



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
У НОВОМ САДУ




Александар Стојановић

**Програмирање
ограничења и клијент-
сервер комуникација у
реалном времену**

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:

Програмирање ограничења и клијент-сервер комуникација у реалном времену

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 :	Програмирање ограничења и клијент-сервер комуникација у реалном времену
Језик публикације, ЈП :	Српски
Језик извода, ЈИ :	српски / енглески
Земља публикавања, 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 / страница / цитата / табела / слика / графикона / прилога
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 OptaPlannera они моделују и дефинисањем низа ограничења решавају. Такође, посвећена је пажња имплементацији web

	socket сервера, који омогућава оцењивање гимнастичара у реалном времену.
Датум прихватања теме, ДП:	
Датум одбране, ДО:	
Чланови комисије, КО:	
Председник	др Име Презиме, звање
Члан	др Име Презиме, звање
Ментор	проф. др Горан Сладић
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 :	Constraint programming and real time client-server communication
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 :	br. poglavlja / stranica / citata / tabela / slika / grafikona / priloga
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	Ime i prezime, zvanje na eng., PhD
member	Ime i prezime, zvanje na eng., PhD
mentor	Goran Sladić, PhD, full. prof., FTN Novi Sad
Mentor's signature	

Садржај

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА.....	4
KEY WORDS DOCUMENTATION.....	6
1. Увод.....	10
2. Проблеми планирања.....	12
2.1 Проблем планирања је НП-комплетан	12
2.2 Hard и Soft ограничења	14
2.3 Врсте решења проблема планирања.....	14
2.4 Како OptaPlanner решава проблеме планирања	15
3. Клијент - сервер комуникација у реалном времену.....	16
3.1 Web socket	16
3.2 Конкурентно програмирање у Go-у.....	17
4. Апликација за организацију и спровођење гимнастичких такмичења.....	19
4.1 Архитектура апликације.....	19
4.2 Модел података	21
4.2.1 Сервис за аутентификацију и ауторизацију	21
4.2.2 Сервис за креирање и пријављивање на такмичења.....	22
4.2.3 Сервис за креирање и оптимизацију распореда такмичара по справама.....	24
4.2.4 Сервис за оцењивање такмичара	25
5. Имплементација	27
5.1 Креирање и оптимизација распореда	27
5.1.1 Модел проблема.....	27
5.1.2 Анотирање модела OptaPlanner анотацијама	28
5.1.3 Дефинисање ограничења.....	32
5.1.4 Проблем загављивања у локалном оптимуму	34
5.2 Оцењивање такмичара у реалном времену.....	36
5.2.1 Кориштене структуре података	37
5.2.2 Отварање конекције и комуникација помоћу web socketa	38
6. Демонстрација.....	43
7. Закључак	53
ЛИТЕРАТУРА	55
БИОГРАФИЈА	57

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. Такође, ручни унос оцена и њихова даља обрада склони су грешкама које су недопустиве у професионалном спорту, одузимају много времена што доводи до фрустрације такмичара, њихових тренера и проблем по организацију такмичења. 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 обављају се 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.

У другом поглављу биће објашњени основни појмови проблема планирања и на који начин их специјализована библиотека OptaPlanner [1] решава.

У трећем поглављу биће објашњени основни појмови комуникације између сервера и клијента у реалном времену и потребне технологије да би се она омогућила.

У четвртном поглављу биће описана спецификација, модел и архитектура апликације за организацију и спровођење гимнастичких

такмичења у коју су интегрисани претходно описани појмови оптимизације и комуникације у реалном времену.

У петом поглављу ће бити описана конкретна имплементација система. Биће приказани 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

Свака организација се суочава са проблемима планирања: обезбеђивањем производа или услуга са ограниченим 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.

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 Проблем планирања је НП-комплетан

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 П (још увек се не зна да ли су класе П и НП једнаке). 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
- Брзи алгоритам, на пример у паковању у контејнер, стављајући прво највеће предмете, вратиће решење које је далеко од оптималног.

2.2 Hard и Soft ограничења

Обично, проблем планирања има барем два нивоа ограничења. Hard ограничења су ограничења која не смеју бити прекршена (на пример један професор не може предавати два предавања истовремено), док soft 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 soft ограничења (професор воли да предаје понедељком ујутру). Такође, могуће је направити категоризацију унутар ове две групе ограничења у односу на приоритет ограничења (hard 1, hard 2, soft 1, soft 2, soft 3) где би испуњење ограничења вишег ранга имало већи приоритет.

2.3 Врсте решења проблема планирања

Постоји неколико категорија решења:

- 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) hard 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 и за мали скуп података број могућих решења је огроман и свака имплементација алгорита је приморана да прође бар кроз неки подскуп ових решења.

2.4 Како OptaPlanner решава проблеме планирања

OptaPlanner [1] је библиотека отвореног кода у Јави 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, дефинишу се ограничења и захтеви које решење мора испунити.

У даљем тексту, дат је општи опис решавања проблема планирања уз помоћ OptaPlanner-а, док ће конкретан начин решавања проблема планирања у контексту апликације за организовање и спровођење гимнастичких такмичења бити описан у петом поглављу.

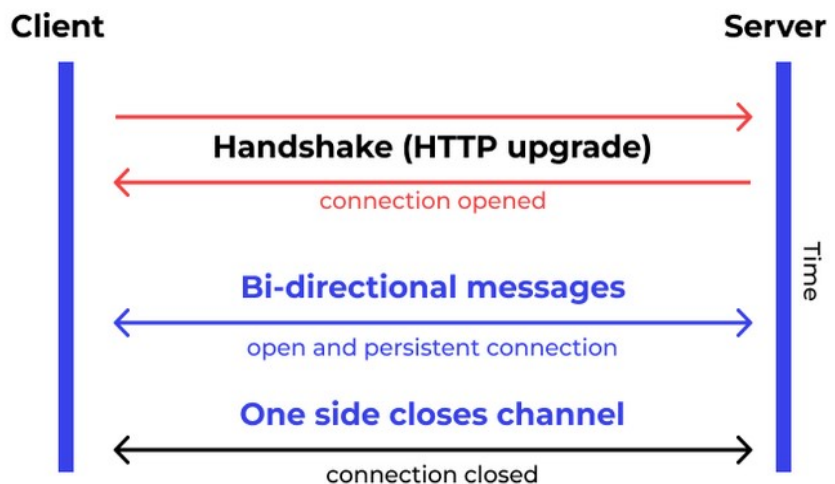
- Дефинисање модела специфичног за домен које укључује дефинисање ентитета, њихових својстава, веза и ограничења која се односе на проблем.
- Конфигурација решавача са одговарајућим алгоритмима, стратегијама претраге и условима за завршетак.
- Израчунавање резултата. OptaPlanner рачуна резултат за свако 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.

3. Клијент - сервер комуникација у реалном времену

У случајевима коришћења где је потребно да сервер обавести клијента у случају неког догађаја, класична HTTP 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-a.

3.1 Web socket

Web socket протокол [3] 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 [7] који често користе веб претраживачи. Циљ ове технологије је да обезбеди механизам за апликације унутар web 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 конекције [4]

Будући да серверска апликација мора имати имплементирану логику за примање, обраду и слање порука прослеђених путем web socket-a, битно је изабрати програмски језик који би омогућио конкурентан приступ овим захтевима. Један од језика који ово омогућује је Go [5] помоћу горутина и канала [6].

3.2 Конкурентно програмирање у Go-у

Конкурентно програмирање у многим програмским језицима отежано је детаљима потребним за имплементацију исправног приступа дељеним ресурсима. Go 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 превентоване овим дизајном.

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 Go 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.

4. Апликација за организацију и спровођење гимнастичких такмичења

Будући да би апликација требала да подржи читав процес организације и спровођења такмичења, функционалне захтеве можемо груписати у три дистинктне групе:

- 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 (Application service)
- Сервис за креирање и оптимизацију распореда такмичара по справама (Scheduling service)
- Сервис за оцењивање такмичара (Scoring service)
- API gateway
- Јавна клијентска апликација
- Приватна клијентска апликација.

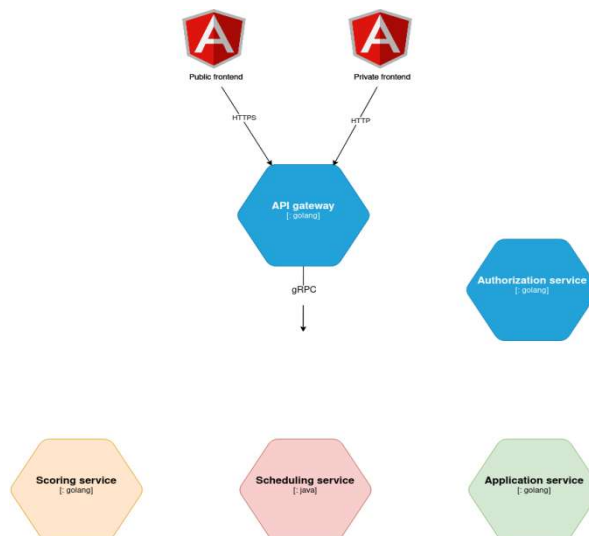
Микросервис за креирање и оптимизацију распореда је, због компатибилности са OptaPlanner-ом, имплементиран у Јавином Spring Boot [8] радном оквиру, док су сви остали микросервиси и API gateway имплементирани у Go-у. Клијентске апликације имплементиране су уз

помоћ Angular [9] радног оквира и библиотека Angular Material [10], библиотека корисничког интерфејса која садржи готову имплементацију дизајна компоненти корисничког интерфејса, и RxJS [11], библиотека која олакшава рад са асинхроним позивима.

Сваки од микросервиса има себи намењену базу података. Већина микросервиса користи се PostgreSQL [14] релационом базом података, док сервис за креирање и оптимизацију распореда своје податке складишти у MongoDB [15] документ бази. Сви микросервиси и базе података покрећу се унутар Docker [13] 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.

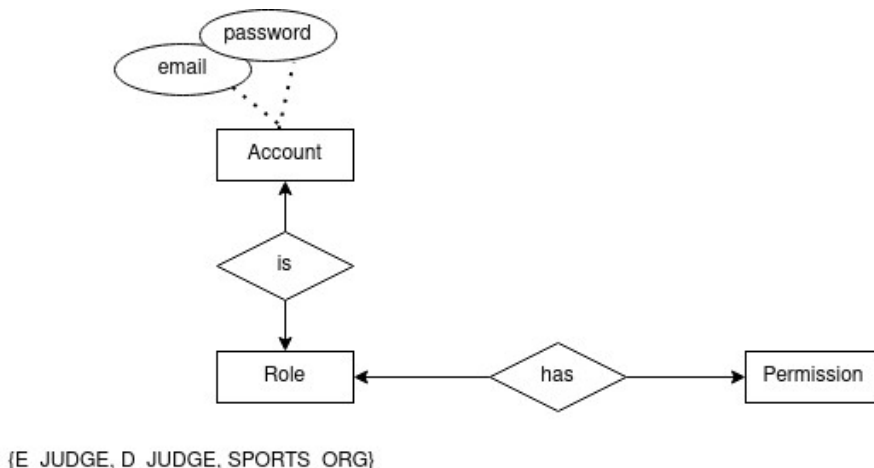
Међусервисна комуникација обавља се помоћу gRPC [12] радног оквира за позивање удаљених процедура, док клијентске апликације са API gateway-ем комуницирају помоћу HTTP протокола.



Слика 4.1 Архитектура апликације

4.2 Модел података

4.2.1 Сервис за аутентификацију и ауторизацију

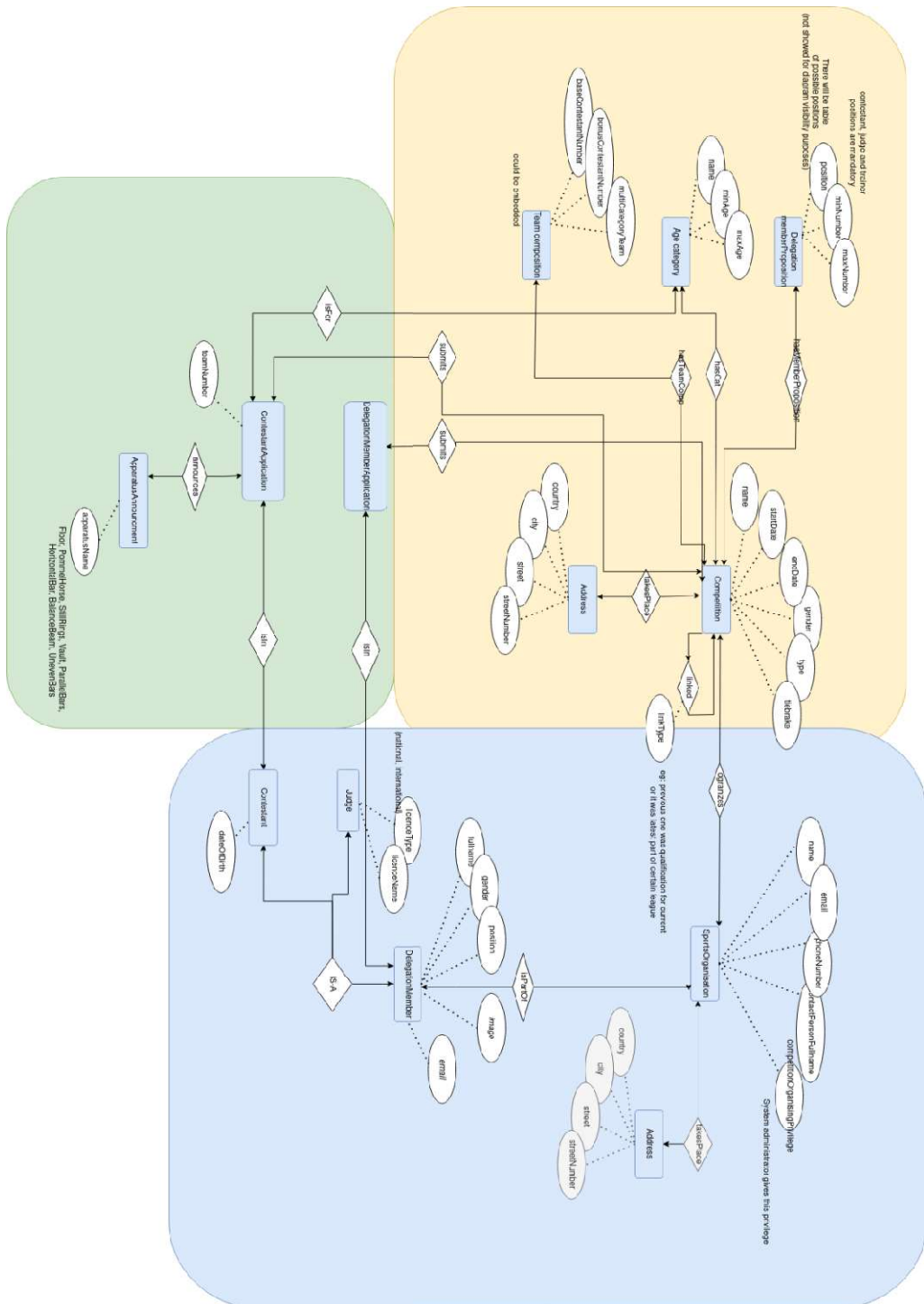


Слика 4.2 Модел података сервиса за аутентификацију и ауторизацију

Сервис за аутентификацију и ауторизацију користи се RBAC [16] моделом контроле приступа на нивоу пермисија. Овај ауторизациони модел изабран је због мноштва функционалности за које дозволу имају различити типови корисника, те се помоћу пермисија постиже много прецизнија и једноставнија дефиниција и контрола дозвола приступа.

Модел података је прилично једноставан. Кориснички налог садржи креденцијале потребне за аутентификацију, такође додељена му је рола која има одређене пермисије. Након аутентификације корисника интерни сервиси користе редукован скуп поља корисничког налога за потребе идентификације корисника који користи њихове функционалности, док се клијентским апликацијама прослеђује JWT токен [17].

4.2.2 Сервис за креирање и пријављивање на такмичења



Слика 4.3 Модел података сервиса за креирање и пријављивање на такмичења

Унутар овог модела, моделовани су сви ентитети потребни за регистрацију спортских организација, њихових судија и такмичара, креирање нових такмичења и пријава судија и такмичара на иста.

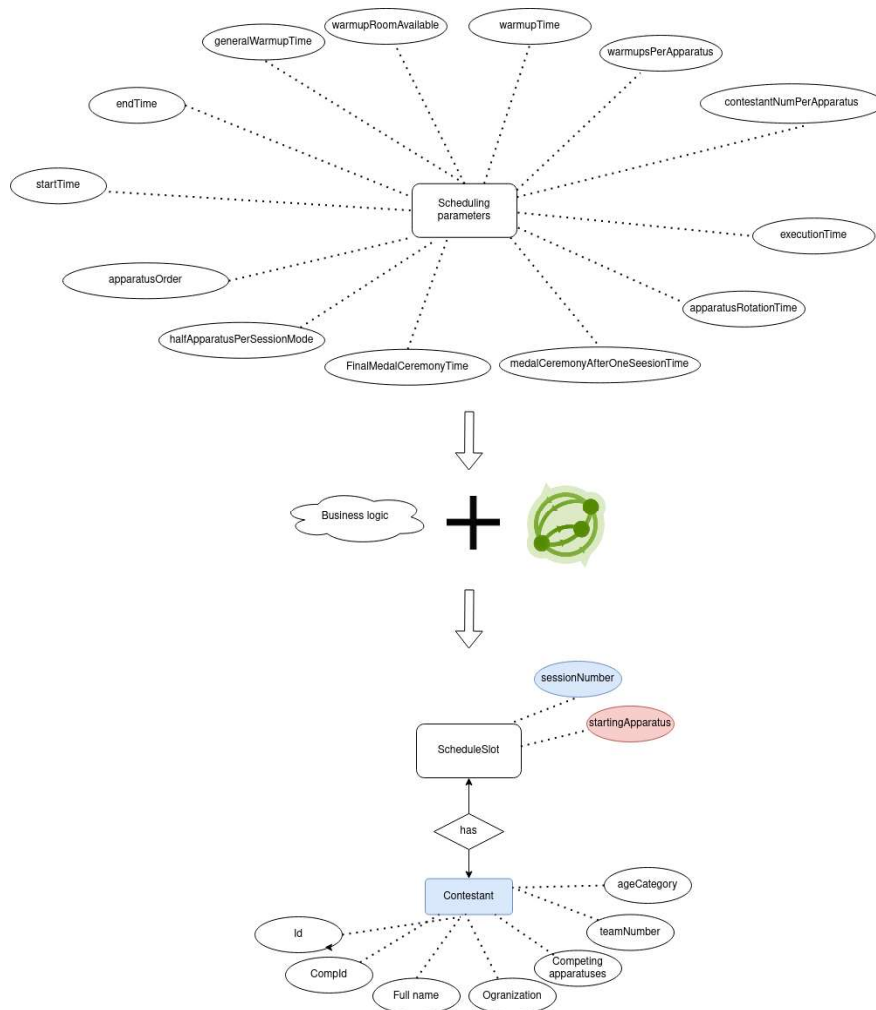
Спортска организација садржи основне информације и адресу на којој се налази.

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 правима. Једно такмичење може бити искључиво организовано за жене или мушкарце. Tiebreak 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.

4.2.3 Сервис за креирање и оптимизацију распореда такмичара по справама

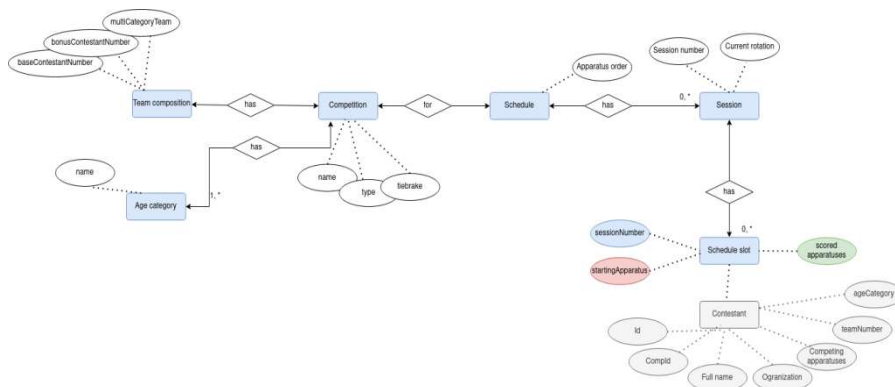


Слика 4.4 Модел података сервиса за креирање и оптимизацију распореда такмичара по справама

Улога овог сервиса је да на основу унетих параметара од стране организатора такмичења, уз помоћ OptaPlannera, 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.

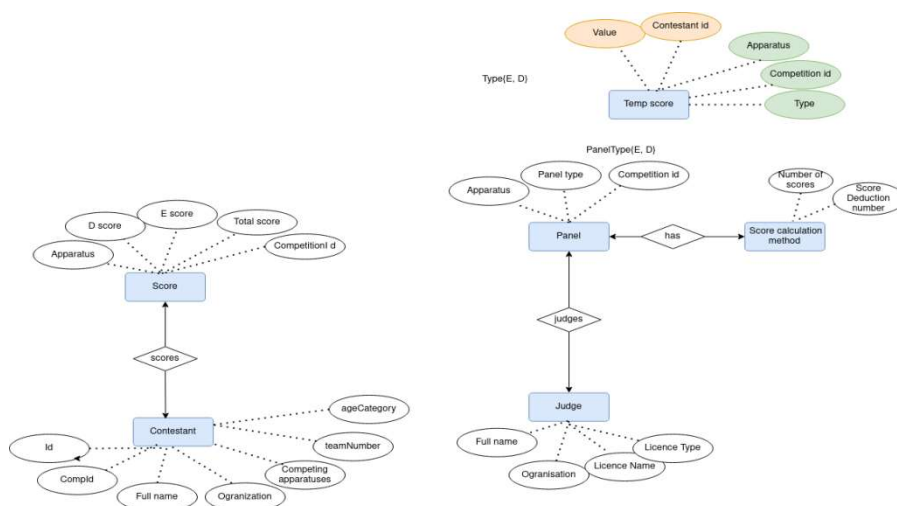
Делови модела које OptaPlanner користи за креирање и оптимизацију распореда биће објашњени у оквиру поглавља имплементације овог сервиса.

4.2.4 Сервис за оцењивање такмичара



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

Будући да овај сервис користи информације о пријављеним такмичарима и судијама из сервиса за креирање и пријављивање на такмичења, као и распоред такмичара по справама креиран у сервису за креирање и оптимизацију распореда такмичара по справама он наслеђује део ентитета из претходно наведених сервиса, с тиме да је број поља редукован на само она која су потребна за извршавање функционалности овог сервиса.



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

Остатак модела 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.

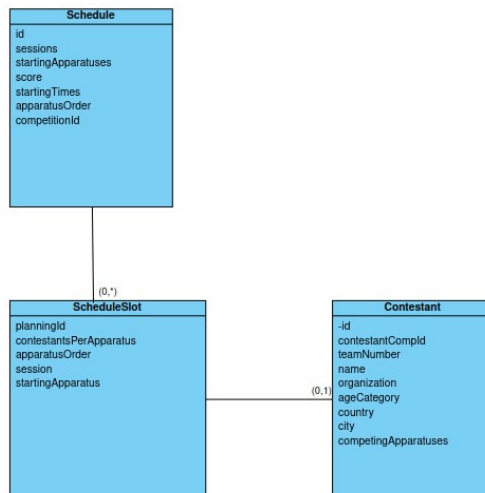
5. Имплементација

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.



Слика 5.1 Модел оптимизационог проблема

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 (`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.

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.

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

    //Helping fields
    int contestantsPerApparatus;
    List<ApparatusType> apparatusOrder;

    //Using Integer instead of int, because it is nullable
    @PlanningVariable(nullable = false)
    Integer session;

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

Листинг 5.1 Класа `ScheduleSlot`

Будући да је класа `ScheduleSlot` та која се оптимизује, она се аотира аотацијом `@PlanningEntity`. Класа аотирана са `@PlanningEntity` аотацијом у себи мора да садржи још неколико одговарајуће аотираних поља.

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

`@PlanningVariable` аотација 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.

```

@PlanningSolution
@Data
@NoArgsConstructor
@AllArgsConstructor
@Document
public class Schedule {
    @Id
    UUID id;
    //Things that change inside schedule: slot (session,
    apparatus)
    // States that contestant is readonly.
    // ProblemFactCollectionProperties are available to
    constraint streams
    @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
    // Solution is feasible if all hard score levels are at
    least 0
    private BendableScore score;

    //For persisting order when reading from database
    private List<Long> startingTimes;
    private List<ApparatusType> apparatusOrder;
    @Indexed
    private UUID competitionId;
}

```

Листинг 5.2 Класа Schedule

Класа Schedule анотирана је анотацијом @PlanningSolution и представља решење оптимизационог проблема.

Помоћу анотација `@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 задатаом временском интервалу.

```
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() *numOfApparatusesInSessi
on;
    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

Анотацијом `@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.

Анотацијом `@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, `Hard` и `Soft` резултате (који заједно формирају коначан резултат) додатно раслојавамо и то `Hard` на два, а `Soft` резултат на три хијерахијска слоја. Као оптимално решење у овом случају узима се решење чија су оба слоја `Hard` резултата једнака нули и уз то има максималне вредности на сваком слоју `Soft` резултата.

5.1.3 Дефинисање ограничења

`OptaPlanner` се служи `Java Streams API`-јем [18] да дефинише ограничења оптимизационог проблема на функционалан начин који је веома близак `SQL` упитима. Како тачно ограничења функционишу најлакше је објаснити помоћу примера.

```
private Constraint
apparatusNumGreaterThanGiven(ConstraintFactory factory)
{
    // Group by the number of contestants per apparatus
    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");
}
```


Решавач у овом примеру 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 Hard резултат на првом слоју за један.

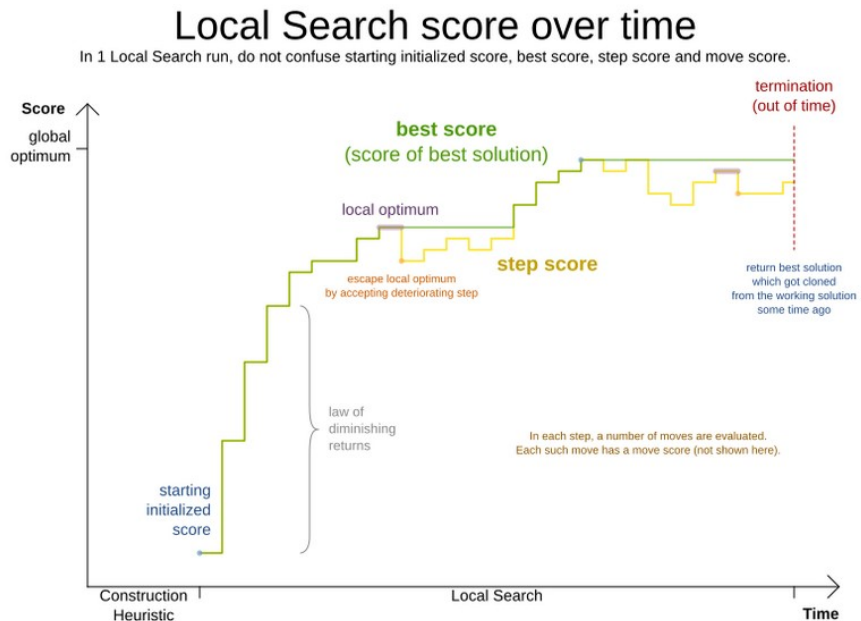
```
private Constraint
contestantsWithSameCityInSameSessionOnSameApparatus(Con
straintFactory factory) {
return factory.forEach(ScheduleSlot.class)
.join(ScheduleSlot.class,
equal(ScheduleSlot::getStartingApparatus,
ScheduleSlot::getStartingApparatus)
)
.filter(
(slot1, slot2) ->
slot1.getContestant().getCity().equals(slot2.getContest
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 Soft ограничења

У овом примеру решавач проналази све парове интервала који имају исту почетну справу и такмичари долазе из истог града. За сваки пронађени пар, решавач повећава Soft резултат на другом слоју за два.

5.1.4 Проблем заглављивања у локалном оптимуму

Уколико другачије није конфигурисан, OptaPlanner користи подразумеване алгоритме за иницијализовање почетног (Конструкционе хеуристике [19]) и налажење оптималног решења. Као



подразумевани алгоритам за проналажење оптималног решења користи се Hill Climbing алгоритам локалне претраге [20] 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.

Слика 5.2 Пример конвергирања алгоритма ка најбољем решењу [21]

Тренутна имплементација користи Табу претрагу [22] 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.

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-a, 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

```
type Server struct {
//bool value is irrelevant, we use map because of
convenient search and delete
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
//Used for filtering to which client response will be
sent
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 5.6 Кориштене структуре података

Структура `Server` састоји се од колекције клијената који су успоставили `web socket` конекцију са сервером, канала за емитовање порука (`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 помоћу `socket` успостављене `web socket` конекције клијентској апликацији.

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.

5.2.2 Отварање конекције и комуникација помоћу `web socketa`

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

```
func (server *Server) Start() {
    //Runs forever
    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)
        }
    }
}
```

Улога ове методе 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 прослеђује одговор клијентима којима је намењен.

```
func (server *Server) sendToAll(response
*EventResponse) {
// Send only to clients with same competitionId and
same apparatus
for client := range server.clients {
//If admin sends message everyone gets it, also admin
always gets every message
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 5.8 sendToAll метода сервера

На захтев клијентске апликације нова web socket конекција отвара се уз помоћ методе `OpenConnection`. 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 == "" {
        // If "apparatusStr" is not provided in the query,
        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
    }

    //Upgrade connection to web socket duplex
    //Check origin function is used for CORS (this one
    allows everything)
    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 5.9 OpenConnection метода сервера

На самом крају 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;
        }
        //Restarting all data for next contestant
        this.rotationFinished = false;
        this.currentContestant = event.data.response;
        this.tempScoreSubmitted = false;
        this.score = null;
        this.contestantScored = false;
        this.sendEvent(ScoringEvent.RetrievedNextCurrentApparatusContestant);
        break;
    }
}
```

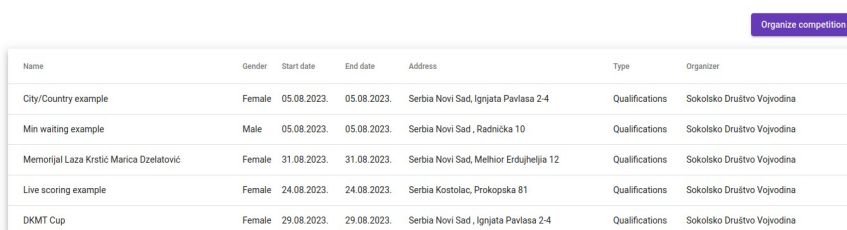
Листинг 5.10 Пример комуникације клијентске апликације са сервером

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.

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 Krsitic Marica Dzelatovic	Female	31.08.2023.	31.08.2023.	Serbia Novi Sad, Melhior Endujhelja 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

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. 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. На крају креира старосне категорије што подразумева дефинисање

назива, минималне и максималне старости такмичара у тој категорији.

Create competition

2 Add delegation member propositions

Name *	Kup Srbije
Start date *	10/27/2023
End date *	10/27/2023
Gender *	Male
Type *	Qualifications
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 6.2 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 6.3 Креирање пропозиција за састав делегације

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.

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

Create

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 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.

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 Đukić	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 администатор задовољан генерисаним распоредом, може да започне такмичење. Пре почетка оцењивања судије се додељују Е и Д судијским панелима, као и правило за рачунање Е оцене за сваку справу понаособ.

D panel

Assign

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	Sokolisko Društvo Vojvodina
Milica Đukić	gcotrainer+2@gmail.com	National	Amateur Licence	Sokolisko 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

Next

Слика 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 России
Ekaterina Petrova	gcotrener+10@gmail.com	National	Amateur Licence	Спортивное Общество России

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, док администратор може да надгледа генерално стање такмичења и завршава ротације и сесије, када је то могуће.

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 искалкулише и потврди финалну оцену такмичара на тој справи и након тога се прелази на оцењивање следећег такмичара.

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.

Score boards

All around

JUNIORS

Place	Contestant	Organization	Floor	Uneven Bars	Balance Beam	Vault	Total
1	Jovana Petrović	Sokolsko 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 Самарнова	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 Bukić	Sokolsko 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	Sokolsko 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)	53.399998
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	Sokolsko 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	Magyar 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 би требала 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 OptaPlannera 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
<https://www.optaplanner.org/docs/optaplanner/latest/planner-introduction/planner-introduction.html>
- [2] Недертиминистики проблеми и НП-комплетни проблеми
Garey, M.R.; Johnson, D.S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness.
- [3] Web socket протокол <https://www.rfc-editor.org/rfc/rfc6455>
- [4] Web socket конекција <https://www.wallarm.com/what-a-simple-explanation-of-what-a-websocket-is>
- [5] Go <https://go.dev/doc/>
- [6] Конкурентност у Go-у
https://go.dev/doc/effective_go#concurrency
- [7] Same-origin policy https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy
- [8] Spring Boot <https://spring.io/projects/spring-boot>
- [9] Angular <https://angular.io/>
- [10] Angular Material <https://material.angular.io/>
- [11] RxJS <https://rxjs.dev/>
- [12] gRPC <https://grpc.io/>
- [13] Docker <https://www.docker.com/>
- [14] PostgreSQL <https://www.postgresql.org/>
- [15] MongoDB <https://www.mongodb.com/>
- [16] RBAC <https://auth0.com/docs/manage-users/access-control/rbac>
- [17] JWT <https://jwt.io/>
- [18] Java streams API <https://www.javatpoint.com/java-8-stream>
- [19] Construction heuristics
<https://www.optaplanner.org/docs/optaplanner/latest/construction-heuristics/construction-heuristics.html>

- [20] Hill climbing local search
<https://www.optaplanner.org/docs/optaplanner/latest/local-search/local-search.html#hillClimbing>
- [21] Local search concepts
<https://www.optaplanner.org/docs/optaplanner/latest/local-search/local-search.html#localSearchConcepts>
- [22] Tabu search
<https://www.optaplanner.org/docs/optaplanner/latest/local-search/local-search.html#tabuSearch>

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.