TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Roald Välja 142332IAPB

LÕPUTÖÖDE KAITSMISTE AJAKAVA KOOSTAMINE KASUTADES OPTAPLANNERIT

Bakalaureusetöö

Juhendaja: Riina Maigre
Doktorikraad

Tallinn 2021

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Roald Välja

25.05.2021

Annotatsioon

Antud lõputöös üritatakse lahendada lõputööde kaitsmiste ajakava koostamise ülesannet. Lõputöö kaitsmise ajakavasid koostatakse käsitsi Excelis, mistõttu ei ole lihtne arvestada näiteks kaitsjate soove ja võimalusi ning hilisem muudatuste sisseviimine on keeruline.

Lõputöö eesmärk oli luua kaitsmiste ajakava planeerija prototüüp kasutades OptaPlanner't, mis võtab arvesse lõppkasutaja poolt valitud kitsendusi, nende kaalud ja võimaldab määrata, kui kaua planeerija planeerib ajakava. Planeerijale lisaks loodi veebiliides, mille kaudu saab lõppkasutaja sisestada sisendandmeid vigadeta, määrata kitsenduste kaalusid ja käivitada planeerijat.

Töö tulemusena saadi valmis tarkvara prototüüp, mis suudab lahendada ajakava planeerimise osa automaatselt. Loodud tarkvara võiks lihtsustada ja kiirendada lõppkasutaja tööd lõputööde kaitsmiste ajakava planeerimisel. Tarkvara võimaldab kiiret ümber planeerimist, juhul kui tulemus kasutajat ei rahulda.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 31 leheküljel, 6 peatükki, 6 joonist, 1 tabelit.

Abstract

Scheduling Thesis Defenses Using OptaPlanner

The aim of this bachelor thesis work was to develop a prototype application for university thesis defense schedulers, that would make scheduling an easier and quicker process. The author used OptaPlanner planning engine to handle optimization algorithms for scheduling.

Usually thesis defense schedules are done manually using Excel, which is a significantly long and arduous work. As the planning is done manually, it is not possible to take into account the preferences of defenders or make significant subsequent changes to the schedule. As a result, the scheduler cannot accept a lot of requests made by the students or the supervisors.

As a result of this thesis work, the author has created a prototype application that can handle scheduling automatically, which can also be configured by the end user. The author has also created a web interface, where the end user can insert the required input data without errors and configure the planner by choosing how long to plan the schedule and what restrictions to consider.

The created prototype is designed to simplify the process of creating schedules for thesis defenses and could increase the speed in which the scheduling part is done. The application also allows for quick rescheduling in case the result is not satisfactory.

The thesis is in Estonian and contains 31 pages of text, 6 chapters, 6 figures, 1 tables.

Lühendite ja mõistete sõnastik

XML	Extensive Markup Language, laiendatav märgistuskeel
JSON	JavaScript Object Notation
HTML	Hypertext Markup Language, hüperteksti märgistuskeel
CSS	Cascading Style Sheets, kaskaadlaadistik
HTTP	Hypertext Transfer Protocol, Hüperteksti edastusprotokoll
POST	HTTP meetod andmete postitamiseks
GET	HTTP meetod andmete taotlemiseks
WAR	Web Application Resource
JSP	Java Server Page
AJAX	Asyncronous JavaScript And XML

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Lõputööde ajakavade planeerimine on keeruline ja aegavõttev tegevus, eriti kui seda tehakse käsitsi. Planeerimise lihtsustamiseks on olemas automaatseid planeerijaid, aga lähteülesanne, sisendid ja väljundid tuleb ikka iga probleemi jaoks eraldi ära kirjeldada ja visualiseerida.

Kahjuks pole TalTechis olemas automaatset planeerijat, millega koostada lõputööde kaitsmiste ajakava, näiteks saja kaitsja jaoks. Lõputööde kaitsmiste ajakava koostatakse käsitsi, mille jaoks kulub palju aega ning pole lihtne koostada plaane, kus võetakse arvesse palju eelistusi ja nõudeid. Seetõttu pole lõputööde autoritel, juhendajatel ega komisjoni liikmetel võimalik anda tagasisidet või teatada sobimatutest või eelistatud aegadest. Sellised võimalused on erandkorralised.

Antud töö eesmärk oli luua rakendus lõputööde kaitsmiste ajakava koostamise lihtsustamiseks, pakkudes välja võimalikult täpselt nõudeid rahuldava ajakava lahendusi. Rakendus annab kasutajale lisaks liidese, mille abil saab koostada planeerijale vigadeta sisendandmeid.

Teises peatükis teeb autor ülevaate kasutatavatest tehnoloogiatest ja selgitab, miks valiti just need. Kolmandas peatükis analüüsib autor planeerija nõudeid ja kitsendusi, mis on eeldused planeerija koostamiseks. Selles peatükis tuuakse välja ka nõuete täpsemaks väljaselgitamiseks läbi viidud intervjuu vastused. Neljandas peatükis annab autor ülevaate realiseeritud rakenduse prototüübist, mida iga osa teeb ja kuidas nad on omavahel seotud. Selles peatükis vaadatakse üle ka mida on prototüübis testitud. Viiendas peatükis kontrollitakse loodud rakenduse prototüübi tulemusi ning antakse hinnang kasutatud planeerimismootorile.

Rakenduse repositoorium asub aadressil: https://github.com/RoaldValja/iaib

2 Taust

Enne planeerimismootori valimist võrreldi erinevaid planeerijaid, et leida kõige sobivam lõputöös kasutamiseks. OptaPlanner sai valitud, kuna see sobis autori programmeerimiskeele valikuga ning sellega sai teada ka Drools'i olemasolust ja kasust.

2.1 OptaPlanner

OptaPlanner on kergekaaluline, üldotstarbeline, manustatav kitsenduste rahuldamise mootor, mis optimeerib planeerimise probleeme. OptaPlanner'i kasutamine võimaldab kasutajal kasutada planeerimisel suuremat hulka andmeid ja keerulisemaid kitsendusi, selle asemel, et käsitsi ise planeerida. OptaPlanner'ga on võimalik leida planeerimise probleemile hea lahendus mõistliku ajaga. Kitsendusi on võimalik kirjutada objektorienteeritud keeles või Drools'i reeglitena. OptaPlanner toetab mitmeid optimeerimise algoritme, millega saab tõhusalt läbi proovida suurt hulka võimalikke lahendusi. Kuna iga optimeerimise algoritm sobib erinevatele kasutusjuhtudele, pole alguses alati võimalik teada millist algoritmi peab oma kasutusjuhul kasutama. OptaPlanner lubab väga lihtsalt vahetada optimeerimise algoritmi, muutes ainult mõne rea koodi või XML-i (Extensive Markup Language) [2] .

OptaPlanner'ga plaanitava probleemi lahendamiseks tuleb läbi teha neli sammu. Esiteks tuleb ära modelleerida planeerimise ülesanne. Teiseks tuleb ära konfigureerida kuidas hakatakse probleemi lahendama. Näiteks millist optimeerimise algoritmi kasutada. Järgmiseks tuleb laadida lahendajasse probleemi andmed, mis on planeerimise probleem. Lõpuks tuleb lahendaja käima panna, et leida ära määratud aja jooksul parim lahendus. [3]

OptaPlanner suudab ise otsida parimat lahendust kasutades kasutaja poolt pandud reeglite punktide pealt saadud lahenduse skoori. OptaPlanner't kasutades pole vaja ise

luua lahenduse leidmiseks algoritmi, vaid tuleb OptaPlanner'le öelda millist algoritmi tuleks kasutada.

2.2 Drools

Drools on ärireeglite juhtimise süsteem koos ettepoole- ja tahapoole aheldamise järelduspõhise reeglite mootoriga, mis lubab kiirelt ja töökindlalt hinnata ärireegleid ning komplekssseid sündmuse protsesse [13] . Drools'i kasutatakse töös sellepärast, et reeglid oleks kirjutatud eraldi isoleeritud osadena. Droolsi reeglid on kirjutatud Drools'i keeles ja väikeste juppidena, mida on lihtne ükshaaval hallata, millel saab skoori inkrementaalselt kalkuleerida ja mida on lihtne läbi testida.

Drools'i reegli näide on toodud joonisel (Joonis 1). See reegel kontrollib, et kas `timeslot` objekt on kaitsmise autori jaoks sobimatu ja kas `timeslot` pole `null` objekt. Kui reegli tingimusi läbitakse, siis skoorist võetakse tugevaid punkte maha.

```
package packageName.solver;
       dialect "java"
import org.optaplanner.core.api.score.buildin.hardsoft.HardSoftScoreHolder;
import thesistimetableplanning.domain.Defense;
global HardSoftScoreHolder scoreHolder;
rule "Defense not on authors unavailable timeslot"
    when
        $defense : Defense(isAuthorsUnavailableTimeslot(),
                    timeslot != null)
    then
        scoreHolder.addHardConstraintMatch(kcontext, -1);
end
```

Joonis 1. Drools'i lihtne reegli näide

Drools'i reeglid kirjutatakse eraldi failidesse, mille nimi lõppeb laiendusega `.drl`. Neid reegleid võib kirjutada ükshaaval failidesse või grupeerida neid kokku ühte või mitmesse `.drl` faili.

Igal `.drl` faili alguses peab olema kirjas paketi nimi, kus antud reeglifail asub, võtmesõna `dialect` ja jutumärkides programmeerimise keel, mida kasutad ja globaalne muutuja, kus hoiad oma reeglite skoori.

Iga reegel kasutab nelja võtmesõna: `rule`, `when`, `then` ja `end`. Võtmesõna `rule` ette kirjutatakse kitsenduse täpne nimi, mis peab olema täpselt sama kitsenduste failis olev reegli nimi, et reegel teaks missuguse reeglitüübiga on tegemist ja mitu punkti mõjutab antud reegel skoori. `When` klauslisse kirjutatakse reegli täitmiseks olevad tingimused ja `then` klauslisse kirjutatakse mida tehakse siis, kui tingimused said täidetud. `End` võtmesõnaga märgistatakse ära reegli lõpp.

2.3 Sarnased projektid

Planeerimise probleemi lahendamiseks on vaja mingisugust kitsenduste lahendamise mootorit, mis teeb programmi loomise lihtsamaks. Kitsenduste lahendajate seas oli olemas mitu erinevat valikut, mille hulgast välja paistsid kõige rohkem OptaPlanner, Minion, Koalog, ja JaCoP. [17]

Minion on avatud lähtekoodiga C++ keeles kirjutatud kitsenduste lahendaja. Seda lahendajat ei valitud, kuna autoril on vähe kogemust C++ keelega. [16] Koalog on kommertslik Java keeles kirjutatud kitsenduste lahendaja. Koalog lahendajat ei valitud, kuna eelistati vabavaralist tarkvara ja Koalog on vananenud lahendaja, mille kodulehte pole enam üleval. [15] JaCoP lahendaja on avatud lähtekoodiga Java keeles kirjutatud kitsenduste lahendaja. JaCoP lahendajat ei valitud, kuna dokumentatsioonis puudusid näidislahendused ja arusaamiseks nõutakse olemasolevaid teadmisi kitsenduste programmeerimisest. [14] OptaPlanner on avatud lähtekoodiga Java keeles kirjutatud kitsenduste lahendaja. OptaPlanner valiti sellepärast, et OptaPlanner’t saab kasutada Java koodis, dokumentatsioon on algajasõbralik ja sisaldab näidislahendusi. [2]

OptaPlanner’ga on loodud mitmeid erinevaid lahendusi erinevatele planeerimise probleemidele. Kõige sarnasemad lahendused lõputööde kaitsmiste ajakava koostamisele oleks näiteks konverentsi planeerimine ja kursuste ajakava koostamine. Need lahendused aga ei sobi täpselt lõputööde kaitsmiste ajakava planeerimiseks. Peale neid näiteid pole autor leidnud sarnaseid lahendusi ajakavade koostamiseks. Autori lahendus erineb teistest lahendustest sellepoolest, et see lahendus on kirjutatud spetsiaalselt lõputööde kaitsmiste ajakava loomiseks.

3 Planeerija analüüs

Planeerija analüüsiks loodi esmalt klassidiagramm planeerijale olulistest andmetest. Probleemispetsiifiliste kitsenduste täpsemaks väljaselgitamiseks viidi läbi intervjuu võimaliku lõppkasutajaga, mille põhjal koostati lõputöö kaitsmiste ajakava planeerimise kitsendused.

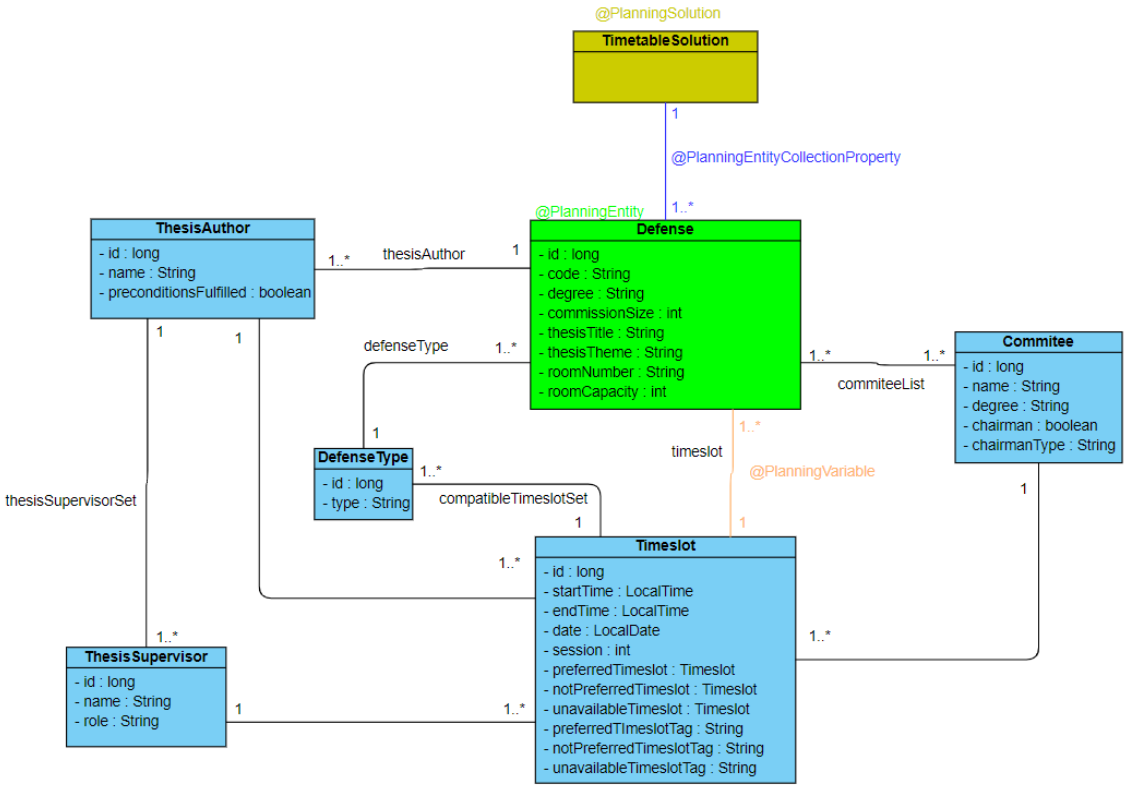
3.1 Planeerija

Planeerija on Java keeles kirjutatud kood, mille ülesanne on leida planeerimise probleemi jaoks lahendusi. Planeerija on selleks tehtud, et automatiseerida kasutaja jaoks lõputööde kaitsmise ajaplaani koostamist, kus kasutaja peab ainult looma probleemi jaoks sisendandmed.

Planeerija koosneb mitmest erinevast osast. Planeerija saab endale planeerimise probleemi sisendandmed kätte JSON (JavaScript Object Notation) kujul failist. Antud andmed jaotatakse laiali domeenimudeli objektidesse, mida kasutatakse Drools'i reeglitega kirjeldatud kitsenduste kontrollimiseks. OptaPlanner'i optimeerimise algoritme kasutades luuakse Drools'i reeglitega probleemi jaoks nii palju erinevaid lahendusi, kui on võimalik kasutaja poolt antud aja jooksul. OptaPlanner valib skoori järgi välja parima lahenduse, mis saadetakse JSON kujul tagasi veebiliidesesse ja eraldi Exceli faili.

3.2 Klassidiagramm

Planeerija domeenimudeli visualiseerimiseks on loodud klassidiagramm, mis on näidatud joonisel (Joonis 2).



Joonis 2: Planeerija klassidiagramm

Planeerija domeenimudel koosneb seitsmest klassist. `Defense` klass on domeenimudeli planeerimise üksus, millele on juurde pandud annotatsioon `@PlanningEntity`. `Defense` klassis on planeeritavaks muutujaks lisatud `timeslot`, mis on objekt klassist `Timeslot` ning millele on lisatud annotatsioon `@PlanningVariable`.

`Defense` klass hoiab veel muutujaid nimega `commiteeList`, mis hoiab nimekirja kõikidest komisjoni liikmetest ning kasutab objektiks klassi `Commitee`. `Defense` klassis on muutuja `defenseType`, mis hoiab objektiks klassi `DefenseType`. `Defense` klassis on viimaseks muutuja `thesisAuthor`, mis hoiab objektiks klassi `ThesisAuthor`. `ThesisAuthor` klass hoiab muutujat nimega `thesisSupervisorSet`, kus on üks või mitu `ThesisSupervisor` objekti.

`DefenseType` klass märgistab ära, kas kaitsmine on lahtine või kinnine. `DefenseType` klassis on veel muutuja `compatibleTimeslotSet`, mis hoiab `Timeslot` objekte, mille järgi jaotatakse `Timeslot` objektid ära, kas nad toimuvad lahtise kaitsmisena või kinnisena.

`Commitee`, `ThesisAuthor` ja `ThesisSupervisor` klassides on muutujad `preferredTimeslot`, `notPreferredTimeslot` ja `unavailableTimeslot`, mis hoiavad `Timeslot` objekte. Nende muutujate järgi teab, milliseid aegu eelistatakse, ei eelistata ning millised ajad üldse neile ei sobi.

`TimetableSolution` klass hoiab endas nimekirju probleemfaktidena klassidest `Commitee`, `DefenseType`, `ThesisAuthor`, `ThesisSupervisor` ja `Timeslot` ning nendele on juurde lisatud annotatsioon `@ProblemFactCollectionProperty`. `TimeslotSolution` hoiab ühte planeerimise üksuse nimekirja, mille klassiks on `Defense`. Sellele nimekirjale on lisatud annotatsioon `@PlanningEntityCollectionProperty`. `TimetableSolution` klassil endal on lisatud annotatsioon `@PlanningSolution`.

3.3 Intervjuu

Projekti kitsenduste täiendamiseks ja andmete täpsemate nõuete väljaselgitamiseks viidi läbi intervjuu Monika Kreininiga 24. aprillil 2019. Intervjuu kirjutan töös lahti järgmisena: esimesena esitan küsimuse, siis annan teada mis vastuse sain ning peale seda kirjutan paar märkust, kui selle küsimuse kohta midagi tekkis.

Küsimus 1: Kuidas lõputöö kaitsmiste ajakava koostamine praegu käib? Millised on selle koostamise sammud?

Vastus: On ette antud päevade arv, kuhu pean ära jaotama tudengite tööd. Igaühele on ette nähtud mingi aeg minutites. Ma hakkan lihtsalt neid panema ajakavasse, aga üritan panna ühe sama juhendaja juhendatavad järjestikku. Siis kui kaitsja toob oma lõputöö ära, tal lastakse kirja panna mingi kuupäev, millal talle kindlasti ei sobi kaitsta. Tabelit tehes üritan arvestada kaitsjate soovidega.

Küsimus 2: Kas Te kasutate mingit programmi, et automatiseerida ajakava koostamist?

Vastus: Ei kasuta. Excelis panen käsitsi.

Küsimus 3: Kas Te võtate kaitsjate ja juhendajate soove arvesse, kui koostate kaitsmiste ajakavasid? Kui jah, siis millised soovid võetakse vastu?

Vastus: Juhendajatel pole soove. Juhendajad otse ei tee sooviavaldusi. Ma panen ajakava kokku ja saadan selle laiali juhendajatele. Siis kui juhendajale kindlasti ei sobi see ajakava, siis ma üritan seda ümber panna enne kui avaldan tudengitele. Ma üritan luua ajakava, kus võimalikult palju juhendajaid saavad tulla.

Märkused: Planeerijas on võimalik kirja panna juhendajate soovid. Lõppkasutaja võib seda kasutada või mitte.

Küsimus 4: Mida Te veel võtate arvesse, kui koostate kaitsjate ajakava?

Vastus: Igal päeval on ajakavas 3-4 sessiooni, kus vahepeal on vaheaeg. Sessiooni ajal ei liigu keegi sisse ega välja. On näiteks 10 inimest, nad tulevad sisse, ootavad ära kõik 10 kaitsjat, siis tuleb vaheaeg, siis tuleb sisse järgmised 10. Vaheajad on mõistliku ajapikkusega.

Küsimus 5: Sellel hetkel ma näitan Monika Kreininile kirja pandud kitsendused ja küsin nende kohta tagasisidet.

Vastus: Ma panen käsitsi, kas mingi tudeng on eeldused ära täitnud või mitte. Ma sooviks, et seda saaks teha automaatselt. Terve komisjon ei ole koguaeg kohal, miinimum on 3 või 4 liiget. Komisjon on suurem tehtud, et nad saaksid roteeruda. Ma teen tavaliselt Doodle komisjoni liikmetele, kus nad panevad kirja, millal nad kindlasti ei saa tulla. Võiks panna sessioonide järgi, millal komisjoni liikmed saaksid roteeruda. Komisjoni liikmed saavad vaikselt sessiooni ajal lahkuda, aga oleks parem, kui nad lahkuksid sessiooni lõpus. Kohal peab olema esimees või aseesimees. Keel ei ole oluline. Sama tüüpi lõputöid ei vaata, aga kui satub, et teemad on sarnased, siis üritan neid panna lähestikku. Kinnised kaitsmised on tavaliselt päeva lõpus. Siis kui teised on ära kaitsnud, siis tulevad kinnised kaitsmised. Võib juhtuda, et on jube raske teha ajakava, siis viimasel juhul panen päeva algusesse, muidu esimese valikuna panen kõige lõppu. Ma ei lase tudengitel eelistada, kuna muidu muutub ajakava koostamine liiga keeruliseks. Ma ei taha, et ajakavasse jäävad augud.

Märkused: Planeerijas on võimalik kirja panna komisjoni liikmete soove. Ajakava pole sessiooniti, aga lõppkasutaja võib järjestada ajakava nii, nagu oleks sessiooniti. Lõputöö kirjutamise keel eemaldati planeerijast.

Küsimus 6: Mida Te peate olulisemaks kui Te koostate kaitsjate ajakava? (Millised kitsendused on olulisemad, milliseid ei saa rikkuda, jne)

Vastus: Küsimus on ära vastatud vastuses nr 3 ja 5.

Küsimus 7: Mis kellaaegadel toimuvad kaitsmised ning kas on võimalik, et mõni kaitsmine toimub erandkorras hiljem?

Vastus: Eelistatult algusega kell 10, lõpp hiljemalt kell 5. Kui on aga palju kaitsjaid, siis võib alustada ka kell 9 ja lõpetada hiljem. Erandkorras hiljem kaitsmist pole juhtunud, aga kui tudeng annab teada, et talle see kava üldse ei sobi, siis ma üritan ajakava ümber teha.

Märkused: Planeerijas saab märkida kellaajad, millal nad on kinnised või lahtised kaitsmised.

Küsimus 8: Kui palju on tudengeid, kes ühel semestril informaatika õppekaval kaitsta soovivad?

Vastus: Tavaliselt ülesandepüstitusi tuleb umbes 100, aga kaitsmisele tuleb umber 60-70% ja üldiselt kõik kaitsevad ära, võibolla on paar kes ei suuda ära kaitsta.

Küsimus 9: Kui tihti tuleb ette ümberplaneerimist?

Vastus: Tuleb, aga mitte nii hullusti, et ma hakkama ei saa. Ma üritan ennetada ümberplaneerimisi sellega, et ma lasen tudengitel kirja panna, mis ajad neile kindlasti ei sobi.

Märkused: Monika Kreinin paneb kaitsmised kirja nii, et kes esimesena lõputöö ära toob, see saab seda esimesena kaitsta. Ta ütles veel, et abiks oleks, kui saaks Exceli ühildada Moodlega, et paneks automaatselt kirja pealkirjad, inimesed, jne.

3.4 Kitsendused

Lõputöös olevad kitsendused on jaotatud tugevateks ja nõrkadeks kitsendusteks. Kitsendused on planeerija jaoks loodud reeglid, mille järgi hinnatakse planeeritud tulemust. Planeeritud tulemust hinnatakse kõrgemalt, kui kitsendusi on rikutud võimalikult vähe.

3.4.1 Tugevad kitsendused

Tugevateks kitsendusteks nimetatakse kitsendusi, mida planeerija ei tohi rikkuda, muidu saadud tulemus pole sobilik. Tugevate kitsenduste ingliskeelsed nimed on välja toodud joonisel (Joonis 3).

```
Defense not on authors unavailable timeslot
Defense not on commission members unavailable timeslot
Defense not on authors unavailable timeslot tag
Defense timeslot only for single author
Defense on commission members unavailable timeslot tag
```

Joonis 3. Need on tugevate kitsenduste ingliskeelsed nimed

- Kaitsmine ei toimu lõputöö kaitsja sobimatul kuupäeva kellaajal.
- Kaitsmine ei toimu komisjoni liikme sobimatul kuupäeva kellaajal.
- Kaitsmine ei toimu lõputöö kaitsja sobimatul aja märksõnal.
- Kaitsmisel on autoril ainult üks kaitsmise aeg.
- Kaitsmine ei toimu komisjoni liikme sobimatul aja märksõnal.

3.4.2 Nõrgad kitsendused

Nõrkadeks kitsendusteks nimetatakse kitsendusi, mida planeerija tohib rikkuda, aga paremaks loetakse tulemuselt, kus nõrkasid kitsendusi on rikutud vähem. Nõrkade kitsenduste ingliskeelsed nimed on välja toodud joonisel (Joonis 4).

```
Defense authors grouped by common supervisor
Defense on authors preferred timeslot
Defense on authors not preferred timeslot
Defense on commission members preferred timeslot
Defense on commission members not preferred timeslot
Defense on authors supervisors preferred timeslot
Defense on authors supervisors not preferred timeslot
Defense on authors supervisors unavailable timeslot
Defense on authors preferred timeslot tag
Defense on authors not preferred timeslot tag
Defense on commission members preferred timeslot tag
Defense on commission members not preferred timeslot tag
Defense on authors supervisors preferred timeslot tag
Defense on authors supervisors not preferred timeslot tag
Defense on authors supervisors unavailable timeslot tag
Defense timeslots grouped by common session and have no holes
between them
Commission member does not swap with a new member in the same
session
```

Joonis 4. Need on nõrkade kitsenduste ingliskeelsed nimed

- Kui ühel juhendajal on mitu juhendatavat, siis nende kaitsmised grupeeritakse kokku.
- Kaitsmine toimub lõputöö kaitsja eelistatud kuupäeva kellaajal.
- Kaitsmine ei toimu lõputöö kaitsja mitte-eelistatud kuupäeva kellaajal.
- Kaitsmine toimub komisjoni liikme eelistatud kuupäeva kellaajal.
- Kaitsmine ei toimu komisjoni liikme mitte-eelistatud kuupäeva kellaajal.
- Kaitsmine toimub lõputöö kaitsja juhendaja eelistatud kuupäeva kellaajal.
- Kaitsmine ei toimu lõputöö kaitsja juhendaja mitte-eelistatud kuupäeva kellaajal.
- Kaitsmine ei toimu lõputöö kaitsja juhendaja sobimatul kuupäeva kellaajal.
- Kaitsmine toimub lõputöö kaitsja eelistatud aja märksõnal.
- Kaitsmine ei toimu lõputöö kaitsja mitte-eelistatud aja märksõnal.

- Kaitsmine toimub komisjoni liikme eelistatud aja märksõnal.
- Kaitsmine ei toimu komisjoni liikme mitte-eelistatud aja märksõnal.
- Kaitsmine toimub lõputöö kaitsja juhendaja eelistatud aja märksõnal.
- Kaitsmine ei toimu lõputöö kaitsja juhendaja mitte-eelistatud aja märksõnal.
- Kaitsmine ei toimu lõputöö kaitsja juhendaja sobimatul aja märksõnal.
- Kaitsmise ajad on grupeeritud sarnase sessiooni järgi ja nende vahel pole tühjasid aegasid.
- Kaitsmisel komisjoni liige ei vaheta oma kohta uue liikmega samal sessioonil.

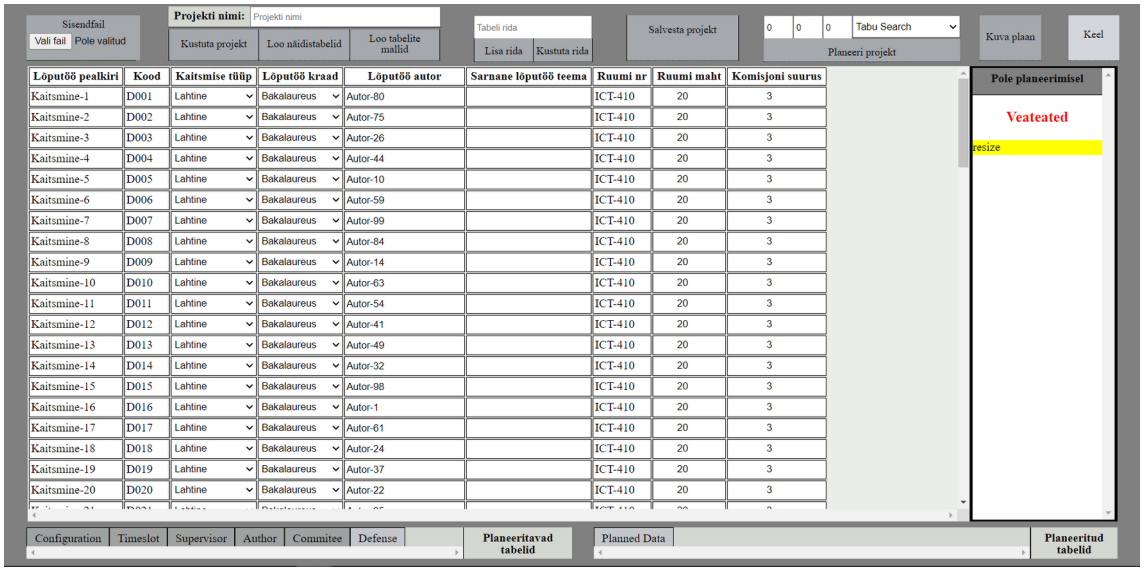
4 Prototüüp

Prototüüp koosneb kahest osast, millest esimene on lõppkasutajale nähtav veebiliides ja teine on planeerija loogikaga tegelev Java programm. Need osad suhtlevad omavahel läbi Jetty serveri. Ühiktestid on kirjutatud JUnitis. Prototüübi kasutusjuhend on olemas lisas (vt Lisa 3).

4.1 Veebiliides

Veebiliidese loomiseks on kasutatud HTML-i (Hyper Text Markup Language), CSS-i (Cascading Style Sheets) ja programmeerimise keelt JavaScript. Veebiliidesele on lisatud juurde väliseid teeke, mis annab juurde funktsionaalsust. Väliste teekide seast on kasutatud Jquery-3.4.1.js [4] , xlsx.full.min.js [5] , xlsx.core.min.js [6] , FileSaver.js [7] ja tableexport.js [8] .

4.1.1 Kujundus



Joonis 5: Planeerija veebiliides

Veebiliidese kujundus jaguneb neljaks osaks. Veebiliidese üleval ribal asuvad kõik menüü nupud. Veebiliidese ekraani keskel asub tabel, kuhu lõppkasutaja paneb planeerija sisendandmed kirja. Veebiliidese paremal ääres asub veateadete kast, kust näidatakse lõppkasutajale, mis vead planeeritud projektil on ning missuguseid vigu on tehtud veebiliidest kasutades. Lisaks kuvatakse paremal ääres planeerijaga seotud andmed. Veebiliidese all ääres asuvad planeeritava ja planeeritud tabelite lipikud, mille kaudu saab lõppkasutaja leida soovitud tabelit. Veebiliidese kujundust on näha joonisel (Joonis 5).

Veebiliidesel nuppu „Sisendfail" vajutades saab lõppkasutaja valida soovitud Exceli faili, kust imporditakse sisendfailid veebiliidese tabelitesse. Nupp „Kustuta projekt" kustutab veebiliideses kõik olemasolevad tabelid ära.

Nupp „Loo näidistabelid" genereerib lõppkasutaja jaoks näidisandmetega planeeritavad tabelid, mida lõppkasutaja saab kasutada sisendandmete täitmise vihjete jaoks. Nupp „Loo tabelite mallid" teeb sama, mis eelmine nupp, aga ta loob tabeleid ainult pealkirjadega.

Järgmised kaks nuppu teevad täpselt seda, mida nupu nimed vihjavad. Nende nuppude ees on loodud üks tekstirida, mida nupp „Loo rida" kasutab.

Veebiliidese menüü nupp „Salvesta projekt" võtab faili nimeks projekti nime ning loob uue Exceli faili, kuhu salvestatakse kõik veebiliideses loodud tabelid. „Planeeri projekt" nupp salvestab sisendandmed JSON faili ja paneb planeerija tööle. Siis kui planeerija on töö lõpetanud, „Planeeri projekt" nupp paneb „Kuva plaan" nupu käima. „Kuva plaan" nupp loob väljundandmetega uued tabelid, mida saab vaadata kasutades planeeritud tabelite lipikuid veebiliidese paremal all ääres.

Lisaks on veel olemas üks abistavat nuppu lõppkasutaja jaoks. Nupp „Keel" muudab kõik nuppude nimed ja kirjeldused valitud keeleks, mille seast saab valida eesti ja inglise keele vahel.

Lõpuks Timeslot tabelis on olemas lisafunktsionaalsus, mis laseb lõppkasutajal luua aegasid automaatselt nii palju kui ta vajab.

4.1.2 Funktsionaalsus

Veebiliides kasutab kahte JavaScripti faili põhilise töö jaoks: `Veebiliides.js` ja `Examples.js`. Veebiliidese funktsionaalsus hoitakse failis `Veebiliides.js` ning projekti malli ja näidisandmed hoitakse failis `Examples.js.`

Meetod `async function planProject()` käivitab Java programmi ning saadab veebiliideses olevad sisendandmed edasi JSON faili `plandata.json`. Seda meetodit käivitab nupp „Planeeri projekt".

Meetod `async function displayPlan()` kuvab andmed plannedData.json failist ning loob nendega planeeritud tabeli veebiliidesesse.

Meetod `function addJSONPlanData()` loob JSON kujul andmed projekti tabelitest ning saadab nad AJAX POST päringuga Java Servletti „ServletPlan". Seda meetodit kasutab funktsioon `planProject()`.

Meetod `function writeJSONObject(name, lengthRow, lengthColumn, data)` paneb tabeli JSON kujul kokku. See on abistav funktsioon, mida kasutab meetod `addJSONPlanData()`.

Meetod `function sendPlannerConfigData()` loob JSON kujul andmed planeerija konfigureerimise jaoks ning saadab nad AJAX POST päringuga Java Servletti „ServletConfig". Seda meetodit kasutab funktsioon `planProject()`.

Meetod `function loadJSON(callback)` tagastab andmed GET päringuga, mida saadeti `plannedData.json` faili ja kirjutab planeeritud andmed planeeritud tabelisse. Seda meetodit kasutab funktsioon `displanPlan()`.

Meetod `function fillTableFromInput(table, row, column, ID)` võtab sisendandmed Exceli failist ning täidab veebiliideses olevad projekti tabelid saadud sisendandmetega. Seda meetodit kasutab funktsioon `getInputFile()`.

Meetod `function getInputFile(input)` võtab sisendiks Exceli faili ning edastab sisendandmed funktsiooni `fillTableFromInput()` ja loob uue projekti. Seda meetodit kasutab nupp „Sisendfail".

Meetod `function newProject(tab)` võtab sisendiks võtmesõna, mille järgi teab meetod kust vanad tabelid ära kustutada. Kui sisend tuleb `getInputFile()` meetodist, siis kustutatakse ära planeeritavad tabelid ja kui sisend tuleb `displayPlan()` meetodist, siis kustutatakse ära planeeritud tabelid. Seda meetodit kasutab nupp „Uus Projekt" ja meetodid `displayPlan()`, `getInputFile()`, `generateTemplateTable()`, `generateExampleTable()`.

Meetod `function generateExampleTable()` loob uue projekti ning täidab veebiliideses olevad tabelid näidisandmetega. Seda meetodit kasutab nupp „Loo Näidistabelid".

Meetod `function generateTemplateTable()` loob uue projekti ning täidab veebiliideses olevad tabelit malli formaadiga, mitte lisades näidisandmeid. Seda meetodit kasutab nupp „Loo Tabelite Mallid".

Meetod `function generateTable(row, column, header)` loob tabeli antud rea suurusega, veeru suurusega ning tabeli nimega. Seda meetodit kasutab funktsioonid `generateTemplateTable()`, `generateExampleTable()`, `getInputFile()` ja `displayPlan()`.

Meetod `function addRow()` loob aktiivsele tabelile uue rea. Meetodiga saab lisada mitu rida korraga. Seda meetodit kasutab nupp „Lisa rida" ja funktsioon `generateTable()`.

Meetod `function addColumn()` loob aktiivsele tabelile uue veeru. Meetodiga saab lisada mitu veergu korraga. Seda meetodit kasutatakse siis kui luuakse uusi aegasid tabelisse juurde.

Meetod `function deleteRow()` kustutab aktiivselt tabelilt ära aktiivse rea. Seda meetodit kasutab nupp „Kustuta rida".

Meetod `function addTab(tabName, tabArea)` lisab projektile uue lipiku, millega saab üles leida selle lipikuga seotud tabelit. Seda meetodit kasutavad funktsioonid `planProject()`, `getInputFile()`, `generateTemplateTable()` ja `generateExampleTable()`.

Meetod `function showTab(table)` teeb antud tabeli aktiivseks ning kuvab veebiliidese tabeli vaates. Seda meetodit kasutavad tabeli nupud tabelite reas.

Meetod `function deleteTab()` kustutab ära aktiivse lipiku ja sellega seonduva tabeli. Seda meetodit kasutab funktsioon `newProject()`.

Meetod `function s2ab(s) [24]` on abimeetod, mis puhverdab sisendandmed Exceli faili jaoks. Seda meetodit kasutab funktsioon `saveProject()`.

Meetod `function getTableData(name)` võtab tabelis olevad andmed ning salvestab tabeli objektina. Seda meetodit kasutavad funktsioonid `saveProject()` ja `addJSONPlanData()`.

Meetod `function saveJSProject()` salvestab veebiliideses olevad tabelid uute Exceli faili. Seda meetodit kasutab nupp „Salvesta projekt”.

Meetod `function languageDropdown()` esitab lõppkasutajale keelevalikud, mis kasutades `function languageChoice(language)` muudab kõikide nuppude nimed ja kirjeldused valitud keeleks. Seda meetodit kasutab nupp „Keel”.

4.2 Andmeedastus JSON-i kujul

JSON on lihtsustatud andmevahetusformaat. JSON on lihtne tekstifail, mis hoiab andmeid organiseeritult ning on alternatiiv XML-le. Antud töös on kasutatud JSON formaati XML-i asemel, et edastada veebiliidesest saadud andmeid Java andmeobjektidesse. JSON-t on kasutatud XML asemel sellepärast, et autoril on rohkem kogemust JSON-i kasutamisel.

Andmeedastus toimub antud töös kahel korral. Andmete lugemisega tegeleb fail `Reader.java` ja andmete kirjutamisega tegeleb fail `Writer.java`. Need failid asuvad `json` paketis.

4.2.1 Andmete lugemine

Andmete lugemise klass koosneb abifunktsioonidest, andmete domeenimudelisse seadmise funktsioonidest ning JSON-i lugemise funktsioonist.

Kõige esimesena kutsutakse välja funktsioon `void readJSON(String filename)`, mis loeb andmed JSON failist sisse ning kasutades abifunktsiooni `void addToTableList(ArrayList<String> List)`, paneb andmed õigetesse `List` objektidesse.

Järgmisena on loodud abistavad funktsioonid, mis lihtsustavad domeenimudelisse seadmise funktsiooni loetavust ja hooldamist. Meetod `void titleCheck(String input, String answer)` kontrollib, kas tabeli tiitlid on õiged. Kui tabeli tiitlil on mingisugune viga, siis tagastatakse veateade. Järgmised kaks meetodit `Set<String> seperateCommasSet(String timeTag)` ja `List<String> seperateCommasList(String timeTag)` loevad mitu komakohta on antud märgendil ning enne tagastamist eraldavad need kas `Set` objekti või `List` objekti vastavalt kumba meetodit välja kutsutakse. Lisaks on loodud meetod `String timeCheck(String time)`, mis kontrollib aja formaate ja kas antud kellaajad on õiged ning tagastab parandatud ajaformaadi, kui antud sisend ei sobi loodud Java funktsioonidega. Järgmisena on loodud meetod `int getRowLength(ArrayList<ArrayList<String>> dataList)`, mis tagastab tabeli veeru suuruse. Ilma selle funktsioonita pannakse andmed valedesse domeenimudeli andmeobjektidesse.

Lõpuks on loodud meetod `TimetableSolution read(String fileName)`, mis kutsub välja `readJSON` meetodi ning kuus domeenimudelisse seadmise funktsiooni. Nende funktsioonide nimed on näha jooniselt 7 (vt Lisa 2). Domeenimudelisse seadmise funktsioonid kontrollivad, kas veebiliidesest saadud sisendandmed on korrektsed ning panevad `List` objektidest õiged andmed domeenimudelisse, jättes välja veebiliidesele vajalikud andmed.

Lisaks on loodud Getterid ja Setterid, mida kasutatakse meetodite testimises. Kasutajalt konfiguratsiooni nõuete lugemiseks on loodud `readJSONConfig()` meetod, mille kaudu saab rakendust käivitamisel konfigureerida.

4.2.2 Andmete kirjutamine

Andmete kirjutamise klass koosneb abifunktsioonidest, domeenimudeli lugemise funktsioonidest ja JSON-sse kirjutamise funktsioonidest.

Kõige esimesena kutsutakse välja funktsioon `void write(TimetableSolution solution)`, mis loeb antud lahenduse objektist olulisi andmeid ja paneb kokku JSON stringi ning kirjutab selle `plannedData.json` faili.

Andmete kirjutamiseks on loodud iga olulise andmekirje jaoks `JSONArray` kollektsioonid ja tavalised `List` kollektsioonid. Kollektsioonid on sorteeritud kuupäeva ja kellaaja järgi ning on lõpuks abifunktsioonidega kokku pandud `JSONObject` objektideks. Abifunktsioone on üksteist tükki, mille erinevused on ainult selles, millist tüüpi andmeid kirjutatakse lõpptulemuste jaoks.

4.3 Domeenimudel

Domeenimudel on loodud selleks, et hoida kõik planeerijale vajalikke sisendandmeid ning luua tulemuse objekte, mille seast valitakse välja parima skooriga tulemus, mille andmed tagastatakse veebiliidesesse ja eraldi Exceli faili.

Domeenimudel koosneb kaheksast erinevast failist. Esimeseks on kitsenduste konfiguratsiooni fail nimega `TimetableConstraintConfiguration.java`. Teiseks on planeerimisüksuste fail nimega `Defense.java`. Järgmisteks on viis probleemfakti faili nimega: `Timeslot.java`, `ThesisSupervisor.java`, `ThesisAuthor.java`, `Commitee.java` ja `DefenseType.java`. Lõpuks on planeerimise lahenduse fail nimega `TimetableSolution.java`.

4.3.1 TimetableConstraintConfiguration

`TimetableConstraintConfiguration.java` fail koosneb kõigist ärireeglite kitsendustest. Esiteks on loodud `final String` muutujad, millega saab Drools'i reeglid siduda kindlate kitsendustega. Teiseks on loodud `HardSoftScore` muutujad, mis hoiavad kitsenduste skoori väärtuseid. Lõpuks on loodud iga kitsenduse jaoks Getterid ja Setterid, et neid algväärtustada.

4.3.2 Timeslot

`Timeslot.java` fail koosneb viiest olulisest muutujatest: `LocalTime startTime`, `LocalTime endTime`, `LocalDate date`, `int session` ja `Set<String> tagSet`. `TagSet` muutuja hoiab märksõnu, mis iseloomustavad `timeslot` objekte. Kõigile muutujatele on loodud Getterid ja Setterid. Kõik teised domeenimudeli klassid kasutavad `timeslot` objekte.

4.3.3 ThesisSupervisor

`ThesisSupervisor.java` fail koosneb põhiliselt kahest muutujast: `String name` ja `String role`. `ThesisSupervisor` klassil on lisaks juurde loodud kuus `Set` kollektsiooni, kus hoitakse `timeslot`'ide märksõnu ja `timeslot`'e iseennast ning need on jaotatud kolmeks: eelistatud, mitte-eelistatud ja sobimatud. Kõigile on loodud Getterid ja Setterid.

4.3.4 ThesisAuthor

`ThesisAuthor.java` fail koosneb kahest lihtsast muutujast: `String name` ja `boolean preconditionsFulfilled`. `ThesisAuthor` hoiab ka `Set` kollektsiooni `ThesisSupervisor`'st ja kuute `Set` kollektsiooni `timeslot` märksõnadest ja `timeslot`'dest. Need kuus `Set`'i jagunevad eelistatud, mitte-eelistatud ja sobimatud kollektsioonideks. Kõigile neile on loodud Getterid ja Setterid.

4.3.5 Commitee

`Commitee.java` fail koosneb neljast muutujast: `String name`, `String degree`, `boolean chairman` ja `String chairmanType`. `Boolean chairman` annab teada, et see komisjoni liige on esimees ning `String chairmanType` annab teada, kas ta on esimees või aseesimees. `Commitee` klassis on veel kuus `Set` kollektsiooni, kus hoitakse `timeslot`'i märksõnu ja `timeslot`'te iseennast ning nad jagunevad kolmeks: eelistatud, mitte-eelistatud ja sobimatud. Nendele kõigile on loodud Getterid ja Setterid.

4.3.6 DefenseType

`DefenseType.java` fail on abistav klass `Defense` klassile, mis hoiab muutujat `String type` ja `Set` kollektsiooni `compatibleTimeslotSet`, millega jagatakse `timeslot`'d kinnisteks ja lahtisteks `timeslot`'deks. Nende jaoks on loodud Getterid ja Setterid.

4.3.7 Defense

`Defense.java` fail hoiab palju erinevaid muutujaid. Esiteks on tal lihtsad muutujad: `String code`, `String degree`, `int commissionSize`, `String thesisTitle`, `String thesisTheme`, `String roomNumber` ja `int roomCapacity`. `Defense` klassil on lisaks `DefenseType defenseType`, mis märgib ära milline on kaitsmise tüüp, `ThesisAuthor thesisAuthor` objekt ja `Set` kollektsioon `Commitee` objektidest nimega `commiteeList` ja `Timeslot timeslot`, millele on lisatud annotatsioon `@PlanningVariable`. Lisaks on üks abistav muutuja `Commitee[] commiteeArray`, kus hoitakse kindel arv komisjoni liikmeid, kes osalevad sellel kaitsmisel.

Igale muutujale on loodud Getterid ja Setterid. Lisaks on juurde loodud abistavaid meetodeid. `SetCommission()` meetod paneb kaitsmisele paika komisjoni liikmed, pannes esimesena esimees paika, siis ülejäänud liikmed.

Meetod `checkWholeSetTimeslot(Set<Timeslot> timeslotSet)` kontrollib, et kas talle antud `timeslot`'de hulgast on sama `timeslot`'i kuupäev ja kellaaeg, mis on kaitsmisel.

Järgmisena on loodud meetodid, mida kasutavad Drools'i reegleid. Need meetodid kasutavad abistavaid meetodeid, et nende koodi lihtsustada ning tagastavad true, kui leitakse `timeslot`, mis vastab meetodi nimele. Nende meetodite nimed on näha joonisel 8 (vt Lisa 2).

Lisaks on loodud abistavad meetodid `timeslotTag`'de kontrollimiseks nimega `checkWholeSetTimeslotTag(Set<String> tagSet)`. Seda meetodit kasutavad järgmised meetodid joonisel 9 (vt Lisa 2).

Peale selle on veel teised meetodid Drools'i reeglite jaoks. `HasSameSupervisorOnSameSession(Defense other)` meetod kontrollib mõlema

kaitsmise juhendajaid, et kas need juhendajad on grupeeritud samal sessiooni `timeslot` objektidel. Lisaks on olemas sarnane meetod komisjoni liikmete jaoks nimega `hasSameCommiteeOnSameSession(Defense other)`. Lõpuks on olemas `overlapsTimeslot(Defense other)` meetod, millega kontrollitakse et mitmel kaitsmisel poleks sama `timeslot` objekt.

4.3.8 TimetableSolution

`TimetableSolution.java` fail hoiab kõiki teisi domeenimudeli objekte endas. Ta hoiab kitsenduste konfiguratsiooni `constraintConfiguration` muutujat, planeerimisüksuse `List` kollektsiooni `defenseList`, probleemfakti `List` kollektsiooni `defenseTypeList`, probleemfakti `List` kollektsiooni `thesisAuthorList`, probleemfakti `List` kollektsiooni `timeslotList`, probleemfakti `List` kollektsiooni `commiteeList`, probleemfakti `List` kollektsiooni `thesisSupervisorList` ning planeerimisskoori muutujat `HardSoftScore score`. Igale muutujale on lisatud Getterid ja Setterid.

4.4 Drools'i reeglid

Drools reeglite jaoks tegi töö autor `defenseTimetableScoreRules.drl` faili, kus kõik ärireeglid on kirjutatud Drools'i süntaksi järgi. Drools'i faili alguses deklareeritakse, et Drools'i reeglite dialekt on Java ning kõik reeglid kasutavad globaalset muutujat `HardSoftScoreHolder scoreHolder`.

Drools'i reeglid jagunevad kaheks osaks: eeldused ja tegevused ning kasutavad nelja võtmesõna: `rule`, `when`, `then`, `end`. `Rule` võtmesõnaga autor deklareerib Drools'i reegli nime, mis on võetud `TimetableConstraintConfiguratsiooni.java` failist. See nimi peab olema täpselt sama kitsendusega, mille skoori kasutatakse, et planeeritud tulemust väärtustada.

`When` võtmesõnaga antakse reeglile tingimused, mida on vaja täita, et planeeritud tulemust muuta või mitte. `Then` võtmesõnaga muudetakse planeeritud tulemuse skoori. `End` võtmesõnaga pannakse reeglile lõpp paika.

4.5 Serveriga ühendamine

Veebiliidese ja Java koodi omavahel ühendamiseks kasutatakse Java Servleti [19] ja Gretty pluginat [20] , mis lubab käivitada programmi Jetty serveris. Lisaks on Java projektile lisatud WAR plugin (Web Application Resource), mis annab rakendusele veebiserveri toetust ja Eclipse-wtp, mis lisab Eclipse IDE-s projektile lisakäske.

Selleks, et veebiliides saaks suhelda planeerijaga, on tehtud HTML fail ümber JSP failiks (Java Server Page). Peale seda on JavaScripti faili lisatud JQuery-ga AJAX GET päring, millega pannakse Javas planeerija tööle ja kuvatakse tagasi vastus, et planeerija on pandud tööle. Lisaks on juurde tehtud AJAX POST päring, millega saadetakse servletile JSON andmed, et Javas saaks kirjutada andmed üle JSON faili. Javas on lisatud failid `ServletConfig.java` ja `ServletPlan.java`, mis kasutab servleti, et saada veebiliidesest saadetud GET ja POST päringuid kätte. [18]

Kasutajal on võimalik konfigureerida, kui kaua ta soovib ajakava planeerida. Veebiliideses on lisatud neli välja, kus saab seadistada tundide, minutite ja sekundite arvu ning planeerimisalgoritmi. Java rakendus saab neid andmeid kätte teise AJAX POST päringuga servletis, mis saadab planeerimiskonfiguratsiooni andmed edasi `planConfig.json` faili, mida planeerija kasutab oma töös.

Veebiserveri loomiseks proovis autor mitmeid erinevaid viise, millest paljud kas ei töötanud või nõudsid mitmeid lisapluginaid, sõltuvusi või ei lahendanud probleemi nii nagu sooviti. Lõpuks valiti välja praegune lahendus ja Vaadin [21] veebiserveri lahendus, millest Vaadin oma ei sobinud, kuna see andis lisaks palju kõrvalisi tunnusjooni, mille kaasamine oleks häirinud veebiliidese eesmärki.

Veebiserveri kasutamine tõi kaasa probleeme, kus planeerija Drools'i reeglid tagastavad veateateid, mida tavalisel Java rakendusel ei tekkinud. Selle lahendamiseks tehti Windowsi jaoks Batch fail ja Linuxi jaoks ShellScript fail, mille sees on `gradlew run` käsk. Selle käsuga pannakse kogu Java projekt Java rakendusena tööle, mis lahendab Drools'i reeglite mittetöötamise probleemi. Lisaks on loodud veel `Ready.txt` fail, mida Java rakendus oma töö lõpus uuendab, lisades sinna tähe x. JavaScriptis loetakse `Ready.txt` faili iga 10 sekundi tagant. Kui failis on täht x, siis loetakse

`PlannedData.json` failist planeeritud andmeid. Sellist lahendust kasutati, kuna Java rakendus ei saa otse veebiliidesele teada anda, kui tema töö on lõpetatud.

4.6 Testimine

4.6.1 Üksustestid

Java koodi osa testimiseks kasutakse JUnit 4.12 [22] . JUnit 5 testimise raamistikku ei võetud kasutusse, kuna Java koodis ei kasutata Java 8 eripäralisi funktsioone ning JUnit 4.12 kasutamisega on töö autoril rohkem kogemust. Testid on eraldatud kolme erinevasse paketti: `domain`, `drools` ja `json`. Testide kirjutamisel kontrollitakse igat funktsiooni vähemalt ühe erineva tulemusega.

4.6.1.1 Domeenimudeli testimine

Enamuse domeenimudeli klasside testimiseks piisas ainult lihtsate testide kirjutamisest, kuna nendel funktsioonidel on ainult üks või kaks tulemust. Domeenimudeli klassid koosnevad Getteritest ja Setteritest. Kuna Setterid on `void` meetodid, siis nende testimiseks on vaja välja kutsuda Getterid ja kuna Getterite jaoks eraldi testide kirjutamine on üleliigne, siis kombineeritakse nad üheks testiks, kus algul paneb muutujasse andmed paika Setteriga ja kontrollib tulemust Getteriga. Testide läbimiseks kasutatakse nelja JUnit meetodit: `assertEquals()`, `assertTrue()`, `assertFalse()` ja `assertNull()`. Vähemate koodiridade kirjutamiseks kasutatakse meetodeid `setUp()`, kus iga testi alguseks pannakse ühised algandmed paika ja `tearDown()`, kus vabastatakse mäluruumi peale teste.

4.6.1.2 Mockitoga testimine

`Defense` klassi testimiseks ei piisanud lihtsast testimisest, kuna ta koosneb kompleksmeetoditest, mis kasutavad teisi domeenimudeli klasse. JUnit 4 testimise meetoditega tekkisid nendel testidel `NullPointerException`’id, mis viitasid, et kasutatavad andmeobjektid pole initsialiseeritud. Vältimaks sõltuvate andmeobjektide initsialiseerimist testides, kasutatakse Mockitot.

Mockito on testimise raamistik, millega saab luua võltsobjekte. Neid objekte kasutatakse lihtsa sõltuvuse loomiseks testidele. [23] Teiste klasside objektide sõltuvuse loomiseks kasutatakse `mock()` meetodit, mis loob klassist võltsobjekti. Võltsobjektiga saab simuleerida klassi tegevusi. Kindla klassi meetodi simuleerimiseks kasutatakse meetotit `when()` ja selle simuleeritud meetodi oodatav tulemus tagastatakse meetodiga `thenReturn()`.

Peale testi läbimist tuleb kontrollida, kas simuleeritud meetodeid on üldse kasutatud. Seda kontrollitakse meetodiga `verify()`, mis kontrollib, kas simuleeritud meetodit kasutati testides. Lisaks saab `verify()` meetodis kasutada `times()` meetodit, millega vaadatakse, mitu korda on simuleeritud meetodit rakendatud.

4.6.1.3 Testandmetega testimine

JSON-ga seonduvate klasside `Reader.java` ja `Writer.java` testimiseks on vaja kaasa anda algseid sisendandmeid. Kasutades veebiliidese funktsionaalsust, luuakse testandmetega JSON fail, mille andmeid sisestatakse Java andmeobjektidesse.

Reader.java fail koosneb enamasti `void` meetoditest, mis algväärtustavad domeenimudeli objekte. `Reader.java` failis on veel paar kompleksmeetodit, mis on loodud, et vähem koodi kirjutada. Kompleksmeetoditel testitakse igat ekstreemumtulemust ühe korra. Nendele testidele pole vaja kaasa anda sisendandmeid JSON failist. `Void` meetodite testimiseks antakse testidele sisendandmed `planDataTest.json` failist ning kontrollitakse igat domeenimudeli objekti, kas sisendandmed on nendes ära väärtustatud.

4.6.1.4 Drools'i reeglite testimine

Drools'i reeglite testimiseks peab kasutama `HardSoftScoreVerifier` klassi, milleks pidi lahti pakkima `OptaPlanner-test` teeki [1] . `ScoreVerifier` meetodi kasutamiseks peab looma `SolverFactory`'ga loodud `defenseTimetableSolverTabuSearchConfig.xml` faili. Testide jaoks on loodud kaks eraldi faili nimedega: `TimetablePlanningScoreHardConstraintTests.java` ja `TimetablePlanningScoreSoftConstraintTests.java`. Mõlemas failis luuakse üksusteste Drools'i testide jaoks, kus tugevaid kitsendusi testitakse

`TimetablePlanningScoreHardConstraintTests.java` failis ja nõrku kitsendusi testitakse `TimetablePlanningScoreSoftConstraintTests.java` failis. Drools'i testid erinevad teistest üksustestidest selle poolest, et üks test kontrollib mitut erinevat Drools'i reegli tulemust. Drools'i reeglite testidel kasutatakse `assertEquals` jms meetodite asemel `HardSoftScoreVerifier` objekti `assertHardWeight` ja `assertSoftWeight` meetodeid. Drools'i reeglite testimiseks on vaja `setUp` meetodis kõik testobjektid algväärtustada, muidu igas testis tekivad NullPointerException'd.

4.6.2 Seleniumtestid

Seleniumtestid on kirjutatud Java keeles ja testimise keskkond on Google Chrome brauser. Seleniumtestide kirjutamiseks on vaja kasutada Selenium client ja webdriver'it [9] . Seleniumtestide käivitamiseks Chrome keskkonnas on vaja kasutada Chrome *driver*'it [10] , mis avab isoleeritud veebilehe Chrome brauseris ning laseb kasutada selle veebilehe käske läbi Java koodi. Konfiguratsiooni failis on antud ka võimalus testida Mozilla Firefox brauseris. Selle jaoks on vaja kasutada Geckodriver'it [11] .

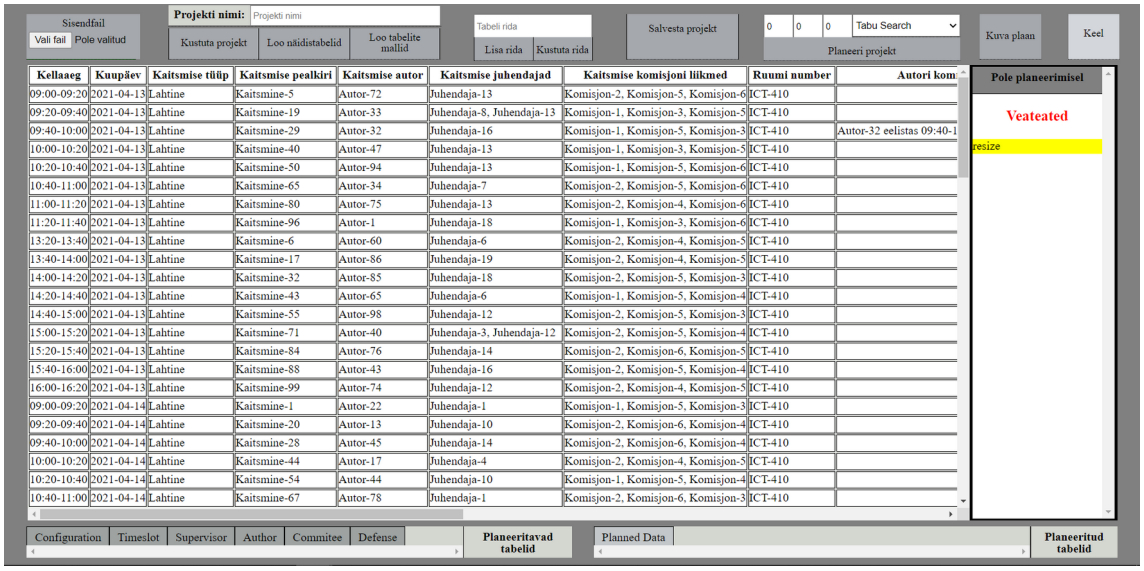
Seleniumtestide konfigureerimiseks on loodud fail `SeleniumConfig.java [12]` , kus saab valida kumba brauserit testimiseks kasutada ning sätestada, kus kohas asuvad `driver.exe` failid.

Testimise jaoks on loodud neli lähtefaili, mille kaudu saab käivitada kindlaid käske veebiliideses. `SeleniumTableButtons.java` failis on kõik veebiliidese tabelite manipuleerimise nuppude käsud. `SeleniumLanguageAndHelp.java` failis on kõik veebiliidese keelte ja juhendi nuppude käsud. `SeleniumIOAndExamples.java` failis on käsud sisendi saamiseks ning näidistabelite loomiseks. `SeleniumErrorMessages.java` failis on käsud, millega saab kontrollida veateateid.

Testid on kirjutatud järgmistesse failidesse: `SeleniumTableButtonsTest.java`, `SeleniumLanguageAndHelpTest.java`, `SeleniumIOAndExamplesTest.java` ja `SeleniumErrorMessagesTest.java`.

5 Tulemus

Planeerija tulemusi saab vaadata veebiliideses (Joonis 6). Kui ajakava on jõudnud ära planeerida, siis veebiliides kuvab ise tulemuse kasutajale ette. Kasutaja võib sealsamas veebiliideses hinnata saadud tulemust või eksportida saadud tulemuse Exceli faili ja seal kontrollida. Kui juhtub, et saadud tulemus pole piisavalt hea, siis seda sama ajakava võib ümberplaneerida, valides pikema aja ja/või mõne muu planeerimisalgoritmi.



Joonis 6: Planeerija tulemuste ekraanipilt

5.1 Testide algandmed

Testandmetega testid on kahte sorti. Esimesel on vähesed andmed, kus on 10 kaitsjat, 10 kaitsmist, 10 juhendajat, 6 komisjoni liiget ja 22 aega kuhu kaitsmised panna. Teisel on palju rohkem andmeid, kus on 100 kaitsjat, 100 kaitsmist, 20 juhendajat, 6 komisjoni liiget ja 125 kaitsmise aega.

Sisendandmetel on suvaliselt pandud eelistusi, sobimatuid aegasid, võtmesõnasid kindlatel aegadel, mis mõjutavad planeerimise tulemusi. Läbi on viidud 9 testi, mis

erinevad planeerimise aja poolest, kaitsmiste arvu poolest ja kasutatud algoritmi poolest. Kasutusele on võetud kolm erinevat planeerimisalgoritmi, millest *Tabu Search* on kohalik otsing, mis säilitab Tabu loendit, et mitte kinni jääda lokaalsesse optimumi. Järgmisena on *First Fit* algoritm, mis käib läbi igast planeerimisüksusest, lähtestades planeerimisüksuseid ühekaupa. Viimasena on *Hill Climbing* algoritm, mis proovib läbi kõiki käike ja valib parima käigu, et jõuda suurima skoorini. Läbitehtud testid on näha tabelis 1.

5.2 Testandmetega testid

Tabel 1. Testandmetega testid.

Testi nr.	**Aeg**	**Kaitsmiste arv**	**Algoritm**	**Tulemus**
1.	5 minutit	10	*Tabu Search*	Komisjoni liikmed vahetasid harva üksteist välja ja juhendajate grupeerimisi oli vähe.
2.	5 minutit	100	*Tabu Search*	Kaitsjatel ja komisjoni liikmetel palju eelistusi. Komisjoni liikmed olid samadel sessioonidel enam-vähem samad. Juhendajaid grupeeriti paremini, aga nende vahel oli auke.
3.	30 minutit	100	*Tabu Search*	Juhendajaid grupeeriti siin kõige paremini. Komisjoni liikmed vahetusid samadel sessioonidel, aga mitte palju ja nendel oli palju mitte eelistatud aegasid.
4.	5 minutit	10	*First Fit*	Planeeris 2 sekundiga. Juhendajaid grupeeriti nii palju kui võimalik. Komisjoni liikmeid ei suudetud hästi grupeerida.
5.	5 minutit	100	*First Fit*	Planeeris ära 18 sekundiga. Palju eelistusi ei tekkinud. Komisjoni liikmeid

				grupeeriti paremini, aga mitte hästi. Juhendajate grupeerimine oli märgatav.
6.	30 minutit	100	*First Fit*	Planeeris ära 20 sekundiga. Komisjoni liikmetel on vähe eelistusi. Komisjoni liikmeid suutis hästi grupeerida. Juhendajate grupeeringuid on palju.
7.	5 minutit	10	*Hill Climbing*	Juhendajaid suutis väga hästi grupeerida, aga komisjoni liikmetest vahetas alati vähemalt üks liige kohtasid.
8.	5 minutit	100	*Hill Climbing*	Juhendajate grupeerimisi on vähe, aga viimastel päevadel on neid näha. Komisjonide grupeerimised on head, aga kõikide eelistusi on vähe.
9.	30 minutit	100	*Hill Climbing*	Tulemusena saadi parema lahenduse kui 5 minuti planeerimisel. Tulemuses on juhendajate ja komisjonide liikmetel on rohkem grupeeringuid.

5.3 Testide hinnang

Antud testidel saadi kõige paremad tulemused *Tabu Search* algoritmiga. Selle algoritmiga leiti kõige rohkem eelistusi ja komisjoni liikmed vahetasid omavahel kohti samas sessioonis kõige vähem. *First Fit* annab kõige kiiremad tulemused, leides lahenduse alla 20 sekundiga, aga grupeerimisi ja eelistusi oli vähem. *Hill Climbing* algoritm andis vahepealsed tulemused, kus eelistusi polnud palju aga grupeeringud olid paremad kui *First Fit* algoritmil. Väiksema kaitsmiste arvuga testid ei suutnud grupeerida juhendajaid, kuna juhendajaid oli sama palju kui kaitsjaid, eelistuse võimalusi oli väga vähe, aga komisjoni liikmeid oli algoritmidel lihtsam grupeerida.

5.4 Tulevikuplaanid

Lõputöö jooksul ei suudetud realiseerida kõike, mida sooviti. Järgnevalt on toodud mõned mõtted, kuidas lõputööde planeerimise rakendust veel täiendada võiks.

Esiteks oleks võimalik kogu programm kokku panna Docker *image*'ina. Docker *image*'ina saaks programmi lihtsalt edasi jagada, käivitada ja juurutada, kuna Docker *image* sees on programmi jaoks olemas kõik olulised failid, sõltuvused, teegid ja kood, millega saab käivitada programmi.

Teiseks võiks programmi ära liidestada ÕISi ja/või TalTech Moodle'ga. Sellega saaks lihtsustada mõnede andmete lisamist veebiliidesesse lõppkasutaja jaoks, kus lõppkasutaja peaks ainult paari nuppu vajutama, et ta saaks põhilised sisendandmed valmis.

5.5 Hinnang OptaPlanner'i kasutamisele

Lõputööde ajakava planeerimisel kasutati planeerimismootorina OptaPlanner't. OptaPlanner'l on olemas väga pikk ja detailne dokumentatsioon, mis kirjeldab igat OptaPlanner'i osa ja sisaldab ka nii tekstilisi kui ka koodilisi näiteid. OptaPlanner't uuendatakse iga kuu ja koos sellega uuendatakse ka dokumentatsiooni, mis näitab et dokumentatsioon on uudne ja OptaPlanner ise on aktiivses arenduses.

OptaPlanner'i kasutamiseks oli vaja juurde õppida Drools'i raamistikku, millest arusaamine polnud kohe kindel ning vajas palju koodi parandamist, et Drools lõpuks tööle saada. OptaPlanner'i iseenda tööle panemisega läks ka aega, kuna lõputöö kaitsmiste ajakava koostamine sisaldas uusi probleeme, mida teised näited ei sisaldanud. Drools'i reeglid tõid endaga kaasa veel lisaprobleeme siis, kui Java kood pandi üles veebiserverisse.

OptaPlanner ise on hea planeerimismootor, millel on olemas põhjalik dokumentatsioon ja koodis kasutamine on lihtne, kuna OptaPlanner teeb ise suure osa planeerimisest ära. OptaPlanner'i osas ei meeldinud väga see, et Drools'i reeglid polnud lihtne kirjutada ja testida.

6 Kokkuvõte

Antud töö eesmärgiks oli luua rakendus lõputööde kaitsmiste ajakava koostamise lihtsustamiseks, pakkudes välja võimalikult täpselt nõudeid rahuldava ajakava lahendusi. Eesmärgi saavutamiseks loodi programm, mis kitsendusi arvesse võttes genereerib sisendandmete põhjal kaitsmiste ajakava.

Töös tehti Javas kirjutatud planeerija, mis kasutab planeerimismootorit OptaPlanner ja Drools'i keeles kirjutatud kitsendusi, mille abil leitakse valitud aja jooksul parim lahendus. Kitsenduste leidmiseks tehti intervjuu Monika Kreininiga, kelle käest saadi veel tagasisidet, mida planeerija võiks teha. Lõppkasutaja jaoks loodi veebiliides, mille abil on sisendandmete sisestamine lihtsam ja veakindlam, kuna lõppkasutajal on väiksem võimalus sisestada vigaseid andmeid. Tulemusi saab vaadata veebiliideses või eksportida Exceli kujule.

Töö koostamisel õppis autor palju uut. Tegemist oli autori esimese OptaPlanner'i mootorit kasutava rakendusega ja Drools'i reeglite kasutamisega. Autor polnud varem loonud veebiserveris veebirakendust, kus tagarakendus on kirjutatud Java keeles.

Töö tulemusena loodi veebirakendus, millega on võimalik planeerida lõputööde ajakava. Loodud rakendus võimaldab kasutajal võtta arvesse ka lõputööde autorite, komisjoni liikmete ja juhendajate soove, eelistusi ja sobimatuid aegasid. Planeerija tulemuste kontrollimiseks on läbi tehtud katseid ning kinnitati, et saadud tulemused on sobilikud ajakavad. Loodud rakendus võiks lihtsustada ja kiirendada lõppkasutaja tööd ajakava loomisel lõputööde kaitsmiste jaoks.

Loodud veebirakendust on võimalik edasi rakendada. Selleks, et rakendust parendada, võiks kogu rakendus viia üle Docker *image*'sse. Teiseks võiks veebiliidest liidestada ÕIS-i või TalTech Moodle'ga, kust saaks võtta planeerija jaoks sisendandmeid paari nupu vajutusega, lisades kasutaja jaoks mugavust ja kiirust.

Rakenduse repositoorium asub aadressil: https://github.com/RoaldValja/iaib

Kasutatud kirjandus

[1] MvnRepository, OptaPlanner Test 7.38.0.Final, 2020. [WWW] https://mvnrepository.com/artifact/org.optaplanner/optaplanner-test/7.38.0.Final (Kasutatud: 13.06.2020)

[2] OptaPlanner, Unit testing constraints with business input from Excel or LibreOffice, 2018. [WWW] https://www.optaplanner.org/blog/2018/08/21/UnitTestingConstraintsWithBusinessInputFromExcelOrLibreOffice.html (Kasutatud: 27.06.2020)

[3] OptaPlanner, OptaPlanner User Guide, 2020. [WWW] https://docs.optaplanner.org/7.38.0.Final/optaplanner-docs/html_single/index.html (Kasutatud: 12.06.2020)

[4] JQuery. v3.4.1, JS Foundation. [WWW] https://code.jquery.com/jquery-3.4.1.js (Kasutatud: 07.03.2020) ---tarkvara kirje

[5] Xlsx full min. 2013, SheetJS. [WWW] https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.16.5/xlsx.full.min.js (Kasutatud: 07.03.2020)

[6] Xlsx core min. 2013, SheetJS. [WWW] https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.16.5/xlsx.core.min.js (Kasutatud: 07.03.2020)

[7] FileSaver.js. 2020, Eli Grey. [WWW] https://github.com/eligrey/FileSaver.js/ (Kasutatud: 08.03.2020)

[8] TableExport. 2020, Eli Grey. [WWW] https://tableexport.v5.travismclarke.com/ (Kasutatud: 08.03.2020)

[9] Selenium. 2020, Jason Huggins. [WWW] https://www.selenium.dev/downloads/ (Kasutatud: 15.10.2020)

[10] ChromeDriver. 2020, Chromium and WebDriver teams. [WWW] https://chromedriver.chromium.org/ (Kasutatud: 16.10.2020)

[11] GeckoDriver. 2020, Henrik Skupin. [WWW] https://github.com/mozilla/geckodriver/releases (Kasutatud: 16.10.2020)

[12] Baeldung, Guide to Selenium with JUnit / TestNG, 2020. [WWW] https://www.baeldung.com/java-selenium-with-junit-and-testng (Kasutatud: 20.10.2020)

[13] Drools, Drools dokumentatsioon, 2021. [WWW] https://docs.jboss.org/drools/release/7.48.0.Final/drools-docs/html_single/ (Kasutatud: 27.01.2021)

[14] JaCoP Guide, JaCoP Library User's Guide, 2020. [WWW] http://jacopguide.osolpro.com/guideJaCoP.html (Kasutatud: 07.03.2021)

[15] Sciweb, Koalog Constraint Solver: fast constraint solving in Java, 2004. [WWW] http://www.sciweb.org/fjcp2004/slides/georget/nii041025.pdf (Kasutatud: 07.03.2021)

[16] Constraint Modeling, Minion, 2015. [WWW] https://constraintmodelling.org/minion/ (Kasutatud: 07.03.2021)

[17] Wikipedia, Constraint satisfaction, 2021. [WWW] https://en.wikipedia.org/wiki/Constraint_satisfaction (Kasutatud: 27.02.2021)

[18] Makble, Gradle Servlet Hello World example in Eclipse, 2021. [WWW] http://makble.com/gradle-servlet-hello-world-example-in-eclipse (Kasutatud: 21.03.2021)

[19] MvnRepository, JavaServlet(TM) Specification 2.5, 2006. [WWW] https://mvnrepository.com/artifact/javax.servlet/servlet-api/2.5 (Kasutatud: 22.03.2021)

[20] Gretty. 2021, Brett Randall. [WWW] https://plugins.gradle.org/plugin/org.gretty (Kasutatud: 22.03.2021)

[21] Vaadin, Starting a Project, 2021. [WWW] https://vaadin.com/docs/v14/guide/start/gradle (Kasutatud: 17.03.2021)

[22] MvnRepository, JUnit 4.12, 2014. [WWW] https://mvnrepository.com/artifact/junit/junit/4.12 (Kasutatud: 10.04.2020)

[23] MvnRepository, Mockito Core 2.23.4, 2018. [WWW] https://mvnrepository.com/artifact/org.mockito/mockito-core/2.23.4 (Kasutatud: 20.04.2020)

[24] StackOverFlow, How to save .xlsx data to file as blob, 2016. [WWW] https://stackoverflow.com/questions/34993292/how-to-save-xlsx-data-to-file-as-a-blob (Kasutatud: 08.03.2020)

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

Mina, Roald Välja

1 Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Lõputööde kaitsmiste ajakava koostamine kasutades OptaPlannerit" mille juhendaja on Riina Maigre

   1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

   1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2 Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3 Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

25.05.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Planeerija abistavad meetodid

```
public void readConfiguration()
public void readTimeslotList()
public void readSupervisorList()
public void readAuthorList()
public void readCommiteeList()
public void readDefenseList()
```

Joonis 7. Need on Reader klassi andmete domeenimudelisse seadmise funktsioonid

```
public boolean isAuthorsUnavailableTimeslot()
public boolean isAuthorsPreferredTimeslot()
public boolean isAuthorsNotPreferredTimeslot()
public boolean isCommissionMembersUnavailableTimeslot()
public boolean isCommissionMembersPreferredTimeslot()
public boolean isCommissionMembersNotPreferredTimeslot()
public boolean isAuthorsSupervisorsUnavailableTimeslot()
public boolean isAuthorsSupervisorsPreferredTimeslot()
public boolean isAuthorsSupervisorsNotPreferredTimeslot()
```

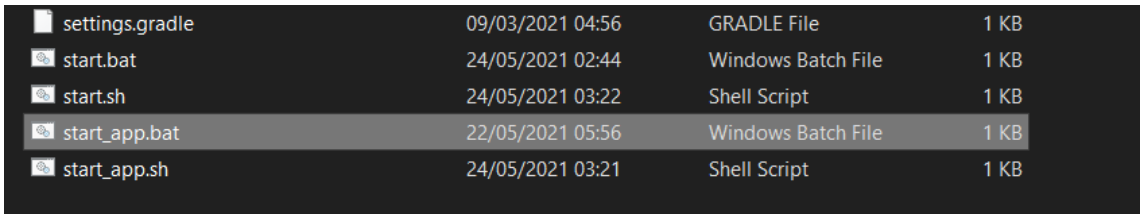
Joonis 8. Need on Defense klassi timeslot'dega seotud abistavate meetodite nimed

```
public boolean isAuthorsUnavailableTimeslotTag()
public boolean isAuthorsPreferredTimeslotTag()
public boolean isAuthorsNotPreferredTimeslotTag()
public boolean isCommissionMembersUnavailableTimeslotTag()
public boolean isCommissionMembersPreferredTimeslotTag()
public boolean isCommissionMembersNotPreferredTimeslotTag()
public boolean isAuthorsSupervisorsUnavailableTimeslotTag()
public boolean isAuthorsSupervisorsPreferredTimeslotTag()
public boolean isAuthorsSupervisorsNotPreferredTimeslotTag()
```

Joonis 9. Need on Defense klassi timeslot tag'dega seotud abistavate meetodite nimed

Lisa 3 – Prototüüp tarkvara kasutusjuhend

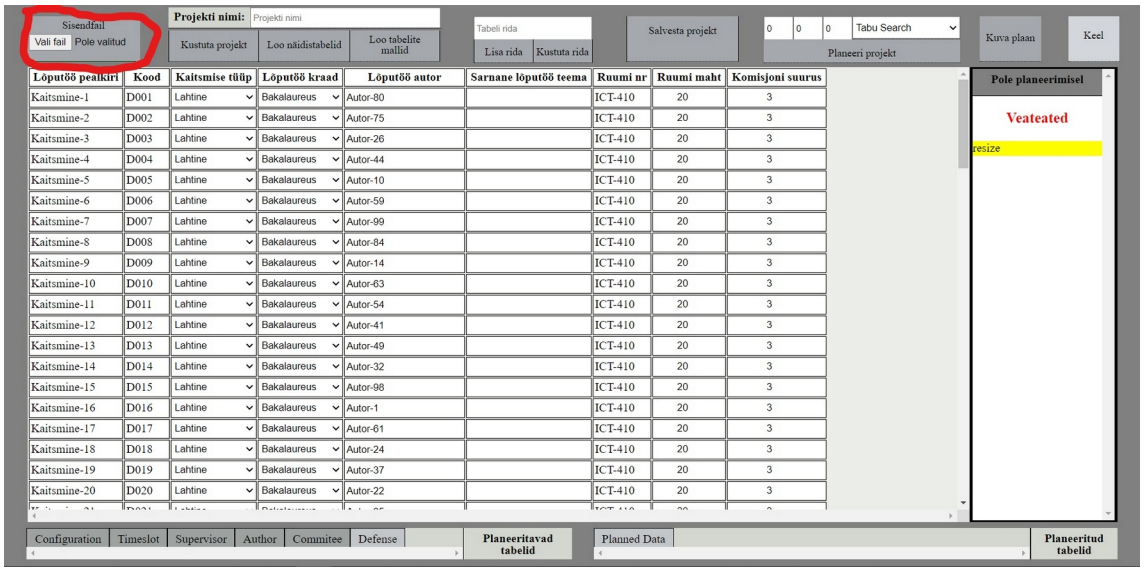
Tarkvara käivitamiseks on vaja teha rakenduse peal topeltklõps või paremklõps ja valida *open*, mis käivitab Jetty serveri. Windowsi kasutajad peavad kasutama start_app.bat faili ja Linux kasutajad peavad kasutama start_app.sh faili. (Joonis 10)



Joonis 10: Need on rakenduse käivitamise failid

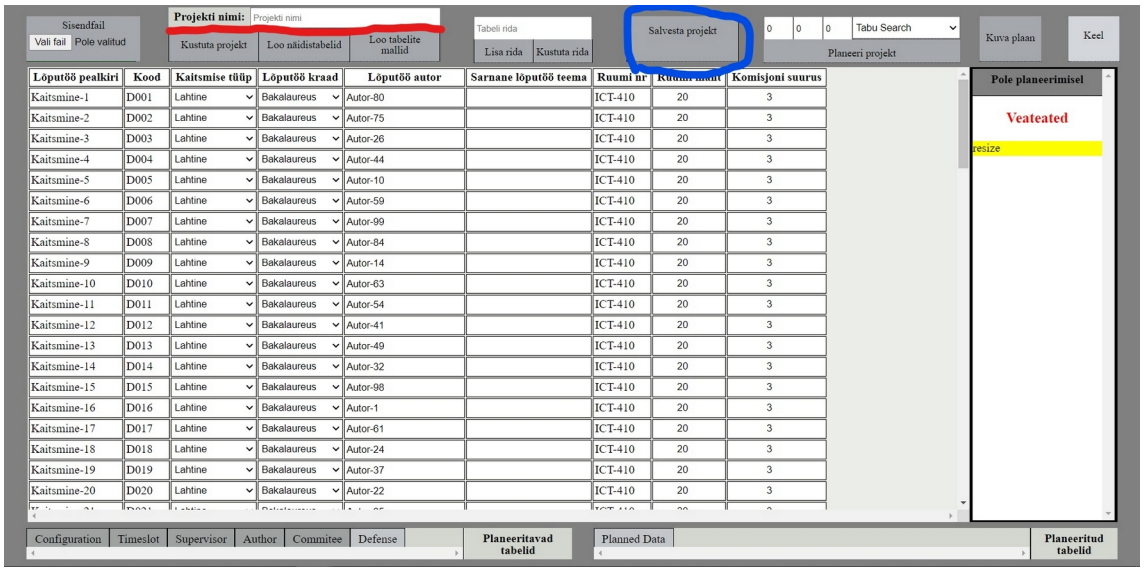
Seejärel tuleb avada oma valitud veebibrauser ja kirjutada URL: „localhost:8080". Sellega avaneb veebiliides, kus saab tegeleda ajakava planeerimisega.

Poolikud, valmis tehtud sisendandmed ja planeeritud andmed hoitakse Exceli failis. Veebiliideses saad valida vasakult ülevalt „Sisendfail" nupuga sisendfailiks Exceli faili, mis loob veebiliideses valmis olemasolevad tabelid. (Joonis 11)



Joonis 11: Veebiliidese sisendfaili nupp

Kui soovid oma tööd salvestada, kirjuta oma projektile omalt poolt valitud pealkiri joonisel punase värviga alla joonitud tekstialale, seejärel vali sinise ringiga märgistatud „Salvesta projekt” nupp ja salvesta oma tabelid Exceli faili. (Joonis 12)



Lõputöö pealkiri	Kood	Kaitsmise tüüp	Lõputöö kraad	Lõputöö autor	Sarnane lõputöö teema	Ruumi nr	Ruumi maht	Komisjoni suurus
Kaitsmine-1	D001	Lahtine	Bakalaureus	Autor-80		ICT-410	20	3
Kaitsmine-2	D002	Lahtine	Bakalaureus	Autor-75		ICT-410	20	3
Kaitsmine-3	D003	Lahtine	Bakalaureus	Autor-26		ICT-410	20	3
Kaitsmine-4	D004	Lahtine	Bakalaureus	Autor-44		ICT-410	20	3
Kaitsmine-5	D005	Lahtine	Bakalaureus	Autor-10		ICT-410	20	3
Kaitsmine-6	D006	Lahtine	Bakalaureus	Autor-59		ICT-410	20	3
Kaitsmine-7	D007	Lahtine	Bakalaureus	Autor-99		ICT-410	20	3
Kaitsmine-8	D008	Lahtine	Bakalaureus	Autor-84		ICT-410	20	3
Kaitsmine-9	D009	Lahtine	Bakalaureus	Autor-14		ICT-410	20	3
Kaitsmine-10	D010	Lahtine	Bakalaureus	Autor-63		ICT-410	20	3
Kaitsmine-11	D011	Lahtine	Bakalaureus	Autor-54		ICT-410	20	3
Kaitsmine-12	D012	Lahtine	Bakalaureus	Autor-41		ICT-410	20	3
Kaitsmine-13	D013	Lahtine	Bakalaureus	Autor-49		ICT-410	20	3
Kaitsmine-14	D014	Lahtine	Bakalaureus	Autor-32		ICT-410	20	3
Kaitsmine-15	D015	Lahtine	Bakalaureus	Autor-98		ICT-410	20	3
Kaitsmine-16	D016	Lahtine	Bakalaureus	Autor-1		ICT-410	20	3
Kaitsmine-17	D017	Lahtine	Bakalaureus	Autor-61		ICT-410	20	3
Kaitsmine-18	D018	Lahtine	Bakalaureus	Autor-24		ICT-410	20	3
Kaitsmine-19	D019	Lahtine	Bakalaureus	Autor-37		ICT-410	20	3
Kaitsmine-20	D020	Lahtine	Bakalaureus	Autor-22		ICT-410	20	3

Joonis 12: Veebiliidese pealkirja ja salvesta nupu alad