning edastatava informatsiooni kogus aja kulutuse kohta. Negatiivsena toodi esile viivitus logi alla kerides ning korraga nähtavate piletite arv võiks suurem olla.

5.2 Kasutamine tarkvaraarenduse projekti aines

2023. aasta veebruarist hakati programmi uut versiooni kasutama tarkvaraarenduse projekti aines hindamaks tudengite mänguprojektide kulgu. Abiõppejõud kasutasid seda kord nädalas saamaks juhendavate tiimidest ülevaadet ja jälgimaks nende progressi aine raames. Üle-nädalaselt kasutati programmi hindamaks tudengite pädevust projektiplaanimises ning ainele kulutatud aega. Kursuse õppejõud sai programmi kasutada selleks, et saada ülevaadet projektide arengust ning abiõppejõudude hinnangutest.

Aine alguses tehti kasutajaintervjuu abiõppejõuga, mille käigus seati programm nullist üles, kasutamaks tarkvaraarenduse projektide hindamiseks. Käesoleva aine jooksul küsiti vabamas vormis jooksvalt tagasisidet ja tähelepanekuid abiõppejõududelt tehtud muudatuste ja võimalike arenduste osas.

Üks esimesi probleeme, mis esile tuli, oli aja kulutuse kattumatus Cognates esitatava info ja GitLabis tudengite poolt esitatava info vahel. See tulenes neljast asjast:

- kui Cognates ei olnud tiimi tähtpunktide külge pandud GitLabist saadud tähtpunkt
- kui pilet liigutati GitLabis ühest tähtpunktist teisse
- kui piletil oli aja kulutust ära võetud
- kui GitLabis polnud piletile määratud tähtpunkti

Lahendusena sai muudetud kulutatud aja arvutamist tiimide tähtpunktides. Selle asemel et vaadata aja kulutust olenevalt tähtpunktist, mille all piletid olid, määrati aja kulutus olenevalt hetkest, millal kirjutati aja kulutamise kommentaar GitLabis. Positiivse tagasisidena saadi palju täpsem aja kulutuse esitatus tiimi projekti vaadetes, kuid muudatusel olid ka negatiivsed tagajärjed. Kuna kõik piletid ja nendega seotud aja kulutused olid esindatud, näidati ka neid, mis ei olnud määratud tähtpunkti külge. Abiõppejõud pidid hinnates arvestama seda, et pilet peab olema tähtpunktiga ühendatud GitLabis. Seetõttu määrati

Cognates piirangud arvestatud aja kulutustele: need pidid olema ühendatud tähtpunktiga GitLabis ja GitLabi tähtpunkt peab olema Cognates ühendatud *Assessment Milestoneiga*. Lihtsustamaks seda protsessi abiõppejõududele sai ka loodud automaatne tähtpunktide ühtimine Cognates, mis määras uued GitLabist saadud tähtpunktid Cognates tiimidele automaatselt. Sai ka lisatud aja kulutuse ära võtmist arvestatav osa programmile. Need muudatused olid abiõppejõudude perspektiivist head, sest Cognates esitatud informatsioon oli täpsem, jooksvalt pidi vähem tiimide kohta käivat informatsiooni manuaalselt uuendama ning vähendas Cognates hindamiseks vaja minevat aega.

Järgmisena tulid esile erinevad kasutajaliidese puudujäägid või iseärasused, mis olid abiõppejõududele segased. Esiteks oli aja kulutuse kumulatiivsel graafikul päevi vaadates näha nooli ja tundide arvukusi. See oli märkimaks kulutatud aja muutust selle päeva lõikes, kuid see ei olnud abiõppejõududele visuaalselt kergesti aru saadav ning lisas liigset informatsiooni sellele graafikule. See sai muudetud näitamaks ainult sel päeval olnud kumulatiivset märgitud aja kulutust, mille kohal hiirt hoiti. See andis abiõppejõududele kiiremini edasi vajaliku informatsiooni sellelt graafikult. Jooksvalt öeldi ka, et kui tehti toiminguid ja kasutati Cognatet, ei olnud alati aru saada, kas tegevus sai tehtud või ei läinud see läbi. Andmaks kasutajale tagasiside tehtud toimingute kohta sai algselt loodud *Alert*'id, mida kuvati edukatel ning edututel tegevustel. Arenduse käigus said need *Alert*'id muudetud *Toast*'ideks, sest edastatav informatsioon edukuse kohta oli lühiaegne põgus tekst andmaks tagasisidet. Samuti oli võimalik järjest ja samaaegselt näidata sooritust erinevate tegevuste kohta. Abiõppejõududele poolt tuli sellele vägagi positiivset tagasisidet, sest nad said kindlalt teada, kas nende tegevused olid edukad, tänu millele läks neil vähem aega kontrollimaks, kas soovitud tegevus sai tehtud.

Kõige suurema mõjuga arendus oli hinde soovituse süsteemi integreerimine kasutajaliideses. Abiõppejõud said tiimide tähtpunktide hindamisel automatiseeritud punktisumma soovitust hindamiskategooriatele. Hetkel esinevad automaatsed hindamised on küllaltki algelised ja arvestavad vaid väikeseid kategooriaid nagu kulutatud aja hulk võrreldes vaja mineva aja hulgaga või tiimi liikme *commit*'idel esinev unikaalsete sõnumite arvukus või keskmine sõnade arv piletis. Ainus on kasutusel aja hulga arvestus, mis aitab abiõppejõududel olla kindel enda hindamises ning võimaldab õppejõul vaadata, kas hindaja poolt pandud punktid on liiga karmid või liiga leebed.

5.3 Tagasiside pärast tarkvaraarenduse projekti ainet

Tarkvaraarenduse projekti aine lõpus küsiti tagasisidet abiõppejõududelt küsitluse näol. Küsimused olid saamaks aru rakenduse hetkeseisust ning võimalikest lisadest tulevikus. Küsimused on lisas 4.

Abiõppejõududelt, kes olid Cognatet kasutanud eelneva semestri erialatutvustuse aines (iti0105-2022), küsiti kui rahul nad olid kasutajakogemusega eelnevas rakenduse versioonis, skaalal 1 kuni 9, enne selle lõputöö raames loodud muudatusi. Küsiti kui rahul nad olid Cognatega semestri alguses ning semestri lõpus, kui rahul nad oleksid kui lisataks kolm väiksemat või kolm suuremat muudatust ning kas nad näeksid Cognatel mingeid teisi kasutusvõimalusi või kasutasid nad ise seda mingil teisel otstarbel. Hinnanguskaalal tähendab 1, et rakenduse kasutamine oli halvem kui manuaalselt tudengite hindamine ning 9 tähendab, et on raske välja mõelda paremat süsteemi.

Küsimused, milles küsiti hinnangut Cognate, kui terviku kohta, olid keskmised hinnangud vastavad:

- Kui rahul olid Cognate kasutajakogemusega eelnevas aines, kus seda sai kasutatud? - 5,75
- Kui rahul olid Cognate kasutajakogemusega semestri alguses? - 6
- Kui rahul oled Cognate kasutajakogemusega semestri lõpus? - 7,6
- Kui lisataks kolm kõige tähtsamat muudatust/edasiarendust, millest eelnevas küsimuses kirjutasid, kui rahul oleksid sa Cognate kasutajakogemusega? (väiksemad muudatused) - 8,4
- Kui lisataks kolm kõige tähtsamat muudatust/edasiarendust, millest eelnevas küsimuses kirjutasid, kui rahul oleksid sa Cognate kasutajakogemusega? (suuremad muudatused) - 8,8

Vastustest on näha seda, et Cognate on abiõppejõududele olnud kasulik tööriist hindamisel. Samuti paistab, et uuendused ja juurdearendused on olnud abiõppejõududele meelepärased. Suurim rahulolu tõus tuli arenduse käigus tehtud muudatustega, mis baseerusid abiõppejõududelt saadaval jooksval tagasisidel. Tuleviku perspektiivis annaksid ka väiksemad muudatused suure mõju Cognate kasulikkusele ning suuremad muudatused oleksid plussiks, kuid pole esmatähtsad nende kasulikkuse mõjus.

Abiõppejõud, kes olid kasutanud Cognatet enne tarkvaraarenduse ainet (iti0301-2023), olid positiivsed selle kasulikkuses. Nende jaoks olid kõige tähtsamad funktsionaalsused võimalus saada ülevaadet tudengite panusest kursusel ja hinnata vastavat panust. Kõige nõrgemateks elementideks olid graafikud, mis olid abiõppejõududele segased, puuduv ülevaate saamine piletitest, mida tudengid olid teinud ja projektide või tähtpunktide seadistamise keerukus.

Samu küsimusi funktsionaalsuse kohta küsiti kõigilt abiõppejõududelt tarkvaraarenduse aine kestvuse jooksul tehtud muudatuste võtmes. Kõige tähtsamad funktsionaalsused olid

abiõppejõudude jaoks samad, mis eelnevalt. Vähem tähtsate funktsionaalsuste arv oli abiõppejõudude silmis langenud. Toodi välja kommentaaride lisamise võimalus, mida individuaalsed abiõppejõud olid vähem kasutanud ja projektide võrdlemise vaade, milles nähti suuremat kasu aine õppejõul.

Abiõppejõududelt küsiti erinevate vaadete arusaadavuse kohta hinnangut skaalal 1 kuni 9. Samuti küsiti, mida muudetaks nendes vaadetes, et need oleks arusaadavamad või mida võiks neis vaadetes veel olla esitatud.

Gruppide vaate keskmine hinnang abiõppejõudude poolt oli 8,6. Vaatega oldi väga rahul ning parandusi või muudatusi välja ei toodud.

Grupi projektide vaate keskmine hinnang abiõppejõudude poolt oli 8,4. Võimalike lisadena toodi välja võimalus näha millised tähtpunktid on tiimidel abiõppejõudude poolt hinnatud ja võimalus nägemaks kiiremini tudengite tiimide panuste erinevust ja ka nende aktiivsust.

Projekti kokkuvõtte vaate keskmine hinnang abiõppejõudude poolt oli 8,4. Probleemina toodi välja tähtpunkti hindamise radargraafik. Graafiku kategooriate punktid on visuaalselt skaleeritud suurima võimaliku punktisumma alusel. Kui ühe kategooria maksimaalpunktid on suuremad, kui teiste, jääb visuaalne tunne, et teistes kategooriates on hinnatavatel punktid puudu isegi kui nende punktisumma on maksimaalne.

Tähtpunkti hindamise vaate keskmine hinnang abiõppejõudude poolt oli 8,0. Võimaliku lisana toodi välja funktsionaalsus, mille kaudu saaks näha tähtpunktiga seotud pileteid ning kindlasti ka neid tähtpunktiga seotud pileteid, millele pole aja kulutus märgitud. Veel toodi välja, et võiks olla võimalik näha raporteid, mis tähtpunktile Gitlabis on lisatud.

Väiksemad võimalikud muudatused, mida abiõppejõud esile tõid:

- Juhend või õpetus, mis selgitaks Cognate kasutamist.
- Veel kategooriad, mille alusel programm annaks hindesoovitust.
- Tudengite UniID asemel oleks neile viidatud täisnimega.
- Projekti raportite tõmbamine GitLabist ja näitamine.

Suuremad võimalikud muudatused, mida abiõppejõud esile tõid:

- Hinded laetaks automaatselt aine Moodle'isse.
- Ühe grupi projektide hindamise täpsuse võrdlus, et oleks näha kas abiõppejõud on võrdselt hinnanud tiime.

- Rakendus laeks andmeid kiiremini.
- Oleks visuaal, mis näitaks projekti harusid, nagu on selleks võimalus GitLabis.
- Tudengid saaksid esitada soovitud hinnet või punktisummat koos seletustega, mille alusel abiõppejõud saaks vaadata, kas nad on seda hinnet väärt.

Tagasisides toodi välja ka kasutusvõimalusi Cognatele, mis ei olnud seotud tarkvaraarenduse ainega. Kõige rohkem toodi välja võimalust vaadata enda ajapanuseid teistes tiimiprojektide ainetes ja lõputöös.

6. Edasiarenduse võimalused

Lõputöö käigus sai küll automatiseerimist edasi arendatud, kuid veel on hulgaliselt kohti, kus kasutaja töö lihtsamaks teha. Tänu võimalusele logida sisse ülikooli kasutajaga on võimalik juurde arendada antud punktide automaatne ülekandumine Moodle'isse. Ka hindamises endas on võimalik lisada veel automaatse hindamise kriteeriume ja võimalusi. Näiteks võiks võimalik olla ühele sõlmele lisada mitu automaatset hindamist, mida ta arvestaks. Veel on võimalik juurde arendada automaatne hindamine, mis jälgiks koodihoidlas tehtud harusid ja *merge request*'e ning annaks hinnangu nende korrektsusele. Uus hindamine võiks kontrollida, kas ühele harule vastab üks pilet ja kas *merge request*'id on loodud ühe inimese poolt ja vastu võetud teiste inimeste poolt.

Gruppide ja projektide üles seadmises leidub kohti, mida lihtsamaks ja automaatsemaks teha. GitLabi grupist automaatsel projektide tõmbamisel tekib praeguses versioonis iga koodihoidla kohta uus projekt. Vajalik käitumine oleks aga ühe tiimi poolt loodud erinevate koodihoidlate grupeerimine ja lisamine ühte projekti. Ka koodihoidlate käsitsi liigutamiseks ühest projektist teise võiks olla mugavam lahendus, kui praegune tõmba ja lase lahti. Veel võiks arendada juurde kasutajate kutsumisel gruppi võimaluse lisada mitu kasutajat korraga. Hetkel on abiõppejõudude lisamine gruppi ükshaaval ajakulukas ja tüütu. Ka grupi üles seadmisel loodava hindamispuu tegemine on ajakulukas ja tihti repetiivne. Hetkel on võimalik kopeerida sõlme selle grupi raames. Sellele saaks juurde arendada võimaluse kopeerida tervet haru või lausa tervet puud teistesse gruppidesse. Võib oletada, et aastast aastasse jäävad ühe aine hindamise kriteeriumid sarnaseks ning terve hindamispuu kopeerimine ja kasutamine uue aasta grupis teeks selle üles seadmise tunduvalt kiiremaks. Veel võiks läbi Cognate olla võimalik vaadata erinevaid projekti harusid.

Veel üks soovitud edasiarendus on arendajatele võimaluse andmine teiste arendajate hindamiseks. Mõnes õppeaines on üks projekti osa teiste tiimide hindamine ning seda võiks saada teha läbi Cognate. Praegune hindamine käib läbi Moodle'i, kuid selle üles sättimine on vägagi aja kulukas ja tüütu ning pole ei abiõppejõududele ega tudengitele mugav. Tudengitel oleks ka parem ülevaade hinnatavatest läbi Cognate ning abiõppejõududel oleks töö lihtsustatud, sest kõik hinnatav oleks ühes keskkonnas. Saaks veel juurde lisada võimaluse kommenteerida hinnatavat tiimi, andes hinnatavale kohest tagasisidet töö probleem kohtade osas ning tähelepanekud mis hindaja tiimil olid.

Ka kasutajaliideses on võimalusi edasiseks arenguks. Paar lisa funktsionaalsust, mida saaks juurde teha on lehe tõlkimine erinevatesse keeltesse ja *dark mode* kasutamise võimaluse lisamine. Uus kasutajaliides on küll varasemast tunduvalt paremini märgistatud ja arusaadavam, kuid endiselt on uuel kasutajal selle tundma õppimiseks vaja mõningast juhendamist. Seetõttu võiks kasutajaliidesele lisada kas lehe kasutamist selgitava video või erinevatesse kohtadesse erinevate tegevuste juhised, mida vajadusel saab lahti võtta.

7. Kokkuvõte

Tallinna Tehnikaülikoolis on IT teaduskonnas tiimiprojektide hindamise lihtsustamiseks kasutusel varasema lõputöö, "GitLabi projektide metaandmete analüüsi ökosüsteem"[1] käigus loodud süsteem Cognate. Cognate kogub GitLabi erinevatest koodihoidlatest andmeid, hoiustab neid, läbi kasutajaliidese visualiseerib need kasutajale ning pakub võimalust nende põhjal tiime hinnata. Süsteem aga polnud lõplik ja selles leidus võimalusi edasiarenduseks. Lõputöö käigus parandasime süsteemis ilmnenud probleeme ja arendasime sellele uusi funktsionaalsusi.

Edasiseks arenduseks jätkasime olemasoleva tagarakenduse arendamist, kuid tegime uue kasutajaliidese. Uus kasutajaliides jälgib TalTechi stiilijuhist ning kasutab uuemat Vue3 raamistikku ja tulevase edasiarenduse lihtsustamiseks TypeScripti. Kasutajaliidest arendades keskendusime selle kasutajasõbralikumaks ja kiiremaks tegemisele.

Probleemide ja peamiste edasiarenduste tuvastamiseks vestlesime kliendiga ning viisime läbi intervjuusid süsteemi kasutavate abiõppejõududega. Sellest saadud info põhjal lisasime süsteemile võimaluse logida sisse TalTechi kasutajaga, uue vaate projektide paremaks omavaheliseks võrdlemiseks ja teavituste süsteemi. Veel arendasime edasi varasemalt poolikuks jäänud hindamise automatiseerimist. Parandatud sai ka intervjuudes ja arenduse käigus ilmnenud programmivead.

Lõputöö tulemusena on Cognate-t lihtsam kasutada, see on terviklikum ja kiirem. Sellegipoolest leidub veel võimalusi edasiseks arenduseks.

Kasutatud kirjandus

[1] Kristjan Kõiv Mart Kaasik. *GitLabi projektide metaandmete analüüsi ökosüsteem*. [Kasutatud: 18/04/2023]. URL: https://digikogu.taltech.ee/et/Item/2c285178-6624-4d25-885e-c92f7aa7bb9f.

[2] GitLab. *REST API*. [Kasutatud: 26/03/2023]. URL: https://docs.gitlab.com/ee/api/rest/.

[3] TalTech. *TalTech Digital Styleguide*. [Kasutatud: 26/03/2023]. URL: https://portal-dev.ttu.ee/styleguide/?path=/story/docs-intro--default.

[4] Vue3 FAQ. *Is Vue 2 Still Supported?* [Kasutatud: 26/03/2023]. URL: https://vuejs.org/about/faq.html#what-s-the-difference-between-vue-2-and-vue-3.

[5] Vuex. *Pinia is now the new default*. [Kasutatud: 26/03/2023]. URL: https://vuex.vuejs.org/#what-is-vuex.

[6] Nuxt.js. *Announcing 3.0*. [Kasutatud: 26/03/2023]. URL: https://nuxt.com/blog/v3.

[7] Jennifer Bland. *Vue 3 Composition API vs Options API*. [Kasutatud: 26/03/2023]. URL: https://www.jenniferbland.com/vue-3-composition-api-vs-options-api/.

[8] XOR. *4 Reasons Why 'var' is Considered Obsolete in Modern JavaScript*. [Kasutatud: 26/03/2023]. URL: https://javascript.plainenglish.io/4-reasons-why-var-is-considered-obsolete-in-modern-javascript-a30296b5f08f.

[9] Chameera Dulanga. *Measuring Performance of Different JavaScript Loop Types*. [Kasutatud: 26/03/2023]. URL: https://blog.bitsrc.io/measuring-performance-of-different-javascript-loop-types-c0e9b1d193ed.

[10] Anthony Fu. *Why I don't use Prettier*. [Kasutatud: 11/04/2023]. URL: https://antfu.me/posts/why-not-prettier.

[11] Markus Oberlehner. *Vue Project Directory Structure: Keep It Flat or Group by Domain*. [Kasutatud: 11/04/2023]. URL: https://markus.oberlehner.net/blog/vue-project-directory-structure-keep-it-flat-or-group-by-domain/.

[12] Alex Jover Morales. *Structuring Vue Components*. [Kasutatud: 11/04/2023]. URL: https://vueschool.io/articles/vuejs-tutorials/structuring-vue-components/.

[13] VSCode. *TypeScript in Visual Studio Code*. [Kasutatud: 26/03/2023]. URL: https://code.visualstudio.com/docs/languages/typescript.

[14] TTÜ Tarkvaraarenduse Instituut. *Kasutajaliidesed (ITI0209)*. [Kasutatud: 26/03/2023]. URL: https://courses.cs.ttu.ee/pages/Kasutajaliidesed_(ITI0209).

[15] Maureen Stone. *A few words on background color*. [Kasutatud: 26/03/2023]. URL: https://www.perceptualedge.com/articles/b-eye/choosing_colors.pdf.

[16] Page Laubheimer. *Circular and Area-Based Graphs Are Difficult to Interpret Quickly or Accurately*. [Kasutatud: 26/03/2023]. URL: https://www.nngroup.com/articles/dashboards-preattentive/.

[17] Ilya Krukowski. *Internationalization vs. localization (i18n vs l10n): What's the difference?* [Kasutatud: 10/04/2023]. URL: https://lokalise.com/blog/internationalization-vs-localization/.

[18] Microsoft. *Overview of Microsoft Graph*. [Kasutatud: 15/04/2023]. URL: https://learn.microsoft.com/en-us/graph/overview.

[19] Cognate. *README.md*. [Kasutatud: 26/03/2023]. URL: https://gitlab.cs.ttu.ee/cognate/cognate-a-la-tal-tech/-/blob/main/README.md.

Lisa 1 - Vana kasutajaliidese koodibaasi struktuur

```
public
src
  App.vue
  assets
    d3
       index.js
  axios-api.js
  components
    AuthLogin.vue
    AuthLogout.vue
    AuthModal.vue
    AuthRegister.vue
    BaseLightswitch.vue
    BaseNavbar.vue
    ColorPicker.vue
    DragBoard.vue
    DragCard.vue
    GroupAdminGeneral.vue
    GroupAdminRepos.vue
    GroupAdminUsers.vue
    GroupRepositoriesManager.vue
    LoadingAnimation.vue
    ProgressBar.vue
    ProjectAssessStudent.vue
    ProjectAssessTeam.vue
    ProjectDeveloper.vue
    ProjectManageSprints.vue
    ProjectManageUsers.vue
    ProjectMilestoneCard.vue
    ProjectTotalStats.vue
    visualizations
       CourseAssessmentTreeLinear.vue
       DonutChart.vue
       GitTime.vue
       ProjectMilestoneBarcharts.vue
       ProjectRadar.vue
  main.js
  routes.js
  store.js
  views
     BaseFeedback.vue
     BaseHome.vue
     BasePricing.vue
     BaseProfileSettings.vue
     GroupAdd.vue
     GroupAdminView.vue
     GroupAssessingTree.vue
     GroupList.vue
     GroupMilestoneSummary.vue
     GroupProjects.vue
     Project.vue
     ProjectAssessSprint.vue
     ProjectManagementView.vue
```

Lisa 2 - Uue kasutajaliidese koodibaasi struktuur (seis 11.04.2023)

```
public
  favicon.ico
  fonts
  index.html
README.md
src
  App.vue
  assets
    images
    locales
  common
    helpers.ts
    mappers
      accountMappers.ts
      groupAssessmentMapper.ts
      groupConfigMappers.ts
      groupsGroupListMappers.ts
      projectConfigMappers.ts
      projectMappers.ts
      projectsMilestoneListMappers.ts
      projectsProjectListMappers.ts
    types
       accountTpyes.ts
       commonTypes.ts
       groupAssessmentTypes.ts
       groupConfigTypes.ts
       groupsGroupListTypes.ts
       projectConfigTypes.ts
       projectsMilestoneListTypes.ts
       projectsProjectListTypes.ts
       projectTypes.ts
  components
    account
      AccountFormLogin.vue
      AccountFormRegister.vue
      AccountProfile.vue
    charts
      ChartBase.vue
    layout
      AppFooter.vue
      AppHeader.vue
      Breadcrumbs.vue
    ui
       DraggableBoard.vue
       DraggableItem.vue
       InformativeTooltip.vue
       TalTechAlert.vue
       TaltechBadge.vue
       TalTechCard.vue
       TalTechCollapse.vue
```

```
      TalTechComments.vue
      TalTechFilterbar.vue
      TalTechLoader.vue
      TalTechModal.vue
      TalTechProgress.vue
echarts
  d3.js
  index.ts
http-client.ts
main.ts
router
  index.ts
services
  IServiceResponse.ts
  README.md
  useRequests.ts
  useUserService.ts
stores
  useGroupStore.ts
  useNotificationStore.ts
  useUserStore.ts
style
  global.less
  multiselect.css
  toastify.css
  variables.less
views
   account
     LoginView.vue
     LogoutView.vue
     ProfileView.vue
     RegisterView.vue
   assessment
     AssessmentTreeChart.vue
     AssessmentView.vue
   feedback
     FeedbackForm.vue
     FeedbackView.vue
   group
     GroupConfigGeneral.vue
     GroupConfigProjectAdd.vue
     GroupConfigProjectsAdd.vue
     GroupConfigRepoAdd.vue
     GroupConfigRepos.vue
     GroupConfigUsers.vue
     GroupConfigView.vue
   groups
     GroupAdd.vue
     GroupInvites.vue
     GroupList.vue
     GroupsView.vue
   home
     Home.less
     Home.vue
     HomeView.vue
   notification
     NotificationForm.vue
     NotificationList.vue
     NotificationView.vue
```

```
project
  BarChartTimeSpent.vue
  LineChartsTimeSpent.vue
  ProjectIssueLogs.vue
  ProjectMilestoneCard.vue
  ProjectView.vue
projectComparision
  ProjectComparisionView.vue
  ProjectPointChart.vue
  ProjectTimeCharts.vue
projectConfig
  ProjectConfigSprints.vue
  ProjectConfigUsers.vue
  ProjectConfigView.vue
projectMilestone
  ProjectMilestoneAssessStudent.vue
  ProjectMilestoneView.vue
  RadarPointsSpread.vue
projects
  BarChartMilestonePoints.vue
  MilestoneSummary.vue
  ProjectList.vue
  ProjectsView.vue
studentView
   StudentView.vue
```

Lisa 3 - Märkmed esimesest intervjuust abiõppejõuga (seis 15.04.2023)

Programmi mõte abiõppejõu perspektiivist

GitLabis tiimide projektide tegevuse jälgimise hõlbustamiseks ja hindamiseks.

Situatsioon enne Cognatet

Oli hull kasutada excelist, sest polnud visuaali. On vaja selgitust saada, et mida kuvatakse ning mis andmed on.

Kõige tähtsam asi, mis lihtsustaks?

Kuskile saab punktid ühte kohta panna. Et ei peaks moodleit kasutama või saatma õppejõule, kes siis ise paneks moodleisse. Võibolla tundide kuvamine, et ei pea ise kokku arvutama, vaid on normaalselt kuvatud. Kõik on ühes kohas, see info ja hindamine. Kuidas peaks hindama oleks ka kirjas. Kõik ei oleks laiali.

Milline võiks olla ülevaate leht kõikidest tiimidest, kui mitte võtta baasiks eelnevalt olnud Cognatet?

Tiimi üldine info, palju kõik kokku panustasid, palju pileteid sai tehtud. Peale vajutades saaks rohkemat informatsiooni. Tiimi nimi, liikmed oleksid ülevaates näha, rohkemat ei suutnud välja mõelda. Abiõppejõud vaatab koguaeg täpsemat infot tiimi kohta. Otsib tiimi ülesse ja vaatab täpsemalt, ei kasuta ülevaadet oma hinnatavate tiimide jaoks.

Tiimi ülevaate leht, mis tuleb esile kui võtta lahti hinnatav tiim.

Iga inimese kohta oleks informatsiooni. Tiimi võrdlus liikmete vahel, et oleks näha kui keegi teeb kahtlaselt vähe või keegi liiga palju. Aeg on kõige tähtsam asi mida kasutada võrdlemiseks, piletite arvu järgi on raske, sest mõned tudengid teevad suuremaid pileteid mõned väiksemaid (päevapikkused piletid vs 10 minuti pikkused piletid). Koodiread ei ole ka hea asi mille alusel võrrelda.

Kõik koos informatsioon ei ole kõige tähtsam, põhifookus on hetkel hinnataval tähtpunktil. Tahaks näha informatsiooni kindla tähtpunkti kohta.

Pole mõtet ükshaaval kulada või eraldi tiimiliikmete kohta vaateid teha. Kui üldvaates on juba näha üldist infot tähtpunktide ja liikmete kohta, pole vaja ükshaaval liikmeid läbi vaadata. Pole mõtet hakata ükshaaval neid kuvama lehel. Vaatan seal enamasti vaid tema aega. Oleks tähtpunktide ülevaade, kus oleks näha iga tähtpunkti kohta, et mis iga liikme ajakulu, punktid ja tehtud piletite arv on. Oleks võimalus valida tähtpunktide vahel, et millist täpselt näha või hinnata.

Mis informatsioon võiks täpse tähtpunkti vaates olla?

Aeg ja võrdlus tiimiliikmete vahel. Pole vahet, mis hetkel ajakasutust kirja pani tähtpunkti jooksul, võib kõik ajakasutuse kokku panna. Palju pileteid kokku oli, milles liige osales.

Gitlabis on branchide graph mille kaudu saab vaadata, kas tegid igale piletile branchi ning kas mergimine on õigesti tehtud. Kas on kasutatud branche, mitte tehtud pushe otse maini. Branche ei mergeiks sama inimene, kes branchi lõi.

Tiimid paneks kirja tähtpunkti kohta raporti. Selle tähtpunkti raportit oleks otse näha, et ei peaks GitLabi wikit avama.

Mis informatsiooni vaatab abiõppejõud veel GitLabis, mida oleks võimalik rakenduses otse näidata?

Piletid, et mis nende pealkirjad on, mis nende kirjeldused on. Kas tiimid on normaalseid pileteid teinud. Kas pilet on seotud mingi tähtpunktiga.

Cognate varasem versioon

Abiõppejõule anti kasutada Cognate varasemat versiooni. Küsiti erinevate vaadete nõrkuste ja tugevuste kohta ja üldiseid tähelepanekuid.

Kasutajad loodi abiõppejõududele ette ning siis jagati need abiõppejõududele. See ka pigem oli hea, et ei pidanud ise tegema kasutajat.

Cognate üldiselt

Oli kaks versiooni veebilehest nö dark-mode ja light-mode. Ei olnud nende kasutuses eelistust, light-modeis oli esilehte veidi raske lugeda. Abiõppejõud ise kasutab igapäevaselt dark-modei rohkem.

Võiks olla onboarding mille kaudu saaks kogu vajaliku info, et kuidas kasutada seda rakendust ning mis kõik võimalused on.

Gruppide vaade

Groups refreshimist ei teinud abiõppejõud ise, ei teadnud ka mis asi see oli.

Light-modeis on kehva näha võrreldes dark-modeiga, asjad kaovad ära, raske näha ning midagi häirib. Loetelu gruppidest on hea, kuid esmasel pilgul ei olnud aru saada, et kuhu peab vajutama, et näha täpse grupi vaadet.

Gruppide assessment tree

Käis ainult selleks, et vaadata, mis punktid olid teatud tähtpunktil ja kuidas need muutusid vastavalt tähtpunktile, muidu ei saanud aru eriti lehest. Kui peaks lisama juurde tähtpunkti, ei saaks sellel versioonil hakkama.

Puu on hea viis punktide nägemiseks, aga järjekord tähtpunktidel oli arusaamatu, et miks asus igaüks seal kus asus. Tähtpunktid olid omavahel segamini.

Juurde lisada pole vaja, annab edasi vajaliku informatsiooni.

Grupi config

Selle vaatega samuti ei puutunud kokku. Ei käinud enda tiime ega nende koodihoidlaid lisamas, see oli tehtud eelnevate arendajate poolt. Arvestades projektiainetes abiõppejõududele suurema koormuse panemist, võis eeldada, et tulevikus peavad abiõppejõud ise lisama grupile tiimid ja/või koodihoidlad.

Oli segane, miks oli kaks eraldiseisvat asja (projekti ehk tiimi lisamine ning koodihoidla lisamine projektile) visuaalselt liiga sarnased ning tundusid nagu sama asi. Tiimi lisamine oli loogiline, kuid koodihoidla lisamisega ei olnud kindel, mis asi oli küsitav GitLab ID.

Tegu oli GitLabis oleva koodihoidla Project ID-ga. Võiksid olla küsimärgid asjade juures, mis seletaks lahti millega tegu on. Või võiks mingi tutorial olla, et kuidas teha.

Koodihoidla lisamisel on segane, mis nime panna koodihoidlale. Kui võtta *slug* GitLabist, oleks kõigil tiimidel järelikult sarnane kui mitte sama koodihoidla nimi. Selle tõttu pandi nimeks pikem versioon url-ist, kus oli ka koodihoidla looja UniID. Projektid on loomise järjekorras võiks olla viisi teist moodi neid järjestada, näiteks tähestiku järjekorras.

Koodihoidlat projektile lisades ei olnud tagasisidet, et see sai lisatud. Sai mitu korda lisatud koodihoidlad. Võiks olla ka viis koodihoidlaid ja/või projekte kustutada. Koodihoidlate ümber tõstmise korral peab neid lohistama, mis on esimene mõte, mis ka tekib, kuid võiks olla ka teine viis projektiga seotud koodihoidla muutmiseks.

Grupi vaade

Laadimisaeg oli suva, vaatasid seda sõõrikut[eelnevalt olnud animatsioon laadimise ajal] rõõmsalt. Tulpad, mis tähistasid iga sprindi punktisummat on hea. Aga esmapilgul oli raske algselt aru saada, mida need tähistavad/näitavad. Tiimiliikmetel olid värvid sarnased, kuid oli võimalus selle muutmiseks. Refresh nupu juures oleks võinud olla kirjas, et mida see teeb.

Projekti vaade

Vaates on palju infot, algsel pilgul ei ole aru saada, et mis on mis ja mida asjad tähistavad. Läks aega, et aru saada projekti ülevaatest. Kokku kulutatud aeg osast polnud aru saada, kas oli tiimi liikmete peale kokku või liikmete keskmine näidatud [oli näidatud tiimi peale kokku]. Kõige all asetsev ajagraafik oli ka väga segane, ei saanud aru, mis toimub. Aru saamise asemel anti sellega alla ja ei kasutatud aktiivselt. Graafik oleks võinud olla seletus, kolm graafikut ühes tegid selle eriti segaseks ja kirjuks. Ei olnud ka aru saada mida iga osa graafikust näitab. Võiks olla eraldi või valida, millist graafikut näitab.

Tähtpunkti vaade

Graafikul piletitele vajutamine, mis ka avas need, oli hea. Üleval vasakul radiaalgraafik näitamaks punktide seisu tähtpunktis tiimi peale kokku ei olnud vajalik. Tudengite hindamisel võiks ka olla punktide sisestamise võimalus, et ei oleks ainult slidei liigutamine. Graafikult võiks saada vaadata iga inimese kohta eraldi informatsiooni ajakasutuse kohta. Hindamisel aitaks kaasa kui oleks automaatse hinde soovitus, mis oleks näidatud hinnatava

punktide kõrval. Kommentaarid võivad olla, need võiks ka näidata tähtpunkti, mille all kirjutati. Võiks ka selles vaates olla neid nähe, mitte ainult projekti üldvaates.

Manage project

Tähtpunkte võiks saada liigutada muul viisil kui lohistades. Liikmete värvi muutmine on positiivne.

Tolleaegne uus frondi versioon

Hea, et teksti on parem lugeda. Tiimide tähtpunktide punktid on paremini nähtav gruppi vaates, sest on visuaalselt suuremad. Tiimide mahutavus võiks parem olla. Tiimid võiks olla tähestiku järjekorras. Saaks valida, et näeks ainult enda juhendatavaid tiime. Dark mode ei ole vajalik, on harjunud kooli lehtedega, millel on sarnane visuaal.

Lisa 4 - Cognate tagasiside küsitlus (seis 21.05.2023)

- Kas oled Cognatet kasutanud enne iti0301-2023 ainet?
- Kui rahul olid Cognate kasutajakogemusega eelnevas aines, kus seda sai kasutatud?
- Mis olid selle parimad funktsionaalsused/elemendid?
- Mis olid selle nõrgemad funktsionaalsused/elemendid?
- Kui rahul olid Cognate kasutajakogemusega semestri alguses?
- Kui rahul oled Cognate kasutajakogemusega semestri lõpus?
- Mis on sinu jaoks kõige tähtsam funktsionaalsus Cognates?
- Mis on sinu jaoks kõige vähem tähtis funktsionaalsus Cognates?
- Kui arusaadav on gruppide vaade?
- Mida sa muudaksid gruppide vaates?
- Kui arusaadav on grupi projektide vaade?
- Mida sa muudaksid grupi projektide vaates?
- Kui arusaadav on projekti kokkuvõtte vaade?
- Mida sa muudaksid projekti kokkuvõtte vaates?
- Kui arusaadav on tähtpunkti hindamise vaade?
- Mida sa muudaksid tähtpunkti hindamise vaates?
- Millised oleksid väiksemad muudatused/edasiarendused, mida sa sooviksid Cognatesse lisada?
- Kui lisataks kolm kõige tähtsamat muudatust/edasiarendust, millest eelnevas küsimuses kirjutasid, kui rahul oleksid sa Cognate kasutajakogemusega?
- Millised oleksid kõige suuremad muudatused/edasiarendused, mida sa sooviksid Cognatesse lisada?
- Kui lisataks kolm kõige tähtsamat muudatust/edasiarendust, millest eelnevas küsimuses kirjutasid, kui rahul oleksid sa Cognate kasutajakogemusega?
- Kas sa näed mingit teist kasutusvõimalus Cognatele / Kas sa kasutaksid Cognatet saamaks ülevaadet mõnest enda projektist?
- Kas on lisa kommentaare?