Tartu Ülikool

Arvutiteaduse instituut

Informaatika eriala

Joosep Näks

DeepMOOC platvormile tagarakenduse dispetšeri arendamine

Bakalaureusetöö (9 EAP)

Juhendajad:

Ahti Põder
Tõnis Hendrik Hlebnikov

Tartu 2022

DeepMOOC platvormile tagarakenduse dispetšeri arendamine

Lühikokkuvõte:

DeepMOOC platvorm on loodav keskkond tudengitele ja õppejõududele progrmeerimisülesannetes esitatud programmikoodi automaattestimiseks. Platvormi idee sündis asjaolust, et praegu Tartu Ülikoolis kasutusel olev lahendus, Virtual Programming Lab, on piiratud programmeerimiskeelte toega ning ei sisalda mõningaid võimalusi, mis oleks kasulikud programmeerimisaineid läbi viies. DeepMOOC platvormi eesmärk on need piirangud kaotada ning tulevikus saada universaalseks platvormiks, kus läbi viia programmeerimisainete raames automatiseeritavaid tegevusi. Need ei pea piirduma ainult klassikalise ühiktestimisega, vaid võivad olla näiteks ka koodi staatilist analüüsimist või ajakulu mõõtmist. Bakalaureusetöö raames arendatakse DeepMOOC platvormile tagarakenduse dispetšerit. Tarkvara eesmärk on korraldada suhtlust eesrakenduse, andmebaasi, failiserveri ning koodi jooksutamise konveieri vahel. Töös arutletakse tehnoloogiliste valikute üle ning kirjeldatakse valminud rakenduse struktuuri. Töö tulemusena valmis tagarakendus kõige selle sooritamiseks ning andmebaasi skeem andmete hoiustamiseks.

Võtmesõnad: tarkvaraarendus, tarkvaratehnika, programmeerimisõpe
CERCS: P175 Informaatika, süsteemiteooria

Backend dispatcher development for the DeepMOOC platform

Abstract:

The DeepMOOC platform is a new environment in the making, for the students and lecturers to carry out automatic testing of the program code from course assignments. The idea for the platform came from the fact that the current solution used in the University of Tartu, Virtual Programming Lab, has limited support for programming languages and is missing some features that would be useful in programming courses. The DeepMOOC platform aims to eliminate these constraints and eventually become an universal go-to platform for all kinds of automated tasks that can be performed in the context of programming subjects. These do not have to be limited to regular unit-testing but can, for example, include statical code analyses and performance measurements. This thesis describes the development and architecture of the backend dispatcher of the DeepMOOC platform. The goal of the dispatcher is to organise the communication between the frontend, database, file server and pipeline. This paper discusses the technological choices and gives an overview of the architecture of the final software. The result of this paper is a backend dispatcher and the DeepMOOC database schema.

Keywords:
CERCS: P175 Informatics, system theory

Sisukord

Sissejuhatus

DeepMOOC on loodav keskkond tudengitele ja õppejõududele programmeerimisalase hariduse omandamiseks ja edasiandmiseks. Platvormi idee on muuta programmeerimisalast õpet huvitavamaks ja kaasahaaravamaks, sealjuures tekitades tudengite lahenduste vahel võistlusmomenti, millel on potentsiaali parandada õppetulemusi.

Edetabelite loomine olemasoleva automaatkontrolli lahenduse VPL (Virtual Programming Lab) [1] juurde Tartu Ülikooli programmeerimisainetes "Algoritmid ja andmestruktuurid" ja "Programmeerimine II" on juba praeguseks näidanud tudengite motivatsiooni kasvu programmeerimisalaste kontseptsioonide omandamisel[1]. DeepMOOC platvorm arendab seda ideed edasi, võimaldades õppejõududel püstitada ülesandeid, millele esitatavat koodi testitakse ning analüüsitakse automaatselt. Tulemuste põhjal tekib jooksev edetabel, mille pingerida moodustub esitatud lahenduste korrektsuse, tööaegade, ajaliste keerukuste või muude mõõdetavate väärtuste alusel.

Võrreldes juba olemasoleva ja Tartu Ülikoolis kasutatava lahendusega VPL, püüab DeepMOOC olla oluliselt kasutajasõbralikum, võimaldades automaatkontrolli lihtsamini ja teistele arusaadavamalt üles seada. Lisaks ei piirdu see tavalise ühiktestimisega (*unit-testing*), vaid on VPList oluliselt võimsam, võimaldades automaatkontrolliks üles seada erinevaid tegevusi. Näiteks on sellega võimalik kontrollida tudengi loodud andmebaasi struktuuri või teostada mõõtmisi programmi ajakulu ning mälukasutuse kohta. Seega saab platvormi valmides automaatkontrolli kasutusele võtta ka ainetes, kus see seni VPLi piirangute tõttu pole võimalik olnud [2].

Platvormi arendamisega seoses on valminud juba üks lõputöö, mis tegeleb DeepMOOC platvormil tudengite koodi hindava konveieri arendamisega. [2] Paralleelselt käesoleva tööga on valmimas teine lõputöö, mis käsitleb DeepMOOC platvormil lahenduste hindamise jaoks keele arendamist. Käesoleva bakalaureusetöö raames arendatakse DeepMOOC platvormi tagarakenduse osana dispetšerit, mille ülesanne on eesrakendusest tulevad päringud vastu võtta, nende põhjal suhelda andmebaasi, konveieri ja failiserveriga ning tulemus eesrakendusele tagastada. Töö eesmärk on saada rakendus valmis piisavas mahus, et see võimaldaks kõiki tudengitele vajalikke operatsioone kasutaja autentimisest esitatud lahenduste hindamise ja tulemuste ning edetabeli tagastamiseni.

Töö on jagatud neljaks osaks. Teoreetilise ülevaate peatükis võrreldakse DeepMOOC platvormi olemasolevate lahendustega ning arutletakse arendamiseks kasutatud tehnoloogiate valikute üle. Tööprotsessi peatükis kirjeldatakse arenduse käiku ja arenduses tekkinud problee-

[1] Põhineb selle töö juhendajate kogemusel, mis on saadud mainitud õppeaineid läbi viies.

me ja nende lahendusi. Projekteerimise peatükis lahatakse rakenduse tööks vajalikke nõudeid ja rakenduse struktuuri. Verifitseerimise peatükis seletatakse nõuete testimiseks vajalikke testijuhtumeid ning üldist testiprotseduuri.

1 Teoreetiline ülevaade

Selles peatükis antakse lühike ülevaade DeepMOOC platvormi olemasolevate sarnaste platvormide kohta. Seejärel seletatakse tehnoloogiate valikuid tarkvara arendamiseks.

1.1 Olemasolevad lahendused

1.1.1 Virtual Programming Lab

Virtual Programming Lab (VPL) [1] on Moodle pistikprogramm, mis võimaldab jooksutada automaatteste tudengite programmeerimisülesannete lahenduste peal. See on hetkel Tartu Ülikoolis kasutusel ning selle puudujääkide tõttu tekkis vajadus luua uus platvorm, milleks on DeepMOOC. Mõned VPL'i olulisematest puudustest on:

- Piiratud võimalused hindamiskriteeriumite kohandamiseks.
- Tagasiside viiside vähesus, näiteks võrdlus teiste esitustega.
- Testitava programmi vähene isoleeritus keskkonnast, (operatsioonisüsteem, Moodle teised komponendid) mis võib tekitada olulisi turvaauke.

VPL sisaldab automaattestide jooksutamiseks teenust nimega VPL-Jail-System, mis ei kasuta isoleeritud keskkonna loomiseks konteinereid ega ka muud virtualiseerimist. Seega on võimalik, et jooksutatav kood saab turvaauke kasutades muuta ka väljaspool testimiskeskkonda olevaid andmeid. DeepMOOC platvormil välditakse koodi ligipääsu süsteemi andmetele testi eraldi konteineris jooksutamisega. [2]

Olulised VPL'i eelised DeepMOOC platvormi ees tulenevad peamiselt sellest, et VPL on Moodle'i pistikprogramm. Tudengite andmeid pole vaja peale Moodle' teise süsteemi duplitseerida ning saab kasutada Moodle' autentimist. DeepMOOC platvormi autentimislahendusena on tulevikus plaanis kasutusele võtta sama teenus, mida kasutab Moodle. Samas andmebaaside ühendamiseks selliselt, et testkeskkond jääks soovitult isoleerituks, pole autorid seni lahendust leidnud ning DeepMOOC hakkab kasutama eraldiseisvat andmebaasi.

1.1.2 Artemis

Artemis [3] on õppeplatvorm programmeerimisülesannete automaattestimiseks. Lisaks toetab see ka muid ülesandetüüpe nagu modelleerimist ja testiküsimusi ning sisaldab ka plagiaadituvastust. Antud töö kontekstis on oluline vaid programmeerimisülesannete automaattestimine. Artemis kasutab testide käivitamiseks versioonihaldussüsteemi Atlassian Bitbucket. Õppejõud loob git repositooriumi testidega ning tudengi jaoks on loodud IntelliJ arenduskeskkonda pistikrakendus Orion, mis tõmbab repositooriumi alla ning kasutab seda koodi testimiseks.

Artemis ei sobi sama eesmärki täitma nagu DeepMOOC, kuna Artemis annab tagasisidet vaid igale tudengile eraldi ning puudub võimalus süsteemi sees esitatud töid võrrelda (näiteks jooksva edetabeli koostamiseks). See kaotab tudengitel võimaluse ennast teistega võrrelda ning võistlemiseesmärgil ennast parandada, mis on DeepMOOC platvormi üks eesmärke.

1.2 Tehnoloogilised valikud

1.2.1 Go

Go on avatud lähtekoodiga Google'i arendatud keel. Go on kasutajasõbralik kuid ei ohverda selle nimel võimsust. [4] Go sobib eriti hästi serveri tarkvara arendamiseks tänu sisseehitatud *go routine*'idele, mis teevad paralleliseerimise väga lihtsaks. [5] Lisaks pakub Go mitmeid kasulikke teeke, mis käesoleva rakenduse arendamist oluliselt aitasid. Näiteks on olemas teegid PostgreSQL andmebaasi ja Amazon S3 failiserveriga suhtlemise jaoks ning teek veebiserveri loomiseks ilma et oleks vajadust täiendavatele raamistikele. Go on DeepMOOC platvormil kasutusel terve dispetšeri ja ka konveieri loomiseks.

1.2.2 PostgreSQL

PostgreSQL on üks populaarsemaid avatud lähtekoodiga relatsioonilise andmebaasi haldamise süsteeme. [6] Töös on seda kasutatud kasutajaandmete ja testitulemuste hoidmiseks.

1.2.3 Amazon S3

Amazon S3 on failihoiustamisteenus, mida pakub Amazon Web Services. DeepMOOC platvormil on Amazon S3 kasutusel tudengite lahendusfailide hoiustamiseks. Selle eelis kohaliku hoiustamise ees on skaleeruvus – kui failid liiga palju ruumi võtavad saab suurema paketi tellida. [7]

2 Tööprotsess

Arendus võttis aega kokku neli kuud. Sellel perioodil põhilised etapid olid *endpoint*ide tööpõhimõttest aru saamine, andmebaasi struktuuri koostamine, enamiku dispetšeri realiseerimine ning failiserveri päringute kohta õppimine. Arendusele eelnevalt õppis autor Go keelt kasutama.

Go keele õppimiseks lahendas autor Go keelseid harjutusi Codewars[1] keskkonnas ning luges Go ametlikku dokumentatsiooni. Esialgselt Go keele selgeks saamine suuri raskusi ei valmistanud tänu selle sarnasustele Pythoni ja C++ keeltega, mida mõlemat autor varem valdas. Segadust tekitas küll esimese hooga Go projekti struktuur kuid peale foorumites näidetega tutvumist osutus ka see täitsa arusaadavaks.

Arendusprotsessis esimene etapp oli üldisest veebiserveri ülesehitusest arusaamine ning lihtsamate *endpoint*ide loomine. Sellega aitas kaasa Go teek http, millel on väga põhjalik dokumentatsioon. [8]

Teiseks etapiks oli andmebaasi struktuuri loomine ning üles seadmine. Selle jaoks kasutas autor SqlDBM[2] veebikeskkonda, kus sai andmebaasi struktuuri diagrammile kanda ning keskkond genereeris ise sql koodi, millega andmebaas koostada. SqlDBM keskkonna genereeritud koodis olid küll mõned puudused aga automaatsest andmebaasiskripti genereerismisest saadud ajavõit kaalus oluliselt üles vigade uurimisele kulunud aja. Näiteks kasutab SqlDBM automaatselt kasvava väärtusega võtmeveeru jaoks võtmesõna *SERIAL* kuigi see on aegunud ning selle asemel peaks kasutama võtmesõna *GENERATED AS IDENTITY*. [9] Puudused sai käsitsi parandandatud ning rohkem probleeme andmebaasi ülesseadmisega polnud.

Kolmas ja kõige sisukam etapp oli dispetšeri realiseerimine ja enamiku koodi kirjutamine. See protsess kulges üldiselt probleemideta. Üks asi mis palju kaasa aitas, oli vahetarkvara (*middleware*) kasutamise võimalus, mille autor etapi lõpupoole avastas. Vahetarkvara laseb luua näiteks koodijupi, mida kutsutakse välja iga kord kui eesrakendus mõne *endpoint*i (otspunkti) poole pöördub enne seda kui vastav *handler* (käsitleja) välja kutsutakse ja võimaldab

[1] https://www.codewars.com
[2] https://sqldbm.com/

kontrollida, et päring on autenditud. See on mugav viis koodi kordamise ärahoidmiseks. Samuti lihtsutab see koodi struktuuri ja garanteerib, et teenusteülesed kohustuslikud toimingud alati täidetud saavad. [10]

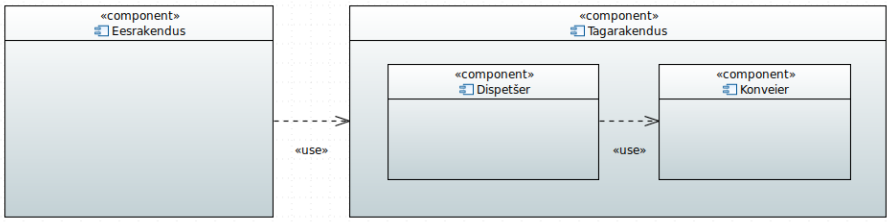
Viimaseks etapiks oli failiserveri sidumine dispetšeriga. Selle jaoks on Amazon loonud sobiva teegi Go keelele, millega saab väga lihtsalt S3 API'le päringuid esitada. Teek oli ka hästi Amazoni enda poolt dokumenteeritud seega probleeme sellega ei tekkinud. [11]

Dispetšeri arendamise ajal polnud veel eesrakendust arendama hakatud. See tegi arendust keerulisemaks, kuna *endpoint*idele ei olnud mingeid eesrakendusest tulenevaid nõudeid ning autor pidi ise välja mõtlema, milliseid *endpoint*e vaja võib minna ja milliseid andmeid need peaks väljastama.

Arendustöö sujuvamaks kulgemiseks koostas autor ka ühiktestid olemasoleva tarkvara kontrollimiseks. Testid valmisid küll vaid http päringute kujul ning seda, kas päringu vastus ning päringujärgsed muudatused andmebaasis ja failiserveris on korrektsed, tuleb käsitsi kontrollida. See on aeganõudev ning platvormi hilisemas arendusjärgus on plaanis luua automaatne tarkvara, mis testitulemusi kontrolliks.

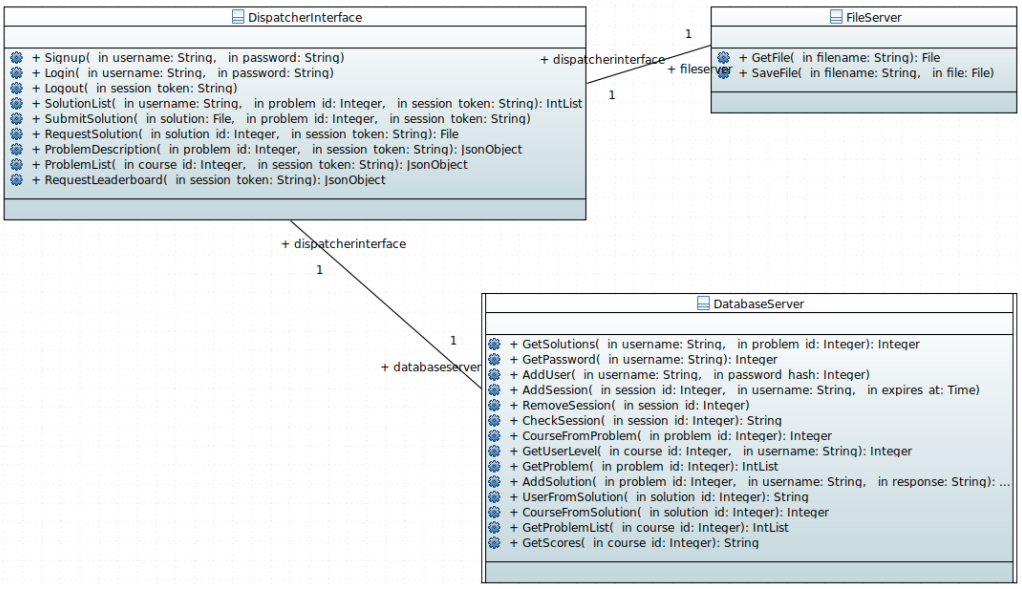
3 Projekteerimine

Rakendus koosneb eesrakendusest, mis suhtleb kasutajaga, ning tagarakendusest, mis realiseerib kogu funktsionaalsuse (joonis 1). Tagarakendus omakorda jaguneb dispetšeriks ja konveieriks. Konveierit kirjeldatakse töös [2]. Käesoleva töö sisuks on dispetšer.



Joonis 1: Süsteemi arhitektuur

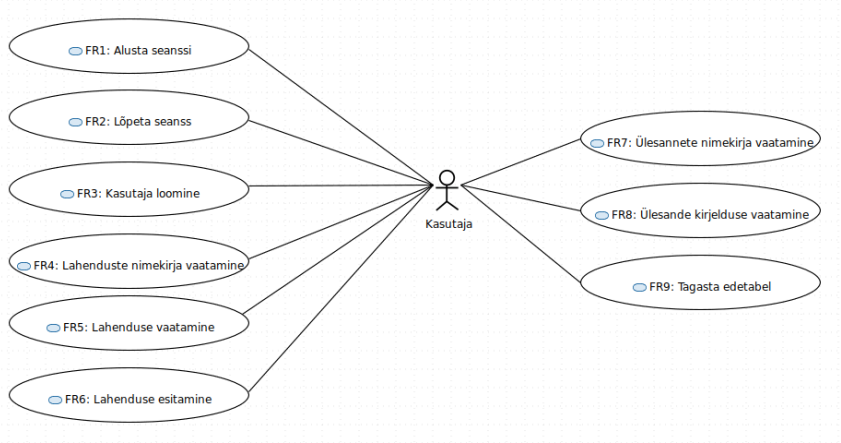
Dispetšeri põhikomponent on liides eesrakenduse jaoks, mis sisaldab kõigi sõnumite töötluseks vajalikku loogikat. Välistest komponentidest kasutab dispetšer failiserverit lahendusefailide salvestamiseks, andmebaasi kasutajate info hoidmiseks ning konveierit lahenduste hindamiseks (joonis 2).



Joonis 2: Dispetšeri arhitektuur

3.1 Nõuded

Dispetšer peab võtma vastu sõnumeid eesrakenduselt, käivitama tagarakenduses sõnumi sisule vastava tegevuse ja tagastama küsitud andmed. Toetatud kasutajategevused on esitatud joonisel 3.



Joonis 3: Süsteemi nõuded

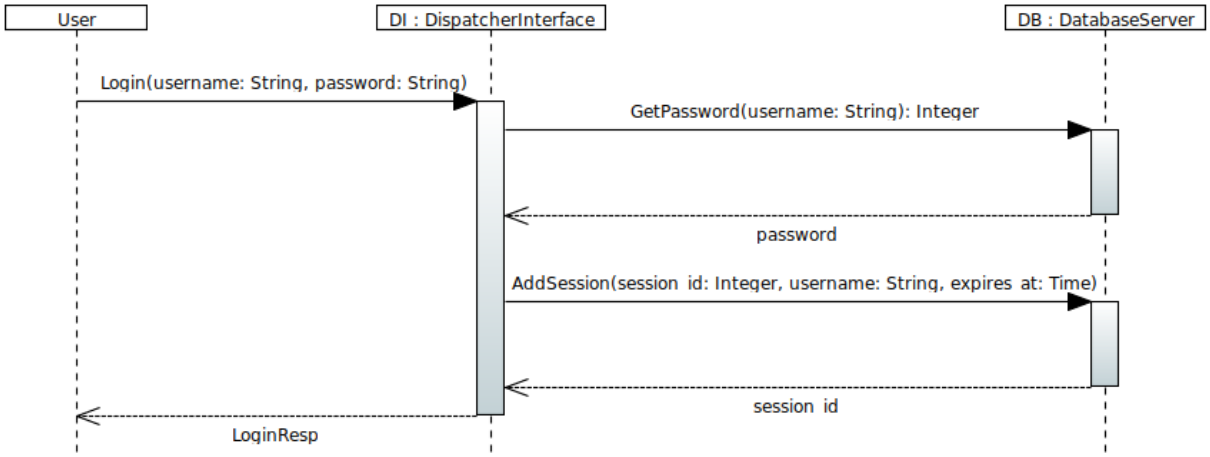
Kõigi päringute juures, välja arvatud “Alusta seanssi”, on vaja kontrollida, et päringu esitajal oleks õigus vastavale tegevusele. Selleks antakse igal sisselogimisel eesrakendusele seansi identifikaator, mis salvestatakse ka andmebaasi aegumisajaga. Edaspidi iga päringuga annab eesrakendus seansi identifikaatori kaasa ning tagarakendus kontrollib andmebaasist, et identifikaator aegunud ei oleks, ning saab andmebaasist ka kätte päringu esitaja kasutajanime ja õiguste taseme. Kõigi päringute puhul, välja arvatud “Lõpeta seanss”, uuendatakse andmebaasis seansi aegumise tähtaega.

Nõuete jooniselt on näha, et kasutaja omab ligipääsu kõigile tegevustele kaasa arvatud uue kasutaja tekitamine. Sellisena realiseeriti süsteem käesoleva lõputöö raames. Lõplik autentimismehhanism koos kasutajate ja kursuste haldamisega on veel kokku leppimata. Antud töö raames realiseeriti päring uue kasutaja tekitamiseks. Kasutaja sidumine kursusega ning kursuses õiguste andmine on hetkel võimalik ainult otse andmebaasis. Seetõttu pole ka kasutaja loomise võimaldamine kõigi kasutajate puhul otseselt turvaauk – on võimalik küll lisada uus kasutaja aga kuna teda kursustega siduda ei saa, siis ei ole sel kasutajal võimalik ka süsteemis sisalduvaid andmeid näha ega muuta.

Nõuded on jagatud funktsionaalseteks ja mitte funktsionaalseteks nõueteks. Funktsionaalsed nõuded on tähistatud lühendiga FR ehk *functional requirement* ja järjekorranumbriga ning mittefunktsionaalsed nõuded on tähistatud lühendiga NFR ehk *non-functional requirement* ja järjekorranumbriga.

3.1.1 FR1: Alusta seanssi

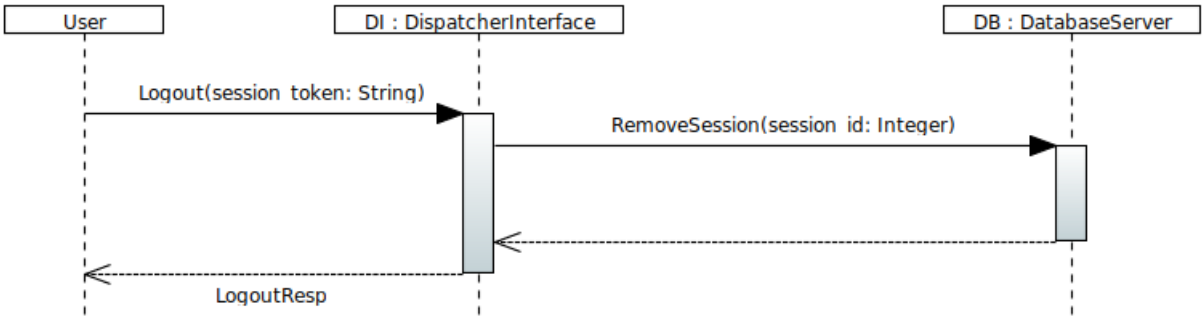
Sesssiooni alustamine on aluseks kõigile muudele toimingutele. Kasutaja sisestab kasutajanime ja parooli. Süsteem kontrollib, et kasutajanimi eksisteeriks andmebaasis ning selle leidmise korral võrdleb saadud salasõna räsi andmebaasis oleva räsiga. Kui sobiv vaste leidub, siis alustab uut seanssi, salvestab selle identifikaatori koos aegumistähtajaga andmebaasi ja tagastab identifikaatori päringu vastuses (joonis 4).



Joonis 4: Seansi alustamise järgnevusskeem

3.1.2 FR2: Lõpeta seanss

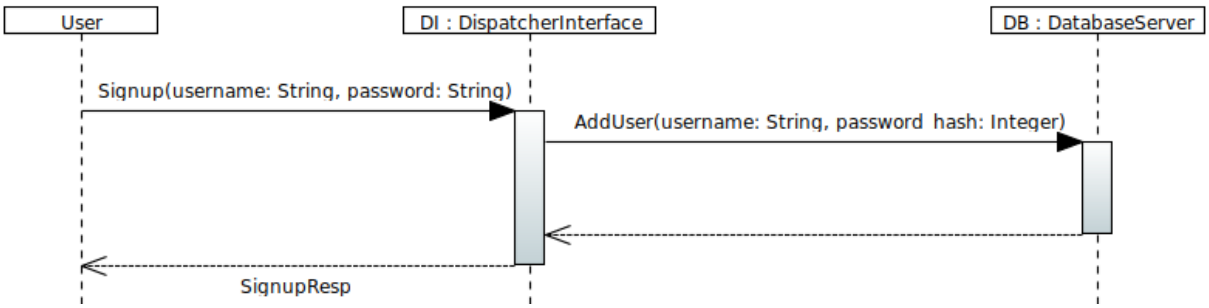
Seansi lõpetamine muudab kasutaja seansi identifikaatori kehtetuks. Kasutaja saadab päringu väljalogimiseks ning süsteem eemaldab seansi identifikaatori andmebaasist (joonis 5).



Joonis 5: Seansi lõpetamise järgnevusskeem

3.1.3 FR3: Kasutaja loomine

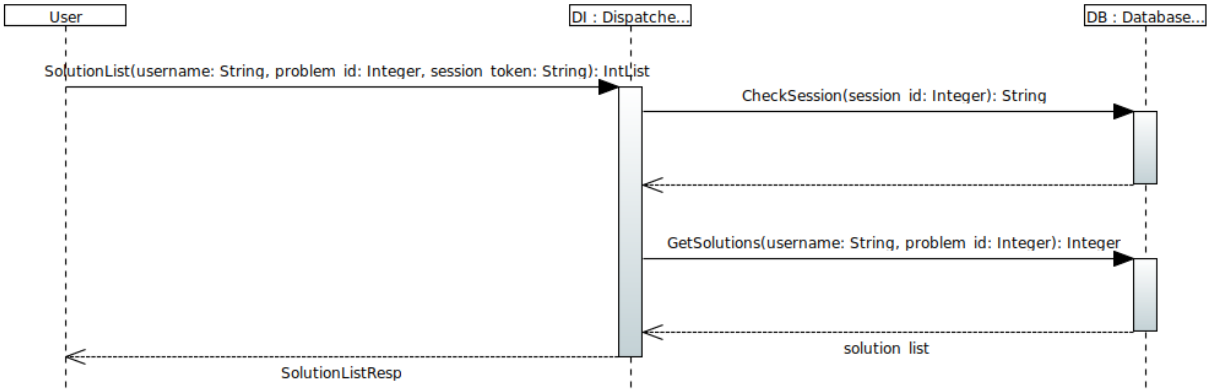
Kasutaja sisestab kasutajanime ja salasõna. Süsteem kontrollib, et kasutajanime juba ei leiduks andmebaasis. Kui ei leidu, salvestatakse kasutajanimi ja salasõna räsi andmebaasi (joonis 6).



Joonis 6: Kasutaja loomise järgnevusskeem

3.1.4 FR4: Lahenduste nimekirja vaatamine

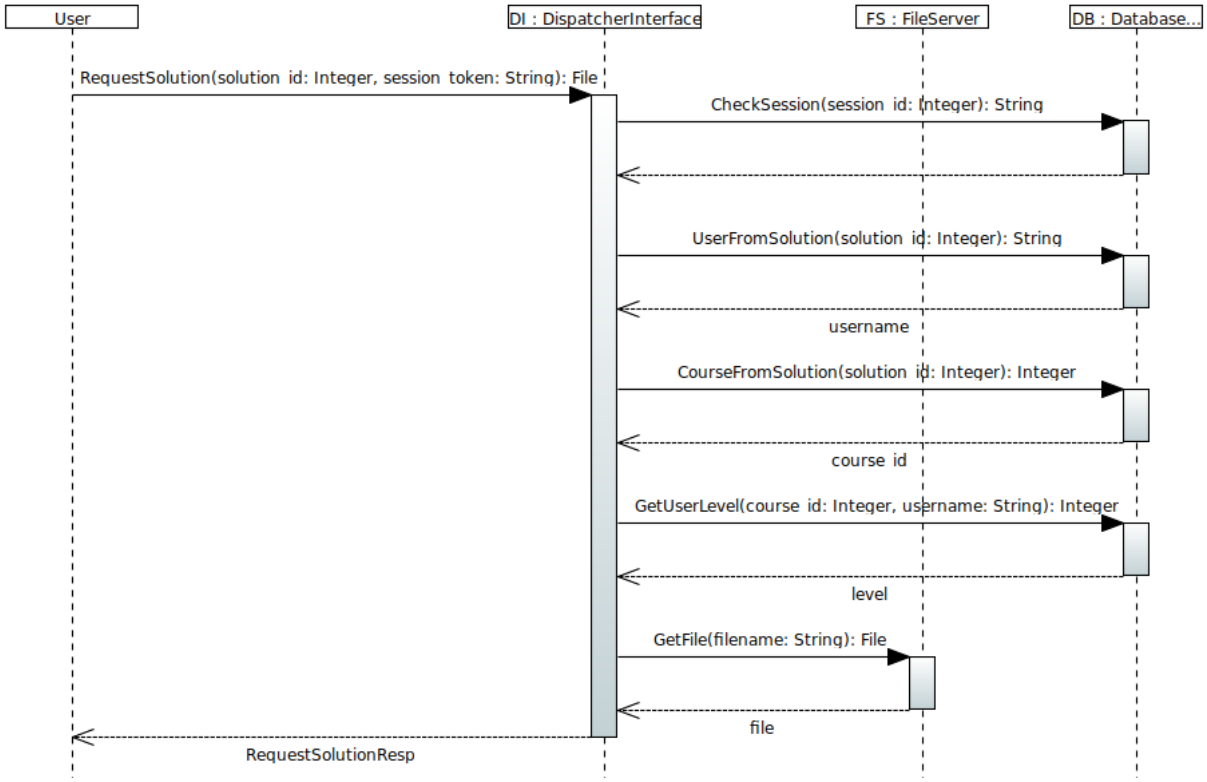
Kasutaja esitab päringu ülesande lahenduste vaatamiseks. Süsteem kontrollib, kas kasutaja küsib enda lahendusi või on sellel kursusel õppejõu rollis. Kui tingimus on täidetud, võtab süsteem andmebaasist lahenduste identifikaatorite nimekirja ning edastab need kasutajale (joonis 7).



Joonis 7: Lahenduste nimekirja järgnevusskeem

3.1.5 FR5: Lahenduse vaatamine

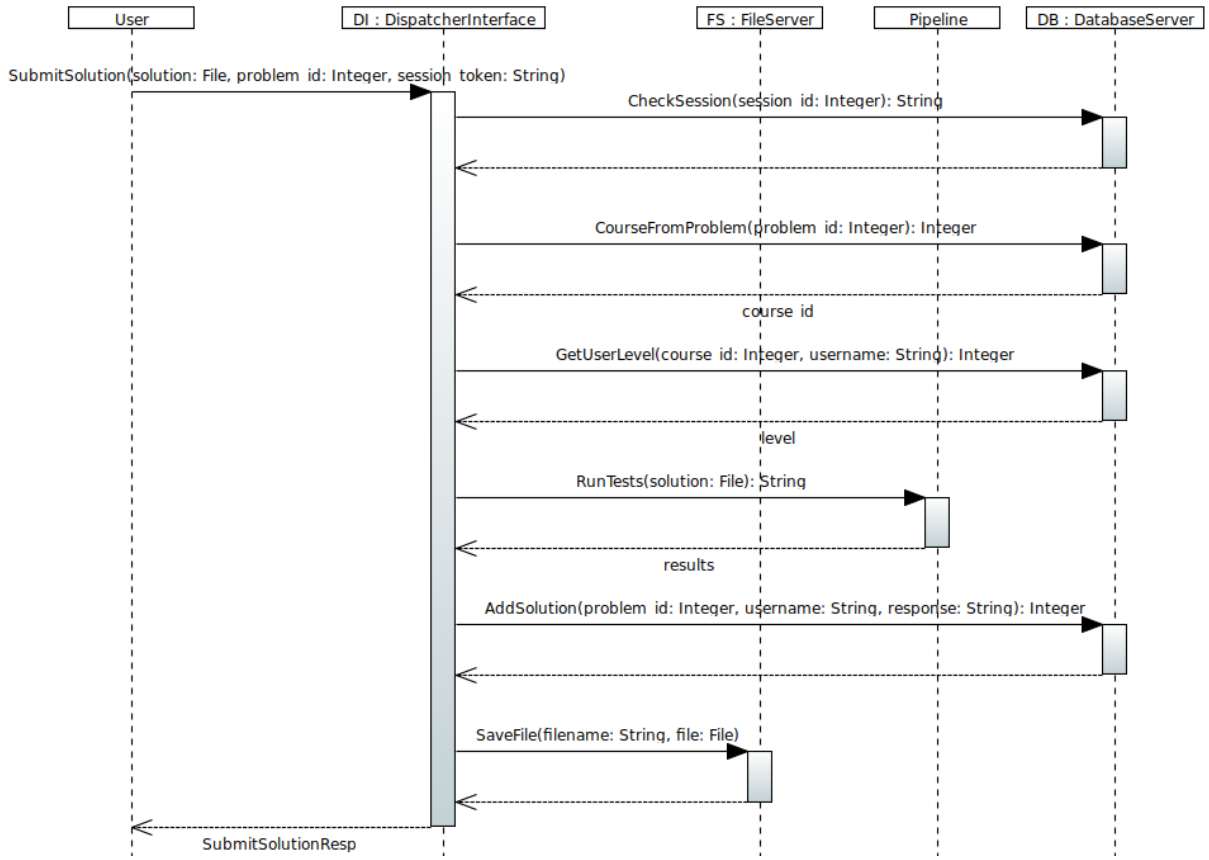
Kasutaja esitab päringu ülesande vaatamiseks. Süsteem kontrollib, kas küsitav lahendus on kasutaja oma või kasutajal on õppejõu õigused sellel kursusel. Kui tingimus on täidetud, võtab süsteem failiserverist lahenduse ja edastab selle kasutajale (joonis 8).



Joonis 8: Lahenduste pärimise järgnevusskeem

3.1.6 FR6: Lahenduse esitamine

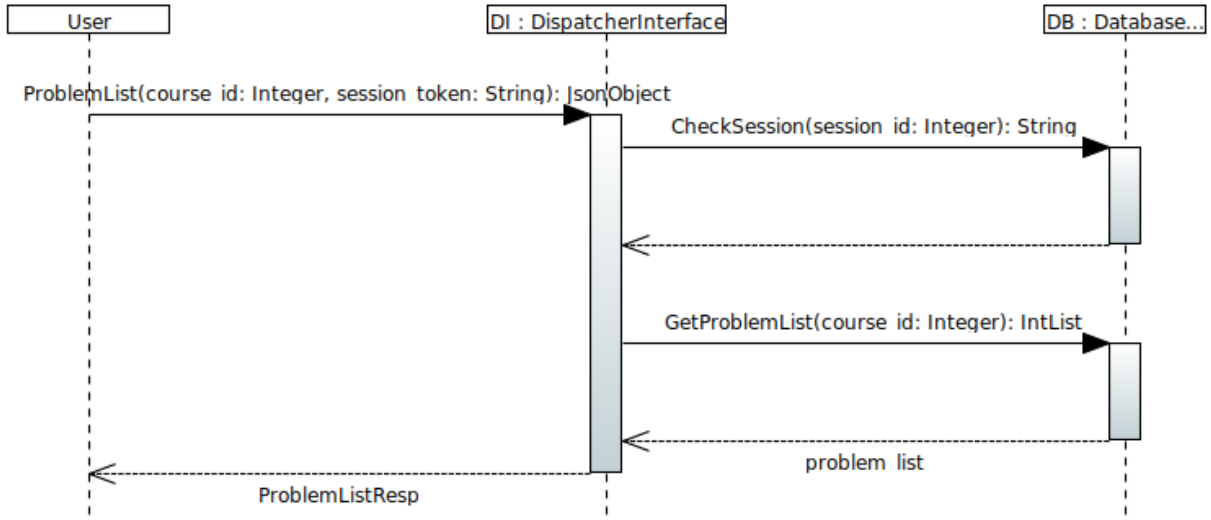
Kasutaja esitab lahenduse. Süsteem kontrollib, et kasutaja on kursusel kirjas ja et lahendus esitatakse olemasolevale ülesandele. Kui tingimus on täidetud, hindab süsteem lahenduse, salvestab tulemuse andmebaasi ning salvestab lahendusfaili failiserverisse (joonis 9).



Joonis 9: Lahenduste esitamise järgnevusskeem

3.1.7 FR7: Ülesannete nimekirja vaatamine

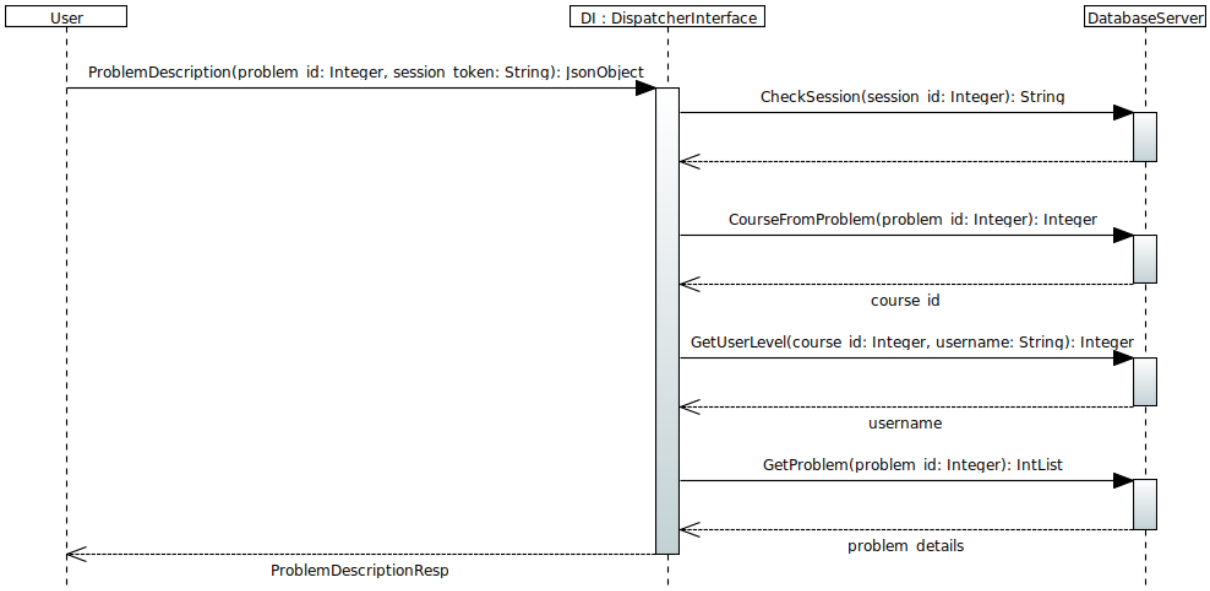
Kasutaja esitab päringu ülesannete nimekirja vaatamiseks. Süsteem kontrollib, kas kasutaja on kursusel kirjas. Kui tingimus on täidetud, võtab süsteem andmebaasist ülesannete identifikaatorid ning edastab need kasutajale (joonis 10).



Joonis 10: Ülesannete nimekirja järgnevusskeem

3.1.8 FR8: Ülesande kirjelduse vaatamine

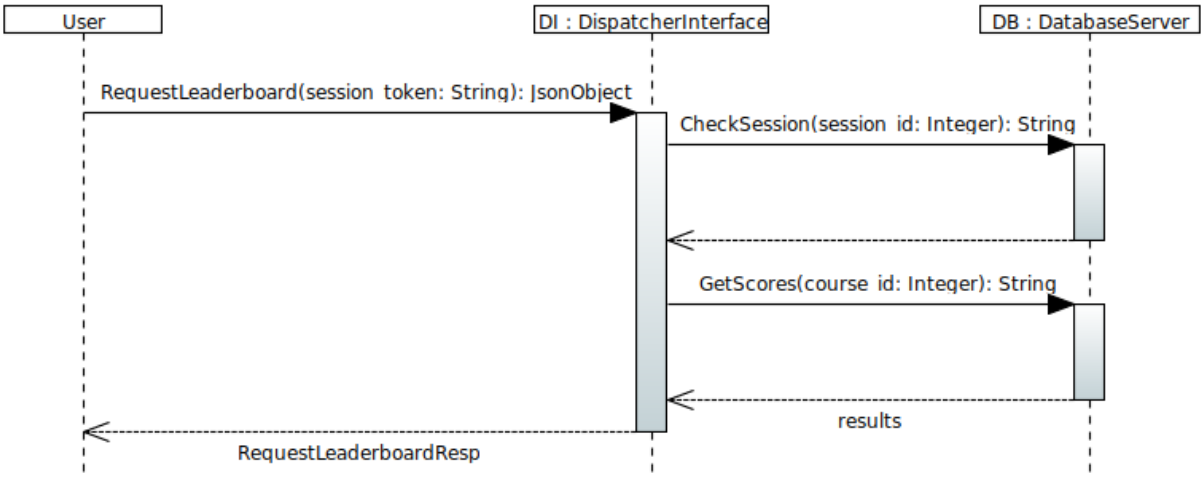
Kasutaja esitab päringu ülesande kirjelduse vaatamiseks. Süsteem kontrollib, kas kasutaja on kursusel kirjas. Kui tingimus on täidetud, võtab süsteem andmebaasist ülesande kirjelduse ning edastab selle kasutajale (joonis 11).



Joonis 11: Ülesande kirjelduse järgnevusskeem

3.1.9 FR9: Edetabeli vaatamine

Kasutaja esitab päringu edetabeli vaatamiseks. Süsteem kontrollib, kas kasutaja on kursusel kirjas. Kui tingimus on täidetud, võtab süsteem andmebaasist tudengite tulemused ning edastab need kasutajale (joonis 12).



Joonis 12: Edetabeli pärimise järgnevusskeem

3.1.10 Mittefunktsioonaalsed nõuded

Lisaks ülaltoodud funktsionaalsetele nõuetele, eeldame, et süsteem on turvaline (ei võimalda ligipääsu kõrvalistele isikutele) ja robustne (käitub ettemääratult ka veaolukordades). Turvalisuse ja robustsuse nõuded on järgnevad.

- NFR1: Süsteemi teenustele saab ligipääsu vaid autenditud kasutaja.
- NFR2: Sisseloginud tavakasutajale näidatakse ainult temaga seotud andmeid (s.o. andmeid kursustest, kuhu ta on registreeritud ja tema enda lahendusi).
- NFR3: Sisseloginud õpetajale näidatakse ainult temaga seotud kursuste andmeid (s.o. andmeid kursustest, kuhu ta on registreeritud ja kõiki kursuse lahendusi).
- NFR4: Kasutajaseanss aegub juhul, kui kasutaja ei ole teinud N sekundi jooksul ühtegi päringut.
- NFR5: Süsteem tagastab veakoodi tundmatu päringu ja vigaste parameetritega päringute puhul.
- NFR6: Süsteem tagastab veakoodi kui üleslaetav fail ületab faili suuruse piirangut.

3.2 Edasilükatud funktsionaalsus

DeepMOOC platvormi üks eesmärke on pakkuda paindlikkust hindamises. Õppejõud peab saama ise otsustada, mida täpselt tudengite lahendustes hinnatakse ja kuidas lõplikes punktides erinevaid hindamise elemente kombineeritakse. Selleks saavad automaattestid iga ülesande kohta väljastada mingi hulga parameetreid, millest hiljem punktid kokku pannakse. Parameetrite kombineerimiseks oli mitu ideed.

Esimene idee oli lubada avaldistes vaid matemaatilisi operatsioone. Õppejõud koostab igale ülesandele sõne kujul malli, millele saab automaattesti väljundid ette anda ning süsteem väärtustab avaldise. See aga seab mõningad piiranguid punktide kombineerimise võimalustele. Näiteks ei saa sellise lahendusega käsitleda teadmata suurusega liste või kombineerida punkte tingimuslikult. Samuti on see on vähem kasutajasõbralik lahendus võrreldes variantidega, mis lubaks mallides ka muutujaid kasutada.

Teine variant oleks lasta õppejõul kirjutada mall mõnes lihtsamas skriptimiskeeles, näiteks Lua või Python ja käivada seda automaattesti väljundite peal. See annaks väga palju valikuvabadust juurde mallide koostamiseks. Samas tekiks võimalus, et mall võib kas tahtlikult või kogemata teha ka muid tegevusi lisaks tulemuste hindamisele. Turvariskide vältimiseks peaks siis leidma ka viisi, kuidas skriptile võimalikult robustsed piirangud seada.

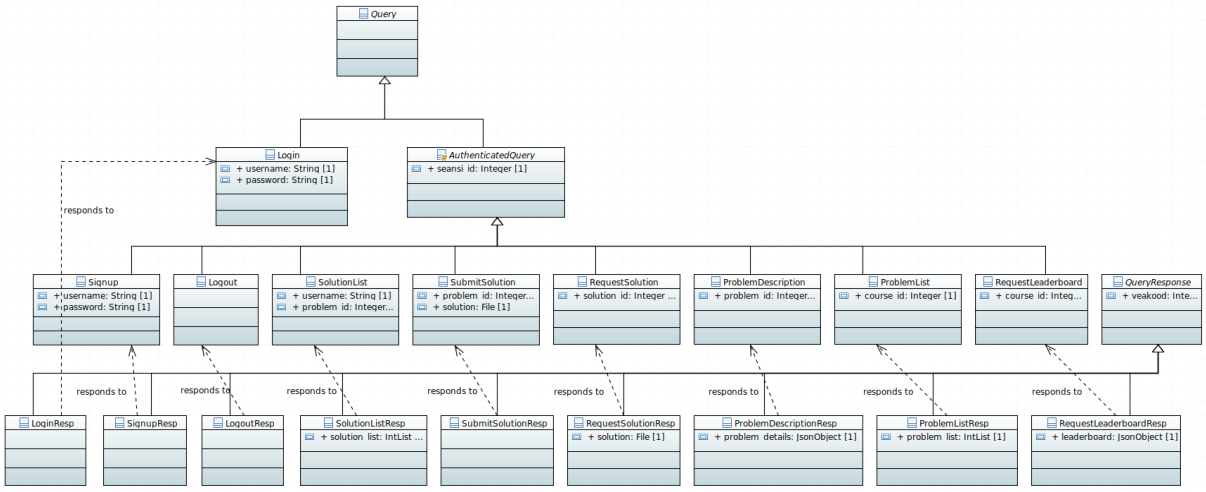
Valituks osutus idee koostada spetsiifiline hindamise keel, milles saab luua malle punktide kombineerimiseks. Sellises keeles saaks ühest küljest võimaldada keerukamate programmide koostamist hinde arvutamiseks ehk kaoks kitsendus, et hindamisreegel peaks olema ühe avaldisega väljendatav. Samas oleks garanteeritud, et programm süsteemis andmeid ei muudaks või muud pahandust teeks. Keele loomisega tegeleb paralleelselt valmiv lõputöö. Hindamise osa pooleliolevast arengust tingituna on dispetšeris ka täielikult realiseerimata nii lahenduste esitamise kui edetabeli kuvamise funktsionaalsus.

Kasutajaõiguste haldamiseks on hetkel ainus võimalus süsteemisiseselt luua kasutaja, mis pole ühegi välise süsteemiga seotud. Nii kausutusmugavuse kui turvalisuse seisukohast oleks parem, kui autentimine toimuks läbi mõne laiemalt levinud autentimismehhanismi. Ideaalis võiks kasutaja olla seotud olemasoleva ülikooli kasutajatunnusega. Kaoks vajadus kasutajatunnuste tekitamiseks ja paroolide haldamiseks. Samuti saaks ÕIS'is kursusele registreerunud korraga DeepMOOC vastava kursusega siduda. Tulevikus on plaan sisselogimine üle viia sama süsteemi peale, mida kasutavad Moodle ja ÕIS.

Hetkel on rakendusest puudu osa süsteemi haldamisega seotud funktsionaalsusest, näiteks uute ülesannete ülesseadmine. Puuduvad osad realiseeritakse edaspidi DeepMOOC platvormi arenduse käigus.

3.3 Sõnumid ja nende vastused

Igale funktsionaalsele nõudele vastab päring-vastus paar nii nagu näidatud joonisel 13.

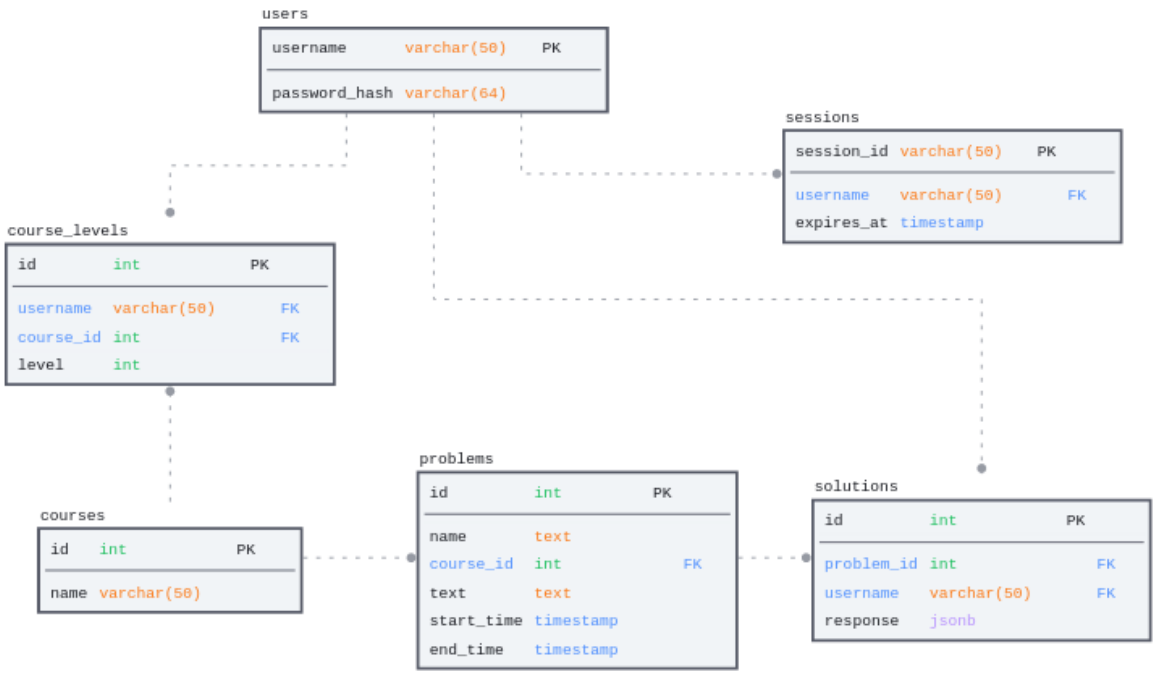


Joonis 13: Dispetšeri poolt aktsepteeritavad sõnumid

3.4 Andmebaasiskeem

Rakenduse andmebaas koosneb kuuest erinevast tabelist: (joonis 14)

- Users - Hoiab kasutajate kasutajanimesid ja salasõnade räsisid.
- Sessions - Hoiab kasutajate seansiidentifikaatoreid.
- Courses - Hoiab erinevaid kursuseid ja nende nimesid.
- Course_levels - Säilitab kasutaja õiguste tasemeid iga kursuse jaoks.
- Problems - Hoiab ülesandeid ja erinevaid detaile ülesannete kohta.
- Solutions - Hoiab esitatud lahenduste identifikaatoreid ja tulemusi.



Joonis 14: Andmebaasi struktuur

3.5 Autentimine ja ligipääs andmetele

Kasutaja autentimine süsteemis toimub kasutajanime ja salasõna alusel. Seansi alustamiseks on eraldi päring (Login), millega tekitatakse seansi identifikaator. Kõigi järgnevate päringute puhul on üheks nõutud parameetriks seansi identifikaator. Identifikaator salvestatakse seansi alustamisel andmebaasi koos aegumisajaga. Edaspidi annab eesrakendus iga päringuga seansi identifikaatori kaasa ning tagarakendus kontrollib andmebaasist, et identifikaator aegunud ei oleks, ning saab andmebaasist ka kätte päringu esitaja kasutajanime ja õiguste taseme. Iga uus

päring nihutab seansi aegumise tähtaega andmebaasis edasi.

Kasutajatele on igal kursusel, millega nad seotud on, määratud õiguste tase. Võimalikud tasemed on tudeng, õppejõud ning admin. Tudengi õigustega kasutaja saab ligi vaid oma lahendustele ning jooksvatele ja läbi saanud ülesannetele. Õppejõud saavad ligi kõigile ülesannetele ning kõigi tudengite lahendusi vaadata, samuti saavad õppejõud hallata kursusel osalevaid tudengeid. Admini õigustega kasutajad saavad hallata kursustel olevaid õppejõude. Kõik andmebaasipäringud on koostatud nii, et nad tagastavad andmeid vaid seansiga seotud kasutajale või kõrgemate õigustega kasutajale.

3.6 Veatöötlus

Kui eesrakendusest saadetakse päring, mille töötlemine ebaõnnestub, tagastatakse vastav veakood. Veakoodid ning nende tähendused on valitud HTTP olekukoodide hulgast. [12] Võimalikud veakoodid, mida tagastatakse on:

- 400 ehk *bad request* juhul, kui päringust on mõni parameeter puudu;
- 401 ehk *unauthorized* juhul, kui päringuga ei ole kaasas seansiidentifikaatorit;
- 403 ehk *forbidden* juhul, kui päringu saatjal puuduvad õigused päringu tegemiseks;
- 413 ehk *request entity too large* juhul, kui päringuga saadetakse fail, mis on liiga suur;
- 500 ehk *internal server error* juhul, kui tekib erind kohas, kus teadaolevalt ei tohiks erind tekkida.

4 Verifitseerimine

4.1 Testijuhtumid

Rakenduse funktsionaalsed nõuded on üksteisest vähesõltuvad, seetõttu on lihtne koostada testijuhtumid funtksionaalsete nõuete kaupa. Testide organiseerimisel on oluline meeles pidada, et kõigi päringute puhul tuleb kasutaja kõigepealt autentida (kõik ülejäänud nõuded sõltuvad FR1'st). Iga nõude testijuhud peavad katma järgmised stsenaariumid:

- Edukas käivitus – teenusele antakse korrektne sisend ja kontrollitakse, et tagastatakse oodatud vastus.
- Autoriseerimata kävitus – päring kutsutakse välja korrektse sisendiga aga ilma kehtiva seansiidentifikaatorita. See stsenaarium kontrollib MFR1 järgimist antud päringus.
- Vigane käivitus – kasutaja on korrektselt autenditud aga päringu sisendsõnum on vigane. Kontrollib MFR4 rahuldatust antud päringu korral.

Sõltuvalt päringu sisust võib vaja olla konstrueerida mitu testi eduka käivituse ja/või vigase käivituse stsenaariumitele.

4.1.1 FR1: Alusta seanssi

Seansi alustamise teenuse katavad järgmised testijuhtumid:

ID	Nimi	Liik	Kirjeldus
FR1.TC1	Edukas sisselogimine	Edukas	Kasutaja sisselogimine õnnestub
FR1.TC2	Teistkordne sisselogimine	Edukas	Sama kasutaja logib teist korda sisse
FR1.TC3	Tundmatu kasutaja	Autoriseerimata	Kasutaja, keda andmebaasis pole, püüab sisse logida
FR1.TC4	Vigane parool	Autoriseerimata	Kasutaja on andmebaasis aga parool on vale

4.1.2 FR2: Lõpeta seanss

Seansi lõpetamise teenuse katavad järgmised testijuhtumid:

ID	Nimi	Liik	Kirjeldus
FR2.TC1	Edukas lõpetamine	Edukas	Lõpetatakse olemasolevat seansi
FR2.TC2	Teistkordne lõpetamine	Autoriseerimata	Sama seansi lõpetamiseks saadetakse kaks järjestikust päringut
FR2.TC3	Puuduv seanss	Autoriseerimata	Päringus puudub seansi ID
FR2.TC4	Tundmatu seanss	Vigane	Saadetakse tundmatu seansi ID
FR2.TC5	Aegunud seanss	Autoriseerimata	Saadetakse aegunud seansi ID

4.1.3 FR3: Kasutaja loomine

Kasutaja loomise teenuse katavad järgmised testijuhtumid:

ID	Nimi	Liik	Kirjeldus
FR3.TC1	Edukas loomine	Edukas	Esitati unikaalne kasutajanimi ja nõuetekohane parool
FR3.TC2	Dubleeritud kasutajanimi	Vigane	Saadetakse kasutajanimi, mis andmebaasis puudub
FR3.TC3	Puuduv kasutajanimi	Vigane	Päringus pole kasutajanime
FR3.TC4	Puuduv parool	Vigane	Päringus pole parooli

4.1.4 FR4: Lahenduste nimekirja vaatamine

Lahenduste nimekirja vaatamise teenuse katavad järgmised testijuhtumid:

ID	Nimi	Liik	Kirjeldus
FR4.TC1	Nimekiri tavakasutajale	Edukas	Tavakasutaja küsis oma lahendusi
FR4.TC2	Nimekiri õppejõule	Edukas	Õppejõud küsis etteantud kasutaja lahendusi
FR4.TC3	Tühi nimekiri	Edukas	Kasutaja küsis oma lahendusi, ühtegi ei leitud
FR4.TC4	Puuduv seanss	Autoriseerimata	Päringus puudub seansi ID
FR4.TC5	Vale kasutaja	Autoriseerimata	Kasutaja küsib teise kasutaja lahendusi
FR4.TC6	Puuduv parameeter	Vigane	Päringus puudub kasutajanimi

4.1.5 FR5: Lahenduse vaatamine

Lahenduse vaatamise teenuse katavad järgmised testijuhtumid:

ID	Nimi	Liik	Kirjeldus
FR5.TC1	Oma lahendus	Edukas	Tavakasutaja küsis oma lahendust
FR5.TC2	Lahendus õppejõule	Edukas	Õppejõud küsis etteantud kasutaja lahendust
FR5.TC3	Puuduv seanss	Autoriseerimata	Päringus puudub seansi ID
FR5.TC4	Vale kasutaja	Autoriseerimata	Kasutaja küsib teise kasutaja lahendust
FR5.TC5	Puuduv lahendus	Vigane	Küsitud lahenduse ID-d ei leita

4.1.6 FR6: Lahenduse esitamine

Lahenduse esitamise teenuse katavad järgmised testijuhtumid:

ID	Nimi	Liik	Kirjeldus
FR6.TC1	Lahendus vastu võetud	Edukas	Kasutaja saatis lahenduse, see salvestati
FR6.TC2	Puuduv seanss	Autoriseerimata	Päringus puudub seansi ID
FR6.TC3	Puuduv lahenduse fail	Vigane	Saadetud pärignus pole faili
FR6.TC4	Fail liiga suur	Vigane	Päringus sisalduv fail on liiga suur

4.1.7 FR7: Ülesannete nimekirja vaatamine

Ülesannete nimekirja vaatamise teenuse katavad järgmised testijuhtumid:

ID	Nimi	Liik	Kirjeldus
FR7.TC1	Nimekiri tavakasutajale	Edukas	Tavakasutaja küsis ülesandeid
FR7.TC2	Nimekiri õppejõule	Edukas	Õppejõud küsis ülesandeid
FR7.TC3	Tühi nimekiri	Edukas	Kasutaja küsis ülesandeid, ühtegi ei leitud
FR7.TC4	Puuduv seanss	Autoriseerimata	Päringus puudub seansi ID
FR7.TC5	Vale kursus	Autoriseerimata	Kasutaja küsib ülesandeid kursusest, kus ta pole registreeritud

4.1.8 FR8: Ülesande kirjelduse vaatamine

Ülesande kirjelduse vaatamise teenuse katavad järgmised testijuhtumid:

ID	Nimi	Liik	Kirjeldus
FR7.TC1	Kirjeldus tavakasutajale	Edukas	Tavakasutaja küsis kirjeldust
FR7.TC2	Kirjeldus õppejõule	Edukas	Õppejõud küsis kirjeldust
FR7.TC3	Puuduv seanss	Autoriseerimata	Päringus puudub seansi ID
FR7.TC4	Vale kursus	Autoriseerimata	Kasutaja küsib ülesande kirjeldust kursusest, kus ta pole registreeritud

4.1.9 FR9: Edetabeli vaatamine

Edetabeli vaatamise teenuse katavad järgmised testijuhtumid:

ID	Nimi	Liik	Kirjeldus
FR7.TC1	Edetabel tavakasutajale	Edukas	Tavakasutaja küsis edetabelit
FR7.TC2	Edetabel õppejõule	Edukas	Õppejõud küsis edetabelit
FR7.TC3	Puuduv seanss	Autoriseerimata	Päringus puudub seansi ID
FR7.TC4	Vale kursus	Autoriseerimata	Kasutaja küsib edetabelit kursusest, kus ta pole registreeritud

4.2 Testiprotseduurid

Testimiseks kasutatakse fikseeritud sisuga testandmebaasi. Testid viiakse läbi käsitsi vastavalt testijuhtumites kirjeldatud tingimustele, tulemust kontrollib testija samuti käsitsi. Süsteemi hallatavuse huvides oleks mõistlik testiprotseduurid tulevikus realiseerida automaatsete ühiktestidena.

Testimiseks tuleb kõigepealt üles seada andmebaas kaasasoleva *sampleDB.sql* skripti põhjal. Testipaketis on failis *testid.md* iga testijuhtumi kohta kirjas päring, millega test läbi viia. Sama fail sisaldab ka kirjeldust, millised muutused keskkonnas (teenuse tagastatud andmed, andmed andmebaasis ja failiserveris) peaksid olema peale päringu jooksutamist toimunud.

Kokkuvõte

Bakalaureusetöö eesmärk oli luua DeepMOOC platvormi tagarakendusele dispetšer, mis korraldaks suhtlust eesrakenduse, konveieri ja andmebaasi vahel. Dispetšer valmis mõningate puudujääkidega, dispetšerist on hetkel puudu edetabeli kuvamise funktsionaalsus ja lahenduste hindamine on täielikult realiseerimata. Lisaks jäi esialgselt skoobist välja eesrakenduse päringute kaudu kasutajate ja süsteemi haldamine. Kogu DeepMOOC platvormist on praeguseks valmis konveier, mille arendamist kirjeldab lõputöö [2], puudu on platvormi eesrakendus ning hetkel on arendamisel hindamiskeel automaattestide tulemustest punktide moodustamiseks.

Töö teoreetilises osas kirjeldati alternatiivseid olemasolevaid platvorme ning seletati arenduses kasutatud tarkvarasid. Tööprotsessi peatükis kirjeldati arenduse käiku ning arenduses tekkinud probleeme ja nende lahendusi. Projekteerimise peatükis kirjeldati rakenduse tööks vajalikke nõudeid ja süsteemi arhitektuuri. Verifitseerimise peatükis seletatakse nõuete testimiseks vajalikke testijuhtumeid ning üldist testiprotseduuri.

Kirjandus

[1] VPL, the Virtual Programming lab for Moodle. https://vpl.dis.ulpgc.es. (28.07.2022)

[2] Anijärv, A, DeepMOOC platvormile tarkvarakonveieri arendamine, Bakalaureusetöö, Tartu Ülikool 2022

[3] Artemis: Interactive Learning with Individual Feedback. https://docs.artemis.ase.in.tum.de (03.08.2022)

[4] Go. https://go.dev/solutions/webdev (29.07.2022)

[5] Why Developers Use Go Language for Web Servers. https://www.zco.com/blog/why-developers-use-go-language-for-web-servers/ (29.07.2022)

[6] PostgreSQL. https://www.postgresql.org/about/ (29.07.2022)

[7] Amazon S3. https://aws.amazon.com/s3/ (29.07.2022)

[8] Go http package documentation. https://pkg.go.dev/net/http (05.08.2022)

[9] PostgreSQL Identity Column. https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-identity-column/ (05.08.2022)

[10] Making and Using HTTP Middleware. https://www.alexedwards.net/blog/making-and-using-middleware (05.08.2022)

[11] AWS SDK for Go API Reference. https://docs.aws.amazon.com/sdk-for-go/api/service/s3/ (05.08.2022)

[12] HTTP response status codes. https://developer.mozilla.org/en-US/docs/Web/HTTP/Status (05.08.2022)

Lisad

I. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Joosep Näks**,
*(*autori nimi)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   DeepMOOC platvormile tagarakenduse dispetšeri arendamine,

   mille juhendajad on Ahti Põder ja Tõnis Hendrik Hlebnikov

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Joosep Näks
8.08.2022