



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
У НОВОМ САДУ




Александар Стојановић

Оптимизација планирања распореда на примеру апликације за гимнастичка такмичења

Дипломски рад
- Основне академске студије -

Нови Сад, 2023.

	УНИВЕРЗИТЕТ У НОВОМ САДУ ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Датум:
	ЗАДАТАК ЗА ИЗРАДУ ДИПЛОМСКОГ (BACHELOR) РАДА	Лист:
		1/1

(Податке уноси предметни наставник - ментор)

Врста студија:	Основне академске студије
Студијски програм:	Рачунарство и аутоматика
Руководилац студијског програма:	проф. др Милан Рапаић

Студент:	Александар Стојановић	Број индекса:	РА 149/2019
Област:	Електротехничко и рачунарско инжењерство		
Ментор:	проф. др Горан Сладић		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ДИПЛОМСКИ РАД СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;
- литература

НАСЛОВ ДИПЛОМСКОГ (BACHELOR) РАДА:

Оптимизација планирања распореда на примеру апликације за гимнастичка такмичења

ТЕКСТ ЗАДАТКА:

Анализирати принципе програмирања ограничења и комуникације у реалном времену. Упознати се са радом *OptaPlanner* библиотеке за програмирање ограничења. Специфицирати и имплементирати апликацију за организацију и спровођење гимнастичких такмичења која ће користити наведену *OptaPlanner* библиотеку. Документовати решење.

Руководилац студијског програма:	Ментор рада:

Примерак за: ☐ - Студента; ☐ - Ментора

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР :	
Идентификациони број, ИБР :	
Тип документације, ТД :	монографска публикација
Тип записа, ТЗ :	текстуални штампани документ
Врста рада, ВР :	Дипломски рад
Аутор, АУ :	Александар Стојановић
Ментор, МН :	проф. др Горан Сладић
Наслов рада, НР :	Оптимизација планирања распореда на примеру апликације за гимнастичка такмичења
Језик публикације, ЈП :	Српски
Језик извода, ЈИ :	српски / енглески
Земља публиковања, ЗП :	Србија
Уже географско подручје, УГП :	Војводина
Година, ГО :	2023
Издавач, ИЗ :	ауторски репринт
Место и адреса, МА :	Нови Сад, Факултет техничких наука, Трг Доситеја Обрадовића 6
Физички опис рада, ФО :	бр. Поглавља 7 / страница 63 / цитата / табела / слика 25 / графикона / прилога / листинга 14
Научна област, НО :	Електротехничко и рачунарско инжењерство
Научна дисциплина, НД :	Софтверско инжењерство
Предметна одредница / кључне речи, ПО :	Гимнастика, ограничења, комуникација, планирање, распоред
УДК	
Чува се, ЧУ :	Библиотека Факултета техничких наука, Трг Доситеја Обрадовића 6, Нови Сад
Важна напомена, ВН :	
Извод, ИЗ :	Описано је шта су проблеми планирања и како се помоћу <i>OptaPlanner</i> -а они моделују и дефинисањем низа ограничења решавају. Такође, посвећена је пажња имплементацији <i>web socket</i> сервера, који омогућава оцењивање гимнастичара у реалном времену.
Датум прихватања теме, ДП :	
Датум одбране, ДО :	
Чланови комисије, КО :	
Председник	др Бранко Милосављевић, ред. проф, ФТН Нови Сад
Члан	др Мирослав Зарић, ванр. проф, ФТН Нови Сад
Ментор	др Горан Сладић, ред. проф, ФТН Нови Сад
Потпис ментора	

KEY WORDS DOCUMENTATION

Accession number, ANO :	
Identification number, INO :	
Document type, DT :	monographic publication
Type of record, TR :	textual material
Contents code, CC :	bachelor thesis
Author, AU :	Aleksandar Stojanović
Mentor, MN :	Goran Sladić, PhD, full. prof., FTN Novi Sad
Title, TI :	Schedule planning optimization using the example of a gymnastics competition application
Language of text, LT :	Serbian
Language of abstract, LA :	Serbian / English
Country of publication, CP :	Serbia
Locality of publication, LP :	Vojvodina
Publication year, PY :	2023
Publisher, PB :	author's reprint
Publication place, PP :	Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6
Physical description, PD :	no. of chapters 7/ pages 63/ quotes / tables / pictures 25/ graphs / appendix / listings 14
Scientific field, SF :	Electrical and Computer Engineering
Scientific discipline, SD :	Software Engineering
Subject / Keywords, S/KW :	Gymnastics, constraints, communication, planning, schedule
UDC	
Holding data, HD :	Library of the Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad
Note, N :	
Abstract, AB :	Planning problems have been described, along with how OptaPlanner is used to model and solve them by defining a series of constraints. Additionally, attention is given to the implementation of a web socket server, which enables real-time evaluation of gymnasts.
Accepted by sci. Board on, ASB :	
Defended on, DE :	
Defense board, DB :	
president	Branko Milosavljević, PhD, full. prof, FTN Novi Sad
member	Miroslav Zarić, PhD, assoc. prof, FTN Novi Sad
mentor	Goran Sladić, PhD, full. prof., FTN Novi Sad
Mentor's signature	

Садржај

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА.....	5
KEY WORDS DOCUMENTATION.....	7
1. Увод.....	11
2. Проблеми планирања.....	13
2.1 Комплексност проблема планирања.....	14
2.2 Чврста и мека ограничења.....	15
2.3 Врсте решења проблема планирања.....	15
2.4 Библиотеке за решавање проблема планирања.....	16
3. Клијент - сервер комуникација у реалном времену.....	19
3.1 <i>Web socket</i> протокол.....	19
3.2 Конкурентно програмирање у Го програмском језику.....	20
4. Апликација за организацију и спровођење гимнастичких такмичења.....	23
4.1 Архитектура апликације.....	24
4.2 Модел података.....	25
4.2.1 Сервис за аутентификацију и ауторизацију.....	25
4.2.2 Сервис за креирање и пријављивање на такмичења.....	27
4.2.3 Сервис за креирање и оптимизацију распореда такмичара по справама.....	29
4.2.4 Сервис за оцењивање такмичара.....	31
5. Имплементација апликације.....	35
5.1 Креирање и оптимизација распореда.....	35
5.1.1 Модел проблема.....	35
5.1.2 Аотирање модела <i>OptaPlanner</i> аотацијама.....	36
5.1.3 Дефинисање ограничења.....	41
5.1.4 Проблем заглављивања у локалном оптимуму.....	43
5.2 Оцењивање такмичара у реалном времену.....	47
5.2.1 Кориштене структуре података.....	48
5.2.2 Комуникација помоћу <i>web socket</i> а.....	50
6. Демонстрација.....	55
7. Закључак.....	67
ЛИТЕРАТУРА.....	69
БИОГРАФИЈА.....	71

1. Увод

Гимнастика, као веома изазован и елегантан спорт, обухвата широк спектар категорија и правила за оцењивање, што захтева прецизност и посвећеност како такмичара, тако и судија. Судије морају пажљиво пратити сваки аспект изведбе, укључујући технику, креативност и вештину, како би донеле најтачније оцене. Овај ручни процес захтева велику концентрацију и сваки додатни посао попут прослеђивања и ручног сабирања оцена представља оптерећење по судије. Ручни унос оцена и њихова даља обрада подложни су грешкама, што је недопустиво у професионалном спорту, и одузимају много времена, што доводи до фрустрације такмичара и њихових тренера док беспотребно чекају резултате. Пре самог такмичења потребно је обавити низ процеса попут објављивања такмичења, регистрације и пријаве такмичара и судија, планирања и формирања финалног распореда такмичара по справама. Нажалост, већина ових процеса тренутно се обавља ручно, што представља велики проблем и посао организаторима.

Циљ овог рада је да, уз помоћ савремених технологија, у што већој мери помогне дигитализацији и аутоматизацији ових процеса. Акценат је стављен на решавање проблема дефинисања распореда такмичења по справама који може да задовољи сва ограничења наметнута од стране гимнастичке организације. Додатно, такмичења захтевају да информације буду емитоване правовремено те је потребно одабрати адекватне технологије како би се задовољио овај захтев. Рад је организован у пет поглавља.

У другом поглављу објашњени су основни појмови проблема планирања и на који начин их специјализоване библиотеке решавају.

Треће поглавље садржи основне појмове комуникације између сервера и клијента у реалном времену и потребне технологије да би се она омогућила.

У четвртом поглављу описани су спецификација, модел и архитектура апликације за организацију и спровођење хуманстичких такмичења у коју су интегрисани претходно описани појмови оптимизације и комуникације у реалном времену.

У петом поглављу приказани су детаљи имплементације система. Објашњени су битни делови апликације и код којим је сама апликација имплементирана, а затим у шестом поглављу и случајеви коришћења карактеристични за ову апликацију.

2. Проблеми планирања

Организације се често суочавају са проблемима планирања попут обезбеђивања производа или услуга са скупом ограничених ресурса (запослени, средства, време и новац). Проблем планирања има оптималан циљ, заснован на ограниченим ресурсима и под специфичним ограничењима. Оптимални циљеви могу бити разнолики, на пример:

- Максимални профит – оптимални циљ резултује највећим могућим профитом
- Минимализован еколошки траг – оптимални циљ има најмањи утицај на животну средину
- Максимално задовољство запослених или купаца – оптимални циљ даје приоритет потребама запослених или купаца. [1]

Могућност постизања ових циљева зависи од броја расположивих ресурса, као што су:

- Број људи
- Количина времена
- Буџет
- Физичка средства, на пример, машине, возила, рачунари, зграде итд.

Морају се узети у обзир и специфична ограничења везана за ове ресурсе, као што је број сати које особа ради, њихова способност да користе одређене машине или компатибилност између делова опреме.

2.1 Комплексност проблема планирања

Недетерминистички проблеми [2] су проблеми у области рачунарских наука који се односе на ситуације у којима постоји више могућих решења и систем не може дефинитивно одабрати једно решење као "тачно" или "нетачно". У недетерминистичким проблемима, систем има могућност да истовремено разматра и исцрпи више различитих путања или решења.

На пример, недетерминистички проблем може бити проблем проверавања да ли постоји подниз дужине k у низу целих бројева чији збир је једнак неком задатом броју. Овакви проблеми могу бити захтевни за решавање јер систем мора истовремено разматрати све могуће комбинације поднизова.

У теорији комплексности, НП-комплетни проблеми [2] су најтежи проблеми у класи НП (недетерминистички, са полиномијалним временом) у смислу да су најмања подкласа НП, која би евентуално могла да остане изван класе P ¹. Разлог је што би детерминистичко решење било ког НП-комплетног проблема у полиномијалном времену било такође решење сваког проблема из класе НП. Класа комплексности која се састоји од свих НП-комплетних проблема се понекад назива НП-Ц.

Један пример НП-комплетног проблема је проблем збира подскупа, који гласи: ако је дат скуп целих бројева, одредити да ли постоји непразан подскуп овог скупа са збиром елемената нула. Ако имамо претпостављени одговор (подскуп), врло је лако проверити да ли му је збир нула, али није познат значајно бржи алгоритам за решавање овог проблема осим да се испроба сваки могући подскуп, што је врло споро.

Импликација овога је да ће решавање проблема бити много теже него што је очекивано, због тога што две честе технике неће бити довољне:

- Испитивање свих могућих опција одузеће превише времена

¹ Још увек је отворено питање да ли су класе P и НП једнаке

- Брзи алгоритам вратиће решење које је далеко од оптималног. На пример при паковању предмета у контејнере, смештајући прво највеће предмете, паковање ће брзо бити завршено, али са далеко више искориштених контејнера него што је у суштини потребно.

Проблеми планирања могу се сврстати у недетерминистичке проблеме из разлога што може постојати више различитих решења која су подједнако добра. Такође, припадају и групи НП проблема, јер је њихов простор могућих решења огроман, а с друге стране се врло лако може проверити да ли је решење валидно.

2.2 Чврста и мека ограничења

Обично, проблем планирања има барем два нивоа ограничења. Чврста (енгл. hard constraint) ограничења су ограничења која не смеју бити прекршена (на пример један професор не може предавати два предавања истовремено), док мека (енгл. soft constraint) ограничења не би требала бити прекршена уколико је то могуће (на пример професор не воли да предаје петком у подне) или би требала бити испуњена ако је то могуће у случају позитивних меких ограничења (професор воли да предаје понедељком ујутру). Такође, могуће је направити категоризацију унутар ове две групе ограничења у односу на приоритет ограничења (чврсто 1, чврсто 2, меко 1, меко 2, меко 3) где би испуњење ограничења вишег ранга имало већи приоритет.

2.3 Врсте решења проблема планирања

Проблеми планирања могу бити изазовни за решавање због огромног броја могућих решења које треба размотрити. Постоји неколико категорија решења на које можемо наићи[1]:

- Могуће решење је било које решење, без обзира да ли крши било који број ограничења. Проблеми планирања обично имају невероватно велики број могућих решења. Многа од тих решења су безвредна.
- Изводљиво решење је решење које не крши никаква (негативна) чврста ограничења. Број изводљивих решења тежи да буде релативан у односу на број могућих решења. Понекад

нема изводљивих решења. Свако изводљиво решење је могуће решење.

- Оптимално решење је решење са највећом оценом. Проблеми планирања обично имају једно или неколико оптималних решења. Увек постоји најмање једно оптимално решење, чак и у случају да нема изводљивих решења, а оптимално решење није изводљиво.
- Најбоље пронађено решење је решење са највећим резултатом које је пронашла имплементација у датом временском периоду. Најбоље пронађено решење ће вероватно бити изводљиво и, ако се има довољно времена, то је оптимално решење.

Проблем проблема планирања лежи у чињеници да је и за мали скуп података број могућих решења огроман и свака имплементација алгорита је приморана да прође бар кроз неки подскуп ових решења.

2.4 Библиотеке за решавање проблема планирања

OptaPlanner [3] је библиотека отвореног кода имплементирана у Јава програмском језику која је дизајнирана за решавање оптимизационих и проблема планирања. Обезбеђује скуп алгоритама за налажење најбољих могућих решења комплексних проблема распоређивања, доделе задатака и алокације ресурса. *OptaPlanner* користи приступ задовољења ограничења, где покушава да нађе решење које задовољава скуп ограничења док оптимизује одређени циљ. Ради тако што моделује проблем као скуп ентитета. Додатно, захтева дефинисање ограничења и захтева које решење мора испунити.

У даљем тексту, дат је општи опис решавања проблема планирања уз помоћ *OptaPlanner* библиотеке који укључује:

- Дефинисање модела специфичног за домен које укључује дефинисање ентитета, њихових својстава, веза и ограничења која се односе на проблем;
- Конфигурацију решавача са одговарајућим алгоритмима, стратегијама претраге и условима за завршетак;

- Израчунавање резултата. *OptaPlanner* рачуна резултат за свако потенцијално решење. Резултат представља колико добро решење испуњава циљеве и ограничења. Циљ је максимизовати или минимизовати овај резултат, у зависности од проблема;
- Процес претраге. *OptaPlanner* користи различите технике претраге да истражи простор решења и побољша резултат. Итеративно врши мале промене на тренутном решењу, процењујући утицај на резултат. Ако промена побољшава резултат, примењује се. Процес претраге наставља се док се не задовоље одређени услови завршетка. Ови услови могу бити максималан број итерација, временско ограничење или конкретан праг за резултат;
- Фина подешавања. *OptaPlanner* омогућава фино подешавање конфигурације решавача и ограничења како би се побољшао квалитет и ефикасност решења.

Узимајући у обзир разноврсне изазове проблема планирања, поред *OptaPlanner*-а, постоје и други алтернативни алати који могу бити корисни:

- *Google OR-Tools* [4] нуди широк спектар решавача за комбинаторичке оптимизације. Ова библиотека садржи имплементације алгоритама за линеарно програмирање, програмирање ограничења, целобројно и мешовито целобројно програмирање и графовске алгоритме. *OR-Tools* се користи за решавање проблема као што су распоређивање задатака, оптимално рутирање, распоређивање ресурса и других комбинаторичких изазова. Ова библиотека омогућава лако коришћење различитих алгоритама и решавача за различите типове задатака.
- *LocalSolver* [5] је комерцијални алат који се истиче хибридным приступом оптимизацији. Ова библиотека комбинује предности локалне претраге и математичког програмирања како би постигла ефикасно и тачно решење за сложене проблеме оптимизације. Коришћењем комбинације алгоритама, *LocalSolver* може савладати изазовне оптимизационе проблеме, укључујући динамичке задатке, задатке са великим бројем променљивих и ограничења, као и низ додатних изазова.

Иако све 3 горе наведене библиотеке теже истом циљу, оне до њега долазе на различите начине. *OptaPlanner* користи објектно оријентисани приступ приликом моделовања ограничења, где при томе нема ограничење које типове података сме да користи за дефинисање модела, док *LocalSolver* и *OR-tools* могу да баратају само са примитивним типовима, сетовима и листама што знатно ограничава њихове могућности моделовања. *OptaPlanner* нуди јединствени решавач који се може користити, за решавање било које врсте проблема, за разлику од *OR-tools* који поседује различите решаваче за различите проблеме, што повећава потребно време за обуку развојног тима. Битно је напоменути да је *OptaPlanner* бесплатан и веома лак за употребу, па се не мора размишљати о унајмљивању посебног тима за развој, о чему би се можда морало размислити приликом коришћења *OR-tools*-а, или плаћања лиценце, што је случај код *LocalSolver*-а. Поред свега овога, *OptaPlanner* се у пракси показао као најмање захтеван по ресурсе и много лакши за скалирање, него друге две библиотеке [6].

У пројекту је коришћена *OptaPlanner* библиотека као најфлексибилније решење од свих претходно наведених које захтева мањи напор за накнадне измене у случају потребе за проширењем дефиниција ограничења. Конкретан начин решавања проблема планирања у контексту апликације за организовање и спровођење гимнастичких такмичења биће описан у петом поглављу.

3. Клијент - сервер комуникација у реалном времену

У случајевима коришћења где је потребно да сервер обавести клијента у случају неког догађаја, класична HTTP (енгл. *HyperText Transfer Protocol*) комуникација која се заснива на захтеву клијента и одговору сервера није одговарајућа. У тим случајевима прелази се на друге видове комуникације попут комуникације помоћу *Web socket* протокола.

3.1 *Web socket* протокол

Web socket протокол [7] омогућава двосмерну комуникацију између клијента, са непоузданим кодом, и удаљеним сервером који је прихватио комуникацију. Безбедносни модел који се користи је модел заснован на пореклу [8] који често користе веб претраживачи. Циљ ове технологије је да обезбеди механизам за апликације унутар веб претраживача којима је потребна двосмерна комуникација са серверима, не ослањајући се на отварање више HTTP конекција. Протокол се успоставља "унапређивањем" класичне HTTP конекције. *Web socket* конекција траје све док је било који од учесника не прекине. Када једна страна прекине везу, друга страна неће моћи да комуницира јер се веза аутоматски прекида.



Слика 3.1 Успостављање и затварање web socket конекције¹

Будући да серверска апликација мора имати имплементирану логику за примање, обраду и слање порука прослеђених путем *web socket* протокола, битно је изабрати програмски језик који би омогућио конкурентан приступ овим захтевима. Један од језика који ово омогућује је Го (енгл. *Go* или *Golang*) [9] помоћу горутина и канала [10].

3.2 Конкурентно програмирање у Го програмском језику

Конкурентно програмирање у многим програмским језицима отежано је детаљима потребним за имплементацију исправног приступа дељеним ресурсима. Го подстиче другачији приступ у којем се дељени ресурси преносе кроз канале и, у ствари, никада активно не деле одвојене нити извршења. Само једна горутина има приступ ресурсу у било ком тренутку. Трке за ресурсима су спречене овим дизајном.

¹ Слика делимично преузета са <https://www.wallarm.com/what/a-simple-explanation-of-what-a-websocket-is>

Горутине носе такав назив јер постојећи термини попут нит, корутина, процес итд, преносе нетачне конотације. Горутина има једноставан модел: то је функција која се извршава истовремено са другим горутинама у истом адресном простору. Лагане су, коштају мало више од доделе простора на стеку. Горутине су мултиплексиране на више нити оперативног система, тако да ако се једна блокира, на пример док се чека на улаз или излаз, друге настављају да раде. Њихов дизајн крије многе сложености креирања нити и њиховог управљања.

Канал у Го програмском језику је ништа више него механизам за размену ресурса између различитих горутина. Горутина може да чита и да пише у канал. Операције читања и писања у канал су блокирајуће у смислу да део кода који је писао у канал неће наставити са извршавањем док нека друга горутина не прочита податак који је уписан у канал, аналогно томе, код који чита из канала неће наставити са извршавањем све док се у каналу из кога чита не нађе податак који може да прочита.

У контексту *web socket*-а, употреба горутина и канала представља ефикасан начин за постизање интерактивне комуникације између сервера и клијената.

На серверској страни, сваки клијент који се повезује добија своју посебну горутину која обрађује његове захтеве. Ово омогућава паралелну обраду многих клијентских конекција и убрзава реакцију сервера. Ове горутине се координирају помоћу канала, који омогућавају безбедну размену података између горутина. Када клијент пошаље поруку преко *web socket*-а, порука се ставља у канал који је повезан са рутином која се бави тим клијентом. Ово омогућава рутини да асинхроно обради поруку и одговори клијенту, док друге горутине на серверу настављају рад без прекида.

Примена ове комбинације рутина и канала доноси две битне користи. Прво, се сервер може ефикасно скалирати да обради велики број конекција, што је посебно важно код апликација које захтевају високо ниво реактивности. Друго, коришћење канала омогућава сигурну комуникацију између рутина, што спречава нежељене уплете података и потенцијалне грешке.

4. Апликација за организацију и спровођење гимнастичких такмичења

Апликација за организацију и спровођење гимнастичких такмичења треба да испрати читав животни век такмичења, од почетка његове организације, све до оцењивања и обраде резултата такмичара. То подразумева организовање такмичења, навођењем основних информација, као и информација које су неопходне за његово спровођење. Након тога, потребно је подржати пријаве такмичара и судија других клубова на то такмичење. Када се пријаве заврше, апликација треба да омогући креирање распореда такмичара по справама, као и распоред судија на судијским панелима. Након успешне расподеле такмичара и судија, судијама треба омогућити оцењивање такмичара у реалном времену и обраду крајњих резултата.

Будући да апликација треба да подржи читав процес организације и спровођења такмичења, функционалне захтеве можемо груписати у три дистинктне групе (слика 4.1, архитектура апликације):

- Регистрација спортских организација, њихових судија и такмичара, креирање такмичења и пријављивање на такмичења;
- Креирање и оптимизација распореда, на основу пријављених такмичара;
- Оцењивање такмичара, прослеђивање оцена и њихова обрада у реалном времену.

4.1 Архитектура апликације

Јасна подела функционалних захтева омогућила је и формирање микросервиса намењих за по једну од група. Апликацију чине следеће компоненте:

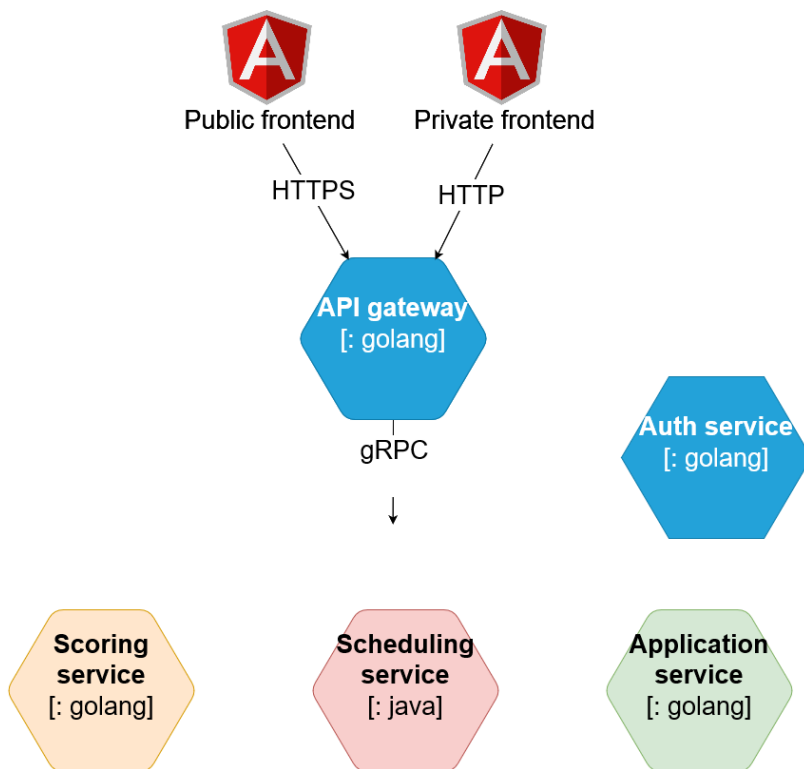
- Сервис за аутентификацију и ауторизацију
- Сервис за креирање и пријављивање на такмичења (енгл. *Application service*)
- Сервис за креирање и оптимизацију распореда такмичара по справама (енгл. *Scheduling service*)
- Сервис за оцењивање такмичара (енгл. *Scoring service*)
- *API* (енгл. *Application Programming Interface*) *gateway*
- Јавна клијентска апликација
- Приватна клијентска апликација

Микросервис за креирање и оптимизацију распореда је, због компатибилности са *OptaPlanner* библиотеком, имплементиран у Јавином *Spring Boot* [11] радном оквиру, док су сви остали микросервиси и *API gateway* имплементирани у Го програмском језику. Клијентске апликације имплементирани су уз помоћ *Angular* [12] радног оквира и библиотека *Angular Material* [13], библиотека корисничког интерфејса која садржи готову имплементацију дизајна компоненти корисничког интерфејса, и *RxJS* [14], библиотека која олакшава рад са асинхроним позивима.

Сваки од микросервиса има себи намењену базу података. Већина микросервиса користи се *PostgreSQL* [15] релационом базом података, док сервис за креирање и оптимизацију распореда своје податке складишти у *MongoDB* [16] документ бази. Сви микросервиси и базе података покрећу се унутар *Docker* [17] контејнера ради лакшег управљања и изолације.

Одлука за креирање јавне и приватне клијентске апликације настала је из разлога што функционалности за креирање и оптимизацију распореда и оцењивање такмичара не би требале да буду јавно доступне, већ само уређајима који се налазе у приватној, локалној мрежи спортске организације која организује такмичење. Више о овоме биће речено у петом поглављу.

Међусервисна комуникација обавља се помоћу *gRPC* [18] радног оквира за позивање удаљених процедура, док клијентске апликације са *API gateway*-ем комуницирају помоћу HTTP протокола.



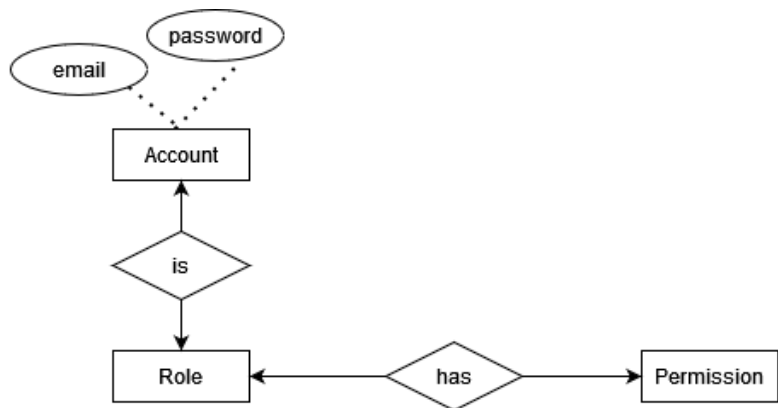
Слика 4.1 Архитектура апликације

4.2 Модел података

4.2.1 Сервис за аутентификацију и ауторизацију

Сервис за аутентификацију и ауторизацију користи се *Role Based Access Control* (RBAC) [19] моделом контроле приступа на нивоу дозвола. Овај ауторизациони модел изабран је због мноштва функционалности за које дозволу имају различити типови корисника, те се помоћу дозвола постиже много прецизнија и једноставнија дефиниција и контрола приступа.

Модел података је прилично једноставан и приказан је на слици 4.2. Кориснички налог садржи крeнцијале потребне за аутентификацију, а такође додељена му је рола која има одређене дозволе. Након аутентификације корисника интерни сервиси користе редукован скуп поља корисничког налога за потребе идентификације корисника који користи њихове функционалности, док се клијентским апликацијама прослеђује *Java Web Token* (JWT) токен [20].



{E_JUDGE, D_JUDGE, SPORTS_ORG}

Слика 4.2 Модел података сервиса за аутентификацију и ауторизацију

4.2.2 Сервис за креирање и пријављивање на такмичења

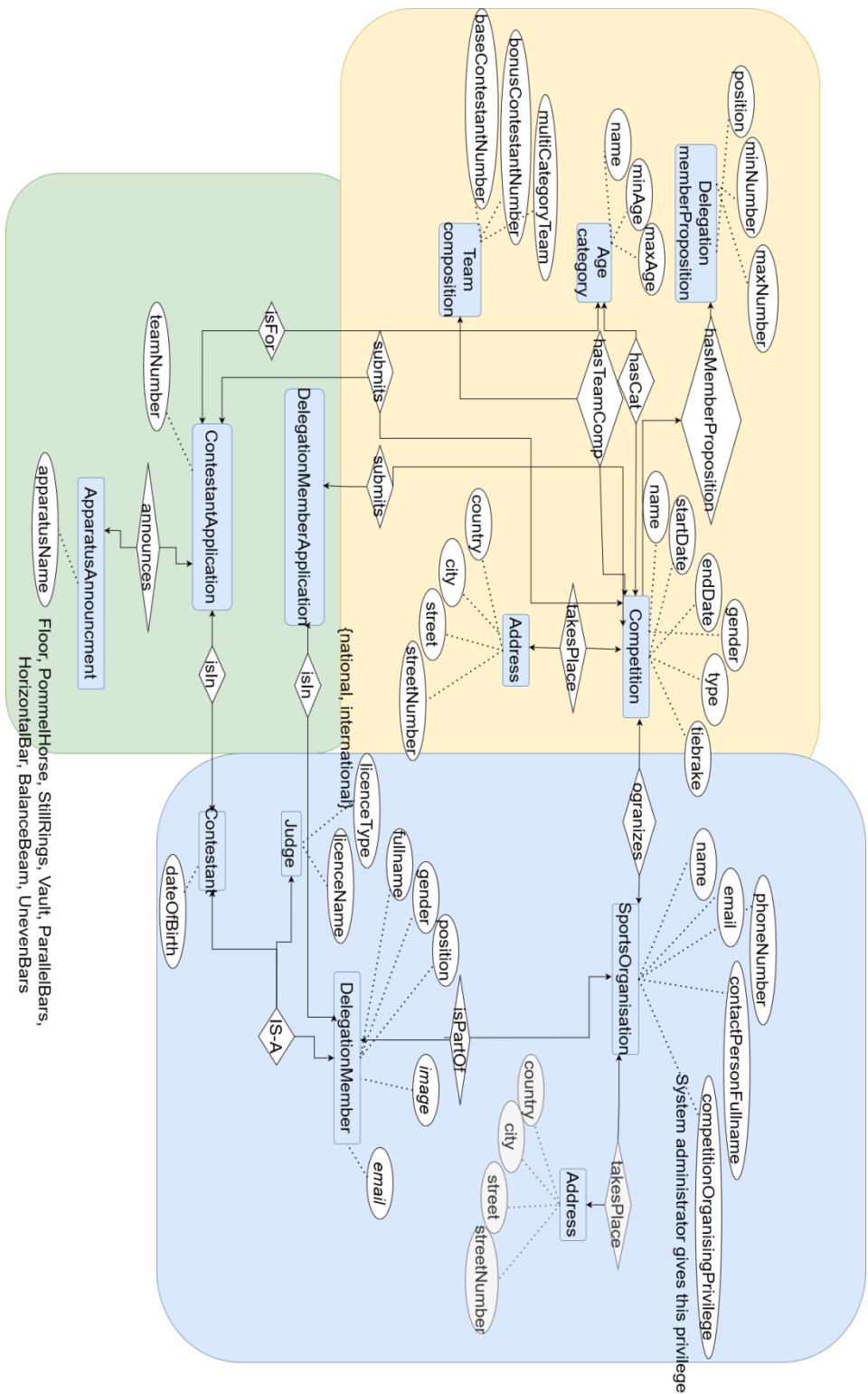
Унутар модела сервиса за креирање и пријављивање на такмичења, моделовани су сви ентитети, који се могу видети на слици 4.3, потребни за регистрацију спортских организација, њихових судија и такмичара, креирање нових такмичења и пријава судија и такмичара.

Спортска организација садржи основне информације и адресу на којој се налази.

Чланови спортске делегације припадају једној спортској организацији и они се даље могу поделити на судије, које поседују лиценцу за суђење, и такмичаре различитих годишта.

Спортска организација има могућност креирања такмичења. Такмичење садржи основне информације попут имена, датума и места одржавања, али и информације које прецизно дефинишу тип, организацију и начин спровођења такмичења. Такмичење може бити квалификационог карактера, тимско, појединачно или финале по справама. Једно такмичење може бити искључиво организовано за жене или мушкарце. Тај брејк особина такмичења биће објашњена у склопу имплементације сервиса за оцењивање. За свако такмичење дефинишу се старосне категорије и састав тимова, као и минимални и максимални бројеви одређених чланова делегације.

Такмичари и судије спортских организација пријављују се на такмичења путем такмичарске или судијске пријаве. Такмичари приликом пријаве најављују које ће справе такмичити и да ли ће бити у склопу тима спортске организације.

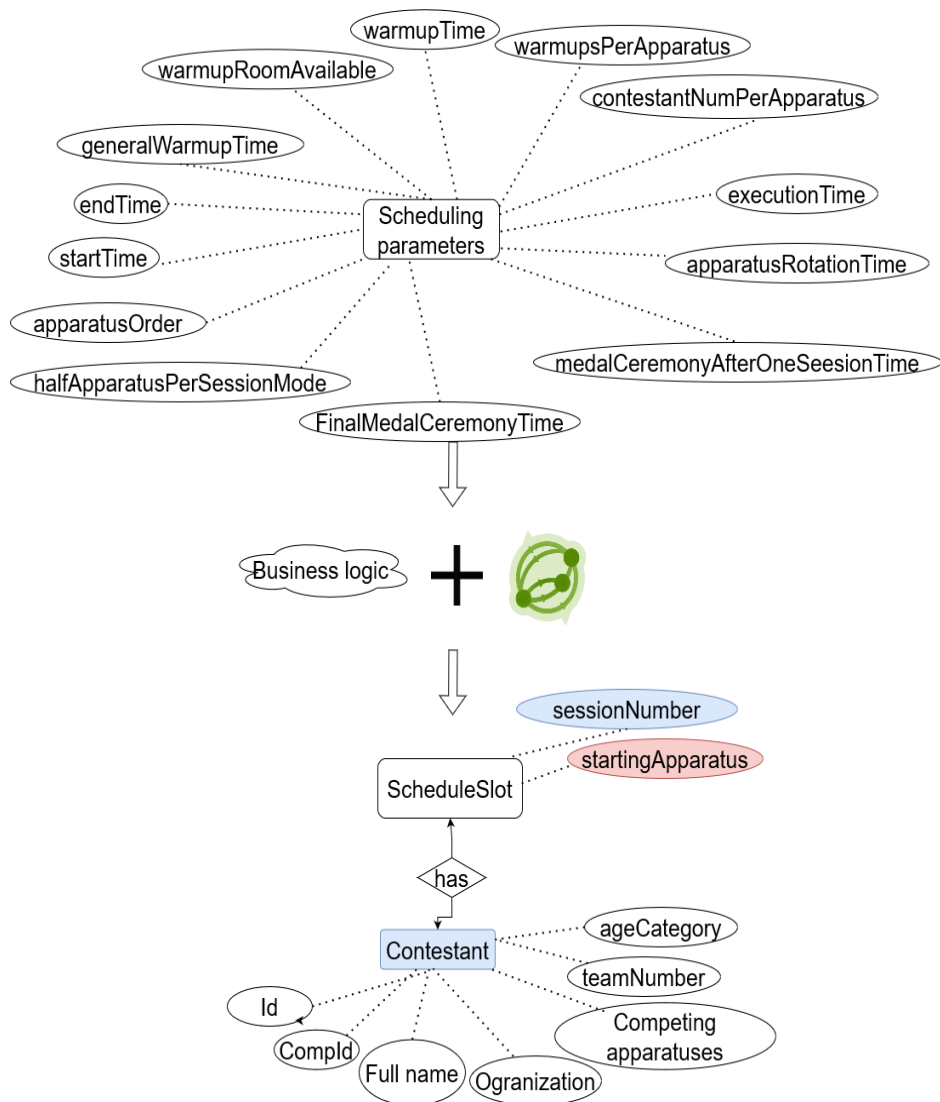


Слика 4.3 Модел података сервиса за креирање и пријављивање на такмичења

4.2.3 Сервис за креирање и оптимизацију распореда такмичара по справама

Улога сервиса за креирање и оптимизацију распореда је да на основу унетих параметара од стране организатора такмичења, уз помоћ *OptaPlanner* библиотеке, изгенерише основне градивне јединице распореда такмичара по справама, односно временске интервале који дефинишу на којој справи и у којој сесији (турнусу) одређени такмичар започиње такмичење. Модел овог сервиса приказан је на слици 4.4.

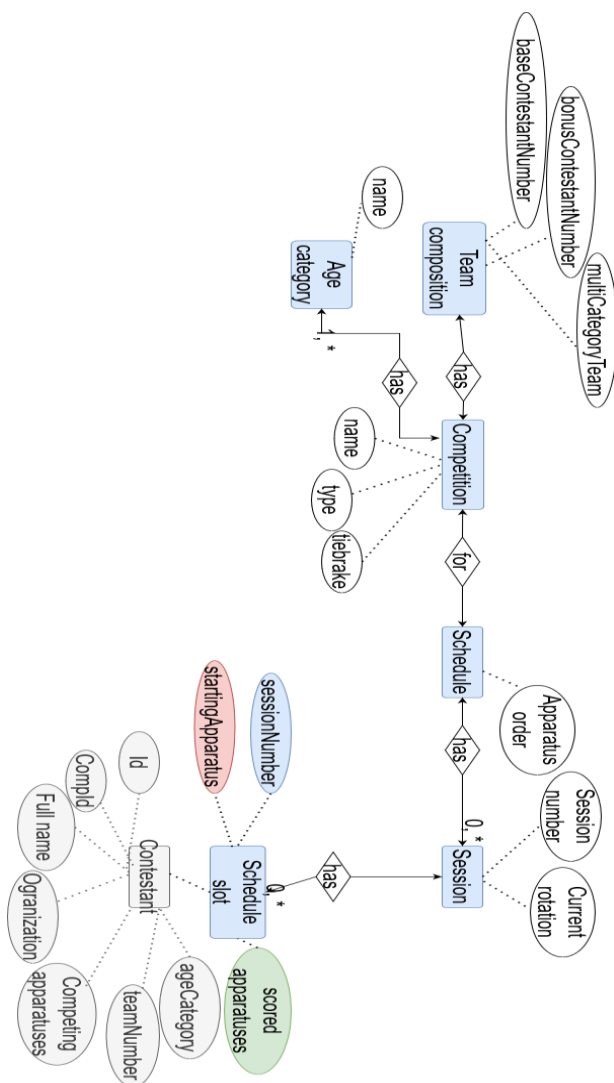
Делови модела које *OptaPlanner* користи за креирање и оптимизацију распореда биће објашњени у оквиру поглавља имплементације овог сервиса.



Слика 4.4 Модел података сервиса за креирање и оптимизацију распореда такмичара по справама

4.2.4 Сервис за оцењивање такмичара

Овај сервис се користи ентитетима претходно наведених сервиса. Информације о пријављеним такмичарима и судијама посуђују се из сервиса за прављење и пријављивање на такмичења. Распоред такмичара по справама преузима се из сервиса за креирање и оптимизацију распореда. Мора се напоменути да је број поља редукован на само она која су потребна за извршавање функционалности овог сервиса, као што је приказано на слици 4.5.



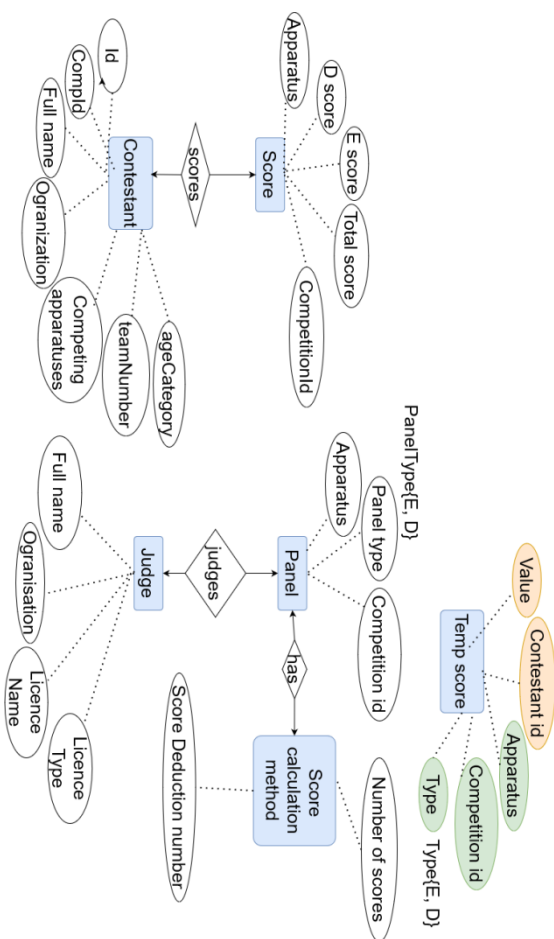
Слика 4.5 Модел података за оцењивање такмичара наслеђен од претходна два сервиса

Остатак модела, приказан на слици 4.6, односи се на ентитете потребне за оцењивање такмичара и обраду резултата у реалном времену. На почетку такмичења, креирају се Д и Е судијски панели

којима се додељује одређен број судија и дефинише се начин рачунања Е оцене.

Након наступа такмичара на справи, сваки од судија који се налазе у Д или Е панелу те справе додељује своју (привремену) оцену, које се, када све судије доделе оцену такмичару, прерачунавају у коначну оцену за ту справу.

Ранг листе такмичара се креирају агрегацијом коначних оцена такмичара по одређеним правилима.



Слика 4.6 Модел података за оцењивање такмичара везан за оцењивање и обраду резултата

5. Имплементација апликације

У наставку биће приказани неки од битнијих делова имплементације апликације за организацију и спровођење гимнастичких такмичења, описане у претходном поглављу.

5.1 Креирање и оптимизација распореда

5.1.1 Модел проблема

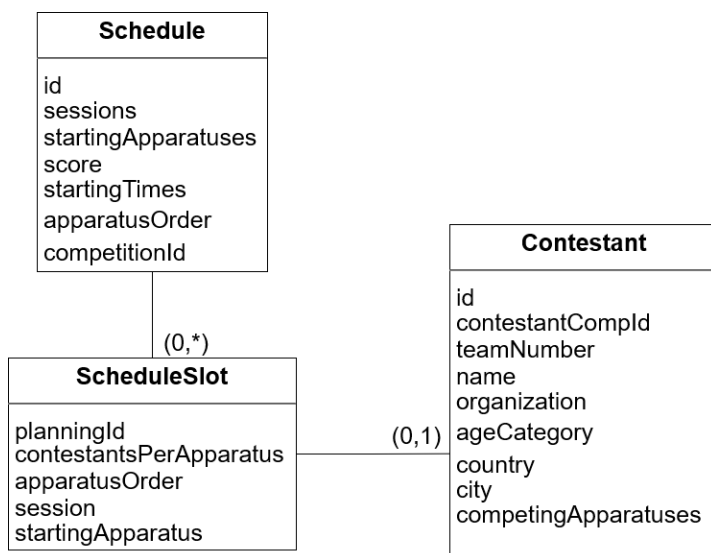
Пре почетка оптимизације требало би се упознати са самим проблемом (модел података проблема је приказан на слици 5.1).

Након завршених пријава такмичара, информације о такмичарима и справама на којим наступају прослеђују се сервису за креирање и оптимизацију распореда, где се прилагођавају моделу оптимизационог проблема (класа *Contestant*). Спрам броја пријављених такмичара креира се одговарајући број временских интервала (класа *ScheduleSlot*) којима се одмах додељује по један такмичар. Не може се десити да је један такмичар додељен у два временска интервала или да није додељен ниједном интервалу.

Класа такмичар (*Contestant*) садржи информације о такмичару које ће се даље користити за оптимизацију распореда такмичара по справама.

Класа временски интервал (*ScheduleSlot*) је централни део проблема оптимизације и она садржи такмичара, који је унапред додељен, а задатак оптимизационог проблема је да му додели почетну справу и сесију тако да сва ограничења буду максимално задовољена

тј. минимално прекршена. Такође, класа временски интервал садржи помоћна поља (*planningId*, *ApparatusOrder*) која омогућавају проверу ограничења о којима ће бити више речено у наставку поглавља.



Слика 5.1 Модел оптимизационог проблема

5.1.2 Анотирање модела *OptaPlanner* анотацијама

Да би могли користити *OptaPlanner* као алат за оптимизацију, морамо измоделовати наш проблем према одређеним правилима. Наиме, сваки проблем који *OptaPlanner* треба да реши треба да се састоји из неколико класа које су анотирани на одговарајући начин.

Будући да је класа *ScheduleSlot*, приказана у листингу 5.1, та која се оптимизује, она се анотира анотацијом *@PlanningEntity*. Класа анотирана са *@PlanningEntity* анотацијом у себи мора да садржи још неколико одговарајуће анотираних поља.

@PlanningId служи решавачу да јединствено идентификује сваки временски интервал и он се додељује на почетку оптимизације.

@PlanningVariable анотација стоји изнад поља које је потребно оптимизовати. На почетку оптимизације, решавач им додељује *null* вредности, а током процеса оптимизације им додељује разне

вредности из прослеђеног скупа могућих вредности, све док не дође до оптималног решења. У случају овог проблема поља која се оптимизују су почетна справа и сесија.

Поред наведених поља, класа временски интервал мора да имплементира методе *hashCode* и *equals* које ради прегледности нису приказане, а потребне су за функционисање решавача (користе се за поређење два временска интервала).

```
@PlanningEntity
public class ScheduleSlot {
    @PlanningId
    long planningId;
    Contestant contestant;

    //Помоћна поља
    int contestantsPerApparatus;
    List<ApparatusType> apparatusOrder;

    @PlanningVariable(nullable = false)
    Integer session;

    @PlanningVariable(nullable = false)
    ApparatusType startingApparatus;
}
```

Листинг 5.1 Класа *ScheduleSlot*

Класа *Schedule*, приказана у листингу 5.2, аотирана је аотацијом *@PlanningSolution* и представља решење оптимизационог проблема.

Помоћу аотација *@ValueRangeProvider* и *@ProblemFactCollectionProperty* назначавамо да ће поља *sessions* и *startingApparatuses* обезбедити податке са којима решавач може да ради. У овом случају обезбедиће све могуће вредности сесија и почетних справа. Информација о почетним справама добија се из прослеђених параметара од стране организатора и такође на основу тих параметара израчунава се максималан број сесија које могу да се реализују у задатом временском интервалу (листинг 5.3).

Аотацијом *@PlanningEntityCollectionProperty* назначавамо да ће поље *slots* представљати колекцију ентитета које треба оптимизовати.

Решавач ће за време оптимизације мапирати вредности из колекција *sessions* и *startingApparatuses* класе *Schedule* на поља *session* и *startingApparatus* свих објеката који се налазе у колекцији *slots* типа *ScheduleSlot*, све док не дође до оптималног решења.

Финално решење представљаће колекција *slots* чији ће сваки члан на крају оптимизације имати додељене одређене вредности сесије и почетне справе.

Аотацијом *@PlanningScore* аотирано је поље које означава резултат за дато решење које се израчунава на основу задатих ограничења о којима ће више бити речено у наставку поглавља. Битно је приметити да је ово поље типа *BendableScore* чиме, као што је наведено као могућност у поглављу 2.2, чврсте и меке резултате (који заједно формирају коначан резултат) додатно раслојавамо и то чврст на два, а мек резултат на три хијерархијска слоја. Као оптимално решење у овом случају узима се решење чија су оба слоја чврстог резултата једнака нули и уз то има максималне вредности на сваком слоју меког резултата.

```

@PlanningSolution
@Data
@NoArgsConstructor
@AllArgsConstructor
@Document
public class Schedule {
@Id
UUID id;
//Ствари које се мењају: сесија и почетна справа
// Наводи да је такмичара могуће само читати, не и
мењати
// ProblemFactCollectionProperties су доступне
ограничењима које користи решавач
@ValueRangeProvider
@ProblemFactCollectionProperty
@Transient
private List<Integer> sessions;

@ValueRangeProvider
@ProblemFactCollectionProperty
@Transient
private List<ApparatusType> startingApparatuses;

@PlanningEntityCollectionProperty
private List<ScheduleSlot> slots;

@PlanningScore(bendableHardLevelsSize = 2,
bendableSoftLevelsSize = 3)
@Transient
// Решење је изводљиво ако су сви чврсти резултати већи
или једнаки са 0
private BendableScore score;

//За одржавање редоследа
private List<Long> startingTimes;
private List<ApparatusType> apparatusOrder;
@Indexed
private UUID competitionId;}

```

Листинг 5.2 Класа *Schedule*

```

private int calculateMaxSessionNum(SchedulingParameters params) {
    long totalTime = Duration.between(params.getStartTime(),
        params.getEndTime()).toMinutes();

    long availableTime =
        totalTime - params.getFinalMedalCeremonyTime();
    int sessionTime = calculateSessionDuration(params);

    return (int) Math.floor((double) availableTime / sessionTime);
}

private int calculateSessionDuration(SchedulingParameters params){
    int generalWarmupTime;
    if(params.isWarmupRoomAvailable()){
        generalWarmupTime = 0;
    }
    else {
        generalWarmupTime = params.getGeneralWarmupTime();
    }

    int numOfApparatusesInSession = params.getApparatusOrder().size();

    int apparatusWarmupTime = params.getWarmupTime() *
        params.getWarmupsPerApparatus() *numOfApparatusesInSession;
    int executionTime = params.getExecutionTime() *
        params.getContestantNumPerApparatus() * numOfApparatusesInSession;
    int rotationTime = params.getApparatusRotationTime() *
        (numOfApparatusesInSession - 1);

    return generalWarmupTime + apparatusWarmupTime + executionTime +
        rotationTime + params.getMedalCeremonyAfterOneSessionTime();
}

```

Листинг 5.3 Код за израчунавање максималног могућег броја сесија

5.1.3 Дефинисање ограничења

OptaPlanner се служи *Java Streams API*-јем [21] да дефинише ограничења оптимизационог проблема на функционалан начин који је веома близак *Structured Query Language* (SQL) упитима. Како тачно ограничења функционишу најлакше је објаснити помоћу примера.

Решавач у примеру, приказаном у листингу 5.4, пролази кроз све комбинације временских интервала једног решења и проналази групе које задовољавају дефинисан упит (у овом случају, проналазе се све групе интервала који имају исту сесију и почетну справу, а њихов укупни број премашује дозвољен број такмичара на једној справи). За сваку пронађену групу, решавач смањује чврст резултат на првом слоју за један.

```
private Constraint
apparatusNumGreaterThanGiven(ConstraintFactory factory)
{
    return factory.forEach(ScheduleSlot.class)
        .groupBy( slot -> new CustomKey(slot.getSession(),
            slot.getStartingApparatus(),
            slot.getContestantsPerApparatus()), count())
        .filter((key, count) ->{
            return count > key.getContestantsNum();
        })
        .penalize(BendableScore.ofHard(
            BENDABLE_SCORE_HARD_LEVELS_SIZE,
            BENDABLE_SCORE_SOFT_LEVELS_SIZE,
            0,
            1
        ))
        .asConstraint("Apparatus number greater than given");
}
```

Листинг 5.4 Пример дефиниције чврстог ограничења

У примеру, приказаном у листингу 5.5, решавач проналази све парове интервала који имају исту почетну справу и такмичари долазе

из истог града. За сваки пронађени пар, решавач повећава меки резултат на другом слоју за два.

```
private Constraint
contestantsWithSameCityInSameSessionOnSameApparatus (Con
straintFactory factory) {
return factory.forEach(ScheduleSlot.class)
.join(ScheduleSlot.class,
equal(ScheduleSlot::getStartingApparatus,
ScheduleSlot::getStartingApparatus)
)
.filter(
(slot1, slot2) ->
slot1.getContestant().getCity().equals(slot2.getContest
ant().getCity())
)
.reward(BendableScore.ofSoft(
BENDABLE_SCORE_HARD_LEVELS_SIZE,
BENDABLE_SCORE_SOFT_LEVELS_SIZE,
1,
2
))
.asConstraint("Contestants from same city in same
session");
```

Листинг 5.5 Пример дефиниције меког ограничења

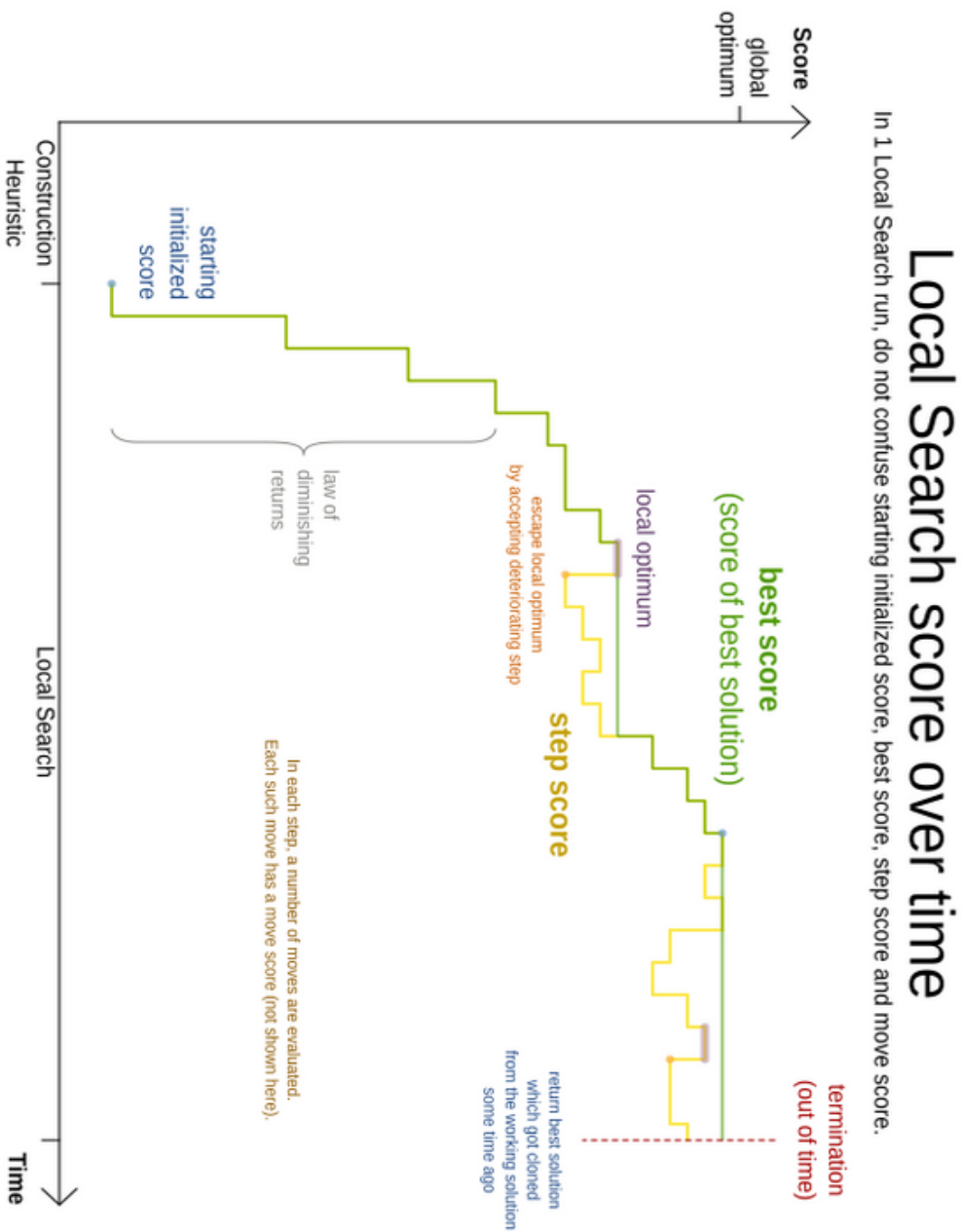
Поред ограничења наведених у примерима изнад, имплементирана су и следећа ограничења:

- Чврста ограничења
 - Такмичари из исте спортске организације морају започети такмичење на истој справи, ако се такмиче у истој сесији
- Мека ограничења
 - Такмичари исте старосне категорије би требали наступати у истој сесији
 - Прво би требали наступати млађи, а затим старији такмичари
 - Треба тежити да се прво у целости попуне сесије које почињу раније

- Ако такмичари нису из исте спортске организације, али долазе из истог града или државе, треба их покушати груписати
- Требало би минимализовати чекање такмичара који не такмиче све справе, односно да такмичар почне са справом таквом да ће у најмањем броју ротација успети да одвежба све своје справе и заврши са такмичењем

5.1.4 Проблем заглављивања у локалном оптимуму

Уколико другачије није конфигурисан, *OptaPlanner* користи подразумеване алгоритме за иницијализовање почетног (Конструкционе хеуристике [22]) и налажење оптималног решења. Као подразумевани алгоритам за проналажење оптималног решења користи се *Hill Climbing* алгоритам локалне претраге [23] који је склон заглављивању у локалном оптимуму и због тога често не успева да пронађе глобално оптимално решење. Из тог разлога, користе се алгоритми који имају могућност да, у тренутку када се заглаве у локалном оптимуму, покушају да истраже делове простора решења који су ван околине тог оптимума и на тај начин побегну из њега ако се испостави да је само локални оптимум, као што је приказано на слици 5.2.



Слика 5.2 Пример конвергирања алгоритма ка најбољем решењу¹

¹ Слика преузета са <https://www.optaplanner.org/docs/optaplanner/latest/local-search/local-search.html#localSearchConcepts>

Гимнастичка апликација је коришћењем подразумеваног алгоритма претраге успешно решавала проблеме који би укључивали до 30 такмичара и не више од 2 старосне категорије у задатом времену терминације од 15 секунди (листинг 5.6).

```
2023-09-03 09:29:28.500 INFO 1 --- [pool-2-thread-1]
o.o.core.impl.solver.DefaultSolver : Solving
ended: time spent (15000), best score ([0/0]hard/[-
96/768/0]soft), score calculation speed (11906/sec),
phase total (2), environment mode (REPRODUCIBLE), move
thread count (NONE).
```

Листинг 5.6 Излаз сервиса за оптимизацију распореда у случају 30 такмичара и 2 старосне категорије

Међутим, са повећањем броја такмичара и увођењем више старосних категорија (у конкретном случају 50 такмичара и 3 старосне категорије), решавач је захтевао све више времена и у неким случајевима се заглављивао у локалном оптимуму, иако би се ручном провером могло закључити да постоји решење оптималније од оног које је он пронашао, где би меки резултат другог степена достигао вредност 1432, уместо тренутних 1305 (листинзи 5.7 и 5.8).

```
2023-09-03 09:32:02.000 INFO 1 --- [pool-2-thread-1]
o.o.core.impl.solver.DefaultSolver : Solving
ended: time spent (15000), best score ([0/0]hard/[-
96/1305/0]soft), score calculation speed (10704/sec),
phase total (2), environment mode (REPRODUCIBLE), move
thread count (NONE).
```

Листинг 5.7 У случају 50 такмичара и 3 старосне категорије решавач се заглављује у локалном оптимуму

```
2023-09-03 10:05:17.000 INFO 1 --- [pool-2-thread-1]
o.o.core.impl.solver.DefaultSolver : Solving
ended: time spent (600000), best score ([0/0]hard/[-
96/1305/0]soft), score calculation speed (10704/sec),
phase total (2), environment mode (REPRODUCIBLE), move
thread count (NONE).
```

Листинг 5.8 Упркос повећаном времену терминације на 10 минута, решавач остаје заглављен у локалном оптимуму

Тренутна имплементација користи Табу претрагу [24] као алгоритам за проналажење оптималног решења. Табу претрага је локална претрага која одржава табу листу у којој се налазе интерни објекти које решавач користи за проналажење решења који су тренутно маркирани као табу и не смеју се привремено користити како би се избегло заглављивање у локалном оптимуму.

Након увођења табу претраге решавач успева у случају 50 такмичара и 3 старосне категорије да изађе из локалног оптимума и пронађе глобални (листинг 5.9).

```
2023-09-03 10:21:46.500 INFO 1 --- [pool-2-thread-1]
o.o.core.impl.solver.DefaultSolver      : Solving
ended: time spent (400000), best score ([0/0]hard/[-
96/1432/0]soft), score calculation speed (10608/sec),
phase total (2), environment mode (REPRODUCIBLE), move
thread count (NONE).
```

Листинг 5.9 Помоћу табу претраге решавач проналази глобални оптимум

5.2 Оцењивање такмичара у реалном времену

Природа проблема оцењивања такмичара у реалном времену, као што је описано у трећем поглављу, намеће потребу преласка са стандардног начина комуникације клијента и сервера по принципу захтев-одговор на комуникацију путем *web socket* протокола, због тога што би све судије и администратор такмичења требали да буду обавештени о приспећу нове оцено у самом тренутку њеног пристизања, без икаквог закашњења.

Решење овог проблема заснива се на *web socket* серверу управљаном догађајима чија ће имплементација бити описана у наставку поглавља.

5.2.1 Кориштене структуре података

Све структуре података у наставку текста приказане су у листингу 5.10.

Структура *Server* састоји се од колекције клијената који су успоставили *web socket* конекцију са сервером, канала за емитовање порука (енгл. *broadcast*), канала за регистрацију и канала за одјављивање клијената са сервера. Сервер такође садржи обрађивач догађаја помоћу кога на основу тренутног догађаја припрема одговарајући одговор.

Структура клијент садржи поља *id*, *Apparatus* и *CompetitionId* помоћу којих се јединствено идентификује и на основу којих сервер одређује да ли је њима одређена порука намењена. Канал *send* клијенту служи за примање порука које треба да пошаље клијентској апликацији помоћу успостављене *web socket* конекције.

Као што је већ поменуто код описа поља структуре *Client*, структуре *EventMessage* и *EventResponse* оба садрже поља *Apparatus* и *CompetitionId* којима се сервер служи да одреди којим клијентима су поруке и одговори на њих намењени.

Структура *EventMessage* садржи поља *Event* које носи информацију који догађај се десио и *ContestantId* који нам говори за ког такмичара је догађај везан.

Структура *EventResponse* садржи поља *Event* и *ContestantId* која имају исту улогу као и код структуре *EventMessage*, као и поље *Response* у коме је смештена структура података која представља одговор сервера на дати догађај.


```

type Server struct {
//bool вредност није битна, мапу користимо само због
једноставније претраге и брисања
clients map[*Client]bool
broadcast chan EventMessage
register chan *Client
unregister chan *Client
eventHandler *EventHandler
}

type EventHandler struct {
client scoring_pb.ScoringServiceClient
}

type Client struct {
id uuid.UUID
socket *websocket.Conn
send chan EventResponse
Apparatus Apparatus
CompetitionId string
}

type EventMessage struct {
Event Event `json:"event,omitempty"`
Apparatus Apparatus `json:"apparatus,omitempty"`
CompetitionId string `json:"competitionId,omitempty"`
ContestantId string `json:"contestantId,omitempty"`
}

type EventResponse struct {
Event Event `json:"event,omitempty"`
Apparatus Apparatus `json:"apparatus,omitempty"`
CompetitionId string `json:"competitionId,omitempty"`
ContestantId string `json:"contestantId,omitempty"`
Response interface{} `json:"response,omitempty"`
}

```

Листинг 5.10 Кориштене структуре података

5.2.2 Комуникација помоћу *web sockets*

Имплементација *web socket* сервера налази се у *API gateway*-у и покреће се позивом методе сервера *Start* унутар засебне горутине.

Улога методе *Start* (листинг 5.11) је да чека да подаци буду уписани у неки од канала сервера и реагује на одговарајући начин. Уколико је податак уписан у канале *register* или *unregister*, сервер уписује или брише клијента из колекције конектованих клијената. Ако је порука стигла на *broadcast* канал, сервер помоћу обрађивача захтева припрема одговарајући одговор за тренутни догађај наведен у поруци и помоћу методе *sendToAll*, приказане у листингу 5.12, прослеђује одговор клијентима којима је намењен.

```
func (server *Server) Start() {for {
    select {
        case client := <-server.register:
            server.clients[client] = true
        case client := <-server.unregister:
            if _, ok := server.clients[client];
        ok {
            close(client.send)
            delete(server.clients, client)
        }

        case message := <-server.broadcast:
            response :=
            server.PrepareResponse(&message)
            server.sendToAll(response)
        }
    }
}
```

Листинг 5.11 *Start* метода сервера

```

func (server *Server) sendToAll(response
*EventResponse) {
// Пошаљи само клијентима са истим competitionId и
истом почетном справом
for client := range server.clients {
//Ако администратор пошаље поруку, сви је добијају.
Администратор добија све поруке
if (client.CompetitionId == response.CompetitionId &&
(client.Apparatus == response.Apparatus ||
client.Apparatus == CompetitionAdmin)) ||
response.Apparatus == CompetitionAdmin {
select {
case client.send <- *response:
default:
close(client.send)
delete(server.clients, client)
}
}
}
}

```

Листинг 5.12 *sendToAll* метода сервера

На захтев клијентске апликације нова *web socket* конекција отвара се уз помоћ методе *OpenConnection*, приказане у листингу 5.13. Унутар ове методе, стандардна *HTTP* конекција унапређује се у *web socket* конекцију, креира се нови клијент и шаље на серверов *register* канал. Након регистрације клијента, покрећу се његове *read* и *write* методе у засебним горутинама које имају улогу примања и слања порука и одговора на њих клијентској апликацији.

```

func (server *Server) OpenConnection(ctx *gin.Context)
{
    competitionId := ctx.Query("competitionId")
    if competitionId == "" {
        return an error contestant
        ctx.JSON(400, gin.H{"error": "competitionId query
        parameter is missing"})
        return
    }

    apparatusStr := ctx.Query("apparatus")
    if apparatusStr == "" {
        // If "apparatusStr" is not provided in the query,
        return an error contestant
        ctx.JSON(400, gin.H{"error": "apparatus query parameter
        is missing"})
        return
    }
    apparatus, err := strconv.Atoi(apparatusStr)
    if err != nil {
        ctx.JSON(400, gin.H{"error": "invalid apparatus query
        parameter"})
        return
    }

    //Унапреди коннекцију у web socket duplex
    // Check origin решава CORS
    connection, err := (&websocket.Upgrader{CheckOrigin:
    func(r *http.Request) bool { return true
    }}).Upgrade(ctx.Writer, ctx.Request, nil)
    if err != nil {
        ctx.Status(http.StatusNotFound)
        return
    }

    client := &Client{
        id: uuid.New(),
        socket: connection,
        send: make(chan EventResponse),
        Apparatus: Apparatus(apparatus),
        CompetitionId: competitionId,
    }

    server.register <- client

    go client.read(server)
    go client.write()
}

```

Листинг 5.13 *OpenConnection* метода сервера

На самом крају, у листингу 5.14, приказан је део примера комуникације са сервером, вођен догађајима, од стране клијентске апликације.

```
if (event.type == "message") {
    switch(event.data.event){
        case ScoringEvent.RetrievedContestantsTempScores:
            this.parseTempScoresResponse(event.data.response)
            this.sendEvent(
                ScoringEvent.RetrievedContestantsTempScores);
            break;
        case ScoringEvent.RetrievedCanCalculate:
            this.canCalculateScore = event.data.response;
            break;
        case ScoringEvent.RetrievedScore:
            this.score = event.data.response;
            if(this.score?.submitted ?? false){
                this.contestantScored = true ;
                this.sendEvent(ScoringEvent.ScoredContestant);
            }
            break;
        case
            ScoringEvent.RetrievedNextCurrentApparatusContestant:
            if(!event.data.response.competingId){
                this.rotationFinished = true;
                break;
            }
        //Иницијализација података за наредног такмичара
        this.rotationFinished = false;
        this.currentContestant = event.data.response;
        this.tempScoreSubmitted = false;
        this.score = null;
        this.contestantScored = false;
        this.sendEvent(ScoringEvent.RetrievedNextCurrentApparatusContestant);
        break;
    }
}
```

Листинг 5.14 Пример комуникације клијентске апликације са сервером

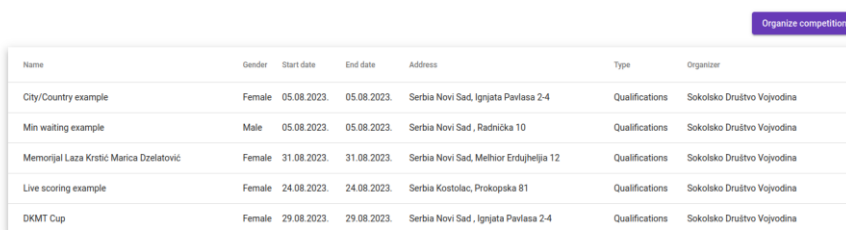
6. Демонстрација

У овом поглављу биће приказан читав ток рада апликације, од креирања такмичења, све до формирања финалних ранг листа такмичара.

Након што се администратор спортске организације пријави у јавну апликацију, у могућности је да прегледа све регистроване судије и такмичаре у тој спортској организацији, као и да види сва већ организована такмичења (слика 6.1).

Такође, администратор може да организује такмичење. На почетку уноси основне информације о такмичењу, попут назива, датума, адресе, као и информације о типу такмичења, начину формирања екипа и начину пресуде победника у случају истог крајњег резултата (слика 6.2). Затим креира пропозиције за састав делегације, односно минимални и максимални број судија и такмичара који се могу пријавити на такмичење у име једне спортске организације (слика 6.3). На крају креира старосне категорије што подразумева дефинисање назива, минималне и максималне старости такмичара у тој категорији (слика 6.4).

Competitions



Name	Gender	Start date	End date	Address	Type	Organizer
City/Country example	Female	05.08.2023.	05.08.2023.	Serbia Novi Sad, Ignjata Pavlase 2-4	Qualifications	Sokolsko Društvo Vojvodina
Min waiting example	Male	05.08.2023.	05.08.2023.	Serbia Novi Sad, Radnicka 10	Qualifications	Sokolsko Društvo Vojvodina
Memorijal Laza Krstic Marica Dzelatovic	Female	31.08.2023.	31.08.2023.	Serbia Novi Sad, Melhior Erduljelija 12	Qualifications	Sokolsko Društvo Vojvodina
Live scoring example	Female	24.08.2023.	24.08.2023.	Serbia Kostolac, Prokopska 81	Qualifications	Sokolsko Društvo Vojvodina
DKMT Cup	Female	29.08.2023.	29.08.2023.	Serbia Novi Sad, Ignjata Pavlase 2-4	Qualifications	Sokolsko Društvo Vojvodina

Слика 6.1 Приказ постојећих такмичења

Create competition

2 Add delegation member propositions

Name *	Kup Srbije
Start date *	10/27/2023
End date *	10/27/2023
Gender *	Male
Type *	Qualifications
<input checked="" type="checkbox"/> Tiebreak	
Country *	Serbia
City *	Belgrade
Street *	Nikole Tesle
Street number *	8
Base contestant number *	3
Bonus contestant number *	1
<input type="checkbox"/> Multi category team	
Create competition	
Add delegation member propositions	

Слика 6.3 Унос основних информација такмичења

Position *	Judge
Minimal number *	2
Maximal number *	4
Add	
Add age categories	

Added propositions:

- judge : min number: 2, max number: 4

Слика 6.2 Креирање пропозиција за састав делегације

The screenshot shows a web form with three input fields and two buttons. The first field is labeled 'Name *' and contains the text 'Juniors'. The second field is labeled 'Minimal age *' and contains the number '1'. The third field is labeled 'Maximal age *' and contains the number '16'. Below these fields are two purple buttons: 'Add' and 'Finish competition creation'.

Name *	Juniors
Minimal age *	1
Maximal age *	16
Add	
Finish competition creation	

Added age categories:

- Juniors : min age: 1, max age: 16

Слика 6.4 Креирање старосних категорија

Након што је такмичење организовано, постаје видљиво и осталим спортским организацијама које тада могу да пријаве своје судије и такмичаре на њега. Приликом пријаве такмичара наводи, се број екипе, старосна категорија и справе на којима ће се такмичар такмичити (слика 6.5).

Create contestant applications

Contestants

Full name
Mina Đukić
Jovana Petrović
Mila Stanković
Jelena Nikolić
Ivana Marković
Maja Stojanović
Ana Janković
Elena Đorđević

Team

Team number *

1

Age category

Name	Min age	Max age
Juniors		16
Seniors	16	100

Apparatuses

Name
Floor
UnevenBars
BalanceBeam
Vault

Слика 6.5 Пријава такмичара на такмичење

Када се рок за пријаву заврши, администратор спортске организације која организује такмичење може да креира распоред такмичара по справама унутар приватне апликације. Унутар форме, приказане на слици 6.6, уноси информације о временском трајању свих делова једне такмичарске сесије (турнуса), максималан број такмичара на једној справи и справе за које ће бити одржано такмичење. Помоћу ових информација сервис за креирање и оптимизацију распореда такмичара по справама генерише оптималан распоред. Примери генерисаног распореда приказани су на сликама 6.7 и 6.8.

Create schedule

Start time*	10 : 00
End time*	15 : 00
<input type="checkbox"/> Warmup room available	
General warmup time*	60
Warmup time*	3
Warmups per apparatus*	1
Contestant number per apparatus*	4
Execution time*	3
Apparatus rotation time*	1
Medal ceremony after one session time*	15
Final medal ceremony time*	0

Available apparatuses

Name

Chosen apparatuses

Name

Floor

UnevenBars

BalanceBeam

Vault

Clear selected apparatuses

Create

Слика 6.6 Унос параметара за креирање распореда

Schedule

Start competition

Session 1

Starting time: 10:00

Floor

Num.	Name	Team	Category	Organization
1	Mina Bukić	1	Juniors	Sokolako Društvo Vojvodina (Serbia, Novi Sad)
2	Jovana Petrović	1	Juniors	Sokolako Društvo Vojvodina (Serbia, Novi Sad)
3	Mila Stanković	1	Juniors	Sokolako Društvo Vojvodina (Serbia, Novi Sad)
4	Jelena Nikolić	1	Juniors	Sokolako Društvo Vojvodina (Serbia, Novi Sad)

UnevenBars

Num.	Name	Team	Category	Organization
9	Zsófia Nagy	1	Juniors	Magyar Sport Egyesület (Hungary, Budapest)
10	Lili Kovács	1	Juniors	Magyar Sport Egyesület (Hungary, Budapest)
11	Emília Varga	1	Juniors	Magyar Sport Egyesület (Hungary, Budapest)
12	Zsuzsanna Farkas	1	Juniors	Magyar Sport Egyesület (Hungary, Budapest)

Слика 6.7 Приказ генерисаног распореда 1

Session 2

Starting time: 12:18

Floor

Num.	Name	Team	Category	Organization
5	Elena Đorđević	1	Seniors	Sokolako Društvo Vojvodina (Serbia, Novi Sad)
6	Ana Janković	1	Seniors	Sokolako Društvo Vojvodina (Serbia, Novi Sad)
7	Maja Stojanović	1	Seniors	Sokolako Društvo Vojvodina (Serbia, Novi Sad)
8	Ivana Marković	1	Seniors	Sokolako Društvo Vojvodina (Serbia, Novi Sad)

UnevenBars

Num.	Name	Team	Category	Organization
13	Reka Oláh	1	Seniors	Magyar Sport Egyesület (Hungary, Budapest)
14	Katalin Papp	1	Seniors	Magyar Sport Egyesület (Hungary, Budapest)
15	Dóra Molnár	1	Seniors	Magyar Sport Egyesület (Hungary, Budapest)
16	Borbála Balogh	1	Seniors	Magyar Sport Egyesület (Hungary, Budapest)


BalanceBeam

Num.	Name	Team	Category	Organization
21	Harper Anderson	1	Seniors	Arizona Gymnastics (United States, Phoenix)

Слика 6.8 Приказ генерисаног распореда 2

Када је администратор задовољан генерисаним распоредом, може да започне такмичење. Пре почетка оцењивања судије се додељују Е и Д судијским панелима, као и правило за рачунање Е оцено за сваку справу понаособ. На слици 6.9 може се видети додељивање Д судија Д судијском панелу, док се на слици 6.10 може видети додељивање Е судија, као и дефинисање правила за рачунање Е оцене.

D panel



Full name	Email	Licence type	Licence name	Sports organisation
Dóra Kovács	gcotrainer+6@gmail.com	National	Amateur Licence	Magyar Sport Egyesület
Jelena Petrović	gcotrainer+1@gmail.com	International	Professional Licence	Sokolako Društvo Vojvodina
Milica Đukić	gcotrainer+2@gmail.com	National	Amateur Licence	Sokolako Društvo Vojvodina
Anastasija Ivanova	gcotrainer+9@gmail.com	International	Professional Licence	Спортивное Общество России
Ekaterina Petrova	gcotrainer+10@gmail.com	National	Amateur Licence	Спортивное Общество России

Assigned judges

Full name	Email	Licence type	Licence name	Sports organisation
Katalin Horváth	gcotrainer+5@gmail.com	International	Professional Licence	Magyar Sport Egyesület



6.9 Додељивање судија Д судијском панелу

E panel

Number of deducted E scores*

1

Assign

Full name	Email	Licence type	Licence name	Sports organisation
Anastasia Ivanova	gcotrener+9@gmail.com	International	Professional Licence	Спортивное Общество Россия
Ekaterina Petrova	gcotrener+10@gmail.com	National	Amateur Licence	Спортивное Общество Россия

Assigned judges

Full name	Email	Licence type	Licence name	Sports organisation
Dora Kovács	gcotrener+6@gmail.com	National	Amateur Licence	Magyar Sport Egyesület
Jelena Petrović	gcotrener+1@gmail.com	International	Professional Licence	Sokolsko Društvo Vojvodina
Milica Đukić	gcotrener+2@gmail.com	National	Amateur Licence	Sokolsko Društvo Vojvodina

Finish

Слика 6.10 Додељивање судија и правила за рачунање Е оцене

Приликом доделе судија панелима, судијама се аутоматски креира налог унутар приватне апликације, а шифра му се доставља путем електронске поште. Након тога, судије се могу улоговати и започети са оцењивањем такмичара, као што је приказано на сликама 6.12 и 6.13, док администратор може да надгледа генерално стање такмичења и завршава ротације и сесије, када је то могуће на начин приказан на слици 6.11.

Competition monitoring

Current session: 1, Current rotation: 1

Finish rotation

Finish session

Слика 6.11 Администраторов интерфејс за надгледање генералног такмичења

Contestant

26 Zsófia Nagy, Magyar Sport Egyesület

Submit score

E score*

9

Submit

Слика 6.12 Интерфејс за оцењивање Е судије

Contestant

26 Zsófia Nagy, Magyar Sport Egyesület

E scores

Judge	Value

D scores

Judge	Value

Submit score

D score*

4

Submit

Calculate score

Submit score

Слика 6.13 Интерфејс за оцењивање Д судије

Када Е или Д судија доделе оцену, она постаје видљива Д судијама те справе. Када све судије доделе своју оцену, Д судија може да израчуна и потврди финалну оцену такмичара на тој справи и након тога се прелази на оцењивање следећег такмичара. Изглед интерфејса Д судије непосредно пред прослеђивање финалне оцене може се видети на слици 6.14.

Contestant

26 Zsófia Nagy, Magyar Sport Egyesület

E scores

Judge	Value
Dóra Kovács	8.7
Milica Đukić	8.9
Jelena Petrović	9

Average E: 8.9

D scores

Judge	Value
Katalin Horváth	4

Average D: 4

Final score: 12.9

Calculate score

Submit score

Слика 6.14 Изглед интерфејса Д судије приликом калкулисања финалне оцене

Када се сви такмичари оцене на свим справама које такмиче, администратор такмичења означава завршетак такмичења и генеришу се финалне ранг листе такмичара. Овим се завршава процес организације и спровођења гимнастичког такмичења. Изглед финалне ранг листе за појединце може се видети на слици 6.15, док је на слици 6.16 приказана финална ранг листа за екипе.

Score boards

All around

JUNIORS

Place	Contestant	Organization	Floor	Uneven Bars	Balance Beam	Vault	Total
1	Jovana Petrović	Sokolsto Društvo Vojvodina (Novi Sad, Serbia)	13.9 (D:4.4 E:9.5)	13.7 (D:4.3 E:9.4)	13.400001 (D:4.3 E:9.1)	13.5 (D:4.1 E:9.4)	54.5
2	Екатерина Самарцова	Спортивное Общество России (Moscow, Russia)	13.6 (D:4.1 E:9.5)	13.5 (D:4.4 E:9.1)	13.7 (D:4.2 E:9.5)	13 (D:4.1 E:8.9)	53.8
3	Mina Đukić	Sokolsto Društvo Vojvodina (Novi Sad, Serbia)	13.9 (D:4.4 E:9.5)	13.400001 (D:4.3 E:9.1)	13.2 (D:4.2 E:9)	13.200001 (D:4.1 E:9.1)	53.7
4	Jelena Nikolić	Sokolsto Društvo Vojvodina (Novi Sad, Serbia)	13.6 (D:4.3 E:9.3)	13.599999 (D:4.2 E:9.4)	13 (D:4 E:9)	13.5 (D:4.1 E:9.4)	53.699997
5	Sophia Brown	Arizona Gymnastics (Phoenix, United States)	13.1 (D:4 E:9.1)	13.599999 (D:4.2 E:9.4)	13.425 (D:4.125 E:9.3)	13.4 (D:4.1 E:9.3)	53.524998
6	Анастасия Иванова	Спортивное Общество России (Moscow, Russia)	13.1 (D:4 E:9.1)	13.599999 (D:4.2 E:9.4)	13.9 (D:4.2 E:9.7)	12.799999 (D:3.9 E:8.9)	53.399996
7	Ava Smith	Arizona Gymnastics (Phoenix, United States)	13.3 (D:4.2 E:9.1)	13.700001 (D:4.1 E:9.6)	13.3 (D:4.2 E:9.1)	13 (D:4.1 E:8.9)	53.3
8	Olivia Williams	Arizona Gymnastics (Phoenix, United States)	13.6 (D:4.1 E:9.5)	12.9 (D:3.9 E:9)	13.225 (D:4.125 E:9.1)	13.5 (D:4.1 E:9.4)	53.225
8	Emma Johnson	Arizona Gymnastics (Phoenix, United States)	13.3 (D:4 E:9.3)	12.9 (D:3.9 E:9)	13.425 (D:4.125 E:9.3)	13.6 (D:4.3 E:9.3)	53.225

Слика 6.15 Изглед финалне ранг листе (појединци)

Team

JUNIORS

Place	Organization	Team	Floor	Uneven Bars	Balance Beam	Vault	Total
1	Sokolsto Društvo Vojvodina (Novi Sad, Serbia)	1	41.5	40.7	39.6	40.7	162.5
2	Arizona Gymnastics (Phoenix, United States)	1	40.2	40.199997	40.15	40.5	161.05
3	Спортивное Общество России (Moscow, Russia)	1	40	40.8	41.4	38.8	161
4	Magyai Sport Egyesület (Budapest, Hungary)	1	40.5	39.8	40.1	37.8	158.2

Слика 6.16 Изглед финалне ранг листе (екипе)

7. Закључак

У овом раду описана је реализација апликације за организацију и спровођење гимнастичких такмичења са нагласком на сервисе за креирање и оптимизацију распореда и комуникацију клијента и сервера у реалном времену. Апликација треба да потпомогне читав процес од самог креирања такмичења, пријаве такмичара и судија, преко креирања и оптимизације распореда такмичара по справама све до самог оцењивања такмичара и обраде њихових резултата.

Описано је шта су проблеми планирања и како се помоћу *OptaPlanner* библиотеке они моделују и дефинисањем низа ограничења решавају. Такође, посвећена је пажња имплементацији *web socket* сервера, који омогућава оцењивање гимнастичара у реалном времену.

На крају је приказана конкретна имплементација система и демонстрација његовог рада кроз читав процес организације и спровођења гимнастичких такмичења.

У даљем развоју апликације би се додатно могло поради на конфигурацији *OptaPlanner* решавача како би се убрзало конвергирање ка глобалном и спречило било какво заглављивање у локалном оптимуму као и додавање нових ограничења која би потпомогла прављењу оптималнијег решења. Поред техничких унапређења, увек постоји простор за унапређење корисничког искуства увођењем нових функционалности у апликацију, попут анализе и креирања статистика успешности сваког од такмичара на такмичењима или праћења њиховог здравственог стања.

ЛИТЕРАТУРА

- [1] OptaPlanner introduction
<https://www.optaplanner.org/docs/optaplanner/latest/planner-introduction/planner-introduction.html> (приступ: 2023-08-14)
- [2] Non-deterministic and NP complete problems Garey, M.R.; Johnson, D.S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness.
- [3] OptaPlanner
<https://www.optaplanner.org/docs/optaplanner/latest/planner-introduction/planner-introduction.html> (приступ: 2023-08-14)
- [4] Google OR tools <https://developers.google.com/optimization> (приступ: 2023-08-15)
- [5] LocalSolver <https://www.localsolver.com/> (приступ: 2023-08-15)
- [6] OptaPlanner competitors comparison
<https://www.optaplanner.org/competitor/> (приступ: 2023-09-01)
- [7] Web socket протокол <https://www.rfc-editor.org/rfc/rfc6455> (приступ: 2023-08-19)
- [8] Same-origin policy https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy (приступ: 2023-08-19)
- [9] Go <https://go.dev/doc/> (приступ: 2023-08-21)
- [10] Конкурентност у Go-у
https://go.dev/doc/effective_go#concurrency (приступ: 2023-08-21)
- [11] Spring Boot <https://spring.io/projects/spring-boot> (приступ: 2023-08-23)
- [12] Angular <https://angular.io/> (приступ: 2023-08-23)
- [13] Angular Material <https://material.angular.io/> (приступ: 2023-08-23)
- [14] RxJS <https://rxjs.dev/> (приступ: 2023-08-23)

- [15] PostgreSQL <https://www.postgresql.org/> (приступ: 2023-08-23)
- [16] MongoDB <https://www.mongodb.com/> (приступ: 2023-08-23)
- [17] Docker <https://www.docker.com/> (приступ: 2023-08-23)
- [18] gRPC <https://grpc.io/> (приступ: 2023-08-23)
- [19] RBAC <https://auth0.com/docs/manage-users/access-control/rbac> (приступ: 2023-08-25)
- [20] JWT <https://jwt.io/> (приступ: 2023-08-25)
- [21] Java streams API <https://www.javatpoint.com/java-8-stream> (приступ: 2023-08-227)
- [22] Construction heuristics
<https://www.optaplanner.org/docs/optaplanner/latest/construction-heuristics/construction-heuristics.html> (приступ: 2023-08-227)
- [23] Hill climbing local search
<https://www.optaplanner.org/docs/optaplanner/latest/local-search/local-search.html#hillClimbing> (приступ: 2023-08-227)
- [24] Tabu search
<https://www.optaplanner.org/docs/optaplanner/latest/local-search/local-search.html#tabuSearch> (приступ: 2023-08-29)

БИОГРАФИЈА

Александар Стојановић рођен је 8. новембра 2000. године у Новом Саду. Основну школу “Јован Јовановић Змај” у Сремској Каменици завршио је 2015. након чега уписује Гимназију “Јован Јовановић Змај” у Новом Саду коју завршава 2019. Исте године уписује се на Факултет техничких наука, смер Рачунарство и аутоматика. Школске 2021/22. се опредељује за усмерење Примењене рачунарске науке и информатика, након чега се школске 2022/23. опредељује за модул Интернет и електронско пословање. Положио је све испите предвиђене планом и програмом.