



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



ЗБОРНИК РАДОВА ФАКУЛТЕТА ТЕХНИЧКИХ НАУКА

Едиција: Техничке науке - зборници

Година: XXXVIII

Број: 12/2023

Нови Сад

Едиција: „Техничке науке – Зборници“
Година: XXXVIII Свеска: 12

Издавач: Факултет техничких наука Нови Сад

Главни и одговорни уредник: проф. др Срђан Колаковић, декан Факултета техничких наука у Новом Саду

Уредништво:

Проф. др Срђан Колаковић
Проф. др Александар Купусинац
Проф. др Борис Думнић
Проф. др Дарко Стефановић
Проф. др Себастиан Балоиш
Проф. др Дејан Лукић
Проф. др Јован Дорић
Проф. др Мирослав Кљајић
Проф. др Немања Тасић
Проф. др Дејан Убавин

Проф. др Милан Видаковић
Проф. др Мирјана Дамњановић
Проф. др Јелена Атанацковић Јеличић
Проф. др Игор Пешко
Проф. др Драган Јовановић
Проф. др Небојша Ралевић
Доц. др Сања Ожват
Проф. др Немања Кашиковић
Проф. др Теодор Атанацковић

Редакција:

Проф. др Дарко Стефановић, главни уредник
Проф. др Жељен Трповски, технички
уредник

Проф. др Драгољуб Новаковић
Проф. др Иван Пинђер
Бисерка Милетић

Језичка редакција:

Бисерка Милетић, лектор
Софија Рацков, коректор
Мр Марина Катић, преводилац

Савет за библиотечку и издавачку делатност ФТН,
проф. др Стеван Станковски, председник.

Штампа: ФТН – Графички центар ГРИД, Трг Доситеја Обрадовића 6, Нови Сад

CIP-Каталогизација у публикацији
Библиотека Матице српске, Нови Сад

378.9(497.113)(082)
62

ЗБОРНИК радова Факултета техничких наука / главни и одговорни уредник
Срђан Колаковић. – Год. 7, бр. 9 (1974)-1990/1991, бр.21/22 ; Год. 23, бр 1 (2008)-. – Нови Сад : Факултет
техничких наука, 1974-1991; 2008-. – илустр. ; 30 цм. –(Едиција: Техничке науке – зборници)

Месечно

ISSN 0350-428X

COBISS.SR-ID 58627591

ПРЕДГОВОР

Поштовани читаоци,

Пред вама је дванаеста овогодишња свеска часописа „Зборник радова Факултета техничких наука“.

Часопис је покренут давне 1960. године, одмах по оснивању Машинског факултета у Новом Саду, као „Зборник радова Машинског факултета“, а први број је одштампан 1965. године. Након осам публикованих бројева у шест година, пратећи прерастање Машинског факултета у Факултет техничких наука, часопис мења назив у „Зборник радова Факултета техничких наука“ и 1974. године излази као број 9 (VII година). У том периоду у часопису се објављују научни и стручни радови, резултати истраживања професора, сарадника и студената ФТН-а, али и аутора ван ФТН-а, тако да часопис постаје значајно место презентације најновијих научних резултата и достигнућа. Од броја 17 (1986. год.), часопис почиње да излази искључиво на енглеском језику и добија поднаслов «Publications of the School of Engineering». Једна од последица нарастања материјалних проблема и несрећних догађаја на нашим просторима јесте и привремени прекид континуитета објављивања часописа двобројем/двогодишњаком 21/22, 1990/1991. год.

Друштво у коме живимо базирано је на знању. Оно претпоставља реорганизацију наставног процеса и увођење читавог низа нових струка, као и квалитетну организацију научног рада. Значајне промене у структури високог образовања, везане за имплементацију Болоњске декларације, усвајање нове и активне улоге студената у процесу образовања и њихово све шире укључивање у стручне и истраживачке пројекте, као и покретање нових мастер и докторских студија, доносе потребу да ови, веома значајни и вредни резултати, постану доступни академској и широј јавности. Оживљавање „Зборника радова Факултета техничких наука“, као јединственог форума за презентацију научних и стручних достигнућа, пре свега студената, обезбеђује услове за доступност ових резултата.

Због тога је Наставно-научно веће ФТН-а одлучило да, од новембра 2008. год. у облику пилот пројекта, а од фебруара 2009. год. као сталну активност, уведе презентацију најважнијих резултата свих мастер радова студената ФТН-а у облику кратког рада у „Зборнику радова Факултета техничких наука“.

Поред студената мастер студија, часопис је отворен и за студенте докторских студија, као и за прилоге аутора са ФТН или ван ФТН-а.

Зборник излази у два облика – електронском на веб сајту ФТН-а (www.ftn.uns.ac.rs) и штампаном, који је пред вама. Обе верзије публикују се сваки месец, у оквиру промоције дипломираних мастера.

У овом броју штампани су радови студената мастер студија, сада већ мастера, који су радове бранили у периоду од 01.09.2023. до 03.10.2023. год., а који се промовишу 27.01.2024. год. То су оригинални прилози студената са главним резултатима њихових мастер радова.

Известан број кандидата објавили су радове на некој од домаћих научних конференција или у неком од часописа. Њихови радови нису штампани у Зборнику радова.

Велик број дипломираних инжењера–мастера у овом периоду био је разлог што су радови поводом ове промоције подељени у три свеске.

У овој свесци са редним бројем 12. објављени су радови из области:

- електротехнике и рачунарства.

У свесци са редним бројем 11. објављени су радови из области:

- машинства,
- грађевинарства,
- саобраћаја,
- графичког инжењерства и дизајна,
- архитектуре и
- мехатронике,

У свесци са редним бројем 1. из 2024. године објављени су радови из области:

- инжењерског менаџмента,
- инжењерства заштите на раду и заштите животне средине,
- математике у техници,
- геодезије и геоматике,
- инжењерства информационих система,
- сценске архитектуре и дизајна,
- биомедицинског инжењерства и
- сценске архитектуре и технике.

Уредништво се нада да ће и професори и сарадници ФТН-а и других институција наћи интерес да публикују своје резултате истраживања у облику регуларних радова у овом часопису. Ти радови ће бити објављивани на енглеском језику због пуне међународне видљивости и проходности презентованих резултата.

У плану је да часопис, својим редовним изласком и високим квалитетом, привуче пажњу и постане довољно препознатљив и цитиран да може да стане раме-уз-раме са водећим часописима и заслужи своје место на СЦИ листи, чиме ће значајно допринети да се оствари мото Факултета техничких наука:

„Високо место у друштву најбољих“

Уредништво

SADRŽAJ

STRANA

Radovi iz oblasti: Elektrotehnika i računarstvo

1. Ален Мујо, ДЕТЕКЦИЈА ОМЕТЕНОСТИ ВОЗАЧА НА ОСНОВУ АНАЛИЗЕ СНИМКА ВОЗАЧА ПРИМЕНОМ МЕТОДА ДУБОКОГ УЧЕЊА	1655-1658
2. Nikola Jovišić, PREDLOG SISTEMA VEŠTAČKE INTELIGENCIJE ZA POMOĆ U MAMOGRAFSKOJ DIJAGNOSTICI	1659-1662
3. Gordana Markov, Luka Strezoski, MIKROMREŽA KAO RESURS ZA OPTIMIZACIJU NAPONA I SNAGA NA DISTRIBUTIVNOM IZVODU	1663-1666
4. Milan Popović, Dejan Jerkan, SAVREMENA RJEŠENJA SPOLJAŠNJEG OSVJETLJENJA PRIMJENOM LED-a	1667-1670
5. Ivana Košutić, Luka Strezoski, OPTIMIZACIJA NAPONA I SNAGA UNUTAR MIKROMREŽE	1671-1674
6. Tara Pogančev, INTEGRACIJA CHAT-GPT-A SA TEHNOLOGIJAMA SEMANTIČKOG VEBA	1675-1678
7. Eva Janković, Nikola Luburić, SISTEM ZA OPTIMIZACIJU RASPOREDA NASTAVE FAKULTETA TEHNIČKIH NAUKA	1679-1682
8. Luna Živković, Milan Vidaković, METODOLOGIJA UZORKOVANJA ISKUSTVA (ESM) KORISNIKA U MOBILNOM RAČUNARSTVU	1683-1686
9. Милица Вучинић, НАМЕНСКИ ЈЕЗИК ЗА МОДЕЛОВАЊЕ МИГРАЦИЈЕ РЕЛАЦИОНЕ БАЗЕ ПОДАТАКА У БАЗУ ПОДАТАКА ЗАСНОВАНУ НА ДОКУМЕНТИМА	1687-1690
10. Аница Ђукић, Никола Лубурић, РАЗВОЈ СОФТВЕРА ЗА ОРГАНИЗАЦИЈУ РАСПОРЕДА ЧАСОВА У ВИСОКОШКОЛСКИМ УСТАНОВАМА	1691-1694
11. Јелена Хрњак, SECURADSL – НАМЕНСКИ ЈЕЗИК ЗА ПОДРШКУ БРЗОГ УСПОСТАВЉАЊА БЕЗБЕДНОСНИХ АСПЕКТА У РАДНОМ ОКВИРУ SPRING	1695-1698

	STRANA
12. Петар Марковић, РАЗВОЈ АПЛИКАЦИЈЕ СТУДЕНТСКЕ СЛУЖБЕ ПРИМЕНОМ ДЕЦЕНТРАЛИЗОВАНИХ И ДИСТРИБУИРАНИХ ТЕХНОЛОГИЈА И ПРОТОКОЛА	1699-1702
13. Марко Његомир, Јелена Сливка, СИСТЕМ ЗА ПРЕПОРУКУ ЗАСНОВАН НА ГРАФОВСКИМ НЕУРОНСКИМ МРЕЖАМА	1703-1706
14. Miloš Popović, DEVOPS PRISTUP RAZVOJU APLIKACIJE SA INTEGRACIJOM GENERATIVNOG JEZIČKOG MODELA	1707-1710
15. Ivan Mršulja, СИСТЕМ ЗА REVERZNU PRETRAGU SLIKA	1711-1714
16. Андријана Јеремић, Јелена Сливка, ПРЕДИКЦИЈА СРЧАНОГ УДАРА НА ОСНОВУ СТАЊА ПАЦИЈЕНТА ПРИМЕНОМ АЛГОРИТАМА МАШИНСКОГ УЧЕЊА	1715-1718
17. Miloš Topolić, PROTOTIP APLIKACIJE ZA PRUŽANJE POSLOVNIH INFORMACIJA I ANALIZA MAKSIMALNE UPOTREBLJIVOSTI	1719-1722
18. Vera Kovačević, PREGLED I IMPLEMENTACIJA ODABRANIH RETROAKTIVNIH STRUKTURA PODATAKA	1723-1726
19. Ivana Miličić, KLASIFIKACIJA POKRETA ŠAKE NA OSNOVU EMG SIGNALA	1727-1730
20. Aleksandar Ignjatijević, RAZVOJ I POSTAVLJANJE BLOKČEJN-BAZIRANOG DISTRIBUIRANOG SISTEMA ZA OCENJIVANJE STUDENATA	1731-1734
21. Jovan Ivošević, POGON ZA VIZUELIZACIJU ANIMIRANIH 3D MODELA	1735-1738
22. Anja Tanović, IMPLEMENTACIJA PARALELNOG K-MEANS ALGORITMA ZA KLASIFIKACIJU CIFARA UPOTREBOM OPENMP I MPI BIBLIOTEKA	1739-1742
23. Jana Janković, IMPLEMENTACIJA PROTOKOLA ZA KOHERENTNOST KEŠ MEMORIJE	1743-1746
24. Nebojša Pilipović, RAZVOJ PYTHON VEZA, BIBLIOTEKE I APLIKACIJE ZA HARDVERSKI AKCELERATOR NEURONSKIH MREŽA	1747-1750
25. Ана Теодоровић, ИМПЛЕМЕНТАЦИЈА ХМРР BOSH ПРОТОКОЛА У ОКВИРУ TR-069 КЛИЈЕНТА	1751-1753
26. Milomir Spajić, POWER-SIGNAL INTEGRITY SIMULACIJE ELEKTRONSKIH KOLA	1754-1757
27. Nikolina Tomić, KOMPARATIVNA ANALIZA PERFORMANSI SISTEMA APACHE KAFKA I RABBITMQ	1758-1761
28. Сара Попарић, РАЗВОЈ РЕСПОНЗИВНЕ ВЕБ АПЛИКАЦИЈЕ ЗА ПОДРШКУ РАДУ НАЦИОНАЛНОГ САВЕЗА ЗА КОЊИЧКИ СПОРТ	1762-1765
29. Милица Ињац, ГЕНЕРИСАЊЕ АРХИТЕКТОНСКИХ ОСНОВА УЗ ПОМОЋ МОДИФИКОВАНОГ ГЕНЕТСКОГ АЛГОРИТМА	1766-1769
30. Стефан Арађанин, Јелена Сливка, ПРИМЕНА МОДЕЛА ВРЕМЕНСКИХ СЕРИЈА ЗА ПРЕДИКЦИЈУ СРЕДЊЕГ ГЛОБАЛНОГ НИВОА МОРА	1770-1773
31. Матија Мијалковић, ПЕРЗИСТЕНЦИЈА NEW WAVE АЛАТА ЗА УПРАВЉАЊЕ РАДНИМ ТОКОВИМА У РЕЛАЦИОНУ БАЗУ ПОДАТАКА	1774-1777

	STRANA
32. Сенка Шоић, МИГРАЦИЈА МИКРОСЕРВИСНЕ АПЛИКАЦИЈЕ СА РЕЛАЦИОНОГ МОДЕЛА ПОДАТАКА У НЕРЕЛАЦИОНИ МОДЕЛ КОРИШЋЕЊЕМ SERVERLESS AWS СЕРВИСА	1778-1781
33. Anđela Čičković, WEB АПЛИКАЦИЈА ЗА PRETRAGU WINDOWS LOGOVA	1782-1785
34. Monika Erdeg, IMPLEMENTACIЈА SISTEMA ZA PLAĆANJE NA AWS INFRASTRUKTURI	1786-1789
35. Sava Njegovan, FIZIČKI MODEL SISTEMA ZA AKTIVNO POTISKIVANJE BUKE	1790-1793
36. Стефан Савић, Јелена Сливка, ПРЕДИКЦИЈА ПОРАСТА НИВОА МОРА КОРИШЋЕЊЕМ АЛГОРИТАМА МАШИНСКОГ УЧЕЊА	1794-1797

ДЕТЕКЦИЈА ОМЕТЕНОСТИ ВОЗАЧА НА ОСНОВУ АНАЛИЗЕ СНИМКА ВОЗАЧА ПРИМЕНОМ МЕТОДА ДУБОКОГ УЧЕЊА**DISTRACTED DRIVER DETECTION BASED ON DRIVER VIDEO ANALYSIS USING DEEP LEARNING METHODS**

Ален Мујо, Факултет техничких наука, Нови Сад

Област – СОФТВЕРСКО ИНЖЕЊЕРСТВО И ИНФОРМАЦИОНЕ ТЕХНОЛОГИЈЕ

Кратак садржај – Људска грешка и расејана возња представљају главни узрок тешких саобраћајних несрећа. Један од начина за решавање овог проблема је уградња камера на инструмент табле унутар возила које би могле да алармирају возача када се упусти у било коју активност која му одвлачи пажњу од возње. У овом раду представљамо систем заснован на дубоком учењу за откривање ометаних возача користећи 2D слике контролне табле. Скуп података је сачињен од 22.388 RGB слика 26 јединствених возача који су класификовани у десет класа, од којих девет представља ометено понашање возача. Модел се састоји од два дела: VGG19 конволуциона неуронска мрежа за издвајање карактеристика из улазне слике и OpenPose модел за процену људске позе. Коначни модел је постигао тачност од 84% на тестном скупу података.

Кључне ријечи: дубоко учење, детекција ометености возача, VGG19, OpenPose, естимација позе човека.

Abstract – Human error and distracted driving is the main cause of severe car accidents. One way of solving this problem is installing dashboard cameras inside the vehicle, which would be able to alarm the driver once he engages in any activity that distracts him from driving. This paper presents a deep learning-based system for detecting distracted drivers using 2D dashboard images. The dataset comprises 22,388 RGB images of 26 unique drivers whose behavior is classified into ten classes, nine representing distracted driver behavior. The model consists of the VGG19 convolutional neural network for extracting features from the input image and the OpenPose model for human pose estimation. The final model achieved an accuracy of 84% on the test dataset

Keywords: deep learning, distracted driver detection, VGG19, OpenPose, human pose estimation

1. УВОД

Према Националној администрацији за безбедност саобраћаја на аутопутевима (NHTSA – National Traffic Safety Administration), 2015. године, 3.477 људи

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији ментор је била др Јелена Сливка, ванр. проф.

је погинуло, а 391.000 је повређено у сударима моторних возила у којима су учествовали расејани возачи [1]. Извештај Светске здравствене организације (СЗО) Global Status Report за 2017. за 2017. процењује 1.25 милиона смртних случајева годишње због саобраћајних несрећа широм света, при чему је до 50 милиона људи задобило повреде без смртог исхода као резултат саобраћајних несрећа [2]. Према NHTSA, петину саобраћајних несрећа изазове ометани возач, а отприлике 90% саобраћајних несрећа је последица људских грешака у Сједињеним Државама [3].

Центар за контролу и превенцију болести (CDC - The Centers for Disease Control and Prevention) дефинише понашање ометане возње узимајући у обзир визуелне (нпр., скретање очију са пута), мануелне (нпр., скидање руку са волана) и когнитивне (нпр., скретање ума са пута) узроке [4]. Да би боље осигурали своје клијенте, State Farm је објавио скуп података 2D слика камере на контролној табли са различитим врстама ометаног понашања. Овај рад се фокусира на креирање система који је у стању да детектује девет типова понашања ометених возача како би их алармирао и спречио било какве потенцијалне незгоде.

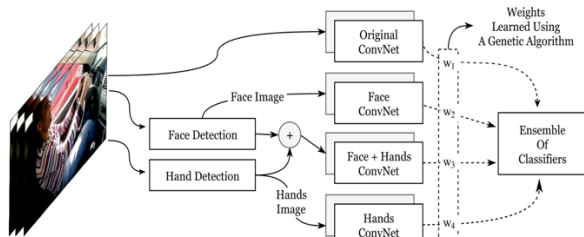
У другом поглављу су приказана два релевантна рада који се баве истим проблемом. Треће поглавље представља скуп података који се користи у овом раду док четврто поглавље приказује коришћену методологију. Методе евалуације и постигнути резултати приказани су у петом поглављу. Последње резимира рад и објашњава потенцијална побољшања.

2. ПРЕТХОДНА РЕШЕЊА

У овом поглављу ће укратко бити објашњени сродни радови који решавају проблем детекције ометаних возача користећи исти скуп података који је описан у трећем поглављу.

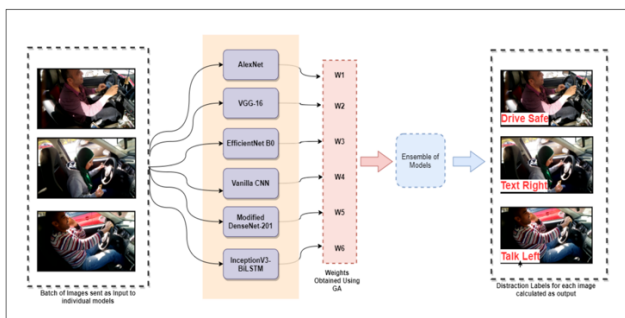
У раду [5] аутори су предложили метод који се састоји од *genetically-weighted* ансамбла конволуционих неуронских мрежа (Convolutional Neural Network, CNN). Коришћени CNN модели су AlexNet и InceptionV3. Ови модели обучени су на четири извора слика: необрађене слике, слике лица, слике руку и слике "лице+руке". Резултат обучавања представља укупно осам модела који производе вектор вероватноће од 10 бројева који одговарају вероватноћама постојећих класа. Коначно предвиђање је израчунато коришћењем отежињеног (*weighted*) система гласања,

насупротив већинског система гласања који би претпоставио да сваки класификатор подједнако доприноси коначном предвиђању. Да би проценили тежину доприноса за сваки класификатор, аутори су користили генетски алгоритам. Скуп података је подељен у размери од 75% за обуку и 25% за тестирање и решење је евалуирано коришћењем тачности као метрике. Предложени метод постигао је тачност класификације од 95.98%. Једно од запажања које су аутори извели је да руке имају већу тежину у предикцији тачне класе него лице. Предложени метод је приказан на слици 1.



Слика 1 - Метода предложена у раду [5].

У раду [6] аутори су користили сличан приступ са *weighted* ансамблима неколико CNN модела приказаних на слици 2. Аутори су користили два скупа података за своје истраживање: AUC (*American University of Cairo*) скуп података о ометаном возачу и *State Farm* скуп података.



Слика 2 - Коначна предикција се даје на основу више различитих модела.

Оба скупа садрже слике које су сврстане у 10 различитих категорија, од безбедне вожње до разних понашања ометања. Предложена методологија обухвата одвојено тренирање неколико различитих модела неуронских мрежа као што су *AlexNet*, *VGG-16*, *EfficientNet B0*, *Vanilla CNN*, модификовани *DenseNet-201* хијерархијски модел и *InceptionV3* интегрисан са слојевима двосмерне дуготрајне меморије (*BiLSTM*).

Сваки од ових модела обучен је на сликама димензије од 224 x 224. Аутори затим израчунавају отежињени збир резултата свих класификатора користећи генетски алгоритам, дајући коначни вектор вероватноће сваке класе.

Овај приступ користи предност идеје да ансамбли могу послужити као јачи класификатори или прецизнији механизам за предвиђање од појединачних слабих класификатора од којих се састоје. Резултати

на тест скупу показали су да је 6-модел *GA-weighted* ансамбл постигао укупну тачност од 96.37% на AUC скупу података и 99.57% на *State Farm* скупу података, надмашујући тачност примене појединачних модела над истим скуповима.

3. СКУП ПОДАТАКА

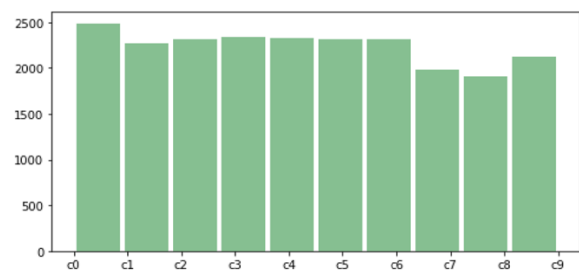
Скуп података који се користи за овај задатак је јавно доступан скуп података који је пре неколико година коришћен за *State Farm Distracted Driver Detection* такмичење на сајту *Kaggle* [7].

Слике су добијене у контролисаном окружењу где је од возача тражено да изведу неколико различитих активности. Садржи 22.388 слика у боји са димензијама 640 x 480 пиксела. Постоји укупно 10 класа у скупу података, које су приказане на слици 3: (c0) безбедна вожња; (c1/c3) слање порука десном/левом руком; (c2/c4) разговара телефоном са десна/лева рука; (c5) управљање радиом; (c6) пијење; (c7) посезање позади; (c8) фризура и шминка; (c9) причање са сапутником (Слика 3).



Слика 3 - Примери 10 класа присутних у скупу података

Свака класа садржи око 2.300 слика, што ово чини уравнотеженим скупом података (Слика 4). Скуп података такође садржи датотеку у којој је свака слика повезана са одређеним идентификационим бројем возача. Постоји 26 јединствених возача на сликама. Тренинг и тест скуп су подељени тако да сваки возач који се налази у тренинг скупу података не буде део и тест скупа података, што се разликује од других радова који су поделили своје податке насумично.



Слика 4 - Дистрибуција броја слика по класама.

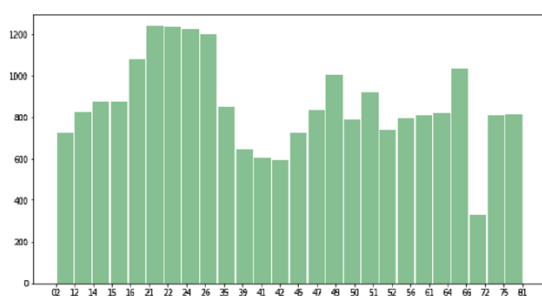
На тај начин циљ модела јесте да научи да препозна активности и да не буде под утицајем самих возача. Тренинг слике се даље деле на основу возача који је присутан на слици насумичним одабиром пет возача и коришћењем њихових слика као скуп података за

валидацију, а преостале слике као скуп података за обуку.

Свака слика је у *JPG* формату и означена је ознаком која указује на активности коју возач ради. Ознаке су нумеричке, одговарају горе наведеним активностима и дате су у засебној *CSV* датотеци.

Поред тога, у *CSV* датотеци постоји колона “*subject*”, која представља јединствени идентификациони број возача са слике.

На слици 5 приказан је број слика за сваког јединственог возача у скупу података.



Слика 5 - Број слика за сваког јединственог возача.

4. МЕТОД

Предложено решење се може поделити у два корака: екстракција особина улазне слике коришћењем *CNN* модела и детекција кључних тачака тела коришћењем модела за естимацију поза људи.

За издвајање особина са улазне слике је коришћен *VGG19* модел са 19 конволуционих слојева који је оригинално трениран на задатку класификације слика на *ImageNet* скупу података од 1000 различитих категорија. Због ограничених *GPU* ресурса, улаз у *VGG19* је слика у боји димензије 84 x 84 x 3.

Уместо коришћења потпуно повезаних слојева димензија 4096 и 1000 које оригинална имплементација модела има, у овом решењу су ти слојеви замењени једним потпуно повезаним слојем од 1024 неурона. За тај вектор од 1024 кажемо да представља издвојене особине са улазне слике.

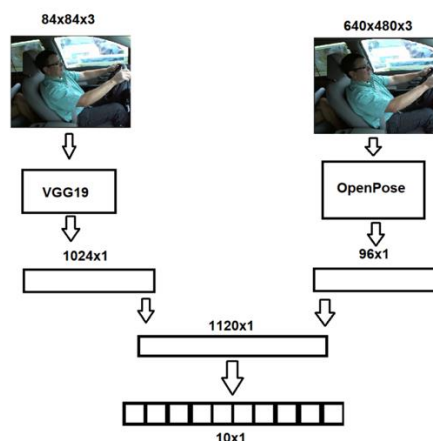
За детекцију кључних тачака тела је коришћен *OpenPose* модел, који је један од првих *real-time multi-person* модела за детекцију људи на слици. За улазну слику модел даје укупно 135 предикција кључних тачака. Улаз у модел је оригинална слика димензија 640 x 480, а излаз је вектор бројева који представља *x* и *y* координате сваке кључне тачке, као и један број који представља поузданост те предикције - *confidence score*.

Кључне тачке које нису детектоване у бар 60% улазних слика, као што су леви или десни чланак, лево ухо, лева или десна пета, су уклоњене. Свака слика је описана са 25 одабраних кључних тачака, тј., вектором од 75 бројева где је свака кључна тачка представљена *x* и *y* координатом и својим *confidence score*-ом.

Како би из овога издвојили додатне карактеристике, на овај вектор су додате још 21 додатне карактеристике које представљају углове између одређених кључних тачака. На пример: угао између кључних тачака зглоба десне руке, десног кука и десног рамена, или

угао између десног зглоба руке, носа и левог зглоба руке. Коначни излаз представља вектор од 96 бројева.

На слици 6 је приказан коначан модел. Излаз из *VGG19* модела за екстракцију особина од 1024 x 1 и излаз *OpenPose* модела за детекцију кључних тачака тела од 96 x 1 су комбиновани у један вектор димензије 1120 x 1. На овај вектор се примењује *softmax* слој који рачуна вероватноћу за сваку од 10 присутних класа. Модел је трениран *cross-entropy* функцијом грешке и *Adam* оптимизатором укупно 10 епоха и вредношћу параметра *batch-size* 40.

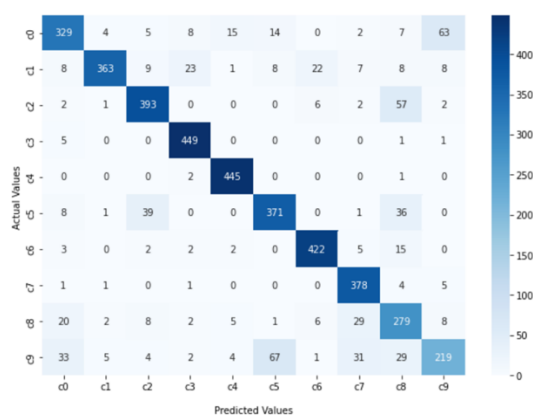


Слика 6 - Предложено решење уључује комбинацију излаза *VGG19* и *OpenPose* модела.

5. РЕЗУЛТАТИ И ДИСКУСИЈА

Скуп података је подељен на основу јединствених возача који су присутни на сликама. Тренинг скуп се састоји од 18.065 слика на којима је присутан 21 возач, од којих је пет возача коришћено као валидациони скуп приликом тренирања. Тест скуп садржи 4.323 слика на којима је присутно пет возача. Оваква подела је извршена како би се модел приликом тренирања фокусирао на саме активности које возачи раде и да не би *overfit*-овао на самим возачима и научио да ради препознавање активности на основу возача.

На слици 7 је приказана матрица конфузије за остварене предикције.



Слика 7 - Матрица конфузије за остварене предикције.

Модел представљен у овом раду је остварио тачност од 84% на тренинг скупу података. У табели 1 су приказане вредности за *precision*, *recall* и *F1-score* за

сваку од 10 класа. Најмања вредност *F1-score* (испод 0.85) је постигнута за класу c0 (безбедна возња), c5 (управљање радиом), c8 (коса и шминка) и c9 (разговор са сувозачем).

<i>class</i>	<i>precision</i>	<i>recall</i>	<i>f1-score</i>
C0	0.80	0.74	0.77
C1	0.96	0.79	0.87
C2	0.85	0.85	0.85
C3	0.92	0.98	0.95
C4	0.94	0.99	0.97
C5	0.80	0.81	0.81
C6	0.92	0.94	0.93
C7	0.83	0.97	0.89
C8	0.64	0.78	0.70
C9	0.72	0.55	0.62

Табела 1 – Прецизност, одзив и *F1-score* за сваку од класа.

Нормална возња (c0) је најчешће погрешно класификована као разговором са сувозачем (c9). Управљање радиом (c5) је најчешће погрешно класификовано као разговор телефоном десном руком (c2) и фризуром и шминком (c8). Коса и шминка (c8) је најчешће класификована као нормална возња (c0) и као посезање позади (c7). Разговор са путницима (c9) најчешће се класификовао као управљање радиом (c5) и нормална возња (c0).

6. ЗАКЉУЧАК

У овом раду је представљен модел за детекцију понашања возача који може да детектује 10 различитих активности. Примена решења може потенцијално смањити број саобраћајних несрећа које су углавном узроковане људским грешкама.

За решавање овог проблема је коришћен јавно доступан скуп података који се састоји од 22.388 слика са девет типова понашања која указују на ометеност возача. Проблем је решаван комбиновањем два модела дубоког учења, *VGG19* и *OpenPose* модела. Задатак првог је био да из улазне слике димензија 84 x 84 издвоји карактеристике које су представљене вектором димензије 1024 x 1. Задатак *OpenPose* модела је био да из улазне слике детектује кључне тачке возача присутних на сликама. Улаз у модел је слика оригиналних димензија, а излаз су координате кључних тачака делова тела. На добијених 25 кључних тачака које су описане својим координатама и *confidence score*-ом је додато још 21 карактеристика које представљају углове између ручно одабраних делова тела. Коначан модел примењује *softmax* слој за класификацију на вектор који је добијен комбиновањем излаза *VGG19* и *OpenPose* модела.

Коначни модел је остварио тачност од 84% која је мања од резултата пријављених у релевантној литератури обзиром на ограничене ресурсе приликом тренирања модела. Циљ овог рада није био представљање модела који постиже најбољу тачност,

јер и најтачнији модел који је спор неће наћи примену у решавању почетног проблема, спречавању саобраћајних несрећа упозоравањем возача. Модел је направио највише грешака класификујући безбедну возњу, руковање радиом, фризуру и шминку и разговоре са путницима.

Главни изазов су били ограничени рачунарски ресурси. Једно од усавршавања овог решења би било могуће постојањем више рачунарских ресурса, тиме би могли да повећамо величину улазних слика за *VGG19* мрежу, што би побољшало екстракцију карактеристика.

7. ЛИТЕРАТУРА

- [1] Timothy M. Pickrell, Hongying (Ruby) Li, and Shova KC. TRAFFIC SAFETY FACTS, 2016. URL <https://www.nhtsa.gov/risky-driving/distracted-driving>.
- [2] W. H. Organization. World health statistics 2017: monitoring health for the SDGs, sustainable development goals, 2017.
- [3] Strickland. How autonomous vehicles will shape the future of surface transportation. Testimony of The Honorable David L. Strickland Administrator NHTSA, 2013
- [4] US Department of Health & Human Services. Distracted Driving, 2016. URL <https://www.cdc.gov/transportationsafety/distracted-driving/index.html>
- [5] Abouelnaga, Y., Eraqi, H.M. and Moustafa, M.N., 2017. Real-time distracted driver posture classification. arXiv preprint arXiv:1706.09498
- [6] Kumar, A., Sangwan, K.S. and Dhiraj, 2021. A computer vision based approach for driver distraction recognition using deep learning and genetic algorithm based ensemble. ICAISC 2021, Proceedings, Part II 20 (pp. 44-56).
- [7] "State farm distracted driver detection" www.kaggle.com/c/state-farm-distracted-driver-detection

Кратка биографија:



Ален Мујо рођен је 1999. године. Основне академске студије завршио је 2021. године на Факултету техничких наука, на ком брани и мастер рад 2023. године из области Електротехнике и рачунарства – Софтверско инжењерство и информационе технологије.
контакт: alen.mujo@gmail.com

**PREDLOG SISTEMA VEŠTAČKE INTELIGENCIJE ZA POMOĆ U MAMOGRAFSKOJ
DIJAGNOSTICI****A PROPOSAL OF AN ARTIFICIAL INTELLIGENCE SYSTEM FOR ASSISTANCE IN
MAMMOGRAPHY DIAGNOSTICS**

Nikola Jovišić, *Fakultet tehničkih nauka, Novi Sad*

**Oblast – ELEKTROTEHNIČKO I RAČUNARSKO
INŽENJERSTVO**

Kratak sadržaj – Mamografija kao dijagnostička metoda za otkrivanje maligniteta široko je u upotrebi i oslanja se na ekspertsko tumačenje radiologa. U ovom radu istražena je prilika da se takav sistem unapredi veštačkom inteligencijom koja bi predstavljala podršku u donošenju odluka u vezi sa tumačenjem mamografskih snimaka. Istražen je model U-net za segmentaciju tkiva, opisana njegova generalna arhitektura i arhitektura prilagođena za opisani problem, kao i njegova primena na nekim od otvorenih skupova podataka. Opisan je proces i opisani su parametri treniranja modela. Izneseni su i diskutovani rezultati primenjenog modela i sve to sumirano u zaključnom poglavlju.

Ključne reči: mamografija; neuronske mreže; U-net; segmentacija

Abstract – Mammography as a diagnostic method for detecting malignancy is widely used and relies on the expert interpretation of radiologists. In this paper, the opportunity to improve such a system with artificial intelligence, which would represent support in decision-making related to the interpretation of mammographic images, was explored. The U-net model for tissue segmentation is investigated, its general architecture and architecture adapted for the described problem are described, as well as its application to some of the open datasets. The process and the parameters of the training are described. The results of the applied model are presented and discussed and all this is summarized in the concluding chapter.

Keywords: mammography; neural networks; U-net; segmentation

1. UVOD

Kada je reč o bolestima dojke najčešći i vrlo efikasan metod dijagnostike jeste mamografija, tj. snimanje dojke rendgenskim zracima. Korisna je jer je neinvazivna (ne narušava se integritet tkiva tokom samog dijagnostičkog procesa), nije bolna i, uprkos nekim verovanjima, nije štetna.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Srđan Vukmirović, red. prof.

Ova metoda daje uvid u unutrašnjost dojke i pomaže pri detekciji sumnjivih masa kao što su benigni i maligni tumori i kalcifikacije. Sa ovakvog snimka donosi se bitna odluka o tome da li:

- pacijenta okarakterisati kao zdravog
- pacijentu direktno dijagnostikovati problem
- pacijenta poslati na dalja ispitivanja (magnetna rezonanca (MRI), biopsija i sl.)

Ovakvi pregledi sprovode se ne samo u slučajevima kada lekar posumnja na problem, već i preventivno, pa tako snimici zdravih pacijenata čine dominantan udeo ukupnih mamografskih snimaka.

Problem nastaje jer postoji manjak visokokvalifikovanog medicinskog kadra u Srbiji (mada je problem zastupljen i u ostatku Evrope, pa i sveta) sposobnog za rešavanje ovog tipa zadataka. Velika potreba za istim ima ozbiljne reperkusije na cenu, kvalitet, dostupnost i pogodnost zdravstvenih usluga, pa čak i na morbiditet i mortalitet, a sve jer ne postoji način da se rutinski donose odluke u koje se može verovati.

Da bi se izbegao najgori scenario velikog broj lažnih negativnih izveštaja (oni koji pacijente sa oboljenjem karakterišu kao zdrave) ovakvo stanje rezultuje problemom u praksi da se opcija slanja pacijenata na dodatna ispitivanja dešava relativno često. Pacijenti se šalju u zdravstvene ustanove drugog i trećeg stepena zdravstvene zaštite, odnosno ustanove koje su namenjene za zdravstvenu zaštitu de fakto obolelih pacijenata, iako je ovde zapravo reč o pacijentima kod kojih ništa još nije dijagnostikovano. Ovo značajno preopterećuje te ustanove pacijentima sa lažnim pozitivnim izveštajima (oni koji pacijente bez oboljenja karakterišu kao sumnjive), liste čekanja u njima vrlo su velike i pacijenti često ostaju primorani da se odluče za druge vidove zdravstvene zaštite (lečenje u privatnim ustanovama, lečenje u inostranstvu i sl.) ili, u još težem slučaju, ostaju uskraćeni za bilo kakav vid zdravstvene zaštite. Zdravi pacijenti po pravilu bivaju izloženi velikom stresu dok ne dođu na red za dalja ispitivanja, da bi se stresa oslobodili tek nakon 6-12 meseci kada je on već ostavio posledice, a obolelim pacijentima, koji zbog zdravih moraju čekati isto toliko, problem može dodatno da se zakomplikuje.

Moderni razvoj veštačke inteligencije pruža nove mogućnosti za usavršavanje dijagnostike, naročito u domenu istraživanja segmentacionih mreža, na čemu je zasnovano ovde predloženo rešenje. Jedan sistem veštačke inteligencije zasnovan na obradi slike mogao bi

se razviti u svrhu automatizacije ili poluautomatizacije procesa odlučivanja u dijagnostici oboljenja dojke.

2. UPOTREBA I OBUČAVANJE U-NET MODELA U SEGMENTACIJI SUMNJIVIH MASA NA DOJCI

U-Net je arhitektura neuronske mreže koja se koristi za semantičku segmentaciju slika, odnosno za razdvajanje i klasifikaciju pojedinačnih piksela na slici. Posebno je dizajnirana za obradu medicinskih slika, kao što su snimci magnetne rezonance (MRI) ili snimci rendgenskih zraka. Glavni cilj U-Net modela je precizno razdvajanje objekata i regiona od interesa na slici kako bi se olakšala analiza i interpretacija medicinskih podataka. Ova arhitektura je nazvana U-Net zbog svog oblika koji podseća na latinično slovo "U". Sastoji se od dve glavne komponente: koder (*coder*) i dekoder (*decoder*). Koder se sastoji od nekoliko slojeva konvolucije koji postepeno smanjuju prostornu dimenziju slike i izvlače bitne karakteristike. Dekoder, s druge strane, koristi slojeve koji povećavaju prostornu dimenziju i pomažu u rekonstrukciji segmentirane slike.

Glavna inovacija koju U-Net donosi je upotreba *skip konekcija* (*skip connections*) koje omogućavaju prenos informacija iz koda direktno u dekoder. Ove veze pomažu u obnavljanju detalja i održavaju konteksta informacija koji bi se mogao izgubiti tokom postupka konvolucije. Osim toga, skip konekcije takođe pomažu u prevazilaženju problema gubitka rezolucije koji se često javlja u dubokim neuronskim mrežama. Treba napomenuti da uobičajena U-Net arhitektura koristi funkciju gubitka poznatu kao *softmax* (*softmax*) koja omogućava klasifikaciju svakog piksela na slici. Ova funkcija gubitka pomaže u optimizaciji mreže kako bi se postigla što preciznija segmentacija. U-Net se pokazao veoma uspešnim u mnogim medicinskim aplikacijama, uključujući segmentaciju tumora, identifikaciju organa i analizu histoloških snimaka. Njegova sposobnost za preciznu segmentaciju i očuvanje konteksta informacija čini ga vrlo korisnim alatom za medicinske stručnjake u procesu dijagnostike i tretmana.

Kako različiti vidovi segmentacije (semantička, panoptička i sl.) imaju slične zahteve koji se tiču same arhitekture (eventualno je potrebno promeniti broj kanala izlaza) slična arhitektura može da se koristi za različite zadatke.

Konkretno, odlučeno je da se za segmentaciju sumnjivih masa na dojci koristi VGG16 arhitektura. Za enkoder deo preuzet je VGG16 iz Keras biblioteke sa *ImageNet* težinama.

Za prilagođavanje ovom modelu je bilo potrebno postojeće *grayscale* slike pretvoriti u trokanalne, a što je ostvareno konkateniranjem istog kanala kroz sva tri. Dimenzije ulazne (a, po prirodi arhitekture U-Net, dakle, i izlazne) slike su 224x224. Kako su dimenzije slika u skupu podataka raznovrsne i značajno veće od toga (neke prelaze i dimenzionalnost od 5000) sve su umanjene na ovu veličinu.

Dekoder je ručno kreiran, sa skip-konekcijama. Podeljen je u pet blokova koji koriste 2D konvolucione slojeve sa veličinom kernela 3x3.

Odgovarajućim preprocesiranjem svi snimci i sve korespondentne maske dovedene su u stanje pogodno za konzumiranje od strane modela.

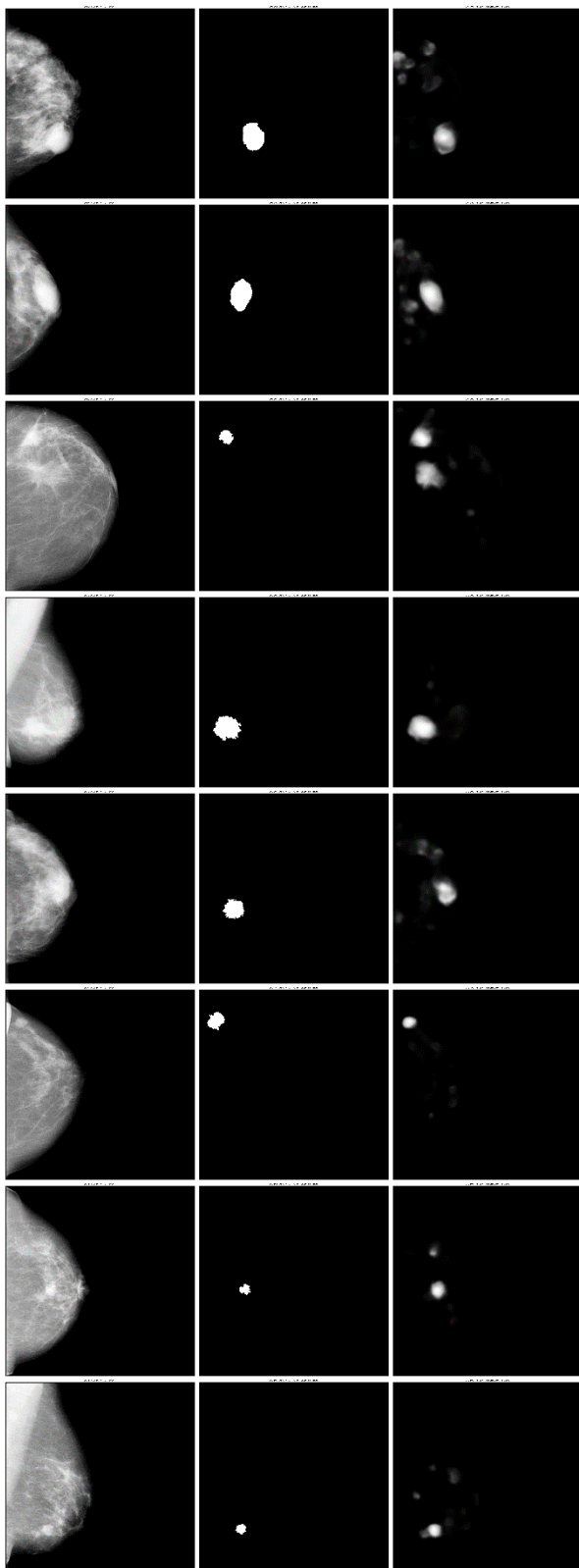
- podaci su podeljeni u 1124 para slika-masku za trening i 343 para slika-masku za testiranje
- primenjene su tehnike uvećavanja-augmentacije skupa (*data augmentation*) - vertikalno i horizontalno obrtanje (*flip*) i promena osvetljenosti (*brightness*), dok su neke druge augmentacije kao što su izduživanje slike izostavljene jer nisu verna simulacija promene u odnosu na originalni skup podataka na koju bi model mogao da naiđe u realnoj upotrebi
- kao metrika za evaluaciju modela primarno je korišćen količnik preseka i unije predviđene i istinite maske (*IoU – intersection over union*) koja je uobičajena metrika u segmentacionim zadacima
- trening je ograničen na 50 epoha
- koristeći Keras *early stop* mehanizam, uz parametar strpljivosti 5 trening je zaustavljen i pre dostizanja ograničenja od 50 epoha, tj. kada 5 epoha za redom u odabranoj metrici više nije bilo napretka
- funkcija cilja (*loss function*) koja je optimizovana u procesu treniranja je binarna unakrsna entropija (*binary cross-entropy*) koja je uobičajena kod binarnih klasifikacionih problema, a što upravo semantička segmentacija na dve klase („jeste“ i „nije“ sumnjiva masa) i jeste – klasifikacija piksela na dve klase
- trening je sproveden na *Huawei Ascend NPU (neural processing unit)* jedinicama
- veličina grupe (*batch size*) za trening je 10
- parametar verovatnoće za ispuštanje (*dropout*) je postavljen na 0.5
- optimizator za optimizaciju parametara U-net mreže koji je iskorišten je ADAM
- tempo obučavanja (*learning rate*) je postavljen na 0.0001
- u treningu, po završetku svake epohe, sačuvan je onaj model koji se pokazao bolje od prethodnog najboljeg po odabranoj metrici i kao konačni proglašen je poslednji – najbolji model

3. REZULTATI I DISKUSIJA

Na slikama prikazani su primeri izlaza iz U-net modela. Levo se nalazi preprocesiran snimak dojke koji je upotrebljen kao ulaz u model, u sredini anotaciona maska za dati primer iz test skupa i desno izlaz iz U-net modela u vidu meke maske (*soft mask*).

Primećuje se da model prilično dobro pronalazi sumnjive mase i da je učestalost lažnih negativnih regiona veoma mala što je veoma bitno. Model dodatno pronalazi neke mase koje u anotaciji nisu naznačene. Manuelnom inspekcijom od strane radiologa, zaključeno je da su tu u pitanju neke benigne mase, kalcifikati, fibrozno tkivo i sl. što u anotaciji nije naznačeno.

Ova pojava lažnih pozitivnih u ovoj fazi nije mnogo problematična jer, u upotrebi modela kao asistencije pri anotaciji vrlo je jednostavno odbaciti objekte (maske, poligone) za koje lekar smatra da nisu potrebni. U produkcionoj fazi, sa druge strane, svakako bi bio razvijan dodatni model za klasifikaciju svake od pronađenih instanci i instance koje nisu od interesa bivale bi odbacivane tada.



Slika 1. Primeri izlaza U-net modela

Tačnost ovog modela iznosi 99%, ali značajnije od ove metrike, kako je na izlazu uvek velik broj piksela pozadine i naivnom predikcijom čitave slike kao pozadine dobijala bi se visoka tačnost, metrika količnika preseka i unije (IoU – *intersection over union*) iznosi 0.25 na test skupu.

Iako nije u pitanju velika vrednost, ova vrednost metrike je zadovoljavajuća uzimajući u obzir da su u pitanju meke

maske i da model pronalazi dodatne mase koje, kao što je rečeno, nisu problematične.

4. LITERATURA

[1] Ciritsis A, Rossi C, Vittoria De Martini I, Eberhard M, Marcon M, Becker AS, Berger N, Boss A. Determination of mammographic breast density using a deep convolutional neural network. *Br J Radiol.* 2019 Jan;92(1093):20180691. doi: 10.1259/bjr.20180691. Epub 2018 Oct 1. PMID: 30209957; PMCID: PMC6435091.

[2] <https://www.kaggle.com/competitions/rsna-breast-cancer-detection/>

[3] R.N.J. Graham, R.W. Perriss, A.F. Scarsbrook, DICOM demystified: A review of digital file formats and their use in radiological practice, *Clinical Radiology*, Volume 60, Issue 11, 2005, Pages 1133-1140, ISSN 0009-9260

[4] <https://www.kaggle.com/code/theoviel/breast-density-classification-using-monai>

[5] Vikash Gupta, Mutlu Demirel, Robert W. Maxwell, Richard D. White, Barbaros Selnur Erdal: "A multi-reconstruction study of breast density estimation using Deep Learning", 2022; [<http://arxiv.org/abs/2202.08238> arXiv:2202.08238]

[6] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich: "Going Deeper with Convolutions", 2014; [<http://arxiv.org/abs/1409.4842> arXiv:1409.4842]

[7] Pablo Fonseca, Julio Mendoza, Jacques Wainer, Jose Ferrer, Joseph Pinto, Jorge Guerrero, Benjamin Castaneda, "Automatic breast density classification using a convolutional neural network architecture search procedure," *Proc. SPIE 9414, Medical Imaging 2015: Computer-Aided Diagnosis*, 941428 (20 March 2015); doi: 10.1117/12.2081576

[8] Ciritsis A, Rossi C, Vittoria De Martini I, Eberhard M, Marcon M, Becker AS, et al. Determination of mammographic breast density using a deep convolutional neural network. *Br J Radiol* 2019; 92: 20180691.

[9] Shen, L., Margolies, L.R., Rothstein, J.H. et al. Deep Learning to Improve Breast Cancer Detection on Screening Mammography. *Sci Rep* 9, 12495 (2019). <https://doi.org/10.1038/s41598-019-48995-4>

[10] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, Ross Girshick: "Segment Anything", 2023; [<http://arxiv.org/abs/2304.02643> arXiv:2304.02643]

[11] Jun Ma, Bo Wang: "Segment Anything in Medical Images", 2023; [<http://arxiv.org/abs/2304.12306> arXiv:2304.12306]

[12] Olaf Ronneberger, Philipp Fischer, Thomas Brox: "U-Net: Convolutional Networks for Biomedical Image Segmentation", 2015; [<http://arxiv.org/abs/1505.04597> arXiv:1505.04597]

[13] <https://towardsdatascience.com/active-learning-overview-strategies-and-uncertainty-measures-521565e0b0b>

Kratka biografija:



Nikola Jovišić rođen je 14.03.2000. u Beogradu. Završio je Srednju medicinsku školu „Dr Ružica Rip“ u Somboru i osnovne akademske studije na Fakultetu Tehničkih nauka u Novom Sadu, smer Računarstvo i automatika. Ispunio je sve obaveze i položio je sve ispite predviđene studijskim programom.

**MIKROMREŽA KAO RESURS ZA OPTIMIZACIJU NAPONA I SNAGA NA
DISTRIBUTIVNOM IZVODU****MICROGRID AS A RESOURCE FOR VOLTAGE AND POWER OPTIMIZATION AT
THE DISTRIBUTION FEEDER**

Gordana Markov, Luka Strezoski, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – Cilj ovog istraživanja jeste da se na jednostavnom primjeru distributivne mreže od 33 čvora u koju je integrisana mikromreža, pokaže optimizacija napona i snaga na izvodu, koristeći primarno mikromrežu kao resurs za optimizaciju, a zatim i ostale resurse koji se nalaze na raspolaganju u mreži.

Ključne reči: Optimizacija, napon, snaga, mikromreža

Abstract – The goal of this research is to demonstrate voltage and power optimization on a feeder of a 33-node distribution network integrated with a microgrid. This optimization will utilize the microgrid as the primary resource and subsequently other resources available within the network.

Keywords: Optimization, voltage, power, microgrid

1. UVOD

Danas se ključni trendovi u energetici mogu pratiti kroz 4D matricu: decentralizacija, dekarbonizacija, demokratizacija i digitalizacija.

Decentralizacija označava da se iz pasivnih distributivnih mreža prelazi u aktivne distributivne mreže. To jeste proizvodnja koja je bila koncentrisana u okviru većih, tradicionalnih, generatorskih jedinica (npr. termoelektrane, hidroelektrane), sada se raspoređuje i dijeli između većih generatorskih jedinica i distribuiranih generatora (DG-ova), čija instalisana snaga može da varira u jako velikom opsegu [1]. Svi izvori obnovljive energije koji su snage do 10 MW se priključuju u distributivnu mrežu, dok se oni snage veće od 10 MW priključuju na prenosnu mrežu [2].

Dekarbonizacija se primarno odnosi na smanjenje zagađenjem CO₂, međutim ona obuhvata mnogo veći spektar zagađenja. Danas se teži dobijanju električne energije, sa što većim udjelom iz čistih obnovljivih izvora. Plan je da se postepeno njihov udio u proizvodnji električne energije povećava, sve dok se termoelektrane u potpunosti ne budu mogle ugaziti, što predstavlja jedan izuzetno zahtjevan i dugotrajan proces obzirom na dva faktora, a to su da su termoelektrane najveći proizvođači električne energije i da integracija velikog broja obnovljivih izvora u distributivnu mrežu predstavlja veoma komplikovan i tehnički zahtjevan poduhvat.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Luka Strezoski, vanr. prof.

U procesu dekarbonizacije energetskog sektora, električna energija koju proizvode građani pojavljuje se u obliku dva osnovna modela: model kupca-proizvođača (eng. prosumer) i model energetskih zajednica (zajednice obnovljivih izvora energije) [3].

Demokratizacija energetskog sektora je proces koji podrazumijeva aktivno uključivanje građana u proces donošenja i usvajanja klimatsko-energetske politike i aktivno učešće u svim segmentima implementacije.

Digitalizacija mreže se odnosi na digitalizaciju što većeg broja uređaja i procesa, a sve to u cilju što efikasnijeg i efektivnijeg funkcionisanja.

Ova četiri procesa kada teknu istovremeno čine jedan veći proces koji se naziva energetska tranzicija, a podrazumijeva prelazak sa tradicionalnih načina proizvodnje, koji većinom obuhvataju neobnovljive izvore energije (ugalj, nafta, plin), na nove, obnovljive izvore energije. Energetska tranzicija sama po sebi zahtjeva i vuče mnoge nove koncepte, a jedan od njih jeste i koncept mikromreže. Prednosti koje donosi implementacija mikromreže su mnogobrojne i zavise od elementa mikromreže, njene snage i pozicije u sistemu. Svakako jedna od prednosti jeste i mogućnost optimizacije napona i snaga na izvodima. U ovom radu biće stavljen naglasak na korištenje mikromreže kao resursa za optimizaciju napona i snaga na distributivnom izvodu. Mogućnost mikromreže da obavlja ovu funkciju proizilazi iz činjenice da mikromreža omogućava više autonomnosti u upravljanju distributivnom mrežom, što je posebno korisno u uslovima povećane distribuirane proizvodnje električne energije. Rezultati istraživanja prikazani su u trećoj glavi tabelarno.

2. OPTIMIZACIJA U DISTRIBUTIVNOJ MREŽI

Optimizacija generalno u elektroenergetskim mrežama je izuzetno važna, jer pomaže u efikasnom upravljanju i korišćenju resursa, obezbjeđujući pouzdanu i ekonomičnu proizvodnju i distribuciju električne energije. U distributivnim mrežama optimizacija se obavlja uzimajući u obzir različite kriterijume i ograničenja.

Optimizacija napona je jedna od glavnih funkcija koju obavlja svaka distributivna mreža i praktično predstavlja regulaciju napona u različitim tačkama mreže, kako bi se održavao željeni naponski nivo u sistemu [4]. Tehničke granice u kojima napon mora da se zadrži, obično iznose $\pm 5\%$ ili $\pm 10\%$ u odnosu na nominalnu vrijednost napona.

Optimizacija snaga, kao funkcija, ima za cilj da maksimalno iskoristi resurse koji se nalaze u distributivnoj mreži, istovremeno obezbjeđujući pouzdanost napajanja i maksimalnu moguću efikasnost. Ovo uključuje pravilno upravljanje proizvodnjom, raspodjelom električne energije duž distributivne mreže, kao i potrošnjom.

Kriterijum gubitaka aktivne snage veoma je značajan zbog činjenice da su gubici, naponi i snage veoma usko povezani. U praksi, postoji potreba za balansiranim pristupom između optimizacije napona i smanjenja gubitaka kako bi se postiglo najefikasnije i najodrživije upravljanje distributivnom mrežom. Ovo obično uključuje upotrebu naprednih softverskih alata

3. REZULTATI ISTRAŽIVANJA

U ovoj glavi će se na nekoliko različitih primjera vršiti optimizacija napona i snaga, uvažavajući pri tome kriterijum minimalnih gubitaka aktivne snage, kao i naponska i strujna ograničenja. Kao glavni resurs za optimizaciju koristila se mikromreža, a zatim transformator T1, Kondenzator 1 i Kondenzator 2. U ovom radu za modelovanje i proračun korišten je program Etap, besplatna verzija 19.0.1, skinuta 08.07.2023. sa sajta koji je u literaturi označen pod rednim brojem [7].

3.1. Višekriterijumska optimizacija

Višekriterijumska optimizacija predstavlja traženje najboljeg rješenja iz niza dopustivih rješenja, koja odgovaraju nizu usvojenih kriterijuma/ograničenja [5].

Ograničenja koja su uvedena:

- naponsko ograničenje za čvorove i
- strujno ograničenje za vodove.

Kriterijum minimalnih gubitaka aktivne snage može se i matematički zapisati u obliku sljedeće formule:

$$\min\{z_{obj}\} = \sum^m R_l * I_{l=1}^2 \quad l = 1, 2, 3 \dots m \quad (3.1.1.)$$

[z_{obj}] -Funkcija koja se minimizira, ali uvažavajući i ograničenja

[R] -Otpornost voda

[I] -Struja po vodu

[l] -Indeks tekućeg voda

[m] -Ukupan broj vodova

Održavanje napona u zadatim tehničkim okvirima od izuzetne je važnosti. Tehničke granice u kojima napon može da se nađe prikazane su sljedećim formulama:

$$u_{min} \leq u_k \leq u_{max}, \quad k = 1, 2, 3 \dots n \quad (3.1.2.)$$

$$u_{min} = 0.95 * u_n \quad (3.1.3.)$$

$$u_{max} = 1.05 * u_n \quad (3.1.4.)$$

[u_n] -Nominalni (nazivni) napon čvora

[u_{max}] -Maksimalna vrijednost napona čvora

[u_{min}] -Minimalna vrijednost napona čvora

[k] -Tekući čvor

[n] -Broj čvorova u sistemu

Održavanje vrijednosti struja po vodovima unutar tehničkih granica je takođe od velikog značaja. Tehničke gra-

nice u kojima vrijednosti struje po vodovima mogu da se nađu izražene su sljedećim formulama:

$$i_{min} \leq i_l \leq i_{max}, \quad l = 1, 2, 3 \dots m \quad (3.1.5.)$$

$$i_{max} = 1.3 * i_n \quad (3.1.6.)$$

$$i_{min} = 0.4 * i_n \quad (3.1.7.)$$

[i] - Trenutna vrijednost struje po vodu

[i_n] - Nominalna (nazivna) vrijednost struje po vodu

[i_{max}] - Maksimalna vrijednost struje po vodu

[i_{min}] - Minimalna vrijednost struje po vodu

[l] - Tekući vod

[m] - Ukupan broj vodova

3.2. Primjer 1 – IEEE 33 test mreža

IEEE 33 test mreža se često koristi za testiranje performansi i stabilnosti elektroenergetskih sistema, proučavanje uticaja promjene opterećenja i analizu gubitaka energije. Napon mreže je 12.66 kV, dok je konfiguracija mreža radialna.

IEEE 33 test mreža sastoji se od :

- **Čvorovi:** Mreža ima ukupno 33 čvora. Svi čvorovi su potrošački (PQ čvorovi), osim čvora br.0, koji predstavlja napojni čvor tj. (ΘV čvor) i ima fiksni modu napona i fazni ugao.

- **Grane:** IEEE 33 test mreža ima 32 grane koje povezuju čvorove. Za svaku granu moguće je zadati impedansu.

- **Opterećenje:** U mreži postoje čvorovi koji predstavljaju potrošače električne energije. Opterećenje može biti rezistivno, induktivno ili kapacitivno i simulira stvarne potrošače u elektroenergetskim sistemima.

- **Generatori:** Takođe, postoje čvorovi koji predstavljaju električne generatore, koji proizvode električnu energiju i predaju je u sistem.

Nakon pokrenute simulacije tokova snaga, rezultati koji su dobijeni u proračunu, poređeni su sa radom koji se nalazi u sklopu IEEE literature, a u sklopu literature ovog rada nalazi se pod brojem [6]. Svi rezultati prikazani su u tabeli 3.2.1.

Tabela 3.2.1. Poređenje rezultata (gubitaka)

	Gubici aktivne snage[kW]
Rezultat ovog rada IEEE33	211.00
Rezultat iz literature IEEE 33	211.22

Nakon poređenja totalnih gubitaka aktivne snage, iz tabele 3.2.1. može se zaključiti da su rezultati približno jednaki, a samim tim i napravljeni model dovoljno dobar.

3.3. Primjer 2 – IEEE 33* test mreža

Prije nego što se izvršila neophodna integracija jedne mikromreže u originalni model IEEE 33 test mreže, obezbjeđeno je da se ova distributivna mreža napaja iz

matične mreže preko transformatora T2 snage 20 MVA i prenosnog odnosa 69/12.66 kV. Nakon toga izvršena je integracija mikromreže, koju je primarno činila jedna solarna elektrana, čija je ukupna instalirana snaga iznosila 1.5 MW. Da bi se osigurala stabilnost, mikromreži je dodat i jedan dizel generator snage 1.2 MW, kao rezervno napajanje. Takođe, u ovoj mikromreži se nalazi i potrošač snage 60 kVA. Ova mikromreža pripada naponskom nivou 0.415 kV, a na matičnu mrežu priključena je preko transformatora T1 čija snaga iznosi 1.5 MVA, a prenosni odnos 12.66/0.415 kV.

Mreža takođe posjeduje mogućnost ostrvskog režima rada. U distributivnoj mreži dodata su i dva kondenzatora, u čvoru 27 Kondenzator 1 snage 300 kVar i u čvoru 29 Kondenzator 2 snage 900 kVar.

Podešenja svih drugih elemenata sistema ostala su identična kao u originalnoj mreži. Nakon pokretanja simulacije došlo se do zaključka da u novoj mreži IEEE 33*, nije bilo narušavanja ni jednog ograničenja. Gubici aktivne snage dati su u tabeli 3.3.1, kao i gubici u originalnoj mreži, radi poređenja.

Tabela 3.3.1. Poređenje gubitaka

	Gubici aktivne snage [kW]
IEEE 33	211.0
IEEE 33*	83.8

Iz ovoga je očigledno da su se u novoj konfiguraciji gubici aktivne snage smanjili za približno 60.2%.

Pošto se u ovom slučaju optimizacija izvršila dodavanjem elemenata u mrežu, sada bi bilo neophodno izvršiti i optimizaciju u ovoj konfiguraciji mreže, prepodešavanjem elemenata. Dakle, cilj narednih simulacija jeste da se smanje gubici u odnosu na gubitke koji su dobijeni u prethodnom primjeru. Neophodno je ispitati da li svi elementi imaju najoptimalnije podešenje, i ukoliko to nije slučaj, pomjerjenje ka boljem rješenju neophodno je raditi u početnim koracima od po 10 %

3.4. Primjer 3 – IEEE 33* test mreža sa optimizovanom proizvodnjom mikromreže

Proizvodnja za koju se dobijaju minimalni gubici aktivne snage iznosi 1297 kW. Povezivanjem mikromreže, čija je proizvodnja veća od njene potrošnje, na distributivnu mrežu, dobija se mogućnost da ona napaja potrošače koji se nalaze u njenoj neposrednoj blizini. Što je veća njena proizvodnja, manji su tokovi snaga koji dolaze od korijena matične distributivne mreže ka ovim potrošačima, pa su samim tim manji i gubici.

Ipak snaga koju injektira mikromreža ne može biti beskonačno velika, jer u jednom trenutku proizvodnja, postaje dosta veća od obližnje potrošnje, dobijaju se sve veći uzvodni tokovi snaga, a samim tim i gubici aktivne snage u mreži. Rezultati koji su dobijeni ovdje poklapaju se sa rezultatima iz rada koji je naveden u literaturi pod rednim brojem [2]. U tabeli 3.4.1. prikazan je odnos gubitaka u prethodnoj i sadašnjoj iteraciji može se reći da su gubici aktivne snage smanjeni za 0.36%.

Tabela 3.4.1. Poređenje gubitaka

	Gubici aktivne snage[kW]
IEEE 33*	83.8
IEEE 33* nakon prve optimizacije	83.5

3.5. Primjer 3 – IEEE 33* test mreža sa optimizovanom proizvodnjom transformatora T1

Prvo se podešavao otcjep na regulacionom transformatoru T1. Vrijednosti koje otcjep može da uzme su -5.0 %, -2.5 %, 0, 2.5 % i 5.0 %. Obzirom da su gubici najmanji kada se otcjep na primaru nalazi u nultom položaju, a otcjep na sekundaru na poziciji -5 %, ovo podešenje transformatora će se zadržati. Da će se gubici smanjivati povećavanjem otcjepa na sekundaru transformatora u negativnu stranu bilo je za očekivati, zato što se ovom radnjom smanjuju struje na sekundaru, a samim tim i gubici. U tabeli 3.5.1. prikazan je odnos gubitaka u prethodnoj i sadašnjoj iteraciji, i može se reći da su smanjeni za 1.08 %.

Tabela 3.5.1. Poređenje gubitaka

	Gubici aktivne snage [kW]
IEEE 33* nakon prve optimizacije	83.5
IEEE 33* nakon druge optimizacije	82.6

3.6. Primjer 4 – IEEE 33* test mreža sa optimizovanom proizvodnjom Kondenzatora 1

Kao rezultat proračuna dobilo se da kako raste kapacitet tako gubici aktivne snage padaju. Ovo se može objasniti činjenicom da je u sistemu neophodno da postoji reaktivna snaga, ali da njen prenos utiče na povećanje pada napona [2]. Sa većim padom napona, veći je i ukupan otpor koji struja mora prevazići kroz vodove, čime se povećavaju i gubici. Zato se teži postavljanju dovoljnog broja kompenzatora reaktivne snage na strateški odgovarajućim mjestima, kako bi se izbjegao prenos, a samim tim i gubici. Iako se ovdje nije konkretno mjenjalo mjesto kondenzatora u sistemu, njegov veći kapacitet uticao je na to da se gubici smanje, ali samo do određenog trenutka, kada njegova proizvodnja kao u slučaju mikromreže postaje opterećenje za vodove i sistem i gubici opet kreću da rastu.

Za kapacitet 630 kVar dobijaju se najmanji gubici. U tabeli ispod prikazan je odnos gubitaka u prethodnoj i sadašnjoj iteraciji, i može se reći da su smanjeni za 4.0 %.

Tabela 3.6.1. Poređenje gubitaka

	Gubici aktivne snage[kW]
IEEE 33* nakon druge optimizacije	82.6
IEEE 33* nakon treće optimizacije	79.3

3.7. Primjer 5 – IEEE 33* test mreža sa optimizovanom proizvodnjom Kondenzatora 2

Ovdje je kao rezultat dobijeno da u kom se god smjeru krene mjenjati kapacitet kondenzatora, gubici se ne smanjuju, tako da je zaključak da je trenutno podešenje elementa najbolje i zbog toga se neće mjenjati.

Tabela 3.7.1. Poređenje gubitaka

	Gubici aktivne snage [kW]
IEEE 33* nakon treće optimizacije	79.3
IEEE 33* nakon četvrte optimizacije	79.3

3.8. Analiza rezultata

Ukupno smanjenje gubitaka dobijeno optimizacijom iznosi 5.36 %, a ako računamo i smanjenje gubitaka koje je izazvano integracijom mikromreže i kondenzatora u sistem ukupno smanjenje gubitaka aktivne snage iznosi 62.42 %. Smanjenje gubitaka po iteracijama u odnosu na prethodnu iteraciju prikazano je u tabeli 3.8.1.

Tabela 3.8.1. Smanjenje gubitaka po iteracijama

	Smanjenje gubitaka aktivne snage [%]
IEEE 33*	60.20 %
IEEE 33* nakon prve optimizacije	0.36 %
IEEE 33* nakon druge optimizacije	1.08 %
IEEE 33* nakon treće optimizacije	4.00 %
IEEE 33* nakon četvrte optimizacije	0.00 %

Iz prethodne tabele 3.8.1. jasno se vidi da se u ovom slučaju najveće smanjenje gubitaka dobilo prostom integracijom mikromreže i kondenzatora, zatim slijedi optimizacija snage kondenzatora, zatim optimizacija podešavanja regulacionog otcjepa na transformatoru i na kraju tek optimizacijom proizvodnje mikromreže.

4. ZAKLJUČAK

Distributivne mreže su podložne stalnim promjenama, koje su izazvane energetsom tranzicijom. Ove promjene utiču na stavljanje novih izazova koji se odnose na izmjenu strukture i načina funkcionisanja savremene distributivne mreže. Neki od izazova svakako bi bili i optimizacija napona i snaga na izvodima. Za optimizaciju u elektroenergetici, generalno, može se reći da je ključna za postizanje energetske efikasnosti, smanjenje emisije štetnih gasova, očuvanje resursa i stvaranje održivijih energetskih sistema.

Optimizacija elektroenergetskih mreža često uključuje upotrebu matematičkih modela i algoritama kako bi se postigli željeni ciljevi uzimajući u obzir različite faktore i ograničenja. Za rješavanje optimizacionih problema, ali i mnogih drugih, često se kao resurs koristi mikromreža. Optimizacija je izvršena uspješno sa smanjivanjem gubitaka aktivne snage za 62.42%. Iz ovih brojeva može se zaključiti koliko može biti značajna integracija mikromreže, kao i adekvatna optimizacija.

5. LITERATURA

- [1] Maria Luisa Di Silvestre, Salvatore Favuzza, Eleonora Riva Sanseverino, Gaetano Zizzo: "How Decarbonization, Digitalization and Decentralization are changing key power infrastructures", Renewable and Sustainable Energy Reviews, Vol. 93, 2018.
- [2] Josip Jukić: "Tehno-ekonomska analiza mjera za smanjenje gubitaka u distributivnoj mreži", Diplomski rad, Sveučilište Josipa Jurja Strossmayera, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, Osijek, Hrvatska, 2021.
- [3] Centar Za Održivu Energetsku Tranziciju (RESET): Džemal Hadžiosmanović, Armin Đuliman, Damir Miljević, Mirza Kušljugić: "Građanska energija i demokratizacija elektroenergetskog sektora u BiH", jun 2022. Dostupno na linku: <https://reset.ba/wp-content/uploads/2022/07/Final-Gradjanska-energija-RESET.pdf>, pristup ostvaren 17.08.2023.
- [4] Nenad Katić, Menadžment sistema u elektroenergetici EMS i DMS, Predavanja, Fakultet Tehničkih Nauka u Novom Sadu, Novi Sad, 2023.
- [5] Darko Šošić, Predrag Stefanov: "Višekriterijumska optimalna rekonfiguracija distributivne mreže primenom algoritma sivih vukova", INFOTEH- JAHORINA, Vol. 16, 2017.
- [6] B. Venkatesh, R. Ranjan, H.B. Gooi, "Optimal Reconfiguration of Radial Distribution Systems to Maximize Loadability", IEEE Transactions on Power Systems, Vol. 19, Issue: 1, 2004.
- [7] Dostupno na linku: <https://etap.software.informer.com/download/>, pristup ostvaren: 08.07.2023.

Kratka biografija:



Gordana Markov rođena je u Trebinju 1999. god. Osnovne studije na Fakultetu tehničkih nauka na odsjeku Elektroenergetski sistemi završila je 2022. godine.
kontakt: gordana.markov18@gmail.com



Dr. Luka Strežoski rođen je u Novom Sadu 1990. Doktorirao je na Fakultetu tehničkih nauka 2017. god., a od 2020 je izabran za šefa katedre. Oblast interesovanja su elektroenergetski sistemi.

SAVREMENA RJEŠENJA SPOLJAŠNJEG OSVJETLJENJA PRIMJENOM LED-a

MODERN OUTDOOR LIGHTING USING LED

Milan Popović, Dejan Jerkan, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U radu je obrađena upotreba LED spoljašnjeg osvjetljenja na primjeru projekta elektroenergetskih instalacija javnog osvjetljenja stambenog bloka 3, 4 i 5 na Jugovićevoj u Novom Sadu. Cilj projekta je da se obezbijedi energetska efikasnost osvjetljenja dovoljnog kvaliteta i kvantiteta za bezbjednost i za povećanje vidljivosti i produktivnosti u prijatnom okruženju. Objašnjeni su osnovni principi rada LED tehnologije i sistema spoljašnjeg osvjetljenja, kao i poređenje softvera za projektovanje osvjetljenja.

Ključne reči: LED osvjetljenje, Energetska efikasnost, Spoljašnje osvjetljenje

Abstract In this paper use of LED outdoor lighting on the example of the electric power installations project of public lighting of the residential blocks 3, 4, and 5 on Jugovićevoj in Novi Sad is analyzed. The aim of the project is to provide energy-efficient lighting of sufficient quality and quantity for safety and to increase visibility and productivity in a pleasant environment. The basic principles of LED technology and outdoor lighting systems are explained, as well as a comparison of lighting design software.

Keywords: LED lighting, Energy efficiency, Outdoor lighting

1. UVOD

Brzina razvoja i usavršavanje LED tehnologije uslovljavaju da svi komercijalni proizvođači rasvjete postepeno odustaju od proizvodnih linija proizvoda sa klasičnim izvorom svjetla i na tržište plasiraju LED varijante proizvoda sa superiornijim tehničkim karakteristikama, dužim vijekom trajanja i velikom uštedom u potrošnji energije. Šira upotreba LED tehnologije podržana je i sve učestalijim smjernicama i direktivama iz Evropske Unije, koje se pre svega odnose na uštedu energije i energetska efikasnost, a nisu zanemarljivi ni razlozi kontrolisanog generisanja otpada i njegove reciklaže, kao i potreba za smanjenjem svjetlosnog zagađenja.

Neefikasnost tradicionalnih svetiljki dovela je do toga da se LED osvjetljenje sve učestalije koristi kao zamjensko za industrijsku, uličnu, tunelsku i ostalu rasvjetu, usled toga što svega 10% utrošene energije tradicionalnih izvora svjetlosti otpada na svjetlo, a čak 90% energije se izgubi na emitovanje toplote. Stoga, LED rasvjeta predstavlja sve češći izbor, budući da je ušteda u odnosu na klasične izvore svjetla očigledna, bez obzira na nešto skuplju vrijednost početne investicije.

NAPOMENA:

Ovaj rad proistekao je iz master rada, čiji mentor je bio dr Dejan Jerkan, docent.

U zavisnosti od načina pretvaranja električne u svjetlosnu energiju razlikuju se dve grupe svjetlosnih izvora:

1. Inkadescetni svjetlosni izvori, ili izvori koji rade na principu zagrevanja vlakna (sijalica sa vlaknom, halogena sijalica),
2. Svjetlosni izvori koji proizvode svjetlosnu energiju pri električnim pražnjenjima u smesi gasova i metalnih para (fluorescentna sijalica, živina sijalica, natrijumova sijalica).

U radu je stavljen akcenat na LED osjetljenje i njegovu primjenu na projektu elektroenergetskih instalacija javnog osvjetljenja stambenog bloka 3, 4 i 5 na Jugovićevoj u Novom Sadu. Softverski alat koji je korišćen pri analizi, proračunu i simulaciji osvjetljenja je DIALux.

2. LED OSVJETLJENJE

Naziv LED potiče od engleskog izraza „Light Emitting Diode“ (dioda koja emituje svjetlost). LED predstavlja posebnu vrstu poluprovodničke diode koja emituje svjetlost kada kroz nju teče struja. Svjetlost koju emituje može biti u vidljivom spektru, kao i u infracrvenom i ultraljubičastom. LED je poluprovodnički izvor svjetlosti koji se dugo samo koristio kao indikaciono svjetlo na raznim tehničkim uređajima.

Zahvaljujući stalnom tehnološkom napretku, LED rasvjeta ima veliku primjenu u širokom spektru područja bilo da se radi o funkcionalnoj ili dekorativnoj rasvjeti. Zbog svog inovativnog dizajna i tehnologije koja garantuje visok kvalitet svjetlosti uz nisku, ekološku orijentisanu potrošnju energije, predstavlja najefikasnije svjetlosno rješenje za svaki ambijent.

Jedna od glavnih prednosti u odnosu na standardne sijalice, jeste ta što LED svjetiljke za istu snagu daju mnogo više svjetlosti. Takođe, jačina svjetlosti koju ispušta ova sijalica uvijek ima konstantnu vrijednost, što nije slučaj kod standardne sijalice.

Opadanje svjetlosti tek kreće da se javlja pred kraj životnog vijeka same sijalice. Za LED rasvjetu se smatra da joj je prošao rok trajanja onda kada jačina svjetlosti koju ona emituje opadne na 70 % jačine svjetlosti nove LED rasvjete. Prosečni vijek LED sijalice je dosta duži nego vijek standardne sijalice. Mogu da traju i do 50.000 sati, što je oko 10 godina.

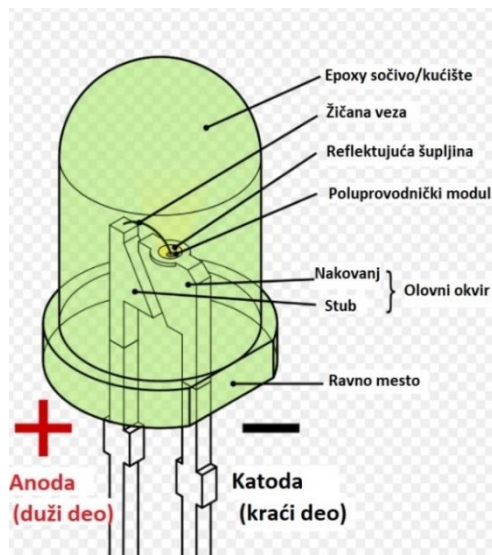
Glavne odlike su im niska potrošnja i vrlo malo generisanje toplote. To smanjuje rizik od požara i čini ih pogodnim čak i za instalaciju na mjestima gdje je teška zamjena. LED sijalice imaju odličan kvalitet boje. Nijansa bijele boje koju ispuštaju je jasna i konzistentna, a važno je napomenuti da prilikom prigušenja energije kod njih ne

dolazi do treperenja, što nije slučaj sa standardnim sijalicama.

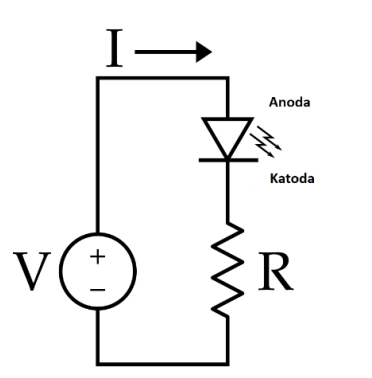
2.1. Princip rada LED diode

LED dioda se sastoji od čipa napravljenog od poluprovodnog materijala koji je dopiran nečistoćama kako bi se napravio p-n spoj. Kao i kod običnih dioda, električna struja teče od p-strane ili anode ka n-strani ili katodi, ali ne i u suprotnom smjeru. Nosioi nalektrisanja, elektroni i šupljine teku u spoj sa elektroda između kojih postoji električni napon. Kada se elektron sudari sa šupljinom, on pada na niži energetske nivo i oslobađa energiju u vidu fotona. Fotoni svjetla se emituju prilikom rekombinacije para elektron-šupljina.

Talasna dužina emitovane svjetlosti, a time i njena boja, zavisi od energetske barijere materijala koji čine p-n spoj. Kod silicijumskih i germanijumskih dioda, elektroni i šupljine se rekombinuju ne-zračećom tranzicijom, koja ne daje vidljivu emisiju, jer su oni materijali sa indirektnom energetskom barijerom. Materijali koji se koriste za izradu LED dioda imaju direktnu energetsku barijeru sa energijama koje odgovaraju skoro infracrvenoj, vidljivoj i skoro-ultraljubičastoj svjetlosti.



Slika 2.1. Delovi LED diode



Slika 2.2. Jednostavno LED strujno kolo sa otpornikom za ograničavanje struje

Ako kroz LED diodu prođe struja jača od maksimalno dozvoljene, dioda će istog trenutka pregoreti. Otpornik koji se koristi za preveniranje prekoračenja maksimalno dozvoljene struje može biti različitih otpornosti.

Ukoliko se koristi otpornik veće otpornosti, to samo znači da će LED sijalica slabije svjetleti.

Tabela 1: Vrijednosti otpornika za najčešće korišćene LED diode

Napon kola	Vrijednost otpornika
3,3 - 5 V	330 Ω
6 - 9 V	560 Ω
12 - 15 V	1 k Ω

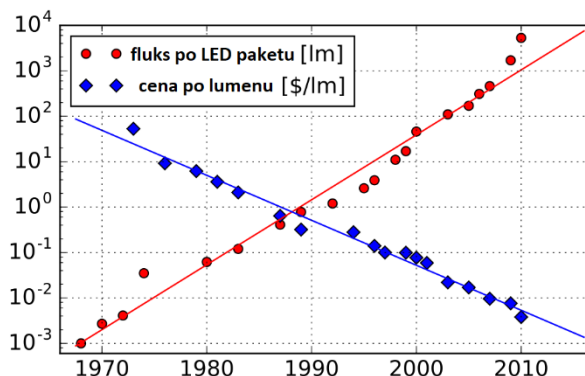
2.3. Odnos cijene i kvaliteta

Haitz-ov zakon je zapažanje i predviđanje stalnog poboljšanja LED dioda, posmatrano tokom dužeg niza godina. Ovaj zakon tvrdi da svake decenije cijena po lumenu (tj. cijena po jedinici korisne svjetlosti koja se emituje) pada za faktor 10, a da se količina svjetlosti koja se generiše po LED paketu povećava za faktor 20, za datu talasnu dužinu (boju) svjetlosti. Haitz-ov zakon oslanja se na optimizaciju procesa proizvodnje poluprovodničkih uređaja.

2.2. Šema napajanja LED diode

Struja u LED ili drugim diodama raste eksponencijalno sa primjenjenim naponom, tako da mala promjena napona može izazvati veliku promjenu struje.

Struja kroz LED diodu mora biti regulisana spoljnim kolom kao što je izvor konstantne struje da bi se sprečilo oštećenje. Pošto su najčešći izvori napajanja izvori (skoro) konstantnog napona, LED lampe moraju da sadrže konvertor ili bar otpornik za ograničavanje struje.



Slika 2.3. Ilustracija Haitz-ovog zakona, koja pokazuje poboljšanje izlazne svjetlosti po LED diodi s vremenom (gdje je na vertikalnoj osi korišćena logaritmička skala)

Prve LED diode koje su emitovale bijelu svjetlost bile su skupe i neefikasne. Međutim, izlaz svjetlosti LED dioda se eksponencijalno povećavao. Najnovija istraživanja i razvoj propagirali su mnogobrojni proizvođači (pre svega japanski, kineski i korejski).

Svjetlosna snaga i efikasnost plavih i skoro ultraljubičastih LED dioda su porasle, a cijena pouzdanih uređaja je pala. Ovo je dovelo do relativno jakih LED dioda bijele svjetlosti za osvjetljenje, koje zamjenjuju inkandescentno i fluorescentno osvjetljenje. Ukoliko se poredi LED i tradicionalno osvjetljenje, LED sijalice su doživele ogromno povećanje električne efikasnosti tokom godina i iako je LED osvjetljenje i dalje skuplje za kupovinu,

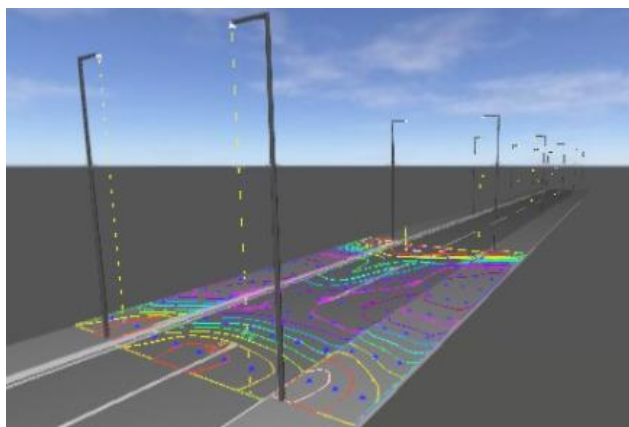
odnos cijene i kvaliteta je znatno bolji nego kod sijalica sa užarnom niti (uzimajući u obzir životni vijek i energetske efikasnost obje vrste osvjjetljenja).

3. SOFTVERI ZA PROJEKTOVANJE OSVJETLJENJA

Softveri za projektovanje osvjjetljenja se, u slučaju projektovanja spoljašnjeg osvjjetljenja, koriste za analizu fotometrije (način distribucije svjetlosti u prostoru), 2D i/ili 3D projektovanje sistema rasvjete, kao i simulaciju različitih sjenarija rasvjete.

U suštini, glavna svrha ove vrste softvera je, nakon završenog procesa modelovanja fizičkih i fotometrijskih karakteristika sistema javne rasvjete, dobijanje i predstavljanje rezultata kombinacijom grafičkih i numeričkih podataka. Zbog toga, najbitnije karakteristike rasvjetnih softvera predstavljaju moć programa za izvršavanje proračuna i opseg mogućnosti funkcije renderinga, koja se odnosi na konvertovanje modela u 2D slike.

Za analizu sistema javne rasvjete posebno je bitna sposobnost za brze i kvalitetne proračune, koji se izvršavaju za određene tačke osvjjetljenih površina, što omogućuje jednostavnost simulacija i fleksibilno prikazivanje rezultata.



Slika 3.1. *Primjer simulacije spoljašnjeg osvjjetljenja*

U nastavku je dat kratak pregled Relux i DIALux softvera za projektovanje osvjjetljenja, s tim da je bitno naglasiti da program DIALux predstavlja najpopularniji softver za projektovanje i simulaciju javne rasvjete, naročito među inženjerima.

Takođe, DIALux je najbolji u kategorijama brzine i kvaliteta dobijenih rezultata za vještačke izvore svjetlosti. Jednostavnost, kompatibilnost, široka baza proizvođača i podataka, besplatna licenca, podrška i osnovna obuka, samo doprinose popularnosti ovog softvera.

3.1. Relux softver

Relux je besplatni, intuitivno upravljiv program za simulaciju osvjjetljenja koji se prvenstveno koristio samo za vještačko osvjjetljenje, a danas se koristi i za proračune vještačkog i dnevnog osvjjetljenja. Osnovan je od strane švajcarske kompanije, koja se već 20 godina bavi specijalizovanjem razvoja softvera za planiranje osvjjetljenja i prezentaciju proizvoda. Najpopularniji je na evropskom tržištu. Koristi dva algoritma kao metode za svoje proračune i dozvoljava

korisniku mogućnost izbora. Algoritam za praćenje zraka svjetlosti je unapređena verzija Radiance programa.

Simulacija osvjjetljenja i senzora je omogućena u realnom vremenu. Dakle, Relux predstavlja prilagođeni program čiji je cilj izvršenje proračuna osvjjetljenja i uključuje čitav niz objekata, kao i kompatibilan interfejs sa AutoCAD-om.

3.2. DIALux softver

DIALux predstavlja besplatan program za dizajniranje osvjjetljenja, razvijen od strane nemačke kompanije 1994. godine. Slično Relux-u, korišten je prvo samo za vještačko osvjjetljenje, iako ima mogućnost uvažavanja dnevnog svjetla. Danas se koristi za proračune kako dnevnog, tako i vještačkog osvjjetljenja, sigurnosnog osvjjetljenja, osvjjetljenja javnih površina, planiranje i izradu dokumentacije.

DIALux koristi algoritam zračenja za svoje proračune. Omogućava dizajniranje uz maksimalnu upotrebu dnevne svjetlosti i energetske efikasne svjetiljke.

Interfejs je nešto komplikovaniji od interfejsa koji koristi Relux softver. Trenutna verzija DIALux-a ima ograničen osnovni opseg objekata i tekstura koje može modelovati. Trenutno je najisplativiji softver za sve proračune osvjjetljenja jer ne postoji naknada za licencu. Softver je primenljiv za složene, kvalitetne proračune kao i foto-realistične prikaze.

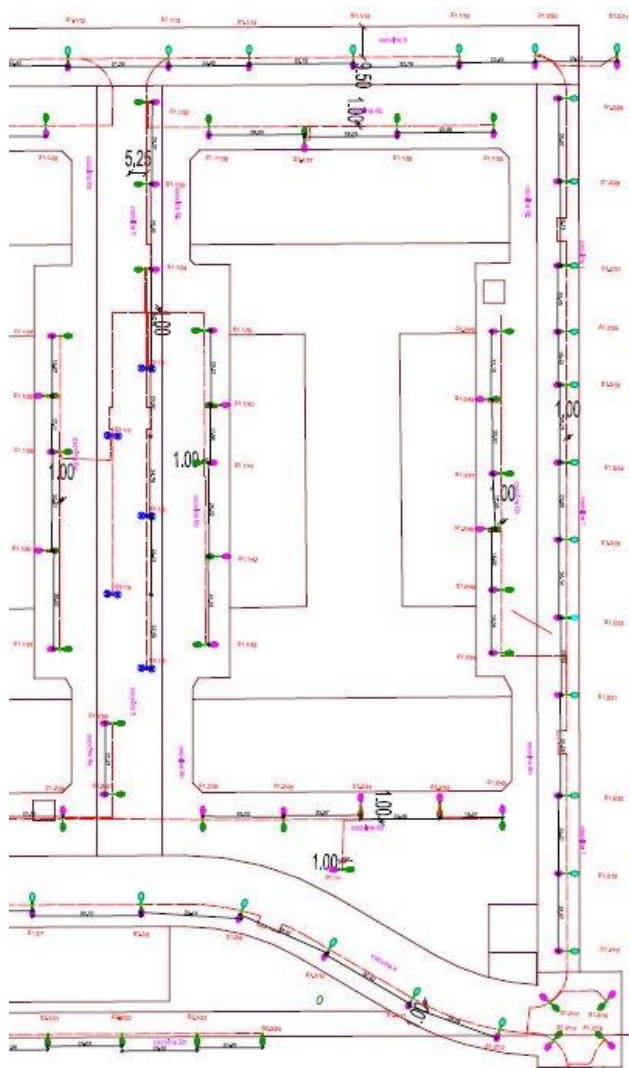
4. PROJEKAT JAVNOG OSVJETLJENJA SAOBRAĆAJNIH POVRŠINA ZA POTREBE OPREMANJA STAMBENOG BLOKA 3, 4 I 5 NA JUGOVIĆEVU U NOVOM SADU

Svjetlotehnički proračun urađen je računarskim programom, dok rezultati proračuna zadovoljavaju važeće standarde za predmetnu lokaciju javnog osvjjetljenja. Projektovanje osvjjetljenja zahteva poznavanje uticaja rasvjete na ljudsko ponašanje, produktivnost i osećaj komfora, ali traži i „osećaj“ za lijepo. Realnu vizuelizaciju tokom projektovanja omogućavaju profesionalni programi za planiranje i preciznu kalkulaciju dizajna rasvjete.

Za proračun osvjjetljenja korišćen je softver Dialux, koji je vodeći softver za projektovanje rasvjete. Za iste svrhe se može koristiti i Relux, koji je takođe široko rasprostranjen. Uz pomoć softvera na jednostavan način se omogućava postavljanje položaja svjetiljki, sa zadatkom da se definiše dovoljna osvjjetljenost za određenu prostoriju, kao i snaga izvora svjetlosti.

Kod načina izbora svjetlotehničke klase za osvjjetljenje saobraćajnice (M3, M4), rizičnog područja (C1) i pješakih zona (P1) su se uvažavale kako preporuke date u Master planu osvjjetljenja Grada Novog Sada iz 2012 godine, tako i važeći standardi Evropske Unije za osvjjetljenje puteva za metodologije izbora klasa i parametara vrijednovanja sa važećom preporukom. Međunarodne komisije za osvjjetljenje saobraćajnica (CIE 115), kao i izrada novog poglavlja u kome su definisani indikatori energetske performansi. Evropski standard EN 13201 iz 2016. godine, kao i važeći SRPS standard SRPS EN 13201.

U radu su uvaženi svi kriterijumi za izbor klasa osvjjetljenja i svjetlotehničkih zahtjeva predmetnog projekta.



Slika 4.1. Blok šema jednog dijela javnog osvjetljenja za stambeni blok 3, 4 i 5 na Jugovićevo u Novom Sadu

5. ZAKLJUČAK

U radu su izloženi osnovni principi projektovanja instalacija osvjetljenja u uslovima mješovitih svjetlotehničkih zahteva. U današnje vrijeme postoji širok spektar električnih izvora svjetlosti, kojima se može postići kvalitetno osvjetljenje, kako unutrašnjosti, tako i spoljašnjosti objekata raznih namjena.

U svrhu postizanja kvalitetnog osvjetljenja, u projektnoj praksi neizbježna je primjena nekog od programa za proračun osvjetljenja, pomoću kojih se lako mogu proračunati parametri rasvjete, pre nego što se pristupi njegovom fizičkom izvođenju. Istovremeno, neophodno je proračunati i zahteve za električnom energijom projektovane instalacije osvjetljenja, zbog pravilnog odabira opreme za njeno napajanje.

Jedan od glavnih uslova koji se nameće modernim elektro-energetskim objektima jeste energetska efikasnost, koja se najbolje postiže upotrebom LED svjetlosnih izvora.

Za izabrani tip svjetiljki je izvršen detaljan fotometrijski proračun u softveru Dialux, na osnovu kojeg se ustanovio optimalan broj i pozicija izvora svjetlosti za osvjetljavanje saobraćajnih površina stambenog bloka 3, 4 i 5 na Jugovićevo u Novom Sadu.

6. LITERATURA

- [1] D. Petrović, „Električno osvjetljenje“, Tehnička knjiga, 1985.
- [2] M. Kostić, „Vodič kroz svet tehnike osvjetljenja“, Minel-Schreder, Beograd, 2000.
- [3] S. Gušavac, „Osnovni principi projektovanja u mrežama srednjeg i niskog napona“, Fakultet tehničkih nauka u Novom Sadu, 2014.
- [4] <https://fluxtechnology.rs>, Korisne informacije o LED rasveti
- [5] <https://dialux.com>

Kratka biografija:

Milan Popović rođen je u Brčkom 1995. godine. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnika i računarstvo – Energetska elektronika i električne mašine odbranio je 2020. god. Iste godine upisao je master studije.

Dejan Jerkan je docent na Fakultetu tehničkih nauka u Novom Sadu, na Katedri za Energetsku elektroniku i pretvarače. Oblast interesovanja su mu modelovanje i dijagnostika električnih mašina, kao i metoda konačnih elemenata

OPTIMIZACIJA NAPONA I SNAGA UNUTAR MIKROMREŽE**OPTIMIZATION OF VOLTAGE AND POWER WITHIN THE MICROGRID**Ivana Košutić, Luka Strezoski, *Fakultet tehničkih nauka, Novi Sad***Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**

Kratak sadržaj – U radu su obrađene teorijske osnove o mikromrežama, DER-ovima, optimizaciji u elektroenergetskim sistemima, kao i matematički modeli vodova, transformatora i DER-ova koji će biti korišćeni dalje u analizi. U toku istraživanja, sprovedena je optimizacija sistema sa jasnim ciljem - minimizacijom gubitaka aktivne snage uz poštovanje naponskih ograničenja. Ovaj proces optimizacije nije usmjeren na globalno rješenje, već je pažljivo pristupljeno svakom resursu pojedinačno. Pronađeni su lokalni optimumi za svaki resurs, uzimajući u obzir njihove specifične karakteristike i ograničenja. Ova metoda pristupa omogućila je postizanje boljeg balansa između proizvodnje i potrošnje energije u mikromreži, čime se unapređuje efikasnost sistema i osigurava njegova stabilnost.

Ključne reči: Mikromreža, Distribuirani generatori, Obnovljivi izvori, Optimizacija

Abstract – The paper discusses the theoretical foundations of microgrids, DERs, optimization in electrical power systems, as well as mathematical models of lines, transformers, and DERs that will be used further in the analysis. During the research, the system optimization was carried out with a clear goal - minimization of active power losses while respecting voltage constraints. This optimization process is not aimed at a global solution, but each resource is carefully approached individually. Local optima have been found for each resource, taking into account their specific characteristics and constraints. This approach method has made it possible to achieve a better balance between energy production and consumption in the microgrid, thus improving the efficiency of the system and ensuring its stability.

Keywords: Microgrid, Distributed Generators, Renewable Energy, Optimization

1. UVOD

Savremeno društvo se suočava s izazovima u elektroenergetskim sistemima koji zahtijevaju viši stepen pouzdanosti, efikasnosti i održivosti. Tradicionalni modeli generacije i distribucije električne energije nailaze na neizvjesnost snabdevanja i negativnom uticaju na životnu sredinu. Koncept distribuiranih generatora (DG) u mikromrežama postavlja nov standard u arhitekturi elektroenergetskih sistema.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Luka Strezoski, vanr. prof.

DG sistemi se smještaju direktno na mjestima potrošnje, omogućavajući modularnost, skalabilnost i smanjenje emisije štetnih gasova. Optimizacija mikromreža, uključujući Volt/Var optimizaciju, igra ključnu ulogu u održavanju napona unutar tehničkih ograničenja i poboljšanju efikasnosti sistema.

Formulisane kriterijumske funkcije je ključni korak u optimizaciji mikromreža, obično se cilja na minimizaciju gubitaka aktivne snage uz očuvanje naponskih ograničenja. Različiti pristupi se koriste u optimizaciji mikromreža, uključujući stohastičke (genetski algoritmi, optimizacija rojevima čestica) i determinističke metode (linearna, kvadratna, cjelobrojna optimizacija). Optimizacija se primjenjuje za poboljšanje efikasnosti, stabilnosti i smanjenje gubitaka u mikromreži.

Rad se sastoji od šest poglavlja, uključujući definicije mikromreža, analizu DG-a, opis optimizacije, matematičke modele i proračune te detaljnu optimizaciju mikromreže sa ciljem smanjenja gubitaka aktivne snage uz optimalno iskorišćenje resursa [2].

2. KONCEPT MIKROMREŽE

Konvencionalni izvori energije se suočavaju s problemima smanjenja rezervi fosilnih goriva, efikasnosti i ekološkog zagađenja, što je promovisalo upotrebu distribuirane proizvodnje električne energije putem tehnologija poput vjetrogeneratora, fotonaponskih ćelija i gorivnih ćelija. DG-ri pružaju održivu i pouzdanu energetska mrežu, smanjujući zavisnost od tradicionalnih izvora energije, smanjujući gubitke tokom prenosa i podržavajući integraciju obnovljivih izvora energije (OI). Integracija DG-a potiče inovacije u upravljanju energijom, poboljšava energetska efikasnost i doprinosi bržem oporavku energetske sistema nakon prirodnih katastrofa. Mikromreže, koje uključuju lokalne izvore energije i uređaje za skladištenje, pružaju stabilnost, pouzdanost i energetska efikasnost sistema. Mikromreže omogućavaju i povećanu autonomiju u donošenju odluka i aktivno učešće potrošača na energetskom tržištu [3].

Mikromreža može raditi u dva režima:

1. Povezani režim rada - kada je povezana sa glavnom mrežom, što znači da je u nesamostalnom pogonu.
2. Ostrvski režim rada - kada nije povezana sa glavnom mrežom, pa funkcioniše autonomno.

Pametne mreže i daljinsko upravljanje čine mikromreže još efikasnijim. Ovaj koncept predstavlja ključni korak realizaciju prethodno pomenutih upravljačkih strategija prema održivijoj i pouzdanijoj energetskoj budućnosti, istovremeno podržavajući ekonomski razvoj i otpornost na promjene.

3. DISTRIBUIRANI ENERGETSKI RESURSI

Distribuirana proizvodnja električne energije (DER) predstavlja promjenu u načinu generisanja električne energije, fokusirajući se na male modularne generatore od 1 kW do 1 MW, smještene blizu potrošača. Ovi DER-ovi pružaju fleksibilnost, ekonomske prednosti i ekološke koristi. Mogućnost brzog pokrivanja povećane potrošnje, modularna konstrukcija i prilagodljivost lokacije čine DER-ove ekonomski privlačnim. Takođe, DER-ovi omogućavaju efikasno kombinovanje proizvodnje električne i toplotne energije. Sa operativnog aspekta, DER-ovi poboljšavaju naponske uslove, smanjuju gubitke energije i doprinose većoj pouzdanosti snabdevanja, uključujući i ruralna područja. Takođe, DER-ovi su ekološki prihvatljiviji i smanjuju opterećenje prenosnih kapaciteta.

Sveobuhvatan pristup DER-ovima donosi brojne ekonomske i operativne prednosti, pružajući efikasna i održiva energetska rješenja [4].

4. OPTIMIZACIJA U ELEKTROENERGETSKIM SISTEMIMA

Optimizacijski problemi se bave pronalaskom ekstremnih vrijednosti ciljne funkcije u okviru određenih ograničenja. Postoje različite podjele unutar optimizacije, ali za inženjerske probleme su posebno relevantne lokalne i globalne metode optimizacije. Što je problem složeniji, to je teže pronaći globalni optimum, pa se često moramo zadovoljiti lokalnim optimumima.

Pronalaženje inženjerski prihvatljivih rješenja, posebno u složenim problemima, može biti izazovno. Početno rješenje često zavisi o iskustvu inženjera i može biti daleko od globalnog optimuma. Stoga se često koristi tehnika pokretanja algoritma iz više početnih tačaka kako bi se pronašlo najbolje lokalno rješenje, iako to ne garantuje globalnu optimalnost. Za složene probleme sa mnogo lokalnih optimuma, potrebne su metode globalne optimizacije koje pretražuju čitav prostor rješenja kako bi osigurale najbolje moguće rješenje [5].

4.1. OPTIMIZACIJA MIKROMREŽE

Problemi optimizacije u kontekstu rada mikromreža, ekonomskog dispečinga i proračuna optimalnog angažovanja agregata su ključni za efikasno upravljanje mikromrežama. Ovi problemi uključuju analizu optimalnih tačaka rada i režima različitih DG, sistema za skladištenje energije, emisiju CO₂, potrošnju električne energije, uvoz i izvoz iz glavne mreže, broj prekida opreme, radne troškove i mnoge druge faktore. Kriterijumi optimizacije u ovom kontekstu mogu biti ekonomske prirode, tehnički, usmjereni na očuvanje životne sredine ili kombinacija ovih faktora.

Cilj ovih istraživanja je pronaći najefikasnija rješenja koja su ekonomski isplativa, tehnički izvodljiva i ekološki prihvatljiva za rad mikromreža.

Za postizanje ovih ciljeva, primjenjuju se različite metode optimizacije, uključujući stohastičke metode vještačke inteligencije (kao što su genetski algoritmi i optimizacija rojevima čestica), kao i determinističke metode zasnovane na linearnoj, kvadratnoj i cjelobrojnoj optimizaciji [3]. Optimalna regulacija napona i reaktivnih snaga (Volt/Var

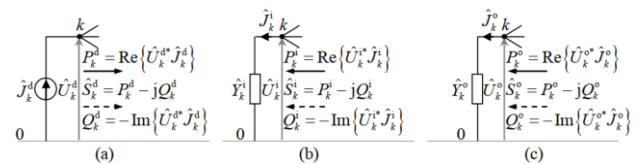
optimizacija - VVO) je ključna komponenta upravljanja mikromrežama. Ciljevi VVO uključuju održavanje napona unutar tehničkih ograničenja, smanjenje štete na potrošačkoj strani zbog odstupanja napona i smanjenje gubitaka snage u distributivnoj mreži. Postizanje ovih ciljeva zahteva koordinaciju između različitih komponenti sistema, uključujući regulacione transformatore, kompenzacione uređaje i distribuirane generatore. Ova regulacija se postiže kontrolom regulacionih sklopki, podešavanjem regulacionih sklopki i postavljanjem kondenzatorskih baterija na odgovarajućim mjestima. Da bi se postigli optimalni rezultati u VVO, razmatraju se različiti optimizacioni kriterijumi kao što su minimalna devijacija napona, minimalni gubici snage i minimalna potrošnja. Ovi kriterijumi se koriste kako bi se odredile optimalne pozicije regulacionih sklopki i kondenzatorskih baterija za postizanje željenih ciljeva uzimajući u obzir tehničke i ekonomske faktore.

5. POSTAVKA PROBLEMA

U ovome dijelu rada biće opisani matematički modeli vodova, regulacionog transformatora i modeli DER-ova za proračun tokova snaga

5.1. MODELI DER-ova ZA PRORAČUNE TOKOVA SNAGA AKTIVNIH DISTRIBUTIVNIH MREŽA

Pogonska kola DER-a sva tri redoslijeda prikazana su na slici 5.1.1.



Slika 5.1.1. Pogonska kola razmatranog trofaznog DERk direktnog (a), inverznog (b) i nultog redoslijeda (c) [1]

1. Napon direktnog redoslijeda na priključku savremenog DER na mrežu diktiraju i DER i mreža (DER preko svoje upravljačke strategije). Rezultat je injektirana struja direktnog redoslijeda.
2. Napon inverznog redoslijeda na priključku savremenog DER a mrežu diktiraju i DER i mreža (DER preko svoje upravljačke strategije, ukoliko se specificira struja inverznog redoslijeda različita od nule). Rezultat je injektirana struja inverznog redoslijeda.
3. Kod savremenih DER se nulta komponenta struje uvek anulira, pa je nulto pogonsko kolo uvijek u praznom hodu [1].

Koristeći matricu transformacije i poštujući smjer veličina prikazan na prethodnoj slici, trofazne snage DER_k, i tradicionalnih naizmjeničnih mašina i savremenih DER-ova, mogu izraziti u domenu simetričnih komponenti:

$$\begin{aligned} \hat{S}_{k\Sigma} &= P_{k\Sigma} + jQ_{k\Sigma} = 3(\hat{S}_k^+ - \hat{S}_k^- - \hat{S}_k^0) = 3(\hat{U}_k^+ \hat{I}_k^{+*} - \hat{U}_k^- \hat{I}_k^{-*} - \hat{U}_k^0 \hat{I}_k^{0*}) \\ &= 3[(\text{Re}\{\hat{U}_k^+ \hat{I}_k^{+*}\} + j\text{Im}\{\hat{U}_k^+ \hat{I}_k^{+*}\}) - (\text{Re}\{\hat{U}_k^- \hat{I}_k^{-*}\} + j\text{Im}\{\hat{U}_k^- \hat{I}_k^{-*}\}) - \\ &\quad - (\text{Re}\{\hat{U}_k^0 \hat{I}_k^{0*}\} + j\text{Im}\{\hat{U}_k^0 \hat{I}_k^{0*}\})] \end{aligned} \quad (5.1.1.)$$

OSNOVNE VELIČINE	$P_k^+, Q_k^+, U_k^+, \theta_k^+, P_k^-, Q_k^-, U_k^-, \theta_k^-, P_k^0, Q_k^0, U_k^0, \theta_k^0$
IZVEDENE VELIČINE	$\widehat{f}_k^+ = \frac{P_k^+ - jQ_k^+}{U_k^+ e^{-j\theta_k^+}}, \widehat{f}_k^- = \frac{P_k^- - jQ_k^-}{U_k^- e^{-j\theta_k^-}}, \widehat{f}_k^0 = \frac{P_k^0 - jQ_k^0}{U_k^0 e^{-j\theta_k^0}}$

Tabela 5.1.1. Veličine u domenu simetričnih komponenti asocirane trofaznom DER [1].

Upravljačke strategije za DER-ove variraju zavisno o njihovim specifičnostima i tipovima. Neki parametri su unaprijed poznati ili definirani za određene tipove DER-ova, dok se ostali moraju izračunati. Ova raznolikost omogućava detaljnije analize i simulacije različitih tipova DER-ova u elektroenergetskom sistemu. Detaljnije istraživanje upravljačkih strategija u različitim tipovima DER-ova zahtjeva dodatnu analizu i nije obuhvaćeno u ovom kontekstu [1].

5.2. Modeli DER-ova za proračune tokova snaga aktivnih distributivnih mreža

Algoritam počinje inicijalizacijom postupka, odnosno učitavanjem podataka o mreži, numeracijom grana i čvorova po lejerima, setovanjem indeksa iteracija (h) na početnu vrijednost. Nakon inicijalizacije započinje sam iterativni postupak proračuna radnog režima mreže. Sam proračun se sprovodi u tri koraka, koji se ponavljaju iz iteracije u iteraciju, pri čemu se na kraju svake od iteracija vrši provjera zadovoljenosti kriterijuma konvergencije, ako prethodno deklarirani kriterijum nije zadovoljen, ulazi se u narednu iteraciju proračuna, odnosno ako je kriterijum zadovoljen, izlazi se iz proračuna [1].

1. Korak: Proračun injektiranih struja
2. Korak: Proračun struja po granama
3. Korak: Proračun napona u čvorovima

Konkretno, u kreiranom programskom proračunu zadati kriterijum konvergencije postavljen je na vrednost $\varepsilon = 0.000001$, a sam broj iteracija ograničen je na 50, u suprotnom, smatra se da proračun divergira.

6. REZULTATI IZ ANALIZE MIKROMREŽE

U ovom istraživanju, osnovni model predstavlja modifikiranu IEEE 13-bus testnu mrežu. U radu za modelovanje i proračun korišten je program ETAP, besplatna verzija 19.0.1, skinuta 15.06.2023. sa sajta koji će biti naveden u literaturi. Ova verzija ETAP-a je testirana od strane Microsoft SQL Server DBMS-a [6].

6.1. Optimizacija solarnih elektrana

U istraživanju je analizirano smanjenje gubitaka aktivne snage u mreži uz poštivanje naponskih ograničenja. Početno, solarna elektrana koristi 510 kW od ukupnog kapaciteta od 800 kW, a zatim postupno mijenja svoju snagu u koracima od 10% kako bi pronašla rješenje koje zadovoljava definirane kriterije. Optimalna vrijednost za P11 sa 510kW na 662KW, P22 sa 510kW na 458.8KW. Nakon optimizacije, snaga iz distributivne mreže se smanjila, uz blagu redukciju snage konvencionalnih generator. Povećanje snage solarnih elektrana smanjilo je ukupne gubitke aktivne snage za približno 3.1%. Ovaj rezultat naglašava važnost efikasnog upravljanja energijom za smanjenje gubitaka u mikromreži. U tabeli 6.1.1. prikazani su ukupni gubici u mikromreži.

UKUPNI GUBICI PRIJE OPTIMIZACIJE	22.6kW
UKUPNI GUBICI NAKON OPTIMIZACIJE SOLARNE ELEKTRANE P11 i P22	21.9kW

Tabela 6.1.1.. Ukupni gubici nakon optimizacije solarnih elektrana

6.2. Optimizacija baterije

U istraživanju su analizirani uticaj variranja snage baterije na performanse mikromreže s početnim kapacitetom baterije od 30 kW, gdje je inicijalna snaga baterije iznosila 15.12 kW. Kroz dva različita istraživanja, proučavani su efekti povećanja i smanjenja snage baterije u odnosu na tu početnu vrijednost. Povećanje snage baterije može imati različite uticaje na napone i ukupne gubitke u mikromreži, u zavisnosti od konfiguracije mreže. Povećanje snage baterije može pomoći u održavanju stabilnih naponskih prilika u mreži, smanjiti ukupne gubitke energije, poboljšati faktor snage i stabilnost napona, te bolje upravljati opterećenjem.. U tabeli 6.2.1. prikazani su ukupni gubici u mikromreži.

UKUPI GUBICI MIKROMREŽE NAKON OPTIMIZACIJE SOLARNE ELEKTRANE P11 i P22	21.9kW
UKUPI GUBICI MIKROMREŽE NAKON OPTIMIZACIJE BATERIJE	21.7kW

Tabela 6.2.1.. Ukupni gubici nakon optimizacije solarnih elektrana

U optimalnom scenariju za bateriju snage od 23 kW postignuti su najbolji rezultati. Gubici u mreži su smanjeni sa 21.9 kW na 21.7 kW, što je smanjenje od otprilike 1%. Takođe, napon na čvoru gdje je baterija povezana povećan je sa 96.16% na 96.25%, poboljšavajući stabilnost i kvalitet električne energije u mreži. Potreba za snagom iz distribucione mreže smanjena je sa 190.1 kVA na 187.9 kVA. Ostali resursi u mreži nisu doživjeli promjene. Ovaj optimalni scenario doprinosi unapređenju energetske efikasnosti i održivosti elektroenergetskih sistema.

6.3. Optimizacija kondenzatora

U okviru ovoga istraživanja, izvršena su detaljna ispitivanja uticaja variranja kapacitivnosti kondenzatora na performanse mikromreže. Kapacitivnost kondenzatora ima značajan uticaj na ukupne gubitke i naponske prilike u električnim sistemima. Ukupna kapacitivnost kondenzatora je 3x80kVAr, inicijalno stanje podrazumevalo je kapacitivnost kondenzatora od 3x40kVAr.

UKUPNI GUBICI U MIKROMREŽI NAKON OPTIMIZACIJE BATERIJE	21.7kW
UKUPI GUBICI MIKROMREŽE NAKON OPTIMIZACIJE KONDEZATORA CAP1	21.4kW

Tabela 6.3.1.. Ukupni gubici nakon optimizacije solarnih elektrana

U tabeli 6.3.1. prikazani su ukupni gubici u mikromreži. Analizom raspodjele tokova snaga po granama prije i nakon optimizacije kondenzatora, primjećuje se da postoji povećanje napona na čvorovima gdje su kondenzatori povezani, preciznije sa 96.25% na 67.06%, kao i sa 96.27% na 97.08%.

Kao zaključak, možemo utvrditi da povećanje kapaciteta kondenzatora u istraživanoj mikromreži, sa 40 kVAR na 56 kVAR za oba kondenzatora, pozitivno utiče na naponske profile čvorova u kojima su kondenzatori povezani. Kad su u pitanju ukupni gubici aktivne snage, uočeno je smanjenje sa 21.7 kW na 21.4 kW. Kapacitivni kondenzatori poboljšavaju faktor snage i smanjuju gubitke u mreži kompenzirajući reaktivnu snagu.

Također, održavaju stabilnost naponskih prilika. Važno je pažljivo odabrati kapacitivnost kako bi se postigao željeni efekt bez negativnih posljedica.

6.4. Optimizacija dizel generatora

U mikromreži se koristi dizel generator kapaciteta 750 kW, a kroz dvije studije provedene su promjene u njegovoj snazi. Prvo istraživanje povećalo je snagu za 10%, dostižući 660 kW, dok je drugo smanjilo snagu za 10% na 540 kW kako bi se istražili utjecaji na distribuciju energije, gubitke i stabilnost mikromreže, uz poštivanje naponskih ograničenja. Optimalno održavanje snage dizel generatora na 540 kW pokazalo se najboljim, budući da povećanje na 660 kW značajno povećava gubitke u mikromreži. Snaga od 540 kW rezultira smanjenjem ukupnih gubitaka na 21.3 kW i promjenama u tokovima snaga iz distributivne mreže. Potreba za snagom iz DM povećana sa 176.9 kW na 209 kW. Dodatno smanjenje snage za 10% dalje pogoršava ukupne gubitke i može dovesti do nestabilnosti sistema te promjene raspodjele snaga i naponskih prilika. Ova analiza naglašava ključnost optimizacije rada dizel generatora za postizanje energetske učinkovitosti rezultata mikromreže.

6.5. Analiza rezultata

U tabeli 6.5.1. prikazani su sažeti rezultati dobijeni iz prethodnih odeljaka. Prvi red u tabeli prikazuje rezultate prije optimizacije. Prvo je izvršena optimizacija solarnih elektrana, a zatim optimizacija baterija. Nakon toga, usledilo je traženje optimalne kapacitivnosti kondenzatora, ispitivanje uticaja dizel generatora na mikromrežu, kao i analiza uticaja promjene pozicije regulacione sklopke. U tabeli 6.5.1. prikazani su ukupni gubici u mikromreži.

UKUPNI GUBICI U MIKROMREŽI PRIJE OPTIMIZACIJE	22.6 kW
UKUPI GUBICI MIKROMREŽE NAKON OPTIMIZACIJE SOLARNE ELEKTRANE P11 i P22	21.9 kW
UKUPI GUBICI MIKROMREŽE NAKON OPTIMIZACIJE BATERIJE	21.7 kW
UKUPI GUBICI MIKROMREŽE NAKON OPTIMIZACIJE KONDEZATORA CAP1	21.5 kW
UKUPI GUBICI MIKROMREŽE NAKON OPTIMIZACIJE KONDEZATORA CP2	21.4 kW
UKUPNI GUBICI MIKROMREŽE NAKON OPTIMIZACIJE DIZEL GENERATORA GEN1	21.3 kW
UKUPNI GUBICI MIKROMREŽE NAKON PRONALAZENJA OPTIMALNE POZICIJE REGULACIONE SKLOPKE	21.3kW

Tabela 6.5.1. Poređenje rezultata nakon optimizacije

Dobijeni rezultati optimizacije jasno ukazuju na značajno smanjenje gubitaka snage, sa 22.6 kW na 20.3 kW, predstavljajući smanjenje od 5.75%. Nakon implementacije svih optimizacijskih koraka, ukupni gubici su smanjeni na 21.3 kW. U odnosu na ukupnu proizvodnju mikromreže od 2 MW, ovi gubici čine svega oko 1.065%, što se smatra izuzetno prihvatljivim i efikasnim postignućem. Trebalo bi naglasiti da ovo istraživanje, iako nije identifikovalo globalni optimum, uspešno je pronašlo niz lokalnih minimuma. Navedeni minimumi postignuti su za određene vrijednosti podešavanja solarnih elektrana, baterija, snage dizel generatora, kapacitivnosti kondenzatora i pozicije regulacione sklopke. Ciljno usmerenje prema minimizaciji gubitaka aktivne snage vodilo je ovo istraživanje, kao korak ka unapređenju i optimizaciji mikromreže.

7. ZAKLJUČAK

Optimizacija je ključna u mikromrežama za postizanje efikasnosti, gubici snage se smanjuju sa pažljivim pristupom svakom resursu. Lokalni optimumi se pronalaze uzimajući u obzir karakteristike i ograničenja resursa. U ovome istraživanju to je rezultiralo smanjenjem gubitaka snage za 5.75%, poboljšavajući ravnotežu između proizvodnje i potrošnje. Nakon svih koraka optimizacije, ukupni gubici padaju na 1.065% od ukupne proizvodnje, postižući visoku energetske efikasnost.

Održivost i pouzdanost sistema osiguravaju se pažljivim upravljanjem resursima, stvarajući održiva energetska rešenja za budućnost.

8. LITERATURA

- [1] L.Strezoski: *Modelovanje i fundamentalni proračuni aktivnih distributivnih mreža*; Fakultet tehničkih nauka, Novi Sad, 2021.
- [2] Goran Švenda, Ivana Krstić, Sonja Kanjuh, Milena Jajčanin, Dejan Vuletić, "VOLT VAR WATT OPTIMIZATION IN DISTRIBUTION NETWORK WITH HIGH PENETRATION OF RENEWABLE ENERGY SOURCES AND ELECTRIC VEHICLES", 2022.
- [3] Aleksandr Selakov, "Optimalno upravljanje mikromrežama u karakterističnim radnim režimima", Doktorska disertacija, Univerzitet u Novom Sadu, Novi Sad, Srbija, 2017.
- [4] Angel A. Bayod-Ru'jula, "Future development of the electricity systems with distributed generation", dostupno na sajtu ACADEMIA pristup odobren : 24.07.2023.
- [5] Željko Popović, Analiza i upravljanje distributivnih mreža, predavanja, Fakultet tehničkih nauka u Novom Sadu, Novi Sad 2022.
- [6] Link: <https://etap.software.informer.com/download/>, pristup ostvaren: 15.06.2023.

Kratka biografija:



Ivana Košutić rođena je u Gacku 1999. god. Osnovne studije na Fakultetu tehničkih nauka na odsijeku elektroenergetski sistemi završila je 2022. kontakt: ivanakosutic13@gmail.com



Dr. Luka Strezoski rođen je u Novom Sadu 1990. Doktorirao je na Fakultetu tehničkih nauka 2017. god., a od 2020 je izabran za šefa katedre. Oblast interesovanja su elektroenergetski sistemi.

INTEGRACIJA CHAT-GPT-A SA TEHNOLOGIJAMA SEMANTIČKOG VEB**INTEGRATING CHAT-GPT WITH SEMANTIC WEB**Tara Pogančev, *Fakultet tehničkih nauka, Novi Sad***Oblast – SEMANTIČKI VEB**

Kratak sadržaj – Rad istražuje načine na koje se savremena tehnologija računarske inteligencije – ChatGPT, može integrisati sa tehnologijama semantičkog veba. Demonstrira kako se na osnovu podataka iz dve ontologije (ACM Curriculum i Bibo Ontology) mogu kreirati a potom i izvršiti SPARQL upiti na osnovu korisničkog zahteva.

Ključne reči: semantički veb, veštačka inteligencija, ChatGPT

Abstract – This work explores ways of integrating ChatGPT with the technologies of semantic web. It demonstrates how using ACM Curriculum and Bibo Ontologies we can generate and perform SPARQL queries using AI.

Keywords: semantic web, artificial intelligence, ChatGPT

1. UVOD

Doba rapidno rastućih informacionih tehnologija za sobom je ponelo i eksponencijalnu ekspanziju računarske inteligencije. Adaptacija novonastalih modela postepeno pronalazi svoj standard u industriji, kako bi se posredstvom računarske inteligencije olakšali brojni izazovi razvoja aplikacija.

Semantički web, poznat i kao Web 3.0 predstavlja ekstenziju World Wide Web-a kroz seriju standarda kako bi podaci bili mašinski čitljivi nezavisno od njihovog izvora. Fokus se svodi sa veb aplikacija na prezistirane podatke. Njihov oblik i korelacije detaljno su opisani putem **ontologija**. Podaci se čuvaju u **RDF** (Resource Description Framework) formatu. Ovakav način perzistencije podataka dozvoljava nam da koristimo upitni jezik specifično kreiran za RDF – **SPARQL** (SPARQL Protocol and RDF Query Language).

U ovom projektu, integracija sa tehnologijama računarske inteligencije svodi se na korišćenje ChatGPT-a, kako bi se na osnovu korisničkog zahteva pisanog u prirodnom jeziku za našu ontologiju izgenerisao validni SPARQL upit, čijim bi izvršavanjem bili vraćeni adekvatni podaci.

2. ŠTA JE CHAT-GPT

ChatGPT [1] je alat zasnovan na veštačkoj inteligenciji koji koristi procesiranje prirodnog jezika (NLP – *Natural language processing*) kako bi uz čovekovu interakciju kreirao realističnu i prirodnu konverzaciju. ChatGPT je forma generativnog AI – koji koristi ljudske unose za generisanje korespondentnih slika, video zapisa, ili u ovom slučaju, teksta.

GPT predstavlja *Generative Pre-trained Transformer* (generativni pretrenirani transformer). Model je obučen sa nadgledanjem kroz povratnu informaciju od strane čoveka koja bi nagrađivala tačnije i prirodnije odgovore. Korišćeno je duboko učenje kako bi odgovori rezultovali humanim tekstom kroz neuronske mreže transformera. Oni su specijalizovani za predikciju sekvence (generišu narednu reč, rečenicu, paragraf...).

Izuzetan alat kreiran je od strane kompanije OpenAI, u novembru 2022. godine. Samo nekoliko nedjelja po svom objavljivanju postao je nezamenjiva asistencija u raznim poslovnim branšama.

3. SPECIFIKACIJA ZAHTEVA

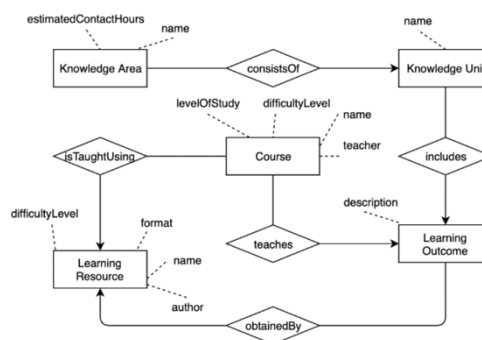
Specifikacija prototipske aplikacije koju ovaj rad istražuje za cilj ima da pokaže jednostavan primer integracije dve tehnologije – **semantički web i ChatGPT**.

Domen podataka kojima se aplikacija bavi jesu informacije o studijskim programima, kursovima, oblastima znanja, naučnim materijalima sa svojim bibliografskim podacima...

Podaci su predstavljeni u **RDF** formatu, opisani pomoću dve ontologije:

- *ACM Curriculum Ontology*
- *Bibo Ontology (The Bibliographic Ontology)* [2]

Primarni fokus jeste ACM Curriculum ontologija, čija je šema prikazana na slici 1.



Slika 1. Šema ACM Curriculum ontologije

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Milan Segedinac, vanr. prof.

Pored demonstracije ručno napisanih upita, glavni zahtev aplikacije jeste integracija našeg sistema sa ChatGPT modelom. Na korisnikov zahtev pisan prirodnim jezikom model treba da izgeneriše adekvatan SPARQL upit koji se izvršava na serveru. Rezultat istog se potom vraća korisniku. Pre generisanja zahteva, ChatGPT model potrebno je obučiti znanjem o ontologiji, relacijama između entiteta kojima se barata, kao i snabdeti adekvatnim primerima SPARQL upita.

4. ARHITEKTURA SISTEMA

Arhitektura sistema sačini se od 3 ključne komponente:

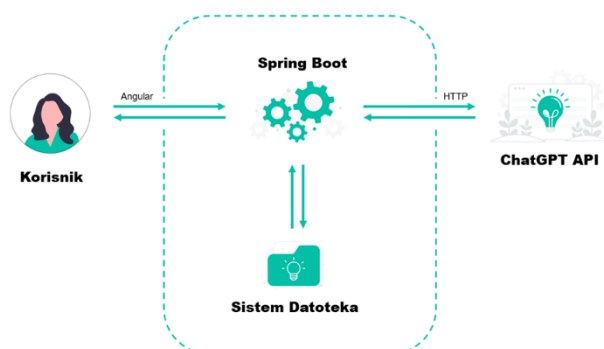
- Klijentska aplikacija
- Serverska aplikacija
- ChatGPT API

Kao razvojno okruženje za klijentsku aplikaciju korišćen je Angular 2. **Angular** je razvijen od strane kompanije Gugl, u septembru 2016. godine, i od tada je doživio rapidan razvoj šestomesečnim nadogradnjama i novim verzijama. Arhitektonski obrazac koji Angular diriguje jeste MCV – Model-View-Controller.

Na serverskoj strani je korišćen **Spring Boot** – besplatno okruženje za razvoj Java aplikacija. Za cilj ima da pojednostavi rad programera uvodeći razne konvencije, automatske konfiguracije i integracije sa brojnim tehnologijama.

Kao eksterni servis korišćen je **API ChatGPT-a** za generisanje SPARQL upita. Visoki nivo određivanja ponašanja API-ja pomaže programerima da odgovore bolje prilagode svojim potrebama. Pored odabira konkretnog modela koji žele da koriste, programeri mogu da specificiraju domen u kome će model davati odgovore. Moguće je navesti stil generisanog teksta, kao i dati potrebne kontekstualne informacije pre konkretnog korisnikovog upita.

Arhitektura sistema definiše korišćenje sistema datoteka za perzistenciju podataka. Kako se koriste tehnologije semantičkog veba, u .owl fajlovima definisane su arhitekture ontologija, dok se sami podaci čuvaju u .rdf fajlovima. Nacrtna arhitektura sistema prikazana je na slici 2.



Slika 2. Nacrtna arhitektura

5. PROTOTIPSKA IMPLEMENTACIJA

Prototipska implementacija aplikacije zasnovana je na demonstraciji osnovnih funkcionalnosti opisanih tehnologijama semantičkog veba.

Komunikacija između klijentske Angular i serverske Spring Boot aplikacije rađena je pomoću HTTP poziva. Takođe, putem istih poziva server komunicira sa ChatGPT API-em.

5.1 Predefinisani upiti

Prvi način na koji korisnik može da interaguje sa aplikacijom jeste pozivom predefinisanih SPARQL upita. Definisani od strane programera, ovi upiti primaju ulazne parametre pomoću kojih se definiše pitanje koje korisnik želi da postavi. Definisana pitanja su:

1. Koji **resursi** su potrebni za učenje specifičnog **kursa**?
2. Koji **resursi** su potrebni za učenje specifične **jedinice znanja**?
3. Za koje **kurseve** su potrebni **materijali** od određenog **autora**?
4. Koji **kursevi** daju znanje za specifičan **ishod učenja**? Koji **autori** su kreatori za **materijale** tih **kurseva**?
5. Koji **dokumenti** su citirani od strane određenog **autora** u specificiranim **formatima materijala**?

5.2 ChatGPT integracija

Kako bi OpenAI omogućio integraciju sa svojim alatima, potrebno je aktivirati API ključ na njihovom sajtu. Kao pomoć programerima u korišćenju ovog servisa dostupna je i zvanična dokumentacija [3].

Pre slanja samog zahteva, neophodno je definisati telo istog. Osnovne informacije koje su definisane u opisanom slučaju su:

- Maksimalan broj tokena odgovora: **200**
- Temperatura (nasumičnost modela): **0.7**
- Model: **text-davinci-003**
- Ulazni tekst

Najbitniji od navedenih parametara je ulazni tekst. Nivo njegove detaljnosti pomoći će modelu da razume širu sliku konteksta u kome korisnik postavlja pitanje. Početak ovog ulaza jeste detaljna definicija ontologije. Potom, na osnovu prethodno definisanih SPARQL upita iste prosleđujemo modelu kao primere. Na samom kraju, dodajemo korisnikov tekst na osnovu koga generisanje počinje.

Opisani deo ontologije ulaznog teksta je sledeći:

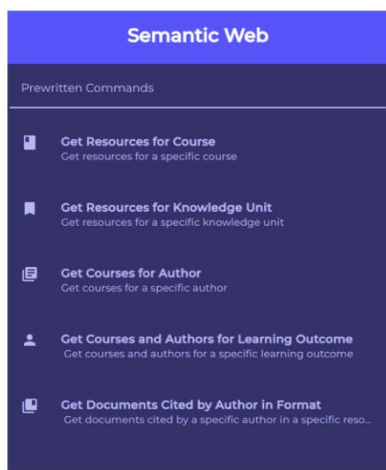
„I have a Semantic web project and it uses a custom ACM Curriculum Ontology. Its entities are: KnowledgeArea (described with estimatedContactHours and name). KnowledgeArea consists of KnowledgeUnit which has a property name. KnowledgeUnit includes LearningOutcome, which has a property description. LearningOutcome is obtainedBy a LearningResource which has difficultyLevel, format, name and author. A Course isTaughtUsing LearningResource. A course has leveOfStudy, difficultyLevel, name and teacher. Course teaches one or more LearningOutcome. The namespace prefix is acm:, and the namespace url is: http://www.semanticweb.org/sasaboros/ontologies/2020/11/se_c_ontology#. If asked to connect the Course with the learning Outcome it teaches, please use the 'teaches' connector between them. I will give you

some example queries for this Ontology. (Query examples...) I will provide you with a question, and your job is to provide me with the SPARQL query for getting said data in described ontology. Do not include any properties the user does not mention himself. The question is: ...“

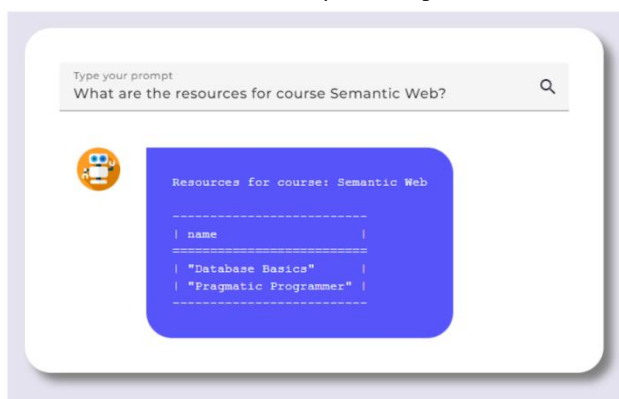
Nakon definisanja ulaznog teksta, preostalo je samo da pozovemo model putem API-a. Vraćen rezultat je u obliku String-a, i kao takav se prosjeđuje servisnoj kasi *SparqlService*. Nakon jednostavnih obrada objekta upita generisanog od strane modela, izvršava se poziv istog nad ontologijom. Rezultati se korisniku vraćaju na isti način kao i kod predefinisanih SPARQL upita.

5.3 Demonstracija klijentske aplikacije

Korisničko iskustvo definisano je lakoćom korišćenja klijentskog interfejsa. Na slikama 3 i 4 prikazan je izgled istog. Sa leve strane dostupni su predefinisani upiti, dok se u desnom odeljku ekrana nalazi konverzacioni prostor za postavljanje proizvoljnih upita.



Slika 3. Predefinisani upiti



Slika 4. Konverzacioni prikaz

6. VERIFIKACIJA I VALIDACIJA

Rezultati ChatGPT modela većinski su se pokazali tačnim. Zbog sohasitičke prirode istog, unošenje identičnog ulaznog teksta više puta daće različite rezultate. U testiranim primerima, rezultati ovih upita nisu se vidno razlikovali, iako su sami upiti bili drugačiji.

Model je pokazao dobru mogućnost razumevanja ontologije, i generisanja takvih upita čijim će se izvršavanjem izvući stvarni rezultati. Pored toga, model

se uspešno snalazi i sa jednostavnim matematičkim apstrakcijama SPARQL upita, poput direktive za prebrojavanje - COUNT.

U početnim iteracijama rešenja, upiti koji su iziskivali upravo COUNT direktivu često su bili netačni, kako bi generisan kod imao sintaksne greške. Ovaj problem rešen je dodavanjem primera pravilno napisanih upita u ulazni tekst modela.

Prilikom testiranja modela primećeno je da je veoma bitan način na koji se upit formira. Kako bi generisani upiti bili tačniji, potrebno je koristiti što više pojmova iz same ontologije (nazive klasa i poveznika).

Na primer, ukoliko bismo pitanje „How many knowledge units does each of the knowledge areas consist of?“ pokušali da svedemo na skraćenu verziju oblika „How many knowledge units does each of the areas have?“, sa pretpostavkom da se *knowledge area* može skratiti na *area*, kao i da se *consist of* podrazumeva sa *have*, rezultati upita ne bi bili tačni.

Jedan od primera na kojima model ima predvidivo, ali nepoželjno ponašanje jeste davanje upita koji su nevezani za ontologiju. Na primer, na pitanje „What are the most popular breeds of cats?“, model će izgenerisati sledeći upit:

```
PREFIX acm:
<http://www.semanticweb.org/sasaboros/ontologie
s/2020/1 1/sec_ontology#> SELECT ?breed WHERE
{?cat acm:breed ?breed . ?cat acm:popularity
?popularity . FILTER (?popularity > 0) ORDER BY
DESC(?popularity) LIMIT 10}
```

Iako je u pitanju sintaksno ispravan SPARQL upit, vidimo da se model prilagodio postavljenom zahtevu iako je znao da pomenute klase ne postoje u definisanoj ontologiji. Sa trenutnom implementacijom, ChatGPT će krenuti sa pretpostavkom da sve navedene klase u ulaznom pitanju postoje, te će na osnovu toga generisati upit čije će pokretanje izazvati grešku na serverskoj aplikaciji.

Ovakvo ponašanje nije neočekivano od strane ChatGPT v3.5. I kompletan model često je (pogotovo na programerskim temama) sklon davanju netačnih odgovora, kako bi zadovoljio nameru korisnika. Pojednostavljeno, model će se potruditi da korisniku da potvrđan odgovor čak i ako on ne postoji. Primer za ovo bilo bi generisanje koda sa metodama koje nisu definisane u specifikaciji, ali čije bi ime sugerisalo rešenje za problem koji korisnik navodi.

I ovako nepovoljno ponašanje modela moglo bi se popraviti dodatnim proverama, treniranjem, ili strožijim pravilima uz manji stepen kreativnosti ChatGPT-a.

7. ZAKLJUČAK

Rapidan razvoj veštačke inteligencije na svim poljima pruža veoma dobar potencijal za implementaciju semantičkog veba. AI ne samo da može da poboljša načine na koje menjamo podatke, već i iz korena promeni tehnike kojima im pristupamo.

Posredstvom savremenih modela, informacijama već danas možemo da pristupamo ne na nivou aplikacija ili

dokumenata, već na nivou znanja koje postoji u širem kontekstu.

Rad istražuje načine na koji se jedan od trenutnih *cutting-edge* generativnih modela – ChatGPT može iskoristiti sa tehnologijama semantičkog veba. Konkretno, istraženo je kako ChatGPT može da generiše SPARQL upite za ontologije definisane pomoću OWL-a, nad podacima čuvanim u RDF formatu.

Rezultati ove integracije su vrlo uspešni, međutim, bitno je naglasiti da kvalitet generisanog upita veoma zavisi od teksta unesenog od samog korisnika. Pravilno korišćenje imena klasa, atributa i veza dovešće do generisanja boljih upita koji se mogu pravilno izvršavati nad odabranom ontologijom.

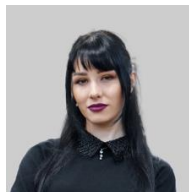
Prostori za poboljšanje ovog prototipskog rešenja primarno se zasnivaju na detaljnijem treniranju modela, i mogućnosti da razume korisničke namere čak i ako prosleđen zahtev semantički odstupa od pojmova definisanih ontologijom. Dodatno, rešenje bi se moglo poboljšati integracijom dodatnih ontologija, i temeljnije uvezanih podataka.

Iako je ovo samo jedan od primera moguće integracije modela veštačke inteligencije sa semantičkim vebom, neosporivo je da će u skoro dolazećem periodu AI igrati ključnu ulogu u načinu na koji pristupamo podacima na internetu. Svedoci smo stupanja u jednu novu, inteligentnu eru za tehnološki svet, čije su mogućnosti neiscrpna tema za istraživanje i napredak.

8. LITERATURA

- [1] Definition: ChatGPT, <https://www.techtarget.com/whatis/definition/ChatGPT> (pristupljeno u maju 2023.)
- [2] Bibo Ontology, <https://github.com/structuredynamics/Bibliographic-Ontology-BIBO> (pristupljeno u julu 2023.)
- [3] ChatGPT documentation, <https://platform.openai.com/docs/api-reference/chat> (pristupljeno u maju 2023.)

Kratka biografija:



Tara Pogančev rođena je 3. februara 2000. u Novom Sadu. Školske 2018/2019. godine se upisuje na Fakultet tehničkih nauka u Novom Sadu, na studijskom programu računarstvo i automatika. Nakon diplomiranja u julu 2022. godine, upisuje master studijski program Inteligentni sistemi na istom fakultetu.

kontakt: tarapogancev@gmail.com

**SISTEM ZA OPTIMIZACIJU RASPOREDA NASTAVE FAKULTETA TEHNIČKIH
NAUKA****TIMETABLE OPTIMIZATION SYSTEM FOR FACULTY OF TECHNICAL SCIENCES**

Eva Janković, Nikola Luburić, *Fakultet tehničkih nauka, Novi Sad*

**Oblast – ELEKTROTEHNIČKO I RAČUNARSKO
INŽENJERSTVO**

Kratak sadržaj – U sklopu ovog rada dizajniran je i implementiran sistem za optimizaciju rasporeda nastave, sa posebnim fokusom na organizacionu strukturu Fakulteta tehničkih nauka. Opisan je domen problema, korisnički zahtevi i arhitektura sistema, dat je pogled na implementaciju sistema, kao i moguća unapređenja sistema. Rezultat rada jeste softver koji pruža mogućnost optimizacije rasporeda nastave koji je dodatno integrisan u širi sistem. Širi sistem omogućava korisniku upravljanje celim tokom, od ulaznih podataka, preko optimizacije i zaključno sa kreiranjem rasporeda u različitim formatima.

Ključne reči: sistem baziran na pravilima, problem planiranja rasporeda, predstava domenskog znanja

Abstract – As part of this paper, a system for optimization of teaching schedules was designed and implemented. The system has a special focus on the organizational structure of the Faculty of Technical Sciences. The problem domain, user requirements and system architecture are described. An overview of the system implementation is given, as well as the possible system improvements. The result of the paper is a teaching schedule optimization software. The software is furthermore integrated into a wider system that allows the user to manage the entire process, from input data, through optimization, and exports of the schedule into different formats.

Keywords: rule-based system, scheduling problem, representation of domain knowledge

1. UVOD

Problem kreiranja rasporeda (engl. scheduling problem) prisutan je u različitim aspektima savremenog društva. Dobro planirani raspored doprinosi efikasnosti i boljem iskorišćenju vremena i resursa. Efikasno upravljanje vremenom utiče na smanjenje stresa i povećanje produktivnosti. Ovo se odnosi na individualne nivoe, ali i na kolektivne strukture kao što su obrazovne institucije [1] i kompanije.

Rad se fokusira na rešavanje problema rasporeda u kontekstu visokoškolskih ustanova, i to konkretno rasporeda nastave Fakulteta tehničkih nauka (FTN).

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio doc. dr Nikola Luburić.

Raspored nastave podrazumeva definisanje nastavnih termina za sve studentske programe Fakulteta na nivou semestra. Studenti su u okviru studentskih programa raspoređeni u studentske grupe koje pohađaju termine nastave kao celina. Na nivou akreditacije studentskog programa, propisani su predmeti koje studenti pohađaju. Predavanja i vežbe propisanih predmeta održavaju predavači u adekvatnim prostorijama. Termin nastave okuplja studentske grupe i predavače i potrebno mu je na optimalan način dodeliti prostoriju i vremenski interval održavanja.

Srž problema trenutnog rasporeda nastave, na čije rešavanje je rad usmeren, jeste izbacivanje radnih subota i ranih i kasnih termina, kao i smanjenje perioda nekorisnog vremena (pauza) u rasporedima studenata i nastavnog osoblja. Motivacija za izučavanje i rešavanje opisanog problema javlja se iz želje za podizanjem kvaliteta obrazovanja i radnog okruženja kroz bolje organizovanje vremena.

Primarni cilj rada jeste razvijanje sistema za automatizovano kreiranje rasporeda na osnovu ulaznih podataka uz oslanjanje na enkapsulirano domensko znanje. Prikupljanje ograničenja koje raspored treba da ispuni urađeno je kroz anketiranje predavača i razgovore sa domenskim ekspertom. Time je domen problema sužen i predstavljen u vidu pravila, činjenica i zaključaka. Ovakav problem spada u grupu problema koji se mogu rešavati upotrebom ekspertskih sistema [2]. Ekspertski sistemi su računarski sistemi koji enkapsuliraju znanje eksperta iz određene oblasti [3].

U narednom poglavlju dat je osvrt na domen problema i stanje u oblasti. Poglavlje 3 opisuje potrebne slučajeve korišćena i arhitekturu sistema. U poglavlju 4 nalazi se opis konkretne implementacije. Na kraju se nalazi poglavlje zaključka koje rezimira celokupan rad.

2. DOMEN PROBLEMA

Zainteresovanost za poddomen problema rasporeda obrazovnih ustanova značajno je porasla 2002. godine kada je kreirano prvo internacionalno takmičenje, International Timetabling Competition (ITC-2002). Takmičenje nastoji da pruža nove javno dostupne skupove podataka. Najskorije takmičenje, ITC2019 [4], uvelo je nov skupa podataka prikupljen u realnim uslovima na visokoškolskim ustanovama.

Generalno, u literaturi i takmičenjima, problem rasporeda visokoškolskih ustanova podrazumeva dodelu prostorija i fiksnih vremenskih intervala terminima prateći određeni

skup ograničenja [5]. Rad [6] daje opsežan pregled oblasti problema rasporeda u obrazovanju. Ovakvi problemi spadaju u grupu NP-teških problemi (nedeterministički u polinomijalnom vremenu teški) [7][8]. Skup ograničenja kojim su definisani ovi problemi se najčešće deli u dve podgrupe, ograničenja koja se moraju ispoštovati da bi se rešenje smatralo prihvatljivim i ograničenja koja bi trebalo ispoštovati, opciona ograničenja. Broj prekršaja opcionih ograničenja obrnuto je proporcionalan kvalitetu rešenja.

Glavna razlika problema rasporeda opisanog kroz ITC takmičenje i problema rasporeda FTN-a je što vremenski intervali nisu fiksni u slučaju FTN-a. Dodatno, takmičenje posmatra sve prostorije ravnopravno, uzimajući u obzir samo njihov kapacitet; ne postoji pojam o pripadnosti organizacionim jedinicama.

Zbog opisanog, implementacija sistema nije bazirana samo na informacijama iz literature već prvenstveno na znanju domenskog eksperta.

U okviru rada [9], pobednici ITC2019, ističu značaj upotrebe dve faze prilikom rešavanja problema. Prva faza je konstrukciona heuristika (engl. construction heuristic), gde se vodi računa samo o obaveznim ograničenjima. Druga faza je faza lokalne pretrage (engl. local search) koja teži da pronađe što bolje rešenje. Zbog istaknutog, u okviru rada je odlučeno da se optimizacija sastoji iz dve faze.

U literaturi se javljaju različiti organizacioni algoritmi kao uspešni. Autori rada [9] oslanjaju se na upotrebu algoritma baziranog na grafovima. U okviru radova [10][11][12] autori ističu tabu pretragu, baziranu na metaheuristikama, kao izuzetno korisnu, dok autori radova [13][14] baziraju optimizaciju na genetskom algoritmu. Testiranjem različitih algoritama na realnim podacima i praćenjem dokumentovanih smernica [15], odabran je algoritam baziran na metaheuristikama.

3. SPECIFIKACIJA SISTEMA

U okviru poglavlja detaljnije su prikazani konkretni korisnički zahtevi, arhitektura sistema, kao i globalna arhitektura sistema u okviru koga je predloženi sistem integrisan.

3.1. Korisnički zahtevi

Kao uvodna faza izrade sistema, posmatranjem organizacione strukture Fakulteta, izdvojene su najznačajnije osobe za prikupljanje zahteva. Inicijalno su izdvojene dve grupe: studenti i nastavno osoblje. Zahtevi studenata su bazirani na razgovoru i prethodnim iskustvima. Dodatno, pregledana je i arhiva dotadašnjih žalbi na prethodne rasporede.

Prikupljanje zahteva za unapređenje rasporeda nastavnog osoblja urađeno je kroz anketiranje. Anketirani su zaposleni Katedre za informatiku koji predstavljaju podskup svih zaposlenih Fakulteta. Zbog velike brojnosti katedre i opštosti pitanja, zaključci su generalizovani na nivo Fakulteta. Anketa se sastoji iz sedam pretpostavki vezanih za optimalni raspored gde se od anketirane osobe očekuje da na skali od 1 do 5 navede nivo slaganja sa svakom pretpostavkom. U tabeli 3.1. nalaze se sumirani rezultati.

Tabela 3.1. Opis zahteva i srednja vrednost njihove značajnosti

Opis pretpostavljenog zahteva	Srednja vrednost značajnosti
Održavanje predavanja i vežbi samo radnim danima	4,95
Održavanje predavanja i vežbi na OAS i OSS između 9:00 i 17:00	4,17
Održavanje predavanja i vežbi na OAS i OSS između 7:00 i 15:00	2,43
Održavanje predavanja i vežbi na MAS između 15:00 i 19:00	2,64
Održavanje svih vežbi u učionicama katedre	4,36

Kako bi se kreirao optimalan raspored nastave, nije dovoljno uvažiti samo zahteve nastavnog osoblja. Pošto je FTN visokoškolska ustanova na teritoriji Republike Srbije, potrebno je ispoštovati i zakonska ograničenja, kao i sve dodatne propise koje sam Fakultet postavlja. Sledeći korak istraživanja podrazumevao je pronalaženje svih značajnih dostupnih dokumenata. Kao najznačajniji dokument istakao se „Pravilnik o izradi rasporeda časova nastave i rasporeda ispita na Fakultetu tehničkih nauka“ koji je javno dostupan na internet stranici FTN-a [16]. Pravilnik navodi značajne zahteve za kreiranje rasporeda nastave koji su navedeni u nastavku:

- student može imati najviše 8 sati nastave u jednom danu,
- pauze između termina nastave su 15 minuta, osim ako drugačije nije odobreno,
- članovi dekanata i organizacije Fakulteta sredom od 8 do 14 časova ne smeju imati nastavu,
- u terminima nastave sa više izvođača, raspored nastave se formira na osnovu jednog predavača.

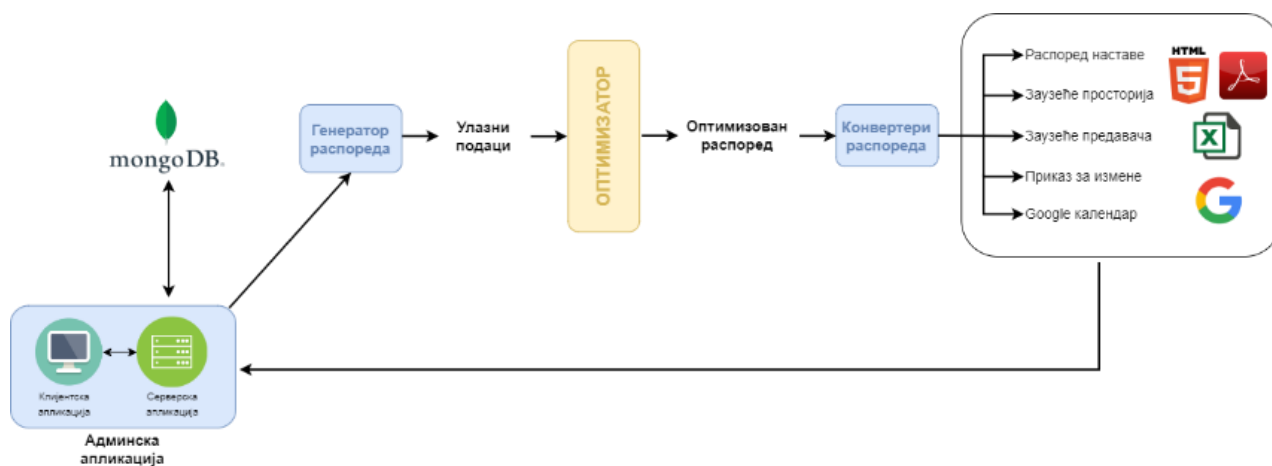
3.2. Slučajevi korišćenja

Proučavanjem trenutnog informacionog sistema FTN-a, kao i organizacione strukture, zaključeno je da bi sistem optimizacije imao jednog tipa korisnika, administratora. Administrator bi bio zaposleni zadužen za formiranje rasporeda nastave, ili više zaposlenih u daljoj perspektivi. Izdvojeni su sledeći slučajevi korišćenja:

- dopremanje podataka značajnih za optimizaciju,
- pokretanje optimizacije,
- prekidanje optimizacije,
- iterativna optimizacija,
- simulaciona optimizacija,
- preuzimanje optimizovanog rasporeda,
- preuzimanje rezultata optimizacije i
- preuzimanje podataka o ispunjenosti ograničenja.

3.3. Arhitektura sistema

Arhitektura sistema prikazana je na slici 3.1. Sistem je kreiran kao serverska veb aplikacija upotrebom SpringBoot radnog okvira [17].



Slika 3.1. Arhitektura integracionog sistema

Srž aplikacije bazira se na OptaPlanner biblioteci [18] koja pruža mogućnost rešavanja problema sa definisanim ograničenjima (engl. constraint satisfaction problems) među koje spada problem rasporeda nastave.

3.4. Arhitektura integracionog sistema

U cilju lakše upotrebe i veće održivosti sistema, dizajniran je integracioni sistem koji nudi konzistentan tok kreiranja rasporeda nastave od ulaznih podataka, preko optimizacije do izlaznih podataka, to jest rasporeda.

4. Implementacija sistema

Izvorni kod celokupne aplikacije je javno dostupan.¹ U daljem tekstu opisan je pristup rešavanju problema rasporeda i sva pravila u sistem, kao i konfiguracija OptaPlanner biblioteke i upotrebljeni optimizacioni algoritam.

4.1. Rešavanje problema rasporeda

Problem rasporeda spada u probleme sa ograničenjima. Težina i vreme optimizacije varira od ulaznih podataka. U slučaju optimizacije rasporeda nastave FTN-a, za potrebni vremenski interval (72 sata), testiranjem je zaključeno da optimalno rešenje nije dostižno. Rešenje nije dostižno kada krši neka od obaveznih pravila, to jest postignuto rešenje nije prihvatljivo. Kako bi sistem proizveo prihvatljivo rešenje za zadati vremenski interval, promenjen je pogled na optimizacioni problem. Problem je modelovan kao problem sa nedosežnim skupom ograničenja (engl. overconstrained problem).

Promena pogleda na problem omogućila je postizanje prihvatljivog rešenja u zadatom vremenskom roku. Optimizacioni proces je promenjen tako da omogući da se nekim terminima ne dodeli vremenski interval održavanja, čime se oni ne uvrštavaju u finalni raspored nastave. Naknadno, administrator ima mogućnost ručnog dodavanja vremenskog intervala takvim terminima.

4.2. Ograničenja u okviru problema rasporeda

Sva ograničenja prikupljena su u prvoj fazi istraživanja. Obuhvataju osnovna ograničenja koja su nametnuta fizičkim zakonima, gde predavač ili studentska grupe ne mogu biti na dva mesta istovremeno, ali i složenija ograničenja koja su indirektna posledica organizacione

strukture FTN-a, kao pripadnost prostorija određenim katedrama. Prilikom izrade sistema, broj nivoa koji se pokazao optimalan bio je tri: ograničenja koja je neophodno ispoštovati, ograničenja koja je poželjno ispoštovati i preostala ograničenja. Ograničenja koja je poželjno ispoštovati dodatno delimo u još dva nivoa. Na ovaj način određena pravila istog nivoa imaju prednost nad drugima. Na slici 4.1. izlistana su sva ograničenja.

ОБАВЕЗНА ОГРАНИЧЕЊА	конфликт просторије
	конфликт предавача
	конфликт студентске групе
	конфликт члана организације факултета
	конфликт члана деканата
	максималан број часова за студента
	обавезан тип просторије
	обавезан капацитет просторије
ПОЖЕЉНА ОГРАНИЧЕЊА - НИВО 1	веза предавача и просторије
	избегавање преклопљеног рада
	лимитирање времена почетка
	лимитирање времена краја
ПОЖЕЉНА ОГРАНИЧЕЊА - НИВО 2	употреба опште просторије
	одржавање предавања истог предмета у истом дану
	лимитирање пауза за студента
	лимитирање пауза за предавача
ПРЕОСТАЛА ОГРАНИЧЕЊА	максималан број часова за предавача
	лимитирање радних дана за предавача
	непромењивост просторије за студента
	непромењивост просторије за предавача

Slika 4.1. Ograničenja grupisana po nivoima

4.3. Konfiguracija OptaPlanner biblioteke

OptaPlanner biblioteka nudi moćnu podršku za rešavanje optimizacionih problema, i to posebno problema sa ograničenjima. Pruža bogat interfejs i poseban način definisanja pravila koji se oslanja na kombinaciju funkcionalnog programiranja i SQL jezika. Rešavač (engl. solver) predstavlja srž biblioteke i omogućava proces optimizacije. Biblioteka je konfigurisana kroz konfiguracioni fajl. Istaknuta su sledeća dva parametra:

- podešavanje konstrukcione heuristike – odabran je *First fit decreasing* algoritam [19]. Prilikom konstrukcije rešenja, algoritam kreće od „kompleksnijih“ objekata (veći broj studenata, duži vremenski interval).
- podešavanje uslova završetka optimizacije – optimizacija će biti prekinuta ukoliko nije pronađeno bolje rešenje u prethodna 3 časa ili ukoliko je ukupno prošlo 72 časa od početka optimizacije.

¹ <https://github.com/evaj10/raspored-optimizator/releases/tag/v2.1.1>

- podešavanje algoritma optimizacije – odabran je *Late acceptance* [20]. Algoritam pamti najbolje rešenje koje je staro par koraka (ne mora biti trenutno najbolje). Proverava mali broj kombinacija pri svakom koraku, te pravi brze korake. Ovo je posebno značajno za probleme velikih dimenzija, kakav je problem rasporeda FTM-a.

5. ZAKLJUČAK

U radu je prikazan sistem za optimizaciju rasporeda nastave, sa posebnim fokusom na organizacionu strukturu Fakulteta tehničkih nauka. Glavni povod razvoja sistema je automatizacija procesa kreiranja rasporeda nastave u cilju podizanja kvaliteta obrazovanja i radnog okruženja. Ideja iza sistema je okupljanje svih ograničenja i faktora za kreiranje optimalnog rasporeda nastave i njegovo kreiranje. Kroz programski kod i prateću dokumentaciju, koja podrazumeva i ovaj rad, enkapsulirano je domensko znanje.

Glavni vid unapređenja sistema ogleda se u daljem istraživanju pogodnih algoritama optimizacije u cilju smanjenja vremena izvršavanja i povećanja efikasnosti sistema. Odabir algoritma je moguće uraditi oslanjajući se na benčmark (engl. benchmark) testove podržane od strane OptaPlanner biblioteke. Jedna od značajnih tehnika jeste „zavadi, pa vladaj“ (engl. divide and conquer), gde se raspored nastave može zgodno podeliti na računarske i laboratorijske vežbe i auditorne tipove nastave (vežbe i predavanja) koja, u osnovi, predstavljaju dva disjunktna skupa termina.

6. LITERATURA

- [1] Häfner, A., Stock, A., & Oberst, V. (2015). Decreasing students' stress through time management training: An intervention study. *European journal of psychology of education*, 30, 81-94.
- [2] Jacob, R. J. K., & Froscher, J. N. (1990). A software engineering methodology for rule-based systems. *IEEE Transactions on Knowledge and Data Engineering*, 2(2), 173-189.
- [3] Jackson, P. (1986). *Introduction to expert systems*.
- [4] Такмичење у оптимизацији распореда наставе за 2019. годину, <https://www.itc2019.org/home> [датум приступа август 2023]
- [5] Wren, A. (1995, August). Scheduling, timetabling and rostering—a special relationship?. In *International conference on the practice and theory of automated timetabling* (pp. 46-75). Berlin, Heidelberg: Springer Berlin Heidelberg.
- [6] Kristiansen, S., & Stidsen, T. R. (2013). A comprehensive study of educational timetabling—A survey. Department of Management Engineering, Technical University of Denmark. DTU Management Engineering Report, 8.
- [7] Al-Betar, M. A., Khader, A. T., & Liao, I. Y. (2010). A harmony search with multi-pitch adjusting rate for the university course timetabling. *Recent advances in Harmony search algorithm*, 147-161.
- [8] Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability. A Guide to the*
- [9] Holm, D. S., Mikkelsen, R. Ø., Sørensen, M., & Stidsen, T. J. (2022). A graph-based MIP formulation of the International Timetabling Competition 2019. *Journal of Scheduling*, 25(4), 405-428.
- [10] Hertz, A. (1991). Tabu search for large scale timetabling problems. *European journal of operational research*, 54(1), 39-47.
- [11] Alvarez-Valdes, R., Crespo, E., & Tamarit, J. M. (2002). Design and implementation of a course scheduling system using tabu search. *European Journal of Operational Research*, 137(3), 512-523.
- [12] Minh, K. N. T. T., Thanh, N. D. T., Trang, K. T., & Hue, N. T. T. (2010). Using tabu search for solving a high school timetabling problem.
- [13] Weare, R., Burke, E., & Elliman, D. (1995). A hybrid genetic algorithm for highly constrained timetabling problems. Department of Computer Science.
- [14] Burke, E. K., & Petrovic, S. (2002). Recent research directions in automated timetabling. *European journal of operational research*, 140(2), 266-280.
- [15] Актуелна документација OptaPlanner библиотеке, <https://www.optaplanner.org/docs/optaplanner/latest/optimization-algorithms/optimization-algorithms.html#whichOptimizationAlgorithmsShouldIUse> [датум приступа август 2023]
- [16] Правилник о изради распореда наставе, ФТН, <http://www.ftn.uns.ac.rs/520569743/pravilnik-o-izradi-rasporeda-casova-nastave-i-rasporeda-ispita-na-fakultetu-tehnickih-nauka-od-31-01-2019-> [датум приступа август 2023]
- [17] SpringBoot радни оквир, <https://spring.io/projects/spring-boot> [датум приступа август 2023]
- [18] OptaPlanner библиотека, <https://www.optaplanner.org/docs/optaplanner/latest> [датум приступа август 2023]
- [19] First fit decreasing алгоритам, <https://www.optaplanner.org/docs/optaplanner/latest/construction-heuristics/construction-heuristics.html#firstFitDecreasing> [датум приступа август 2023]
- [20] Late acceptance алгоритам, <https://www.optaplanner.org/docs/optaplanner/latest/local-search/local-search.html#lateAcceptance> [датум приступа август 2023]

Kratka biografija:



Eva Janković rođena je 10. oktobra 1998. godine u Šapcu. Nakon završetka Šabačke gimnazije, 2017. godine upisuje Fakultet tehničkih nauka u Novom Sadu, smer Softversko inženjerstvo i informacione tehnologije. Studije završava u roku, 2021. godine i iste godine upisuje master akademske studije istog smera.

METODOLOGIJA UZORKOVANJA ISKUSTVA (ESM) KORISNIKA U MOBILNOM RAČUNARSTVU**EXPERIENCE SAMPLING METHODOLOGY (ESM) FOR ELICITING USER REQUIREMENTS IN MOBILE APPROXIMATE COMPUTING**Luna Živković, Milan Vidaković, *Fakultet tehničkih nauka, Novi Sad***Oblast – PRIMENJENE RAČUNARSKE NAUKE I INFORMATIKA**

Kratak sadržaj: Eksponencijalne promene nastale usled sve intenzivnije upotrebe informacionih tehnologija, predstavljaju zabrinjavajući faktor zagađenja životne sredine. Razvoj hardvera mobilnih telefona sve teže prati svakodnevno korišćenje istih od strane milijardi korisnika, kao i sve veću kompleksnost aplikacija koje zahtevaju znatnu upotrebu energetske resurse. Takođe, ograničenja baterije izričito ugrožavaju dalju evoluciju mobilnih telefona. Jedan od novih načina redukovanja potrošnje energije koju telefoni zahtevaju, jeste "približno računanje" (eng. approximate computing) koje žrtvuje preciznost zarad očuvanja energije. Vodeći se ovim, razvili smo context-aware okvir koji se fokusira na ispunjavanje korisničkih očekivanja, dok koristi minimalnu moguću energiju.

Ključne reči: Android, Java, Aware, Weka Yandex, mobile

Abstract: The exponential change in the way Information and Communication Technology is consumed has been so significant that there is an increasing awareness of the potential environmental effects. Underlying mobile hardware does not keep pace with the increased usage of mobile phones in everyday life, as well as the complexity of new apps which demand great energy resources. Limitations in battery technology are especially threatening further mobile computing evolution. A novel approach for reducing the energy appetite of mobile apps comes from the approximate computing field, which proposes techniques that, in a controlled manner sacrifice computation accuracy for higher energy savings. Following this train of thought, we built a context-aware framework that focuses on fulfilling users expectation while using the lowest amount of energy possible.

Keywords: Android, Java, Aware, Weka Yandex, mobile

1. UVOD

Zagađenje životne sredine predstavlja jednu od najaktuelnijih tema današnjice, dok eksponencijalne promene nastale usled sve intenzivnije upotrebe informacionih tehnologija, predstavljaju jedan od glavnih uzroka zagađenja.

Imajući u vidu da danas preko 3 milijarde ljudi svakodnevno koristi pametne telefone, način na koji komuni-

ciramo, posluujemo, putujemo i upoznajemo ljude se znatno transformisao. Iako upotreba mobilnih telefona donosi mnoge koristi, „karbonski otisak“, koji je emitovan u gotovo svakom stadijumu proizvodnje i upotrebe istih, predstavlja značajnu pretnju za životnu sredinu. Na osnovu studija sprovedenih na temu emisije ovih sistema, pretnja se neće vremenom smanjiti sama od sebe, već je potrebno preduzeti brze i znatne mere kako u industrijskom, tako i u političkom sektoru.

Sve veća upotreba i kompleksnost aplikacija zahteva znatne energetske resurse, što razvoj hardvera mobilnog telefona sve teže prati. [1]. Da bi poboljšali dužinu trajanja baterije i smanjili uticaj ICT-a (drugim rečima, kako bi smanjili potrošnju energije), prvo treba da razumemo korisnika i njegove potrebe. U različitim situacijama, korisnici su zadovoljni različitim rezultatima i stoga se treba fokusirati na minimum koji možemo da predstavimo korisnicima, a da oni budu zadovoljni. Imajući u vidu da će svaku aplikaciju, svaki proizvod koristiti različiti ljudi, sa različitim profilima, obrazovanjem i shvatanjima, programeri moraju ostati što objektivniji pri izradi aplikacije.

Ovde uvodimo pojam „približnog računanja“ (eng. approximate computing), kao nov načina redukovanja potrošnje energije koju telefoni zahtevaju. Ono se fokusira na smanjenje preciznosti zarad očuvanja energije. Vodeći se ovim, razvili smo okvir koji ispunjava korisnička očekivanja, dok koristi minimalnu moguću energiju.

Zadatak rada predstavlja kreiranje *Android (Java)* okvira koji će omogućiti praćenje zadovoljstva korisnika tokom upotrebe određene aplikacije i dalju primenu dobijenih rezultata za usavršavanje aplikacije. Krajnji cilj je ispunjenje korisničkih očekivanja uz maksimalno smanjenje potrošnje energije. Kao primer primene okvira, korišćena je aplikacija *NewPipe* za pregled videa, inspirisana postojećim rešenjem (*YouTube*). Uključivanjem minimalnog seta senzora (senzori za praćenje baterije, lokacije, korisničke aktivnosti i osvetljenja ekrana), pratimo kontekst u kom se korisnik nalazi. Dobijene rezultate upoređujemo sa prethodno prikupljenim podacima, gde je korisnik davao ocene na kraju gledanja različitih videa u različitim kontekstima. Kao rezultat povezivanja ovih podataka, aplikacija prikazuje video najniže prihvatljive rezolucije. Na ovaj način produžava se život baterije, to jest, smanjuje se potrošnja energije, a zadovoljstvo korisnika ostaje na visokom nivou. Uvažavajući individualizam u preferencijama gledanja, razvijamo metod za iterativnu personalizaciju prikazanog sadržaja. Imajući u vidu da kontekstno svesne aplikacije zahtevaju kontinuirano

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji sje mentor bio dr Milan Vidaković, red. prof.

prikupljanje i tumačenje podataka iz jednog ili više očitavanja senzora, naš metod se zasniva na kombinaciji *active* i *reinforcement* učenja. Pomoću ova dva načina učenja, smanjuje se količina upita (na kraju gledanja videa gde korisnik daje ocenu kao znak zadovoljstva rezolucijom u datom kontekstu) koja je potrebna da bi se obradile preferencije pojedinačnih korisnika, što opet doprinosi uštedi energije, kao i poboljšanju korisničkog iskustva. Ono što je najvažnije, ovo istraživanje ima za cilj da stvori i jasan generički okvir koji će voditi razvoj budućih sličnih aplikacija.

2. PREGLED AKTUELNOG STANJA U OBLASTI

U ovom poglavlju predstavljene su postojeće literature, čije teme su slične temi koja je opisana u ovom radu. Spomenute literature su razmatrane u cilju boljeg razumevanja *experience sampling*-a i *approximate computing*-a. Nakon dužeg istraživanja velikog broja javno dostupnih literatura, detaljnije je istražena nekolicina istih, gde svaki od radova na svoj način ima značajan uticaj u razumevanju i dubljem razmišljanju o načinu rešavanja problema na koje se susretalo pri izradi.

2.1. ESM

Experience sampling metoda postoji već duže vreme i odnosi se na procenu psihičkog stanja korisnika u određenim situacijama. Danas, pored automatskog prikupljanja podataka, mobilni telefoni omogućuju interakciju sa korisnikom tokom čitavog dana, odnosno, u prirodnom okruženju korisnika, umesto prisilnog okruženja u kojem se ispitanici često ponašaju drugačije. [2] Međutim, postoje neka ograničenja, kao što je sprovođenje ankete bez ometanja načina života korisnika ili pitanje kako odlučiti koji je trenutak relevantan ako korisnici žive nepredvidivim životom (mnogo se kreću) ili koje kriterijume koristiti pri izboru ispitanika (sada kada gotovo svi koji poseduju mobilne uređaje mogu da učestvuju). Tehničke poteškoće u vezi sa ESM-om primarno predstavlja trajanje baterije. Da bi smanjili potrošnju, programeri treba da su u mogućnosti da definišu koliko često treba da se vrši uzorkovanje podataka. U studiji [3] pokazano je da bi idealan broj učesnika bio 19, studija ne bi trebalo da traje duže od mesec dana i predlaže da se anketa povuče nakon par minuta, jer ako se povuče kasnije, kontekst će biti drugačiji i odgovor neće biti relevantan. Prilikom izrade ankete, fokus treba da bude i na tome kada je prikazati korisniku. S jedne strane, od istraživača se traži da prikupi što bogatije podatke, a sa druge strane, da ne smeta korisniku. Loša strana *esm sensing*-a predstavlja značajnu potrošnju energije zbog stalne potrebe za praćenjem trenutnog konteksta u kojem se korisnik nalazi. Štaviše, samo snimanje konteksta u kojem se korisnik nalazi možda neće biti dovoljno jer u stvarnom životu mnogo više može uticati na mišljenje korisnika.

2.2 Approximate computing

Široka upotreba mobilnog računarstva nam omogućava da proverimo kako se menja zadovoljstvo korisnika dobijenim rezultatima sa promenom konteksta. Brojni aspekti situacije i okruženja mogu uticati na percepciju korisnika o rezultatu. Dakle, smatramo da je „kontekst“

složen termin, čiji pogled dobijamo preko senzora ugrađenih u mobilni uređaj.

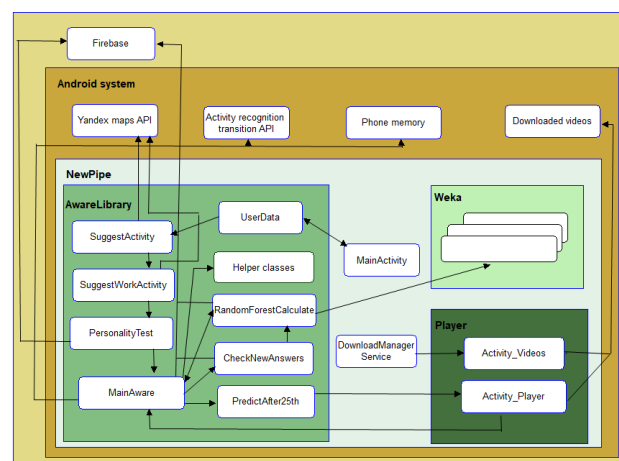
Approximate computing, odnosno približno računanje predstavlja tehnologiju koja žrtvuje tačnost rezultata zarad energetske efikasnosti. Koristeći ovaj metod, u mogućnosti smo da korisniku damo minimum koji će zadovoljiti njegove potrebe u cilju uštede energije. Međutim, do danas nema dovoljno podrške za pisanje i modelovanje *approximate* programa na mobilnim platformama, što usporava ukupan napredak. Eksperimenti su pokazali značajne uštede resursa, na primer, utrošenu energetska efikasnost sa aproksimacijama zasnovanim na neuronskim mrežama [4], ili 2,5 puta veću brzinu kada se određeni obrasci zamene *approximate* kodom. Percepcija korisnika o rezultatu se menja sa okruženjem i situacijom u kojoj se nalazi i stoga kontekst treba predstaviti kao veoma složen pojam. Uz pomoć ESM-a ćemo pitati korisnika koliko je zadovoljan sadržajem koji se prikazuje u određenom kontekstu, dok nam mašinsko učenje pomaže da povežemo korisnika i situaciju u kojoj se nalazi kako bismo modelirali promenu očekivanja kvaliteta rezultata u različitim kontekstima [5].

3. KORIŠĆENE TEHNOLOGIJE

Za implementaciju okvira korišćen je *Android Studio* [6], programski jezik *Java* [7]. Važne komponente projekta predstavljaju *AWARE* [8] biblioteka, koja je korišćena za praćenje senzora tokom sesije, kao i *WEKA* [9] biblioteka pomoću koje implementiramo *random forest* [10] algoritam. Za sugerisanje i pretraživanje lokacije na kojoj korisnik živi ili radi korišćen je *Yandex API* [11]. Svi značajni podaci čuvaju se u *Firebase* [12] bazi podataka.

4. SPECIFIKACIJA APLIKACIJE

4.1. Android ESM sensing biblioteka



Slika 1 - Arhitektura aplikacije i veze između klasa

U svetlo plavoj boji je aplikacija *NewPipe*, koju smo uzeli kao osnovu. *AwareLibrary* okvir je označen zelenom bojom. Takođe, tamno zelenom bojom označen je *Player* koji koristimo za reprodukciju download-ovanih videa, a svetlo zelenom bojom je naznačena *Weka* biblioteka koju koristimo za implementaciju *random forest* algoritma. *AwareLibrary* okvir je zasnovan na aplikaciji otvorenog koda *NewPipe* [13]. Sve funkcionalnosti koje imple-

mentira *NewPipe* su i dalje dostupne korisniku. Jedina promena koju korisnik može da primeti odnosi se na deo sa preuzetim videima. S obzirom da *NewPipe* aplikacija ne pruža mogućnost preuzimanja videa različitih rezolucija, ovaj deo je morao biti odrađen preko modula *Player*. Kada korisnik otvori stranicu sa preuzetim videima, redirektovan je na aplikaciju koja prikazuje sve videoe koji se nalaze u uređaju.

Dalje, iz ove aplikacije pozivamo praćenje senzora, čuvanje i obradu podataka koji su važni za ovaj rad, a predstavljaju deo *AwareLibrary* okvira.

Da bismo razumeli vezu između prikazivanja aplikacije na određen način i konteksta u kojima je prikazana, napravili smo okvir koji povezuje navedeno. Koristeći prikupljanje podataka ugrađenih u naš okvir, izračunali smo zadovoljstvo korisnika određenim video zapisima koji su pušteni, a kasnije, uz korišćenje *active* i *reinforcement learning*-a, napravili smo odgovarajući model koji odgovara preferencijama svake osobe pojedinačno uz najmanju moguću količinu upita.

Kada se aplikacija otvori prvi put, tražimo od korisnika da unese svoju kućnu i adresu na kojoj radi, koja kasnije pomaže da se utvrdi lokacija od značaja. Ovo je obično važno jer se korisnik može osećati drugačije na različitim lokacijama što može uticati na njegova očekivanja. Nakon toga, korisnik treba da odgovori na deset kratkih pitanja u vezi procene ličnosti koja takođe mogu imati značajan psihološki uticaj na preferencije. Ukoliko aplikacija nije otvorena prvi put i u bazi već postoje podaci vezani za isti uređaj, u klasi *AlarmReminder* postavljamo *scheduler* koji svaki dan u ponoć setuje broj odgledanih videa dnevno na 0. U klasi *HelpClassReminder* postavljamo *scheduler* koji u 18h svaki dan proverava da li je korisnik odgledao minimum 5 videa do tad i ako nije prikazuje se notifikacija kako bi podsetili korisnika da pogleda videoe. Nakon toga, prikazuje se početna strana *NewPipe* aplikacije.

Nakon toga, sam programer, koristeći naš okvir, može da pozove jednostavnu funkciju gde označava koje senzore želi da uključi tokom koje aktivnosti i na koje parametre želi da se fokusira i na osnovu toga izgradi model. Funkciju poziva bilo gde u svom programu, gde smatra da je važno započeti prikupljanje podataka o kontekstu. U ovom primeru, uključeni su svi mogući senzori na početku gledanja videa.

Pri pozivu ove funkcije proverava se da li je korisnik odgledao više od 25 videa ukupno. Ako je pregledao, na osnovu prikupljenih podataka računamo koji kvalitet videa bi najviše odgovarao datom kontekstu i iz modula *Player* pozivamo odgovarajući video odgovarajuće rezolucije.

Ukoliko korisnik još nije pregledao 25 videa, na kraju svakog gledanja, poziva se druga funkcija koja zaustavlja prikupljanje podataka i pokreće obradu istih. Nakon zaustavljanja senzora prikupljamo podatke iz *AWARE* biblioteke i izračunavamo za svaki senzor koja je vrednost bila najčešća.

Zatim, koristeći *random forest* algoritam, predviđamo kako će korisnik oceniti ovu sesiju (u našem slučaju gledanje videa), nakon čega biramo jednu od četiri strategije *reinforcement* učenja koje su objašnjene u odeljku 4.3. U zavisnosti od rezultata algoritma, korisniku prikazujemo ili ne prikazujemo anketu.

Pored toga, da bismo motivisali korisnike, nakon određenog vremenskog perioda nagrađujemo ih otkrivanjem podataka o ličnosti, statistika o korišćenju aplikacije itd.

4.2. Razumevanje percepcije približnog mobilnog računarstva (approximate computing)

Approximate computing je računarska paradigma koja omogućava programerima da kompromituju rezultat računanja u cilju uštede energije. *Approximate computing* za mobilne uređaje uveden je sa mobilnim telefonima i razlikuje se u pogledu konteksta korišćenja koji se menja u toku upotrebe i čini predviđanje zadovoljstva još složenijim.

Imajući u vidu da na korisnikovo mišljenje utiču i karakteristike njegove ličnosti, njih lako računamo pomoću već definisanih pitanja za test ličnosti: Big Five Inventori. BFI-10 je izmišljen za kontekst sa ograničenim vremenom, sadrži samo 10 kratkih pitanja koja mogu brzo odrediti kakva je korisnik osoba. Pošto na zadovoljstvo korisnika značajno utiču lokacija na kojoj se korisnik nalazi, trajanje baterije, aktivnost korisnika i doba dana (osvetljenje), uzećemo u obzir ta četiri senzora.

Programer ima mogućnost programskog pokretanja i zaustavljanja senzora jednostavnim pozivanjem funkcije. Pošto se češće dešava da se kontekst (aktivnost, lokacija) promeni tokom jedne sesije gledanja, na kraju nje izračunavamo najčešću lokaciju/aktivnost/prosečno trajanje baterije/prosečno osvetljenje. Zatim, koristeći *random forest* algoritam, izračunavamo verovatnoću da će korisnik biti zadovoljan u određenom kontekstu, gledajući video sa određenim kvalitetom.

4.3. Inteligentno ispitivanje i modelovanje korisničkih očekivanja

Da bismo predvideli zadovoljstvo korisnika određenim parametrima, razvili smo model koji omogućava iterativno učenje preferencija korisnika u nekom kontekstu.

Aktivno učenje predstavlja strategiju u kojoj algoritam mašinskog učenja postiže veću tačnost ako je u mogućnosti da izabere iz kojih podataka će učiti. U našem našem slučaju, ono može smanjiti broj anketa koje se šalju korisniku i povećati zadovoljstvo korisnika jer bi ga prekidao samo kada njegov odgovor može značajno poboljšati kvalitet modela.

Da bismo odredili da li da korisniku pošaljemo anketu ili ne, razvili smo četiri različite strategije aktivnog učenja, kao i okvir koji pomaže u određivanju kako se približiti strategiji koja najbolje odgovara svakom korisniku. Počevši od *random forest* algoritma, izračunavamo pouzdanost da će korisnik dati video oceniti određenom ocenom (1-5). Posle svakog izračunavanja primenjujemo jednu od strategija aktivnog učenja:

- Margin of confidence*: odlučuje da pita korisnika ako razlika između prva dva najpouzdanija predviđanja padne ispod 20%.

- Random sampling*: nasumično bira da li da pita korisnika ili ne.

- Random sampling* zasnovan na vrednosti fizičke aktivnosti korisnika: pošto aktivnost korisnika predstavlja ključni parametar, u ovoj strategiji pitamo korisnika da li je u vozilu ili hoda. Imajući u vidu da je najčešće stanje korisnika mirovanje, prioritet dajemo situacijama u kojima korisnik nije miran.

•*Least confidence sampling*: traži od korisnika da oceni kvalitet videa ako najpouzdanije predviđanje padne ispod 0,5.

Da bismo balansirali između ova četiri, a ne samo fokusirali se na onu koja izgleda kao strategija koja najbolje funkcioniše, koristimo reinforcement algoritam učenja koji se naziva *upper confidence bound* (UCB). Osnovna jednačina UCB-a je sledeća:

$$A_t = \operatorname{argmax} (Q_t(a) + c \ln(t) N_t(a))$$

Za svaku od *active learning* strategija izračunavamo $A(t)$. Prvi deo jednačine predstavlja eksploatacioni deo. $N_t(a)$ je trenutna procena nagrade za akciju a predstavlja prosečnu vrednost prethodnih nagrada. Ako uzmemo u obzir samo ovaj deo jednačine, on će uvek birati akciju sa najvećom nagradom. Drugi deo jednačine predstavlja istraživanje mogućnosti. Ako je broj pokušaja akcije a mali, posledično će nesigurnost u rezultat biti velika, što će činiti verovatnije da će ova radnja biti odabrana.

Nakon odabira strategije i odluke da li da pitamo korisnika ili ne, ako je odluka bila pozitivna, a korisnikov odgovor promenio predviđanje u pravu klasu, nagradu jemo akciju jednim bodom i razlikom između poslednjeg i novog predviđanja. Ako predviđanje ostane isto, nagrada će biti jedan poen. S druge strane, ako je odlučeno da anketu ne prikazemo korisniku, nakon trećeg puta nepitanja, korisnik će dobiti upitnik i ako je odgovor drugačiji od prethodno predviđenog, akcija će dobiti negativne poene, međutim ako je odgovor isti kao što je predviđeno, dobiće jedan bod. U početku će UCB izabrati svaku akciju jednom, pošto u bazi nema prethodnih podataka.

5. OPIS APLIKATIVNOG REŠENJA

U ovom poglavlju opisano je aplikativno rešenje koje obuhvata sve funkcionalnosti opisane u prethodnim poglavljima, koristeći delove šeme baze podataka namenjene za čuvanje podataka o korisnicima, podešavanjima naloga, vezama između korisnika, fotografijama i komentarima. Aplikativno rešenje je zamišljeno kao veb aplikacija i implementirano je pomoću tehnologija koje su prethodno navedene. Svaka funkcionalnost je detaljno opisana, navodeći ulogu korisnika za koju je namenjena kao i postupak koji je neophodno uraditi za izvršavanje iste. Prilikom popunjavanja formi, navedeni su koraci koje je neophodno ispratiti, tako što je naglašeno koje informacije treba uneti, kao i koje dugme kliknuti za potvrđivanje željene radnje.

6. ZAKLJUČAK

Osnovni zadatak ovog rada je bio razvoj i demonstracija Android okvira za prikupljanje i obradu informacija o korisničkom iskustvu. Osnovna ideja je bila smanjenje potrošnje energije prilikom korišćenja mobilnog telefona, odnosno, optimizacija delova aplikacija u skladu sa kontekstom u kom se korisnik nalazi.

Za demonstraciju ovog okvira, korišćena je aplikacija nalik YouTube-u koja podržava funkcionalnosti vezane za video zapise.

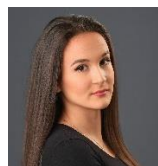
Dosadašnji rad obuhvata beleženje podataka iz senzora mobilnog telefona u toku korišćenja aplikacije, obradu tih podataka i predlaganje korisničkog zadovoljstva različitim parametrima u različitim kontekstima.

Aplikacija je razvijana u trenutno aktuelnim tehnologijama i ima dobru osnovu za dalji razvoj.

7. LITERATURA

- [1] Joseph A Paradiso and Thad Starner. 2005. Energy scavenging for mobile and wireless electronics. *IEEE Pervasive computing* 4, 1 (2005), 18–27
- [2] https://link.springer.com/chapter/10.1007/978-3-319-31413-6_8
- [3] Abdesslem et al. 2010; Froehlich et al. 2007
- [4] <http://lrss.fri.uni-lj.si/Veljko/docs/Pejovic18AMC.pdf>
- [5] <https://ieeexplore.ieee.org/abstract/document/7092486>
- [6] file:///C:/Users/lunaz/OneDrive/Desktop/Master%20rad/hab%20erl_silva_pmf_2017_diplo_sveuc.pdf
- [7] https://www.theseus.fi/bitstream/handle/10024/133782/Lyytinen_Jere.pdf?sequence=1&isAllowed=y
- [8] <https://awareframework.com/what-is-aware/>
- [9] [https://en.wikipedia.org/wiki/Weka_\(machine_learning\)](https://en.wikipedia.org/wiki/Weka_(machine_learning))
- [10] <https://yandex.com/dev/maps/mapsapi/?from=mapsapi>
- [11] https://link.springer.com/chapter/10.1007/978-1-4842-2943-9_3
- [12] https://www.researchgate.net/profile/Chunnu-Khawas/publication/325791990_Application_of_Firebase_in_Android_App_DevelopmentA_Study/links/5bab55ed45851574f7e6801/Application-of-Firebase-in-Android-App-Development-A-Study.pdf
- [13] <https://www.simplilearn.com/tutorials/machine-learning-tutorial/random-forest-algorithm#:~:text=A%20Random%20Forest%20Algorithm%20is,more%20it%20will%20be%20robust.>

Kratka biografija:



Luna Živković, rođena 14.02.1999. u Novom Sadu. Završila osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu, smer Računarstvo i automatika.

Kontakt: lunazivkovic@gmail.com



Milan Vidaković rođen je u Novom Sadu 1971. Doktorirao je na Fakultetu tehničkih nauka 2003. god, a od 2014. je u zvanju redovni profesor iz oblasti Primenjene računarske nauke i informatika.

НАМЕНСКИ ЈЕЗИК ЗА МОДЕЛОВАЊЕ МИГРАЦИЈЕ РЕЛАЦИОНЕ БАЗЕ ПОДАТАКА У БАЗУ ПОДАТАКА ЗАСНОВАНУ НА ДОКУМЕНТИМА**A DOMAIN SPECIFIC LANGUAGE FOR MIGRATION OF RELATIONAL DATABASE TO DOCUMENT-ORIENTED DATABASE**

Милица Вучинић, Факултет техничких наука, Нови Сад

Област – ЕЛЕКТРОТЕХНИЧКО И РАЧУНАРСКО ИНЖЕЊЕРСТВО

Кратак садржај – У овом раду представљен је текстуални наменски језик за моделовање миграције релационе базе података у базу података засновану на документима. Модел миграције састоји се од модела релационе базе, модела базе засноване на документима и моделу мапирања. На основу језика, могуће је развити више генератора при чему су у овом раду описани генератори за креирање валидатора колекције и индекса (миграција ограничења) као и генератори за сваку врсту мапирања миграције (мигрирање података). Резултујући код је генерисан у програмском језику python. За развој мета-модела коришћено је радно окружење Eclipse Modeling Framework, радни оквир Xtext за развој текстуалне синтаксе и језик Xtend за развој генератора.

Кључне речи: релационе базе података, нерелационе базе података, миграција, наменски језици

Abstract – In this thesis, we present a textual domain-specific language for migration of relational database to document-oriented database. Migration model consists of relational database model, document-oriented database model and mapping model. Multiple generators can be created based on this language and in this thesis presented generators are: collection validator and index generator (migration of constraints) and generator for each mapping type (data migration). Resulting code is generated in python programming language. Eclipse Modeling Framework was used for the creation of the meta-model. Xtext was used to develop the concrete language syntax, while Xtend was used for the implementation of documentation generators

Keywords: Relational databases, Nosql databases, Migration, Domain-Specific Languages

1. УВОД

Успоном интернета и експоненцијалним растом количине података, појављују се нови изазови са којима релационе базе не успевају да се изборе у потпуности.

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији је ментор био др Милан Челиковић, доцент.

Нерелационе базе података, такође познате и као NoSql базе података [1], ступају на сцену са новим приступима решавања проблема. Пројектоване тако да подрже велике количине неструктурираних и полуструктурираних података, интеграцију са дистрибуираним системима као и скалабилне архитектуре много ефикасије од релационих система.

Приликом поређења NoSql база података са релационим, постаје евидентно да NoSql базе података нуде шири спектар модела података при чему сваки од њих задовољава специфичне захтеве и случајеве коришћења. Једна од тих врста модела јесте модел података заснован на документима.

Циљ овог рада јесте представити један од могућих начина миграције података и ограничења из постојеће релационе базе података у базу података засновану на документима. Решење које је у овом раду представљено јесте наменски језик за моделовање те миграције Migration Domain Specific Language (migDsl) као и пратећи генератори кода. Да би се моделовали детаљи овог процеса, описани наменски језик укључује концепте релационе базе података, концепте базе података засноване на документима и подржане врсте мапирања од којих се миграција састоји.

2. ТЕОРИЈСКЕ ОСНОВЕ МОДЕЛИМА ВОЂЕНОГ РАЗВОЈА СОФТВЕРА И НАМЕНСКИХ ЈЕЗИКА

Моделима вођен развој софтвера (Model-Driven Software Engineering, скраћено MDSE) се може дефинисати као методологија која примењује предности моделовања у активностима развоја софтвера. Као и свака методологија, састоји се од следећих аспеката: концепти, нотације, процеси, правила и алати [2].

Језици за моделовање представљају централну компоненту MDSE методологије. Могу се поделити на наменске језик (енг. domain-specific language) и језике опште намене (енг. general-purpose language). Језици опште намене представљају нотације за моделовање употребљиве у било ком сектору или домену. Типични примери језика опште намене су UML (Unified Modeling Language) и Петријеве мреже. Наменски језици су језици развијени за конкретан домен, контекст или компанију са циљем да омогуће моделовање прилагођено искључиво том домену. Сваки језик има три кључна елемента: апстрактну синтаксу (структура језика и начини комбинације

основних елемената), конкретну синтаксу (репрезентација језика) и семантику (значење елемената). Трансформације, поред самих модела, представљају главну компоненту *MDSE* методологије. Постоје две врсте трансформација: трансформације модела у модел и трансформације модела у текст.

3. ПРЕГЛЕД СТАЊА У ОБЛАСТИ

Традиционално, *etl* (*extract, transform, and load*) процес се обављао коришћењем алата, као што су *Informatica*, *Talend*, *SAP Data Services* и остали. Ови алати обично поседују *drag and drop* интерфејс који олакшава креирање *data pipeline*-ова. Овај начин је познат и као *tool-based* приступ.

Међутим, код *tool-based* приступа се јављају нека ограничења, па осим њега постоји и *code-based* приступ који подразумева писање програма и скрипти од стране програмера. Једна од главних предности *code-based* приступа је флексибилност. Са језицима као што су *python* и *R* могуће је извршити много комплексније трансформације које одговарају специфичном проблему. Осим тога, *code-based* приступ омогућава успешну сарадњу на нивоу тима, захваљујући контроли верзија. Још једна предност је могућност рада на *cloud* платформама што може донети уштеду и могућност скалирања.

У раду [3] изложен је приступ *code-based etl*-а чије команде се излажу коришћењем наменског језика, уместо програмског или упитног језика. Граматика је изражена у Бакус-Науровој форми, синтакса се темељи на релационој алгебри и резултат који настаје из *dsl* скрипте је код у програмском језику *Java*.

Циљ рада [4] јесте да се предложи правила трансформација за три различите категорије нерелационих база: колонске, граф и базе оријентисане ка документима. Правило трансформације је дефинисано за сваку врсту везе која постоји у релационој бази података (један на један, један на више, више на више, наслеђивање-специјализација,

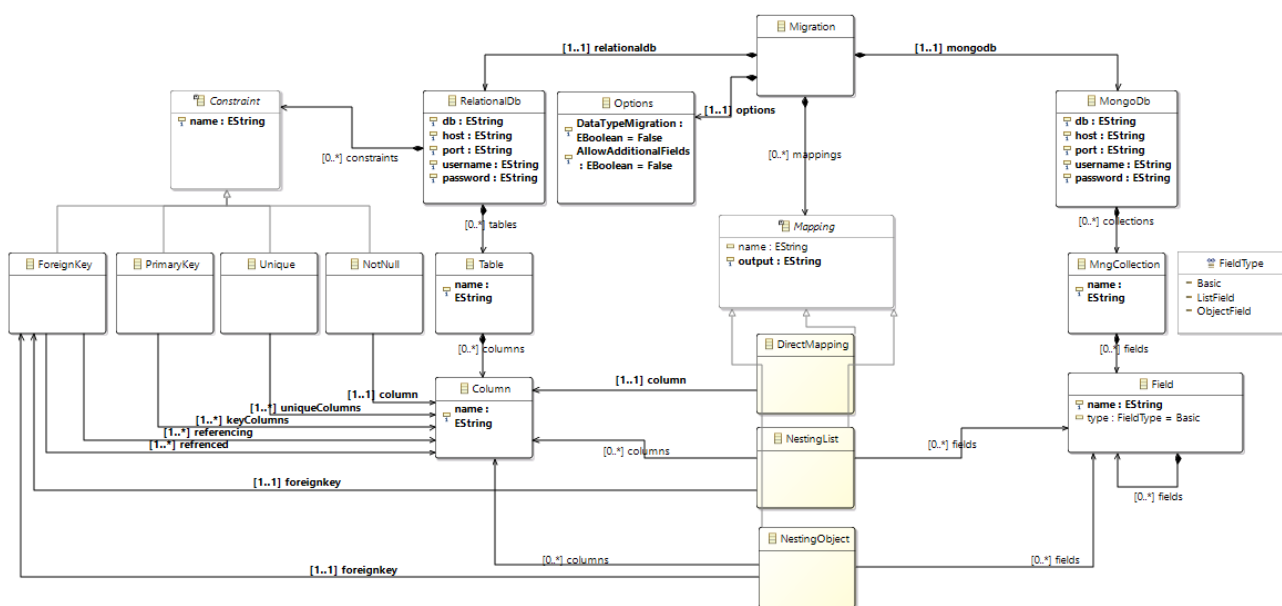
наслеђивање-унија и агрегација). Што се тиче база оријентисаних ка документима, издвојена су два начина моделовања веза, а то су референцирање и угњеждавање, који уједно представљају и два могућа резултата трансформација.

Циљ рада [5] јесте да се установи која *NoSql* категорија подржава комплетну миграцију из релационе базе, знајући да су усвојене различите структуре које не имплементирају одређена правила релационог модела. Аутори су, након детаљне анализе релационог модела, установили 19 битних критеријума. Најбољи резултат, односно најмање губитка информација, има миграција у базу података засновану на документима.

Etl процеси и миграција података представљају кључне аспекте у управљању подацима и омогућавају ефикасну интеграцију, трансформацију и анализу података унутар организације. Познавање и разумевање различитих приступа и алата је од суштинског значаја за успех у модерном пословном окружењу а прави избор свакако зависи од природе проблема који се решава.

4. НАМЕНСКИ ЈЕЗИК ЗА МИГРАЦИЈУ РЕЛАЦИОНЕ БАЗЕ ПОДАТАКА У БАЗУ ПОДАТАКА ЗАСНОВАНУ НА ДОКУМЕНТИМА

Први корак у развоју наменског језика *migDsl* је креирање апстрактне синтаксе која је приказана помоћу мета-модела (Слика 4.1.). За креирање мета-модела коришћен је радни оквир *Eclipse Modeling Framework (EMF)* који као језик за мета-моделовање користи језик *Ecore* [6]. Ограничења су описана помоћу наменског језика *Object Constraint Language (OCL)*, односно имплементације овог језика под називом *Eclipse OCL*. На основу апстрактне синтаксе креирана је конкретна синтакса, користећи радни оквир *Xtext*.



Слика 4.1. – Апстрактна синтакса наменског језика *migDsl*

4.1. Апстрактна синтакса

Коренски концепт апстрактне синтаксе је миграција (*Migration*). Миграција мора да има једну релациону базу података (*RelationalDb*), једну *Mongo* базу података (*MongoDb*) и опције миграције (*Options*). Осим тога, коренски елемент може да има више мапирања кроз које се остварује процес миграције.

Релациона база може да има више табела (*Table*) од којих свака може имати више колона (*Column*). Са друге стране, *Mongo* база података има низ својих колекција (*MngCollection*) које се састоје од поља (*Field*). За разлику од колоне, поље може да има своја угњевжана поља, објекте и листе. Поља колекција и колоне табела представљају улазе и излазе мапирања по правима врсте мапирања.

Ради потребе мигрирања и самих ограничења у новонасталу базу података, осим табела, релациона база има и листу ограничења (*Constraint*). Уведен је концепт за сваку врсту ограничења, осим за *check* ограничење.

У језику *migDsl* тренутно су подржана три типа мапирања:

- *DirectMapping* - моделује директно мапирање постојеће колоне у релационој бази података на ново поље документа. Поседује само асоцијацију према колони.
- *NestingObject* - Резултат ове врсте мапирања је нови објекат чији је назив назначен у оквиру обележја *output* апстрактне класе *Mapping*. Поља новог објекта су настала од референцираних колона (асоцијација *columns*) и постављена је и референца на њих после креирања (асоцијација *fields*). За ово мапирање мора постојати страни кључ при чему су изворне колоне из табеле у којој се налази референцирана колона тог страног кључа.
- *NestingList* - Разлика у односу на прошлу врсту мапирања је то што не настаје један објекат, већ листа објеката. Овога пута, референциране колоне мапирања морају припадати табели у којој се налази референцирајућа колона страног кључа (спојној табели).

4.2. Конкретна синтакса са примером

За наменски језик *migDsl* почетна верзија граматике креирана је на основу мета-модела описаног у претходном поглављу. У том процесу, овај радни оквир креће од коренског елемента, који је у овом случају Миграција.

Да би се кренуло са писањем конкретне синтаксе језика *migDsl*, потребно је пре свега попунити фајл конфигурације. Фајл треба да садржи креденцијале за конекције над изворном и одредишном базом као и опције миграције. У претходном поглављу, приказан је велики број референци од којих се метамодел састоји, односно за креирање мапирања, потребно је да концепти релационе базе већ постоје. Да корисник не би морао ручно да уноси део мета-модела који описује релациону шему, она је учитана у метамодел на основу садржаја фајла конфигурације.

Да би се добавила читава шема релационе базе података и читала у метамодел, прави се низ упита

ка речнику података. Конкретно, ка табелама *all_tables*, *all_tab_columns*, *user_constraints* и *user_cons_columns*.

Пример који је одабран за демонстрацију је релациона шема позоришта и креирање нове колекције *actors* у *Mongo* бази података (*Листинг 4.1.*).

У табели *ACTOR*, колона *NAME_SC* се мапира у поље *name*, а колона *STATUS_AC* у поље *status*. Веза глумца и позоришта је моделована кроз страни кључ *ACTOR_THEATRE_FK*, стварајући објекат *theatre*. Табела *ASSIGNMENT* представља спојну таблу за везу глумца и улога. За добијање листе прошлих улога, користи се четврто мапирање. Пето мапирање директно мапира колону *SALARY* у поље *salary*.

```
{
    output: "actors.name"
    input: "ACTOR.NAME_AC"
}
{
    output: "actors.status"
    input: "ACTOR.STATUS_AC"
}
{
    NESTING_OBJECT
    output: "actors.theatre"
    inputs: ACTOR_THEATRE_FK
    ("THEATRE.NAME_TH", "THEATRE.WEBSITE_TH", "THEATRE.ADDRESS_TH")
    AS ( "actors.theatre.name", "actors.theatre.website",
        "actors.theatre.address" )
}
{
    NESTING_LIST
    output: "actors.roles"
    inputs: ASSIGNMENT_ROLE_FK
    ("ROLE.NAME_RO", "ROLE.TYPE_RO", "ROLE.ID_RO")
    AS ( "actors.roles.role_name", "actors.roles.role_type",
        "actors.roles.role_id" )
}
{
    output: "actors.salary"
    input: "ACTOR.SALARY_AC"
}
```

Листинг 4.1 - Пример конкретне синтаксе језика *migDsl*

5. ГЕНЕРАТОРИ КОДА

Генерисање кода подразумева генерисање два фајла у програмском језику *python*. Садржај првог фајла представља код за дефинисање ограничења, односно валидатора за сваку новокреирану колекцију. Осим тога, за сваку колекцију генеришу се и индекси. У обзир се узимају опције миграције које су постављене у фајлу конфигурације. Садржај другог фајла су *sql* упити ка изворној бази података, учитавање података, њихова трансформација у одговарајући облик за мапирање и пренос у одредишну базу података. Сви параметри конекције су преузети из мета-модела, а специфицирани су у конфигурационом фајлу.

Ако је опција *DataTypeMigration* постављена на *True*, за свако поље се умеће нови шаблон у коме се за поље налази изворна колона, прави упит ка релационој бази података који на основу назива колоне и табеле враћа тип податка. Затим се тај тип податка из *Oracle* базе мапира на одговарајући тип у *Mongo* бази података. Опција валидатора *additionalProperties* такође зависи од вредности у опцијама миграције. Ова опција дозвољава или забрањује појаву поља у колекцији која нису дефинисани унутар валидатора.

Приликом генерисања *python* фајла за миграцију података, пролази се кроз листу свих мапирања миграције, и у зависности од врсте мапирања позива се одговарајући шаблон. Пре овога, на самом почетку фајла, уметнути су и статички шаблони за потребне импорте и остваривање конекција. Пример шаблона за директно мапирање приказан је на листингу 5.1.

```
def String getDirectMappingContent(Mapping mapping, RelationalDb relDb){

    val map = mapping as DirectMapping
    val input_table_name =
Utils.getTableFromReferencedColumn(relDb, map.column)
    val collection_name =
Utils.getCollectionNameFromOutput(map.output)
    val output_field = Utils.getFieldNameFromOutput(map.output)
    ""
    # %% Direct mapping of «map.column.name» «IF
map.name!=null»:«map.name» «ENDIF»

    cursor = connection.cursor()
    cursor.execute("SELECT «map.column.name» FROM
«input_table_name»")
    rows = cursor.fetchall()
    cursor.close()
    simple_list = []
    simple_list = [item[0] for item in rows]

    collection = database['«collection_name»']

    for index,i in enumerate(simple_list):
        new_document = {"«output_field»": i}

        if collection.count_documents({}) == index:

            collection.insert_one(new_document)

        else:
            cursor = collection.find()
            count = 0
            for document in cursor:
                count += 1
            if count == index+1:

                collection.update_one({'_id': document['_id']}, {'$set':
new_document})
            ""
    }
```

Листинг 5.1 – Шаблон за директно мапирање

6. ЗАКЉУЧАК

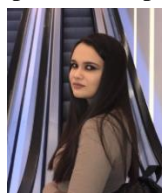
Наменски језик *migDsl* омогућава креирање формалног модела једне миграције из релационе базе података у базу података засновану на документима. Миграција се састоји од низа мапирања који референцирају концепте релационе базе и базе засноване на документима, при чему су сви део апстрактне синтаксе језика. У раду је описано и више генератора који имају за циљ генерисање програмског кода. Резултујуће скрипте садрже код за извршавање сваког мапирања, валидатора и индекса за колекције. Доступне су и додатне опције миграције.

Овај наменски језик могуће је проширити концептима који би подржали миграцију у дистрибуирани систем. Осим генератора за миграцију података и валидацију шеме, могли би се додати и генератори скрипти за остале административне задатке. На пример, скрипта за извршавање *backup*-а или конфигурацију појединачних кластера. Једна од предности овог наменског језика је његова проширивост. Поред три подржана типа мапирања, на једноставан начин је могуће додати још типова. Примери би били: мапирање у коме поље колоне настаје тако што се обави нека операција над колоном табеле или комбиновање више колона за резултујуће поље документа.

7. ЛИТЕРАТУРА

- [1] Vatika Sharma, Meenu Dave. SQL and NoSQL Databases. Доступно на: https://www.researchgate.net/profile/Meenu-Dave/publication/303856633_SQL_and_NoSQL_Databases/links/5758557f08ae9a9c954a7573/SQL-and-NoSQL-Databases.pdf
- [2] Marco Branbilla, Jordi Cabot, Manuel Wimmer, Model-Driven Software Engineering in Practice Morgan & Claypool Publishers 2012.
- [3] Sunisa Junsawang, Yachai Limpiyakorn. A Domain Specific Language for Scripting ETL Process. 2017. Доступно на: http://www.wcse.org/WCSE_2017/041.pdf.
- [4] Obaid Alotaibi, Eric Pardede. Transformation of Schema from Relational Database (RDB) to NoSQL Databases. 27.11.2019. Доступно на: <https://www.mdpi.com/2306-5729/4/4/148>.
- [5] Abdelhak Erraji, Abderrahim Maizate, Mohammed Ouzzif. Multi-criteria Analysis Between Nosql Databases Categories Toward A Complete Migration From Relational Database. 15.1.2022. Доступно на: <http://www.jatit.org/volumes/Vol100No1/6Vol100No1.pdf>
- [6] Eclipse Modeling Framework (EMF) документација. Доступно на: <https://www.eclipse.org/modeling/emf>

Кратка биографија:



Милица Вучинић рођена је 1999. год. у Невесињу, Босна и Херцеговина. Факултет техничких наука уписала је 2018. године. Дипломски рад из области Електротехника и рачунарство – Рачунарске науке и информатика одбранила је 2022. године. Мастер студије на Факултету техничких наука уписала је 2022. године.

**РАЗВОЈ СОФТВЕРА ЗА ОРГАНИЗАЦИЈУ РАСПОРЕДА ЧАСОВА У
ВИСОКОШКОЛСКИМ УСТАНОВАМА****DEVELOPMENT OF SOFTWARE FOR ORGANIZING CLASS SCHEDULES IN
HIGHER EDUCATION INSTITUTIONS**

Аница Ђукић, Никола Лубурић, Факултет техничких наука, Нови Сад

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – Овај рад описује развој софтвера за организацију распореда наставе у високошколским установама поштујући DDD приступ развоју софтвера. Описан је рад високошколске установе, представљени захтеви, анализирана слична решења и представљена слојевита архитектура по којој је развијена серверска страна развијеног система.

Кључне речи: високошколске установе, DDD приступ развоју софтвера, организација наставе, слојевита архитектура

Abstract – This paper describes the development of software for organizing instruction in higher education institutions, following the Domain-Driven Design (DDD) approach to software development. The work outlines the higher education institution, presents the requirements, analyzes similar solutions, and introduces a layered architecture used on the server-side of the system.

Keywords: Higher Education Institutions, Domain-Driven Design, Instruction Organization, Layered Architecture

1. УВОД

Од кад постоје образовне установе, организација наставе и прављење распореда часова представљају значајан проблем, како у основним и средњим школама, тако и у високошколским установама. Међутим, у високошколским установама овај проблем је посебно изражен зато што треба ускладити наставу на предметима различитих студијских програма који се врло често мењају, доступност наставног особља и капацитете и доступност просторија за извођење наставе.

Решавање овог проблема би у великој мери олакшао и смањило свакодневни посао службеницима којима је један од послова прављење распореда часова. Такође, решавање овог проблема би могло да допринесе квалитету наставе ако би настава била организована тако да се између часова праве довољно мале паузе и уколико настава не би почињала рано ујутру и завршавала се у касним вечерњим сатима.

НАПОМЕНА:

Овај рад произтекао је из мастер рада чији ментор је био др Никола Лубурић, доцент.

За постизање свега наведеног потребан је софтвер који омогућава прављење оптимизованог распореда часова поштујући сва правила која постоје у високошколским установама.

С обзиром на комплексност проблема, овакав софтвер би било најбоље изградити по DDD (Domain Driven Design) [1] приступу за развој софтвера.

DDD приступ развоја софтвера се бави изазовом разумевања домена проблема (енгл. *Problem Domain*) [1] и креирања одрживог решења које је корисно за решавање проблема унутар тог домена [1].

Прикупљањем захтева од службеника, анализом докумената и кроз разговор са професорима и студентима, направљен је софтвер који омогућава генерисање распореда часова, поштујући сва правила која постоје у раду високошколских установа. Међутим, овом софтверу је потребно омогућити податке. Тема овог рада је развој софтвера који омогућава организацију распореда часова кроз унос, измену, брисање и прослеђивање неопходних података софтверу за прављење распореда часова.

2. ДОМЕН ПРОБЛЕМА

Према DDD приступу, да би се проблем боље разумео потребно је домен проблема разложити на поддомене који су релевантни за решавање проблема.

Високошколска установа се према [2] састоји из неколико поддомена, али овде су разматрани само поддомени који су везани за распоред часова, а то су:

- Академски послови
- Финансије и администрација
- Управљање објектима
- Студентски послови

2.1. Академски послови

Овај поддомен је кључан за функционисање саме установе и често је организован око академских дисциплина и поља студија. Наставници и сарадници су обично организовани у департмане [2].

2.1.1. Организационе јединице

Департмани су образовне, научне, уметничке и предузетничке организационе јединице Факултета које реализују један или више акредитованих студијских програма који припадају најмање једној научној области.

Департмани као своје организационе јединице могу имати: одсеке, катедре, лабораторије и друге организационе јединице [3].

Катедру чине наставници и сарадници, који изводе наставу из сродних наставних предмета, а који су изабрани за уже научне односно уметничке области које припадају катедри [3].

2.1.2. Наставни план и програм

Наставни план и програм је уређен скуп предмета и грађе предмета изучавања на Факултету, развијених по редоследу извођења, броју часова предавања и вежби и међусобној условљености са другим предметима у плану [4].

Наставни план и програм је важан за распоред часова, да би се знало колико часова вежби и предавања се изводи на неком предмету унутар неког студијског програма.

2.1.3. Студијски програм

Студијски програм је скуп обавезних и изборних студијских подручја, односно предмета, са оквирним садржајем, чијим се савладавањем обезбеђују неопходна знања и вештине за стицање дипломе одговарајућег степена и врсте студија [3, 4].

Врсте студија према [3] су академске студије и струковне студије, а њихови степени су: студије првог степена (основне академске, основне струковне и специјалистичке струковне студије), студије другог степена (мастер академске, мастер струковне и специјалистичке академске студије) и студије трећег степена (докторске академске студије).

2.1.4. Реализација студијског програма

Реализација студијског програма представља извођење предмета наставног плана и програма који је дефинисан за одређени студијски програм. Као што је већ напоменуто, студијски програм је скуп обавезних и изборних предмета, а реализација студијског програма је извођење тих предмета.

2.2. Финансије и администрација

Факултети су послодавци који морају управљати стотинама или хиљадама запослених. То захтева уређивање платног списка, здравственог осигурања и пензионих доприноса свих запослених [2].

2.2.1. Кадровска евиденција

Овај поддомен је за распоред часова важан због кадровске евиденције запослених наставника и сарадника на факултету. Из кадровске евиденције се може добити ажуриран списак запослених са свим информацијама које су релевантне за распоред часова.

2.3. Управљање објектима

Факултети обично поседују десетине зграда које служе у различите сврхе. Поред зграда, они су власници и одговарају за путеве, тротоаре, паркинге, паркове, фонтане и све остало на територији кампуса [2].

2.3.1. Евиденција просторија

Овај поддомен високошколске установе је важан због евиденције просторија. Овде се могу добити информације о просторијама које могу бити важне за распоред часова.

Наставно особље обично држи наставу у просторијама које припадају њиховој организационој јединици. Ово правило се најчешће односи на вежбе.

2.4. Студентски послови

Академски послови су сржи установе, али ваннаставно искуство такође представља значајну компоненту студентског живота. Када се разматрају сектори студентских послова, обично се подразумевају студентске организације, активности на кампусу и слично, али такви сектори су много разноврснији, пружајући студентима различите видове подршке [2].

2.4.1. Студентска служба

Поред сектора за ваннаставне активности, у студентске послове спада и студентска служба, којој су према [5] нека од задужења упис нових студената и упис студената у наредну школску годину.

2.4.2. Студентске групе

Студентска служба је важна због евиденције уписаних студената и њиховим распоређивањем у студентске групе. Битан је број студената и број студентских група, као и број студената унутар студентских група за сваку годину и степен студија на сваком студијском програму.

3. ЗАХТЕВИ ЗА СОФТВЕРСКО РЕШЕЊЕ

Софтверско решење подржава следеће захтеве:

- Манипулисање предметима, предавачима, просторијама, студијским програмима и студентским групама
- Додавање предмета у реализацију
- Измена предмета у реализацији
- Уклањање предмета из реализације
- Прављење копије унетих података
- Прављење распореда часова

Манипулација ентитетима подразумева операције додавања, измене и брисања ентитета, као и претрагу.

3.1. Додавање предмета у реализацију

Корисник може да дода предмет у реализацију студијског програма. Додавање предмета у реализацију студијског програма се односи на унос предавача који ће учествовати у реализацији наставе тог предмета на студијском програму коме тај предмет припада.

3.2. Измена предмета у реализацији

Корисник може да измени предмет у реализацији. Измена предмета у реализацији студијског програма се односи на измену предавача који ће учествовати у реализацији наставе тог предмета.

3.3. Уклањање предмета из реализације

Корисник може да уклони предмет из реализације студијског програма. Уклањањем предмета из реализације значи да се предмет више не изводи.

3.4. Прављење копије унетих података

Корисник може да направи копију свих података које је унео на нивоу семестра и да у сваком моменту прегледа податке из било ког семестра.

3.5. Прављење распореда часова

Корисник може да направи распоред од података изабраног семестра. Када се прављење распореда заврши систем ће кориснику на мејл проследити направљени распоред часова.

4. ПРЕГЛЕД СЛИЧНИХ РЕШЕЊА

4.1. aSc TimeTables

aSc TimeTables [6] је комерцијални софтверски алат за прављење распореда часова који пружа напредне могућности оптимизације, омогућавајући кориснику да брзо и ефикасно прави распоред који задовољава све захтеве.

Предности овог система су напредне могућности оптимизације, интуитиван кориснички интерфејс и подршка за различите типове школа и факултета. Недостаци овог система су високи трошкови лиценце и додатно време за прилагођавање и учење.

4.2. UniTime

UniTime [7] је софтверско решење отвореног кода које пружа могућност планирања и генерисања распореда часова. С обзиром да је систем отвореног кода корисници имају могућност да имплементирају и додају сопствене функционалности.

Предности овог система су бесплатно коришћење, прилагодљивост, развијена заједница корисника и подршка. Недостаци овог система су мање интуитиван интерфејс и потреба за већом техничком експертизом.

4.3. TimetablePro

TimetablePro [8] је софтверско решење које пружа могућност ефикасног генерисања распореда часова за факултете. Овај систем је посебно дизајниран за средње и велике образовне институције и нуди разноврсне функционалности организације распореда.

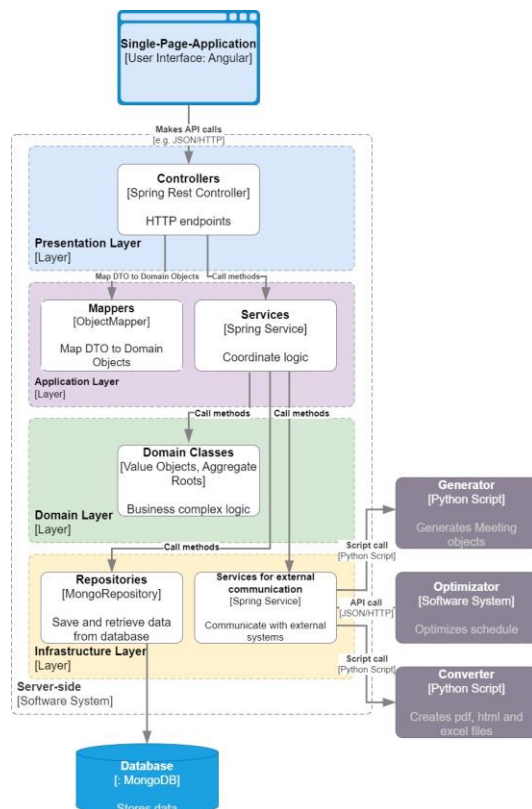
Предности овог система су напредне могућности оптимизације, прилагођавање специфичним захтевима и интуитиван интерфејс. Недостаци овог система су високи трошкови лиценце и интерфејс који може захтевати додатно време за упознавање.

5. ДИЗАЈН РЕШЕЊА

У овом поглављу је представљен дизајн софтвера који је развијен као решење.

5.1. Архитектура решења

На слици 5.1 приказана је архитектура развијеног решења у складу са C4 model – component view [9].



Слика 5.1. Архитектура софтверског решења

Софтверско решење се састоји од клијентске и серверске стране. Клијентску страну чини кориснички интерфејс који омогућава кориснику да приступи функционалностима система.

Серверска страна је имплементирана по слојевитој архитектури (енгл. *Layered architecture*) [10] коју чине:

- Презентациони слој (енгл. *Presentation layer*)
- Апликациони слој (енгл. *Application layer*)
- Доменски слој (енгл. *Domain layer*)
- Инфраструктурни слој (енгл. *Infrastructure layer*)

5.1.1. Презентациони слој

Презентациони слој служи за пружање информација кориснику и интерпретацију захтев корисника [10].

Овај слој чине класа контролера које су задужене за пријем захтева корисника, његово прослеђивање сервисним класама апликационог слоја и враћање одговора кориснику.

5.1.2. Апликациони слој

Апликациони слој је задужен за дефинисање задатака које би софтвер требало да обавља и координацију објеката домена. Овај слој не садржи пословна правила или знање, већ само координише и делегира рад објектима домена [10].

Овај слој се састоји од класа сервиса и мапера. Класе сервиса координишу рад доменских објеката, објеката других сервиса и класа задужених за рад са базом.

5.1.3. Доменски слој

Доменски слој је задужен за представљање пословних концепата и пословних правила [10].

Овај слој је пословно срце софтвера и састоји се од доменских класа које садрже пословну логику софтвера.

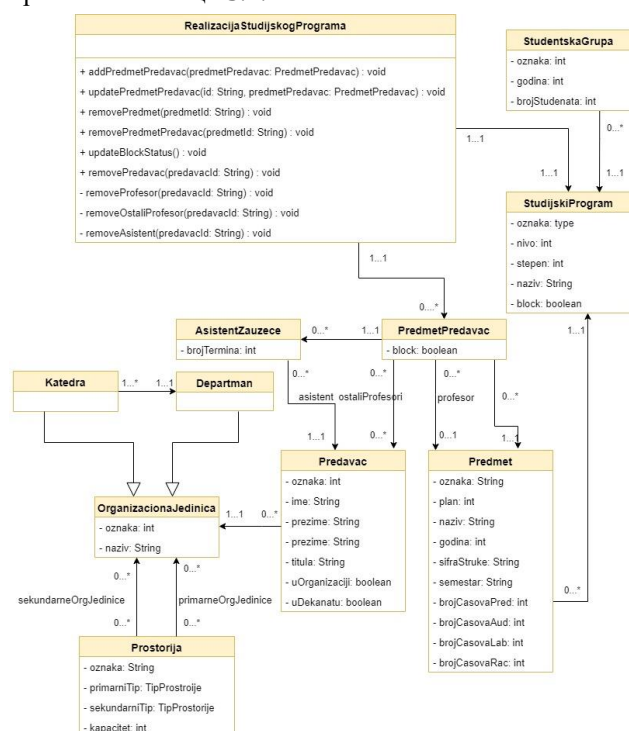
5.1.4. Инфраструктурни слој

Инфраструктурни слој обезбеђује генеричке техничке могућности. Овај слој омогућава комуникацију и размену података са другим системима [10].

У овом слоју се налазе класе репозиторијума које су задужене за рад са базом података и класе сервиса за комуникацију са екстерним системима.

5.2. Дизајн модела података

На основу поглавља 2 формиран је модел података приказан на слици 5.2.



Слика 5.2. Модел података софтверског решења

6. ЗАКЉУЧАК

У овом раду представљена је потреба за развојем софтвера за прављење распореда часова. Описан је рад високошколске установе тако што је разложена на поддомене који су релевантни за решавање проблема. Изложени су сви захтеви које софтвер подржава, представљена је архитектура и модел података.

За сада је претпоставка да ће софтвер користити само запослени коме је једно од задужења прављење распореда часова. Међутим, идеја је да се омогући да софтвер користе и предавачи који би имали могућност прегледа свог распореда и уређивање личних података који се користе у распореду часова. Поред тога, професори би имали могућност да додају и уклањају асистенте на својим предметима.

Такође, могуће је проширити систем пружањем могућности да се интегрише са другим системима високошколске установе и тиме омогући прикупљање података. На пример, повезивањем са системом кадровске службе би се могли прикупити ажурирани

подаци о наставном особљу или повезивањем са системом студентске службе би се могли прикупити подаци о студентским групама итд.

7. ЛИТЕРАТУРА

- [1] S. Millett and N. Tune, "Patterns, principles, and practices of domain-driven design," John Wiley & Sons, 2015.
- [2] R. M. Hendrickson, J. E. Lane, J. T. Harris, and R. H. Dorman, "Academic leadership and governance of higher education: A guide for trustees, leaders, and aspiring leaders of two-and four-year institutions," Stylus Publishing, LLC, 2013.
- [3] Факултет техниких наука, "Статут Факултета техниких наука од 02.07.2018. године са изменама и допунама од 17.10.2018. и од 28.12.2018. године," Доступно на: <http://www.ftn.uns.ac.rs/n676372513/statut-fakulteta-tehnickih-nauka-od-02-07-2018--godine-sa-izmenama-i-dopunama-od-17-10-2018--i-od-28-12-2018--godine> [Датум приступа: 28.07.2023.]
- [4] Д. Вилотић, Д. Шевић, Р. Дорословачки, "Q2.RU.04 – Поступак развоја наставних планова и програма (студијских програма), издање број 6," Факултет техниких наука, Универзитет у Новом Саду, 2019.
- [5] А. Купусинац, Н. Бркљач, С. Колаковић, "Q2.NA.03 – Поступак рада студентске службе, издање број 9," Факултет техниких наука, Универзитет у Новом Саду, 2023.
- [6] aSc Timetables. Доступно на: <https://www.ascstimetables.com/> .[Датум приступа: 30.07.2023.]
- [7] UniTime. Доступно на: <https://www.unitime.org/> .[Датум приступа: 30.07.2023.]
- [8] "Timetable Pro." Softonic. Доступно на: <https://timetable-pro.en.softonic.com/> . [Датум приступа: 30.07. 2023].
- [9] C4 Model. "The C4 model for visualizing software architecture." Доступно на: <https://c4model.com/> . [Датум приступа: 01.08.2023.]
- [10] E. Evans, "Domain-Driven Design: Tackling Complexity in the Heart of Software," Addison-Wesley Professional, 2003.

Кратка биографија:



Аница Ђукић рођена је у Новом Саду 1999. године. Мастер рад на Факултету техниких наука из области Електротехнике и рачунарства – Електронско пословање одбранила је 2023. године.
контакт: anica.djukic99@gmail.com

**SECURADSL – НАМЕНСКИ ЈЕЗИК ЗА ПОДРШКУ БРЗОГ УСПОСТАВЉАЊА
БЕЗБЕДНОСНИХ АСПЕКТА У РАДНОМ ОКВИРУ SPRING****SECURADSL - A DOMAIN-SPECIFIC LANGUAGE FOR RAPID CONFIGURATION
OF SECURITY ASPECTS IN THE SPRING FRAMEWORK**

Јелена Хрњак, Факултет техничких наука, Нови Сад

**Област – ЕЛЕКТРОТЕХНИЧКО И
РАЧУНАРСКО ИНЖЕЊЕРСТВО**

Кратак садржај – У овом раду описан је наменски језик *securaDSL* за моделовање *Spring* апликација са безбедносном конфигурацијом подржаном за три безбедносна механизма: основну аутентификацију, JWT и OAuth2.0. Поред наменског језика, развијени су генератори извршивог кода на основу модела креираног помоћу језика *securaDSL*. Употреба овог наменског језика и генератора олакшава, убрзава и унапређује квалитет рада доменских експерата у пољу безбедносних конфигурација софтвера. За развој мета-модела коришћено је окружење *Eclipse Modeling Framework* које за мета-моделовање користи језик *Ecore*. Додатна ограничења описана су помоћу наменског језика *Object Constraint Language*. За развој текстуалне синтаксе коришћен је радни оквир *Xtext*, а за развој генератора програмски језик *Java* и језик *Xtend*.

Кључне речи: наменски језици, развој софтвера вођен моделима, моделовање веб апликација, безбедносна конфигурација

Abstract – In this paper, we present *securaDSL*, a domain-specific language designed for modelling *Spring* applications with security configuration supporting three security mechanisms: Basic Authentication, JWT and OAuth2.0. Additionally, we present multiple generators that produce executable code based on models created using the *securaDSL*. The utilization of this domain-specific language and generators simplifies, accelerates, and enhances the quality of work for domain experts in the field of software security configurations. For the development of the meta-model, we used *Eclipse Modeling Framework* environment, utilizing the *Ecore* language for meta-modeling. To create the textual syntax, we used the *Xtext* framework, while we implemented generators using the *Java* programming language and *Xtend* language.

Keywords: Domain-Specific Languages, Model-Driven Software Development, Web Applications modeling, Security Configuration

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији ментор је био др Владимир Димитријески, доцент.

1. УВОД

Корисници веб апликација често нису свесни колико личних података оне прикупљају, обрађују и складиште. Неовлашћен приступ тим подацима могао би да доведе до злоупотребе и нарушавања приватности корисника. Стога, обезбеђивање високог нивоа безбедности представља важан део развоја веб апликација.

Безбедност у веб апликацијама представља скуп мера и механизма који су примењени како би се корисници, систем и подаци заштитили од различитих видова напада, злоупотреба и крађа. Имплементација жељеног нивоа аутентификације и ауторизације, као два основна концепта у области безбедности, представља основу ефикасне заштите.

Аутентификација представља процес потврђивања идентитета корисника или ентитета који приступа систему. Ауторизација се односи на контролу приступа одређеним ресурсима или функционалностима система.

Java је објектно-оријентисан програмски језик, а платформска независност, једноставност и објектна оријентисаност су особине које чине овај језик једним од најпопуларнијих [1]. Радни оквир *Spring* чини развој серверског дела веб апликација у програмском језику *Java* бржим, једноставнијим и сигурнијим што га чини најпопуларнијим радним оквиром за ову намену [2].

Међутим, обезбеђивање одговарајуће заштите за апликације у радном оквиру *Spring* представља сложен и временски захтеван процес, те је самим тим подложен грешкама. С обзиром да се безбедносни аспекти изнова конфигуришу при почетној имплементацији сваке апликације, овакав посао постаје и репетативан.

Са циљем уклањања наведених недостатака, тежи се оптимизацији и аутоматизацији развоја безбедних веб апликација. Једно од могућих решења представља аутоматско генерисање почетне *Spring* веб апликације са конфигурисаним безбедносним аспектима које се врши на основу параметара које корисник унесе, а који дефинишу основне карактеристике апликације и њених елемената.

За постизање овог циља креиран је наменски језик *Secura Domain-Specific Language (securaDSL)* за моделовање *Spring* веб апликација уз генераторе који на основу модела генеришу извршиви код написан у програмском језику *Java* и радног оквира *Spring*. Користећи језик *securaDSL*, експерти у пољу

безбедносних конфигурација могу брзо и једноставно да дефинишу параметре апликација и конфигуришу различите безбедносне механизме помоћу синтаксе која им је лако читљива.

Имплементација сигурних веб апликација на овај начин постаје једноставнија и ефикаснија, а уједно се смањује могућност грешака у процесу развоја.

2. ТЕОРИЈСКЕ ОСНОВЕ МОДЕЛИМА ВОЂЕНОГ РАЗВОЈА СОФТВЕРА И НАМЕНСКИХ ЈЕЗИКА

Развој софтвера вођен моделима (енгл. *Model-Driven Software Development*) је методологија у којој модел представља централну тачку у процесу развоја софтвера. Сваки модел се креира помоћу неког језика за моделовање. Језици за моделовање обухватају апстрактну и конкретну синтаксу, као и семантику. Апстрактна синтакса дефинише структуру језика и начин на који се различити концепти могу комбиновати без обзира на репрезентацију [3]. Конкретна синтакса, која може бити текстуална или графичка, описује специфичну репрезентацију језика за моделовање [3]. Семантика језика за моделовање описује значења и ограничења дефинисаних концепата помоћу којих се креирају модели, те самим тим и модели имају јасно дефинисану семантику.

Наменски језици, познати и као језици специфични за домен (енгл. *Domain-Specific Languages*), представљају језике за моделовање који су уско специјализовани и пројектовани за специфичан домен, контекст или компанију. Језик *securaDSL* представља наменски језик јер је пројектован и прилагођен домену безбедносних конфигурација *Spring* веб апликација.

Трансформације модела су битан корак при развоју софтвера вођеним моделом и подразумевају аутоматско генерисање циљног модела или текста на основу изворног модела. У оквиру овог рада имплементирани су трансформације модела у текст, који представља извршиви код веб апликације.

2.1 Опис технологија коришћених за развој наменског језика *securaDSL* и генератора

При развоју наменског језика *securaDSL*, први корак представља креирање апстрактне синтаксе која је приказана помоћу мета-модела. За креирање мета-модела коришћен је радни оквир за моделовање и генерисање кода *Eclipse Modeling Framework (EMF)* који као језик за мета-моделовање користи језик *Ecore*.

Додатна ограничења описана су помоћу наменског језика *Object Constraint Language (OCL)*, односно имплементације овог језика под називом *Eclipse OCL*. Након тога, на основу апстрактне креирана је конкретна синтакса помоћу радног оквира *Xtext*.

За развој генератора коришћен је програмски језик *Java* и језик *Xtend*.

Xtend доприноси детекцији типова података, побољшању синтаксе ламбда израза за лаку претрагу и издвајање података из модела и омогућава употребу шаблона помоћу којих се може описати изглед генерисаног кода.

3. ПРЕГЛЕД БЕЗБЕДНОСНИХ МЕХАНИЗАМА У *SPRING* АПЛИКАЦИЈАМА

Код који се генерише написан је у програмском језику *Java* коришћењем радног оквира *Spring*. Овај радни оквир је организован у модуле који нуде функционалности за подршку различитих аспеката развоја апликација.

Подржана су три релациона система за управљање базама података: *PostgreSQL*, *MySQL* и *Oracle Database*. Сва три система користе и проширују језик *SQL* за рад са подацима.

Радни оквир *Spring* пружа могућност безбедносне конфигурације веб апликација помоћу библиотеке *Spring Security*. Конфигурација се врши у складу са различитим безбедносним механизмима и на различитим нивоима, а конфигурацију је потребно прилагодити специфичним потребама система. Подржана су три безбедносна механизма: основна аутентификација, стандард *JSON* веб токен и стандард *Open Authorization*.

3.1 Основна аутентификација

Основна аутентификација (енгл. *Basic Authentication*) представља метод у ком се корисник идентификује помоћу корисничког имена и лозинке. При сваком захтеву се у заглављу захтева налазе идентификациони параметри корисника и на основу тога се потврђује идентитет и право приступа ресурсу или функционалности система. Овај метод се једноставно имплементира и користи, те је погодан за једноставније системе. Лозинке преносе у заглављу захтева, што их чини подложним нападима и може угрозити сигурност апликације, па се самим тим препоручује коришћење додатних безбедносних механизма.

3.2 Стандард *JSON* веб токен

JSON веб токен (енгл. *JSON web token, JWT*) представља формат за представу токена за аутентификацију. Састоји се од три дела: заглавље (енгл. *header*), главног дела (енгл. *payload*) и потписа (енгл. *signature*). Токен се генерише при свакој успешној аутентификацији и додељује се пријављеном кориснику, при чему садржи све неопходне податке о њему. При сваком захтеву се проверава валидност *JWT* токена на основу информација из њега и одређује се да ли је кориснику дозвољен приступ ресурсу или функционалности система.

3.3 Стандард *Open Authorization*

Стандард *Open Authorization (OAuth)* представља стандард за доделу права приступа који омогућава корисницима да доделе овлашћења апликацијама за приступ њиховим подацима који се налазе у другим апликацијама. Уместо уношења идентификационих параметара, сервер за доделу права приступа генерише токен који се користи за приступ ресурсима апликације. Овај стандард користи велики број компанија као што су *Google* и *Facebook*, како би омогућиле корисницима да поделе податке са својих корисничких налога са другим апликацијама. Последња верзија овог механизма је *OAuth2.0*.

4. ПРЕГЛЕД ПОСТОЈЕЋЕГ СТАЊА У ОБЛАСТИ

Савремене методе развоја софтвера теже брзом, безбедном и ефикасном развоју и смањењу потребе за ручно писаним кодом, што је довело до повећаног броја решења за генерисање различитих видова и нивоа софтвера.

Spring Initializer [4] је алат за брзо креирање основне структуре *Spring* апликација. Погодан је за једноставне пројекте, али је неопходна ручна конфигурација безбедносних аспеката, базе података и имплементација осталих слојева апликације.

У радовима [5,6] описани су алати који омогућавају и олакшавају процес спецификације и конфигурације микросервисне архитектуре, а уз то и генерисање извршивог кода за овако конфигуриране апликације. Ова решења нису погодна за генерисање монолитних апликација, иако су се показала као ефикасна за развој микросервисне архитектуре.

Анализа постојећих решења за моделовање апликација довела је до закључка да ниједно од решења не испуњава све захтеве у потпуности. Главни проблем који је уочен је недостатак могућности за конфигуравање безбедносних аспеката апликације.

5. НАМЕНСКИ ЈЕЗИК ЗА ПОДРШКУ БРЗОГ УСПОСТАВЉАЊА КОНФИГУРАЦИЈЕ БЕЗБЕДНОСНИХ АСПЕКТА У РАДНОМ ОКВИРУ *SPRING*

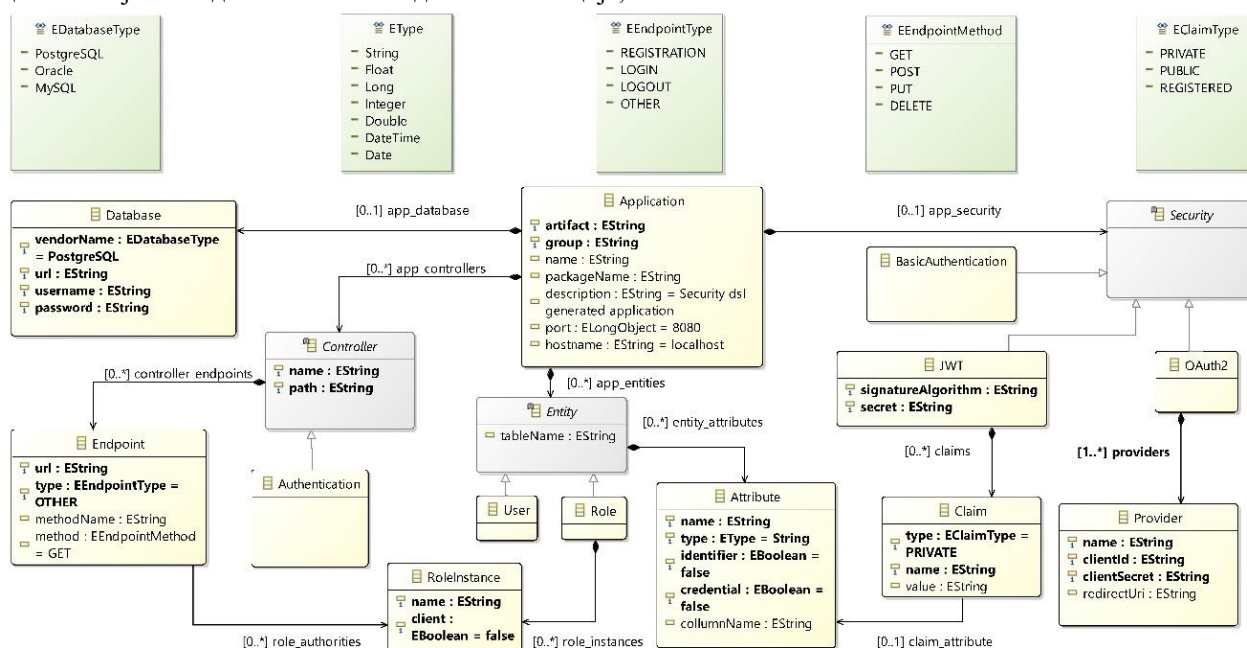
Недостаци описани у претходном поглављу, доводе до потребе за развојем наменског језика који би омогућио брзо успостављање конфигурације безбедносних аспеката. Да би наменски језик омогућио брзу и ефикасну конфигурацију у радном оквиру *Spring*, неопходно је да подржи моделовање свих неопходних концепата за иницијализацију апликације. Основни концепти се могу сврстати у целине које се односе на метаподатке апликације.

параметре базе података, слој за моделовање података складиштених у бази података, обраду захтева корисника и безбедносну конфигурацију. Наменски језик *securaDSL* подржава конфигурацију три безбедносна механизма: основну аутентификацију, стандард *JWT* и *OAuth2.0*. Овим се обезбеђује флексибилност и могућност одабира оптималног безбедносног механизма у зависности од потреба и захтева система.

5.1 Апстрактна синтакса

Апстрактна синтакса омогућава опис структуре наменског језика *securaDSL* и представљена је помоћу мета-модела (Слика 4.1). Коренски концепт апстрактне синтаксе је апликација (*Application*) и садржи податке неопходне за њену иницијализацију. Могуће је подесити параметре за повезивање са базом података (*Database*), а поред тога, могу се дефинисати ентитети (*Entity*) који одговарају табелама у бази података. Описани су ентитети који се односе на кориснике (*User*) и улоге (*Role*). При дефинисању ентитета, неопходно је навести обележја (*Attribute*) за сваки ентитет. За концепт који се односи на обраду захтева корисника (*Controller*) је могуће додати методе (*Endpoint*) које се односе на различите функционалности апликације. Навођењем инстанци улога (*RoleInstance*) које се налазе у систему и повезивањем са одређеним методама омогућена је контрола приступа. Контролер за аутентификацију (*Authentication*) може да садржи методе за регистрацију, пријаву на систем и одјаву са система.

Посебан део апстрактне синтаксе односи се на сигурносни слој (*Security*), где су подржана три безбедносна механизма: основна аутентификација (*BasicAuthentication*), стандард *JWT* (*JWT*) и *OAuth2.0* (*OAuth2*). У зависности од жељеног механизма могу се дефинисати додатни параметри описани адекватним концептима.



Слика 4.1 – Апстрактна синтакса наменског језика *securaDSL*

Сваки од наведених концепата може се додати или изоставити у зависности од потреба корисника.

5.2 Конкретна синтакса

Коришћењем радног оквира *Xtext*, на основу мета-модела генерисана је почетна верзија текстуалне конкретне синтаксе. Ова граматика прилагођена је домену како би била лако читљива и интуитивна за развојне тимове којима је *securaDSL* намењен. Почетна верзија граматике подсећа на уобичајене синтаксе за дефинисање разних конфигурационих фајлова, те таква граматика не захтева велике измене узимајући у обзир да циљну групу чине експерти у пољу безбедносне конфигурације. Пример модела *Spring* веб апликације са конфигурисаним безбедносним механизмом *OAuth2.0* приказан је на Листингу 5.1.

```
application:
  artifact: "securaDSL"
  group: "uns.ftn"
  port: 8080
  hostname: "localhost"

database:
  vendor: PostgreSQL
  url: "localhost:5432/securaDSL"
  username: "securaDSL"
  password: "securaDSL"

security:
  OAuth2.0:
    providers: [
      {
        name: "google",
        clientId: "x",
        clientSecret: "x"
      },
      {
        name: "github",
        clientId: "x",
        clientSecret: "x"
      }
    ]
```

Листинг 5.1 - Пример модела *Spring* веб апликације са конфигурисаним безбедносним механизмом *OAuth2.0*

6. ГЕНЕРИСАЊЕ *SPRING* ВЕБ АПЛИКАЦИЈА СА БЕЗБЕДНОСНОМ КОНФИГУРАЦИЈОМ

С обзиром да се *Spring* апликација може поделити у целине које су претходно описане и да постоји више подржаних безбедносних механизма, ради прегледности развијено је више генератора. Развијени су следећи генератори: генератор статичких датотека, генератор општих конфигурационих фајлова, генератор слоја који моделује податке из базе података, генератор слоја за обраду захтева корисника и генератори конфигурационих фајлова за основну аутентификацију, стандард *JWT* и стандард *OAuth2.0*.

Генератори формирају излаз на основу података из модела описаног помоћу наменског језика *securaDSL*, чиме се генерише извршиви код за *Spring* веб апликацију са конфигурисаним одабраним безбедносним механизмом. Који генератори ће генерисати излазни код зависи од тога који су концепти дефинисани у моделу. Нпр. уколико је дефинисана основна аутентификација, генератори конфигурационих фајлова за стандарде *JWT* и *OAuth2.0* неће имати излаз.

7. ЗАКЉУЧАК

Наменски језик *securaDSL* описан у овом раду садржи концепте неопходне за моделовање *Spring* апликација, а додатно подржава конфигурацију три безбедносна

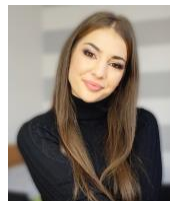
механизма: основне аутентификације, *JWT* и *OAuth2.0*. Поред наменског језика, описани су генератори који су развијени у циљу генерисања извршивог кода на основу модела креираног помоћу језика *securaDSL*. Коришћење наменског језика за моделовање *Spring* веб апликација са конфигурисаним безбедносним аспектима и пропратних генератора значајно убрзава процес развоја софтвера и умањује могућност грешке која настаје при ручном писању кода. Постојање модела апликације помаже у уочавању делова које захтевају измену, а које би иначе изазвале дораду која би утицала на велики број линија кода и самим тим довела до грешака које се тешко идентификују и отклањају. Структура саме апликације је видљивија и подложнија дискусији унутар развојног тима. Аутоматско генерисање кода повећава ефикасност и доприноси конзистентности, што унапређује квалитет саме апликације.

Остављен је простор за проширење тренутног начина за моделовање и генератора за безбедносни механизам *OAuth 2.0* увођењем регистрације на систем. Могуће је увести подршку за додатне безбедносне механизме, друге програмске језике и радне оквира увођењем нових концепата. Ово би захтевало измену апстрактне и конкретне синтаксе, али и развој нових генератора који би креирали код у различитим програмским језицима. Оваква унапређења би допринела значају и примени наменског језика *securaDSL* и развијених генератора.

ЛИТЕРАТУРА

- [1] Sujay, Vailshery, L. (2023). Most used programming languages worldwide as of 2023. [Online], Приступљено датума: 31.8.2023. <https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/>
- [2] Maple, S., & Binstock A. (2018). JVM Ecosystem Report 2018: About your Platform & Application. [Online], Приступљено датума: 31.8.2023. <https://snyk.io/blog/jvm-ecosystem-report-2018-platform-application/>
- [3] Brambilla, M., Cabot, J., & Wimmer, M. (2012). Model-driven software engineering in practice.
- [4] Spring Initializer. [Online], Приступљено датума: 31.8.2023. <https://start.spring.io/>
- [5] Šuljkanović, A., Milosavljević, B., Indić, V., Dejanović, I. (2022). Developing Microservice-Based Applications Using the Silvera Domain-Specific Language
- [6] Terzić, B., Dimitrieski, V., Kordić, S., Milosavljević, G., Luković, I. (2017). MicroBuilder: A Model-Driven Tool for the Specification of REST Microservice Architectures

Кратка биографија:



Јелена Хрњак рођена је 21. августа 1999. године у Бачкој Тополи у Републици Србији. Школске 2018/2019. године уписала се на Факултет техничких наука, смер. Дипломски рад из области Електротехнике и рачунарства одбранила је 2022. године.

**РАЗВОЈ АПЛИКАЦИЈЕ СТУДЕНТСКЕ СЛУЖБЕ ПРИМЕНОМ
ДЕЦЕНТРАЛИЗОВАНИХ И ДИСТРИБУИРАНИХ ТЕХНОЛОГИЈА И ПРОТОКОЛА
DEVELOPMENT OF STUDENT SERVICE APPLICATION USING DECENTRALIZED
AND DISTRIBUTED TECHNOLOGIES AND PROTOCOLS**

Петар Марковић, Факултет техничких наука, Нови Сад

Област – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – У овом раду је имплементирана децентрализована апликација за рад са студентском службом. У оквиру овог рада је описана целокупна архитектура апликације коју чине паметни уговор постављен на Ethereum мрежи, сервер репрезентован као Express апликација и клијентска React апликација. Такође, објашњене су делатности свих горепоменутих компоненти и начин сарадње.

Кључне речи: блокчејн, Ethereum, паметни уговор, IPFS

Abstract – This paper presents an implementation for a decentralized student services app. The architecture of this application consists of three components, a smart contract that is deployed on an Ethereum testnet, a server layer represented as an Express application and a React client application. Furthermore, this paper provides an in depth perspective on the responsibilities of each said component and how these components interact.

Keywords: blockchain, Ethereum, smart contract, IPFS,

1. УВОД

Централизовани систем се може дефинисати као структура где постоји једна управљачка компонента која има потпуну контролу. Оваква структура је представљала норму због своје лакоће развоја и приступачности одржавања, међутим тешко је занемарити и њене недостатке, конкретно лоше перформансе за удаљене кориснике и склоност грешкама. Како би се ублажиле ове ниспојаве, користе се алтернативе у виду децентрализованих и дистрибуираних система.

За разлику од ауторитативног централизованог система, децентрализовани систем има више власника при чему сваки складишти копију ресурса којима корисник има приступ. Дистрибуиран систем, иако сличан децентрализованом, разликује се по томе што сваки корисник доноси своје одлуке и коначно понашање система представља збир одлука ових корисника.

Овај рад описује једну потпуно децентрализовану апликацију студентске службе.

НАПОМЕНА:

Овај рад произтекао је из мастер рада чији ментор је био др Горан Сладић, ред. проф.

Сама апликација је вишеслојна и сваки слој имплементира адекватне дистрибуиране и децентрализоване технологије које су детаљније описане у наставку рада.

2. ТЕОРИЈСКЕ ОСНОВЕ

2.1. Блокчејн

Блокчејн се неретко дефинише као јавно доступна дистрибуирана главна књига, која је подељена између чворова рачунарске мреже [1]. Његова форма је облика ланац блокова који по структури подсећа на једноструку спрегнуту листу. Као технологија је први пут описана 1991. године. Оригинална идеја је била да временски означити дигиталне документе и да самим тим смањити ризик антидатирања или злонамерног манипулисања документима [2], али је њена прва практична употреба била 2009. у виду платформе познате као Bitcoin [3].

Идеја блокчејна јесте да омогући чување и дистрибуцију информација, али не и њену измену. Блок представља градивну јединицу блокчејна и сваки блок садржи одређени скуп података, сопствени хеш и хеш претходног блока. Неки од најбитнијих података који се налазе у сваком блоку су трансакције, временска одредница блока, величина блока и енкриптовани број (*nonce*).

2.2. Консензус алгоритми

Консензус алгоритам је процедура уз помоћ које сви учесници блокчејн мреже могу да дођу до заједничког закључка о стању самог блокчејна у реалном времену. Овај механизам омогућава мрежи да постигне поузданост тако што формира ниво поверења између различитих чворова док истовремено осигурава безбедност у окружењу. Суштински, консензус алгоритам обезбеђује да сваки нови блок који се дода на блокчејн буде једина права истинита верзија која је одобрена од стране чворова блокчејна.

2.3. Ethereum

Ethereum је децентрализована глобална платформа заснована на блокчејн технологији која омогућава свим својим корисницима да креирају безбедну дигиталну технологију. Ethereum такође има своју валуту, познату као Ether (ETH) [4], као и програмски језик Solidity [5].

Паметни уговор је обичан програм написан у Solidity програмском језику који се покреће на Ethereum-овом блокчејну. Он садржи код и податке који су

ускладиштени на посебној адреси на самом блокчејну. Сви учесници могу да обављају интеракције са паметним уговором, али ниједан од њих нема контролу над њим. Ове интеракције се постижу покретањем трансакција које затим извршавају функције дефинисане у паметном уговору. Свака трансакција захтева и одређену количину "гаса", што је вид додатне таксе – наплате за извршавање кода у паметном уговору.

Темељ читаве оперативне структуре *Ethereum*-а јесте *Ethereum Virtual Machine (EVM)*. *EVM* је задужен за извршавање и постављање паметних уговора уз помоћ њене мреже дистрибуираних јавних чворова. Како би се постигао консензус, сваки *Ethereum*-ов чвор ради на *EVM*. Виртуелна машина је потпуно изолована и самим тим код унутар *EVM* нема приступ мрежи, фајл системима или другим процесима [6].

2.4. Interplanetary File System

Interplanetary File System (IPFS) се може дефинисати као протокол дистрибуираног складишта датотека који омогућава рачунарима да складиште и услужују датотеке, али и фолдере, као део једне велике *Peer-to-Peer (P2P)* мреже. Свакој датотеци која је отпремљена на *IPFS* окружењу је могуће приступити и преузети је преко било којег уређаја који је повезан на *IPFS* [7].

Важан концепт *Interplanetary File System*-а јесте такозвани *Content Identifier (CID)*, односно идентификатор садржаја. Свакој датотеци која се отпреми на овом протоколу се додели јединствена адреса која је изведена из хеша њеног садржаја. *CID* који се додели ресурсу је јединствен и непромењив (имутабилан), што значи да ни једно треће лице не може да измени његову првобитну вредност. Сами подаци се заправо складиште и добављају преко његовог хешираног идентификатора садржаја. Ово значи да је *IPFS* мрежа заправо систем базиран на адресирање садржајем што је у потпуној супротности са типичном централизованом мрежом, где корисници зависе од одређеног поузданог ауторитета да рукује подацима употребом неког вида система базираног на локацији као што је *Uniform Resource Locator (URL)*.

3. МОДЕЛ СИСТЕМА

У склопу ове децентрализоване апликације постоје три примарна ентитета: *User*, *Exam* и *ExamResult*. *User* представља корисника ове апликације. Сваки корисник се јединствено идентификује путем адресе његовог новчаника. Поред адресе, за корисника се још складишти име и презиме, електронска пошта и рола. Рола може бити професор или студент, и сходно овој роли, он има могућност и приступ различитим функционалностима, конкретно професор може да оцени пријављени испит који он држи, док студент има право да пријави надолazeћи испит. Заједничка функционалност за обе улоге јесте то што имају приступ листи испита и пријава, односно професор може да види све своје оцењене и неоцењене испите, док студент има преглед свих својих оцењених испита и испита које може да пријави.

Следећи ентитет јесте *Exam*, односно испит. У склопу овог ентитета чува се јединствени идентификатор испита, назив предмета повезаног са тим испитом,

адреса *Ethereum* дигиталног новчаника професора који је надлежан за тај испит и датум и време одржавања испита.

Последњи ентитет јесте *ExamResult*, тачније резултат или пријава испита. Од атрибута он пре свега садржи јединствени идентификатор пријаве, адресу *Ethereum* дигиталног новчаника студента који је поднео пријаву, јединствени идентификатор асоцираног испита за који је направљена пријава и додељену оцену. Када се првобитно пријави испит, његова оцена је 0, док након оцењивања испита ова вредност може бити између 5 и 10.

4. ИМПЛЕМЕНТАЦИЈА СИСТЕМА

5.1. Иницијална поставка

Након поставке новчаника, наредни корак је креирање налога на *Alchemy*-у [8]. *Alchemy* представља блокчејн платформу за програмере која пружа низ алата за развој. Ова платформа замењује стандардне локалне чворове са чворовима који имају бржу и скалабилнију децентрализовану архитектуру. За потребе овог пројекта, креиран је *Alchemy* пројекат за *Ethereum Sepolia* тест мрежу. Након креирања апликације, неопходно је извући *Application Programming Interface (API)* кључ како би се успешно аутентификовао корисников чвор и омогућило слање захтева ка блокчејну. Последњи важни предуслов пре почетка развоја апликације јесте упознавање са *Etherscan* платформом [9]. *Etherscan* се дефинише као блокчејн претраживач за *Ethereum*. Преко ове платформе могуће је пронаћи важне податке сачуване на ланцу. Поред тога, *Etherscan* се може користити и за верификацију паметних уговора. Ова верификација је нужна, јер преко ње се обезбеђује поверење и континуитет у извршењу пословних процеса паметног уговора. Како би се програмеру омогућило да верификује свој паметни уговор приликом поставке на блокчејн, неопходно је направити налог на *Etherscan* платформи и затим креирати свој *API* кључ.

5.2. Серверски део апликације

Solidity као програмски језик је Тјуринг комплетан. Самим тим, фактички је могуће да се све информације и операције које би биле од значаја за једну студентску службу ускладиште на паметном уговору, и да се затим њима управља по потреби. Међутим, треба се узети у обзир чињеница да *Ethereum* имплементира *Proof of Work* консензус механизам и да свака трансакција захтева одређену количину таксе која мора да се исплати рударима како би она била верификована. Ова количина таксе расте сразмерно са тежином рачунарске операције коју та трансакција захтева, али и са величином и комплексношћу података који се складиште на самом паметном уговору. Осим тога, *Solidity* ради на нижем нивоу апстракције у односу на друге често коришћене програмске језике што негативно утиче на читљивост самог кода и лакоћу писања истог.

Због свега наведеног, *Solidity* паметни уговор је складиштио минималну потребну количину података. Овај уговор је попуњен подацима приликом његове поставке на *Ethereum* блокчејн мрежу и имплементирао је функционалности регистрација и оцењивања

испита на што је могуће рачунски незахтеван начин. Сам паметни уговор служи и као сервис за плаћање, пошто је приликом пријаве испита неопходно приложити одређену количину средстава која се након успешне трансакције чувају у уговору.

Наравно, било је неопходно имплементирати сервис који би имао детаљне податке о свим ентитетима једног студентског система. Приликом избора и развоја сервера, од круцијалног значаја је било да се обезбеди максимално децентрализовано окружење. Из овог разлога је одлучено да се са подацима рукује у склопу *OrbitDB* базе података. *OrbitDB* представља дистрибуирано *P2P* складиште података реализовано на *IPFS*-у које је дизајнирано за рад са децентрализованим апликацијама. Како *OrbitDB* пружа *JavaScript API*, одлучено је да сервис буде имплементиран у том језику.

5.2.1. Паметни уговор

За потребе развоја блокчејн стране рада, коришћен је радни оквир *Foundry* [10]. *Foundry* је окружење за развој децентрализованих апликација засновано на *Ethereum* блокчејну. Јединствена карактеристика овог радног оквира јесте његов скуп уграђених алата командне линије.

У склопу ове платформе развијен је *Solidity* паметни уговор назван *StudentService*. На самом почетку паметног уговора дефинишемо скуп варијабли стања. Свака од дефинисаних променљивих има приватни модификатор приступа и самим тим видљива је само у оквиру уговора. Прва варијабла *EXAM_FEE* је константа типа неозначени број од 256 бита и она представља цену пријаве једног испита. Наредна варијабла *exams* представља низ идентификатора свих испита. Последње три променљиве су типа *mapping*. По структури *mapping* је налик хеш табели и користи се за складиштење парова кључ-вредност. Прва од ове три променљиве названа *users* представља скуп свих корисника у систему. Кључ је адреса новчаника корисника, а вредност је типа *boolean* и верификује да ли корисник са том адресом постоји у студентској служби. Последње две варијабле *studentExamResults* и *professorExamResults* складиште информације о резултатима испита. Тачније, прва варијабла као кључ чува адресу новчаника студента, а као вредност чува низ идентификатора свих испита које је пријавио. Кључ друге варијабле је адреса професора, а вредност представља низ идентификатора пријављених испита за које је одговоран.

У наставку програма налазе се компоненте за обраду догађаја (енгл. *event*). Уз помоћ *event* компоненти, могуће је успоставити комуникацију са клијентском апликацијом и обавестити је да је дошло до промене на блокчејну након одређене трансакције. Најчешће им се придодату параметри који представљају важне информације о трансакцији. У склопу ове апликације постоје 2 догађаја: *examRegistered* и *examGraded*.

Solidity садржи конструкторе. У оквиру ове посебне функције може се писати код и ова функција ће се извршити само једном након креирања уговора. Конструктор се користио како би се поставиле иницијалне вредности у оквиру горе наведених варијабли стања.

Последњи део паметног уговора састоји се из функција. Функције су подељене у три групе: *view* функције коришћене за приступ вредностима варијабле стања, *pure* функције које представљају помоћне функције валидације и главне функције у склопу којих су имплементиране функционалности.

У паметном уговору постоје две главне функције: *registerExam* и *gradeExam*. Прослеђени параметри ових функција се прво валидирају употребом горепомнутих *view* и *pure* функција. Уколико су параметри исправни, функција наставља са извршавањем и на крају се оклада адекватан догађај.

У *Foundry* окружењу је могуће написати и тестове. Ови тестови спадају у један вид паметног уговора где је у самој декларацији уговора неопходно додати *is Test* како би скуп алата знао да ову датотеку третира као тестну. Узимајући у обзир релативну једноставност самог паметног уговора, за потребе овог пројекта написани су базични тестови који тестирају *view* и *pure* функције.

Након финализације процеса развоја и тестирања, може се прећи на поставку и верификацију уговора. Ово се обавља путем уноса инструкција у командну линију коју *Foundry* и конкретно њен пакет *forge* подржавају. Како би се поставио уговор на мрежу, неопходно је унети команду *forge create*. Поред ове команде неопходно је и проследити низ параметара, конкретно приватни кључ новчаника који ће извршити поставку уговора, *HTTPS* крајњу тачку са *API* кључем дефинисаним у *Alchemy* платформи, параметре конструктора и *API* кључ *Etherscan* налога.

5.2.2. Серверски слој

За развој серверској слоја коришћен је *Express.js*, популарни радни оквир за развој веб апликација у *JavaScript* језику. Овај радни оквир је првенствено изабран пошто за *IPFS* и децентрализовану *OrbitDB* базу података постоји обимно документована *JavaScript* имплементација

Приликом покретања апликације се подиже и *OrbitDB* база података. Да би се база података конфигурирала, прво је неопходно направити инстанцу *IPFS*-а коју ће *Orbit* користити као комуникациони слој. Приликом дефинисања *IPFS* чвора, неопходно је навести и конкретну локацију, односно репозиторијум, где ће подаци овог чвора бити ускладиштени.

У склопу овог пројекта, компоненте су модуларно подељене у зависности од њихове одговорности. *model* директоријум садржи *JavaScript* класе које репрезентују шеме ентитета који се складиште у табелама базе података. Укупно постоје три модела и то: *Exam*, односно испит, *ExamResult*, односно резултат или пријава испита и *User*, односно корисник. За сваки од ових модела постоји одвојена база података.

За приступ и манипулацију подацима користе се датотеке у *dao* директоријуму. Сам *dao* је акроним за *Data Access Object*, слој веб апликације задужен за рад са базом података. Како бисмо могли да извршавамо операције над *OrbitDB*, потребно је сваки пут програмски отворити базу података и дефинисати табелу којој желимо да приступимо. Када се изврше неопходне операције над базом података, добра

пракса је да се она одмах затвори како би се ослободили ресурси.

Следећа компонента архитектуре која ће бити описана јесте сервисни слој. Сврха овог слоја јесте да енкапсулира пословну логику саме апликације и да употребом функционалности из претходно описаног *dao* директоријума руководи ентитетима који су сачувани у децентрализованом бази података.

Последња значајна компонента серверске архитектуре јесте *controllers*. Овај слој садржи низ контролера чији примарни циљ јесте да буде медиатор између крајњег корисника и серверске апликације и омогућив корисницима да приступе и управљају функционалностима које су раније описане.

5.2. Кориснички (*frontend*) део апликације

За развој корисничког дела апликације, коришћена је једна од најпознатијих *JavaScript* библиотека: *React*. У комбинацији са *React*-ом, употребљена је и једна популарна библиотека за развој и дизајнирање корисничког интерфејса звана *Chakra UI*.

Нажалост, *React* као библиотека нема уграђену подршку за рад са блокчејном и паметним уговорима, због чега је било неопходно инсталирати зависности *react-moralis* и *web3uikit*. *react-moralis* пружа функционалности уз помоћу којих могу да се позивају функције постављеног паметног уговора и да се добава стање повезаног новчаника са страницом, док *web3uikit* садржи стилизоване и лагане компоненте корисничког интерфејса за развој *Web 3.0* апликација. Како би се позвале функције постављеног паметног уговора из прошлог одељка, користи се *useWeb3Contract* функционалност коју пружа *react-moralis* библиотека. За позивање ове функционалности, неопходно је дефинисати жељени назив функције у оквиру компоненте и проследити *ABI* паметног уговора, адресу постављеног паметног уговора, назив функције који се позива из паметног уговора, списак параметара паметног уговора и потенцијално количину *ETH* која се шаље паметном уговору.

Уколико позивом ове функције добијемо позитиван одговор од стране паметног уговора, наредни корак јесте да пошаљемо адекватан захтев *Express* серверу како бисмо управљали са подацима у њиховој целости. Ради олакшане комуникације са самим сервером, коришћен је популарни *JavaScript HTTP* клијент *axios*.

6. ЗАКЉУЧАК

У овом раду имплементирана је децентрализована апликација студентске службе. Како би се олакшао развој, тестирање, компајлирање и постављање паметног уговора који представља основу апликације, коришћена је *Foundry* развојна платформа. Узимајући у обзир колико може бити скупо складиштење велике количине података и вршење комплексних рачунских операција у *Solidity* уговору, конкретни паметни уговор је складиштио само идентификаторе ентитета од значаја и имплементирао плаћање криптовалутама, а детаљнији садржај и пословна логика је била имплементирана у одвојеном *Express* серверу. Овај сервер је складиштио и руковао подацима на децентрализован начин употребом *OrbitDB*, база

података изграђена на *IPFS* протоколу. Како би се постигла комуникација са овим паметним уговором који је постављен на *Ethereum* блокчејну, на клијентској страни је било неопходно подесити *Moralis* платформу.

Правци даљег развоја апликације би били фокусирани на креирање компатабилне мобилне апликације. Поред тога, потенцијално би се могло додати још функционалности или проширити постојеће (отказивање испита, овера семестра, одбрана дипломског рада итд.). Такође, са становишта оптимизације, сам паметни уговор би се могао рефакторисати тако да извршавање операција над њим захтева још мање гаса.

На самом серверу се могу постићи унапређења, пре свега имплементацијом неког вида контроле приступа која је базирана на ролама.

Поред тога, овај сервер је написан у обичном *JavaScript* језику и самим тим је динамички типизиран језик, што може да има негативан утицај на перформансе и безбедност. Решење би било надоградити постојећи пројекат употребом *TypeScript*-а, или мигрирати базу кода сервера на статички типизиран програмски језик.

7. ЛИТЕРАТУРА

- [1] <https://www.investopedia.com/terms/b/blockchain.asp> (приступљено у септембру 2023.)
- [2] Haber, Stuart, and W. Scott Stornetta. How to timestamp a digital document. Springer Berlin Heidelberg, 1991.
- [3] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." Decentralized business review (2008).
- [4] <https://ethereum.org/en/eth/> (приступљено у септембру 2023.)
- [5] <https://soliditylang.org/> (приступљено у септембру 2023.)
- [6] <https://ethereum.org/en/developers/docs/evm/> (приступљено у септембру 2023.)
- [7] <https://docs.ipfs.tech/concepts/what-is-ipfs/> (приступљено у септембру 2023.)
- [8] <https://www.alchemy.com/> (приступљено у септембру 2023.)
- [9] <https://etherscan.io/> (приступљено у септембру 2023.)
- [10] <https://book.getfoundry.sh/> (приступљено у септембру 2023.)

Кратка биографија:



Петар Марковић рођен је у Нишу 1999. године. Мастер рад на Факултету техничких наука из области Софтверско инжењерство и информационе технологије – Електронско пословање одбранио је 2023. године

контакт: petarns99@yahoo.com

СИСТЕМ ЗА ПРЕПОРУКУ ЗАСНОВАН НА ГРАФОВСКИМ НЕУРОНСКИМ МРЕЖАМА**RECOMMENDER SYSTEM BASED ON GRAPH NEURAL NETWORKS**Марко Његомир, Јелена Сливка, *Факултет техничких наука, Нови Сад***Област – РАЧУНАРСТВО И АУТОМАТИКА**

Кратак садржај – Главна идеја рада била је да се направе модели машинског учења утемељени на графовским неуронским мрежама који би вршили препоруке производа корисницима. Од доступних података о куповним навикама корисника направљен је хетерогени бипартитан граф који садржи чворове корисника и производа, а везе у њему представљају интеракције које су корисници имали са производима. Модели су тестирани на Brazilian E-commerce скупу података. Први модел је користио један енкодер са слојевима графовске неуронске мреже за енковање чворова, и један декодер ивица, чија је улога била да прави предикцију оцене коју би корисник дао производу. Други модел је користио посебне енкодере за кориснике и за производе, а такође је имао један декодер ивица. Модел са два енкодера се показао као бољи када се посматра RMSE метрика, а при том је имао и стабилније тренирање.

Кључне ријечи: графовске неуронске мреже, системи за препоруку, GraphSAGE.

Abstract – This study aimed to create a machine-learning model based on graph neural networks that can provide item recommendations for users of an e-commerce system. For model training, the data had to be converted to a heterogenous bipartite graph, where users and products were represented as nodes, while edges represented the interactions between them. Models were tested on the Brazilian E-commerce dataset. The first model used one encoder containing graph neural network layers for encoding nodes and one decoder for edges, which aimed to make rating predictions that the user would assign to the product. The second model used separate encoders for users and products and had an edge decoder. The model with two encoders achieved better RMSE than the single encoder model and had a more stable training process.

Keywords: graph neural networks, recommender systems, GraphSAGE.

1. УВОД

У системима електронске трговине, системи за препоруку су веома битни. У понуди често постоји веома велики број производа, и битно је препоручити

НАПОМЕНА:

Овај рад произтекао је из мастер рада чији ментор је била др Јелена Сливка, ванр. проф.

подкуп тих производа који ће повећати задовољство корисника, а тиме и вероватноћу да ће доћи до куповине [1].

За препоруке су се до сада успешно примењивале једноставније технике као што су *Content-based filtering* [2] или мало напреднији приступ са факторизацијом матрица [3]. Предности оваквих система је што су брзи за извршавање, а мана је што не могу да користе додатне информације о интеракцији свих корисника у систему са производима.

Ове мане је делимично било могуће надоместити коришћењем ансамбла једноставнијих модела [2]. У скорије време је постало јасно да се велик број информација о куповним навикама корисника може представити у структури графа, па је на њих онда могуће применити и графовске неуронске мреже [4]. Ове архитектуре су у стању да искористе обиље информација о корисницима, производима и њиховим интеракцијама, и могу се посматрати као модели који комбинују предности *Content-based filtering* приступа са колаборативним филтрирањем, код кога се узимају и информације о интеракцијама других корисника у обзир, а при том су у стању да избегну проблеме које су претходно поменути приступи имали [2][3][4].

У овом раду су коришћени GraphSAGE [5] слојеви неуронске мреже да би се изградили једно-енкодерски и дво-енкодерски модели. Ови модели су тренирани и евалуирани на *OList Brazilian E-commerce* скупу података [6]. На основу овог скупа података је направљен хетерогени бипартитан граф корисника, производа и њихових интеракција. Модели из овог рада се тренирају да предвиде коју оцenu би корисник дао производима, и затим се ти производи могу сортирати да би се добила коначна листа препорука.

У овом раду ће прво бити приказана претходна решења за овај проблем. Након тога ће бити описани детаљи о имплементацији решења. Затим ће се анализирати добијени резултати. У последњем поглављу ће бити представљен закључак рада.

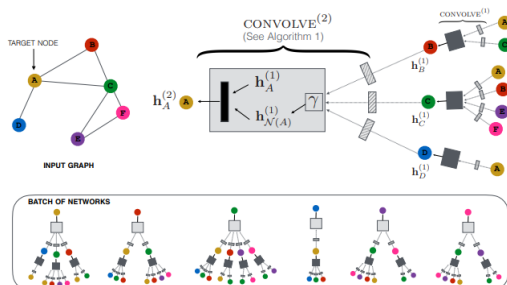
2. ПРЕТХОДНА РЕШЕЊА

Један од основних приступа у системима за препоруку је *Content-based filtering* [2], у коме се корисницима препоручују производи који су слични онима са којима су раније имали интеракцију. Овај приступ је брз због своје једноставности, али не узима у обзир информације о потрошачким навикама свих корисника у систему. Још једна мана овог приступа је што корисницима препоручује ствари сличне онима

које су већ купили, што не мора значити да желе да купе одмах поново сличан производ.

У раду са факторизацијом матрица [3], аутори су представили својства производа и својства корисника помоћу матрица, а коначно рангирање производа за сваког корисника се добија множењем матрица. Овакав приступ може имати проблема када корисници нису имали довољно интеракција са производима. У раду [3] су истраживани различити алгоритми у циљу да се овај проблем реши, као и да се побољша способност скалирања оваквих алгоритама.

Аутори рада [7] су покушали да реше недостатке претходно поменутих приступа коришћењем конволутивних слојева графовских неуронских мрежа (енг. *Graph Convolutional Neural Networks* - GCN). У раду је направљен систем за препоруку за веб-платформе које раде са великим количинама података. Експерименти су вршени на подацима са *Pinterest* сајта. Сви подаци у претворени у форму графа, и затим је направљен *PinSAGE* модел који је приказан на слици 1, и који има два GCN слоја које је ефикасно могао да обради велику количину података, тако што је био у стању да експлоатише локалну структуру графова и употребљава паралелизацију. Овакав модел је у стању да користи и информације о свим корисницима и производима у графу, и да умањи проблем када неки корисник нема много интеракције са производима, због тога што граф садржи обиље контекстних информација о сличним корисницима и производима. Због тога је овај модел постигао воље резултате од претходно поменутих стандардних метода за препоруку.



Слика 1. Двослојни *PinSAGE* модел [7].

У раду [8] је представљен модел за колаборативно филтрирање помоћу графовских неуронских мрежа (*Neural Graph Collaborative Filtering* - NGCF). У овом раду су интеракције између корисника и производа моделоване помоћу бипартитног графа. Модел користи GCN слојеве, и у стању је да прави персонализоване препоруке а да при том искористи и информације о куповним навикама других корисника. Овај модел је показао значајно побољшање у односу на класично колаборативно филтрирање и у односу на друге модерне приступе са којима је поређен у раду.

Аутори рада [9] су демонстрирали значај увођења мета-путања у хетерогеним графовима код система за препоруку. Мета-путања је веза између два чвора која нису директно повезана у оригиналном графу, али је таква веза додата јер их неки други тип чвора повезује. У раду је коришћена хетерогена графуска

неуронска мрежа са пажњом (*Heterogenous Graph Attention Network* – HGAT). Додавање нових веза у виду мета-путања и коришћења механизма пажње који додељује значај свим везама је довело до побољшања у препорукама у односу на друге моделе са којим су вршена поређења.

Овај рад показује значај истовременог повећања комплексности структуре података и флексибилности модела за постизање бољих резултата.

3. МЕТОД

У наредним поглављима су описан је скуп података кориштен у раду, начин на који је извршено претпроцесирање и архитектура испробаних модела за препоруке.

3.1. Скуп података

Скуп података који је коришћен у раду је *OList Brazilian E-commerce* [6], и он се састоји од више табела са информацијама о корисницима, производима, поруџбинама, и рецензијама са оценама.

У скупу података постоји око 150 000 поруџбина, и од тога 650 поруџбина нема оцену. Подаци у скупу података су анонимизовани, тако да није могуће препознати кориснике, али ни производе.

За кориснике су доступне географске информације. Поруџбине садрже информације о цени и датуму поручивања, а такође постоје и информације о оценама поруџбина.

Производи садрже опште информације о производу, као што су димензије, тежина, цена и категорија производа.

3.2. Претпроцесирање података

За рад са графовским неуронским мрежама, од информација из скупа података је направљен хетерогени бипартитни граф. У том графу чворови представљају кориснике и производе, док ивице представљају интеракцију производа са корисницима.

Географске информације о корисницима нису значајне у свом изворном облику, па су подаци о корисницима додатно обогаћени прикупљањем са интернета. Тако су додате информације о величини и густини градова корисника, као и бруто регионалном производу.

Ивице у графу садрже информације о оцени коју је корисник дао производу, као и количини производа који су купљени. За експеримент са другим типом модела су додате и мета-путање између производа у графу.

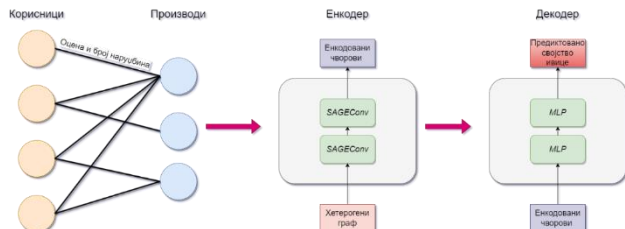
Све информације о поруџбинама које нису биле потпуне јер није постојала информација о оцени су одбачене, јер то је представљало само 0.43% укупног броја поруџбина.

Просечан број интеракција које су корисници имали са производима је само 1.21, што је последица реалне ситуације са много пасивних корисника, и мањим бројем веома активних купаца. Оваква расподела је добар тест за могућност графовских неуронских мрежа да се изборе са тим проблемом.

3.3. Архитектура решења

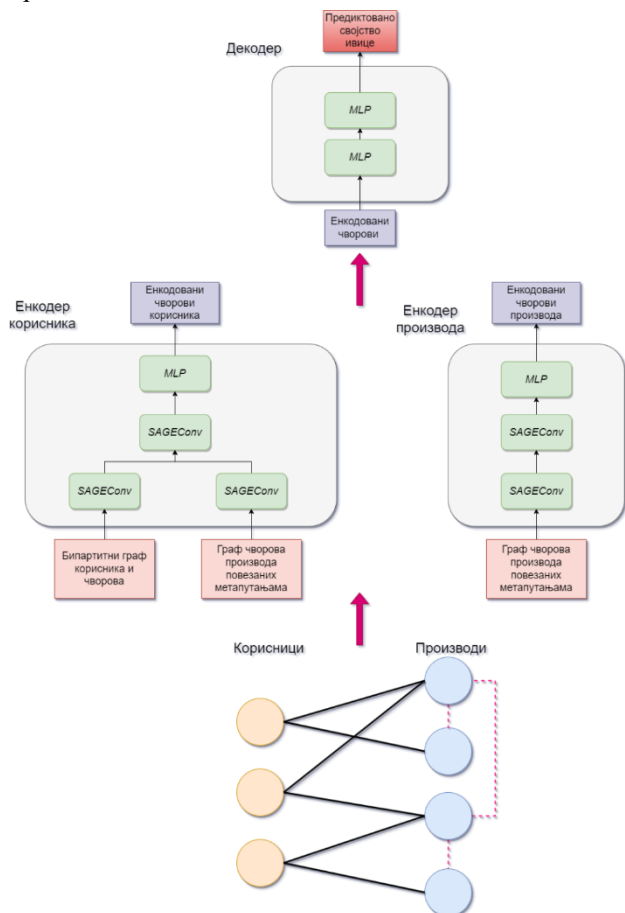
За прављење препорука су коришћена два модела графовских неуронских мрежа, са једним и са два енкодера. Обе мреже имају декодер ивица који има задатак да изврши предикцију оценоу коју би корисник дао неком производу.

Једно-енкодерски модел се састоји од два SAGEConv слоја. Енкодер прима информације о производима и корисницима у виду хетерогеног бипартитног графа. Резултати енкодера се прослеђују декодеру ивица као што је приказано на слици 2.



Слика 2. Једно-енкодерски модел.

Дво-енкодерски модел има посебне енкодере за производе и ивице. У оба енкодера се прослеђује хетерогени граф обogaћен додатним информацијама у виду мета-путања. Посебно енкодирани производи и корисници се прослеђују декодеру ивица као што је приказано на слици 3.



Слика 3. Дво-енкодерски модел.

Оба модела се тренирају и евалуирају трансдуктивно, што значи да су сви чворови доступни приликом тренирања и евалуације, док су ивице подељене на три скупа, тренинг скуп који чини 80% ивица, и

валидациони и тест скуп од који сваки има 10% ивица.

Због тога што је скуп података небалансиран, јер је најзаступљенија оцена 3 на скали од 1 до 5, коришћена је отежињена средња квадратна грешка (*weighted mean squared error* - WMSE) као функција губитка приликом тренирања.

Системи за препоруку се често евалуирају помоћу $\text{top}@k$ метрике, која мери однос препоручених производа и првих k производа сортираних по томе како их је корисник оценио. У овом скупу података, просечан број оцена је 1.21, па је због тога коришћена средња квадратна грешка (*Mean Squared Error* - MSE) као метрика за евалуацију.

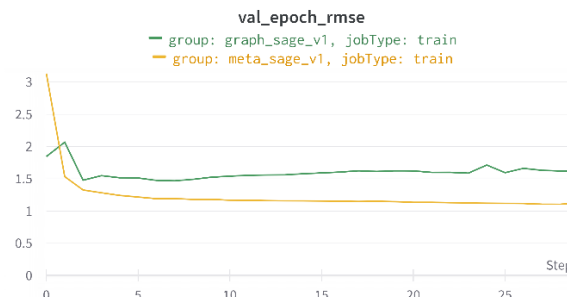
4. РЕЗУЛТАТИ И ДИСКУСИЈА

Оба модела су тренирана 300 епоха. Након сваке епохе се вршила и валидација, и чуван је модел који је добио најмању RMSE вредност. Приликом тренирања оба модела коришћен је ADAM (*Adaptive Momentum*) оптимизатор са брзином учења 0.01. У табели 1 су дати добијени резултати за оба модела на тест скупу.

Модел	RMSE
Једно-енкодерски модел	1.43
Дво-енкодерски модел	1.118

Табела 1. Перформансе модела графовских неуронских мрежа.

Из резултата се јасно види да је дво-енкодерски модел постигао боље резултате од једно-енкодерског модела. Ово је последица тога што је дво-енкодерски модел имао богатији граф података на располагању због додавања мета-путања између производа, као и чињенице да је имао већу флексибилност што му је и омогућило да створи боље репрезентације корисника и производа. При том је дво-енкодерски модел такође имао и стабилније тренирање. Са слике 4 се види да RMSE крива стабилно опада за дво-енкодерски модел током целог тренирања, док крива за једно-енкодерски модел показује одређену нестабилност због наглих скокова. Осим тога, ове две криве показују и да је дво-енкодерски модел мање склон преприлагођавању од једно-енкодерског модела, што се види у благом расту криве једно-енкодерског модела након што је достигао најбољи резултат већ после пар епоха.



Слика 4. RMSE крива током тренирања за једно и дво-енкодерски модел.

5. ЗАКЉУЧАК

У овом раду су представљени модели за препоруку утемељени на графовским неуронским мрежама. Идеја је била да се за проблем препоруке искористе напредне архитектуре графовских неуронских мрежа које су у стању да искористе информације о корисницима, производима и о њиховим интеракцијама. Овај приступ се због тога може посматрати као комбинација колаборативног филтрирања, које узима и куповне навике других корисника у обзир, и *content based* модела, који узимају у обзир сличност производа са производима са којима је корисник раније имао интеракцију.

За потребе ових модела, од табела из *OList Brazilian E-commerce* [6] скупа података табела је направљен хетерогени бипартитни граф корисника и производа. Везе у овом графу представљају интеракцију корисника са производима, и носе информацију о томе како је корисник оценио производ, и колику количину је претходно купио. Географске информације о корисницима су искоришћене да се прикупе додатни подаци са интернета за густину насељености места из кога корисник долази, као и информације о бруто-регионалном производу, због тога што ови додатни подаци могу носити информацију о куповним навикама корисника.

У раду су направљена два модела графовских неуронских мрежа. Једно-енкодерски модел је изграђен од GraphSAGE [5] слојева графовске неуронске мреже, и он ради са хетерогеним бипартитним графом корисника, производа и њихових интеракција. Декодеру ивица се затим пропушта излаз енкодера који врши предикцију коју оцену би корисник дао неком производу. Дво-енкодерски модел користи посебне енкодере изграђене од GraphSAGE слојева за кориснике и производе, и резултате из оба енкодера прослеђује декодеру ивица који је врши предикцију оцено као и претходни модел. У дво-енкодерски модел се шаље хетерогени граф који је додатно обогаћен метапутањима између производа са којима је неки корисник претходно имао интеракцију.

Перформансе оба модела се мере помоћу RMSE метрике, и дво-енкодерски резултат је постигао бољу вредност од 1.118 RMSE у односу на 1.48 RMSE коју је постигао једно-енкодерски модел. Ово је последица богатијег графа података који дво-енкодерски модел може да искористи због своје веће флексибилности.

Предмет даљих истраживања би могло бити убацивање додатних временских димензија за куповину, због тога што куповне навике корисника често одликује и периодичност. Ове информације се лако могу убацивати као додатна обележја у ивице графа, и графовске неуронске мреже су у стању да искористе све те додатне информације.

Још један правац истраживања би могло бити додавање додатних мета-путања између корисника, и додавање других типова мета-путања. Овакав приступ ди довео до веома повезаног графа, и због тога би се могли користити слојеви графовске неуронске мреже

са пажњом, који би били у стању да процене колико пажње треба да се обрати на који тип везе.

6. ЛИТЕРАТУРА

- [1] Wu, L.L., Joun, Y.J. and Lee, J., 2013, January. Recommendation systems and consumer satisfaction online: moderating effects of consumer product awareness. In *2013 46th Hawaii International Conference on System Sciences* (pp. 2753-2762). IEEE.
- [2] Koren, Y., 2009. The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 81(2009), pp.1-10.
- [3] Koren, Y., Bell, R. and Volinsky, C., 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8), pp.30-37.
- [4] Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M. and Monfardini, G., 2008. The graph neural network model. *IEEE transactions on neural networks*, 20(1), pp.61-80.
- [5] Xu, K., Hu, W., Leskovec, J. and Jegelka, S., 2018. How powerful are graph neural networks?. *arXiv preprint arXiv:1810.00826*.
- [6] Olist and André Sionek, 2018, Brazilian E-Commerce Public Dataset by Olist Data set. Kaggle. <https://doi.org/10.34740/KAGGLE/DSV/195341>.
- [7] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L. and Leskovec, J., 2018, July. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 974-983).
- [8] Wang, X., He, X., Wang, M., Feng, F. and Chua, T.S., 2019, July. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval* (pp. 165-174).
- [9] Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P. and Yu, P.S., 2019, May. Heterogeneous graph attention network. In *The world wide web conference* (pp. 2022-2032).

Кратка биографија:



Марко Његомир рођен је 1991. године у Апатину, Република Србија. Основне академске студије завршио је 2022. године на Факултету техничких наука, на ком брани и мастер рад 2023. године из области Електротехнике и рачунарства – Софтверско инжењерство и информационе технологије.
контакт:
marko.njegomir@uns.ac.rs

DEVOPS PRISTUP RAZVOJU APLIKACIJE SA INTEGRACIJOM GENERATIVNOG JEZIČKOG MODELA**DEVOPS APPROACH TO APPLICATION DEVELOPMENT WITH INTEGRATION OF A GENERATIVE LANGUAGE MODEL (GPT)**

Miloš Popović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – PRIMENJENE RAČUNARSKE NAUKE I INFORMATIKA

Kratak sadržaj – U ovom radu opisan je razvoj i praćenje (monitoring) aplikacije po DevOps principima. Aplikacija služi za generisanje plana objava i ideja na društvenim mrežama uz pomoć GPT generativnog jezičkog modela.

Ključne reči: DevOps, monitoring, generativno jezički modeli.

Abstract – This paper describes the development and monitoring of an application following DevOps principles. The application is used for generating social media posting schedules and ideas with the assistance of the GPT generative language model.

Keywords: DevOps, monitoring, generative language models

1. UVOD

DevOps pristup se sve više prepoznaje kao ključni faktor za brz i pouzdan razvoj, implementaciju i održavanje aplikacija. Sa druge strane, generativni jezički modeli, kao što je "Chat GPT", pružaju mogućnost za automatizaciju procesa komunikacije i kreacije sadržaja. Integracija ovih generativnih jezičkih modela u aplikacije poboljšava korisničko iskustvo pošto na jednostavan način možemo omogućiti da aplikacija vrši kompleksnije provere korisničkog unosa, kao i da omogućimo generisanje sadržaja personalizovanog za svakog korisnika, bez potrebe za ručno programiranim kompleksnim pravilima i logikom.

Za potrebe rada razvijana je aplikacija koja korisnicima, koji u ovoj aplikaciju predstavljaju vlasnici biznis naloga na društvenim mrežama, omogućava da generišu ideje za objave na društvenim mrežama. Proces aplikacije se sastoji iz dva osnovna koraka. U prvom koraku korisnik unosi delatnost za koju želi da generiše objave na društvenim mrežama. Nakon toga GPT model generiše pitanja karakteristična za tu delatnost na koje korisnik treba da odgovori.

Drugi korak integriše te odgovore u novi zahtev za generisanje ideja i plana objava za društvene mreže.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Miroslav Zarić, vanr. prof.

1.2. GPT - Jezički generativni model

GPT (Generative Pre-trained Transformer) je vrsta naprednog računarskog modela zasnovanog na dubokom učenju koji se koristi za obradu prirodnog jezika. Ovaj model je obučen na ogromnom skupu tekstualnih podataka i sposoban je za razumevanje, generisanje i obradu teksta na prirodnom jeziku. GPT modeli su razvijeni da bi razumeli kontekstualne veze između reči i fraza u tekstu i mogu da izvršavaju različite zadatke, uključujući prevođenje jezika, generisanje teksta, odgovaranje na pitanja i mnoge druge.

Ono što GPT modele čini posebnim jeste njihova sposobnost prepoznavanja i generisanja ljudski razumljivog teksta na osnovu ulaznih informacija. Modeli kao GPT-3, na primer, mogu generisati vrlo složene tekstove koji zvuče kao da su napisani od strane stvarne osobe. Ovi modeli su postali ključni u mnogim aplikacijama koje zahtevaju obradu prirodnog jezika, kao što su automatsko odgovaranje na pitanja, generisanje sadržaja, personalizovane preporuke i drugo.

GPT pored tekstualnog upita prima i sledeće parametre:

- **Temperature:** Ovaj parametar kontroliše koliko je generisanje teksta predvidljivo. Više vrednosti povećavaju nasumičnost odgovora, dok niže vrednosti (bliže 0) deluju determinističnije i odabiraju najverovatniji izlaz.
- **Maximum length:** Ovaj parametar kontroliše maksimalnu dužinu odgovora koji generiše model. Ako je postavljena visoka vrednost, model će biti sposoban da generiše duže odgovore.
- **Stop sequences:** Ovaj parametar omogućava da se postave određene sekvence teksta na kojima može se zaustaviti generisanje teksta, na primer, ako se uključi reč "kraj" kao stop sekvencu, model će prestati sa generisanjem teksta kada naiđe na ovu reč.
- **Top P:** Predstavlja širinu vokabulara iz kojeg model može uzeti reči pri generisanju teksta. Na primer, ako je Top P niže, model će koristiti samo najverovatnije reči. Ako je Top P više, raznovrsnije reči mogu biti uključene.
- **Frequency Penalty:** Ovaj parametar smanjuje verovatnoću često korišćenih fraza. Više vrednosti znače veću kaznu za česte fraze.
- **Presence Penalty:** Ovaj parametar smanjuje verovatnoću reči koje su već upotrebljene. Više vrednosti znače veću kaznu za reči koje su već prisutne u tekstu.

1.3. Jaeger Telemetry

Jaeger telemetry je sistem za prikupljanje i analizu telemetrijskih podataka u kontekstu praćenja i analize performansi i dostupnosti aplikacija i servisa. Ovo je jedan sistem otvorenog koda koji pomaže inženjerima da brže otkrivaju i rešavaju probleme u svojim aplikacijama, smanjuje vreme do otkrivanja grešaka i na taj način poboljšava performanse aplikacije.

Jaeger je posebno koristan za identifikaciju uskih grla u performansama, otkrivanju grešaka i rešavanja problema u sistemu. Osim toga, omogućava efikasno analiziranje performansi tokom vremena, što je ključno za optimizaciju aplikacija.

1.4. FluentBit

Fluentbit je softver otvorenog koda za dobavljanje i prosleđivanje podataka u realnom vremenu. On je dizajniran da omogući efikasno prikupljanje i obradu podataka iz raznih izvora u raznim okruženjima. Fluentbit je najčešće korišćen za prikupljanje, filtriranje i prosleđivanje logova.

Podržava razne izvore podataka kao što su tekstualni logovi, standardni ulaz/izlaz aplikacije (stdio), JSON poruke, sistemski logovi, i mnoge druge formate. Omogućava filtriranje podataka koristeći brojne filtere kako bi se izdvojili i obradili željeni podaci. Ovo pomaže u smanjenju obima podataka koji se prosleđuju, čim se smanjuje opterećenje mreže i sistema za prikupljanje podataka. Fluentbit ima brojne izlazne destinacije, uključujući Elasticsearch, InfluxDB, Amazon S3, i mnoge druge. Ovo omogućava integraciju sa različitim sistemima za analizu i vizualizaciju podataka.

1.5 Prometheus

Prometheus je takođe sistem otvorenog koda za nadgledanje i alarmiranje, razvijen od strane SoundCloud-a. On predstavlja moćan alat za prikupljanje, nadgledanje i analizu podataka o radu aplikacija i infrastrukturi u realnom vremenu. Ovo je veoma korisno u okruženjima kod kojih je neophodno pružiti visoku dostupnost aplikacije [1].

Prometheus koristi tzv „strugače“ - skrejpere (eng. "scrapers") za prikupljanje metrika iz različitih izvora, kao što su aplikacije, servisi i infrastruktura. Skrejpери na određeni period (*long pooling*), zahtevaju informacije sa ciljanih izvora i prikupljaju metrike. Omogućava postavljanje alarma i pravila koja prate metrike i izazivaju alarme kada se zadovolji određeni uzorak [1].

Prometheus je dizajniran da bude lak za instaliranje i konfigurisanje, a ima i dobre performanse i skalabilnost. Možete ga koristiti i na malim i na velikim infrastrukturnim okruženjima.

1.5. Grafana

Grafana je popularna platforma otvorenog koda za prikaz i analizu podataka koja omogućava korisnicima da vizualizuju podatke iz različitih izvora na jednom mestu, kao što su metrike, logovi, tragovi (traces). Glavni akcenat Grafane je na monitoringu i analizi sistemskih podataka, ali je takođe korisna i za prikaz istorijskih podataka, kreiranje panela za upravljanje.

Popularni alati koji mogu da se koriste sa Grafanom su:

- Jaeger ili Zipkin: Za praćenje performansi i analiza traga, Jaeger i Zipkin su popularni alati koji se koriste zajedno sa Grafanom.
- Grafana Loki: Loki je alat za skladištenje logova razvijen od strane istih ljudi koji su radili na Prometheus-u. Loki omogućava skalabilno i efikasno skladištenje logova i može se integrisati direktno sa Grafanom. U ovom projektu korišće se FluentBit koji šalje logove na Loki.
- Prometheus: U kombinaciji sa Grafanom, omogućava vam vizualizaciju metrika i pravljenje panela za praćenje performansi aplikacija i infrastrukture.
- ELK Stack: Za praćenje logova, možete koristiti ELK stek koji uključuje Elasticsearch za skladištenje logova, Logstash za obradu i filtriranje logova, i Kibana (koji se takođe može integrisati sa Grafanom) za vizualizaciju i analizu logova.

Grafana ima bogat ekosistem dodataka (plagina) koji omogućavaju integraciju različitih izvora podataka i alata za logove, tragove (*tracing*) i metrike.

1.6. Pregled relevantne literature

DevOps integriše oblasti, razvoja i operacija, koristeći automatizaciju isporuke i nadzora infrastrukture. Ovaj pristup ima za cilj da omogući brži razvoj, bolju saradnju između timova i pouzdanu isporuku softvera.

Ključni elementi DevOps pristupa uključuju:

- Kontinualna integracija: Proces u kojem se promene u kodu redovno integrišu u glavnu granu repozitorijuma, praćene testovima kako bi se očuvala stabilnost aplikacije.
- Kontinualna isporuka: Automatski proces isporuke softvera u produkciji nakon uspešne integracije i testiranja.
- Automatizacija Infrastrukture: Upotreba skripti i alata za definisanje infrastrukture kao koda radi brže implementacije i skaliranja resursa.
- Praćenje Logova i Metrike: Skup alata i praksi za praćenje performansi aplikacije i identifikaciju problema u realnom vremenu.

”DevOps” nije jedan univerzalni proces ili metodologija kao što je na primer skram (eng. Scrum). ”DevOps” treba da se posmatra kao konceptualni okvir koji organizacije treba da prilagode svom jedinstvenom okruženju [3].

U radu ”Ebert, C., Gallardo, G., Hernantes, J. and Serrano, N., 2016. DevOps” navedeni su različiti alati koji se koriste u ovom pristupu razvoja softvera [2]:

1. Apache Ant: Alat za izgradnju aplikacija.
2. Maven: Slično kao Apache Ant samo unapređen
3. Jenkins: Ovo je alat za kontinuiranu integraciju.
4. Puppet: Ovo je alat za upravljanje konfiguracijama na serverima.
5. Chef: Ovaj alat takođe služi za automatizaciju upravljanja konfiguracijama.

6. Ansible: Ovo je alat za upravljanje konfiguracijom preko SSH (Secure Shell).
7. Loggly: Ovaj alat omogućava prikupljanje log podataka iz aplikacija, platformi i servera u realnom vremenu.
8. Nagios: Ovo je alat za nadgledanje i upravljanje zdravljem sistema.
9. New Relic: Omogućava praćenje performansi aplikacija.

Praktične primene GPT modela su opisane u radu "GPT-3: What's it good for?" Robert Dale [4]:

1. Kreativno pisanje: GPT-3 može generisati kreativne tekstove, uključujući pripovedanja, poeziju i dijaloge. Ovaj aspekt tehnologije može biti koristan u kreiranju filmova, video igara i drugih kreativnih projekata.
2. „Pojačano“ tj. unapređeno pisanje: GPT-3 može služiti kao alatka za pomoć u pisanju. Neke aplikacije već postoje koje omogućavaju korisnicima da unesu svoj tekst i dobiju alternativnu verziju na osnovu stila i sadržaja. Ovo može biti korisno za pisanje mejlova, blogova, reklamnih materijala itd.
3. Pitanja i odgovori: GPT-3 može odgovarati na pitanja na osnovu prethodno datog teksta. Ovaj aspekt tehnologije može biti koristan u obrazovanju, kada je potrebno dobiti odgovore na pitanja iz različitih oblasti znanja.
4. Prevod: GPT-3 može biti koristan za mašinski prevod teksta sa jednog jezika na drugi. Međutim, treba biti oprezan prilikom korišćenja ove funkcionalnosti, posebno kod prevoda pravnih dokumenata ili drugih važnih tekstova, jer GPT-3 može generisati netačne prevode.
5. Pomoć u fitnessu: Postoje aplikacije koje koriste GPT-3 za odgovaranje na pitanja iz oblasti fitnesa i zdravlja. Ovo može biti korisno za dobijanje saveta o vežbanju i ishrani.

Uprkos svojim moćnim sposobnostima, GPT-3 još uvek nema pravo razumevanje prirodnog jezika i suočava se sa izazovima u zadacima poput zaključivanja na osnovu prirodnog jezika i razumevanja pročitano.

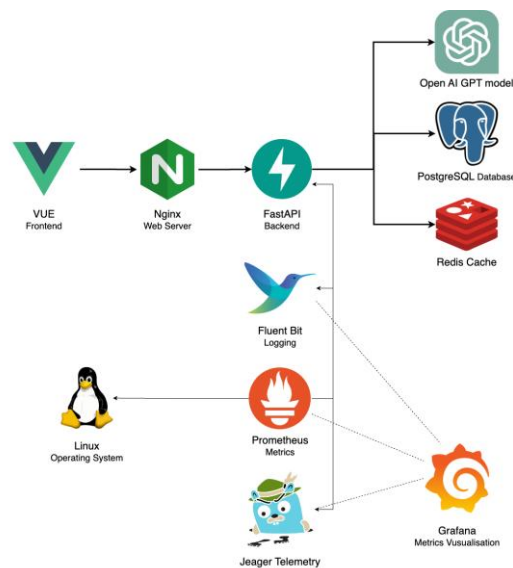
Ipak, GPT-3 je inspirisao nove ideje u oblasti dubokog učenja i ima potencijalnu primenu u inteligentnim pomoćnim zadacima [5].

2. SPECIFIKACIJA SISTEMA

Na slici 2.1 prikazana je arhitektura sistema pomenutog projekta za generisanje ideja za objave na društvenim mrežama pomoću GPT modela.

Aplikacija se sastoji iz Vue klijentskog dela aplikacije, koji predstavlja korisnički interfejs. Sa serverske strane se nalazi Nginx server koji je jedini otvoren prema internetu.

Nginx sve servere prosleđuje FastAPI aplikaciji (Glavnom servisu). Servisni deo aplikacije komunicira sa GPT modelom pomoću REST API-ja, PostgreSQL relacionom bazom podataka i Redis kešom.

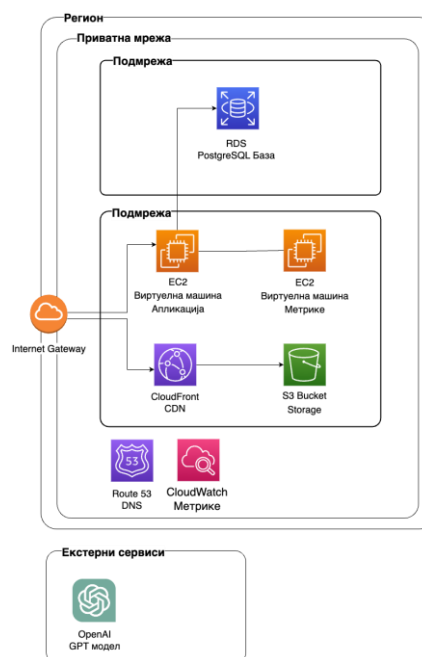


Slika 2.1 Arhitektura sistema

Od alata za praćenje (monitoring), koristi se Jaeger za praćenje tragova (tracing), FluentBit za agregaciju logova aplikacije, Prometheus za dobavljanje metrika sistema i aplikacije i Grafana za vizuelizaciju svih ovih podataka.

3. ISPORUKA NA CLOUD

Da bi aplikacija bila javno dostupna podignuta je na AWS cloud servis. na slici 3.1.1 je prikazana, infrastruktura servisa na AWS-u.



Slika 3.1.1 Infrastruktura aplikacije na AWS-u

Aplikacija je dodatno prilagođena da pomoću *opentelemetry* biblioteke generiše i šalje tragove sa informacijama o trajanju zahteva na Jaeger server, takođe pomoću iste biblioteke se ispisuju logovi koji se šalju na Fluentbit alat koji prikuplja logove i prosleđuje ih na Grafana Loki alat za agregaciju i dobavljanje logova.

Na mašini na kojoj je podignuta aplikacija instaliran je servis Node Exporter koji čita podatke o samoj mašini kao što su opterećenje procesora, zauzeće RAM-a, zauzeće masovne memorije i objavljuje ih na port 9100. Podatke sa tog porta čita alat Prometheus koji na određen vremenski period preuzme vrednosti i sačuva. Konačno Grafana alat se koristi za vizuelizaciju i pretragu svih podataka o radu aplikacije (tragovi, logovi i metrike). Svi dodatni servisi za monitoring aplikacije su podignuti na posebnoj EC2 instanci, osim Node Exportera koji je instaliran na instanci gde se nalazi aplikacija kako bi preuzimao informacije o toj mašini.

Za frontend aplikaciju je korišćen S3 bucket za smeštanje statičkih fajlova i CloudFront za distribuciju tih fajlova (frontend aplikacije). CloudFront je povoljno rešenje pošto je veoma skalabilno i naplaćuje se po količini zahteva, takođe pruža besplatan sertifikat za naš domen. Sertifikat za backend deo aplikacije je obezbeđen pomoću Certbot alata koji obezbeđuje "Let's Encrypt" sertifikat i automatski ga obnavlja.

GitHub je korišćen za kontrolu verzija projekta tokom razvijanja. Shodno tome GitHub akcije su upotrebljene za automatizaciju isporuke aplikacije. Prilikom postavljanja koda na glavnu granu repozitorijuma pokreće se akcija koja ima sledeće korake:

1. Preuzimanje sadržaja iz repozitorijuma
2. Pokretanje testova u aplikaciji
3. Kopiranje "backend" programskog koda na udaljeni server (EC2 instancu)
4. Pokretanje skripte na serveru za ažuriranje
 - 4.1. Pravi kopiju trenutno aktivne aplikacije
 - 4.2. Pokreće kopiju privremeno na posebnom portu
 - 4.3. Menja port Nginx-a na privremeni (koji i dalje ima staru verziju aplikacije)
 - 4.4. Menja fajlove glavne aplikacije sa novom verzijom
 - 4.5. Pokreće se nova verzija
 - 4.6. Preusmerava se port u Nginx-u na početni na kom se sada nalazi nova aplikacija
 - 4.7. Gasi se i briše privremena verzija
5. Izgrađuje se Vue 3 "Frontend" aplikacija
6. Ažurira se "Frontend" aplikacija
 - 6.1. Briše se stari sadržaj iz S3 Bucket-a
 - 6.2. Otprema se novi statički fajlovi za "frontend"
 - 6.3. Invalidira se sadržaj CloudFront-a da bi se preuzela nova verzija na produkciji

5. ZAKLJUČAK

U ovom radu prikazan je jedan primer jezičkog generativnog modela u aplikaciju koja se isporučuje na platformi za računarstvo u oblaku korišćenjem DevOps principa. Prikazani su i načini za detaljno praćenje same aplikacije, kao što su praćenje logova, performansi servisa pomoću tragova (tracing) i opterećenja samog servera na kojoj se nalazi aplikacija. Prikazan je jedan od načina za isporuku aplikacije na AWS. Implementirana je kontinualna integracija, monitoring servisa na produkciji. Rešeni su sertifikati kako bi se omogućila bezbedna HTTPS konekcija.

Potencijalna unapređenja aplikacije bi išla u smeru zamene upotrebe virtuelnih mašina, koje može biti nezgodno za održavanje, kontejnerizovanim servisnim modulima i isporuci na neki od servisa za rad sa kontejnerima. Mana tog pristupa je što može biti skuplji u odnosu na jednostavne virtuelne mašine. Pošto aplikacija trenutno ne rukuje sa velikim brojem korisnika i nije od kritičnog značaja da imamo savršenu dostupnost, možemo da prihvatimo nedostatke korišćenja virtuelnih mašina za servise.

6. LITERATURA

- [1] Gibbs, J., 2018. Prometheus: Games User Research Tool Development Using Best Practices.
- [2] Ebert, C., Gallardo, G., Hernantes, J. and Serrano, N., 2016. DevOps
- [3] Erich, F., Amrit, C. and Daneva, M., 2014. Report: Devops literature review. University of Twente, Tech. Rep.
- [4] Dale, R., 2021. GPT-3: What's it good for?. Natural Language Engineering, 27(1), pp.113-118.
- [5] Zhang, M. and Li, J., 2021. A commentary of GPT-3 in MIT Technology Review 2021. Fundamental Research, 1(6), pp.831-833.

Kratka biografija:



Miloš Popović rođen je 25.01.2000. godine u Šapcu. 2018. godine upisuje osnovne studije na Fakultetu tehničkih nauka u Novom Sadu na smeru Softversko inženjerstvo i informacione tehnologije. 2022. godine je diplomirao i potom upisao master studije na istom fakultetu.

SISTEM ZA REVERZNU PRETRAGU SLIKA**REVERSE IMAGE SEARCH SYSTEM**Ivan Mršulja, *Fakultet tehničkih nauka, Novi Sad***Oblast – PRIMENJENE RAČUNARSKE NAUKE I INFORMATIKA**

Kratak sadržaj – Ovaj rad predlaže alternativni pristup reverznoj pretrazi slika korišćenjem modela mašinskog učenja za izvlačenje tagova i svojstava iz slike. Svođenjem vektorske pretrage na pretragu teksta, predloženi metod povećava efikasnost i tačnost pretrage slika.

Ključne reči: Reverse pretraga slika, mašinsko učenje, računarska vizija.

Abstract – This paper proposes an alternative approach to reverse image search by leveraging machine learning models to extract tags and features from an image. By reducing vector image search to a full-text search, the proposed method improves the efficiency and accuracy of image retrieval.

Keywords: Reverse image search, machine learning, computer vision.

1. UVOD

Reverzna pretraga slika (*reverse image search*) je tehnika pretrage koja omogućava korisnicima da pronađu slične ili identične slike na internetu koristeći već postojeću sliku kao ulazni parametar [1]. Umesto da korisnici unose tekstualne upite, mogu otpremiti sliku ili uneti URL slike kako bi pronašli druge slike sa sličnim vizuelnim karakteristikama [2]. Algoritmi za reverznu pretragu slika analiziraju ključne elemente slike, kao što su boje, oblici, teksture ili drugi vizuelni detalji, kako bi pronašli slike sa sličnim karakteristikama iz baza podataka ili veb stranica širom interneta. Ova tehnika je korisna u mnogim situacijama, kao što su pronalaženje izvora slike, identifikacija objekata, pronalaženje sličnih vizuelnih sadržaja, otkrivanje plagijata ili pronalaženje više informacija o određenom objektu na slici. Reverzna pretraga slika može biti od pomoći u istraživačkim, kreativnim i bezbednosnim kontekstima, pružajući korisnicima mogućnost da pronađu relevantne slike na osnovu već postojeće vizuelne reference [2].

U ovom radu, rešavan je problem implementacije jednog ovakvog sistema, gdje korisnik može unijeti proizvoljnu sliku kao uzorak upit, a sistem mu za isti vraća slike koje najbolje odgovaraju na njega. Pored toga, omogućeno je i indeksiranje slika. Takođe, adresiran je i problem velikog broja korisnika koji bi potencijalno koristili ovaj

sistem, te se razvoj vodio praksama servisno-orijentisane arhitekture gdje je svaki servis moguće nezavisno skalirati u proizvoljnom obimu.

1.1. Svođenje problema pretrage vektora na pretragu teksta

Jedinstvenost ovog rešenja leži u formatu svojstava koja se koriste prilikom indeksiranja i pretrage. Većina današnjih sistema se oslanja na vektorske reprezentacije dokumenata kako bi implementirala pretragu. Iako je ovo veoma efikasan način pretrage vizuelno sličnih slika, veliki je problem što se eksplicitno u obzir ne uzimaju svi objekti sa slike već se čitava slika reprezentuje kao neki *black-box* skup svojstava.

Sa druge strane, moje rešenje svodi pretragu slika na pretragu teksta, modelom mašinskog učenja (YOLOv5) se prepoznaju objekti, koji se posmatraju kao ključne riječi prilikom indeksiranja u *Elasticsearch* sistemu za pronalaženje informacija, pomoću kojih možemo prepoznati sve kontekstualno slične slike sa uzorkom dok se prostor boja koristi kao opcion filter kako bismo mogli dobiti vizuelno najsličnije slike.

1.2. Elasticsearch

Elasticsearch je visoko skalirajući, distribuirani sistem za pretragu i analizu podataka. On je osmišljen da brzo i efikasno indeksira, čuva i pretražuje velike količine strukturisanih i nestrukturisanih podataka. *Elasticsearch* je baziran na *Apache Lucene* biblioteci, koja pruža moćne algoritme za pretragu. Takođe podržava brojne napredne funkcionalnosti kao što su pretraga punim tekstom (*full-text search*), filtriranje, agregacija, geolokacija, analiza teksta, vektorsko indeksiranje i pretraga i još mnogo toga.

Osim pretrage, *Elasticsearch* takođe omogućava analizu podataka i vizualizaciju rezultata. Kroz *Kibana* interfejs, korisnici mogu pristupiti bogatim vizualizacijama, izveštajima i nadzoru nad podacima. *Elasticsearch* je dizajniran da bude otporan na kvarove i pruža mogućnost replikacije podataka i distribucije preko više čvorova.

Takođe podržava horizontalno i vertikalno skaliranje, omogućavajući dodavanje novih čvorova kako bi se postigla bolja brzina i otpornost na visoka opterećenja, kao i unapređenu dostupnost usluge. Sa svojom fleksibilnošću, skaliranjem i moćnim alatima za pretragu i analizu, *Elasticsearch* je postao popularan izbor za razne primene kao što su pretraga veb stranica, analiza logova i sistemi za praćenje događaja (*event tracking systems*) [4].

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivanović, red. prof.

1.3. YOLOv5

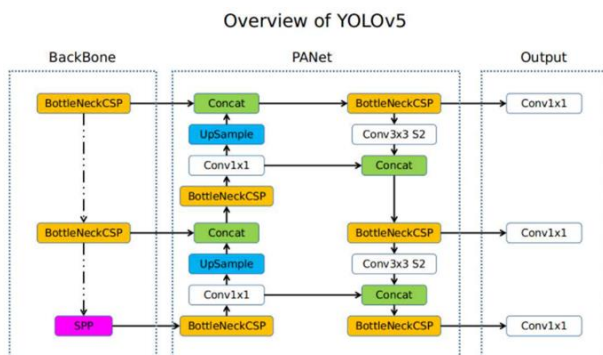
YOLOv5 model za detekciju objekata baziran je na *end-to-end* (e2e) arhitekturi i sastoji se od tri glavne komponente:

1. Vlasničke mreže (*backbone network*)
2. Glavne mreže (*neck network*)
3. Mreže izlaza (*output network*)

Vlasnička mreža (*backbone*) u YOLOv5 modelu je CSPDarknet53 mreža, koja se bazira na konvolucionim slojevima. Ova mreža je modifikovana verzija originalne Darknet53 mreže i pruža bolju reprezentaciju svojstava i razumevanje konteksta objekata. Ona je odgovorna za identifikaciju i izdvajanje bitnih svojstava iz ulaznih slika.

Glavna mreža (*neck network*) je dodatni sloj između vlasničke mreže i mreže izlaza. Ona ima za cilj da usavrši svojstva iz vlasničke mreže i podesi ih tako da bi lakše detektovala objekte različitih veličina i proporcija. U specifikaciji YOLOv5 modela, koristi se PANet mreža koja je korišćena i u starijem YOLOv4 modelu.

Mreža izlaza (*output network*) predstavlja poslednju komponentu modela i sadrži konvolucione slojeve koji izvode konačne predikcije o klasama i lokacijama objekata. Ona preobrađuje izlazne "feature" iz glavne mreže u *anchor-box*-ove i vraća koordinate, sigurnost i klase detektovanih objekata za svaki *bounding-box*. U pojedinoj literaturi, ovaj segment u "pipeline"-u modela naziva se i "head network" [5]. Na slici 1 prikazana je gore opisana arhitektura.



Slika 1. Arhitektura YOLOv5 modela

1.4. Pregled relevantne literature

U radu [3] prikazana je *Similar Looks* funkcionalnost unutar Pinterest aplikacije koja omogućava korisnicima da izaberu predmet unutar slike, stavljajući okvir oko njega, a zatim vraća vizuelno slične slike/predmete. Glavni cilj ove funkcije je pomoći korisnicima da pronađu stvari koje ne mogu imenovati [2].

Indeksiranje slika se u ovom sistemu vrši po *incremental fingerprinting service* (IFS) sistemu. Ovaj sistem funkcioniše po principu da svaka slika ima svoj otisak koji se sastoji od svih odvojenih objekata koji se nalaze unutar te slike (npr. cipele, torbe, čaše itd.). Svaki objekat ima oznaku koja specifikuje ime klase kojoj objekat pripada [3].

Prilikom pretrage, sistem koristi tehnologiju prepoznavanja objekata da lokalizuje i klasifikuje predmete (objekte)

na slikama. Iz ovako detektovanih objekata se izvlače karakteristike i koriste se za dalju pretragu. Drugim riječima, skraćuje se zadatak detekcije objekata više klasa u klasifikaciju kategorija, tako da umesto da se traži podudaranje u svim slikama na Pinterest-u, prvo se dobavljaju slike koje su u istoj kategoriji.

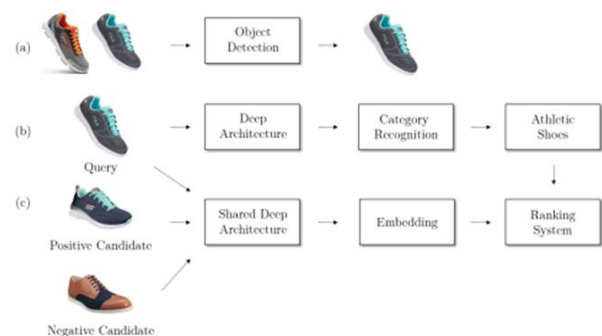
Ovo je odličan pristup jer se upitna slika upoređuje sa slikama koje imaju visoku vjerovatnoću da su slične. Prethodni korak filtriranja povećava stopu tačnih pozitiva. Sistem se zasniva na konceptima dubokog učenja, kao što su konvolutivne neuronske mreže (CNN) i detektori objekata zasnovani na dubokom učenju (*Single Shot Detectors*, SSD).

Sistem je implementiran korišćenjem radnog okvira Caffe (Java tehnologija). Podaci sistema se skladište na Amazon S3-u [2][3].

U radu [1] tok rada vizualne pretrage je sledeći: korisnik *upload*-uje sliku ili je snima kamerom, a rezultat će biti slične slike i predmeti; korisnik može izabrati da li želi da kupi (slično "Shopping" sekciji kod Google pretraživača) ili dalje istražuje predmete [2]. Ovaj sistem obuhvata tri glavne faze:

1. Obuka modela: U sistemu se koristi nekoliko modela dubokih neuronskih mreža kako bi se poboljšala relevantnost rezultata, kao što su AlexNet, ZFSPNet, GoogleNet, ResNet, a takođe se koristi i zajednički algoritam kmeans kako bi se izgradio inverzni indeks u nivou-0 podudaranja, pri čemu nivo 0 označava najmanju sličnost sa upitnom slikom [1].
2. Razumevanje upita (slike): Iz upitne slike izvlače se različite karakteristike koje opisuju njen sadržaj, uključujući karakteristike prepoznavanja kategorije, prepoznavanja lica, boje i otkrivanja duplikata, enkodere dubokih neuronskih mreža (DNN) i detekciju objekata (slika 2).
3. Pretraga slika: Pronalazi vizualno slične slike na osnovu izvučenih karakteristika i namere korisnika.

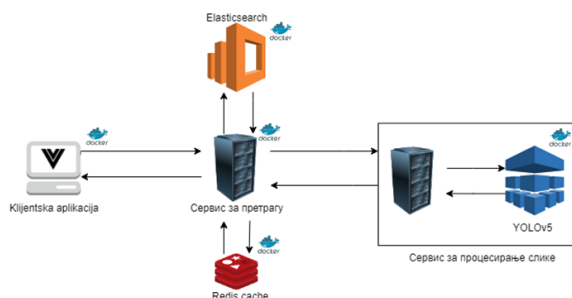
Svi modeli se treniraju na obučavajućem skupu podataka koji su prikupljeni za određene domene npr. kupovina. Za nadzor obuke DNN koriste se više funkcija gubitka, kao što su Softmax, Pairwise-loss i Triplet-loss funkcije [1].



Slika 2. Primjene DNN modela koji se koriste u Bing-ovom sistemu

2. SPECIFIKACIJA SISTEMA ZA REVERZNU PRETRAGU SLIKA

U ovom poglavlju prikazana je arhitektura sistema sa aspekta dijagrama razmeštaja. Dijagram razmeštaja (slika 3) prikazuje softverske komponente i veze uz pomoć kojih iste komuniciraju.



Slika 3. Dijagram razmeštaja

Ovo softversko rješenje se sastoji iz dva zasebna servisa:

1. Servisa za pretragu i indeksiranje implementiranog u *Java Spring Boot* radnom okviru
2. Servisa za procesiranje slike implementiranog u *Python FastAPI* radnom okviru uz oslonac na *OpenCV* i *NumPy* biblioteke za implementaciju i integraciju neophodnih algoritama mašinskog učenja koji se koriste prilikom procesiranja slike

Pored ova dva servisa bitno je napomenuti i da postoji odvojeni frontend sloj implementiran u *Vue3* radnom okviru koristeći *JavaScript* programski jezik.

Servis za pretragu i indeksiranje komunicira sa servisom za procesiranje slike kako bi izvukao svojstva (*features*) sa slike koja je data kao uzorak za upit.

Pored funkcije pronalaženja informacija i indeksiranja, ovaj servis vrši i keširanje svojstava slike koristeći *Redis key-value* bazu podataka kako bi ubrzao vrijeme pretrage za više sukcesivnih upita sa istom slikom (pri listanju paginiranih rezultata), što predstavlja veoma bitan faktor u radu ovog sistema kako bi bio pogodan za korišćenje. Indeksiranje kao i pretraga je implementirana koristeći *Elasticsearch* sistem za pronalaženje informacija.

3. EKSPERIMENT

Eksperiment je sproveden u vidu ankete:

1. Svaki učesnik prvo instalira aplikaciju koristeći *docker-compose* alat.
2. Nakon toga, svaki korisnik treba da indeksira indeksni skup podataka kroz formu za indeksiranje.
3. Na kraju, za svaku od slika u upitnom skupu podataka, korisnik ima polje u anketi da ocijeni ocjenom od 1 do 5 odziv, redoslijed odgovora, kao i redoslijed odgovora prilikom korišćenja opcije za sortiranje pomoću prostora boja.

Prilikom davanja ocjena, korisnik ima na uvid cijeli indeksni skup podataka kako bi mogao sam da donese odluku o broju *true positive* instanci u datom skupu podataka.

1.3. Skup podataka

Testiranje se vrši upotrebom ručno sakupljenog skupa podataka korišćenjem *Google Images* servisa. Sakupljeno je 30 slika za indeks skup i 10 slika za upitni skup. Bitno je napomenuti da su slike izabrane tako da postoji veliki broj slika koje mogu da odgovore na više od jednog upita kako bi konačna procjena svih metrika bila ostavljena subjektivnom osjećaju korisnika.

1.4. Evaluacija

Evaluacija sistema se vrši na veoma jednostavan način. Korisnicima je naznačeno u upitniku da po sopstvenom nađenju izvuku mjeru odziva i redoslijed odgovora za svaki od *sample* upita.

4. REZULTATI I TUMAČENJA

Rezultati prethodno opisanog eksperimenta predstavljeni su u tabeli 1. Za svaku uzoračku sliku izvučene su sledeće statistike: prosječan odziv kod svih ispitanika, prosječna preciznost bez korišćenja dodatnih filtera, prosječna preciznost pri korišćenju sortiranja u prostoru boja. U ispitivanju je učestvovalo 10 kandidata.

Tabela 1. Rezultati eksperimenta

Sadržaj slike	Odziv	Preciznost (default)	Preciznost (HSV sort)
auto	5	3	3,25
torta	4	3,25	3
mačka	4,5	4,25	4,25
pas	3	4,25	4
igra u pijesku	4,75	2,75	4,25
pica	2,75	2	4,75
ptice	5	5	4,25
televizor	3,25	4	4
ulica	4,5	5	4
pribor za jelo	3,75	4,25	4,25
Prosjek	4,05	3,775	4

Iz priloženih rezultata može se zaključiti da je odziv sistema zadovoljavajuć sa prosječnom ocjenom 4,05 od 5, dok je redoslijed odgovora ocijenjen sa 3,775 i 4, zavisno da li je rezultat sortiran pomoću ugrađenog value parametra ili uz pomoć prostora boja. Najlošiji odziv je dobijen prilikom upita u kojem je kao uzorak korišćena slika sa picom. Kada je ovaj slučaj detaljnije analiziran, došlo se do nalaza da je pri procesiranju slike, pica detektovana kao kolač, stoga nije vraćena velika količina

slika na kojima se nalazio relevantan objekat. Način da se ovaj problem riješi je dodatno dotreniranje *YOLOv5* modela nad klasom objekta koja ima najviše problema prilikom klasifikacije.

5. ZAKLJUČAK

U ovom radu vidjeli smo jedan način implementacije sistema za reverzno pretraživanje slika. Ovakav sistem omogućava korisnicima da umesto da se oslanjaju na ključne reči ili opise za pretragu slika, na efikasan način pronađu željene informacije na osnovu same slike. Ovo može da ima raznoliku primenu, od pomoći u identifikaciji različitih predmeta i lokacija, do potpunog unapređenja iskustva kupovine preko interneta.

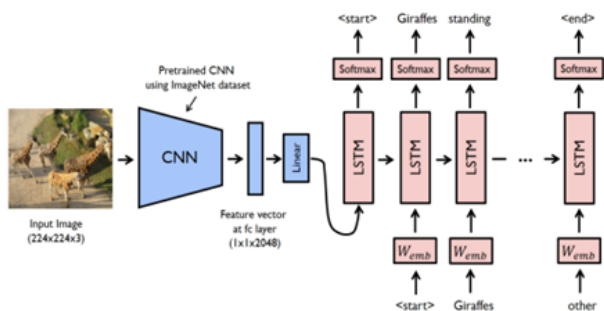
Sistem se sastoji iz dva odvojena servisa: servisa za pretragu koji vrši indeksiranje i pretragu koristeći *Elasticsearch* sistem za pronalaženje informacija, kao i keširanje koristeći *Redis in-memory* bazu, i servisa za procesiranje slike koji koristi moćne modele mašinskog učenja kako bi izvukao svojstva iz slike i predstavio je u formatu koji je lako pretraživ.

Za razliku od većine sistema koji vrše pretragu koristeći vektorski način pretrage (npr. *Web-Scale Responsive Visual Search*), ovaj sistem svodi problem pretrage slike na problem pretrage teksta gde se svojstva izvlače automatski, nasuprot tradicionalnom pristupu gde su se unosila ručno. Ovaj inovativan pristup već dugo je zastupljen u *Pinterest*-ovom *Similar Search* servisu i veoma lako može biti nadopunjen ručnim unosom opisa slike, ključnih reči itd.

5.1. Dalji razvoj sistema

S obzirom na to da je implementiran samo podskup različitih mogućnosti procesiranja slika, dodavanjem bilo kakvih dodatnih načina procesiranja moguće je postići poboljšanje sistema. Jedna stvar koja najviše ima smisla jeste da se postojeći način pretrage ukombinuje sa vektorskim načinom pretrage, tako da postojeći način isfiltrira kontekstno relevantne rezultate dok vektorska pretraga pronalazi vizuelno najsličniju sliku. Olakšavajuća okolnost kod ovog pristupa je ta što *Elasticsearch* podržava vektorsko indeksiranje i pretragu *out-of-the-box* [6].

Takođe, dodavanje modula za automatsko generisanje opisa ili ključnih reči iz slike korišćenjem mašinskog učenja moguće je poboljšati kvalitet izvučenih svojstava i proširiti domen primene gde bi se pored reverzne pretrage mogla implementirati i standardna tekst bazirana pretraga.



Slika 4. Prijedlog arhitekture pipeline-a za generisanje opisa slike

Jedan od načina na koji bi ovo moglo biti uvedeno jeste korišćenjem *CNN*-a za ekstrakciju obeležja (slika 4) i rekurentnu neuronsku mrežu (poput *LSTM*-a) ili transformer za generisanje teksta [7].

Još jedan zanimljiv način implementacije jeste integracija ovog sistema sa nekim *LLM*-om koji može generisati opis/ključne reči slike (na primer, *ChatGPT4*).

6. LITERATURA

- [1] Hu, H., Wang, Y., Yang, L., Komlev, P., Huang, L., Chen, X., Huang, J., Wu, Y., Merchant, M. and Sacheti, A., 2018, July. Web-scale responsive visual search at bing. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 359-367).
- [2] Al-Lohibi, H., Alkhamisi, T., Assagran, M., Aljohani, A. and Aljahdali, A.O., 2020. Awjedni: a reverse-image-search application. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 9(3), p.49.
- [3] Jing, Y., Liu, D., Kislyuk, D., Zhai, A., Xu, J., Donahue, J. and Tavel, S., 2015, August. Visual search at pinterest. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1889-1898).
- [4] Elasticsearch, B.V., 2018. Elasticsearch. software, version, 6(1).
- [5] Zhu, X., Lyu, S., Wang, X. and Zhao, Q., 2021. TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 2778-2788).
- [6] Amato, G., Bolettieri, P., Carrara, F., Falchi, F. and Gennaro, C., 2018, June. Large-scale image retrieval with elasticsearch. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (pp. 925-928).
- [7] Yadav, A., Vishwakarma, A., Panickar, S. and Kuchiwale, S., 2020. Real time video to text summarization using neural network. *Int. Res. J. Eng. Tech*, 7, pp.1828-36.

Kratka biografija:



Ivan Mršulja rođen je 31.01.2000. godine u Kotoru. Smjer softversko inženjerstvo i informacione tehnologije na Fakultetu tehničkih nauka u Novom Sadu upisao je 2018 godine. Osnovne studije završio je u septembru 2022. godine. Od oktobra 2022. godine upisuje master studije i angažovan je kao saradnik u nastavi.

Kontakt: ivan.mrsulja@uns.ac.rs

ПРЕДИКЦИЈА СРЧАНОГ УДАРА НА ОСНОВУ СТАЊА ПАЦИЈЕНТА ПРИМЕНОМ АЛГОРИТАМА МАШИНСКОГ УЧЕЊА**HEART ATTACK PREDICTION BASED ON THE PATIENT'S CONDITION USING MACHINE LEARNING ALGORITHMS**

Андријана Јеремић, Јелена Сливка, *Факултет техничких наука, Нови Сад*

Област – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – Срчани удар или акутни инфаркт миокарда је изумирање дела срчаног мишића због наглог престанка циркулације кроз неку од артерија које исхрађују срце. Један је од најчешћих узрока смрти у развијеним земљама и земљама у развоју. Чешће погађа мушкарце него жене, а после уласка у климакс, ризик од настанка болести се изједначава. Присутнији је код људи старијих од 40 година. Фактори кардиоваскуларних ризика су различити и на неке од њих се може утицати, а на неке не. Предикција срчаног обољења код особе може бити врло непредвидива и дуготрајна, а често и неуспешна. Зато је данас све заступљенија употреба вештачке интелигенције за решавање овог проблема. У овом раду је предвиђано да ли ће се код пацијента јавити ово обољење на основу његовог здравственог стања. Обучени следећи модели: логистичка регресија, наивни Бајес, метод потпорних вектора, модел *K* најближих суседа, метод насумичне шуме, *Light Gradient-Boosting Machine*, *eXtreme Gradient Boosting* и *Categorical Boosting (CatBoost)*. Последњи модел се испоставио као најбољи над коришћеним скупом података. Интерпретацијом овог модела дискутовани су фактори који највише утичу на предикцију појаве срчаног удара.

Кључне ријечи: срчани удар, машинско учење, *CatBoost*, *XGBoost*, *SHAP values*.

Abstract – A heart attack or acute myocardial infarction is the death of part of the heart muscle due to a sudden cessation of circulation through one of the arteries that feed the heart. It is among the most common causes of death in developed and developing countries. It affects men more often than women, and after entering the climax, the risk of the disease becomes even. It is more common in people over 40 years old. Cardiovascular risk factors differ; some can be influenced, and others cannot. The process of detecting or predicting a heart disease in a person can be very unpredictable, long-lasting, and often unsuccessful. Thus, applying artificial intelligence to solve this problem is becoming more common today. This paper used machine learning models to predict whether a patient will have a heart attack based on his health

condition. The following models were trained: logistic regression, Naïve Bayes, Support Vector Machine (SVM), *K*-Nearest Neighbors (KNN), Random Forest method, *Light Gradient-Boosting Machine (LightGBM)*, *eXtreme Gradient Boosting (XGBoost)* and *Categorical Boosting (CatBoost)*. The last model was the best-performing model, and it was interpreted to uncover factors most affecting heart attack prediction.

Keywords: heart attack, machine learning, *CatBoost*, *XGBoost*, *SHAP values*.

1. УВОД

Срчани удар, односно акутна миокардијална инфаркција, је једна од манифестација исхемијске болести срца која је најчешћи узрок смрти у земљама широм света. Фактори ризика који предиспонирају одређену популацију да оболи од срчаног удара могу бити разни, али се сматра да међу главне спадају: атеросклероза, старост, пол и повишени крвни притисак. С обзиром на висок степен смртности када је у питању ово обољење, машинско учење је веома корисно јер може помоћи у предикцији ове болести код пацијената на основу њиховог здравственог стања.

У овом раду су тренирани модели машинског учења да, на основу параметара биохемијске анализе и физичког стања пацијената, предвиђају да ли ће се код особе јавити срчани удар или не. Предикција се врши на основу горе поменутих, али и многих других фактора који утичу на појаву овог обољења. Модел са највишим перформансама је интерпретиран у циљу откривања фактора који највише утичу на предикцију срчаног удара.

У наредном поглављу дат је преглед сродних радова. Детаљи решења и имплементације система су описани у поглављу 3, а поглавље 4 садржи анализу добијених резултата. Последње, 5. поглавље, представља закључак рада.

2. ПРЕТХОДНА РЕШЕЊА

Након одабира теме рада, извршена је детаљна анализа сродних истраживања и других радова на исту или сличну тему. На основу резултата тих радова, одлучено је која методологија ће бити примењена као и сам циљ којем се тежи.

У раду [1] илустровани су доступни скупови података везани за срчане болести који су углавном сирове природе, што је сувишно и недоследно. Дакле, постоји потреба за претходном обрадом оваквих

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији ментор је била др Јелена Сливка, ванр. проф.

скупова података. Скуп високо-димензионалних података мора се свести на скуп ниско-димензионалних. Рад [1] указује и на значај издвајања кључних карактеристика из скупа података.

Избор важне карактеристике смањује рад на обучавању алгорита и самим тим резултира смањењем временске сложености. Време игра и виталну улогу у доказивању ефикасности сличних алгоритама.

Рад [2] се бави имплементацијом модела учења за анализу стварних случајева срчаног удара на основу различитих атрибута. Коришћени модели су логистичка регресија, стабло одлучивања, метод насумичне шуме и модел К најближих суседа, од којих су последња два дала најбоље резултате. Сходно томе, метод насумичне шуме и модел К најближих суседа су имплементирани и у овом раду. Скуп података коришћен у раду [2] садржи 303 инстанце и представља део података искоришћених и за ово истраживање.

Тема рада [3] јесте “Предикција срчаног удара коришћењем машинског учења” и методологија обухвата само логистичку регресију.

Решење овог пројекта евалуирано је уз помоћ унакрсне валидације. Тачност добијена за логистичку регресију јесте 87%. Скуп података коришћен у раду [3] јесте евидентно шири односно обимнији у односу на рад [2], али на основу тачности која је добијена и додатне анализе, закључак је да постоје модели машинског учења који дају боље резултате у овом случају иако је и овај проценат врло висок.

Истраживање рада [4] се првенствено фокусира на скуп података о пацијентима, који се претходно обрађује, а затим се над њим примењују различити алгоритми машинског учења (KNN, SVM, логистичка регресија, метод насумичне шуме, наивни Бајес, неуронска мрежа). Коришћени скуп података исти је као и скуп података уврштен у рад [2], с тим што је у раду [4] имплементирана и SMOTE (Synthetic Minority Oversampling Technique) техника због неуравнотежене природе овог скупа података. Ова техника синтетичког узорковања примењена је и у овом раду као и већина модела машинског учења из рада [4].

Пројекат описан у раду [5] представља упоређивање перформанси различитих класификатора – стабла одлучивања, метода потпорних вектора, метода насумичне шуме и метода наивни Бајес, од којих је већина искоришћена и у овом пројекту. Рад [5] предлаже и класификатор ансамбла који врши хибридную класификацију комбинујући најбоље карактеристике и јаких и слабих класификатора.

Коришћен је UCI Heart Disease скуп података који обухвата 76 атрибута који су сведени на 14 најбитнијих. У рад [5] уврштен је и XGBoost (eXtreme Gradient Boosting) модел машинског учења, који представља један од најбитнијих модела обрађених и у овом раду. Модел предвиђања креиран је након поређења тачности сваке од техника машинског учења обрађених у раду [5]. Циљ је био искористити различите метрике евалуације, као што су матрица конфузије, тачност, прецизност, одзив и F1-мера. Класификатор XGBoost имао је највећу тачност (81%), када се упореде сви тестирани класификатори.

3. МЕТОД

У наредним поглављима су описани скуп података коришћен у раду и имплементација система за предикцију срчаног удара.

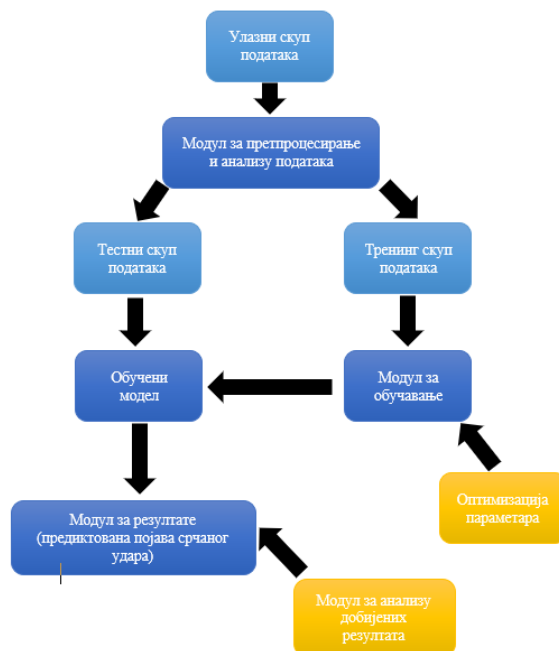
3.1. Скуп података

Скуп података који је коришћен у овом раду обухвата 1190 инстанци и преузет је са линка [6]. Он обједињује четири скупа података који се односе на пацијенте из следећих области: Мађарске, Кливленда, Швајцарске и Лонг Бича. Ови скупови података садрже 14 атрибута, који су, након истраживања и анализе других радова са истом темом, кориговани у 12 најбитнијих: године (age), пол (sex), тип бола у грудима (chest_pain_type), крвни притисак у мировању (resting_bp_s), холестерол (cholesterol), ниво шећера у крви (fasting_blood_sugar), електрокардиографија у мировању (resting_ecg), максималан број откуцаја срца (max_heart_rate), ангина изазвана вежбањем (exercise_angina), ST депресија изазвана вежбањем у односу на одмор (oldpeak), максимум ST сегмента изазван вежбањем (ST_slope) и циљни атрибут (target).

Дакле, овај скуп података садржи више предикаторских (независних) променљивих и једну циљну (зависну) променљиву која представља исход и која је бинарна вредности (0 – срчани удар се није догодио, 1 – срчани удар се догодио).

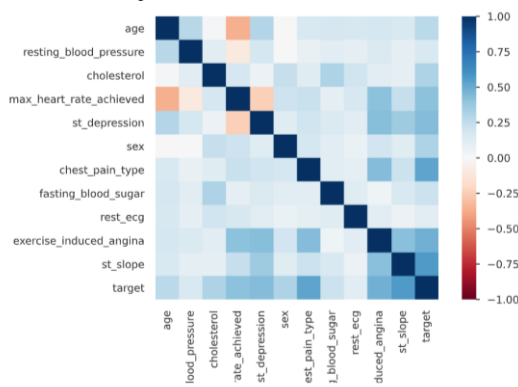
3.2. Имплементација система за предикцију срчаног удара

У овом поглављу је представљена имплементација система за предикцију срчаног удара, који се састоји од неколико модула (Слика 1).



СЛИКА 1 - СИСТЕМ ЗА ПРЕДИКЦИЈУ СРЧАНОГ УДАРА
Модули од највећег значаја јесу модул за претпроцесирање и анализу података, модул за обучавање и модул за генерисање резултата.

У оквиру модула за претпроцесирање и анализу података извршена је експлоративна анализа података и извршено је профилисање података. Као резултат профилисања скупа података помоћу *ProfilerReport*-а *Pandas* библиотеке, добијен је детаљнији опис инстанци, одређена корелација између различитих атрибута, као и информација о постојању дупликата и одступања. Утврђено је да улазни скуп података обухвата више мушких него женских пацијената, да је распон година широк – између 28 и 77 година, као и да има највише пацијената са крвним притиском између 100 и 150. Корелација атрибута дата је на Слици 2, са које се може видети да атрибут који директно утиче на појаву срчаног удара јесте тип бола у грудима, као и ангина изазвана вежбањем у корелацији са максимумом ST сегмента изазваног вежбањем. На појаву овог обољења најмање утиче максималан број откуцаја срца у корелацији са годинама пацијента.



СЛИКА 2 - КОРЕЛАЦИЈА АТРИБУТА

Применом експлоративне анализе (*Exploratory Data Analysis, EDA*) закључено је да је већа вероватноћа да се срчани удар јави код мушкараца, да на то највише утиче асимптоматичан тип бола у грудима и повишени притисак. Такође, већа је вероватноћа појаве срчаног удара уколико је вредност атрибута који се односи на максималан број откуцаја срца мања, што и јесте логично – што мањи број откуцаја срца, већи су изгледи да ће се срчани удар догодити.

Пре обучавања модела, скуп података подељен је у тренинг и тест скуп у односу 80:20. Тренинг скуп је улаз у модул за обучавање и над тим подацима је извршена петострука унакрсна валидација за оптимизацију хипер-параметра и примењена *SMOTE* техника. За потребе предикције срчаног удара која је описана у овом раду су изабрани следећи модели: логистичка регресија, наивни Бајес, метод потпорних вектора, модел К најближих суседа, метод насумичне шуме, *LightGBM*, *XGBoost* и *CatBoost*. Након обучавања, модели су примењени на тест скупу података.

Као резултат сваког од поменутих алгоритама машинског учења, у оквиру модула за генерисање резултата израчуната је *F*-мера - мера евалуације која комбинује прецизност и одзив. Одлучено је да би рачунање *Fbeta*-мере, при чему је *beta* параметар већи од 1, било погодније за сам предмет и циљ овог рада, јер би се на тај начин минимизовао број лажно негативних (*FN* – *false negative*) случајева. Резултати

добијени у овом модулу додатно су анализирани уз помоћ *SHAP* вредности и кластеровања.

4. РЕЗУЛТАТИ И ДИСКУСИЈА

Сви коришћени модели су тестирани на тест скупу података који је издвојен из првобитног скупа података. Након испробавања више вредности за параметар *beta*, дакле на основу мерења са различитим вредностима овог параметра, одлучено је да се користи *F3*-мера. Ова мера дала је боље резултате у односу на друге испробане *Fbeta*-мере и код ње је вредност лажно негативних случајева максимално минимизована. Вредности ове мере евалуације, као и вредности прецизности и одзива дате су у Табели 1.

Класификатор	Прецизност	Одзив	<i>F3</i> -мера
<i>CatBoost</i>	83.3%	99.2%	97.4%
<i>XGBoost</i>	82.8%	99.2%	97.3%
Метод насумичне шуме	75.4%	100%	96.8%
<i>LightGBM</i>	74%	99.2%	95.9%
Метод потпорних вектора	77.8%	97.6%	95.2%
Логистичка регресија	77.8%	97.6%	95.2%
Модел К најближих суседа	73.7%	97.6%	94.5%
Наивни Бајес	65.1%	99.2%	94.3%

ТАБЕЛА 1 - ПРИКАЗ ВРЕДНОСТИ ПАРАМЕТАРА ПРЕЦИЗНОСТ, ОДЗИВ И *F3*-МЕРА

Оно што можемо уочити из Табеле 1 јесте да су сви имплементирани модели машинског учења високих перформанси. Најбоље резултате дао је *CatBoost*, док наивни Бајесов модел постиже најслабије резултате за предикцију срчаног удара. Највише се истичу модели новије генерације а то су *CatBoost* и *XGBoost* са *F3*-мером респективно 97.4% и 97.3%. Ова два модела по својим перформансама надмашују и све моделе споменуте у сродним радовима који су описани у поглављу 2. Најбољи резултат постигнут је у истраживању [4] и он износи 96.1%. Скуп података коришћен у истраживању [4] је четири пута мањи у однос на скуп података коришћен у овом раду, док су обележја скупа идентична. Остали модели машинског учења истакнути у Табели 1, који су коришћени и у већ поменутих радовима, дали су боље резултате у односу на резултате тих радова. Иако скуп података коришћен у овом поглављу не спада по броју података у велике скупове, ипак је број инстанци које он обухвата већи од броја ентитета у скуповима података већине анализираних постојећих радова.

На Слици 3 приказан су резултати *SHAP* метода примењеног над подацима над којима је обучен *CatBoost* модел. Анализом ових резултата уочено је да, уколико код особе постоји асимптоматичан тип бола у грудима или је вредност *st_slope_flat* атрибута висока, велика је вероватноћа да се јави срчани удар. Притом, холестерол, године и количина шећера у крви значајно доприносе овој појави. Исто тако, особе без срчаних обољења углавном имају вредност 51 *normal* за *st_slope* атрибут и мања је вероватноћа да се срчани удар јави код жена.

Приликом кластеровања коришћена су два најпопуларнија алгорита, *K-Means* и *DBScan*, и то оба над

SHAP вредностима. Након извршеног кластеровања, генерисана су правила путем *ScopeRules*-а, како би се аутоматски добили одговори који атрибути и њихове вредности су најбитније за који кластер. Узимајући у обзир резултате кластеровања и пронађена правила, може се закључити да су неки од атрибута, који имају највећи утицај на појаву срчаног удара мушки пол, асимптоматичан тип бола у грудима, повишен холестерол и ST сегмент изазван вежбањем са вредношћу *flat*.



СЛИКА 3 - SHAP ВРЕДНОСТИ

У погледу повлачења паралеле између ових резултата и резултата сродних радова, закључује се да се они делимично поклапају. Неки од атрибута, који су се показали као најутицајнији исти су у овом и другим истраживаним радовима описаним у поглављу 2. Са друге стране, постоје и атрибути као што је холестерол, који се анализом *SHAP* вредности и кластеровања истиче као врло битан у овом раду, док га један од радова [3] наводи као потпуно нерелевантан.

5. ЗАКЉУЧАК

У раду је приказан систем за предикцију срчаног удара коришћењем алгоритама машинског учења. Решење овог проблема довело би до знања који то фактори највише утичу на појаву акутног инфаркта миокарда, што би могло помоћи у превенцији таквих случајева.

Доступни подаци су претпроцесирани и анализирани како би се добио детаљнији опис инстанци и установио однос између различитих атрибута. На тренинг скупу података обучено је више модела. Коришћене мере евалуације су *F3*-мера, прецизност и одзив. У овом раду добијена су одређена побољшања тачности у односу на радове који су описани у поглављу 2. Такође, коришћени су новији и напреднији модели машинског учења који су по својим перформансама надмашили све остале. Коришћене су *SHAP* вредности, које на прегледан начин приказују утицај различитих параметара на крајњи исход и које нису коришћене у радовима који

су анализирани. Извршено је и кластеровање са извођењем правила за сваки кластер, које даје нове информације о атрибутима и њиховим вредностима.

Један од главних недостатака овог рада представља релативно мали скуп података. Прикупљање додатних узорака, као и укључивање нових атрибута би побољшало резултата рада. Због испробавања већег броја различитих методологија у раду, остаје простор за даљу оптимизацију хипер-параметара модела који су имплементирани, зарад побољшања њихових перформанси. Машинско учење може да буде веома корисно јер може помоћи људима у откривању болести у раној фази на основу њиховог здравственог стања. Због свега наведеног, циљ овог рада и јесте био формирање модела који ће на основу одређених параметара и њихових вредности утврдити да ли постоји вероватноћа да особа добије срчани удар. Узимајући у обзир добијене резултате, машинско учење се показало као поуздано решење за предикцију ове болести. Треба напоменути да је веома битно одабрати одговарајуће атрибуте, класификаторе и методологије које ће се применити. Такође, могуће је и одрадiti симулацију коришћењем других алата како бисмо упоредили резултате и видели да ли они доприносе побољшању истих.

6. ЛИТЕРАТУРА

- [1] Sultana, M., Haider, A., & Uddin, M. S. (2016, September). Analysis of data mining techniques for heart disease prediction. In 2016 3rd international conference on electrical engineering and information communication technology (ICEEICT) (pp. 1-5). IEEE.
- [2] Nabeel, M., Awan, M. J., Raza, M., Muslih-Ud-Din, H., & Majeed, S. (2021, November). Heart Attack Disease Data Analytics and Machine Learning. In 2021 International Conference on Innovative Computing (ICIC) (pp. 1-6). IEEE.
- [3] Bhardwaj, A., Kundra, A., Gandhi, B., Kumar, S., Rehalia, A., & Gupta, M. (2019). Prediction of heart attack using machine learning. IITM
- [4] Waqar, M., Dawood, H., Dawood, H., Majeed, N., Banjar, A., & Alharbey, R. (2021). An efficient SMOTE-based deep learning model for heart attack prediction. Scientific Programming, 2021
- [5] Mohan, S., Thirumalai, C., & Srivastava, G. (2019). Effective heart disease prediction using hybrid machine learning techniques. IEEE access, 7, 81542-81554.
- [6] Heart-Disease-Dataset <https://www.openml.org/search?type=data&status=active&id=43682&sort=runs>

Кратка биографија:



Андријана Јеремић рођена је 1997. године у Београду, Република Србија. Основне академске студије завршила је 2019. године на Факултету техничких наука, на комбрани и мастер рад 2023. године из области Примењене рачунарске науке и информатика – Електронско пословање.

контакт: andrijana.jeremi@gmail.com

**PROTOTIP APLIKACIJE ZA PRUŽANJE POSLOVNIH INFORMACIJA I ANALIZA
MAKSIMALNE UPOTREBLJIVOSTI****PROTOTYPE OF APPLICATION FOR PROVIDING BUSINESS DATA AND ANALYSIS
WITH MAXIMUM USABILITY**

Miloš Topolić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – U ovom radu opisana je potreba za poslovnim informacijama i grupe korisnika koje imaju tu potrebu. Opisano je šta treba uzeti u obzir prilikom kreiranja aplikacije za pružanje poslovnih informacija i analiza. Razvijen je prototip aplikacije.

Ključne reči: Prototip aplikacije, poslovne informacije, analize

Abstract – This paper presents the need for business information and gives insight about the user groups which have that need. It highlights what needs to be considered when creating an application which provides business information and analysis. Application prototype is developed.

Keywords: Application prototype, business information, analysis

1. UVOD

U neprestano menjajućoj modernoj trgovini, potraga za preciznim i pravovremenim poslovnim informacijama postala je nezaobilazan aspekt strateškog donošenja odluka. Savremeni poslovni ekosistem napreduje potpomognut podacima, oslanjajući se na uvide iz različitih izvora kako bi podstakao rast, ublažio rizike i stekao prednost nad konkurentima. U tom kontekstu, ovaj rad za izazov ima da iskoristi bogatstvo poslovnih informacija koje su danas dostupne, unapređujući njihovu dostupnost i povećavajući njihovu vrednost kroz razvoj veb aplikacije.

Primamljivost poslovnih informacija u današnje vreme se ne može dovoljno naglasiti. Preduzeća, bez obzira na njihovu veličinu ili industriju kojoj pripadaju, preplavljena su morem podataka. Tržišni trendovi, inteligencija konkurenata, finansijski pokazatelji, uvid u lanac snabdevanja i ponašanje kupaca samo su deo ogromnog skupa podataka kojim preduzeća moraju da se kreću kako bi bila uspešna. Ovo obilje informacija eksponencijalno se proširilo u prethodnim godinama, odražavajući digitalnu transformaciju koja se širi kroz sve industrije. Digitalno doba je dovelo do toga da se podaci generišu, razmenjuju i konzumiraju izuzetnom brzinom.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivetić, red. prof.

U takvoj situaciji, kompanije ulažu značajne resurse kako bi pribavile informacije potrebne za osnovano donošenje odluka. Od naširoko poznatih kompanija i konglomerata, pa sve do novoosnovanih startapa, želje i potrebe za sveobuhvatnim i trenutnim poslovnim uvidima doveli su do partnerstva sa dobavljačima podataka, koji raspolažu rizičnima toliko bitnih poslovnih informacija.

Ipak, izazov koji se i dalje javlja jeste pristupačnost i upotrebljivost ovih rizičnih podataka. Biznisi su često preplavljeni u moru različitih izvora podataka, formata i platformi, što ne ide u prilog efikasnom dobavljanju bitnih informacija. Stoga, nije teško primetiti značaj razvijanja veb aplikacije koja neće samo standardizovati prezentaciju poslovnih informacija, već će i za prioritet imati pružanje izuzetnog korisničkog iskustva.

2. POTREBE ZA POSLOVNIM INFORMACIJAMA

Digitalno doba u kome živimo, i konstantni tehnološki napredak potpomažu industriju poslovne analitike, omogućavajući lakše prikupljanje podataka, ali i njihovu distribuciju. Potreba za poslovnim informacijama nije rezervisana samo za uzak krug ljudi i zanimanja. Tačni i uniformni poslovni podaci pogoduju profesionalcima iz raznih oblasti, ali i pojedincima.

2.1. Cilj aplikacije

Cilj aplikacije ove vrste jeste da krajnjim korisnicima aplikacije obezbedi informacije o poslovnim entitetima. Neke od informacija su dostupne svima na samom interfejsu aplikacije, dok se sve ostale informacije, analize i prognoze nalaze u izveštajima koje korisnici aplikacije mogu da kupe. Na kraju, informacije koje aplikacija pruža, odnosno prodaje, korisnicima pomažu u donošenju poslovnih odluka.

Da bi korisnici došli do informacija koje su im potrebne, aplikacija korisnicima omogućava pretragu poslovnih entiteta, odnosno kompanija. Pretraga po nazivu poslovnog entiteta omogućava korisnicima aplikacije da dođu do željenih informacija, ukoliko tačno znaju koju kompaniju traže. Osim pretragom po nazivu, korisnici aplikacije do poslovnih entiteta mogu doći i pretragom po drugim kriterijumima, kao što su delatnost kompanije, datum osnivanja, osnivačima kompanije i slično. Alternativni način kako korisnici aplikacije mogu doći do informacija o poslovnim entitetima jeste preko rang lista. Na primer, to može biti rang lista kompanija sa najvećim profitom u određenoj geografskoj oblasti.

Nakon što korisnici pronađu željene poslovne entitete, aplikacija im pruža uvid u osnovne informacije o poslovnim entitetu. Te informacije služe prvenstveno da bi naterale korisnike aplikacije da kupe neki od izveštaja o istom poslovnim entitetu. Iz tog razloga, većina informacija je sakrivena od korisnika, upućujući ga na kupovinu izveštaja koje aplikacija nudi, gde će korisnici aplikacije moći da vide ostatak dostupnih informacija.

U izveštajima se nalaze sve dostupne informacije o poslovnim entitetima. U to su uključeni profil kompanije, vlasnička struktura, filijale i ogranci. Finansijski podaci pružaju uvid u poslovanje nekoliko godina unazad, sa indikacijama kako su se određene vrednosti kretale. Prisutne su takođe i razne ocene kompanije, kreditni izveštaji, procene rizika i slično. Izveštaji zapravo ostvaruju primarni cilj aplikacije, a to je da pomognu korisnicima aplikacije da donesu određene poslovne odluke.

Obzirom da je predviđeno da aplikacija donosi profit, to se dešava prvenstveno prodajom izveštaja. Moguća je kupovina pojedinačnih izveštaja, ali je predviđen i mehanizam pretplate, koji bi pretplatnicima omogućio uvid u sve izveštaje. Mehanizam pretplate bi pogodio velikim biznisima, koji po svojoj prirodi sarađuju sa velikim brojem poslovnih entiteta, dok bi se kupovinom pojedinačnih izveštaja služili pretežno manji biznisi, ili pojedinci.

2.2. Kome je namenjena aplikacija?

Aplikacija je prvenstveno namenjena biznis korisnicima svih veličina, ali nije isključeno ni da u nekim slučajevima i pojedinci koriste aplikaciju. Najveću potrebu za korišćenjem aplikacije imaju oni korisnici koji sarađuju sa velikim brojem kompanija. Grupe ciljanih korisnika aplikacije su:

- Investitori i finansijski analitičari, kojima su često potrebne detaljne informacije o kompanijama na osnovu kojih će donositi odluke o ulaganju. Finansijski izveštaji, kreditne ocene, procene rizika i drugi indikatori koje aplikacija nudi pomogle bi im da procene finansijsko zdravlje i potencijal rasta kompanije.
- Vlasnici preduzeća i preduzetnici, bilo da se radi o većim ili manjim biznisima, mogu koristiti aplikaciju za istraživanje konkurenata, potencijalnih poslovnih partnera i trendova tržišta. Takođe bi imali koristi od uvida u standarde industrije, trendove rasta i ključne igrače u svojoj oblasti.
- Istraživači tržišta, odnosno profesionalci uključeni u istraživanje tržišta i analizu konkurencije mogu koristiti aplikaciju za prikupljanje podataka o kompanijama u određenim industrijama. Informacije koje aplikacija nudi pomogle bi im u razumevanju tržišnih trendova i ponašanja kupaca.
- Profesionalci u oblastima prodaje i marketinga, koji aplikaciju mogu da koriste kako bi identifikovali potencijalne klijente ili poslovne partnere na osnovu njihove finansijske

stabilnosti i pozicije u industriji. Marketing profesionalci mogu da iskoriste podatke koje aplikacija nudi kako bi prilagodili svoje strategije određenim kompanijama ili industrijama.

- Konsultanti i savetnici u raznim oblastima, kojima je ovakva aplikacija korisna za istraživanje potencijalnih ili trenutnih klijenata. Pre davanja saveta, mogu da prikupe informacije o finansijskom zdravlju i istoriji poslovanja kompanije.
- Akademski istraživači, naročito u oblasti ekonomije i biznisa, mogu koristiti ovakvu aplikaciju za prikupljanje podataka za akademske studije i radove. Pristup obimnim informacijama o kompanijama mogu podržati njihove analize i zaključke.
- Novinari, koji pokrivaju poslovne vesti ili istraživačke priče, mogu koristiti aplikaciju za prikupljanje osnovnih informacija o kompanijama, naročito kada izveštavaju o akvizicijama, finansijskim skandalima ili trendovima u industriji.
- Državni organi, koji ovakvu aplikaciju mogu da koriste za praćenje i procenu kompanija koje posluju u okviru njihove nadležnosti.
- Opšta javnost, čak i pojedinci bez posebnih profesionalnih uloga, aplikaciju mogu koristiti za istraživanje kompanija. Mogu istraživati potencijalne poslodavce ili mogućnosti investiranja. Pojedinci mogu koristiti aplikaciju i jednostavno ukoliko žele da budu informisani o različitim industrijama.

2.3. Koji su preduslovi za kreiranje aplikacije?

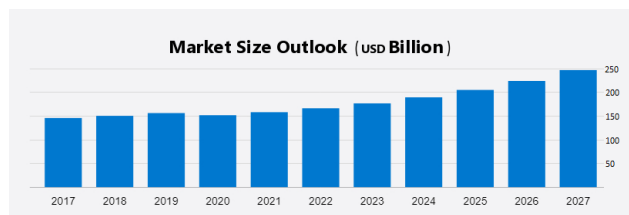
Osnovni resurs koji je potreban da bi se realizovala aplikacija ove vrste jesu podaci. Od količine, dostupnosti i tačnosti podataka direktno zavisi uspeh aplikacije. Prikupljanje podataka i stvaranje sopstvenih baza podataka od nule je jednostavno suviše skupo u svakom pogledu. Iz tog razloga, aplikacije ove vrste su najčešće realizovane u partnerstvu sa nekim od globalnih dobavljača biznis informacija i analitika. Partnerstva između dobavljača podataka i ovakvih aplikacija je korisno za obe strane. Sa jedne strane, kreatorima ovakvih aplikacija je rešen praktično nemoguć zadatak prikupljanja podataka, dok dobavljači podataka jačaju svoju poziciju i šire svoje tržište.

Dobavljači podataka nude podatke kao uslugu, takozvani Data as a Service model. Dobavljači podataka skupljaju, održavaju i omogućavaju pristup obimnim skupovima podataka. Konzumenti podataka putem API-ja koje dobavljači podataka nude mogu da se pretplate, i da njihove podatke integrišu u svoje sisteme ili aplikacije.

3. ANALIZA POSTOJEĆIH REŠENJA

Globalno tržište poslovnih informacija doživljava stalni rast, sa veličinom tržišta procenjenom na približno 150

milijardi dolara. Ovaj dinamični sektor beleži godišnju stopu rasta od 6.21%, što naglašava njegov sve veći značaj u savremenom poslovnom okruženju. Na Slici 1 može se videti vrednost tržišta poslovnih informacija, kao i prognoze kako će se veličina tržišta kretati u narednim godinama.



Slika 1: Vrednost tržišta poslovnih informacija

Ključni pokretač ovog rasta jeste sve veće oslanjanje na donošenje odluka zasnovano na podacima u svim industrijama. Biznisi svih veličina i svih sektora prepoznaju ključnu ulogu pravovremenih i tačnih informacija u cilju sticanja prednosti nad konkurencijom, upravljanja rizicima i prepoznavanja poslovnih mogućnosti. Štaviše, predviđeno je da će kombinovana godišnja stopa rasta dostići 8.2% u periodu između 2022. i 2027. godine, što znači kontinuiranu ekspanziju.

Za analizu postojećih rešenja, uzete su kao primeri dve kompanije koje imaju jako prisustvo u Srbiji, kada je reč o aplikacijama koje pružaju informacije o poslovnim entitetima. To su Bisnode i CompanyWall.

Bisnode i CompanyWall su dobro poznati igrači u oblasti poslovnih informacija i analitika. Bisnode se može pohvaliti širokim međunarodnim domašajem, opslužujući brojne evropske zemlje sveobuhvatnim paketom usluga, uključujući procene kreditnog rizika, istraživanja tržišta i poslovnu logiku. Poznat po kvalitetu podataka i pouzdanosti, Bisnode snabdeva biznise sa globalnim operacijama, nudeći opsežan skup podataka i detaljne analitike. Nasuprot tome, CompanyWall je specijalizovan za poslovne informacije određenog tržišta, pružajući lokalizovane uvide bitne prvenstveno tim istim tržištima, među kojima je i srpsko. Snaga CompanyWall-a leži u lokalnoj stručnosti i ažurnim informacijama, što ga čini značajnim za biznise koji posluju u Srbiji, ili koji su zainteresovani za srpsko tržište.

Dok Bisnode nudi globalnu pokrivenost sveobuhvatnim paketom usluga, CompanyWall je svoju prednost stekao nudeći lokalno relevantne i specifične podatke. Time se izbor između ova dva zasniva na geografskom fokusu i specifičnim potrebama za podacima.

4. TEHNOLOŠKI KONCEPTI

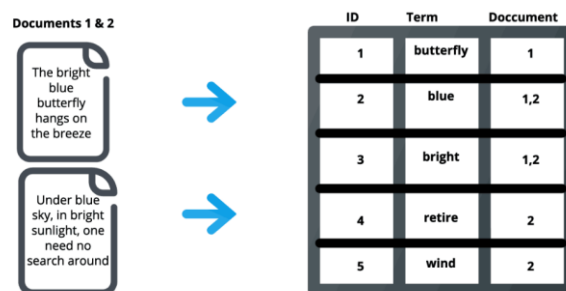
U vreme konstantnog digitalnog napretka, tehnološko znanje je često odlučujući faktor između uspeha i stagnacije. Razvoj veb aplikacije za pružanje poslovnih informacija predlagane u ovom radu predstavlja priliku da se iskoriste savremeni tehnološki koncepti i implementacije. Korišćenje pravih tehnologija ne samo da olakšava efikasnu isporuku podataka i analiza, već obezbeđuje konkurentsku prednost na tržištu gde je korisničko iskustvo od velikog značaja.

4.1. Pretraga

Pretraga velikog, potencijalno milionskog broja entiteta ključan je izazov u kreiranju aplikacije. Potencijalno rešenje, koje je veoma zastupljeno sa obimnom dokumentacijom, jeste Elasticsearch [1].

Elasticsearch funkcioniše tako što grupiše podatke u dokumente u JSON obliku. Dokumenti su po svojim karakteristikama grupisani u indekse, slično kao u bazu podataka. Elasticsearch za efikasnu pretragu koristi invertovani indeks, strukturu podataka koja mapira reči na njihove lokacije u dokumentima. Neki od osnovnih logičkih koncepta bitnih za razumevanje Elasticsearch-a su:

- Dokumenti su osnovne jedinice informacije koje mogu biti indeksirane u Elasticsearch-u. Skladište se u JSON formatu. Pandan dokumentu u relacionoj bazi podataka bi bio jedan red. Dokument ne mora biti samo tekst, već bilo kakav podatak koji se može naći u JSON-u.
- Indeksi su kolekcije dokumenata sličnih karakteristika. Indeksi su entiteti najvišeg nivoa po kojima se može pretraživati u Elasticsearch-u. Indeks se može zamisliti kao baza podataka u šemi relacionih baza podataka.
- Invertovani indeks je struktura podataka koja čuva mapiranje sadržaja na njegovu lokaciju u dokumentima. Pojednostavljeno, invertovani indeks je struktura nalik heš mapi koja upućuje od reči ka dokumentu. Koristeći invertovane indekse, Elasticsearch brzo pronalazi najbolja poklapanja za datu pretragu čak i u ogromnim setovima podataka. Na Slici 2 data je vizuelna reprezentacija invertovanog indeksa.



Slika 2: Vizuelna reprezentacija invertovanog indeksa

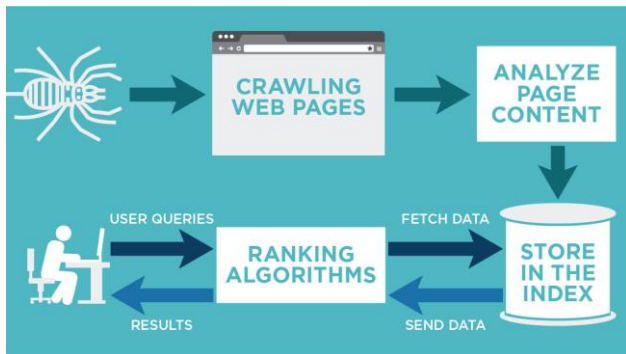
4.2 SEO

Search Engine Optimization, ili SEO, su tehnike kojima veb sajt ili veb stranica optimizuju, sa ciljem povećanja njihove vidljivosti od strane internet pretraživača.

Internet pretraživači koriste automatizovane programe koji se zovu crawler-i. Uloga crawler-a je da otkriju, analiziraju i indeksiraju veb stranice. Za vlasnike veb sajtova bitno je da znaju kako crawler-i rade. Držeći se najboljih SEO praksi, u velikoj meri se može povećati vidljivost i pristupačnost veb sajta među rezultatima pretrage internet pretraživača.

Postoje tri faze pretrage [2]. Prva faza, odnosno crawling, jeste pronalaženje stranica koje postoje na internetu. U

fazi indeksiranja, internet pretraživači pokušavaju da shvate o čemu je stranica. Ova faza uključuje procesiranje i analiziranje tekstualnog sadržaja i ključnih tagova i atributa stranice. Poslednja faza jeste serviranje rezultata pretrage. Kada korisnik unese upit u pretragu, internet pretraživači pretražuju indeks za stranice koje se slažu sa upitom, i vraćaju stranice za koje misle da su najkvalitetnije i najrelevantnije za korisnikov upit. Na Slici 3 sumiran je ceo ovaj proces.



Slika 3: *Crawling proces*

Neke od metoda kojima se može poboljšati SEO rejting sajta su:

- upotreba <title> tagova i metadata opisa,
- definisanje robots.txt fajla i
- kreiranje sitemap fajla.

4.3. Renderovanje aplikacije

Bitna odluka prilikom kreiranja veb aplikacije jeste odabir načina renderovanja aplikacije [3]. Neke od metrika koje su bitne u izboru načina renderovanja aplikacije jesu:

- TTFB (Time to First Byte), predstavlja vreme od kada korisnik klikne na link do trenutka kada dolazi prvi bajt sadržaja.
- FCP (First Contentful Paint), vreme kada sadržaj postaje vidljiv korisniku.
- INP (Interaction to Next Paint), metrika koja procenjuje da li stranica konzistentno reaguje na input korisnika.
- TBT (Total Blocking Time), metrika vezana za INP, računa vreme tokom kojeg je glavna nit bila blokirana prilikom učitavanja stranice.
- TTI (Time to Interactive), vreme kada stranica postaje interaktivna.

Postoje tri načina renderovanja veb aplikacija. Prvi je renderovanje stranica na serveru, što podrazumeva da se celokupni HTML generiše na serveru kao odgovor na korisničku navigaciju. Drugi način je statičko generisanje stranica, koje se dešava prilikom build-a aplikacije. Treći način jeste renderovanje stranica direktno u browser-u, odnosno renderovanje na klijentu.

Kako bi se dobili što bolji rezultati, usve češće se koristi hibrid kombinacije renderovanja na serverskoj i renderovanja na klijentskoj strani. Kada se uzmu u obzir

zahtevi aplikacije koja je predmet ovog rada, ima smisla odabrati hibridni pristup renderovanja aplikacije. Stranice za koje je bitno da imaju visok SEO rejting, trebalo bi generisati statički, ili na serverskoj strani. Stranice aplikacije koje su interaktivnije i za koje SEO nije bitan, pogodno je implementirati sa renderovanjem na klijentskoj strani. Iz tog razloga, aplikaciju bi bilo pogodno implementirati koristeći Next.js [4].

5. ZAKLJUČAK

Istraživanja su otkrila da biznisi širom sveta ulažu značajne finansijske resurse kako bi dobili pristup neophodnim poslovnim informacijama. Ove investicije, koje često dosežu i milijarde dolara godišnje, samo naglašavaju značaj podataka u modernim procesima donošenja odluka. Ipak, povrat ovih investicija sputan je fragmentacijom i složenošću izvora podataka, što kompanijama otežava da iskoriste pun potencijal raspoloživih informacija.

Rešenje predloženo radom nastalo je kao rezultat napora da se smanji jaz između gomila sirovih podataka i njihovih konzumenata. Dizajn i funkcionalnosti rešenja izvedeni su pod motivacijom da se dolazak do željenih informacija maksimalno olakša za krajnje korisnike aplikacije.

Iako predstavljeno rešenje ne predstavlja revoluciju u dobavljanju potrebnih poslovnih informacija, uzeti su u obzir nedostaci postojećih rešenja aplikacija ove vrste. Upravo ti nedostaci trebalo bi da budu prevaziđeni dizajnom i implementacijom predloženim u ovom radu, omogućavajući maksimalnu upotrebljivost rešenja.

6. LITERATURA

- [1] Elasticsearch, „Elasticsearch Guide“, <https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html>, Elasticsearch dokumentacija
- [2] Google, „In-depth guide to how Google Search works“, <https://developers.google.com/search/docs/fundamentals/how-search-works>, SEO koncepti i objašnjenja
- [3] J. Miller, A. Osmani, „Rendering on the Web“, <https://web.dev/rendering-on-the-web/>, Artikel o renderovanju web aplikacija
- [4] Vercel, „Rendering“, <https://nextjs.org/docs/pages/building-your-application/rendering>, Renderovanje Next.js aplikacije

Kratka biografija:



Miloš Topolić rođen je u Novom Sadu 1996. god. Osnovne akademske studije završio je 2019. godine na Fakultetu tehničkih nauka u Novom Sadu. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Elektronsko poslovanje odbranio je 2023.god.

PREGLED I IMPLEMENTACIJA ODABRANIH RETROAKTIVNIH STRUKTURA PODATAKA**OVERVIEW AND IMPLEMENTATION OF SELECTED RETROACTIVE DATA STRUCTURES**

Vera Kovačević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – Retroaktivne strukture podataka podržavaju modifikaciju niza operacija izvršenih nad datom strukturom. U ovom radu opisani su koncepti parcijalne i potpune retroaktivnosti, te je ilustrovana parcijalna retroaktivnost pomoću odabranih struktura: red, stek i red sa prioriteto. Glavni fokus rada jeste opis metoda implementacije odabranih struktura pomoću dvostruko povezane liste i strukture treap. Zaključeno je da se može postići vrijeme izvršavanja slično odgovarajućim standardnim strukturama.

Ključne reči: napredne strukture podataka, retroaktivne strukture podataka, red, stek, red sa prioriteto

Abstract – Retroactive data structures support modification of a series of operations performed on a given structure. In this paper, the concepts of partial and complete retroactivity are introduced, and partial retroactivity is illustrated using following structures: queue, stack and priority queue. The main focus of the paper is the description of methods of implementation of selected structures using doubly linked list and treap structure. It is concluded that execution time similar to the corresponding standard structures can be achieved.

Keywords: advanced data structures, retroactive data structures, queue, stack, priority queue

1. UVOD

Jedna od vrsta naprednih struktura podataka su vremenske strukture (eng. temporal data structures), koje uvode vrijeme kao novu dimenziju. Na osnovu toga na koji način koriste vrijeme kao novu dimenziju, one mogu biti perzistentne (eng. persistent data structures) ili retroaktivne (eng. retroactive data structures) [1].

Retroaktivne strukture podataka podržavaju modifikaciju niza operacija izvršenih nad datom strukturom. To podrazumijeva naknadno dodavanje, brisanje ili izmjenu operacija koje su izvršene u nekom trenutku u prošlosti. Na taj način se postiže da promjena koja je naknadno izvršena u prošlosti utiče na trenutno stanje, kao i na sva stanja između trenutka te promjene i sadašnjeg trenutka. Razlikuju se parcijalno retroaktivne strukture, koje dozvoljavaju modifikaciju podataka u prošlosti, ali se upiti mogu izvršavati isključivo u sadašnjem trenutku, i potpuno retroaktivne strukture, koje dozvoljavaju izvršavanje upita i u prošlosti [1].

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bila dr Dunja Vrbaški, docent.

Naredna poglavlja ovog rada baviće se upravo istraživanjem različitih metoda implementacije odabranih retroaktivnih struktura. U sljedećem poglavlju biće dat pregled naučnih radova od čijih autora potiče sama ideja, kao i nekih radova koji se nadovezuju na osnovnu ideju. Nakon toga će biti objašnjena opšta teorija o retroaktivnosti i opisani predloženi načini implementacije. Zatim, biće prikazana implementacija tri odabrane retroaktivne strukture, a nakon toga slijedi diskusija rezultata, primjena i prijedlozi za dalja unapređenja implementacije.

2. STANJE U OBLASTI

Ideja za izradu ovog rada potekla je od kursa „Napredne strukture podataka“ profesora Erika Demejna sa univerziteta MIT [1]. Profesor je osnovnu ideju o retroaktivnim strukturama podataka predstavio je u radu [2], gdje se uvodi koncept retroaktivnih struktura i opšta teorija o retroaktivnosti, a zatim se istražuju različite varijante retroaktivnosti, parcijalna i potpuna. Demonstrirano je da je moguće implementirati efikasne retroaktivne verzije reda, reda sa dva kraja, reda sa prioriteto i disjunktog seta. Nakon što su sredinom dvijehiljaditih godina objavljeni prethodno navedeni radovi, ova oblast se nastavila razvijati u pravcu unapređenja metoda implementacije i vremena izvršavanja, kao i primjene u rješavanju praktičnih problema.

Autori rada [2] opisali su opšti način transformacije parcijalno retroaktivne u potpuno retroaktivnu strukturu pod uslovom da postoji efikasna perzistentna verzija date strukture. Nekoliko godina kasnije, u radu [3], originalni autor osmislio je novi pristup baziran na hijerarhijskim kontrolnim tačkama (eng. *hierarchical checkpointing*), koji predstavlja opštu logaritamsku transformaciju parcijalno retroaktivne u potpuno retroaktivnu strukturu. Na osnovu teorema iznijetih u radu [3], konkretna implementacija navedene strukture opisana je u radu [4]. Korišćeno je samobalansirajuće binarno stablo pretrage (eng. *self-balanced binary search tree*).

Drugi pravac razvoja ove oblasti vezan je za praktičnu primjenu retroaktivnih struktura. U radu [7] ilustrovana je primjena retroaktivnog reda sa prioriteto u rješavanju problema pronalaženja najkraćeg puta u dinamičkim grafovima.

3. METODOLOGIJA

Uopšteno, može se reći da svaka struktura podataka sadrži niz modifikacija podataka i upita u toku vremena. Prema opštoj teoriji o retroaktivnosti, listu operacija izvršenih

nad strukturom definišemo kao $U = [u_{t_1}, \dots, u_{t_m}]$, gdje u_{t_i} predstavlja operaciju izvršenu u trenutku t_i i $t_1 < \dots < t_m$. Pretpostavka je da se u jednom trenutku izvršava samo jedna operacija.

Kao što je već rečeno, parcijalno retroaktivne strukture pored izvršavanja upita nad podacima u sadašnjem trenutku podržavaju i sljedeće dvije operacije:

- $Insert(t, insert(x))$ - dodavanje operacije dodavanja elementa x u niz operacija u trenutku t ,
- $Insert(t, delete())$ - dodavanje operacije brisanja operacije koja je izvršena u trenutku t .

Dakle, omogućeno je da se „putuje kroz vrijeme“ do nekog prethodnog stanja strukture i da se nad tim stanjem izvrši operacija koja će potencijalno promijeniti bilo koju operaciju između tog trenutka i sadašnjeg trenutka. Pri definisanju ovih operacija podrazumijeva se da se izvršavaju samo validne operacije, odnosno da se ne može izvršiti brisanje u trenutku t ako u tom trenutku nije izvršena nijedna operacija. Za razliku od parcijalne, potpuna retroaktivnost podržava i direktno posmatranje prošlosti, a operacija koja se dodaje može se definisati kao:

- $Query(t, search(x))$ - pretraživanje elementa x u bilo kom trenutku t u prošlosti.

3.1. Retroaktivni red

Red (eng. *queue*) je linearna struktura koja prati princip FIFO (eng. *First In, First Out*), što znači da će prvi dodati element biti prvi uklonjen. Podržava dvije primarne operacije: *enqueue*, koja dodaje element na kraj reda, i *dequeue*, koja uklanja element sa početka reda.

Red podržava operacije *enqueue(x)*, *dequeue()*, *front()*, koja vraća sljedeći element na redu za obradu i *back()*, koja vraća posljednji dodat element. Dakle, da bi se dobio retroaktivni red potrebno je podržati sljedeće operacije: $Insert(t, enqueue(x))$, $Insert(t, dequeue())$, $Query(t, front())$ i $Query(t, back())$.

Kada je u pitanju parcijalno retroaktivni red, vrijednost t kod upita uvijek će biti ∞ , odnosno sadašnji trenutak. Metoda za postizanje retroaktivnosti kod ove strukture data je u radu [2] i zasniva se na korišćenju dvostruko povezane pomoćne liste. U pomoćnoj listi se čuvaju sve operacije sortane po trenutku izvršavanja. Struktura takođe sadrži i dva pokazivača: F , koji pokazuje na prvi element u redu i B , koji pokazuje na posljednji element. Kada se retroaktivno doda operacija *enqueue*, novi element se dodaje u listu na odgovarajuće mjesto u zavisnosti od trenutka izvršavanja, i po potrebi se modifikuju pokazivači. Analogno tome se izvršava i uklanjanje operacije *enqueue* i dodavanje i uklanjanje operacije *dequeue*.

3.2. Retroaktivni red sa dva kraja

Kombinovanjem funkcionalnosti steka i reda nastaje red sa dva kraja (eng. *deque*, *double-ended queue*), poseban tip reda koji podržava dodavanje i uklanjanje elemenata i na početku i na kraju reda.

U retroaktivnoj verziji koriste se dvije pomoćne liste, U_L i U_R , koje čuvaju sve operacije na lijevom, odnosno desnom kraju sortirane po trenutku izvršavanja. Pored njih, koristi se i modifikacija binarnog stabla koja čuva prefiksne sume, kao i promjenljive L i R . Prefiksna suma je kumulativna suma elemenata niza, gdje svaki element rezultatnog niza predstavlja sumu svih

elemenata originalnog niza do posmatranog elementa. Promjenljive L i R predstavljaju pokazivače na trenutni krajnji lijevi i krajnji desni element. Za listu U_R operacijama dodavanja pridružuje se težina $+1$, dok se operacijama brisanja pridružuje težina -1 i te težine se čuvaju u binarnom stablu. Prema tome, vrijednost pokazivača R u trenutku t može se izračunati kao prefiksna suma elemenata sa vremenom izvršavanja do tog trenutka. Analogno tome koristi se i binarno stablo za prefiksne sume liste U_L .

3.3. Retroaktivni red sa prioritetom

Ova vrsta reda pored elemenata čuva i njihove prioritete, što omogućava da se prvo pristupi elementima sa višim prioritetom. Da bi se postigla parcijalna retroaktivnost potrebno je podržati sljedeće operacije: $Insert(t, insert(k))$, $Insert(t, delete-min())$, $Delete(t)$ i $Query(find-min())$. Kao pomoćna struktura koristi se skup elemenata koji se trenutno nalaze u redu sa prioritetom označen sa Q_now .

Ideja na kojoj se zasniva metoda implementacije parcijalno retroaktivnog reda sa prioritetom je da je kardinalitet skupa Q_now uvijek jednak razlici broja izvršenih operacija dodavanja i operacija brisanja. Kada se dodaje operacija $insert(k)$ u bilo kom trenutku, u skup Q_now se dodaje jedan element, a isto se dešava i kada se briše operacija $delete-min()$. U posebnom slučaju, kada se ubacuje element k sa najvišim prioritetom u datom trenutku i on bi trebalo da bude uklonjen nekom ranijom $delete-min()$ operacijom, umjesto njega se vraća element koji je bio zahvaćen operacijom brisanja prije dodavanja elementa k .

Definisane su formule za određivanje koji element treba biti ubačen u skup Q_now u odgovarajućim slučajevim. Da bi se lakše odredio taj element uvodi se koncept bridge. Ako je Q_t skup elemenata koji su sadržani u redu sa prioritetom u trenutku t i ako je on podskup skupa Q_now , onda postoji bridge u trenutku t .

Skup Q_now čuva se u binarnom stablu čiji čvorovi sadrže ključeve i pokazivače na odgovarajuće operacije. Slično kao kod retroaktivnog reda sa dva kraja, za određivanje bridge trenutaka koristi se modifikacija binarnog stabla koja čuva prefiksne sume.

4. IMPLEMENTACIJA I DISKUSIJA REZULTATA

Nakon analize različitih metoda za implementaciju retroaktivnosti kod linearnih struktura, odabrano je nekoliko njih za ilustraciju ovog koncepta, a to su dvije jednostavnije implementacije pomoću dvostruko povezane liste (parcijalno retroaktivni red i parcijalno retroaktivni stek) i jedna kompleksnija implementacija koja koristi i pomoćne strukture podataka (parcijalno retroaktivni red sa prioritetom).

4.1. Implementacija parcijalno retroaktivnog reda

Osnova implementacije parcijalno retroaktivnog reda je dvostruko povezana lista koja čuva elemente reda i pomoćna lista koja čuva sve *enqueue* i *dequeue* operacije. Klase koje se koriste su sljedeće:

Node – predstavlja jedan element liste,

Operation – predstavlja jednu operaciju,

enqueue ili *dequeue* i *PartiallyRetroactiveQueue* – predstavlja parcijalno retroaktivni red.

Klasu `PartiallyRetroactiveQueue` čine dvostruko povezana lista elemenata koji se trenutno nalaze u redu i lista svih operacija sortirana prema rastućem redsolijedu trenutaka izvršavanja. Dakle, atributi klase su referenca na prvi element reda (eng. `first`), referenca na posljednji element reda (eng. `last`) i lista operacija (eng. `operations`).

Implementirana struktura podržava sljedeće operacije:

- `insert_enqueue(value, time)` - dodavanje operacije `enqueue` u određenom trenutku,
- `insert_dequeue(time)` - dodavanje operacije `dequeue` u određenom trenutku,
- `delete_operation(time)` - uklanjanje operacije izvršene u određenom trenutku,
- `get_first()` - vraća prvi dodat element,
- `get_last()` - vraća posljednji dodat element.

4.2. Implementacija parcijalno retroaktivnog steka

Implementacija parcijalne verzije retroaktivnog steka se u velikoj mjeri oslanja na implementaciju parcijalno retroaktivnog reda, sa nekim izmjenama. Koriste se slične klase: `Node`, `Operation` i `PartiallyRetroactiveStack`.

Podržane su sljedeće operacije:

- `insert_push(value, time)` - dodavanje operacije `push` u određenom trenutku,
- `insert_pop(time)` - dodavanje operacije `pop` u određenom trenutku,
- `delete_operation(time)` - uklanjanje operacije izvršene u određenom trenutku,
- `get_top()` - vraća posljednji dodat element.

4.3. Implementacija parcijalno retroaktivnog reda sa prioritetom

Retroaktivni red sa prioritetom je kompleksnija fstruktura od prethodno navedenih, jer jedna retroaktivna operacija može dovesti do kaskadnog efekta i promjene životnog ciklusa drugih elemenata. Zbog toga je i njena implementacija kompleksnija i oslanja se na pomoćnu strukturu `treap` (`tree + heap`).

`Treap` je binarno stablo koje istovremeno ima svojstva binarnog stabla pretrage (eng. `binary search tree`, `BST`) i strukture `heap`. Čvorovi ovog stabla sadrže dvije vrijednosti: ključ (eng. `key`) i prioritet (eng. `priority`). Ključevi omogućavaju održavanje svojstava binarnog stabla pretrage, a prioriteti su nasumične vrijednosti koje omogućavaju održavanje svojstava strukture `heap`.

Klase koje se koriste u implementaciji parcijalno retroaktivnog reda sa prioritetom su sljedeće:

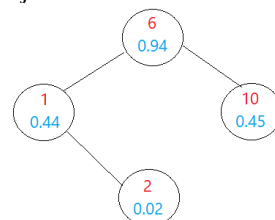
- `TreapNode` – predstavlja jedan element strukture `treap`,
- `Treap` – pomoćna struktura za čuvanje elemenata,
- `MinPrefixSumAggregator` – predstavlja agregator prefiksnih suma koji podržava čuvanje prefiksne sume i intervala nad kojim je izračunata,
- `ZeroPrefixTreap` – modifikacija strukture `treap` za čuvanje prefiksnih suma,
- `PartiallyRetroactivePriorityQueue` – predstavlja parcijalno retroaktivni red sa prioritetom.

Klasa `PartiallyRetroactivePriorityQueue` sadrži četiri strukture: `queue_now` – `treap` koji čuva sve elemente koji se trenutno nalaze u redu, `inserts` – `treap` koji čuva sve operacije dodavanja elementa, `deleted_inserts` – `treap` koji čuva sve operacije uklanjanja elementa i `bridges` – `ZeroPrefixTreap` koji čuva `bridge` trenutke sa dodatnim informacijama.

S obzirom na to da se ova implementacija oslanja na metodu opisanu u ovom radu, razlikuju se dva slučaja: ubacivanje elementa u red prilikom dodavanja novog elementa ili uklanjanja operacije brisanja i uklanjanje elementa iz reda prilikom brisanja najmanjeg elementa ili uklanjanja operacije dodavanja.

Prvi slučaj pokriven je metodama `add_insert` i `remove`. Metoda `add_insert` prvo privremeno dodaje novi element u strukturu `deleted_inserts`, kako bi on prilikom izbora koji element treba da se ubaci u red bio posmatran jednako kao i svi obrisani elementi. Iz strukture `deleted_inserts` prvo se pronađe posljednji `bridge` trenutak, a zatim element sa maksimalnom vrijednošću. To je element koji će se vratiti u `queue_now`, što znači da to ne mora uvijek biti onaj koji je posljednji ubačen, nego onaj koji ima najniži prioritet za izbacivanje.

Drugi slučaj čine operacije inverzne prethodno navedenim, a pokrivaju ga metode `add_delete_min` i `remove`, pri čemu se bira najmanji element reda koji je na redu za izbacivanje.



Slika 1 Prikaz pomoćne strukture `queue_now`

Na slici 1. je prikazan primjer pomoćne strukture `queue_now`, gdje su crvenom bojom označeni ključevi, a plavom prioriteti.

Metode koje podržava retroaktivni red su sljedeće: `add_insert(time, value, data)`, `add_delete_min(time)`, `remove(time)`.



Slika 2 Prikaz promjene stanja parcijalno retroaktivnog reda sa prioritetom

Na slici 2. dat je prikaz korišćenja parcijalno retroaktivnog reda sa prioritetom kroz vrijeme. Zelenom bojom označene su operacije dodavanja elementa, narandžastom operacije uklanjanja minimalnog elementa, a crvenom brisanje operacija.

4.4. Diskusija rezultata

Efikasnost retroaktivnih struktura podataka u literaturi se mjeri vremenskom složenošću algoritma, odnosno pomoću notacije „veliko O“ [5].

Kako se operacije kod retroaktivnog reda i steka čuvaju u pomoćnoj listi sortiranoj po trenucima izvršavanja, vrijeme izvršavanja jedne operacije podrazumijeva pronalaženje odgovarajućeg mjesta za novu operaciju u sortiranoj listi i ubacivanje samog elementa u listu. Ovaj korak se izvršava u vremenu $O(\log n)$, pri čemu je n

veličina liste. Međutim, samo ubacivanje ili brisanje elementa se izvršava u vremenu $O(n)$. Algoritam bi se mogao unaprediti korišćenjem nekog oblika binarnog stabla za čuvanje operacija umjesto liste, kako bi se vrijeme ubacivanja ili brisanja elementa svelo na logaritamsko vrijeme izvršavanja.

Što se tiče parcijalno retroaktivnog reda sa prioritetom, svaka operacija podrazumijeva dva koraka: pronalazak bridge trenutka i pronalazak maksimalnog elementa koji se ubacuje u red ili minimalnog elementa koji se uklanja iz reda. Oba koraka koriste pomoćne strukture zasnovane na implementaciji strukture treap, čije operacije dodavanja i uklanjanja se izvršavaju u logaritamskom vremenu, te je vrijeme izvršavanja jedne operacije $O(\log n)$.

5. PRIMJENA

U praksi postoji nekoliko slučajeva u kojima se mogu primijeniti retroaktivne strukture. Osnovna primjena je oporavak niza transakcija u slučaju greške brisanjem ili izmjenom transakcije koja je izazvala grešku u prošlosti. U radu [6] naveden je primjer gdje više meteoroloških stanica šalje podatke jednom centralnom sistemu i gdje u takvom sistemu retroaktivne strukture omogućuju naknadnu ispravku podataka, što utiče i na statističke podatke koji se računaju na osnovu tih zapisa, kao što je prosjek temperaturnih vrijednosti u nekom periodu.

Zatim, ove strukture se mogu primijeniti i u rješavanju nekih geometrijskih problema, kao na primjer lociranje tačaka u ravni (eng. *planar point location*) [1]. Takođe, moguća je i njihova primjena u dinamizaciji statičkih algoritama pomoću dinamičkog dodavanja i brisanja elemenata, kao na primjer dinamizacija algoritma *Dijkstra* [7].

6. ZAKLJUČAK

U ovom radu predstavljeno je istraživanje retroaktivnih struktura podataka, kao jedne od oblasti naprednih struktura podataka proučavanih na istoimenom kursu na univerzitetu MIT. Opisani su koncepti parcijalne i potpune retroaktivnosti, te je ilustrovana parcijalna retroaktivnost pomoću odabranih struktura: red, stek i red sa priotitetom.

Glavni fokus rada jeste opis metode implementacije parcijalno retroaktivnog reda i steka pomoću dvostruko povezane liste i implementacije parcijalno retroaktivnog reda sa prioritetom pomoću strukture treap. Zaključeno je da nije uvijek moguće implementirati efikasnu retroaktivnu strukturu, ali i da se ovim implementacijama može postići vrijeme izvršavanja slično odgovarajućim standardnim strukturama.

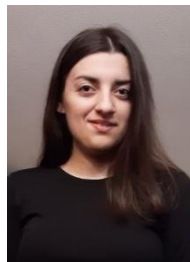
Jedan od pravaca u kojem bi se moglo nastaviti ovo istraživanje je prilagođavanje implementiranih struktura nekim praktičnim problemima.

Takođe, pošto su implementirane samo parcijalno retroaktivne verzije, sljedeći korak bio bi implementacija potpuno retroaktivnih struktura na osnovu opisane metodologije. Ovo bi otvorilo i mogućnost poređenja performansi implementiranih parcijalno i potpuno retroaktivnih struktura, kao i mogućnost izbora između parcijalno i potpuno retroaktivnih verzija u rješavanju praktičnih problema.

7. LITERATURA

- [1] MIT Open Courseware, „Advanced Data Structures“, Prof. Erik Demaine, ocw.mit.edu, Stranica kursa „Napredne strukture podataka“, pristupano: maj 2023.
- [2] E. D. Demaine, J. Ianoco, S. Langerma, „Retroactive Data Structures“, Proceedings of the Fifteenth Snnual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Lousiana, USA, January 11-14, 2004.
- [3] E. D. Demaine, T. Kaler, Q. Liu A. Sidford, A. Yedidia, „Polylogarithmic Fully Retroactive Priority Queues via Hierarchical Checkpointing“, MIT CSAIL Cambridge, MA, USA, 2015.
- [4] J. W. Andrade Junior, R. Duarte Seabra, „Fully Retroactive Priority Queues using Persistent Binary Search Trees“, *Journal of Computer Science*, 16(7), pp. 906-915, July 2020.
- [5] GeeksForGeeks, „Complete Guide On Complexity Analysis – Data Structure and Algorithms Tutorial“, geeksforgeeks.org, članak na sajtu GeeksForGeeks, pristupano: maj 2023.
- [6] S. Agarwal, P. Panwaria, „Implementation, Analysis and Application of Retroactive Data Structures“, Proc. of the Intl. Conf. on Advances in Computer Science and Electronics Engineering, 2012.
- [7] Sunita, D. Garg, „Dynamizing Dijkstra: A solution to dynamic shortest path problem through retroactive priority queue“, *Journal of King Saud University - Computer and Information Sciences* Volume 33, Issue 3, pp. 364-373, March 2021

Kratka biografija:



Vera Kovačević rođena je u Kozarskoj Dubici 1998. god. Fakultet tehničkih nauka u Novom Sadu, studijski program Softversko inženjerstvo i informacione tehnologije, upisala je 2017. godine. Nakon završenih osnovnih studija, 2021. godine, upisala je master akademske studije, smjer Softversko inženjerstvo i informacioni sistemi – Inteligentni sistemi.

kontakt:
verakovacevic98@gmail.com

KLASIFIKACIJA POKRETA ŠAKE NA OSNOVU EMG SIGNALA CLASSIFICATION OF HAND GESTURE BASED ON EMG SIGNALS

Ivana Miličić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – BIOMEDICINSKO INŽENJERSTVO

Kratak sadržaj – U ovom istraživanju realizovani klasifikatori za klasifikaciju pokreta šake na osnovu EMG signala mišića podlaktice u cilju realizacije komandnog interfejsa mioelektrične proteze šake bili su: mašina na bazi vektora nosača (engl. Support Vector Machine - SVM), klasifikator k – najbližih suseda (engl. K – Nearest Neighbors - KNN) i veštačke neuronske mreže (engl. Artificial Neural Network - ANN). Klasifikatori su obučavani za svakog ispitanika i za grupu. Redukcija dimenzionalnosti izvršena je primenom tehnike razlaganja na glavne komponente (engl. Principal Component Analysis - PCA) u cilju poboljšanja rezultata klasifikacije. Bolje performanse svih klasifikatora zabeležene su u pojedinačnim slučajevima, a PCA nije doprinela poboljšanju klasifikacije. SVM je u poređenju sa ostalim klasifikatorima kod većine ispitanika postizao najveće vrednosti tačnosti dok rezultati osetljivosti pokazuju da su SVM i KNN kod svih ispitanika prepoznavali kada je reč o pokretu istezanja ručnog zgloba. ANN u pojedinačnim slučajevima nakon PCA najmanja odstupanja među ispitanicima beleži prilikom klasifikacije pokreta istezanja ručnog zgloba.

Ključne reči: EMG, mioelektrične proteze šake, SVM, KNN, ANN, PCA

Abstract – This study implemented classifiers for hand movement classification based on EMG signals from the forearm muscles to create a command interface for a myoelectric hand prosthesis. The classifiers used were Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Artificial Neural Networks (ANN). The classifiers trained for each participant and the group as a whole. Principal Component Analysis (PCA) is used to improve classification results. All classifiers achieved better performance in individual cases. PCA did not contribute to the improvement of classification. SVM achieved the highest accuracy values for most participants compared to the other classifiers. Sensitivity results indicated that SVM and KNN can recognize wrist extension movements for all participants. After PCA, ANN showed low deviations among participants in classifying wrist extension movements.

Keywords: EMG, myoelectric hand prosthesis, SVM, KNN, ANN, PCA

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio prof. dr Nikola Jorgovanović.

1. UVOD

Amputacija gornjih ekstremiteta dovodi do ozbiljnih funkcionalnih i estetskih ograničenja što se prevazilazi korišćenjem protetičkih uređaja. U osnovi se njihov rad zasniva na snimanju elektromiografske aktivnosti mišića koja predstavlja električne impulse nastale tokom kontrakcija. Merenje signala omogućeno je postavkom površinskih elektroda iznad mišića od interesa, a nakon obrade mogu se dalje koristiti kao upravljački signali mioelektričnih proteza šaki.

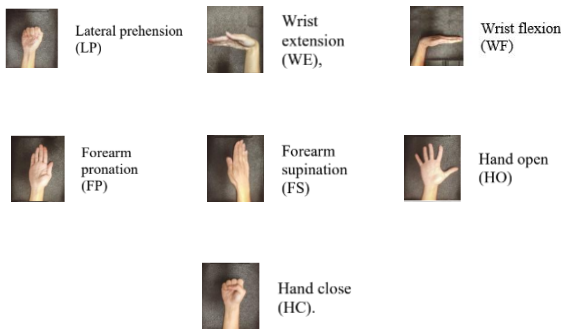
Noviji pristupi za dobijanje komandnih signala zasnivaju se na primeni klasifikacionih algoritama koji omogućavaju prepoznavanje različitih pokreta na osnovu mišićne aktivnosti većeg broja mišića. Nekoliko komercijalnih rešenja, kao što su COAPT i Myo Plus, primenjuju klasifikacione algoritme kako bi generisali odgovarajuće komande za rad mioelektričnih proteza šaki, a ovakav pristup korisnicima omogućava da povrate deo izgubljene funkcionalnosti doprinoseći im veći stepen samostalnosti [1].

2. ANALIZA BAZE PODATAKA

U ovom radu korišćena je javno dostupna baza *GrabMyo* i u okviru nje posmatrani su EMG snimci mišića zabeleženih sa gornjeg dela podlaktice mereni pomoću 8 parova elektroda (slika 1). Signali su filtrirani Betervortovim filtrom propusnika opsega četvrtog reda (10 – 500 Hz) i noć filtrom radi uklanjanja mrežnog šuma [2]. Posmatrana je grupa od 6 ispitanika koji su imali zadatak da izvedu 7 pokreta šake: bočni hvat (LP), istezanje ručnog zgloba (WE), savijanje ručnog zgloba (WF), supinacija podlaktice (FS), pronacija podlaktice (FP), otvaranje šake (HO) i zatvaranje šake (HC) (slika 2). Dodatni pokret predstavljao je EMG zabeležen u toku mirovanja. Izvođenje svakog pokreta trajalo je 5 sekundi, a ispitanici su imali 7 pokušaja da ih izvedu. Nakon svakog pokušaja sledila je pauza u trajanju od 10 sekundi.



Slika 1. Postavka elektroda za merenje (preuzeto iz [2])



Slika 2. Pokreti šake (preuzeto iz [2])

3. METOD

Na početku ovog rada navedeni su korišćeni klasifikacioni algoritmi. U nastavku poglavlja ukratko će biti navedene neke od glavnih karakteristika svakog od njih kao i karakteristike tehnike primenjene za redukciju dimenzionalnosti. SVM je efikasan algoritam koji pravi optimalne granice između posmatranih klasa i dovodi do postizanja visokih performansi klasifikatora i u visokodimenzionalnom prostoru obeležja. kNN donosi odluke zasnovane na najbližim susedima novog uzorka za klasifikaciju, dok ANN predstavlja model koji može naučiti kompleksne obrasce u podacima. PCA je tehnika redukcije dimenzionalnosti podataka koja izdvaja komponente koje sadrže najveći procenat varijanse u skupu podataka. Ova tehnika omogućava efikasno smanjenje broja obeležja dok čuva što više informacija, čime olakšava klasifikaciju [3].

U programskom okruženju *Python* implementirana su sva tri klasifikatora. Svih 7 pokušaja uzeto je u razmatranje za svakog ispitanika kao i za grupu. Za svakog ispitanika u bazu za trening ulazilo je prvih 5 pokušaja, a u bazu za test poslednja 2 pokušaja. Klasifikatori su se obučavali tokom 3 runde unakrsne validacije koja je za rezultat dala optimalne parametre za svakog od njih. Nakon obuke i testiranja izračunate su mere uspešnosti koje su posmatrane.

3.1. Mere uspešnosti klasifikatora

Tačnost je mera koja pokazuje udeo ispravno klasifikovanih uzoraka (engl. *True Positive - TP*) prema ukupnom broju uzoraka (engl. *True Negative - TN*, *False Positive - FP*, *False Negative - FN*) (jednačina 1) [3].

$$Tačnost = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

Osetljivost (engl. *true positive rate, recall*) meri udeo ispravno klasifikovanih uzoraka iz klase pozitivna (engl. *True Positive - TP*) (jednačina 2). Visoka osetljivost ukazuje na sposobnost klasifikatora da identifikuje pozitivne slučajeve [3].

$$Osetljivost = \frac{TP}{TP + FN} \quad (2)$$

Računanje osetljivost može se vršiti na dva načina: primenom mikroprosečnih i makroprosečnih mera. Mikroprosečne mere svode se na sumiranje TP, FP i FN za sve klase, nakon čega se osetljivost računa na standardan način koji je prikazan jednačinom 2.

Mere uspešnosti su pokazatelji efikasnosti različitih klasifikacionih algoritama. Tačnost pruža opštu sliku koliko je klasifikator tačan, dok osetljivost ukazuje njegovu sposobnost da pronade specifične slučajeve od interesa.

3.2. Izdvajanje obeležja

Obeležja dobijena iz frekvencijskog domena baziraju se na posmatranju spektra EMG signala. Promene unutar spektra koje nastaju usled izvođenja različitih pokreta mogu primenom odgovarajućih tehnika da pruže diskriminatorne informacije koje doprinose tačnijoj realizaciji klasifikacije [4]. Metoda koja je korišćena za dobijanje obeležja iz frekvencijskog domena naziva se tehnika frekvencijske podele (engl. *Frequency Division Technique - FDT*) [5].

EMG signal dobijen tokom trajanja svakog pokreta je segmentiran prozorom dužine 200 ms i preklapom od 150 ms. Zatim je za svaki prozor u okviru EMG signala jednog pokreta izračunata Furijeova transformacija čime je omogućen prelazak iz vremenskog u frekvencijski domen. Nakon toga je dobijena frekvencijska osa podeljena na 6 podopsega koji su redom: 20-92, 92-163, 163-235, 235-307, 307-378, 378-450 Hz [5]. Nakon toga se svaki od tih opsega predstavlja jednim brojem koji označava sumu amplituda svih frekvencijskih komponenta koje se nalaze unutar njega. Ukoliko vrednosti $f_{i,1}$ i $f_{i,E}$ predstavljaju gornju i donju vrednost frekvencije u okviru jednog podopsega, za svaki podopseg i -ta karakteristika se računa prema jednačini 3. L predstavlja broj podopsega na koje je spektar podeljen, $X(\cdot)$ označava amplitudu Furijeove transformacije za svaku i – tu frekvencijsku komponentu, a $\mathcal{F}[\cdot]$ je nelinearna transformacija koja se primenjuje radi izbegavanja malih vrednosti koje mogu uticati na rezultat klasifikacije. Ovaj postupak ponavlja se za svaki EMG signal pokreta i za svaki kanal, a kako ih je u ovom slučaju bilo 8 ukupan broj obeležja iznosi 48.

$$FDT_i = \mathcal{F} \left[\sum_{i=1}^{n_i} |X(f_{i,j})| \right], i = 1, 2, \dots, L \quad (3)$$

4. REZULTATI I DISKUSIJA

Dobijeni rezultati ukazuju da su manje vrednosti i tačnosti, ali i osetljivosti dobijene u grupnom slučaju i to kod sva tri posmatrana klasifikatora. Ovakav rezultat je bio očekivan iz razloga što se u pojedinačnim slučajevima uzimaju u obzir individualne karakteristike svakog ispitanika koje se zanemaruju kada se posmatra grupa. Iz tog razloga se u daljem tekstu rezultati tačnosti i osetljivosti dobijeni za grupu ispitanika neće spominjati. Vrednosti tačnosti klasifikacije razlikovale su se od ispitanika do ispitanika, što je i očekivano jer to nastaje kao posledica individualnih karakteristika. Primećuje se da se kod većine po vrednostima tačnosti izdvaja SVM. Ovaj rezultat i ne čudi s obzirom da je poznato da je glavna karakteristika ovog klasifikatora upravo postizanje dobrih performansi u visokodimenzionalnom prostoru, kakav je bio prisutan i u ovom problemu. Efekat redukcije dimenzionalnosti u većini slučajeva nije doveo do poboljšanja rezultata, a tamo gde jeste ona nisu mnogo

velika. Na primeru ispitanika 1, 3 i 6 primećuje se da je i pre i nakon PCA kod svakog isti klasifikator postigao najveću tačnost, ali uz malo povećanje vrednosti nakon redukcije. Kod prvog ispitanika to je bio kNN koji je pre redukcije postizao tačnost od 91.56 %, a nakon nje 91.95 %; kod trećeg je to SVM koji je pre redukcije imao tačnost od 92.74 %, a nakon nje ona je bila 93.35 %. SVM je bio najbolji klasifikator kod šestog ispitanika i tačnost je pre PCA iznosila 92.41 %, a nakon nje 92.57 %. Dobijene vrednosti prikazane su u tabeli 1 i 2.

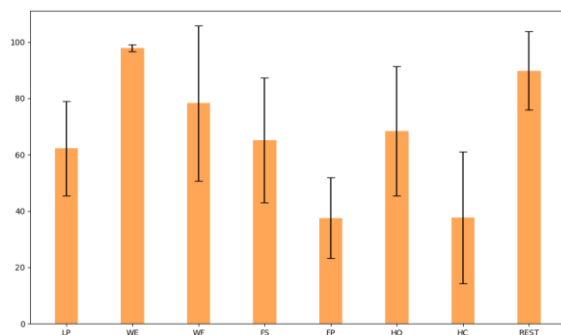
Klasifikator	Ispitanik 1	Ispitanik 3	Ispitanik 6
kNN	91.58 %	91.87 %	91.45 %
SVM	91.10 %	92.74 %	92.41 %
ANN	89.73 %	91.31 %	89.35 %

Tabela 1. Tačnosti klasifikatora za tri ispitanika pre primene PCA

Klasifikator	Ispitanik 1	Ispitanik 3	Ispitanik 6
kNN	91.95 %	91.36 %	92.05 %
SVM	91.47 %	93.35 %	92.57 %
ANN	91.74 %	92.02 %	91.53 %

Tabela 2. Tačnosti klasifikatora za tri ispitanika posle primene PCA

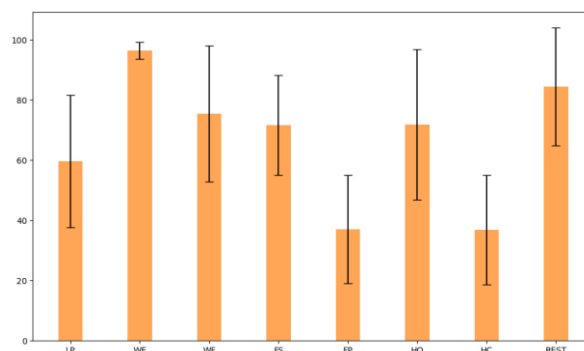
Postojeća literatura označava vrednosti osetljivosti veće od 80% kao kritične jer takve vrednosti obezbeđuju pouzdanost i ponovljivost klasifikacije i nad novim skupovima podataka [6]. Osetljivost predstavlja ključnu meru uspešnosti klasifikacije pokreta na osnovu EMG signala jer ukazuje na sposobnost klasifikatora da tačno prepozna određene pokrete. Analizirajući rezultate osetljivosti primećujemo da su SVM i kNN postigli visoku osetljivost kod svih ispitanika naročito u slučaju pokreta istezanje ručnog zgloba. Ovi rezultati prikazani su na grafikonima od 1 do 4, gde mala odstupanja vrednosti sugerišu da su ovi klasifikatori veoma dobro prepoznali ovaj pokret. Važno je istaći da je visoka osetljivost ostala i nakon primene PCA.



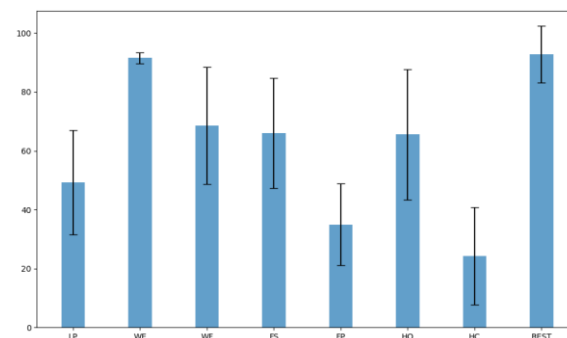
Grafik 1. Osetljivost SVM klasifikatora postignuta u pojedinačnim slučajevima za svaki pokret pre PCA

Pored toga, interesantno je da su SVM i kNN takođe pokazali visoku osetljivost prilikom detekcije EMG signala tokom mirovanja. Ovo je posebno važno jer ukazuje na njihovu sposobnost prepoznavanja stanja

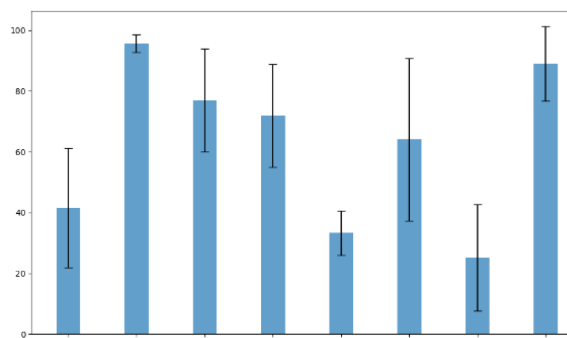
mirovanja mišića (na grafikonima pokret imenovan kao mirovanje (engl. *REST*)), što može biti korisno u medicinskim i kliničkim primenama. Međutim, za druge pokrete rezultati nisu pokazali tako visoke vrednosti osetljivosti.



Grafik 2. Osetljivost SVM klasifikatora postignuta u pojedinačnim slučajevima za svaki pokret posle PCA



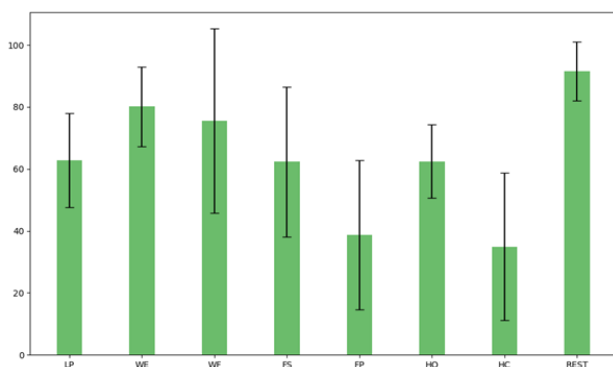
Grafik 3. Osetljivost kNN klasifikatora postignuta u pojedinačnim slučajevima za svaki pokret pre PCA



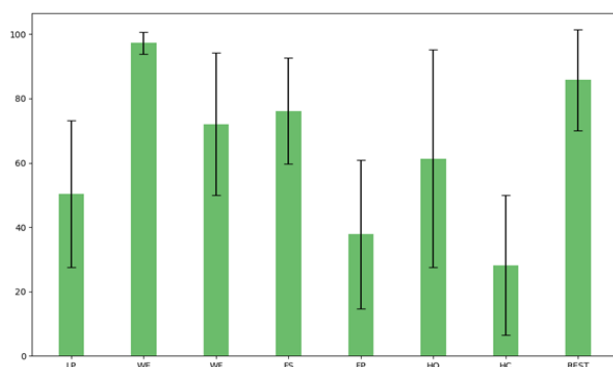
Grafik 4. Osetljivost kNN klasifikatora postignuta u pojedinačnim slučajevima za svaki pokret posle PCA

Analizirajući rezultate osetljivosti dobijene za ANN u pojedinačnim slučajevima primećeno je da nije izdvojen dominantan pokret kao što je to bio slučaj sa prethodno pomenutim klasifikatorima. Ovo sugerise da ANN može biti manje specifičan u prepoznavanju određenih pokreta i da je prisutna veća varijabilnost u rezultatima. Interesantno je primetiti da su najmanja odstupanja u osetljivosti primećena u slučaju klasifikacije pokreta u mirovanju i to pre primene PCA (grafik 5) što ukazuje na sposobnost ovog klasifikatora za prepoznavanje stanja mirovanja mišića. Međutim, nakon primene PCA najmanja greška je zabeležena tokom klasifikacije pokreta

istezanja ručnog zgloba. Ovo sugerise da je PCA doprinela poboljšanju osetljivosti ANN za ovaj pokret (grafik 6).



Grafik 5. Osetljivost ANN klasifikatora postignuta u pojedinačnim slučajevima za svaki pokret pre PCA



Grafik 6. Osetljivost ANN klasifikatora postignuta u pojedinačnim slučajevima za svaki pokret posle PCA

5. ZAKLJUČAK

U ovom istraživanju realizovani su klasifikatori za klasifikaciju pokreta šake na osnovu EMG signala mišića podlaktice. Dobijeni rezultati su pokazali prednost prilagođavanja klasifikatora svakom ispitaniku što dovodi i do postizanja boljih vrednosti mera uspešnosti klasifikatora.

SVM i KNN su postigli visoku osetljivost za pokret istezanja ručnog zgloba. Ovi rezultati ostaju i nakon primene PCA što ukazuje na njihovu stabilnost i pouzdanost. Takođe, zapaža se da su SVM i kNN efikasno detektovali EMG signal tokom mirovanja mišića. Međutim, za druge pokrete rezultati nisu bili dobri. Razlog tome može biti i korišćenje ograničenog skupa podataka što svakako u narednim istraživanjima treba uzeti u obzir u cilju poboljšanja rezultata klasifikacije.

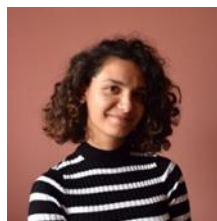
ANN je u odnosu na prethodna dva klasifikatora bio manje specifičan prilikom klasifikacije pokreta jer nije izdvojio jedan dominantan pokret kao što je to bio slučaj kod SVM i kNN klasifikatora. Primećeno je da je nakon PCA situacija bolja jer se značajno povećala osetljivost za pokret istezanja ručnog zgloba kod svih ispitanika.

Personalizovan pristup klasifikaciji pokreta na osnovu EMG signala može biti od suštinskog značaja u dijagnostičkim i rehabilitacionim aplikacijama. Buduća istraživanja treba da se usmere na razvoj novih ili unapređenje postojećih modela klasifikatora i razumevanje faktora koji utiču na individualne razlike u EMG signalima kako bi se unapredila pouzdanost i preciznost klasifikacije. Primena ovakve tehnologije može dovesti do unapređenja kvaliteta života pacijenata i korisnika kroz personalizovanu medicinsku negu.

6. LITERATURA

- [1] A. M. Simon, K. L. Turner, L. A. Miller, L. J. Hargrove, and T. A. Kuiken, 'Pattern recognition and direct control home use of a multi-articulating hand prosthesis', in *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*, 2019, pp. 386–391.
- [2] Jiang, N., Pradhan, A., & He, J. (2022). Gesture Recognition and Biometrics ElectroMyogram (GRABMyo) (version 1.0.2). *PhysioNet*.
- [3] Tijana Nosek, Branko Brkljac, Danica Despotovic, Milan Secujski, Tatjana Loncar-Turukalo. Praktikum iz mašinskog učenja, materijal sa predmeta Prepoznavanje oblika na osnovnim studijama Biomedicinskog inženjerstva
- [4] Angkoon Phinyomark, Pornchai Phukpattaranont, Chusak Limsakul (2012). Feature reduction and selection for EMG signal classification. *Expert Systems with Applications* 39, 7420–7431
- [5] A. Pradhan, N. Jiang, V. Chester, and U. Kuruganti, "Linear regression with frequency division technique for robust simultaneous and proportional myoelectric control during medium and high contraction-level variation," *Biomedical Signal Processing and Control*, vol. 61, p. 101984, 2020.
- [6] Panyawut Sri-iesaranusorn, Attawit Chaiyaroj, Chatchai Buekban, Songphon Dumnin, Ronachai Pongthornseri, Chusak Thanawattano and Decho Surangsrit (2021). Classification of 41 Hand and Wrist Movements via Surface Electromyogram Using Deep Neural Network. *Frontiers in Bioengineering and Biotechnology*

Kratka biografija:



Ivana Miličić rođena je u Kikindi 2000. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnika i računarstvo – Biomedicinsko inženjerstvo odbranila je 2023.god.
kontakt: i_milicic@hotmail.com

**RAZVOJ I POSTAVLJANJE BLOKČEJN-BAZIRANOG DISTRIBUIRANOG SISTEMA
ZA OCENJIVANJE STUDENATA****DEVELOPMENT AND DEPLOYMENT OF A BLOCKCHAIN-BASED DISTRIBUTED
SYSTEM FOR GRADING OF ASSIGNMENTS**

Aleksandar Ignjatijević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu će prvo biti predstavljeno istraživanje na temu Hyperledger Fabric blokčejna, a zatim osnove privatnih blokčejnova, kao i razvoj jedne takve mreže i njeno postavljanje. Pošto je akcenat rada na blokčejn i postavljanju istog, biće predstavljen razvoj Hyperledger Fabric mreže kao i postavljanje iste uz pomoć Kubernetes-a.

Ključne reči: distribuirani sistemi, blokčejn, Hyperledger Fabric, deployment, Kubernetes

Abstract – In this paper, an initial presentation of the research on the topic of Hyperledger Fabric blockchain will be provided, followed by the fundamentals of private blockchains, as well as the development and deployment of such a network. Since the focus of the study is on blockchain and its deployment, development of a Hyperledger Fabric network and its deployment utilizing Kubernetes will also be elucidated.

Keywords: distributed systems, blockchain, Hyperledger Fabric, deployment, Kubernetes

1. UVOD

Pojavom blokčejn tehnologije u svetu informacionih tehnologija 21. veka, otvorila se jedna od najintragantnijih tema koja, prema mišljenju nekih, postavlja osnovu za novu fazu interneta (veb 3.0). Dok je blokčejn prepoznat kao stub kriptovaluta, njegove ključne osobine poput postojanosti podataka, autonomnosti, tolerancije na greške, mogućnosti revizije, otpornosti i decentralizovanog izvršavanja otvaraju vrata za njegovu široku primenu u različitim domenima, uključujući vladine sektore, obrazovanje, finansije, industriju, zdravstvo, Internet stvari (IoT) i istraživačke sfere.

Jedan od osnovnih aspekata blokčejn tehnologije je sposobnost da omogući neizmenljivo, transparentno i sigurno čuvanje podataka, čime se otvara put ka novim paradigmama upravljanja informacijama. Univerzitetsko okruženje, koje je bogato različitim formama akademskih izvora i informacija, može imati značajne koristi od primene ove tehnologije.

U ovom radu biće opisana implementacija i postavljanje platforme za upravljanje i čuvanje podataka o seminar-

skim radovima studenata. Razvijeno rešenje omogućava praćenje svakog koraka u procesu izrade, evaluacije i arhiviranja seminarskih radova, garantujući ne samo autentičnost i originalnost radova, već i pružajući uvide u akademski razvoj svakog studenta kroz vreme. Pored razvoja i implementacije, biće predstavljen i proces postavljanja ovog rešenja.

Rešenje je implementirano pomoću Hyperledger Fabric-a, koji predstavlja jedan od vodećih projekata u okviru Hyperledger-a i nudi modularnu i proširivu arhitekturu, omogućavajući efikasno upravljanje i verifikaciju podataka. U savremenim računarskim okolinama, orkestracija kontejnera postala je imperativ za efikasno upravljanje resursima i automatizaciju desetina, stotina ili čak hiljadama kontejnera. Postavljanje blokčejna izvršeno je uz pomoć Kubernetes-a koji se izdvojio kao vodeći orkestracioni sistem, pružajući robustne mehanizme za automatizaciju, skaliranje, raspoređivanje i operativnu koordinaciju kontejnerizovanih aplikacija u klasterima.

2. Razvoj i postavljanje privatne blokčejn mreže

Ovo poglavlje posvećeno je teorijskim osnovama razvoja i postavljanja privatnog blokčejna. To pre svega podrazumeva osnove Hyperledger Fabric-a i Kubernetes-a.

2.1. Hyperledger Fabric

Hyperledger Fabric predstavlja platformu za razvoj distribuiranih rešenja koja se zasnivaju na blokčejn tehnologiji. Hyperledger Fabric poseduje modularnu arhitekturu dizajniranu kako bi podržala različite vrste implementacija komponenti. Cilj ovakvog arhitekturnog dizajna je prilagođavanje različitim scenarijima korišćenja ove platforme. Pored pomenutih pogodnosti, prednost ove platforme leži u visokom nivou skalabilnosti i sigurnosti, odnosno privatnosti transakcija, koja je preko potrebna u poslovnim okruženjima. Razvoj Hyperledger Fabric platforme je započeo IBM, a verzija 1.0 je objavljena sredinom 2017. godine, dok se u trenutku pisanja ovog rada, Hyperledger Fabric nalazi na verziji 2.5.4. objavljenoj u avgustu 2023. godine. Do danas, ni jedna druga implementacije distribuiranog dnevnika transakcija nije korišćena od strane svetskih giganata poput AWS-a, Azure-a, IBM-a, Google i Oracle-a [1].

Jedan od bitnih koncepata ove platforme je postojanje različite vrste čvorova u okviru mreže. Pored toga, Hyperledger Fabric je uveo koncept kanala. Kanali imaju svrhu da omoguće privatnost između članova mreže, zato što poseduju sopstvene dnevnike transakcija. Korišćenje

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dušan Gajić, vanr. prof.

kriptografskih metoda omogućava samo članovima, koji učestvuju u transakciji, dekriptovanje i izvršavanje transakcije [2].

Kao što je pomenuto, *Hyperledger Fabric* nudi mogućnost odabira implementacije konsenzus algoritma u zavisnosti od potreba sistema, što predstavlja svakako pozitivnu stranu ovog projekta.

2.2. Kubernetes

Kubernetes je prenosiva, proširiva platforma otvorenog koda za upravljanje kontejnerskim radnim opterećenjima i uslugama. *Kubernetes* olakšava i deklarativno konfiguiranje i automatizaciju i ima veliki ekosistem koji se brzo razvija. *Kubernetes* usluge, podrška i alati su široko dostupni [3].

Ime *Kubernetes* potiče iz grčkog, što znači kormilar ili pilot. *K8s* kao skraćenica je rezultat brojanja osam slova između "K" i "s". *Google* je otvorio projekat *Kubernetes* 2014. godine. *Kubernetes* kombinuje preko 15 godina *Google*-ovog iskustva u vođenju proizvodnih radnih opterećenja u velikom obimu sa najboljim idejama i praksama zajednice [3].

Kontejneri su dobar način za pakovanje i pokretanje aplikacija. U produkciji je potrebno upravljati kontejnerima koji pokreću aplikacije i osigurati da nema prekida u radu. Na primer, ako jedan kontejner prestane da radi, drugi kontejner treba da se pokrene umesto njega [3]. Ovakvo ponašanje bi bilo znatno lakše kontrolisati sistemom, a *Kubernetes* je upravo jedan takav sistem. *Kubernetes* pruža okvir za otporno pokretanje distribuiranih sistema. Brine se o skaliranju i prelasku na rezervnu varijantu aplikacije, nudi oblike primene i još mnogo toga.

3. IMPLEMENTACIJA BLOKČEJN REŠENJA

U ovom poglavlju biće opisana implementacija blokčejn rešenja, kao i proces puštanja u rad jednog takvog rešenja. Biće opisana infrastruktura na kojoj je izvršeno postavljanje, kao i delovi sistema. Implementacija blokčejn rešenja je izvršena uz pomoć *Hyperledger Fabric*-a, pamentni ugovori su pisani u programskom jeziku *Go*, a blokčejn mreža je postavljena uz pomoć *K3s Rancher*-ove distribucije *Kubernetes*-a.

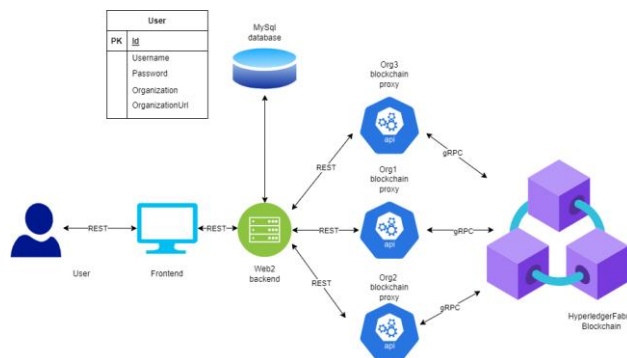
3.1. Open-rev platforma i arhitektura rešenja

Open-rev je platforma za deljenje seminarskih radova kreirana sa idejom da studenti, profesori, kao i potencijalni poslodavci, na jednom mestu imaju uvid u profesionalni razvoj studenata kroz njihove seminarske radove. Svaki rad je moguće oceniti, a kako bi se smanjila mogućnost pristrasnog ocenjivanja radova, uvedeno je i ocenjivanje samih recenzija radova, kao i kredibilitet ocenjivača.

Blokčejn implementacija se svodi na migraciju već postojećeg *web2* rešenja *Open-rev*, koje se nalazi na URL-u <https://www.open-rev.com>. Sastavni delovi sistema predstavljaju:

- korisnički interfejs,
- *back-end*,
- *MySQL* bazu podataka,
- *Hyperledger Fabric* proxy i,
- *Hyperledger Fabric* blokčejn mreža.

Na slici 1 dat je prikaz arhitekture rešenja.



Slika 1. Arhitektura Open-rev platforme

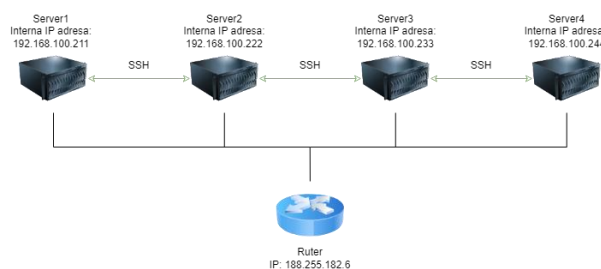
Hyperledger Fabric blokčejn mreža je organizovana u 3 organizacije. Svaka organizacija ima po jednog *peer*-a.

Već postojeći *back-end* se koristi za autentifikaciju i autorizaciju korisnika, kredencijali se drže centralizovani u *MySQL* bazi podataka, a komunikacija sa *blokčejnom* se odvija preko *proxy* servera. Postavljen je jedan *proxy* server, a planirano je postavljanje dodatna dva *proxy* server-a koji odgovaraju ostalim organizacijama koje su učesnici u blokčejn mreži. U pomenutoj centralizovanoj bazi, pored kredencijala i uloge u sistemu, nalazi se i podatak o tome kojoj organizaciji pripada korisnik, te se uz pomoć njega vrši rutiranje na *proxy* server odgovarajuće organizacije.

3.2. Infrastruktura

Infrastruktura se sastoji od četiri identična servera koji se nalaze na istoj lokalnoj mreži. Kako bi *bootstrap*-ovanje *Kubernetes* klastera, kao i funkcionisanje istog, bilo moguće, omogućena je *SSH* komunikacija između svih servera.

Na svim serverima je instalirana Ubuntu 20.04 distribucija *Linux*-a. Serveri raspolažu sa 16Gb RAM-a, kao i sa 2Tb internog skladišta, i procesorom Intel Xeon CPU E3-1220 v3 koji poseduje 4 jezgra i radi na 3.10GHz. Serveri su povezani na isti ruter i dostupni uz pomoć posvećenog port-a otvorenog na ruteru. Na slici 2 dat je prikaz infrastrukture sistema.



Slika 2. Prikaz infrastrukture

Pored *on-premise* infrastrukture, iznajmljen je i *droplet* na platformi *DigitalOcean*. *Droplet* raspolaže sa jednim virtualnim jezgrom, 25Gb skladišta i 1Gb RAM-a. Operativni sistem na *droplet*-u je Ubuntu 20.04. Svrha *droplet*-a je da bude *NFS* server (engl. *Network File System*), na kome će se čuvati sertifikati organizacija, blokovi postanka (engl. *Genesis blocks*) i sve ostalo što je osetljivo za samu blokčejn mrežu. *NFS* je distribuirani sistem datoteka. Svrha *NFS*-a je da obezbedi deljenje

datoteka putem mreže, na način koji veoma podseća na deljenje datoteka koje su na lokalnom sistemu [4].

3.3. Pametni ugovori

Pametni ugovori predstavljaju transakciju, odnosno parče koda koji sadrži neku transakcionu logiku koja kontroliše stanje objekta u okviru *world state*-a. Postoje dva tipa transakcije:

- *Evaluate* transakcija i
- *Submit* transakcija.

Evaluate transakcije su transakcije koje ne menjaju *world state*, već samo zahtevaju neke podatke koji se već nalaze u *world state*-u. Pored toga što ne menjaju *world state*, *evaluate* transakcije se ne pamte na *ledger*-u.

Submit transakcije su prvenstveno transakcije koje menjaju *world state*, a mogu i da vraćaju podatke sa *ledger*-a. O izvršavanju ovih transakcija se čuva podatak na *ledger*-u.

3.4. Proxy server

Kako je *Golang* programski jezik korišćen za implementaciju pametnih ugovora, odlučeno je da se i za implementaciju *proxy* servera koristi isti programski jezik. Za implementaciju *proxy* servera korišćena je *Gin* biblioteka za *web* servere napisana u *Golang*-u.

Komunikaciju između *proxy* servera i *blokčejna* je realizovana upotrebom *gRPC*-a, kao i sva ostala komunikacija u okviru *blokčejna*. *gRPC* (engl. *Remote Procedure Call*) je *open-source RPC framework* koji se koristi za komunikaciju visokih performansi između servisa razvijen od strane *Google*-a.

3.5. Testna mreža i njene komponente

U toku razvoja softverskog rešenja korišćena je blokčejn testna mreža [5]. Za tu svrhu iskorišćen je *docker-compose* alat, a testna mreža se podiže i spušta po potrebi. *Hyperledger Fabric* na svom *Github* nalogu poseduje repozitorijum po imenu *hyperledger/fabric-samples* i u okviru tog repozitorijuma priložene su implementacije osnovnih funkcionalnosti koje nudi *Hyperledger Fabric* privatna blokčejn mreža, poput primera pametnih ugovora, primera pozivanja transakcija od strane eksternog sistema, kao i primer testne mreže i pokretanja iste [6]. Testna mreža, ponuđena od strane *Hyperledger*-a sastoji se od dve organizacije koje poseduju po jednog *peer*-a, te je bilo potrebno proširiti postojeće ponuđene *docker-compose yaml* datoteke, kao i skripte koje se koriste za pokretanje testne mreže, kako bi bilo moguće podržati tri organizacije sa tri *peer*-a. Kako je sadržaj potrebnih datoteka za postavljanje testne mreže poprilično obiman, neće biti prikazan u okviru ovog rada, međutim sve potrebne datoteke nalaze se na repozitorijumu projekta.

4. POSTAVLJANJE MREŽE UZ POMOĆ KUBERNETES-A

Postavljanje *Hyperledger Fabric*-a je obavljeno uz pomoć *Kubernetes* orkestratora. S obzirom da postavka *Hyperledger Fabric*-a podrazumeva postavljanje više komponenti, proces postavke je izvršen u nekoliko faza. Proces postavke se može okvirno podeliti na:

- fazu pripreme,

- fazu postavljanja *Fabric CA*,
- fazu generisanja sertifikata i početnih artifakata,
- fazu postavljanja *orderer*-a i *peer*-ova,
- fazu instaliranja *chaincode*-a,
- fazu postavljanja *proxy* servera i
- fazu postavljanja *ingress* kontrolera.

Manifesti se primenjuju sa izvršavanjem komande *kubectl apply -f naziv_manifesta* u okviru direktorijuma u kom se nalazi manifest. Manifesto su dostupni na *Github* repozitorijumu projekta.

4.1. Faza pripreme

U okviru faze pripreme izvršeno je par ključnih koraka koji su potrebni za dalje faze. Prvi korak u ovoj fazi podrazumeva prebacivanje već postojećih skripti, kao i prebacivanje ključnih konfiguracionih *yaml* datoteka na zakupljeni *droplet* na platformi *DigitalOcean*, koji služi kao *NFS* server. Nakon uspešnog prebacivanja svih potrebnih datoteka, odvojen je deo skladišta na klasteru u vidu *persistentVolume*-a (skr. *PV*). Naredni korak u pripremnoj fazi je postavljanje *persistentVolumeClaim*-a (skr. *PVC*). *PVC* predstavlja zahtev za korišćenje *PV*-a. U narednim koracima, svi *Deployment*-i *orderer*-a, *peer*-ova i *CA*-eva će koristiti taj kreirani *PVC*.

4.2. Faza postavljanja Fabric CA

Ova faza podrazumeva kreiranje *Deployment* i *Service* manifesta za *Fabric CA* koji je zadužen za svaku organizaciju pojedinačno, kao i samog *orderer*-a. *Deployment* je korišćen kako bi se omogućilo potencijalno skaliranje ili ažuriranje podešavanja postavljenih *Pod*-ova. Drugi deo postavljanja *Fabric CA* predstavlja postavljanje *Service*-a, kako bi ti *Pod*-ovi koji su kreirani uz pomoć *Deployment*-a, bili dostupni na mreži unutar klastera

4.3. Faza generisanja sertifikata i početnih artifakata

U ovoj fazi postavljanja *Hyperledger Fabric*-a generišu se sertifikati koji će koristiti *orderer* i organizacije, a za to je kreiran manifest koji će kreirati *Job*. *Kubernetes Job* kreira *Pod* kako bi obavio određeni posao, a to u ovoj konkretnoj fazi predstavlja generisanje potrebnih sertifikata. U tu svrhu koristi se *hyperledger/fabric-ca-tools Docker* slika za kreiranje kontejnera u kome će se izvršiti skripte zadužene za generisanje pomenutih sertifikata. U narednom koraku ove faze generišu se početni artifakti uz pomoć drugog *Job*-a. Rezultat ovog *job*-a su datoteke *genesis.block*, *Org1MSPanchors.tx*, *Org2MSPanchors.tx*, *Org3MSPanchors.tx* i *mychannel.tx* datoteke.

4.4. Faza postavljanja orderer-a i peer-ova

Poput postavljanja *Fabric CA*, postavljanje *orderer*-a se svodi na kreiranje *Deployment*-a i *Service*-a na veoma sličan način kao i pomenuti *Fabric CA*. Za kreiranje kontejnera koristi se *hyperledger/fabric-orderer Docker* slika. Kako je *Hyperledger Fabric* platforma za rešenja sa distribuiranom glavnom knjigom i mrežom sa kontrolisanim pristupom, uređivački servis je veoma bitna komponenta sistem i čini ga grupa *orderer* čvorova. Ovaj servis je zadužen za uređivanje transakcija u blokove. Postavljeno je pet *orderer Pod*-ova uz pomoć *Deployment*-a i *Service*-a.

4.5. Faza instaliranja *chaincode*-a

U pripremnoj fazi, prebačen je *chaincode* direktorijum u okviru koga se nalazi direktorijum *packaging*. Sadržaj ovog direktorijuma predstavljaju tri *tgz* datoteke: *basicv1-org1.tgz*, *basicv1-org2.tgz* i *basicv1-org3.tgz*. Ove datoteke se sastoje iz dve datoteke: *code.tar.gz* i *metadata.json*. Instalacija *chaincode*-a koristi ove datoteke. Rezultat instalacije je *hash* instaliranog *chaincode*-a i on je potreban za uspešno postavljanje *Deployment*-a eksternih *chaincode* servera, te je veoma bitno zapamtiti taj *hash*. Nakon uspešnog instaliranja *chaincode*-a, sledeći korak u ovoj fazi je postavljanje *Deployment*-a i *Service*-a za *chaincode* servere. Nakon postavljanja *Deployment*-a i *Service*-a, potrebno je potvrditi (engl. *approve*) *chaincode* na svakom od *peer*-ova. Kako svaki *peer Pod* koristi i kontejner za *CouchDb*, moguće je povezati se na interfejs *CouchDb*-a na adresi http://localhost:5984/_utils/.

4.6. Faza postavljanja proxy servera

Poput postavljanja *orderer*-a i *peer*-ova, postavljanje proxy servera se svodi na kreiranje *Deployment*-a i *Service*-a, na veoma sličan način kao i pomenuti resursi. Za kreiranje kontejnera koristi se prethodno kreirana *Docker* slika *oakenknight/openrev-gateway-api*. Uz pomoć *environment* varijabli postavlja se identitet organizacije kojoj pripada određen proxy server, kanal na koji će se povezivati, naziv *chaincode*-a koji će biti korišćen i podaci o *MinIO* bucket-u koji će biti korišćen za skladištenje radova. Port koji će biti otvoren na *api-staging Pod*-u je port 80. *MinIO* predstavlja distribuirani datotečki sistem postavljen sa svrhom perzistentnog čuvanja seminarskih radova na *Kubernetes* klasteru.

4.6. Faza postavljanja proxy servera

Ingress kontroler predstavlja objekat koji služi za eksterni pristup *Service*-ima u okviru *Kubernetes* klastera, a kako je potrebno da proxy server koristi *back-end Open-rev* platforme, potrebno je postaviti i ovaj *Kubernetes* resurs. Prilikom kreiranja *Ingress* kontrolera, postavlja se putanja na kojoj će biti dostupan *API* koji se izlaže uz pomoć *Ingress* kontrolera. Kao *Ingress* kontroler korišćen je *Nginx Ingress* controller.

5. ZAKLJUČAK

U okviru ovog rada, razmatrana je primena blokčejn tehnologije u okruženju akademskih institucija, konkretno u procesu praćenja i upravljanja seminarskim radovima studenata. Postavljanje takvog sistema implementirano je koristeći *Hyperledger Fabric* platformu, koja je omogućila kreiranje transparentnog i sigurnog okruženja za čuvanje i praćenje podataka o seminarskim radovima. Ovaj sistem je pokazao značajne prednosti u pogledu integriteta, revizije i decentralizovanog pristupa podacima, dok je u isto vreme obezbedio mehanizme za očuvanje privatnosti i autentičnosti informacija.

Implementacijom blokčejn rešenja uz pomoć *Hyperledger Fabric* platforme, omogućena je pouzdana i transparentna razmena informacija među relevantnim učesnicima - studentima, mentorima i akademskim osobljem. *Hyperledger Fabric* se pokazao kao robustna platforma koja može da odgovori na zahteve takvog sistema u smislu performansi, sigurnosti i skalabilnosti.

Ovaj rad otvara prostor za dalja istraživanja u primeni blokčejn tehnologije u obrazovnim institucijama, i pruža temelj za razvoj proširenih sistema koji mogu uključivati i druge aspekte akademskog života. Takav razvoj može doprineti stvaranju efikasnijih, transparentnijih i sigurnijih okruženja za akademsku zajednicu, čime se unapređuje celokupni akademski proces. Takođe, analiza izazova i prednosti u implementaciji blokčejn tehnologije pružila je vredne uvide koji mogu voditi buduće inicijative u oblasti obrazovanja i tehnologije.

Kod dostupan na linku:

<https://github.com/OakenKnight/OpenRev>.

6. LITERATURA

- [1] Hyperledger Fabric Whitepaper, https://8112310.fs1.hubspotusercontent-na1.net/hubfs/8112310/hyperledger_fabric_whitepaper.pdf, pristupljeno 06.09.2023.
- [2] What is Hyperledger Fabric, <https://hyperledger-fabric.readthedocs.io/en/latest/blockchain.html>, pristupljeno 06.09.2023.
- [3] Kubernetes documentation: Overview, <https://kubernetes.io/docs/concepts/overview/>, pristupljeno 19.09.2023
- [4] Network File System, <https://www.ibm.com/docs/en/aix/7.1?topic=management-network-file-system>, pristupljeno 19.08.2023
- [5] Fabric Samples, <https://github.com/hyperledger/fabric-samples>, pristupljeno 19.08.2023.
- [6] Hyperledger Fabric documentation: Test network, https://hyperledger-fabric.readthedocs.io/en/release-2.5/test_network.html, pristupljeno 19.08.2023.

Kratka biografija:



Aleksandar Ignjatijević rođen je u Novom Sadu 19. aprila 1998. godine. Fakultet tehničkih nauka u Novom Sadu, studijski program Računarstvo i automatika upisao je 2017. godine. Nakon završenih osnovnih studija, 2021. godine, upisao je master akademske studije iz iste oblasti.

kontakt: a.ignjatijevic@gmail.com

POGON ZA VIZUELIZACIJU ANIMIRANIH 3D MODELA**3D ANIMATED MODEL VIEWING ENGINE**Jovan Ivošević, *Fakultet tehničkih nauka, Novi Sad***Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**

Kratak sadržaj – Rad se bavi analizom i implementacijom problema vezanih za prikazivanje animiranih 3D modela. Predstavljena je struktura modela datoteke i način učitavanja i čuvanja podataka prilagođen za efikasnu upotrebu. Analizirani su metodi kompresije, animacije i prikazivanja takvih modela. Za dalji rad razmatrane su mogućnosti implementiranja podrške virtualne realnosti i potencijalni problemi koje takva implementacija donosi.

Ključne reči: Animacija, 3D, model, pogon, virtuelna, realnost, inverzna, kinematika, kompresija, gltf

Abstract – The paper attempts to provide an analysis of and solutions to problems related to rendering 3D animated models. An analysis of the current leading model structure is provided, alongside with solutions to loading and storing the data in an efficient manner. Methods of compression, animation and rendering animated 3D models are considered. Virtual reality support is considered, alongside potential problems such support can lead to.

Keywords: Animation, 3D, model, engine, virtual, reality, inverse, kinematics, compression, gltf

1. UVOD

Sa visokom zastupljenošću tehnike u svakom aspektu života, sasvim je prirodna pojava potrebe za prikazom trodimenzionalne grafike.

U kompjuterskim video igrama i filmskoj industriji 3D grafika služi kao alat za vizuelno, ne nužno realno, predstavljanje sveta autora prikazivanog dela. U simulatorima i u medicini služi kao alat za što verniji prikaz posmatranog domena te industrije. Za takve alate najbitniji su brzina odziva i vernost prikaza.

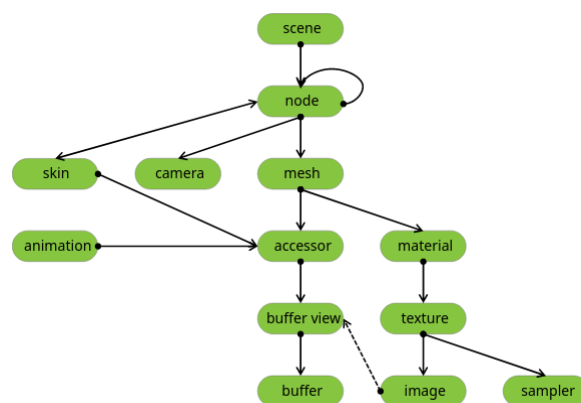
Predstava objekata u digitalnom svetu najčešće se realizuje pomoću modela. Učitavanje 3D modela, parsiranje podataka u internu reprezentaciju zgodnu za upotrebu na platformi, proračuni vezani za prikazivanje tih modela i prikazivanje modela su glavni problemi koji će se razmatrati u ovom radu.

2. OPIS GLTF FORMATA

Olakšani prenos i opisivanje 3D modela su prednosti glTF formata [1]. Kako se izveze iz softvera za 3D modeli-

ranje, u istom formatu se može i uvesti u aplikaciju. Datoteka može biti predstavljena u binarnom “.glb” ili tekstualnom “.gltf” obliku. Tekstualni oblik pogotovo je zgodan za prenos između više različitih API-eva jer koristi JSON predstavu podataka. Dodatna prednost JSON predstave jeste u njenoj razumljivosti za ljude.

Slika 1 prikazuje shemu glTF formata. Može se primetiti da datoteka ne čuva samo informacije o modelu, već postoji i podrška za čuvanjem informacija koje opisuju scenu u kojoj se model nalazi, kao i “kameru” iz čije perspektive je namenjeno posmatranje te scene. Za potrebe ovog rada će se posmatrati samo oni delovi formata koji direktno opisuju model, a ne i samu scenu ili kameru.



Slika 1. Shema glTF formata

Datoteka je logički organizovana kao niz stabala. Veze između čvorova svakog stabla fizički su realizovani preko indeksa čvorova (eng. nodes) u nizovima podataka. U JSON predstavi, na primer, svi čvorovi svih scena nalaze se u nizu “nodes” u korenu glavnog objekta modela. Struktura stabla dobija se tako što svaki čvor ima niz svojih potomaka, ukoliko taj čvor ima potomke. Niz potomaka ne sastoji se od samih objekata čvorova, već od celobrojnih nenegativnih vrednosti koje predstavljaju redni broj čvora u nizu čvorova.

3. KOMPRESIJA

Kao glavni vid kompresije i dekompresije glTF modela koristi se Draco biblioteka razvijena od strane Guglovog “Chrome Media” tima. Draco je softver otvorenog koda koji služi za komprimovanja 3D geometrijskih mreža i oblaka tačaka. U trenutku pisanja ovog rada, izovni kod biblioteke služi se sledećim metodama kompresije:

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivetić, red. prof.

- Edgebreaker algritam [2, 3],
- aritmetičko kodiranje i
- rANS.

3.1 ANS

Asymmetric Numeral Systems je metod kodiranja razvijen od strane Jaroslava Dude 2009. Ideja je relativno prosta: poruka se posmatra kao niz simbola i kodira se u jedinstven prirodan broj x . Ukoliko je x mali broj biće potrebno manje bitova za njegovo kodiranje i obrnuto ukoliko je x veći broj. Drugi način posmatranja jeste da se statističke osobine same poruke mogu upotrebiti tako da se učestalije poruke kodiraju u manji broj bitova, a manje učestale poruke u veći broj i tako dostigne efikasna kompresija. Ovaj metod je aproksimacija aritmetičkog kodiranja i nudi sličnu efikasnost kompresije, s tim da ANS ima prednost u odnosu na aritmetičko kodiranje u tome što se informacija smešta u jedan prirodan broj, umesto u dva broja koja opisuju opseg.

Motivšući primer za ANS može biti primer kodiranja sekvence binarnog broja $b_1b_2...b_i$ u kom je b_1 najznačajnija cifra. Inkrementalno kodiranje, to jest pretvaranje binarnog broja u prirodni broj dekadnog sistema moguće je izvršiti kodirajućom funkcijom C definisanom formulom 1:

$$C(x_i, b_{i+1}) := 2x_i + b_{i+1} = x_{i+1} \quad (1)$$

Funkcija C kao rezultat daje novu kodiranu vrednost x_{i+1} . Analogno tome inverzna funkcija za inkrementalno dekodiranje broja nazad u binarnu reprezentaciju definisana je formulom 2:

$$D(x_{i+1}) := (\lfloor \frac{x_{i+1}}{2} \rfloor, x_{i+1} \bmod 2) = (x_i, b_{i+1}) \quad (2)$$

Funkcija D prima binarnu predstavu kao ulaz i vraća prethodnu binarnu predstavu i poslednji dodati bit. Kako bi se izbegli problemi pri kodiranju niza nula, npr. "000" potrebno je x_0 dodeliti početnu vrednost "1". Ukoliko je učestalost jedinica i nula uniformna, to jest $P(0) = P(1) = 0.5$, primenom formule 1 dobija se da je entropija jednog bita jednaka jedinici, to jest da entropija kodirane poruke raste za jedan pri kodiranju svakog bita.

Pošto optimalna kodirajuća funkcija uzima prethodno kodiranu vrednost x_i i novi simbol b_{i+1} i kodira te vrednosti sa minimalnom entropijom, važi formula 3:

$$\begin{aligned} H(x_{i+1}) &= H(C_{opt}(x_i, b_{i+1})) \\ &= H(x_i) + H(b_{i+1}) \\ &= \log_2(x_i) - \log_2(p_{b_{i+1}}) \\ &= \log_2\left(\frac{x_i}{p_{b_{i+1}}}\right) \\ \Rightarrow C_{opt}(x_i, b_{i+1}) &\approx \frac{x_i}{p_{b_{i+1}}} \end{aligned} \quad (3)$$

Dakle ukoliko kodirajuća funkcija prati ovakvu aproksimaciju, onda se garantuje optimalno kodiranje. Uzimajući neuniformnu distribuciju parnih i neparnih brojeva (nula i jedinica) nekog alfabeta, tako da je verovatnoća pojave jedinice $p_1 = p < 1 - p = p_0$, gde je p_0 verovatnoća pojave nule, poželjno je da se u prvih N brojeva pojavi $N \cdot p$ jedinica i $N \cdot (1-p)$ nula. Kodirajuća funkcija koja ispunjava ovaj uslov prikazana je u formuli 4 uz svoju dekodirajuću funkciju.

$$\begin{aligned} C(x_i, b_{i+1}) &= \begin{cases} \left\lceil \frac{x_i+1}{1-p} \right\rceil - 1 & \text{za } b_{i+1} = 0 \\ \left\lfloor \frac{x_i}{p} \right\rfloor & \text{inace} \end{cases} \\ b_{i+1} &= \lceil (x_{i+1} + 1) \cdot p \rceil - \lceil x_{i+1} \cdot p \rceil \\ x_i &= \begin{cases} x_{i+1} - \lceil x_{i+1} \cdot p \rceil & \text{za } b_{i+1} = 0 \\ \lceil x_{i+1} \cdot p \rceil & \text{inace} \end{cases} \end{aligned} \quad (4)$$

Ovako definisane kodirajuće i dekodirajuće funkcije rađaju ANS varijantu definisanu za binarni alfabet poznata kao uABS - uniform Asymmetric Binary System. Funkcionisanje ovakvog sistema kodiranja zasnovano je na principu deljenja N prirodnih brojeva na parne i neparne (odgovarajući nulama i jedinicama) proporcionalno zastupljenosti jedinica i nula koji čine poruke za kodiranje.

Proširujući ovaj koncept na alfabet koji se sastoji od više od 2 različita simbola dobija se rANS - range variant Asymmetric Numeral Systems metoda kompresije. Za alfabet od n simbola s potrebno je izvršiti kvantizaciju raspodele simbola sa 2^n , to jest podeliti skup od N prirodnih brojeva na 2^n podskupova. Ovim se garantuje aproksimacija verovatnoće pojave simbola s - p_s na $f_s/2^n$, gde je f_s prirodan broj koji označava broj elemenata odgovarajućeg podskupa za simbol s . Izračunavajući funkciju raspodele $\Phi(s) = f_0 + f_1 + \dots + f_{s-1}$ dobijamo kodirajuće i dekodirajuće funkcije u formuli 5:

$$\begin{aligned} C(x_i, s_{i+1}) &= \left\lfloor \frac{x_i}{f_s} \right\rfloor \cdot 2^n + \Phi(s) \\ &\quad + (x_i \bmod f_s) \\ s_{i+1} &= \text{symbol}(x_{i+1} \bmod 2^n) \\ | \Phi(s) &\leq x_{i+1} \bmod 2^n < \Phi(s+1) \\ x_i &= D(x_{i+1}) = f_s \cdot \left\lfloor \frac{x_{i+1}}{2^n} \right\rfloor - \Phi(s) \\ &\quad + (x_{i+1} \bmod 2^n) \end{aligned} \quad (5)$$

Kodirajuća funkcija pronalazi odgovarajući opseg vrednosti prirodnih brojeva i "smešta" simbol u taj opseg, kodirajući ga jedinstvenim prirodnim brojem.

Dekodirajuća funkcija prima jedinstven prirodni broj i obrnutim procesom pronalazi njemu odgovarajući simbol. Zanimljiva posledica kvantizacije sa 2^n jeste ta što se celokupno množenje, deljenje i sve moduo operacije mogu svesti na binarno pomeranje što čini programsku implementaciju efikasnijom.

4. SKELETALNI SISTEM MODELA

Skeletalni sistem modela je mehanizam koji omogućuje pravilno prenošenje animiranog pokreta s jedne celine modela na drugu. Sačinjen je od hijerarhijski organizovanih zglobova. Nad zglobovima je moguće primeniti prostorne transformacije. Svaki zglob utiče na temena mrežne strukture modela u nekoj meri. Jedan zglob može uticati na više temena i na jedno teme može uticati više zglobova. Najčešće je dozvoljeno da najviše četiri zglobova utiču na jedno teme. Primena prostornih transformacija nad zglobovima odražava se na temena koja su pod uticajem tog zgloba i na druge zglobove u sprezi. Uticaj prostorne transformacije zgloba nad temenom proporcionalan je koeficijentu uticaja tog zgloba nad tim temenom. Za svako teme čuvaju se identifikatori zglobova koji utiču na to teme i koeficijenti uticaja tih zglobova. Koeficijenti uticaja zglobova nad temenom takođe se nazivaju i “težine” (eng. weights). Suma svih težina jednog temena mora da bude jednaka jedinici.

Zglobovi su u hijerarhiji organizovani kao stablo. Jedan zglob može imati više potomaka ili ni jednog, a najviše jednog roditelja. Uticaj prostorne transformacije nad jednim zglobovima prenosi se na njegovog roditelja, ukoliko postoji. Roditelj dalje prenosi uticaj transformacije na svog roditelja i uticaj se tako prenosi kroz stablo sve do korenskog zgloba, uključujući i njega. Ovaj način prenošenja pokreta od listova stabla ka korenu naziva se inverzna kinematika.

Svaki zglob poseduje lokalnu i globalnu transformaciju i inverznu transformaciju veznosti. Pozicija, orijentacija i veličina zgloba u njegovom lokalnom prostoru definisane su u lokalnoj transformaciji, a u globalnom prostoru definisane su u globalnoj transformaciji. Svaki 3D model ima svoju početnu poziciju, orijentaciju i veličinu - početnu pozu. Dok je model u početnoj pozi, animator vezuje zglobove skeletalnog sistema za model, te se ova poza takođe naziva i “poza vezivanja” (eng. bind pose). Matrica transformacije veznosti zglobova smešta zglobove u inicijalna stanja poze vezivanja. Premeštanje temena iz prostora mrežne strukture u lokalni prostor zgloba za koji je vezan vrši se množenjem inverznom matricom transformacije veznosti. Pošto je inverzna matrica veznosti poznata u trenutku pravljenja modela, često se izvozi zajedno sa modelom kako bi se izbegli skupi proračuni invertovanja matrica u toku njene učestale upotrebe pri prikazivanju modela. Lokalni prostor zgloba je relativan u odnosu na prostor njegovog roditelja, osim za korenski zglob koji nema roditelja. Iz ovog sledi da su transformacije nad zglobovima relativne u odnosu na roditelja, a ne na globalni prostor u kom se model nalazi.

Novo stanje temena v u odnosu na staro stanje temena v dobijeno pod uticajem transformacije T_{world} zgloba j koeficijentom uticaja w dobija se formulom 6.

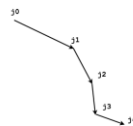
$$v' = w \cdot T_{world}(j) \times v \quad (6)$$

Novo izračunato stanje temena je deformacija temena pod uticajem transformacije zgloba. Deformacija dobijena primenom proračuna transformacije zglobova inverznom

kinematikom daje relativno realistične rezultate i izbegava pojavu priklještena mrežne strukture. Realizam pokreta i deformacije mrežne strukture moguć je zahvaljujući načinu dobijanja globalne transformacije T_{world} zgloba. Množenjem globalne transformacije roditelja p lokalnom transformacijom zgloba j dobija se globalna transformacija zgloba j kao što se može videti u formuli 7.

$$T_{world}(j) = T_{world}(p) \times T_{local}(j) \quad (7)$$

Pošto je globalna transformacija korenskog zgloba jednaka njegovoj lokalnoj transformaciji koja je najčešće inicijalizovana u samom modelu i pošto se sve uzastopne transformacije daljih potomaka mogu posmatrati kao relativni “pomeraji” u odnosu na roditeljsku transformaciju - sledi da se konačna transformacija nekog zgloba potomka dobija rekurzivnim prolaskom kroz stablo počevši od korenskog zgloba. Ilustracija pojednostavljenog primera zglobova nalazi se na slici 2.



Slika 2 Pojednostavljena hijerarhija zglobova

Na slici 2 zglobovi su označeni kao j_i . Zglob j_0 je korenski zglob, a j_1 njegov zglob potomak. Isto tako j_2 je potomak od j_1 i tako dalje sve do j_4 . Da bi se dobila globalna transformacija zgloba j_4 potrebna je globalna transformacija zgloba j_3 . Isto tako, za treći zglob je potrebna globalna transformacija drugog. Jasno je da je u pitanju rekurzivni proračun počevši od korenskog zgloba. Formula 8 prikazuje postupke dobijanja globalnih transformacija svih zglobova.

$$\begin{aligned} T_{world}(j_0) &= T_{local}(j_0) \\ T_{world}(j_1) &= T_{world}(j_0) \times T_{local}(j_1) \\ T_{world}(j_2) &= T_{world}(j_1) \times T_{local}(j_2) \\ T_{world}(j_3) &= T_{world}(j_2) \times T_{local}(j_3) \\ T_{world}(j_4) &= T_{world}(j_3) \times T_{local}(j_4) \end{aligned} \quad (8)$$

5. ZAKLJUČAK

U radu je opisana struktura 3D modela, metodi kompresije i skeletalnog sistem animacije modela.

Struktura 3D modela u glTF formatu pokazala se kao prilično jednostavna za analizu i upotrebu. U tekstualnom formatu moguće je “golim okom” ili alatima za analizu JSON formata izvući korisne informacije o prirodi modela. Razvoj softverskog rešenja za upotrebu ovog formata olakšan je zbog jasno definisane

specifikacije za širok spektar upotreba. Pokrivene su čak i upotrebe svojstvenih ekstenzija.

Metodi kompresije opisani u radu značajno utiču na performanse 3D programa, pogotovo kada je u pitanju prenos 3D modela preko mreže. Razvoj asimetričnih numeričkih sistema kao metod kodiranja značajno je doprinelo mogućnostima kompresije ovih tipova podataka što se može primetiti primenom ove metode u razvoju Draco [5], Zstandard [6], LZFSE [7] i sličnih kompresora.

U radu je opisana skeletalna animacija kao glavni metod animiranja 3D modela. Animatori inverznom kinematikom mogu vrlo lako napraviti "organski" realne animacije koje se pridržavaju ograničenja modela uz smislenu transformaciju mrežne strukture. Skeletalna animacija oslikava ovakav način animiranja i pojednostavljuje logiku potrebnu za implementaciju ovakvog sistema. Dovoljno je uzeti u obzir hijerarhiju skeletalne strukture i redosled primene transformacija. Kretanjem kroz stablo skeletalnog sistema relativno jednostavno dobijaju se konačne transformacije temena mrežne strukture modela.

6. LITERATURA

[1] Khronos group, glTF™ 2.0 Specification, <https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html>.

[2] J. Rossignac, "Edgebreaker: connectivity compression for triangle meshes," in IEEE Transactions on Visualization and Computer Graphics [3] J. Rossignac, "Edgebreaker: connectivity compression for triangle meshes," in IEEE Transactions on Visualization and Computer Graphics

[3] M. Isenburg and P. Alliez, "Compressing polygon mesh geometry with parallelogram prediction," IEEE Visualization, 2002. VIS 2002., 2002

[4] Google, Draco, <https://github.com/google/draco>

[5] Facebook, Zstandard, <http://facebook.github.io/zstd/>

[6] Apple, LZFSE, <https://github.com/lzfse/lzfse>

Kratka biografija:



Jovan Ivošević je rođen 1998. godine u Novom Sadu. Završio je Tehničku školu "Pavle Savić" u Novom Sadu, 2017. godine. Fakultet Tehničkih Nauka u Novom Sadu upisao je iste godine. Ispunio je sve obaveze i položio je sve ispite predviđene studijskim programom.

IMPLEMENTACIJA PARALELNOG K-MEANS ALGORITMA ZA KLASIFIKACIJU CIFARA UPOTREBOM OPENMP I MPI BIBLIOTEKA**IMPLEMENTATION OF PARALLEL K-MEANS ALGORITHM FOR DIGIT CLASSIFICATION USING OPENMP AND MPI LIBRARIES**

Anja Tanović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu prikazan je razvoj paralelnog k-means algoritma za klasifikaciju cifara. Opisan je osnovni k-means algoritam klasterizacije, kao i njegovo prilagođenje na algoritam za klasifikaciju skupa slika. Analizirane su mogućnosti paralelizacije algoritma i implementirane su različite verzije paralelnog k-means algoritma korišćenjem OpenMP i MPI biblioteka.

Ključne reči: Paralelni algoritmi, K-means algoritam, OpenMP, MPI, Klasifikacija cifara

Abstract – This paper presents the development of the parallel k-means algorithm for digit classification. The basic k-means clustering algorithm is described, as well as its adaptation to the image set classification algorithm. The possibilities of parallelizing the algorithm were analyzed and different versions of the parallel k-means algorithm were implemented using OpenMP and MPI libraries.

Keywords: Parallel algorithms, K-means algorithm, OpenMP, MPI, Digit Classification

1. UVOD

Paralelni algoritam u računarstvu je algoritam podeljen na određen broj delova koji se izvršavaju na različitim procesorskim uređajima. Kombinovanjem rezultata pojedinačnih segmenata na kraju izvršavanja dobija se rezultat algoritma. Transformacija programa iz serijskog u paralelni oblik izvršavanja zahteva detaljnu analizu samog algoritma. Neki algoritmi se mogu jednostavnije podeliti na više nezavisnih zadataka, dok neke algoritme karakteriše izuzetna zavisnost između koraka izračunavanja, te je njihova paralelizacija kompleksnija.

K-means algoritam klasterizacije, poznat i pod nazivom klasterizacija metodom k-srednjih vrednosti, je algoritam čiji cilj je particionisanje skupa podataka u k klastera, gde svaki klaster predstavlja grupu podataka koji imaju slične osobine ili značenje. Ovaj problem je računski težak (NP-težak problem), tako da je njegovo izvršavanje vremenski zahtevno za veće grupe podataka, kao što su na primer skupovi podataka korišćeni u oblasti mašinskog učenja. Kako bi se omogućila efikasna klasterizacija velikih skupova podataka primenom k-means algoritma, potrebna je paralelizacija algoritma.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vuk Vranjković, vanr. prof.

U drugom poglavlju rada biće ukratko opisani osnovni koraci k-means algoritma i implementacija k-means algoritma koji se koristi za klasifikaciju cifara. U trećoj celini biće opisana paralelizacija k-means algoritma korišćenjem OpenMP i MPI biblioteka za paralelno programiranje. U četvrtom poglavlju dati su rezultati i ubrzanje programa nakon testiranja različitih metoda paralelizacije algoritma. Na samom kraju, u petoj glavi, biće dati kratak pregled rezultata celokupnog rada i diskusija o mogućim unapređenjima algoritma.

2. K-MEANS ALGORITAM ZA KLASIFIKACIJU CIFARA

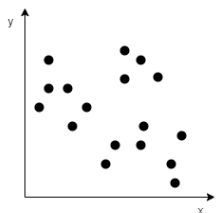
K-means algoritam je jedan od najstarijih i najšire korišćenih algoritama za klasterizaciju u oblasti istraživanja podataka (engl. *data mining*). Cilj algoritma je da se pronalaskom k klastera koji se ne preklapaju, podeli skup podataka na k grupa [1].

2.1. Osnovni koraci k-means algoritma

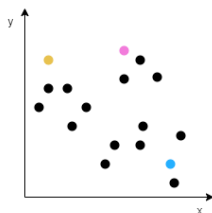
K-means algoritam može se ukratko opisati kroz sledeće korake. Prvo je potrebno izabrati k tačaka iz skupa podataka koje će predstavljati početne centroide, gde je k pozitivan ceo broj koji označava željeni broj klastera. Centroidi predstavljaju težišta klastera oko kojih se nalaze ostale tačke, i njihova pozicija se menja kroz iteracije algoritma. Nakon izbora početnih centroida, svaka tačka iz skupa podataka dodeljuje se najbližem centroidu. Grupa tačaka koja je dodeljena jednom centroidu predstavlja jedan klaster.

Težište svakog klastera se zatim ažurira izračunavanjem srednjeg rastojanja između svih pripadnika datog klastera. Nakon promene težišta može se desiti da su neke od tačaka više udaljene od centorida klastera kojem pripadaju nego od centroida nekog od susednih klastera. Kako bi se dobila najbolja podela skupa na klastere potrebno je opisati proces dodele tačaka najbližim centroidima klastera i izračunavanje pozicija novih centroida ponoviti sve dok se ne desi da se centri klastera više ne menjaju [1].

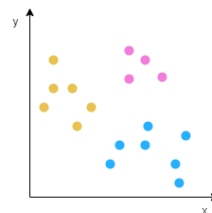
Ilustrativni primer izvršavanja k-means algoritma nad skupom tačaka, za vrednost $k = 3$, prikazan je na slikama 1-3. Opisani koraci k-means algoritma mogu se implementirati korišćenjem nekog od standardnih programskih jezika. Za opisivanje k-means algoritma u ovom radu korišćen je C++ jezik.



Slika 1. Početni skup tačaka



Slika 2. Izbor početnih centroida



Slika 3. Konačni rezultat klasterizacije

2.2. Klasifikacija cifara upotrebom k-means algoritma

K-means algoritam često se koristi za klasifikaciju slika u oblasti mašinskog učenja. Prepoznavanje cifara pisanih rukom je važan problem u oblasti optičkog prepoznavanja karaktera i često je korišćeno kao test za različite algoritme mašinskog učenja. MNIST baza podataka rukom pisanih cifara predstavlja standard za testiranje algoritama mašinskog učenja namenjenih za ovu svrhu. Sastoji se iz 60000 cifara namenjenih za trening i 10000 za testiranje. Svaka slika u bazi je veličine 28×28 piksela. Baza podataka data je formi CSV datoteke [2].

K-means je nenadgledani (engl. *unsupervised*) algoritam mašinskog učenja čiji je cilj da particioniše n tačaka podataka u k klastera. Svaki podatak koji predstavlja odgovarajuću cifru ima dimenzionalnost $28 \cdot 28 = 784$. Reprezentacija cifara u programu urađena je pomoću klase *Image*. Polja klase definišu koordinate tačke u 784-dimenzionalnom sistemu (niz *coord[dim]*, gde je *dim* dimenzionalnost), labelu koja opisuje cifru predstavljenu na slici (*label*), klaster kojem tačka pripada (*cluster*) i rastojanje između tačke i najbližeg klastera (*minDist*). U klasi je definisana i metoda za izračunavanje rastojanja između date tačke i tačke prosleđene kao parametar.

K-means algoritam za treniranje i testiranje sastoji se iz sledećih koraka:

1. Učitavanje skupa podataka iz CSV datoteke
2. Izbor početnih centroida iz skupa tačaka
3. Izračunavanje rastojanja između tačaka i centara klastera i dodela tačaka najbližim klasterima
4. Izračunavanje koordinata novih centara klastera
5. Provera da li je bar jedan od centroida promenio svoje koordinate. Ukoliko jeste, potrebno je ponoviti korake 3, 4 i 5.
6. Izračunavanje tačnosti treniranja
7. Testiranje nad test skupom podataka
8. Izračunavanje tačnosti testiranja
9. Upis rezultata u CSV datoteku

Rezultati treniranja i testiranja ispisuju se na terminalu, a koordinate centroida upisuju se u posebnu CSV datoteku kako bi se u budućnosti mogli koristiti za predviđanje nad novim podacima cifara. Dodatno, meri se i vreme potrebno za treniranje i testiranje, koje će se koristiti za poređenje između serijske i paralelnih verzija algoritma.

3. PARALELIZACIJA K-MEANS ALGORITMA UPOTREBOM OPENMP I MPI BIBLIOTEKA

Korisno bi bilo omogućiti brže treniranje prethodno opisanog algoritma iz više razloga. Prvo, algoritam daje bolje rezultate za veće vrednosti parametra k , ali on tada postaje i značajno sporiji. Osim toga, algoritam može

generisati različite rezultate pri istim parametrima zbog izbora početnih centroida. To znači da bi bilo neophodno izvršavati algoritam više puta, za velike vrednosti k , kako bi se dobili kvalitetni rezultati klasifikacije. Ovakvo treniranje bi moglo oduzeti mnogo vremena. Takođe, implementirani algoritam za klasifikaciju može se koristiti i za druge skupove slika a ne samo za cifre. Potencijalno neki od skupova mogu imati još veći broj slika, i same slike mogu biti značajno većih dimenzija, što bi dodatno usporilo algoritam. Jedno od mogućih rešenja za ubrzanje programa je paralelizacija algoritma.

3.1. Analiza mogućnosti paralelizacije k-means algoritma

Neophodan korak u paralelizaciji svakog algoritma je njegova detaljna analiza. Potpuno razumevanje algoritma olakšava podelu algoritma na odvojene zadatke i implementaciju efikasnog paralelnog algoritma. Korisno bi bilo usresrediti se na delove algoritma koji najduže traju i pokušati njih paralelizovati. U prethodnom poglavlju dati su koraci algoritma i može se zaključiti da se koraci 3, 4 i 5 ponavljaju potencijalno veliki broj puta, te bi njihova paralelizacija značajno pomogla pri ubrzanju programa. S obzirom na to da su ovi koraci međusobno zavisni, nije moguće istovremeno ih obavljati, već je potrebno paralelizaciju uraditi unutar svakog od koraka.

Posao koji je potrebno obaviti u koraku 3 je izračunavanje rastojanja između različitih centroida i tačaka. Ovo računanje se može podeliti na nezavisne taskove, s obzirom na to da se koordinate tačaka i centroida ne menjaju tokom celog koraka. Izvršavanje koraka 4 zahteva izračunavanja novih koordinata centroida koja se takođe mogu nezavisno obaviti. Provera da li je došlo do promene odgovarajućeg centra klastera nije zavisna od provere nad drugim centrom, tako da je i korak 5 moguće paralelizovati.

Analizom logike algoritma i programskog koda utvrđeno je da je moguće paralelizovati određene korake k-means algoritma i obezbediti kraće trajanje programa. Kako bi se algoritam efikasno implementirao, koristiće se biblioteke za paralelno programiranje, OpenMP i MPI.

3.2. Implementacija paralelnog k-means algoritma upotrebom OpenMP biblioteke

K-means algoritam moguće je paralelizovati korišćenjem modela deljene memorije pomoću OpenMP biblioteke. Paralelizacija se postiže dodavanjem *pragma* direktiva u kodu, vodeći računa o međuzavisnosti iteracija i balansiranju opterećenja. Prethodno analizirani koraci implementirani su u programskom kodu u obliku *for* petlji. Za paralelizaciju petlji korišćene su *pragma omp parallel for num_threads(t) schedule(OMP_SCHEDULE, chunksize)*

direktive, gde parametar t predstavlja broj niti koje se koriste za obavljanje zadataka, a *OMP_SCHEDULE* predstavlja strategiju raspodele posla između niti. Pokazuje se da je za petlje sa približno podjednakom količinom posla najbolje koristiti *static* raspodelu jer su iteracije podeljene nitima pre izvršavanja, dok je za petlje sa različitim trajanjem bolje koristiti *dynamic* raspodelu. Dodatno, parametar *chunksize* utiče na brzinu izvršavanja tako što se njegovim podešavanjem prave različite kombinovane raspodele [3].

Kako bi se u dobijeno ubrzanje programa moglo uporediti sa originalnim serijskim programom, neophodno je izmeriti vreme izvršavanja programa. Za merenje vremena korišćena je OpenMP funkcija *omp_get_wtime()*.

3.3. Implementacija paralelnog k-means algoritma upotrebom OpenMP i MPI biblioteka

Paralelizacija k-means algoritma pomoću OpenMP biblioteke omogućava veću brzinu izvršavanja programa na računaru sa više jezgara. Međutim, ako se za treniranje algoritma koriste računari koji su ograničeni po broju procesorskih jezgara, može se desiti da postignuto ubrzanje nije dovoljno dobro. U tom slučaju bi bilo korisno ako bi više računara moglo da radi zajedno na treniranju, doprinoseći većem ukupnom broju jezgara. Ovakav scenario omogućava MPI biblioteka koja koristi model razmene poruka između procesa na različitim računarima.

Empirijski je pokazano da najveći deo vremena u k-means algoritmu oduzima korak u kojem se izračunavaju rastojanja između tačaka i centara klastera. Ovaj posao je već podeljen na različita jezgra računara, ali bi bilo korisno kada bi još jezgara moglo da učestvuje u računanju. Paralelizacija je urađena tako što je svakoj niti dodeljen određen broj iteracija petlje u kojima se računaju rastojanja između klastera i odgovarajućih tačaka, što se može posmatrati kao da je svaka nit dobila određenu grupu tačaka nad kojima se računaju rastojanja. S obzirom na to da je broj tačaka u skupu podataka veoma veliki, ima smisla dodatno podeliti ovaj posao na više delova, tako što se kreira više procesa koji imaju svoju grupu tačaka. Uz to, svaki proces ima mogućnost da podeli svoju grupu tačaka na svoja procesorska jezgra, korišćenjem *pragma* direktiva i niti, kao i ranije.

Problem koji nastaje nakon izračunavanja svih rastojanja, je to što sledeći korak izračunavanja centara klastera zahteva znanje o svim tačkama i njihovim pripadnostima klasterima nakon proračuna. To znači da je neophodno da svi procesi pošalju ostalima svoje rezultate o tačkama. Prilikom procene vremena potrebnog za slanje podataka između računara, izračunato je da slanje slika u obliku klase *Image* oduzima previše vremena. Kako bi se ovo rešilo neophodno je rekonstruisati kod tako da se minimizuje količina podataka koja se šalje. Kada se bolje pogledaju polja klase *Image* može se zaključiti da nakon izračunavanja nije potrebno slati sva polja jer se samo polja *cluster* i *minDist* menjaju nakon izračunavanja rastojanja između tačaka i centroida. Stoga, klasa *Image* može se podeliti na 2 strukture koje zajedno predstavljaju jednu tačku, ali se posmatraju odvojeno. Nove strukture nazvane su *Image_part1* i *Image_part2*. Struktura *Image_part1* sadrži polja *coord*[dim] i *label*, a struktura

Image_part2 sadrži polja *cluster* i *minDist* koja se menjaju tokom izvršavanja algoritma. Slanjem samo strukture *Image_part2* u svakoj iteraciji algoritma, umesto prethodne strukture *Image*, značajno se skraćuje vreme potrebno za slanje poruka između procesa.

Dakle, nakon izračunavanja rastojanja, svaki proces prosleđuje svim ostalim procesima svoje rezultate o rastojanjima i pripadnosti klasterima, koji se nalaze unutar struktura *Image_part2*. U ovu svrhu koristi se funkcija *MPI_Allgatherv*, koja je proširena verzija često korišćene MPI funkcije *MPI_Allgather*.

Kako bi se omogućilo pokretanje paralelnog k-means programa na više računara, potrebno je grupisati računare u takozvane MPI klastere, unutar lokalne računarske mreže. Komunikacija između računara je omogućena preko mreže, putem SSH protokola (engl. *Secure Socket Shell*), dok su podaci koji se koriste deljeni pomoću NFS protokola (engl. *Network File System*) koji obezbeđuje deljenje direktorijuma među računarima.

4. REZULTATI TESTIRANJA PARALELNIH IMPLEMENTACIJA K-MEANS ALGORITMA

Cilj predstavljenih paralelnih implementacija je skraćivanje vremena izvršavanja algoritma. U nastavku su dati rezultati testiranja paralelnih implementacija korišćenjem OpenMP i MPI biblioteke, kao i poređenje brzine izvršavanja u odnosu na serijsku implementaciju. Za pokretanje programa korišćena su dva računara istih karakteristika, koja poseduju po šest procesorskih jezgara.

U tabeli 1 dati su rezultati merenja vremena izvršavanja serijskog programa sa različitim parametrom k . Ovi podaci korišćeni su kasnije pri proračunu ubrzanja paralelnih implementacija.

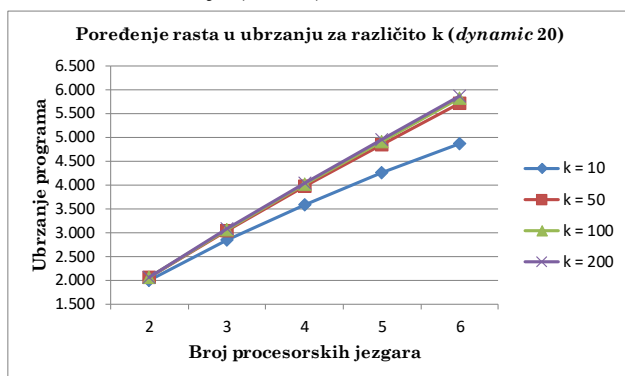
Tabela 1. Rezultati testiranja serijske implementacije algoritma

Serijska implementacija programa	Vreme izvršavanja programa [s]
$k = 10$ broj iteracija: 45 tačnost treniranja: 59.2217% tačnost testiranja: 59.32%	115.518
$k = 50$ broj iteracija: 194 tačnost treniranja: 82.51% tačnost testiranja: 83.14%	2343.61
$k = 100$ broj iteracija: 120 tačnost treniranja: 87.6933 % tačnost testiranja: 88.55%	2911.29
$k = 200$ broj iteracija: 79 tačnost treniranja: 90.6933 % tačnost testiranja: 91 %	3846.89

Iz datih rezultata može se uočiti da trajanje algoritma značajno raste sa rastom parametra k , što je i očekivano s obzirom na to da sa rastom broja centroida se povećava i broj prolazaka kroz petlje koda. Osim toga, tačnost treniranja i testiranja raste za veće vrednosti k , i dostiže 91% prilikom korišćenja 200 klastera.

Prva paralelna implementacija k-means algoritma koristi OpenMP biblioteku i deli algoritam na poslove koje obavljaju različite niti na dostupnim procesorskim

jezgrima računara. Strategija raspodele posla među nitima može biti različita u zavisnosti od konkretnog slučaja. Algoritam je testiran za različite strategije i parametre *chunksize* kako bi se pronašla najbrža implementacija. Rezultati ukazuju na to da je efikasnije koristiti *dynamic* strategiju nego *static*, ali da nema velike razlike u izboru *chunksize* parametra. Najbolji rezultati dobijeni su za strategije *dynamic*, 10 i *dynamic*, 20, a nadalje će se koristiti strategija *dynamic*, 20. Kada je u pitanju ubrzanje programa, primećuje se sličan rast u ubrzanju programa prilikom rasta vrednosti parametra *k*, ali se značajno bolji rezultati dobijaju za veće vrednosti *k*. Važan pokazatelj uspešne paralelizacije je činjenica da je dobijeno ubrzanje veoma blisko broju jezgara koja se koriste u izvršavanju (slika 4).



Slika 4. Poređenje rasta u ubrzanju programa

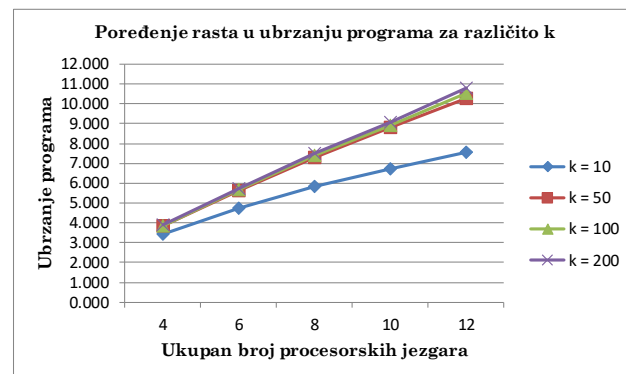
Testiranje je zatim obavljeno za implementaciju koja koristi OpenMP i MPI biblioteku, na dva računara. U tabeli 2 dato je dobijeno ubrzanje, a na slici 5 prikazan je rast u ubrzanju za različite vrednosti *k*. Treba naglasiti da se dati rezultati odnose na implementaciju programa u kojoj oba računara imaju pristup skupu podataka i nije potrebno prosleđivanje skupa ostalim procesorima.

Tabela 2. Rezultati testiranja paralelne implementacije algoritma upotrebom OpenMP i MPI biblioteka

Ubrzanje programa	broj procesorskih jezgara korišćeno od strane jednog procesa	Dva računara, 2 procesa	
		ukupan broj jezgara za izračunavanje	Ubrzanje programa (broadcast nije potreban)
k = 10	2	4	3.440
	3	6	4.762
	4	8	5.813
	5	10	6.741
	6	12	7.553
k = 50	2	4	3.841
	3	6	5.625
	4	8	7.300
	5	10	8.791
	6	12	10.280
k = 100	2	4	3.869
	3	6	5.683
	4	8	7.381
	5	10	8.933
	6	12	10.561
k = 200	2	4	3.903
	3	6	5.747
	4	8	7.529
	5	10	9.076
	6	12	10.775

Dati rezultati pokazuju da je ubrzanje korišćenjem dva računara značajno i da je rast ubrzanja sličan kao i u prethodnoj implementaciji. Razmatranjem celokupnog

testiranja, može se zaključiti da je paralelni k-means algoritam 10.775 puta brži od serijske implementacije algoritma, ako se koristi 12 procesorskih jezgara. Kraće vreme izvršavanja programa značajno olakšava svako dalje testiranje ili optimizovanje algoritma.



Slika 5. Poređenje rasta u ubrzanju programa korišćenjem dva računara

5. ZAKLJUČAK

U ovom radu prikazan je razvoj k-means algoritma za klasifikaciju cifara, i kreiranje njegovih različitih paralelnih implementacija. Za paralelizaciju korišćene su biblioteke OpenMP i MPI i omogućeno je pokretanje algoritma na jednom ili više računara. Paralelni k-means algoritam daje ispravne rezultate, a postignuto ubrzanje programa je blisko broju jezgara koja se koriste.

Paralelni k-means algoritam se može unaprediti kako bi se dobilo još veće ubrzanje programa. Iako paralelni algoritam daje dovoljno veliko ubrzanje, veliki deo koda je ostao serijski napisan. Moguće je dodatno analizirati i rekonstruisati serijske delove koda, i onda ih paralelizovati. Osim toga, upotrebom neke druge biblioteke za paralelno programiranje potencijalno se mogu ostvariti bolji rezultati. Kombinovanje odgovarajućih biblioteka i dobro definisanog koda pruža veće mogućnosti u paralelizaciji algoritma.

6. LITERATURA

- [1] J. Wu, "Cluster Analysis and K-means Clustering: An Introduction. U: Advances in K-means Clustering." Springer Theses. Springer, Berlin, Heidelberg, 2012. doi: 10.1007/978-3-642-29807-3_1.
- [2] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," in IEEE Signal Processing Magazine, vol. 29, no. 6, strane 141-142, Nov. 2012, doi: 10.1109/MSP.2012.2211477.
- [3] <https://610yilingliu.github.io/2020/07/15/ScheduleinOpenMP/> posjećeno septembar 2023.

Kratka biografija:



Anja Tanović rođena je u Novom Sadu 1999. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Energetika, elektronika i telekomunikacije odbranila je 2022.god.
kontakt: anjatanovic99@gmail.com

IMPLEMENTACIJA PROTOKOLA ZA KOHERENTNOST KEŠ MEMORIJE**IMPLEMENTATION OF CACHE COHERENCE PROTOCOL**

Jana Janković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu prikazana je implementacija MESI protokola za koherentnost keš memorije, kao i ostatka multiprocesorskog sistema sastavljenog od RISC-V procesora i dva nivoa keš memorije. Pomoću bihevijalne simulacije testirano je ponašanje sistema u situaciji kada se iz drugog nivoa keš memorije izbacuje blok koji su modifikovala dva jezgra.

Ključne reči: Multiprocesorski sistemi, Koherentnost keš memorije MESI, FPGA

Abstract – This paper presents the implementation of the MESI protocol for cache coherence, as well as the rest of the multiprocessor system composed of RISC-V processors and two levels of cache memory. Behavioral simulation was used to test the behavior of the system in a situation where modified block is evicted from the second level cache.

Keywords: Multiprocessor Systems, Cache coherence, MESI, FPGA

1. UVOD

Decenijama unazad mikroprocesorski sistemi doživljavali su eksponencijalan rast performansi zasnovan, kako na povećanju frekvencije signala takta, tako i na arhitekturnim poboljšanjima. Dok su brzine čipova rasle zahvaljujući Murovom zakonu projektanti su se bavili ugrađivanjem novih tehnika za poboljšanja performansi procesora, među kojima su protočna obrada, predviđanje grananja, analiza toka podataka i spekulativno izvršavanje [1]. Protočna obrada uvodi paralelizam na nivou instrukcija podjelom njihovog izvršavanja na faze. Superskalarni procesori nastavljaju sa iskorišćavanjem paralelizma na nivou instrukcija omogućavajući paralelno izvršavanje nezavisnih instrukcija u više protočnih obrada.

Nažalost, postoje praktična ograničenja u napretku navedenih trendova razvoja. Dalja povećanja frekvencije, koja uzrokuju dalja povećanja dinamičke snage, čine problem disipacije snage teže rješivim. Među instrukcijama takođe postoji ograničena količina paralelizma koja se može iskoristiti do trenutka gdje zavisnost resursa onemogućava upotrebu više protočnih obrada.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vuk Vranjković, vanr. prof.

Navedena ograničenja i nejenjavajuće potrebe za boljim performansama dovode da, početkom 2000-ih godina dođe do neizbježnog prelaska na smještanje više prostijih procesora na jedan čip, odnosno pojave multiprocesorskih sistema.

Brzina prenosa podataka između glavne memorije i procesora nameće dodatni problem kroz činjenicu da ne može da prati nagli porast brzine procesora. Keš memorije rješavaju taj problem jer omogućavaju procesoru brži pristup podacima, usljed čega se u multiprocesorskim sistemima, radi postizanja prihvatljivih performansi, svakom procesoru dodjeljuje jedan ili dva privatna nivoa keš memorije. Uvođenje privatnih keš memorija u multiprocesorske sisteme sa dijeljenom memorijom stvara problem koherentnosti keš memorije. Rješenje ovog problema predstavljaju protokoli za koherentnost keš memorije.

2. PROBLEM KOHERENTNOSTI KEŠ MEMORIJE

Memorija u osnovi predstavlja skup lokacija koje posjeduju određenu vrijednost. Kada se desi operacija čitanja ona bi trebalo da vrati poslednju vrijednost upisanu na tu lokaciju. Sekvencijalni programi se oslanjaju na ovu osobinu kada koriste memoriju da komuniciraju vrijednost sa mjesta gdje se ona računa, do mjesta gdje se ona koristi [2]. U multiprocesorskim sistemima, u kojima se dijeljeni adresni prostor koristi za komunikaciju između procesa koji se izvršavaju na više različitih procesora, takođe je potrebno da pomenuta osobina bude zadovoljena. Privatne keš memorije mogu da predstavljaju problem jer procesi tada vide zajedničku memoriju kroz različite keš memorije. Naime, istovremeno, u različitim keš memorijama, može postojati više kopija jedinstvenih podataka. Sadržaj istih bio bi nedosljedan kada bi procesorima bilo dozvoljeno da slobodno modifikuje isključivo svoje kopije podataka.

U osnovi dva zahtjeva moraju biti ispoštovana da bi se za multiprocesorski memorijski sistem reklo da je koherentan:

- Pročitana vrijednost sa memorijske lokacije mora odgovarati poslednjoj upisanoj vrijednosti na tu lokaciju. Ovaj zahtjev naziva se zahtjevom za propagacijom upisa (eng. write propagation).
- Izvršavanje više memorijskih operacija nad jednom memorijskom lokacijom mora biti viđeno u istom redoslijedu od strane svih procesora u sistemu. Ovaj zahtjev naziva se zahtjevom za serijalizacijom transakcija (eng. transaction serialization).

3. PROTOKOLI ZA KOHERENTNOST KEŠ MEMORIJE

Protokoli za koherentnost keš memorije obezbjeđuju komunikaciju između keš memorija, prilikom pristupa podacima, u cilju uspostavljanja koherentnosti.

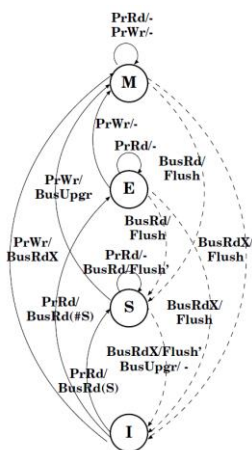
3.1. Podjela protokola za koherentnost keš memorije

Na osnovu načina na koji se izvodi propagacija upisa, dijele se na protokole sa invalidacijom upisa (eng. write invalidate) i protokole sa ažuriranjem upisivanja (eng. write update). Kod protokola sa invalidacijom upisa, kada jezgro želi da upiše novu vrijednost, inicira se transakcija koja će invalidirati kopije u drugim keš memorijama. Kod protokola sa ažuriranjem upisa, kada jezgro želi da upiše novu vrijednost, inicira se transakcija koja će ažurirati vrijednosti kopija u drugim keš memorijama.

Na osnovu toga da li se poruke šalju svim keš memorijama ili samo onim koje posjeduju podatak, protokoli se mogu svrstati u dvije grupe: protokoli sa direktorijumom (eng. directory protocols) i osluškjući protokoli (eng. snoopy protocols). Protokoli sa direktorijumom prate u kom kešu su prisutne kopije kojih blokova i te informacije prikupljaju u strukturi zvanoj direktorijum. Zahvaljujući direktorijumu poruke se mogu poslati samo onim keš memorijama koje posjeduju blok. Kod osluškjućih protokola poruka se šalje svim keš memorijama u sistemu. Osluškjući protokoli su pogodni za implementaciju u sistemima sa zajedničkom magistralom, obzirom da magistrala po prirodi stvari omogućava razgllašavanje poruka. U takvim sistemima svaka keš memorija posjeduje kontroler koji osluškuje poruke koje se postavljaju na magistralu.

3.2. MESI protokol

MESI protokol je jedan od najčešće korištenih protokola za keš memorije sa odloženim ažuriranjem (eng. write back).



Slika 1. Dijagram stanja MESI protokola [3]

Dijagram stanja MESI protokola prikazan je na slici 1. Punom linijom prikazani su prelazi koji se dešavaju kao odgovor na zahtjeve procesora, dok su isprekidanom linijom prikazani prelazi koji predstavljaju odgovor na zahtjeve sa magistrale. Protokol posjeduje četiri stanja u kojima se mogu naći blokovi. Blok je u modifikovanom

stanju (eng. modified, M) kada se njegov sadržaj razlikuje od onog u glavnoj memoriji i validan je u samo jednoj keš memoriji. Blok je u ekskluzivnom stanju (eng. exclusive, E) kada mu je sadržaj isti kao i u glavnoj memoriji i nalazi se samo u jednoj keš memoriji. Blok je u dijeljenom stanju (eng. shared, S) kada mu je sadržaj isti kao i u glavnoj memoriji i nalazi se u više keš memorija. Blok je u nevalidnom stanju (eng. invalid, I) kada mu je sadržaj nevažeći. Zahtjevi koji se mogu pojaviti na magistrali prilikom pristupa bloku prikazani su u tabeli 1.

Tabela 1. Zahtjevi na magistrali

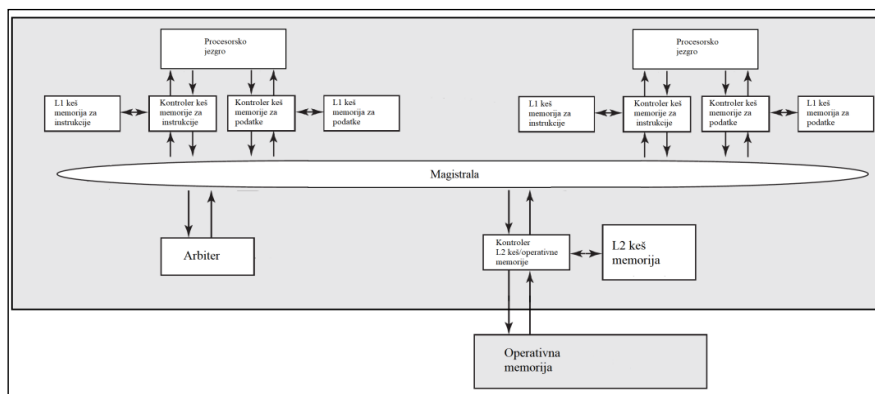
Naziv zahtjeva	Opis
PrRd	Zahtjev procesora za čitanjem bloka
PrWr	Zahtjev procesora za upisom u blok
BusRd	Zahtjev na magistrali za čitanjem bloka
BusRdx	Zahtjev na magistrali za ekskluzivnim čitanjem (upisom) bloka od strane procesora u čijem se kešu ne nalazi blok
BusUpgr	Zahtjev na magistrali za upisom u blok od strane procesora u čijem se kešu nalazi blok
Flush	Zahtjev na magistrali koji označava da se blok upisuje u glavnu memoriju
Flush'	Zahtjev na magistrali koji označava da je blok postavljen na magistrali. Uveden je za poboljšanje performansi i nije nužan za ispravan rad.

4. IMPLEMENTACIJA PROTOKOLA ZA KOHERENTNOST KEŠ MEMORIJE

4.1. Opis multiprocesorskog sistema

Organizacija multiprocesorskog sistema, za koji je implementiran protokol za koherentnost keš memorije, prikazan je na slici 2. Sistem se sastoji od više RISC-V procesorskih jezgara koji su povezani zajedničkom magistralom. Svako od jezgara posjeduje privatni prvi nivo keš memorije dok je drugi nivo dijeljen među njima. Prvi nivo je podijeljen na keš za instrukcije i keš za podatke. Svako od keš memorije je dodijeljen kontroler zadužen za uspostavljenje koherentnosti u sistemu. Kako je više uređaja povezano na magistralu, u sistemu se nalazi i arbiter koji reguliše pravo pristupa magistrali. Parametri sistema koji se mogu mijenjati su: broj procesorskih jezgara, veličina keš bloka i kapacitet keš memorija prvog i drugog nivoa.

Keš memorije prvog nivoa su 4-smjerne set-asocijativne memorije dok je keš memorija drugog nivoa 8-smjerna keš memorija. U svim keš memorijama sistema se koristi pseudo LRU algoritam zamjene zasnovan na binarnom stablu. Ovaj algoritam izabran je zbog dobrog kompromisa između performansi i broja bita potrebnih za njegovu implementaciju. Da bi se smanjila količina saobraćaja na magistrali za sve nivoe keš memorije korištena je metoda sa odloženim ažuriranjem nižeg nivoa hijerarhije uz koju ide metoda alociraj pri promašaju upisa. Keš drugog nivoa je inkluzivan sa keševima prvog nivoa zbog jednostavnije implementacije kontrolera. Protokol za koherentnost koji se koristi je MESI.



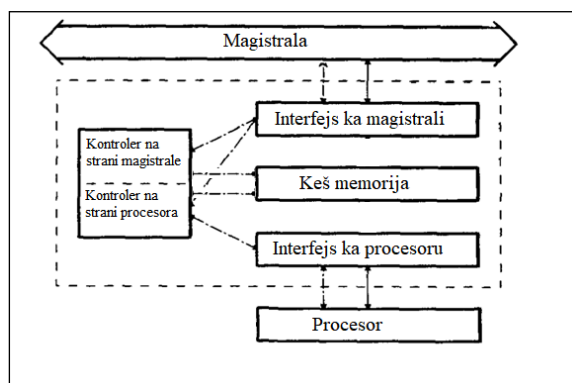
Slika 2. Organizacija multiprocesorskog sistema [4]

U keš memoriju za instrukcije se ne može upisivati, tako da za nju nije potrebno implementirati nijednu od metoda za ažuriranje nižih nivoa hijerarhije. Takođe, zbog toga, njen kontroler ne mora koristiti MESI protokol. Dovoljno je voditi evidenciju o tome da li je blok u memoriji validan ili ne.

U ovom projektu implementirano jezgro podržava RV32I skup instrukcija. Dodatno je implementirana i fence.I instrukcija, kako bi se postigla eksplicitna sinhronizacija memorije za instrukcije i podatke. Unutar procesora realizovana je protočna obrada sa pet faza.

4.2. Implementacija kontrolera koherentnosti keš memorije

Kontroler keš memorije za podatke mora istovremeno da odgovara na zahtjeve procesora i osluškuje dešavanja na magistrali. Zbog toga ga je najlakše posmatrati kao dva kontrolera jedan na strani procesora i jedan na strani magistrale [2,5]. Na slici 3 ilustrovan je takav kontroler. Kontroler na procesorskoj strani se ponaša u skladu sa zahtjevima procesora i zavisi od trenutnog stanja zahtjevang bloka u keš memoriji. Da bi se ispoštovao MESI protokol on mora posjedovati i pristup magistrali. Kontroler na strani magistrale osluškuje zahtjeve postavljene na magistrali i u skladu sa MESI protokolom i stanjem bloka u keš memoriji reaguje na njih. Keš memorija posjeduje dva porta kako bi joj istovremeno mogla pristupiti oba kontrolera.



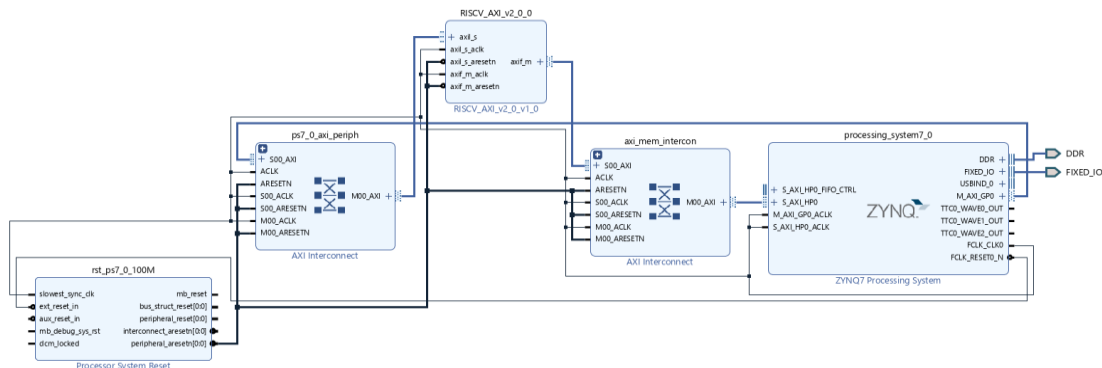
Slika 3. Unutrašnja organizacija kontrolera [5]

Kada kontroler na strani procesora utvrdi u kom je stanju zahtjevani blok u keš memoriji on će od arbitera tražiti dozvolu da postavi odgovarajući zahtjev na magistralu.

Nakon što dobije dozvolu i postavi zahtjev na magistralu kontroler na strani procesora čeka da kontroleri drugih keš memorija u sistemu odgovore na zahtjev. Kada dobije odgovor, kontroler može promijeniti stanje bloka i odgovoriti na zahtjev procesora. Može se desiti da, dok kontroler na strani procesora čeka na dozvolu pristupa magistrali, kontroler na strani magistrale promijeni stanje bloka kom procesor želi da pristupi. Kontroler na procesorskoj strani mora biti u mogućnosti da detektuje tu promjenu stanja i da u skladu sa njom promijeni zahtjev koji se postavlja na magistralu.

Problem može nastati i kada i jedna i druga strana kontrolera istovremeno žele da promijene stanje istog bloka. Te situacije se rijetko dešavaju zbog potrebe postavljanja zahtjeva na magistralu prilikom prelaska iz jednog stanja u drugo. Primjer za to, u MESI protokolu, bi bila situacija u kojoj procesorska strana želi da modifikuje blok u E stanju i promijeni mu stanje u M, dok zahtjev sa magistrale želi da promijeni stanje tog bloka u S ili I. Kontroleru na strani magistrale se tada daje prednost jer je on prethodno taj blok postavio na magistralu i procesorskoj strani se ne može dozvoliti da ga modifikuje.

Kada se odgovara na zahtjev sa magistrale, često je potrebno prvo postaviti blok podataka na magistralu i zatim mu promijeniti stanje. Može se desiti da u trenutku kada se javi zahtjev za koji je potrebno prenijeti blok, procesor želi da modifikuje isti, bez potrebe za pristupanjem magistrali. Procesoru se može omogućiti da tada modifikuje blok, međutim, svaki naredni upis za vrijeme trajanja prenosa, zbog potrebe da sistem bude koherentan, mu ne može biti omogućen. To se postiglo uvođenjem signala *ProcWants* i *SnoopHas*. Kontroler na strani magistrale provjerava da li je signal *ProcWants* postavljen na 1 i drži vrijednost signala *SnoopHas* na 0. Ukoliko jeste, čeka da vrijednost signala postane 0, nakon čega u sljedećem taktu prelazi u stanje u kom postavlja vrijednost signala *SnoopHas* na 1 i započinje slanje bloka. Kontroler na strani procesora prvo provjerava vrijednost signala *SnoopHas*. Ako je ona 0, vrijednost signala *ProcWants* postaje 1, u suprotnom je 0. Na taj način procesor dobija prednost, kada prije početka prenosa želi da upiše novu vrijednost u blok. Čim se prekine upis (*ProcWants* postane 0), kontroler na strani magistrale od sljedećeg takta započinje prenos i ne dozvoljava dalji upis u blok.



Slika 4. Blok dizajn sistema

4.3 Simulacija sistema

Pomoću bihevijalne simulacije testirano je ponašanje sistema u situaciji kada se modifikovani blok izbacuje iz drugog nivoa keš memorije. Sistem koji se testira sadrži dva jezgra, veličina keš bloka je 64B, kapacitet keš memorije prvog nivoa je 1KB i kapacitet keš memorije drugog nivoa je 4KB. Programi koji su namjenjeni da se izvršavaju na procesorima smješteni su na različitim memorijskim lokacijama. Procesori moraju posjedovati informaciju o početnoj adresi programa koji izvršavaju. U sistem se, zbog toga, uvode registri u koje se mogu, preko AXI-Lite interfejsa, upisati početne adrese programa i registar za signal dozvole takta.

```

lui a0, 6          lui a0, 6
addi a1, a0, 8     lw a1, 0(a0)
sw a1, 0(a0)       addi a1, a1, 4
                    sw a1, 4(a0)
lui a0, 5          lui a0, 7
lw a0, 0(a0)       lw a0, 0(a0)
lui a0, 4          lui a0, 8
lw a0, 0(a0)       lw a0, 0(a0)
lui a0, 3          lui a0, 9
lw a0, 0(a0)       lw a0, 0(a0)
lui a0, 1          lui a0, 10
lw a0, 0(a0)       lw a0, 0(a0)

```

Slika 5. Programi za testiranje izbacivanja modifikovanog bloka iz keš memorije drugog nivoa

Za testiranje ponašanja sistema, u situaciji kada se modifikovan blok izbacuje iz keš memorije drugog nivoa, koriste se programi prikazani u kodnom listingu, slika 5. U ovim jednostavnim asemblerskim kodovima jedan procesor prvo pristupa lokaciji 0x6000 i na nju upisuje vrijednost 0x6008, dok drugi procesor prvo čita sadržaj lokacije 0x6000 uvećava je za 4 i upisuje na lokaciju 0x6004. Procesori zatim čitaju sadržaj memorijskih lokacija različitih blokova, sve dok ne dođe do potrebe da se jedan blok izbaci iz keš memorije drugog nivoa kako bi se novi upisao. Blok koji se mora izbaciti je prvi blok koji su modifikovala oba procesora. Prilikom izbacivanja bloka, na osnovu vrijednosti koje se upisuju nazad u operativnu memoriju može se zaključiti da je u sistemu postignuta koherentnost

4.4 Implementacija sistema na Zybo razvojnoj ploči

Sistem testiran u simulaciji upakovan je u IP jezgro. Zapakovano jezgro posjeduje AXI-Lite Slave i AXI-Full Master interfejsa. IP jezgro je povezano sa Zynq procesorskim sistemom kao na slici 4. Nad sistemom sa slike 4 izvršene su sinteza i implementacija. Maksimalna frekvencija rada sistema je 71.42MHz. Kritičnu putanju

predstavlja putanja računanja adrese za pristup memoriji u trećoj fazi protočne obrade. Putanja prolazi kroz multiplexere koji određuju koji signali se propuštaju na ulaze ALU jedinice, zatim kroz ALU jedinicu i stiže na adresni ulaz memorije za podatke, tagove i LRU stanja. Najveće kašnjenje je kroz kanale za rutiranje je 10.855ns; kašnjenje kroz logičke elemente je 2.420ns. Sistem se na kraju testira u Vitis alatu, pomoću C aplikacije.

5. ZAKLJUČAK

U radu su prikazani bazni dijelovi implementacija MESI protokola za koherentnost keš memorije, i ostatka multiprocesorskog sistema, sastavljenog od RISC-V procesora i dva nivoa keš memorije. Rezultati testiranja ponašanja sistema pomoću bihevijalne simulacije se poklapaju sa rezultatima nakon testiranja sistema u Vitis alatu, pomoću C aplikacije.

Potencijalno poboljšanje performansi navedenog sistema moglo bi ići u smjeru korištenja magistrale dijeljenih transakcija i/ili primjenom principa ekskluzije.

6. LITERATURA

- [1] W. Stallings, "Computer Organization and Architecture: Designing for Performance", 9th ed. Boston, MA: Pearson, 2013.
- [2] D. Culler, J. P. Singh, and A. Gupta, "Parallel Computer Architecture: A Hardware/Software Approach", San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.
- [3] T. Suh, "Integration and evaluation of cache coherence protocols for multiprocessor socs", 2006.
- [4] V. Nagarajan, D. J. Sorin, M. D. Hill, and D. A. Wood, "A Primer on Memory Consistency and Cache Coherence", 2nd ed. Morgan & Claypool Publishers, 2020
- [5] R. H. Katz et al., "Implementing a cache consistency protocol", ACM SIGARCH Computer Architecture News, vol. 13, no. 3, pp. 276-283, 1985.

Kratka biografija:



Jana Janković rođena je u Bijeljini 1999. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva - Mikroročunarska elektronika, Embedded sistemi i algoritmi, odbranila je 2022. god kontakt: janajankovic@gmail.com



RAZVOJ PYTHON VEZA, BIBLIOTEKE I APLIKACIJE ZA HARDVERSKI
AKCELERATOR NEURONSKIH MREŽA

DEVELOPMENT OF PYTHON BINDINGS, LIBRARY AND APPLICATION FOR A
HARDWARE CNN ACCELERATOR

Nebojša Pilipović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu je izložen proces razvijanja softverskih slojeva i aplikacije za korištenje hardverskog akceleratora neuronskih mreža baziranog na modifikovanoj CoNNA arhitekturi. Modifikovana je postojeća C++ biblioteka, razvijen Python interfejs za istu, a zatim razvijena i sama aplikacija koja ima mogućnost izvršavanja više mreža odjednom, sa proizvoljnom raspodelom po jezgrima.

Ključne reči: *embedded softver, CNN akcelerator, embedded biblioteka, multiprocesorski sistemi*

Abstract – This paper presents a development process of the required software stack, and the application, for the control of a hardware neural network accelerator, based on a modified CoNNA architecture. The existing C++ library was modified, a Python interface to it was developed, and lastly the user application itself was developed, with the ability to run multiple networks concurrently, on an arbitrary number of available cores.

Keywords: *embedded software, CNN accelerator, embedded library, multicore systems*

1. UVOD

Mašinsko učenje je trenutno jedno od najpopularnijih polja u IT akademiji i industriji. Koriste se za obradu i inteligentno prepoznavanje šablona u slikama, audio zapisima, analognim signalima i ostalim tipovima ulaza. Jedan od modela mašinskog učenja su konvolucione neuronske mreže (CNN). Ove mreže sadrže bar jedan sloj, gde se vrši konvolucija između kernela i njegovog receptivnog polja na ulazu. Ova operacija je računski zahtevna kada se radi o količini kao što je broj piksela na slici, odbiraka u zvučnom zapisu i slično.

Implementacije konvolucionih neuronskih mreža su široko dostupne u softverskom svetu, i mogu se koristiti za razne primene, uključujući i izvršavanje mreža za klasifikaciju iz ovog rada. Ali, zbog mogućnosti paralelnih konvolucija istovremeno, softverska implementacija će uvek biti daleko sporija od akceleratora kao što je CoNNA akcelerator.

Za primenu obrade slike u realnom vremenu (sigurnosne kamere, autonomna vozila i slično), uvek se preferira

hardverska implementacija, jer softverska ne bi bila dovoljno brza, ili uopšte upotrebljiva za dati zadatak.

U ovom radu je realizovana Python aplikacija koja koristi dati hardverski akcelerator. Pre same aplikacije, razvijeni su svi potrebni međuslojevi od dostupne C++ biblioteke: C interfejs, Python interfejs, serijalizacija podataka za komunikaciju sa bibliotekom, Python klasa i ostale pomoćne konstrukcije koje su potrebne za ispravnu komunikaciju i rad korisničke aplikacije.

Krajnja aplikacija omogućava učitavanje više tipova mreža, i njihovu alokaciju na više jezgara akceleratora. Moguć je simultan rad više mreža odjednom.

Odrađene su vremenske analize za više vrsta implementacija aplikacije, zarad dobijanja maksimalne brzine obrade slike.

2. POSTOJEĆI SLOJEVI

Ovaj rad se bazira na prethodno razvijenom hardverskom akceleratoru za konvolucione neuronske mreže. Radi se o modifikovanoj verziji CoNNA akceleratora predstavljenom u radu [1].

Dostupan alat za rad sa njim koji je bio na raspolaganju jeste Linux drajver i upravljačka biblioteka napisana u jeziku C++. Celi sistem je namenjen da se izvršava na MPSoC platformi Xilinx Zynq ZCU102.

2.1. Komunikacija sa akceleratorom

Akcelerator je razvijen kao višejezgarni hardverski modul. Ovaj hardverski IP modul je sposoban da izvršava sve standardne slojeve koji se koriste u konvolucionim mrežama:

- Standardni konvolucionni sloj
- Depthwise separable slojevi
- Pooling sloj
- Fully-connected sloj i ostali

Koristeći kombinacije tih slojeva, ovaj akcelerator podržava popularne mreže kao što su ResNet, MobileNet, SqueezeNet, SSD, Inception i bilo koje druge proizvoljne mreže koje koriste podržane slojeve. U ovom radu, testirane su mreže MobileNet V1 sa SSD postprocessingom.

Sa akceleratorom se komunicira AXI-Lite transakcijama, za njegovu konfiguraciju i pristup registrima. Za prenos samih podataka (slika), u pozadini se koristi AXI-Full interfejs. Najbitniji registri za kontrolu prikazani su u tabeli 1.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Predrag Teodorović, red. prof.

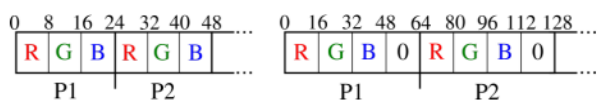
Tabela 1. Osnovni kontrolni registri

Adresa	Registar	Sadržaj	
0x0000	Kontrolni registar	Bit	Namena
		[0]	Reset
		[1]	Start, pokreće celu mrežu
0x0004	Adresa opisa mreže	[31:2]	Ne koristi se
		[31:0]	- Adresa u memoriji gde se nalaze podaci o mreži

Pored njih, postoje i statusni registri koji govore o stanju jezgra, postojanju greške, sadrže informacije o slojevima mreže i slično. Ukratko, za uspešno upravljanje jezgrom, potrebno je učitati podatke o mreži u memoriju, učitati sliku za obradu u memoriju, zatim u kontrolne registre upisati adresu podataka o mreži, i pokrenuti jedno ili više jezgara.

2.3. Format ulaznih podataka i predprocesiranje

Akcelerator, pored specifičnog formata podataka o slojevima mreže, zahteva i ulazne podatke u određenom formatu. U ovom radu će biti korištena MobileNet V1 mreža i SSD mreža, i tip ulaza je fotografija. Fotografija će inicijalno biti u formatu LxWxD, gde je dubina D = 3. Akcelerator zahteva izmenjen format predstave piksela u slici zavisno od mreže koju izvršava. MobileNet V1 mreža zahteva skaliranje vrednosti piksela na 16-bitnu vrednost. SSD mreža zahteva ulaze skalirane na opseg od 0 do 1, a zatim normalizovane u odnosu na skup podataka kojim je mreža trenirana.



Slika 1. Očekivani format ulaza

3. PYTHON VEZE ZA BIBLIOTEKU

Kao što je rečeno, unapred je bila dostupna biblioteka za rad sa akceleratorom. Ona definiše dve klase: *AiScaleController* i *SsdController*. Bilo je potrebno preneti funkcionalnost biblioteke u Python. Razvijene su Python veze za datu biblioteku (eng. bindings). Za ovo je korištena standardna Python biblioteka *ctypes*. Ona pruža mogućnost interakcije Python interpreter sa deljenim bibliotekama nastalim iz C koda [2].

3.1. C interfejs

Ctypes omogućava interakciju samo sa bibliotekama koje poštuju isti binarni interfejs (eng. Application Binary Interface - ABI) kao i C jezik. S druge strane, interakcija sa C++ bibliotekama nije moguća. Ovo je iz razloga što C++ nema jedinstven standardizovan ABI i načini povezivanja biblioteka mogu se razlikovati od kompajlera do kompajlera. Ovo znači da postojeća biblioteka za akcelerator *libaiscale*, koja je pisana u C++, neće moći odmah da se koristi kao takva. Rešenje koje je odabrano za ovo je pravljenje C interfejsa za C++ funkcije i metode. Na ovaj način biće korišteni isključivo C primitivni tipovi podataka na interfejsu, odabrane funkcije će biti povezane u C stilu, i *ctypes* neće imati problema da poziva te funkcije.

Interfejs je napravljen za bitne metode originalnih klasa, odnosno one važne za ispravnu konfiguraciju akceleratora. Ovo je podrazumevalo definiciju slobodnih funkcija u *extern "C"* bloku, koje za zadatak imaju da

konvertuju C tip u C++ tip, i pozovu original metodu. Zaglavlja funkcija u ovom bloku ne smeju imati tipove koji nisu podržani u C jeziku, kao što je *vector* ili klasa. Na primeru koda na slici 2 prikazana je jedna slobodna funkcija koja predstavlja C interfejs za original metodu.

```
extern "C" {
    void * AiScaleController_params(AiScaleCnnData* data, int
    ~ noOfCnns, int global_debug_level = 0){
        std::vector<AiScaleController::AiScaleCnnData> in_vec;
        for (int i = 0; i < noOfCnns; i++){
            AiScaleController::AiScaleCnnData data_c(data[i].weightfile,
            ~ data[i].configfile);
            in_vec.push_back(data_c);
        }
        return new AiScaleController(in_vec, global_debug_level);
    }
}
```

Slika 2. Primer C interfejsa za konstruktor

Svaka slobodna interfejs-funkcija u *extern "C"* bloku ima argumente i povratne vrednosti koje ogledaju one originalnih metoda. Naravno, tipovi su konvertovani u C kompatibilne tipove, npr. vektor u niz, klasa u strukturu i slično. Ovakav konstruktor u memoriji inicijalizuje jedan objekat klase, i vraća njegovu adresu. Ona mora biti generičkog, *void*, tipa jer tip klase nije podržan ovde. Sve ostale metode imaju po jedan dodatni argument, koji služi da im se prosledi ova adresa, kako bi nad tim objektom u memoriji pozvali njegovu metodu. Na ovaj način sve slobodne funkcije u C interfejsu se ponašaju kao metode kojima je vlasnik jedinstveni objekat u memoriji stvoren konstruktorom sa slike 2.

3.2. Python omotač klase

Nakon kompajliranja biblioteke sa interfejs funkcijama linkovanim u C stilu, moguće je koristiti deljenu biblioteku iz Python jezika. Napravljena je Python klasa *AiScaleController* (i *SsdController*) koja oponaša klasu iz originalne biblioteke. Ona ne redefiniše identične metode, nego poziva već postojeće iz deljene biblioteke koristeći funkcionalnosti *ctypes* biblioteke.

```
from ctypes import cdll
cdll.LoadLibrary("libaiscale.so")
#Bez argumenta
cdll.ssd_sanity_check()
#Sa argumentima
cdll.getConfigAddress.argtypes = [ctypes.c_uint32]
cdll.getConfigAddress.restype = [ctypes.c_uint32]
res = cdll.getConfigAddress(net)
```

Slika 3. Učitavanje biblioteke i poziv funkcije

Unutar konstruktora klase učitava se kompajlirana biblioteka u memoriju. Primer učitavanja biblioteke i pozivanja jednostavne funkcije iz nje prikazan je na slici 3. Ctypes upravljački objekat za učitavanje biblioteke čuva se kao polje *AiScaleController* klase, kroz celi život objekta. Svaka metoda ove Python klase pristupa učitanoj biblioteci preko tog upravljačkog objekta, kako bi pozvala i izvršila originalnu implementaciju. Pored učitavanja biblioteke, u konstruktoru Python omotač-klase poziva se i konstruktor originalne klase iz učitane biblioteke.

On će vratiti generičku adresu originalnog objekta *AiScaleController* klase. Ovo je drugo trajno važno polje Python klase, jer predstavlja objekta vlasnika svih metoda koje budu pozivane. Na način sa slike 3, sve metode ove

Python klase pozivaju originalne metode izvorne C++ klase, uz potrebne međukonverzije tipova podataka.

3.3. Aplikacioni interfejs visokog nivoa

Konačni sloj, pre samog razvijanja korisničke aplikacije je aplikacioni interfejs (eng. Application Programming Interface - API) na nešto višem nivou apstrakcije. Dok se za korišćenje *AiScaleController* i *SsdController* klasa zahtevala određena familijarnost sa interfejsom hardverskog sistema i načinom komunikacije, ovaj konačni nivo interfejsa definiše funkcije za upravljanje akceleratorom sa višeg nivoa apstrakcije. Ideja je da ovaj deo Python biblioteke nudi jednostavne pozive funkcija kao što su inicijalizacija akceleratora, jednostavan odabir tipa mreže, jednostavno procesiranje pojedinačnih okvira slike i slično - bez potrebe korisnika da ulazi u detalje o izlaznim baferima, rasporedu i formatu ulaza, rezultata, kontrolnih registara i slično.

Funkcije koje su u ovom sloju definisane predstavljaju bazu API-ja za razvijanje aplikacije i one su sledeće:

- `__init__(mem_off, core_off)` - Konstruktor koji stvara glavni objekat. Koristi memorijski i jezgarni ofset za pomeranje mreže u memoriji u slučaju da je početna regija već zauzeta drugom mrežom.
- `loadDesc(desc)` - Učitava podatke o mreži u memoriju, kao što je format IFM, format kernela i sl.
- `loadNets(nets)` - Učitava pojedinačne konfiguracije jezgara (težine kernela, tip sloja, i sl.) u svako jezgro koje se koristi.
- `processFrame(image)` - Procesira ulaznu numpy sliku: konfiguriše jezgra, kopira sliku u DRAM, pokreće sva potrebna jezgra transakcijama ka kontrolnim registrima, a zatim dobavlja rezultate, deserijalizuje ih i interpretira na dogovoren način.
- `softMax(array)` - nad 1000 članova radi probablističku gradaciju, radi lakše interpretacije rezultata.
- `oneHot(array)` - Radi one hot kodovanje. Odbacuje sve osim najverovatnije klase.
- `findLabel(array)` - Vraća indeks elementa niza čija je vrednost 1.

4. KORISNIČKA APLIKACIJA

Korisnička aplikacija koristi najviši sloj - Python biblioteku koja definiše klasu *AiScaleAPI*. Aplikacija je razvijena na više načina, radi ispitivanja performansi. Razvijen je i vrlo jednostavan grafički meni za izbor mreža i alokaciju po jezgrima.

4.1. Multithread verzija

Prva verzija aplikacije, razvijena je na Linux Ubuntu 20.04 operativnom sistemu, na četvor jezgarnom Cortex-A53 procesoru na ZCU102 platformi. Ona je pravljen da se izvršava na više niti, koristeći Python multithreading. Raspodela poslova po softverskim nitima je sledeća:

- Dobavljanje slike sa kamere je smešteno u zasebnu nit. Ovo ima smisla jer se ovde često čeka na resurs (kameru), što bi gušilo ostatak koda.

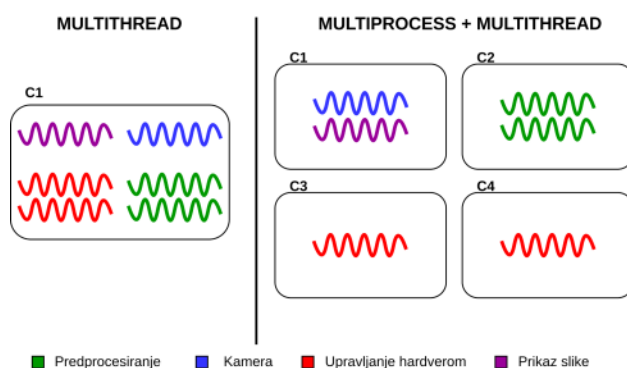
- Predprocesiranje je smešteno u zasebnu nit. Ovo je nešto CPU-intenzivniji zadatak i, ponovo, ima smisla odvojiti ga od niti koji čekaju na resurs. Ova stavka potencijalno podrazumeva dve niti, u slučaju kada se izvršavaju dve mreže istovremeno: njihov tip predprocesiranja nije isti, i potrebna je po jedna nit za svaku mrežu.
- Kontrola akceleratora je smeštena u zasebnu nit. Telo ove niti podrazumeva korišćenje prethodno stvorene klase *AiScaleAPI* za prosleđivanje predprocesirane slike hardveru, i čekanje na rezultat. Ova nit je resursno intenzivna jer većinu vremena čeka na obradu hardvera. I ovde takođe je potrebno stvoriti dve niti za ovu namenu, u slučaju izvršavanja dve simultane mreže: jedan *AiScaleAPI* objekat po mreži.
- Prikaz slike u grafičkom okruženju je smešten u zasebnu nit. Iz pogleda modularizacije funkcionalnosti, ovo je takođe smešteno u zasebnu nit.

Sve niti komuniciraju putem LIFO redova u vidu Python klase *Deque*. Ovo je optimalno jer njihove definicije *read* i *write* metoda dozvoljavaju nitima da "spavaju" i ustupe mesto drugima za to vreme.

4.2. Multiprocesna verzija

Standardan Python interpreter CPython ima ograničenje u domenu niti. Postojanje tzv. GIL-a (eng. Global interpreter lock) onemogućava raspodelu niti na više fizičkih jezgara, nego sve niti se izvršavaju na jednom jezgri, unutar jednog procesa [3]. Ovo se može zaobići kreiranjem više nezavisnih procesa, i delegiranjem zadataka svakom od njih, gde će se svaki proces izvršavati na zasebnom jezgri procesora. Ovo potencijalno može ubrzati rad aplikacije, ako u njoj ima dosta CPU-intenzivnih zadataka.

Procesor Cortex-A52 ima četiri fizička jezgra, pa su u ovoj verziji sva četiri iskorišćena za rad aplikacije. Ovo olakšava standardna Python biblioteka *multiprocessing*, koja pojednostavljuje stvaranje više procesa, i što je najvažnije, interprocesnu komunikaciju, koja inače nije jednostavna zbog odvojenog memorijskog prostora zasebnih procesa. Modifikovana je prva verzija da koristi ovaj način izvršavanja, i performanse su se u nekim delovima povećale. Na slici 4 prikazan je način raspodele poslova po nitima i jezgrima (C1-C4) za obe verzije aplikacije.



Slika 4. Raspodela poslova i niti po jezgrima

Kod multiprocesorske varijante, najveći napredak se dobio smeštanjem rutina za predprocesiranje ulaza u posebno jezgro, iz razloga što su to računski intenzivne rutine sa malo ili bez čekanja na resurse.

5. REZULTATI I ZAKLJUČAK

Korisnička aplikacija je razvijena, sa jednostavnim menijem gde se može specificirati koja mreža se izvršava na koliko hardverskih jezgara. Ovo je u potpunosti funkcionalno. Teoretski, izvršavanje mreže na više jezgara umesto jednog bi trebalo višestruko puta ubrzati rad. U krajnjoj verziji ovo nije u potpunosti tako, iz razloga što je uočeno da znatan deo vremena na softverskoj strani se potroši na predprocesiranje. Brzine rada takođe zavise od tipa mreže koji se izvršava: razlike u hardverskom delu su manje primetne, ali razlike u vrsti predprocesiranja mogu uvesti znatne razlike u brzinama. Naravno, veličine ulaznih slika takođe drastično igraju ulogu u ovome (224x224 za MobileNet - brže, 300x300 za SSD - nešto sporije). Izvršavanje obe mreže simultano je moguće, ali će imati različite FPS (eng. Frames per second).

Procenjivanje vremenskih karakteristika konačne implementacije urađeno je i primitivnim merenjem vremena koristeći standardnu Python biblioteku *time*, ali i stranim multiprocesnim profajlerima kao što je *viztracer*. Kombinacija tip mreže - broj jezgara ima puno, i u tabeli 2 će biti prikazane osnovne karakteristične konfiguracije. Osim hardverskih vremena, vremena u tabeli su okvirna i variraju zavisno od sporednih opterećenja operativnog sistema. Ovo može biti izraženo na skromnijim CPU jezgrima, gde je svaka nit dragocena.

Tabela 2. Hardverska i softverska vremena izvršavanja

MobN/SSD Jezgara -->	1/0	0/1	2/0	0/2	4/0	2/2
[HW] MobN / SSD Total	29.6 / -	- / 32.3	15.1 / -	- / 16.6	7.8 / -	15 / 16.5
[HW] Samo SSD	-	0.53	-	0.53	-	0.53
[SW] Postproces	5 / -	- / 0.2	5 / -	- / 0.2	5 / -	5 / 0.2
[SW+HW] MobN / SSD Total	46 / -	- / 46	26.6 / -	- / 25	27 / -	27 / 26
[SW] Predproces MobN / SSD	20 / -	- / 50	20 / -	- / 50	20 / -	35 / 70

* Vremena su u milisekundama

Dostignuta je propusna moć celog sistema od oko 18 FPS za MobileNet mrežu, i par FPS niže za SSD mrežu (veća slika, nešto komplikovanije predprocesiranje). Ovo je tako čak i pri multiprocesnoj realizaciji softverske strane. Subjektivno, slika izgleda dovoljno tečno, jasno se razaznaju oblici i mreža kao što je SSD izgleda responzivno, i precizno nalazi okvire objekata. Na slici 5

može se videti prikaz grafičkog izlaza aplikacije dok obe mreže rade simulatno.

Za narednu verziju potrebno je razmotriti načine dalje optimizacije trenutnih konfiguracija - jedna od ideja jeste razvoj jezgra za predprocesiranje u hardveru, kako bi se Cortex-A52 rasteretio tog dela posla. Dalje, u budućnosti je potrebno dodati i ostale kompatibilne mreže za obradu slike (mreže za segmentaciju slike i slično), i prilagoditi aplikaciju njihovom načinu izvršavanja.



Slika 5. Demonstracija simultanog rada obe mreže

6. LITERATURA

- [1] R. Struharik, B. Vukobratović, A. Erdeljan, i D. Rakanović, „Conna—compressed cnn hardware accelerator“, u 2018 21st Euromicro Conference on Digital System Design (DSD), IEEE, 2018, str. 365–372.
- [2] „ctypes — A foreign function library for Python“, Python documentation.
<https://docs.python.org/3/library/ctypes.html>
(pristupljeno 30. avgust 2023.).
- [3] F. Doglio, Mastering Python High Performance. Packt Publishing, 2015.

Kratka biografija:



Nebojša Pilipović rođen je u Doboju 1998. god. Bečelorski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Embedded sistemi i algoritmi odbranio je 2021.god. kontakt: nb.pilipovic@gmail.com

ИМПЛЕМЕНТАЦИЈА XMPP BOSH ПРОТОКОЛА У ОКВИРУ TR-069 КЛИЈЕНТА**INTEGRATION OF XMPP BOSH PROTOCOL IN THE TR-069 CLIENT**

Ана Теодоровић, *Факултет техничких наука, Нови Сад*

Oblast – РАЧУНАРСТВО И АУТОМАТИКА

Kratak sadržaj – У овом раду је представљено решење за унапређење комуникације између TR-069 клијента и XMPP послужитеља. У оквиру TR-069 клијента додат је XMPP BOSH клијентски модул који омогућава комуникацију коришћењем XMPP протокола преко HTTP порука.

Кључне речи: *Управљање уређајима у Интернету, TR-069, XMPP, BOSH, CPE, ACS*

Abstract – This paper presents a solution for improving communication between TR-069 client and XMPP server. The XMPP BOSH module has been added to TR-069 client, which enables communication via XMPP protocol via HTTP messages.

Keywords: *Device management in Internet, TR-069, XMPP, BOSH, CPE, ACS*

1. УВОД

Послужитељи (eng. Servers) уређаја потрошачке електронике су системи засновани на рачунару, чији је основни задатак комуникација са уређајима са једне стране и манипулација подацима са друге. У овом раду описан је проблем, концепт и реализација XMPP комуникационог протокола и једне од његових најзначајнијих технологија – BOSH. Обухваћене су теме које су повезане са темом рада, као и основни појмови везани за TR-069 протокол, који обезбеђује постојање TR-069 послужитеља и TR-069 клијента који се налази на крајњем корисничком уређају.

2. ТЕОРИЈСКЕ ОСНОВЕ**2.1. TR-069**

TR-069 (eng. Technical Report 069) је техничка спецификација која дефинише комуникациони механизам за даљинско управљање крајњим корисницима [1]. Описује CPE WAN Management Protocol (CWMP), који је намењен за комуникацију између крајњег корисника (CPE, eng. Customer Premises Equipment) и аутоматско-конфигурационог послужитеља (eng. ACS – Auto-Configuration Server). Протокол подржава разне функционалности, од који су најважније: аутоматско конфигурисање и динамичко реконфигурација услуга, управљање софтверским модулом, праћење статуса и перформанси CPE и њихова дијагностика.

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији ментор је био др Илија Башичевић, ред. проф.

Најзначајнија функционалност је аутоматско конфигурисање и динамичка реконфигурација услуга. Спецификација обезбеђује могућност примарне и накнадне конфигурације у случају нових захтева. Управљање софтверским модулом омогућава инсталацију, деинсталацију и ажурирање софтверских модула, као и праћење њиховог стања.

2.2. CWMP параметри

CWMP параметри су претстављени у форми модела података (eng. data model). Улога параметара је да обезбеде ACS-у информације о карактеристикама и статусу CPE, као и да управљају њиховом конфигурацијом. Параметри могу бити специфицирани само за читање или за читање и упис. Параметри имају хијерархијску структуру. Структура је претстављена као објекат. Сваки објекат садржи један или више параметара. Сваки CPE садржи један главни параметар који у зависности од типа уређаја добија вредност Device или Internet Gateway Device.

2.3. XMPP

XMPP (Extensible Messaging and Presence Protocol) је протокол за комуникацију који се користи за потребе размене тренутних порука, звучне и видео позиве, информације о доступности других корисника, одржавање листе контаката, базиран на XML (eng. Extensible Markup Language) стандарду. Предност овог протокола у односу на остале овог типа су то што је :

- Отворен
- Стандардизован
- Проверен
- Децентрализован
- Безбедан
- Прилагодљив
- Флексибилан

2.4. Адресирање

Сви корисници у мрежи имају јединствену XMPP адресу која се назива JID. Структура JID подсећа на адресу електронске поште. Садржи корисничко име (username), назив домена или IP адресу послужитеља који су раздвојени знаком „@”.

Такође, један корисник се може пријавити са више локација (кућа, посао...), тако што ће навести ресурс (resource), који ће бити одвојен косом цртом. На пример, пун назив JID налога би био: „username@domain/resource”.

2.5. BOSH

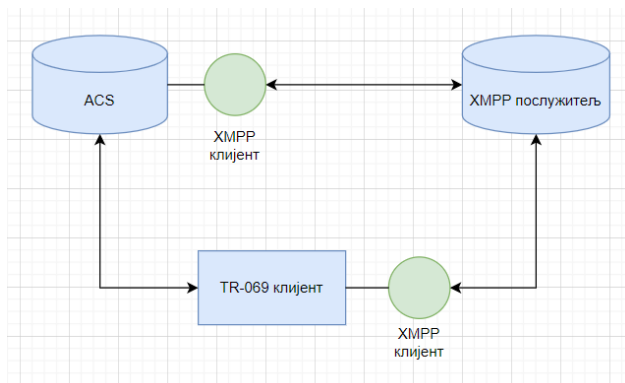
Најчешће коришћен начин преноса за XMPP је дугорочна TCP (Transmission Control Protocol) веза, која у неким случајевима није могућа због природе крајњих уређаја или мреже. Алтернатива овоме је BOSH (eng. Bidirectional-streams Over Synchronous HTTP), који размеђује захтеве и одговоре преко краткотрајних веза. BOSH је технологија за двосмерну комуникацију преко Hypertext Transfer Protocol (HTTP). Дизајниран је тако да преноси податке са минималним кашњењем у оба смера.

Техника коју користи BOSH назива се “HTTP дуго анкетирање” (eng. HTTP long polling) која смањује кашњење и потрошњу пропусног опсега у односу на друге технике HTTP анкетирања [1]. Када клијент пошаље захтев, управљач не одговори одмах, него држи захтев отвореним све док не буде имао податке које треба да пошаље или истекне време неактивности. Клијент тада шаље нови захтев и настала комуникација. Уколико клијент има нови захтев, док је претходни још увек отворен, он шаље нови захтев, управљач на претходни одговара углавном празном поруком, којом затвара претходну петљу и отвара се нова петља која служи за комуникацију, док стара чека нови захтев.

Ова технологија је значајна за комуникацију са web клијентима. Иако је углавном тип поруке који се преноси XML, BOSH технологија може да преноси мешавину елемената који су дефинисани различитим протоколима (XMPP i JSON).

2.6. TR-069 и XMPP

У одређеним случајевима, ACS је спречен да пошаље поруку CPE-у директно због постојања firewall-а или gateway-а. Тај проблем решава XMPP проткол. И ACS и CPE се региструју на XMPP послужитељу, а затим га ACS може користити да пренесе поруку до CPE [2].



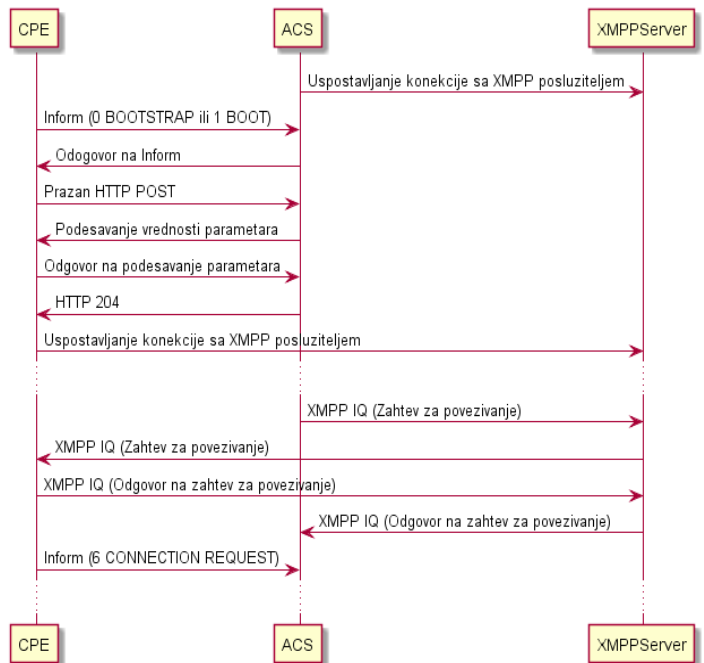
Слика 1. Комуникација TR-060 клијента са XMPP

3. ИМПЛЕМЕНТАЦИЈА

У претходном поглављу описана је намена и значај XMPP проткола. За имплементацију у оквиру TR-069 клијента неопходно је постојање XMPP послужитеља на коме ће се клијент регистровати. За регистрацију је потребно креирати JID, као што је раније описано. На основу њега корисник постаје препознатљив. Такође, CPE мора бити у могућности да одржава константну везу са XMPP послужитељем.

Захтев са повезивање са CPE се обавља на следећи начин:

- ACS успоставља везу са XMPP послужитељем
- ACS омогућава коришћење XMPP-а постављањем параметара
- Уређај успоставља везу са наведеним XMPP послужитељем
- Када ACS пожели да комуницира са уређајем може то урадити преко установљених XMPP порука



Слика 2. Дијаграм захтева за повезивање

ACS би требао да омогући коришћење XMPP-а на CPE-у конфигурисањем XMPP.Connection објекта, ManagementServer.ConnReqXMPPConn параметра и опционо ManagementServer.ConnReqAllowedJabberIDs који садржи листу дозвољених JID-ова.

Вредност ManagementServer.ConnReqXMPPConn параметра мора бити постављена на име путање до XMPP-Connection објекта која се користи за везу. Уколико је објекат избрисан, вредност параметра мора бити постављена на празан низ.

XMPP.Connection претставља XMPP везу између корисника и послужитеља [3]. Може постојати највише један унос за одређени Username, Domain и Resource. На основу ових параметара, формира се JID како је раније описано.

Такође, значајан је објекат XMPP.Connection.Server који описује послужитељ који се користи за везу. Неки од параметара у оквиру овог објекта су: адреса послужитеља (ServerAdress), Port, доступност (Enable)...

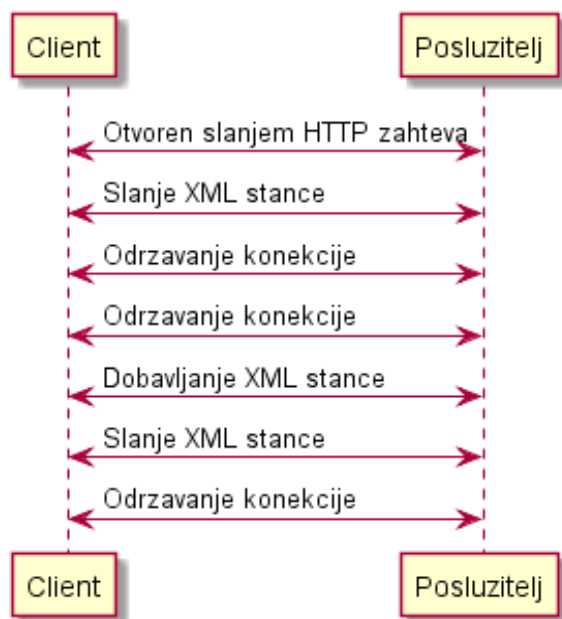
Објекат XMPP.Connection.Stats пружа статистичке информације о вези (број примљених порука, број неуспешно примљених порука, број пренесених порука, број неуспешно пренесених порука).

На слици 2 је приказано успостављање везе са XMPP послужитељем. ACS на основу постављених параметара шаље захтев за везу XMPP послужитељу. Након успешног повезивања, ACS преко XMPP послужитеља

шаље захтев за повезивање (eng. `ConnectionRequest`) у виду XMPP IQ станце. У оквиру овог захтева налазе се корисничко име и шифра који су наведени у параметрима

`Device.ManagementServer.ConnectionRequestUsername` и `Device.ManagementServer.ConnectionRequestPassword`, а који се односе на корисника. Уколико су наведени исправни подаци, корисник шаље позитivan одговор на захтев. Позитivan одговор од стране CPE је празна IQ станца. У случају неуспешног захтева, CPE шаље IQ станцу која садржи `service-unavailable` грешку.

`ManagmentServer.ConnReqAllowedJabberIDs` параметар се поставља опционо, али даје додатну заштиту у виду забране примања порука од корисника чији JID се не налази у листи дозвољених.

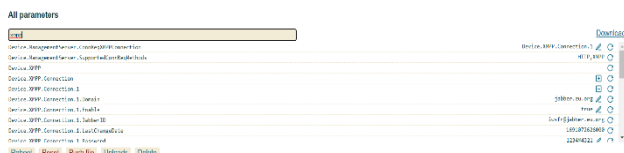


Слика 3. Дијаграм одржавања комуникације

Након успоставе везе, комуникација се одвија као на дијаграму са слике 3.

4. ТЕСТИРАЊЕ

GenieACS је open-source ACS. Окружење GenieACS је приказано на слици 4.



Слика 4. ACS са параметрима

Gajim је бесплатни, open-source клијент за размену тренутних порука у оквиру XMPP протокола. Тестирање преко Gajim-а је рађено тако што је направљен нови налог са сопственим JID-ом. Налог је додат у листу дозвољених параметара како би био омогућен пријем порука. Послате су поруке са различитим садржајем како би се потврдио пријем и исправан одговор на различит садржај.

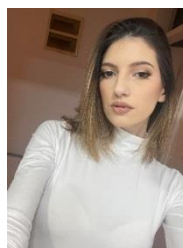
5. ЗАКЉУЧАК

Главни циљ овог рада био је реализација TR-069 клијента. Коришћење XMPP BOSH протокола омогућило је комуникацију између ACS-а и CPE-а без сметњи изазваних постојањем firewall-а или gateway-а. У оквиру CWMP параметара укључени су сви неопходни параметри за реализацију XMPP комуникације. За испитивање решења, као послужитељ, коришћен је Gajim клијент.

6. ЛИТЕРАТУРА

- [1] XEP-0124: Bidirectional-streams Over Synchronous HTTP (BOSH), XMPP Standards Foundation, 2021, <https://xmpp.org/extensions/xep-0124.html> (приступљено у јулу 2023.)
- [2] I. Basicovic, "An analysis of the TR069 (CWMP) protocol," 2023 46th MIPRO ICT and Electronics Convention (MIPRO), Opatija, Croatia, 2023, pp. 460-465
- [3] TR-069 Device:2.12 Root Object definition, Broadband Forum, 2018, <https://cwmp-data-models.broadband-forum.org/tr-181-2-12-0-cwmp.html> (приступљено у јулу 2023.)

Кратка биографија:



Ана Теодоровић рођена је у Шапцу 1997 год. Основне академске студије на Факултету техничких наука завршила је 2020. Исте године уписала је мастер студије, смер Рачунарство и аутоматика. контакт: teodorovic.a4@gmail.com

POWER-SIGNAL INTEGRITY SIMULACIJE ELEKTRONSKIH KOLA

POWER AND SIGNAL INTEGRITY SIMULATIONS OF THE ELECTRONIC CIRCUITS

Milomir Spajić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U radu je opisan teorijski uvod u Signal Integrity i Power Integrity problematiku. Glavni akcenat bio je na izvršavanju Power Integrity simulacija koje se koriste za verifikaciju projektovanih štampanih ploča prije njihove izrade, što omogućava izbjegavanje potencijalnih problema nakon što se pločice izrade.

Ključne reči: Power Distribution Network, Power Integrity, Signal Integrity, simulacije

Abstract – This paper presents the theoretical introduction to Power Integrity and Signal Integrity problems. Main accent was on execution of Power Integrity simulations which are used for verification of the Printed Circuit Boards before production, which enables avoidance of the potential problems after production of the Printed Circuit Boards.

Keywords: Power Distribution Network, Power Integrity, Signal Integrity, simulations

1. UVOD

Električna kola koja se danas projektuju u obliku štampanih pločica postaju sve kompleksnija. Neki od glavnih razloga kompleksnosti ovih kola su napajanja potrebna za sve čipove i komponente na ploči, kao i brzine signala koji prenose informacije, odnosno oštre ivice tih signala. Velika kompleksnost uređaja dovodi i do češćih grešaka koje inženjeri mogu napraviti pri njihovom razvoju.

Pojavom grešaka u dizajnu, dolazi do povećanja cijene razvoja uređaja. Da bi se cijena razvoja uređaja smanjila, potrebno je smanjiti i broj potencijalnih grešaka koje nastaju tokom razvoja. Jedan od načina da se broj grešaka smanji jeste korišćenje softverskih alata za simulacije. Postoje alati za simulacije električnih kola i alati za simulacije električnih pločica. Simulacije koje se rade za provjeru napajanja su Power Integrity simulacije, a simulacije koje se koriste za provjeru signala na ploči su Signal Integrity simulacije. Simulacije koje su odrađene u okviru ovog rada, izvršene su u softverskom alatu Cadence Sigrity. U oblasti Power Integrity neke od najvažnijih simulacija su pad napona na linijama napajanja, impedansa kao i simulacija dikapling kondenzatora. Za Signal Integrity verifikaciju, od interesa je simulacija impedanse, dijagram oka kao i S parametri.

NAPOMENA:

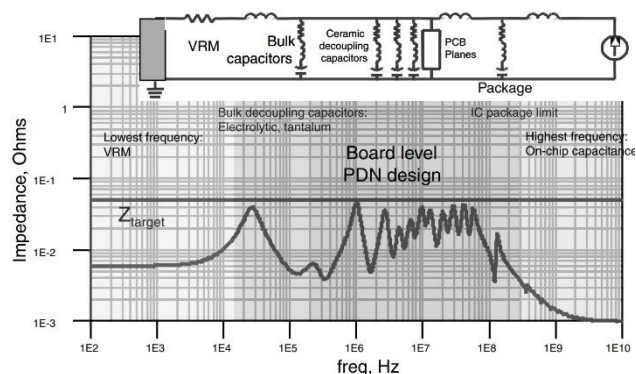
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Miodrag Brkić, vanr. prof.

2. POWER DISTRIBUTION NETWORK (PDN)

Kao uvod u Power Integrity analize, potrebno je upoznati se sa terminom Power Distribution Network (PDN). PDN odnosno mreža napajanja na štampanoj ploči predstavlja sve međusobno povezane elemente od naponskog regulatora (odnosno izvora napona – generatora) do potrošača (što je obično mikrokontroler odnosno čip). Ovo uključuje naponski regulator, dikapling kondenzatore, vije (eng. via), vodove, poligone na štampanoj ploči, ostale kondenzatore na ploči, kontakte za lemljenje između čipa i ploče kao i vodove unutar čipa. Prva stvar o kojoj treba voditi računa prilikom projektovanja PDN je da se obezbedi konstantan napon na priključcima potrošača. Ukoliko se projektuje loša mreža napajanja, dolazi do pojave šuma koji može izazvati nepravilan rad, pa čak i prestanak rada potrošača.

2.1. Ciljna impedansa

Uzrok pojave talasnosti napona na priključcima za napajanje potrošača jeste PDN impedansa. Tokom projektovanja mreže, potrebno je izračunati impedansu za koju će pad napona u najgorem slučaju i dalje biti dovoljno nizak da ne dovede do nepravilnog rada potrošača. Zbog različitih karakteristika komponentata koje su povezane na PDN, impedansa PDN zavisi od frekvencije. Na slici 1. prikazan je impedansni profil jedne PDN mreže koja zadovoljava uslov da u svakoj tački (na svakoj frekvenciji) bude ispod unaprijed određene ciljane impedane.



Slika 1. PDN impedansa u odnosu na ciljnu impedansu

2.2. Dikapling kondenzatori

Na slici 1. prikazan je slučaj kad je PDN impedansa podešena tako da bude niža od ciljane impedanse i na taj način je obezbeđeno da napon potrošača uvijek bude u granicama u kojima je moguć njegov pravilan rad. Ukoliko PDN impedansa nije niža od ciljane impedanse, potrebno je popraviti to na neki način. To je moguće ura-

ditu dodavanjem dikapling kondenzatora. Ovi kondenzatori se postavljaju što bliže potrošaču i smanjuju PDN impedansu. U teoriji impedansa kondenzatora linearno opada sa povećanjem frekvencije, međutim u praksi to nije tako. Nakon određene granične frekvencije, kondenzator počinje da ispoljava osobinu induktivnosti i ponaša se kao zavojnica, pa daljim povećavanjem njegove frekvencije, povećava se i impedansa. Ovo u nekim slučajevima dovodi do potrebe da se koristi više različitih dikapling kondenzatora.

3. POWER INTEGRITY

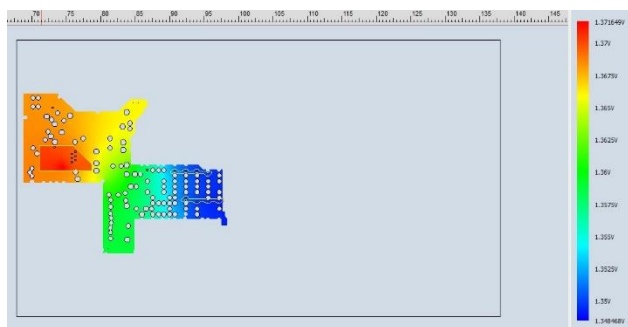
Problemi u *Power Integrity* oblasti su danas česta pojava, pogotovo kod ploča sa brzim signalima i ostrim ivicama signala. Ukoliko PDN impedansa kod ovakvih uređaja nije pravilno podešena, napon potrošača na ploči može biti nestabilan sa velikom količinom talasnosti što dovodi do nepravilnog rada.

Cilj *Power Integrity* analize jeste da se obezbedi pravilna isporuka energije potrošačima na štampanoj ploči. Da bi se mogućnost greške prilikom projektovanja PDN svela na minimum, vrše se simulacije projektovane mreže, tako da se greške mogu ispraviti prije izrade uređaja.

3.1. Pad napona u vodovima napajanja

Ukoliko je struja potrošača značajna, a PDN impedansa nije pravilno isprojektovana, može doći do velike razlike potencijala između izvora i potrošača. Da bi se ovo izbjeglo, moguće je izvršiti analizu pada napona u vodovima napajanja.

Za potrebe ove analize, korišćen je alat *PowerDC* koji je dio *Cadence Sigrity* paketa alata. Analiza je izvršena za jedan naponski nivo, na kojem je potrošaču potrebna najveća struja. U ovom slučaju je to napon od 1.375V a maksimalna struja koja je dobijena iz tehničke dokumentacije potrošača je 6.1A. Važno je napomenuti da je ovo simulacija za najgori slučaj. Potrebno je podesiti model generatora i potrošača a rezultat analize prikazan je na slici 2.



Slika 2. Prikaz rezultata simulacije u grafičkom obliku

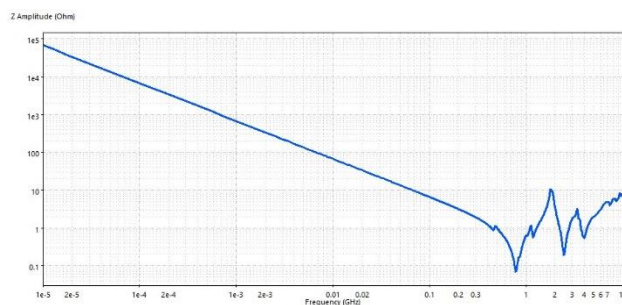
Na slici 2. sa desne strane se može vidjeti i skala napona, po kojoj je najniži napon 1.348V (tamno plava boja) a najviši 1.372V (crvena boja).

Sa lijeve strane je prikaz naponskog nivoa od 1.375V na ploči, sa bojama koje predstavljaju naponske nivoe sa skale desno.

Osim pada napona, kao rezultat ove simulacije moguće je vidjeti i gustinu struje, jačinu struje, gustinu snage i disipaciju snage.

3.2. Simulacija PDN impedanse

Kao što je već ranije pomenuto, značaj PDN impedanse pri projektovanju štampanih ploča je ogroman, pa je potrebno izvršiti i simulaciju iste. Prva i osnovna simulacija koju je moguće odraditi, jeste simulacija koju „vidi“ čip odnosno potrošač kad ostale komponente nisu zalemljene na ploču. Drugim riječima, to je impedansa samo štampane ploče. Ova simulacija može biti odradena korišćenjem alata *PowerSI* iz paketa alata *Cadence Sigrity*. Za simulaciju impedanse potrebno je podesiti portove. Jedan port je izvor, a drugi je potrošač. Ponovo je odradena simulacija na istoj liniji napajanja, odnosno 1.375V. Simulacija je odradena za frekventijski opseg od 10kHz do 1GHz. Rezultat simulacije u Z domenu, prikazan je na slici 3.

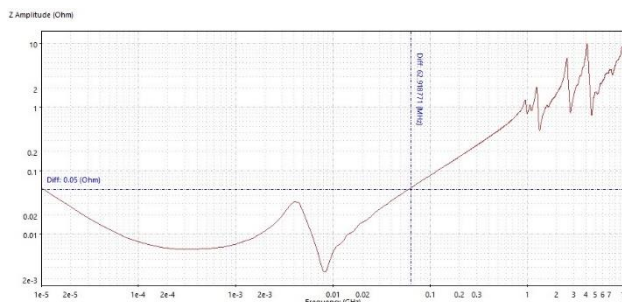


Slika 3. PDN impedansa štampane ploče bez komponentata

Na slici 3. je prikazana amplituda, a osim amplitude moguće je simulirati i fazu, realni i imaginarni dio, Smitov grafik kao i fazno kašnjenje.

3.3. Simulacija impedanse sa dikapling kondenzatorima

Nakon što se proračuna ciljna impedansa, odradi simulacija prazne štampane ploče i odrede dikapling kondenzatori, moguće je izvršiti simulaciju da se provjeri efekat koji dikapling kondenzatori ostvaruju. Potrebno je podesiti sve kondenzatore, odnosno modele kondenzatora. Portovi su podešeni kao što je objašnjeno u prethodnom poglavlju. Rezultati simulacije mogu se vidjeti na slici 4.



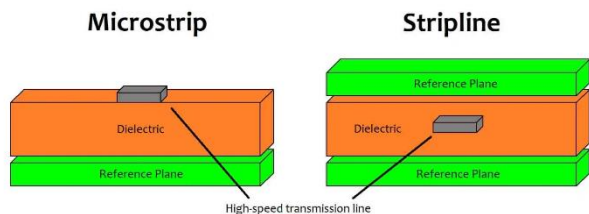
Slika 4. PDN impedansa sa dikapling kondenzatorima

4. TRANSMISIONI VODOVI

Prenos podataka između predajnika i prijemnika se vrši preko prenosnog medijuma koji može biti trasiran i netraisran. U oba slučaja informacija se prenosi prostiranjem elektromagnetnih talasa. Ukoliko je pri prenosu signala vrijeme tranzicije signala kraće od vremena prostiranja signala, tada kažemo da se signal prenosi kroz transmisioni vod.

4.1. Impedansa transmisionih vodova

Kao što je bio slučaj i u analizi napajanja, i kod prenosa signala impedansa je jedan od najvažnijih faktora koji utiču na kvalitet signala. Postoje dvije osnovne strukture vodova na štampanim pločama: mikrostrip vod i strip vod. Mikrostrip vod je vod koji se prostire po površini štampane ploče a strip vod se prostire po nekom od unutrašnjih slojeva štampane ploče. Razlika između mikrostrip i strip voda prikazana je na slici 5.

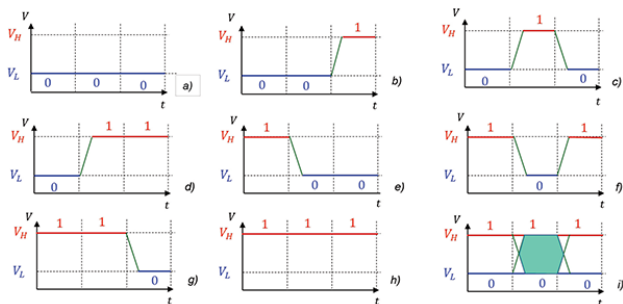


Slika 5. Razlika između mikrostrip i strip vodova

Za proračun impedanse vodova na štampanim pločama postoje brojni onlajn kalkulatori. Većina današnjih alata za projektovanje štampanih ploča takođe ima integrisane alate za proračun impedanse vodova.

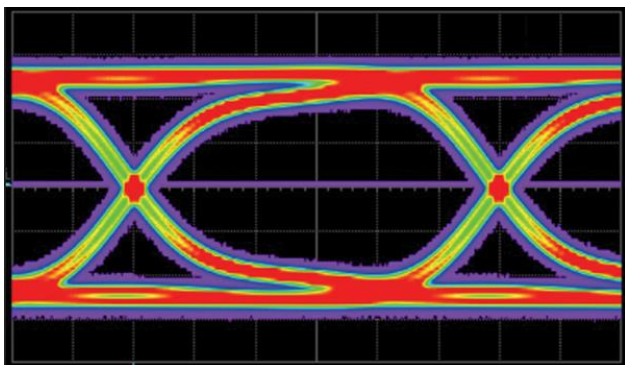
4.2. Dijagram oka

Jedan od prvih i osnovnih mjerenja koje se vrši za provjeru kvaliteta signala jeste dijagram oka. Dijagram oka se dobija superponiranjem rastućih i opadajućih ivica signala u određenom vremenskom periodu a prikazuje se na osciloskopu. Princip dobijanja dijagrama oka prikazan je na slici 6.



Slika 6. Proces dobijanja dijagrama oka

Dijagram oka izmjeren osciloskopom prikazan je na slici 7.



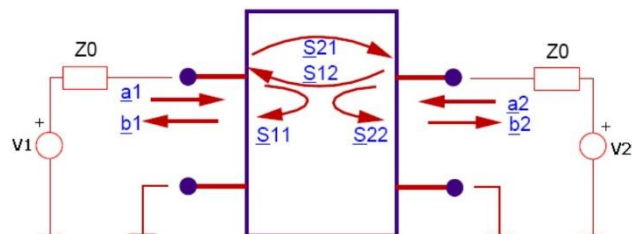
Slika 7. Dijagram oka prikazan na osciloskopu

Parametri koji se obično očitavaju sa dijagrama oka su džiter, varijacije nivoa signala, prosječno vrijeme porasta i opadanja signala i prosječno vrijeme trajanja bita.

4.3. S parametri

S parametri ili parametri rasejanja se koriste za opisivanje linearnih mreža sa n pristupa. Ukoliko se mreža posmatra kao dio transmisione linije, tada je moguće opisati rasejanje i refleksiju talasa korišćenjem S parametara.

S parametri predstavljaju bezdimenzione kompleksne veličine koje zavise od frekvencije i na svakoj frekvenciji se predstavljaju preko amplitude i faze. Na slici 8. je prikazana dvoprístupna mreža sa označenim S parametrima.



Slika 8. Primjer mreže sa dva pristupa

Parametri S_{11} i S_{22} opisuju refleksiju signala na njegovim ulaznim, odnosno izlaznim pristupima. Preko parametara S_{12} i S_{21} opisuje se prostiranje signala od ulaznog do izlaznog pristupa i obratno. S parametri se mogu mjeriti korišćenjem VNA (Vector Network Analyzer) ili se mogu simulirati pomoću različitih softverskih alata.

5. ZAKLJUČAK

Glavna ideja ovog rada, bila je teorijska analiza odnosno opis problema koji mogu nastati prilikom razvoja štampanih ploča.

Bilo je neophodno napraviti uvod u *Power Integrity*, a to je zapravo dio gdje je opisan *Power Delivery Network*.

Nakon toga je dat teorijski uvod u *Power Integrity* simulacije kao i rezultati odrađenih simulacija. Na kraju je dat i teorijski opis transmisionih vodova, koji predstavlja uvod u *Signal Integrity* oblast, i potencijalno *Signal Integrity* simulacije.

Značaj korišćenja simulacija prilikom projektovanja štampanih ploča je očigledna. Simulacije mogu pomoći pri otkrivanju potencijalnih problema tokom razvoja uređaja, a samim tim i do uštede novca.

U današnje vrijeme, alati za simulacije štampanih ploča imaju visoke performanse i samim tim se značaj simulacija povećava.

6. LITERATURA

- [1] Eric Bogatin. Signal and Power Integrity – Simplified. 2nd ed. Pearson Education, Inc, 2004. ISBN: 978-0-13-234979-6.
- [2] Power Integrity Analysis in Your PCB Design Software: <https://resources.altium.com/p/power-integrity-analysis> (pristupljeno u avgustu 2023.)
- [3] Prozessor iMX6: https://www.nxp.com/docs/en/datasheet/IMX6DQCE_C.pdf (pristupljeno u avgustu 2023.)
- [4] Cadence Sigrity: https://www.cadence.com/en_US/home/tools/ic-

[package-design-and-analysis/si-pi-analysis/sigrity-advanced-si.html](https://www.ftn.uns.ac.rs/n4670087/package-design-and-analysis/si-pi-analysis/sigrity-advanced-si.html) (pristupljeno u avgustu 2023.)

- [5] Metoda za određivanje karakteristika širokopojasnog nisko-šumnog pojačavača realizovanog u CMOS tehnologiji:

<https://www.ftn.uns.ac.rs/n4670087/metoda-za-odredjivanje-karakteristika-sirokopojasnog-nisko-sumnog-pojacavaca-realizovanog-u-cmos->

(pristupljeno u avgustu 2023.)

- [6] Brian C. Wadell. Transmission Line Design Handbook. 1st ed. Artech House, Inc, 1991. ISBN: 978-0890064368.

Kratka biografija:



Milomir Spajić rođen je u Ljuboviji 1998. godine
Diplomski rad na Fakultetu tehničkih nauka iz
oblasti Elektrotehnike i računarstva – Primjenjena
elektronika odbranio je 2021.god.
kontakt: milomir.spajic@uns.ac.rs

KOMPARATIVNA ANALIZA PERFORMANSI SISTEMA APACHE KAFKA I RABBITMQ**COMPARATIVE PERFORMANCE ANALYSIS OF APACHE KAFKA AND RABBITMQ SYSTEMS**

Nikolina Tomić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu će biti izvršena komparativna analiza Apache Kafke, moćne platforme za obradu i upravljanje tokovima podataka u realnom vremenu sa RabbitMq-om, koji sa svojom pouzdanom i skalabilnom infrastrukturom za razmenu poruka igra ključnu ulogu u olakšavanju efikasne komunikacije između aplikaciju u kompleksnim sistemima. Rad uključuje i opis implementacije oba ova alata na projektima napisanim u programskom jeziku Java i izvršenog eksperimenta na kom je bazirana analiza.

Ključne reči: Distribuirani sistemi, razmena poruka, pub-sub, Apache Kafka, RabbitMq

Abstract – This work will present a comparative analysis of Apache Kafka, a powerful platform for processing and managing data streams in real time, with RabbitMq, which with its reliable and scalable messaging infrastructure plays a key role in facilitating efficient communication between applications in complex systems. The paper includes a description of the implementation of both of these tools on projects written in the Java programming language and the performed experiment on which the analysis is based.

Keywords: Distributed systems, messaging, pub-sub, Apache Kafka, RabbitMq

1. UVOD

Velike količine novca i intelektualne energije uložene su u tradicionalne sisteme za upravljanje podacima. Ovo upravljanje podacima svodilo se na skladištenje podataka na fajl sistemima i u bazama podataka. Vremenom sve veći problem postaje da se sve spoji u jednu celinu tako da funkcioniše u realnom vremenu [1]. Ovo je problem upravljanja podacima u pokretu. Ovaj problem rešavaju sistemi za razmenu poruka, kao što su Apache Kafka i RabbitMq.

2.1 Pub-sub

Publish subscribe je takođe arhitekturni obrazac koji omogućava asinhronu komunikaciju između različitih komponenti ili servisa u aplikaciji.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dušan Gajić, vanr. prof.

U ovom modelu, publisheri/pošaljci generišu poruke ili događaje i šalju ih bez direktnog znanja o tome ko će ih primiti. Subscriber-i se registruju za primanje određenih vrsta poruka (jedne ili više) i reaguju na njih kada stignu. Ovaj obrazac se često koristi u distribuiranim sistemima, IoT uređajima i aplikacijama koje zahtevaju efikasnu i skalabilnu komunikaciju [2].

2.2 Broker zasnovan na logovima

Broker zasnovan na logovima je softverski sistem gde producer poruke šalje poruku koju broker smešta u log, dok consumer čita poruke sekvencijalno iz loga [3].

3. APACHE KAFKA

Apache Kafka je distribuirani sistem za razmenu poruka zasnovan na pub-sub mehanizmu koji poseduje broker zasnovan na logovima. Služi za procesiranje strimova, podataka u realnom vremenu, kao i za metrike i logovanje. Jedinica podataka unutar Kafke je poruka. Sve poruke su organizovane u teme. Producer smešta poruku u temu, dok se consumeri pretplaćuju na teme i čitaju poruke iz njih [1].

4. RABBITMQ

RabbitMq je softver za razmenu poruka koji se bazira na pub-sub modelu. Koristi se za komunikaciju između visokopropusnih aplikacija i kada se vrši složeno rutiranje do consumera [4].

5. POREĐENJE APACHE KAFKA I RABBITMQ

I Apache Kafka i RabbitMq su sistemi redova poruka koji se koriste u obradi tokova podataka. Oba ova alata pristupaju razmeni poruka iz potpuno različitih uglova.

5.1 Arhitektura

Kafka se oslanja na distribuirani model podataka. Podaci su organizovani u teme, a svaka tema je podeljena u particije. Ovo omogućava horizontalno skaliranje i paralelnu obradu podataka. Takođe Kafka klaster se obično sastoji od više brokera, što doprinosi visokoj dostupnosti i otpornosti na kvarove.

RabbitMq koristi centralizovaniji model obrade podataka. Poruke se šalju na exchange, a zatim se usmeravaju prema redovima na osnovu određenih pravila rutiranja. RabbitMq je tipično organizovan u formi jednog

centralizovanog servera, što ga čini manje složenim u odnosu na Kafku.

5.2 Performanse

I Kafka i RabbitMq nude prenos poruka visokih performansi. Međutim, Kafka je ipak poznata po svojoj niskoj latenciji i sposobnosti za obradu velikih količina poruka u realnom vremenu i samim tim nadmašuje RabbitMq u kapacitetu prenosa poruka.

Kafka može da šalje milione poruka u sekundi jer koristi sekvencijalni disk I/O da omogući razmenu poruka velike propusnosti.

RabbitMq takođe nudi dobre performanse ali latencija može biti nešto veća, posebno u situacijama sa velikim opterećenjem. I RabbitMq se može konfigurisati da šalje milione poruka u sekundi, ali je za to potrebno više brokera. U proseku, performanse su mu hiljade poruka u sekundi.

5.3 Brisanje poruka

Kafka dodaje poruke u log datoteku, koja ostaje dok ne istekne retention period (koji se može konfigurisati). Zbog ovoga, consumeri mogu ponovo obraditi poruke u bilo kom trenutku u predviđenom roku.

RabbitMq broker usmerava poruku do određiškog reda. Consumer kada pročita poruku šalje potvrdni odgovor brokeru, koji zatim briše poruku iz reda.

5.4 Vrste podataka koji se koriste

Kafka najbolje funkcioniše sa operativnim podacima kao što su procesne operacije i aktivnosti sistema, dok je RabbitMq najbolji za transakcijske podatke, kao što su zahtevi korisnika.

5.5 Preuzimanje poruka

Kafka broker služi samo za slanje poruka u red. Produceri objavljuju poruke u red i nisu svesni preuzimanja poruke od strane consumera. Na consumer strani se nalazi sva logika. Aktivni su, prate i čitaju informacije.

Kod RabbitMq se svo ovo rutiranje vrši na brokeru. Producer prati da li je poruka stigla do željenog consumera. Consumeri imaju pasivnu ulogu i samo čekaju da broker stavi poruku u red [5].

5.6 Tok podataka

Kafka koristi neograničen tok podataka, dok RabbitMq koristi ograničeni tok podataka.

5.7 Prioritet poruke

RabbitMq brokeri dozvoljavaju dodavanje poruka u prioritetni red. Umesto slanja i čitanja redovnih poruka po principu "first in first out", broker obrađuje poruke većeg prioriteta pre ostalih.

Za razliku od RabbitMq, Kafka ne podržava prioritete redove i sve poruke tretira isto.

5.8 Redosled poruka

Kafka koristi teme i particije za stavljanje poruka u red. Kada producer pošalje poruku, one idu u određenu temu i particiju. Pošto Kafka ne podržava direktnu razmenu

između producera i consumera, consumeri izvlače poruke sa particije drugačijim redosledom.

RabbitMq šalje i stavlja poruke u red određenim redosledom. Osim ako se poruka višeg prioriteta ne stavi u red, consumeri primaju poruke onim redosledom kojim su one poslate.

5.9 Skalabilnost

RabbitMq može proširiti svoj kapacitet za rukovanje porukama i horizontalno i vertikalno. Uvek se može dodeliti više računarskih resursa RabbitMq serveru da bi se povećala efikasnost razmene poruka.

Isto tako, Kafkina arhitektura dozvoljava dodavanje više particija određenoj temi kako bi se opterećenje poruka ravnomerno raspodelilo.

5.10 Tolerancija na greške

I Kafka i RabbitMq su robusne arhitekture otporne na sistemski otkaz.

Može se grupisati više RabbitMq brokera u klastere i rasporediti se na različite servere. RabbitMq takođe replicira poruke u redu na distribuiranim čvorovima. Ovo omogućava sistemu da se oporavi od kvara koji utiče na bilo koji server.

Kao i RabbitMq, Kafka deli sličnu mogućnost obnavljanja i redundantnost tako što hostuje Kafka klastere na različitim serverima. Svaki klaster se sastoji od replika log datoteke koje se mogu oporaviti u slučaju kvara.

5.11 Logika rutiranja

Kod RabbitMq mogu postojati dva tipa događaja koji se nalaze u istom redu i ne mora se naglašavati consumeru koju poruku prima u svojoj aplikaciji. Ovo odlučuje logika rutiranja koja je dodata u samom RabbitMq brokeru. Napravi se red za svakog consumera. Ukoliko aplikacija treba da obrađuje dva različita događaja, red se konfiguriše da koristi oba ova događaja za tu aplikaciju. Ovako se razdvoji logika rutiranja unutar platforme i ona ne mora da se nalazi u consumeru.

Dok se kod Kafke mora unapred dobro isplanirati koliko se pravi particija u zavisnosti od broja različitih događaja koji se dodaju u red. Sva logika se nalazi u consumeru. Kod Kafke će svaki događaj imati svoju particiju. Ukoliko consumer želi da obrađuje dva događaja on mora da se pretplati na dve particije da bi dobijao ove poruke.

Još jedan način bi mogao biti da se napravi jedna particija sa dva različita događaja, da se na tu particiju pretplate dva consumera: 1 koji želi da prima oba događaja i 1 koji treba da prima samo jedan događaj. U consumeru koji želi da prima samo jedan događaj mogao bi se napraviti filter koji će odbacivati one događaje koje consumer ne treba da obrađuje.

6. KOMPARATIVNA ANALIZA PREFORMANSI

Jako je teško kvantifikovati performanse sistema jer one mogu zavisiti od puno parametara: konfiguracija, hardverski resursi, memorija...

U svrhu ovog istraživanja, sprovedena je detaljna komparativna analiza između dva popularna alata - Apache Kafka i RabbitMq. Za potrebe analize, kreirana su

dva projekta, pri čemu je svaki od njih koristio jedan od navedenih alata. U oba projekta, implementirana je osnovna konfiguracija i dodat je isti Java kod. Jedina razlika između projekata jeste sistem za razmenu poruka koji je integrisan u njih. Na producer strani nalaze se 4 filea iz kojih se podaci čitaju redom, a za svaki podatak se šalju dve poruke. Na consumer strani vrši se agregacija na osnovu pristiglih poruka, zatim se rezultati upisuju u file i meri se vreme potrebno za slanje svih ovih poruka.

U slučaju prvog file-a šalje se 2.000 poruka. Kafka je uspešno obradila ove poruke za 0.2 sec, dok je RabbitMq za to trebalo 2 sec. Razlika je skoro neosetna ali se čak i na ovako malom broju poruka već vidi razlika u performansama, što jasno ukazuje na prednost Kafke. Kafkina log bazirana arhitektura omogućava brzu i efikasnu obradu poruka čak i u manjim scenarijima. Iz ovoga, a takođe i iz truda i vremena uloženog za konfigurisanje sistema u projekte se može doneti zaključak da u nekim situacijama, gde se ne šalje veliki broj poruka ili brzina obrade poruka nije bitna, Kafka nije uvek najbolji izbor zbog njene složenosti. Kafka zahteva određeni nivo konfiguracije i infrastrukture da bi pravilno funkcionisala. Međutim, u manjim scenarijima ovakva kompleksnost može biti nepotrebna i dodatno otežati implementaciju i održavanje. Sa druge strane, RabbitMq se veoma jednostavno konfiguriše, što olakšava rad u manjim okruženjima. Takođe Kafka klaster i ako može biti horizontalno skaliran, i dalje zahteva određene resurse za svoje funkcionisanje. U scenarijima malog broja poruka, često je javlja pitanje da li su troškovi i resursi potrebni za održavanje Kafke opravdani, s obzirom na njihovu potencijalno visoku vrednost. Iz navedenih razloga je u ovakvim situacijama bolje koristiti neki jednostavniji alat za razmenu poruka, kao što je RabbitMq. Ovakvi alati će imati skoro iste performanse a mnogo su lakši za implementaciju i održavanje.

Sledeći file nad kojim su se pokretali projekti jeste file posle čijeg čitanja se šalje 20.000 poruka. Kafka je završila obradu svih poruka za 6 sec, dok je RabbitMq trebalo 10 sec. Kao što se može videti, i u ovom slučaju razlike u performansama i dalje nisu previše izražene, ali Kafkina log bazirana arhitektura omogućava brže pisanje i čitanje poruka čak i kada se koristi jedan broker.

Posle trećeg file-a šalje se 200.000 poruka. U poređenju, Kafka je završila slanje i obradu svih ovih poruka za samo 14 sec, dok je RabbitMq za istu operaciju trebalo čak 48 sec. U ovom slučaju je prvi put prisutna veća razlika. Kafka ima prednost svoje sposobnosti za horizontalno skaliranje, a samim tim povećanje broja poruka nije veliki izazov za Kafka klaster. Sa druge strane, RabbitMq sa svojom centralizovanom arhitekturom pokazuje slabije performanse u scenarijima većeg broja poruka, jer centralni sistem može postati usko grlo.

Poslednji file jeste file nakon čijeg čitanja se šalje 2.000.000 poruka. Kafka je završila slanje i obradu svih ovih poruka za 89 sec, a RabbitMq za 337 sec. Kao što se može primetiti, sa povećanjem broja poruka koje se šalju, razlika između Apache Kafke i RabbitMq je sve veća. U ovako zahtevnim scenarijima Kafka pokazuje svoju punu snagu.

Uz dodatno skaliranje i optimizaciju Kafka bi pokazala još bolje performanse. Kafka je sposobna da obradi ogromnu količinu podataka uz minimalni gubitak performansi. Nasuprot tome, RabbitMq je ograničen svojom centralizovanom prirodom, koja ograničava njegovu sposobnost brze obrade, i zbog koje u ovakvim situacijama zaostaje za Kafkom.

Iz priloženog se može zaključiti da Kafka pruža bolje performanse u poređenju sa RabbitMq, i da ukoliko je brzina slanja i obrade poruka ono što je ključalno za projekat, svakako treba izabrati Kafku. Takođe važno je napomenuti da izbor između ova 2 alata zavisi od konkretnih zahteva projekta. RabbitMq takođe ima svoje prednosti, posebno u slučajevima gde je potrebna jednostavna i brza implementacija ili zahtevno rutiranje.

U oba ova slučaja korišćena je osnovna konfiguracija, tako da se performanse u oba slučaja mogu znatno poboljšati podešavanjem parametara.

7. ZAKLJUČAK

Kafka je posebno dizajnirana da bi se nosila sa zahtevima visokih performansi, skalabilnosti i izdržljivosti. Njena sposobnost horizontalnog skaliranja, kao i sposobnost efikasnog upravljanja velikom količinom podataka, čini je izuzetno pogodnom za scenarije koji zahtevaju obradu u realnom vremenu i sisteme zasnovane na događajima.

Međutim i ako Kafka ima brojne prednosti, uspeh zavisi od pažljivog planiranja, projektovanja i konfigurisanja. Integracija sa postojećim sistemima zahteva duboko razumevanje kako Kafka funkcioniše i kako je najbolje iskoristiti njene performanse.

Sa druge strane, RabbitMq nudi širok spektar mogućnosti za upravljanje redovima. Osim toga, pruža podršku za različite tipove razmene poruka, što omogućava fleksibilnost u komunikaciji. Takođe, RabbitMq podržava prioritet poruka, i osim standardnog AMQP protokola podržava i druge protokole komunikacije kao što su MQTT i STOMP, pružajući tako više opcija za razmenu poruka. Zahvaljujući raznovrsnosti alata i biblioteka koje podržavaju RabbitMq, moguće je lako integrisati ovaj sistem sa raznim aplikacijama i tehnologijama, što ga čini prilagodljivim i pogodnim za različite scenarije.

Na kraju krajeva, najbolji alat zavisi od potreba. Iz ličnog iskustva mogu reći da je Apache Kafka mnogo korišćeniji alat na komercijalnim/klijentskim projektima, i da je njene performanse, skalabilnost, izdržljivost i sposobnost za rad sa događajima čine jako moćnim resursom u modernom svetu podataka. Primena Apache Kafke u raznim domenima otvara mnoge mogućnosti za buduća istraživanja i inovacije u oblasti obrade podataka.

8. LITERATURA

- [1] Gwen Shapira, Todd Palino, Rajini Sivaram, Krit Petty, "Kafka The Definitive Guide - Real-Time Data and Stream Processing at Scale", O'Reilly Media, 2021
- [2] Martin Kleppmann, "Designing data-intensive applications", O'Reilly Media

- [3] “Traditional Message Queues vs. Log-based Message Brokers”, <https://rkenmi.com/posts/traditional-message-queues-vs-log-based-message-brokers/>
- [4] LinkedIn, “RabbitMq Features & Architecture”, <https://www.linkedin.com/pulse/rabbitmq-features-architecture-huzaifa-asif/>, poslednji pristup 23.9.2023.
- [5] aws, <https://aws.amazon.com/compare/the-difference-between-rabbitmq-and-kafka/>, poslednji pristup 23.9.2023.

Kratka biografija:



Nikolina Tomić rođena je u Novom Sadu 24.12.1996. godine. Upisala fakultet 2015. godine, smer Elektrotehnika i računarstvo. Diplomirala 2020. godine sa temom „Bajesove mreže“.



РАЗВОЈ РЕСПОНЗИВНЕ ВЕБ АПЛИКАЦИЈЕ ЗА ПОДРШКУ РАДУ
НАЦИОНАЛНОГ САВЕЗА ЗА КОЊИЧКИ СПОРТ

DEVELOPMENT OF A RESPONSIVE WEB APPLICATION TO SUPPORT THE WORK
OF THE NATIONAL ASSOCIATION FOR EQUESTRIAN SPORTS

Сара Попарић, Факултет техничких наука, Нови Сад

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – Овај рад се бави развојем респонзивне веб апликације за подршку раду националног савеза за коњички спорт, односно апликације која је прилагодљива за коришћење са различитих уређаја. Анализом веб апликација националних и међународних савеза за коњички спорт примећено је да та решења нису погодна за коришћење са мобилних телефона, већ само са лаптоп и десктоп рачунара. Са друге стране, дипломски рад, који представља полазну основу за овај рад, имплементиран је само као мобилна апликација. Због тога, решење имплементирано у овом раду представља респонзивну веб апликацију која може да се користи независно од уређаја који корисник поседује, што је потврђено тестирањем апликације са различитих уређаја.

Кључне речи: мобилна апликација, респонзивна веб апликација, ЦСС, Тејлвинд

Abstract – This paper deals with the development of a responsive web application to support the work of the National Association for Equestrian Sports; an application that is adaptable for use on different devices. Analyzing the web applications of national and international associations for equestrian sports, it was noticed that those solutions are not suitable for use from mobile phones, but only from laptops and desktop computers. On the other hand, the bachelor thesis, which is the starting point for this thesis, was implemented only as a mobile application. Therefore, the solution implemented in this paper represents a responsive web application that can be used independently of the device the user owns, which was confirmed by testing the application from different devices.

Keywords: mobile application, responsive web application, CSS, Tailwind

1. УВОД

Савез за коњички спорт Србије тренутно не поседује систем који би његовим члановима, али и запосленима омогућио обављање административних послова [1].

НАПОМЕНА:

Овај рад произтекао је из мастер рада чији ментор је био др Драган Дину, ванр. проф.

Радам националног савеза за коњички спорт бавио се дипломски рад под насловом „Софтверски пакет за подршку рада националног савеза за коњички спорт” који је реализован као мобилна апликација. У датом раду имплементирани су основне функционалности потребне за рад овог савеза.

Главни недостатак који је примећен при анализи решења дипломског рада је немогућност коришћења апликације са лаптопа, десктоп рачунара или таблета, већ искључиво са мобилног телефона. Из тог разлога, главни задатак овог рада је развој апликације која ће бити погодна за коришћење независно од уређаја који корисник поседује. Како би се то омогућило развијена је нова респонзивна веб апликација; односно апликација коју је могуће користити независно од уређаја, оперативног система, величине или оријентације екрана, или од врсте Интернет претраживача који корисник поседује [2].

Након детаљне анализе поменутих проблема и већ постојећих решења за националне коњичке савезе, у раду је предложено решење у виду дизајна и имплементације респонзивне веб апликације за подршку раду националног савеза за коњички спорт. Такође, рад показује како је уз употребу Тејлвинд (енгл. Tailwind) радног оквира могуће лако и брзо креирати респонзиван кориснички интерфејс.

2. ПРЕТХОДНО ИСТРАЖИВАЊЕ

Мотивација за тему дипломског рада „Софтверски пакет за подршку рада националног савеза за коњички спорт” произилази из чињенице да се коњички спорт у Србији сусреће са бројним препрекама, које спречавају бржи развој самог спорта [1].

Детаљном анализом постојећих софтверских решења која се користе у раду како националних тако и међународних коњичких савеза изведен је закључак да већина постојећих решења не нуди кориснички интерфејс који је погодан за коришћење уз помоћ мобилног телефона, већ само једноставан и прегледан кориснички интерфејс у случају њиховог коришћења путем лаптопа или десктоп рачунара. Из наведених разлога и све веће популарности мобилних телефона, при изради дипломског рада одлучено је да софтверско решење буде реализовано као мобилна апликација.

Како су главни проблеми са којима се национални савези сусрећу проблеми везани за администрацију података везаних за чланове савеза, кључне функцио-

налности које имплементирано решење подржава су: администрација података везаних за коњичке клубове, администрација података везаних за јахаче, администрација података везаних за коње и управљање календаром такмичења [1].

Како је уочено при анализи постојећих решења да већина нуди интуитиван и прегледан кориснички интерфејс, али само за уређаје са већим екранима, док са друге стране решење имплементирано у дипломском раду нуди прегледност само преко мобилних уређаја, долази се до закључка да је потребно пронаћи решење које би омогућило коришћење софтвера независно од уређаја и величине његовог екрана. Због тога, ново решење представља респонзивну веб апликацију, која подразумева развој јединствене апликације чији би се кориснички интерфејс прилагођавао зависно од величине екрана уређаја са ког се приступа апликацији [2].

3. РАЗВОЈ СОФТВЕРСКОГ РЕШЕЊА ЗА РЕСПОНЗИВНУ ВЕБ АПЛИКАЦИЈУ

Респонзивни Веб Дизајн (енгл. *Responsive Web Design* - *RWD*) представља начин креирања веб апликација тако да не зависе од уређаја преко кога се користе. Код ових апликација садржај је респонзиван - прилагодљив величини екрана на ком се приказује сама апликација, за разлику од статичних садржаја који захтевају стално зумирање, скроловање и померање садржаја.

Овим се постиже доживљај да је апликација прављена баш за коришћени уређај, без обзира да ли је он мобилни телефон, таблет или десктоп рачунар. Респонзивни дизајн не скалира само изглед страница на основу величине екрана на ком се садржај приказује, већ може укључивати и промену целокупног дизајна садржаја [2].

3.1. Развој респонзивних веб апликација коришћењем ЦСС кода

ЦСС (енгл. *CSS* - *Cascading Style Sheets*) представља језик за дефинисање презентације докумената написаних помоћу ХТМЛ-а (енгл. *HTML* - *HyperText Markup Language*) или ИксЕмЕл-а (енгл. *XML* - *Extensible Markup Language*). Служи за описивање како би елементи требало да буду приказани на екрану, папиру или неком другом медију, што се остварује описом одређених особина елемената (боја, позиција, облик итд.) [3].

Особине елемената, се помоћу ЦСС-а, дефинишу пратећи следећу синтаксу: особина: вредност;. Особина (енгл. *property*) представља атрибут елемента који се жели променити, и свака поседује скуп дозвољених вредности које могу бити различитог типа. Ове јединице могу бити **апсолутне** и **релативне**.

Апсолутним јединицама се осигурава да одређена компонента, односно да одређена особина компоненте не мења своју вредност зависно од величине екрана на којем се приказује. У апсолутне јединице спадају: *mm*, *cm*, *in*, *px*. Иако не доприносе респонзивности апликације, веома су битне, јер је врло често потребно да се нека особина одржи стално (на пример, дебљина ивице - *border-width*) [4].

Релативне јединице обезбеђују да компонента мења своју величину зависно од величине екрана уређаја на ком се приказује. Оне се користе како би се обезбедила респонзивност веб страница. Ако би се све мере оставиле у апсолутним јединицама, сам изглед странице би морао да се предефинише како би се величине ускладили са променама величине екрана. У релативне јединице спадају: *em*, *rem*, *%*, *vw*, *vh*, *ch* [4].

Некада коришћење релативних јединица није довољно да би се обезбедила потпуна респонзивност, јер се она не огледа само у величини компоненти. Често је потребно неке компоненте не приказати или им променити распоред зависно од величине екрана. Ово је у ЦСС коду могуће учинити коришћењем упита за уређаје (енгл. *media query*). Они представљају упите помоћу којих је могуће таргетирање не само одређених класа уређаја већ и раздвајање типова уређаја по њиховим физичким карактеристикама. Синтакса која се користи за писање ових упита је:

<тип медија> **not|only|and**, (<упит>) [5].

Тип медија (енгл. *mediatype*) представља једну од најважнијих карактеристика описа стилова (енгл. *stylesheet*), јер дефинише како ће документ (ХТМЛ страница) бити приказан на различитим медијима.. Најчешће коришћени типови медија су: *all*, *print*, *screen*, *speech* [6].

После типа уређаја долазе кључне речи које служе за спајање односно раздвајање услова самог упита, и могуће је да се појаве неке од следећих: *and*, *not*, *only* или *or* (.) [5].

Резултат коришћења упита за медије је: ако карактеристике уређаја задовољавају услов задат у упиту, особине дефинисане помоћу ЦСС кода ће бити примењене на документ.

У наставку су дати упити за медије који се користе за дефинисање стилова зависно од уређаја [7]:

- за таблете: **media screen and (min-width: 480px) and (max-width: 1024px)**,
- за лаптоп и десктоп рачунаре: **@media screen and (min-width: 1024px)**,
- за велике екране: **@media screen and (min-width: 1884px)**.

3.2. Развој респонзивних веб апликација коришћењем Тејлвинд радног оквира

Тејлвинд (енгл. *Tailwind*) представља радни оквир ниског нивоа који се користи за брзо писање и једноставно прилагођавање потребама апликације, без потребе за писањем ЦСС кода. Сматра се радним оквиром ниског нивоа зато што за разлику од других не нуди унапред креиране компоненте [8]. Такође, сматра се радним оквиром којем је употребљивост на првом месту (енгл. *utility-first*) што значи да нуди колекцију класа које је користе за стилизовање веб страница [9].

Стилизовање ХТМЛ (енгл. *HTML*) елемената се дешава у самим елементима, односно у оквиру класног (енгл. *class*) атрибута елемента. Такође, и дефинисање промена у изгледу странице зависно од величине екрана се обавља у ХТМЛ фајлу. Све што је

потребно да би се дефинисао изглед зависно од величине екрана јесте да се дода префикс **преломна тачка**: испред назива класе. Преломне тачке (енгл. *breakpoint*) су унапред дефинисане, и одговарају најчешћим ширинама екрана [10]: **sm** (минимална ширина екрана 640px), **md** (минимална ширина екрана 768px), **lg** (минимална ширина екрана 1024px), **xl** (минимална ширина екрана 1280px), **2xl** (минимална ширина екрана 1536px).

Битно је напоменути да се респонзивност веб страница коришћењем Тејлвинд радног оквира обавља по принципу да се прво стилизује изглед странице за мобилне телефоне (енгл. *Mobile-first*), па онда да уз помоћ преломних тачака дефинише изглед за веће екране [10].

Из овога се може закључити да се креирање респонзивних апликација помоћу Тејлвинда своди на писање одговарајућих упита за медије, пратећи одговарајућу синтаксу.

4. РЕШЕЊЕ

Веб апликација имплементирана као надоградња на раније поменути дипломски рад, представља његово проширење, не само у погледу корисничког интерфејса, већ и са аспекта функционалности, услед промене корисничких захтева од стране клијента. Осим што је скуп функционалности проширен, могућности обављања административних послова, уз помоћ апликације, је подељена међу улогама у систему (админ, заступник коњичког клуба, тренер, службено лице). На овај начин је смањено оптерећење запослених у савезу, али је смањена и могућност грешке, јер подацима управљају они који их најбоље познају.

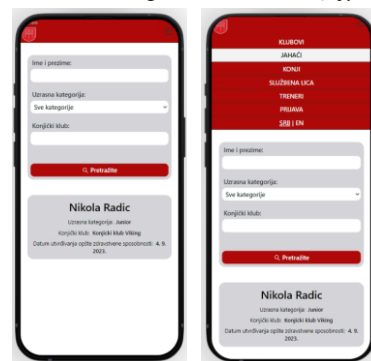
Компоненте апликације на којима је имплементирана примена респонзивности, коришћењем Тејлвинд радног оквира, односно примена апсолутних и релативних јединица, и промена изгледа компоненти зависно од величине екрана, су: трака за навигацију, компонента за претрагу јажача, компонента за преглед регистрованих јажача, трака за навигацију унутар странице са захтевима за регистрацију.

Трака за навигацију (слика 1) представља компоненту која је увек присутна на екрану корисника. Она омогућава корисницима да се брзо и једноставно крећу кроз странице апликације, а такође и омогућава корисницима информацију о томе на којој страници апликације се тренутно налазе. Како ова компонента треба увек да заузима читаву ширину екрана, за њену ширину су коришћене релативне јединице, конкретно 100vw, чиме се постиже ширина од 100% прозора за приказ уређаја.

Такође, и величина текста, маргине између линкова, као и удаљеност од ивица (енгл. *padding*) око тренутно активног линка користе релативне мере како би пратиле величину екрана. Да би трака за навигацију била прегледна и на мањим уређајима уобичајена пракса је да се не приказује класична трака за навигацију, већ да се она сузи у такозвани „хамбургер” (слика 2) који представља дугме, на које када се кликне излиставају се сви линкови које трака за навигацију садржи.



Слика 1. Трака за навигацију

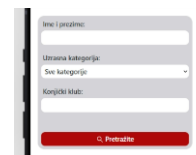


Слика 2. Изглед траке за навигацију на мањим екранима

Компонента за претрагу јажача (слика 3) састоји се из неколико поља за унос текста са одговарајућим лабелама и једним дугметом помоћу којег се врши сама претрага. Респонзивност ове компоненте се огледа у респонзивности њене величине, али и у респонзивности компоненти унутар ње. Величина ове компоненте би требала да заузме скоро читаву ширину екрана, и да са бочних страна има мале маргине. Да би поља за унос и текст унутар њих био увек видљив од средње величине екрана (по Тејлвинд радном оквиру) па ка мањим величинама, уместо хоризонталног распореда поља за унос, коришћен њихов вертикални распоред (слика 4). На овај начин је осигурано да корисници увек имају увид у оно што су унели у поље за претрагу.

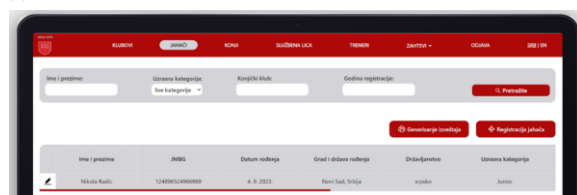


Слика 3. Компонента за претрагу јажача

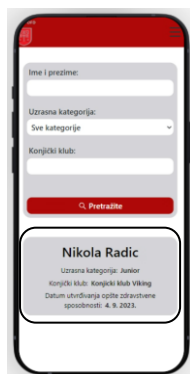


Слика 4. Компонента за претрагу јажача за мање екране

Изглед компоненте за преглед регистрованих јажача може бити приказана уз помоћ табеларног приказа (слика 5) или картица (слика 6). Разлика између ова два приказа је, осим очигледног коришћења различитих визуелних компоненти, и у количини података који могу бити приказани у оквиру њих. Због кога је за мање уређаје коришћен приказ уз помоћ картица како би корисницима најприоритетнији податаци увек били прегледно приказани. Са друге стране, код већих екрана подаци се приказују у табеларном облику, како би корисник имао могућност прегледа веће количине података.

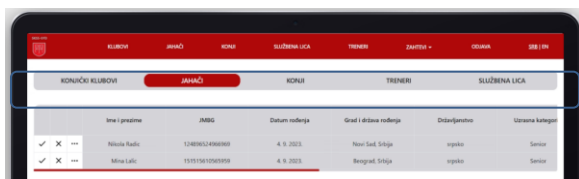


Слика 5. Табеларни преглед регистрованих јажача

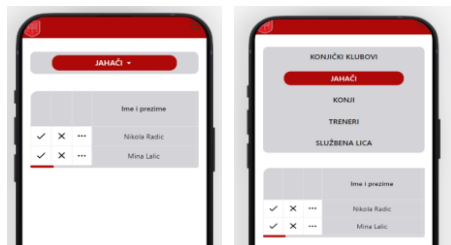


Слика 6. Подаци за јахаче у оквиру картица

Страница са захтевима за регистрацију служи админу како би имао увид у све пристигле захтеве за регистрацију свих могућих ентитета у систему (коњичких клубова, јахача, коња, тренера и службених лица), а да би знао које захтеве тренутно прегледа они су подељени зависно од тога за који су ентитет послати. Трака за навигацију унутар ове странице, осим што на великим екранима (слика 7) прати принципе респонзивности као што су њена ширина, маргине, као и линкови који се налазе унутар ње дефинисани помоћу релативних јединица, има и промену дизајна за мање екране. Ова промена се огледа у томе што када се апликацији приступа путем уређаја са мањим екранима трака за навигацију се приказује као падајући мени (слика 8).



Слика 7. Трака за навигацију кроз захтеве



Слика 8. Трака за навигацију кроз захтеве на мањим екранима

5. ЗАКЉУЧАК

У оквиру овог рада представљена је респонзивна веб апликација намењена за подршку рада националног савеза за коњички спорт. Апликација је реализована коришћењем Риект (енгл. *React*) радног оквира, док је за стилизовање визуелних компоненти коришћен Тејлвинд (енгл. *Tailwind*) радни оквир.

Сам рад представља проширење раније поменутог дипломског рада, а главни разлог за прелазак са мобилне на респонзивну веб апликацију је могућност приближавања саме апликације већем броју корисника, јер веб апликацију не морају да имају инсталирану на свом мобилном телефону. Поред тога респонзивна одлика апликације пружа могућност њеног коришћења и са других уређаја, као што су лаптопови, десктоп рачунари и таблети.

Даље истраживање ће укључити тестирање апликације од стране различитих корисника, међу којима ће бити и одговорна лица Савеза за коњички спорт, како би апликација била максимално искористива будућим корисницима. Поред тога, анализираће се и нове функционисности које би могле да унапреде рад Савеза у будућности.

6. ЛИТЕРАТУРА

- [1] Сара Попарић, „Софтверски пакет за подршку рада националног савеза за коњички спорт“, 2022. [Последњи приступ: 18.09.2023.]
- [2] Oxford Web Studio: Responsive dizajn, урл: <https://www.oxfordwebstudio.com/responsive-dizajn.html> [Последњи приступ: 06.09.2023.]
- [3] CSS: Cascading Style Sheets, урл: <https://developer.mozilla.org/en-US/docs/Web/CSS> [Последњи приступ: 22.09.2023.]
- [4] mdn web docs: CSS values and units, урл: https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units. [Последњи приступ: 11.09.2023.]
- [5] Ethan Marcotte, „Responsive web design“, 2010, урл: <https://alistapart.com/article/responsive-web-design/> [Последњи приступ: 07.09.2023.]
- [6] W3: Media types, урл: <https://www.w3.org/TR/CSS21/media.html> [Последњи приступ: 07.09.2023.]
- [7] Cem Eygi: Media Query CSS Tutorial – Standard Resolutions, CSS Breakpoints, and Target Phone Sizes, урл: <https://www.freecodecamp.org/news/css-media-queries-breakpoints-media-types-standard-resolutions-and-more/> [Последњи приступ: 22.09.2023.]
- [8] Anna Fitzgerald, „Tailwind CSS: What It Is, Why Use It & Examples“, 2022, урл: <https://blog.hubspot.com/website/what-is-tailwind-css> [Последњи приступ: 18.09.2023.]
- [9] Tailwind CSS: Utility-First Fundamentals, урл: <https://tailwindcss.com/docs/utility-first> [Последњи приступ: 15.09.2023.]
- [10] Tailwind CSS: Responsive Design, урл: <https://tailwindcss.com/docs/responsive-design> [Последњи приступ: 18.09.2023.]

Кратка биографија:



Сара Попарић рођена је у Сомбору 28. септембра 1999. године. Факултет техничких наука у Новом Саду, студијски програм Рачунарство и аутоматика уписала је 2018. године. Након завршених основних студија, 2022. године, уписала је мастер академске студије из исте области. контакт: poparic.sara@uns.ac.rs

ГЕНЕРИСАЊЕ АРХИТЕКТОНСКИХ ОСНОВА УЗ ПОМОЋ МОДИФИКОВАНОГ ГЕНЕТСКОГ АЛГОРИТМА**GENERATING FLOOR PLANS USING A MODIFIED GENETIC ALGORITHM**

Милица Ињац, Факултет техничких наука, Нови Сад

Област – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – У раду је приказана једна модификација генетског алгоритма са применама у генерисању архитектонских основа, односно распореда и организације просторија у оквиру једне етаже стамбеног или пословног грађевинског објекта. У раду су посебно разматрани проблеми кодирања јединки, те имплементација оператора укрштања и мутације.

Кључне речи: Генетски алгоритам, архитектонске основе, критеријум оптималности

Abstract – This paper presents a modification of the genetic algorithm with applications in the generation of floor plans, that is, the layout and organization of rooms within one floor of a residential or commercial building. Paper addresses and discusses the problems of encoding individuals, and the implementation of crossover and mutation operators.

Keywords: Genetic algorithm, architectural plan, fitness function

1. УВОД

У раду је приказана једна модификација генетског алгоритма са применама у генерисању архитектонских основа, односно распореда и организације просторија у оквиру једне етаже стамбеног или пословног грађевинског објекта.

Воде се дебате о томе како је људско размишљање условљено пређашњим учењем и самим тим остаје увек у одређеним границама тог учења. Поставља се питање да ли оно научно ограничава креативност [1]. Истражују се приступи који би лишили решење дотадашњих идеја дизајна. Као једно од потенцијалних решења намећу се разни алгоритми. У овом раду је у те сврхе употребљен генетски алгоритам, који је првобитно модификован.

Многи елементи архитектонског пројектовања су суштински креативни и ослањају се на интуицију и даровитост појединца. На свом текућем нивоу развоја ни једна метода вештачке интелигенције није нашла начин да у потпуности замени креативно деловање човека.

Тема овог рада није, дакле, да се направи још један такав покушај, већ напротив да се предложи метода,

односно понуди алат, који би омогућио архитектама и машинама да стварају заједно, синергетски.

1.1. Преглед литературе

Архитектонско пројектовање и визуелизација је сложен процес, који захтева искуство и креативност архитеката. Генерисање новог дизајна уз помоћ разних алгоритама вештачке интелигенције је област која се широко истражује. Примена вештачке интелигенције у архитектонском дизајну је у порасту, но постоје неки важни фактори и изазови које треба размотрити. Набавка скупа података високог квалитета који је потребан за рад неких од алгоритама представља проблем. Поред тога неки алгоритми захтевају предобраду података, а то је поступак који одузима доста времена [2].

Chaillou [3] је користио генеративне супарничке мреже за генерисање архитектонских тлоцрта користећи модел обучен над скупом слика. У његовом раду следило је неколико корака да се коначно генеришу потпуно опремљени архитектонски планови зграде када се као улаз даје облик земљишта. А главни закључак је уједно и позив истраживачима да истраже примену машинског учења у архитектури, где напомиње значај свог рада као одскочну даску за будуће дискусије. Генеративне супарничке мреже у оригиналном, или на неки начин модификованом облику за генерисање основа користили су и: Nauata et al. [4], Radford et al. [5] и Huang et al. [6].

Иако је примена овог типа алгоритма обећавајућа, има значајне изазове и недостатке. Главни недостатак алгоритма је што није у стању да генерише заиста нов и иновативан дизајн. Разлог томе је обучавање над готовим дизајнима основа [2].

Предност генетског алгоритма, који је модификован и као такав коришћен у овом раду, је управо у томе што не захтева базу података, ни предобраду података. Алгоритам генерише нов дизајн у зависности од критеријума оптималности, а не већ виђених дизајна. Међутим, он има неке друге недостатке.

1.2. Опис проблема

Решава се проблем генерисања стана уз помоћ генетског алгоритма. Проблем се може свести на два потпроблема:

- модификацију генетског алгоритама
- распоређивање соба у стану

НАПОМЕНА:

Овај рад произтекао је из мастер рада чији ментор је била др Мирна Капетина, ванр. проф.

Што се тиче генетског алгоритма, потребно га је прилагодити конкретном проблему. Излаз из генетског алгоритма треба да буде објекат.

Потребно је одлучити на који начин ће се кодирати јединка (стан). Потребно је формирати, модификовати и имплементирати све оно што класичан генетски алгоритам подразумева: селекцију, укрштање и мутацију.

Потребно је решити проблем правилног избора свих параметара алгоритма. Један од изазова кад је у питању генетски алгоритам јесте исправано формулисан и касније имплементиран критеријум оптималности.

У конкретном проблему који се решава, јединка представља стан који се састоји од више соба. Потребно је одредити облик, димензије и распоред соба унутар стана.

Начин на који се собе распоређују унутар стана представља други велики проблем. Не сме да се догоди да нека соба није у контакту ни са једном другом. Такође, не сме да дође до преклапања соба унутар стана. Облик генерисаног стана треба да буде смислен, односно без празног простора међу собама у границама у којој уопште није одрживо бавити се таквим пројектовањем. Дакле, проблем је налажење оптималног начина распореда соба унутар стана, као и димензије сваке просторије.

2. ПРЕДЛОЖЕНО РЕШЕЊЕ

Концептуално решење описаног проблема може се поделити у две целине.

2.1. Распоређивање соба у стану

Стан представља јединку и састављен је од низа међусобно повезаних соба. У алгоритму је свака соба представљена као правоугаоник, чије су координате, висина и дужина познате.

Генерише се нулта просторија у координатном почетку. Наредна соба се поставља испод и лево од ње, на начин да им се само ћошкови додирују. Након поставке, соба се креће на предефинисан начин. Кретање је такво да соба која се креће мора у сваком моменту да буде у контакту са неком од већ постављених соба у стану. Соба се креће тако што клизи по зидовима других соба. Колико дуго се соба креће предефинисано је параметром делта. Оног момента кад делта буде једнако нули, соба је позиционирана у стану.

Циклус кретања подразумева постављање собе на најнижу леву тачку у простору. Након тога се креће у смеру супротном од смера кретања казаљке на сату. Нпр. новоформирана соба се креће ка десно колико јој је то дозвољено.

Уколико нема никакву препреку, у виду зида неке друге собе, креће се максимално десно. Након тога, креће се ка горе, лево и на крају доле, враћајући се у свој почетни положај.

Пример кретања једне собе око нулте просторије приказан је на првој слици.



Слика 1. Кретање једне просторије у односу на нулту

Усложњена ситуација је кретање собе уколико постоје препреке у виду зидова других соба. У овом случају просторија се креће једним смером или док не дође до зида, или док не дође до максималног стања. Уколико током кретања соба наиђе на зид, мења правац кретања. Не може да настави кретање у датом смеру, због препреке, а не може ни да остане на истом правцу, јер може да упадне у бесконачну петљу. Не може да се врати одакле је дошла. Да би се то избегло, уведено је ограничење које захтева промену правца кретања оног момента кад се дође до зида. Правило има изузетак у случају да је соба окружена са три своје стране другим собама, тада може да се врати уназад, јер не може нигде другде. Вредност параметра делта је у сваком моменту позната и освежава се након сваког померања собе.

2.2. Генетски алгоритам

Генетски алгоритам је модификован на неколико нивоа у сврхе решавања конкретног проблема. Собе унутар стана представљају хромозоме јединки, док стан са свим својим собама представља јединку. Више различитих станова представљају популацију. Јединке нису кодиране ни бинарно, ни реално, већ је вршена модификација.

За почетак свака соба носи информацију о једној координати (левој доњој), висини и ширини. А сваки стан је кодиран на начин да носи информације о свим собама од којих је сачињен и самим тим познаје њихову позицију, висину и ширину. Механизам поређења станова разликује добар стан од лошег стана, односно фаворизује добре станове. Шта значи добар стан оставља се за дискусију. Формирање критеријума оптималности вршено је кроз неколико итерација, конкретан развој и размишљања представљена су у дискусији.

Формирани су и имплементирани механизми за селекцију, укрштање и мутацију.

Имплементирана је класична турнир селекција, без измена.

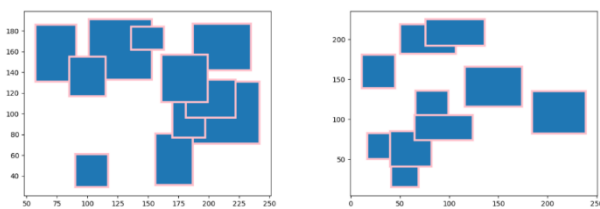
Укрштање представља изазов. Замисао је укрштати станове на начин да се нађу парови соба за укрштање из два стана који су кандидати за укрштање. Проблем до ког долази јесте неједнак број соба у становима. Да би се решио тај проблем уведен је параметар који

означава видљивост. Свака соба може бити видљива, или невидљива, где је збир свих видљивих и невидљивих соба у сваком стану једнак. У остатку алгоритма барата се са видљивим собама. Невидљиве собе служе за укрштање станова. Укрштање два стана врши се укрштањем по две собе, по једна из сваког стана. За сваке две собе које се укрштају добију се нове две собе. Собе се укрштају тако што се комбинују вредности висине, ширине, видљивости и параметра делта.

Финални резултат укрштања два стана су нова два стана настала комбинацијом претходна два. За мутацију је одлучено да се мутира мањи број јединки у већој мери. Мутација се врши на начин да се промене вредности висине, ширине, видљивости и параметра делта за одређени број соба у стану. Мутација се врши само над малим бројем јединки како насумичност не би преовладала.

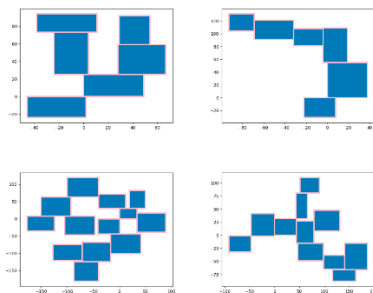
3. РЕЗУЛТАТИ И ДИСКУСИЈА

У првој итерацији решавања проблема, фокус је на решавању најједноставнијег могућег случаја, тј. генерисања соба у простору на било који начин. На другој слици представљени су резултати прве итерације. Примећује се да се неке просторије преклапају, а неке уопште не додирују. То је проблем који је решаван у наредној итерацији.



Слика 2. Прва фаза развоја алгоритма

Резултати измењеног алгоритма, који води рачуна о томе да не дође до преклапања и који захтева повезаност свих соба приказани су на трећој слици. Примећује се да у овом случају станови имају облик који није одржив.



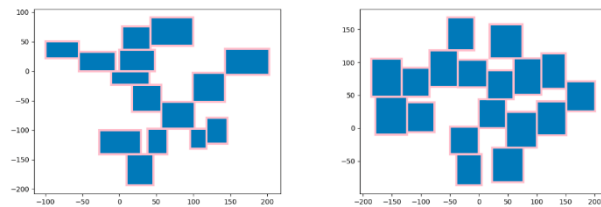
Слика 3. Модификована верзија алгоритма

Тежња наредне фазе је да стан добије облик који би био реалан и изводљив. Главни захтев је да се минимизује празан простор између соба. То се постигло модификацијом критеријума оптималности.

3.1. Критеријум оптималности

Као што је већ споменуто, више критеријума оптималности је имплементирано и вршени су експерименти са разним њиховим комбинацијама.

Иницијално је имплементиран најједноставнији критеријум у сврхе провере рада алгоритма. То је критеријум који фаворизује велике станове. У овом случају алгоритам је врло брзо научио да велики станови имају велики број просторија и тежи томе да повећава видљивост просторија током итерација (слика 4).

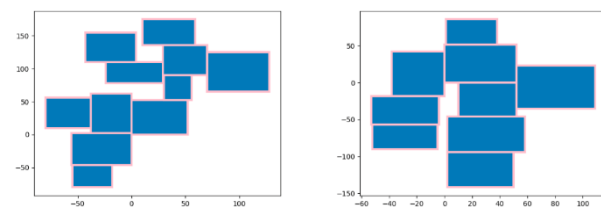


Слика 4. Станови генерисан помоћу критеријума оптималности који фаворизује већу површину

Наредни критеријум оптималности је први од неколико којим покушава да се реши проблем расутости просторија. То је критеријум сума удаљености соба од координатног почетка. Рачуната је удаљеност сваке собе од координатног почетка, где је циљ да та удаљеност буде што мања. Са овим критеријумом је делимично решен проблем разуђености, јер су резултујући станови сконцентрисани око координатног почетка, међутим собе и даље нису довољно повезане и у резултујућем стану су и даље присутне празнине између соба.

Наредни критеријум оптималности минимизује обим. Идеја је да станови са мањим обимом природно теже да густо ређају собе. Резултат овог критеријума су станови са малим бројем соба, у неким случајевима без видљивих соба. Овај резултат је одбачен као недовољно добар, јер стан од пар соба нема увек смисла.

Претходни критеријум оптималности модификован је комбиновањем са површином. Више комбинација је испробано, а за најбољу се испоставило да је количник површине и обима. Овај критеријум оптималности тиме максимизује површину на уштроб обима. Међутим, резултујући стан понекад није довољно повезан и дешава се да су собе повезане само једном тачком, што у реалности нема смисла (слика 5).

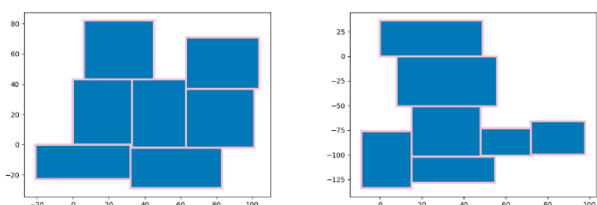


Слика 5. Станови генерисан помоћу критеријума оптималности који минимизује обим, а максимизује површину

Последњи критеријум који је имплементиран максимизује дужине дељених ивица. Идеја овог критеријума јесте да се директно утиче на проблем недовољне повезаности соба унутар стана. Прва итерација овог критеријума је фаворизовала мање станове (3-6 соба) који су били максимално повезани. Станови су

најзад добили облик који има смисла. Како би се критеријум додатно унапредио, направљена је модификација.

Финална верзија критеријума оптималности је количник дужине дељених ивица и обима стана. Овим количником се максимизује дужина дељених ивица и притом се минимизује обим стана. Минимизација обима је, испоставило се, битан додатак који је као резултат дао додатно повезане станове. Примери станова генерисаних коришћењем датог критеријума оптималности су приказани на слици 6.



Слика 6. Станови генерисан помоћу критеријума оптималности који минимизује обим, а максимизује дељене ивице

4. ЗАКЉУЧАК

У раду је представљен један од начина генерисања архитектонских основа. Модификован је и имплементиран генетски алгоритам за примену у архитектонском пројектовању, где се кроз више итерација дошло до задовољавајућег решења. На овај начин генерисано је решење које није у зависности од старих идеја дизајна. Промена критеријума оптималности утиче на промену крајњег резултата. Но, архитектонски смислена решења се могу очекивати онда када се дефинише и архитектонски смислен критеријум оптималности, то је проблем сам за себе и превазилази оквире овог рада.

Рад на датој теми је тек започет, где будући рад тежи пројекту зграде која би допринела повећању зелених површина у граду. Искуство је показало да зелени кров из финансијских разлога не представља реално и одрживо решење кад је у питању повећање зелених површина у градовима. С тим у вези, зелени кров одржавају само пословне зграде, јер им то инвестиционо није проблем, за разлику од стамбених зграда [7]. Истражују се нова решења. У даљем раду предлажу се бетонске жардињере унутар станова чији су распоред и димензије генерисане на исти начин као и стан у овом раду, односно помоћу модификованог генетског алгоритма. Развој даљег рада испланиран је кроз неколико етапа. Увођење непокретних објеката (бетонских жардињера) знатно усложњава тематику. Захтеви природе и они постављени од стране архитеката подразумевају постављање жардињера унутар стана са ограничењима. Главно ограничење је близина прозора ради доступности светлости.

У првој верзији претпоставка је да сви зидови уједно представљају и прозоре. У истој верзији уводи се ново ограничење, које подразумева да жардињере морају бити одмакнуте од зида и међусобно на дистанци, не сме да дође до преклапања непокретних објеката.

Циљ је омогућити слободан прилаз са свих страна жардињере. Решавање проблема са датим ограничењима у даљем раду се додатно може усложнити и проширити увођењем ограничења позиције прозора.

5. ЗАХВАЛНИЦА

Овим путем се захваљујем професору Милану Рапаићу на значајној подршци приликом израде и имплементације рада. Његови коментари, дискусије и помоћ током овог процеса су били од великог значаја.

Такође се захваљујем и професорки Јелени Атанацковић Јеличић на помоћи, нарочито приликом дефинисања самог проблема и разумевања његовог значаја у ширем контексту.

6. ЛИТЕРАТУРА

- [1] R. Mehta, D.W. Dahl, "Creativity: Past, present, and future.", *Consumer Psychology Review* 2, pp. 30-49, 2019.
- [2] S.K. Baduge, S. Thilakarathna, J.S. Perera, M. Arashpour, P. Sharafi, B. Teodosio, A. Shringi, P. Mendis, „Artificial intelligence and smart vision for building and construction 4.0: Machine and deep learning methods and applications“, „*Automation in Construction*“, p. 141, 2022.
- [3] S. Chaillou, "AI & architecture", 2019.
- [4] Nauata, Nelson, K.H. Chang, C.Y. Cheng, G. Mori, Y. Furukawa. "House-gan: Relational generative adversarial networks for graph-constrained house layout generation.", *Computer Vision—ECCV 2020: 16th European Conference, Proceedings, Part I 16 Springer International Publishing*, pp. 162-177, 2020.
- [5] A. Radford, L. Metz, S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks.", *arXiv preprint arXiv:1511.06434*, 2015.
- [6] W. Huang, H. Zheng, "Architectural drawings recognition and generation through machine learning.", In *Proceedings of the 38th annual conference of the association for computer aided design in architecture, Mexico City, Mexico*, pp. 18-20, 2018.
- [7] M. Shafique, K. Reeho, M. Rafiq, "Green roof benefits, opportunities and challenges—A review.", *Renewable and Sustainable Energy Reviews*, pp: 757-773, 2018.

Кратка биографија:



Милица Ињац рођена је 1999. године у Новом Саду. Завршила је основне академске студије на смеру Биомедицинско инжењерство на Факултету техничких наука.

ПРИМЕНА МОДЕЛА ВРЕМЕНСКИХ СЕРИЈА ЗА ПРЕДИКЦИЈУ СРЕДЊЕГ ГЛОБАЛНОГ НИВОА МОРА**APPLICATION OF TIME SERIES MODEL FOR GLOBAL MEAN SEA LEVEL PREDICTION**Стефан Арађанин, Јелена Сливка, *Факултет техничких наука, Нови Сад***Област – РАЧУНАРСТВО И АУТОМАТИКА**

Кратак садржај – Глобално загревање се односи на стално и дугорочно повећање просечне површинске температуре Земље. То је првенствено узроковано људским активностима као што су сагоревање фосилних горива, крчење шума и испуштање штетних гасова у атмосферу, попут угљен-диоксида и метана. Глобално загревање има дубоке ефекте на океане и мора широм света. Пораст нивоа мора резултира разним последицама на Земљи и захтева хитне мере за ублажавање. Овај рад истражује како различити фактори утичу на месечни пораст нивоа мора и његове временске флукуације. Разматрани су фактори попут температуре, стопе топљења глечера, густине мора, салинитета и нивоа угљен-диоксида. Први део рада фокусира се на прикупљању, data wrangling-у и истраживачкој анализи података (енг. *Explorative Data Analysis, EDA*) којима је заједнички атрибут временска одредница. Други део рада фокусира се на примену модела временских серија (енг. *Time Series*) коришћењем *XGB Regressor*-а, у циљу предвиђања тачне промене средњег глобалног нивоа мора. Користећи сређене и анализиране податке, модел може да укључи низ различитих фактора који утичу на ниво мора и да предвиди њихову промену на основу историјских образаца, са веома ниском стопом грешке.

Кључне речи: глобално загревање, средњи глобални ниво мора, машинско учење, временске серије

Abstract – Global warming refers to the ongoing and long-term increase in the Earth's average surface temperature. It is primarily caused by human activities such as burning fossil fuels, deforestation, and releasing harmful gases into the atmosphere, like carbon dioxide and methane. Global warming has profound effects on oceans and seas worldwide. The rising sea levels result in various consequences on Earth and require urgent mitigation measures. This paper investigates how various factors influence monthly sea level rise and its temporal fluctuations. These factors include temperature, glacier melting rates, sea density, salinity, and carbon dioxide levels. The first part of the paper focuses on data collection, data wrangling, and exploratory data analysis

(EDA), all centered around the common element - time. The second part of the paper concentrates on applying the Time Series model, specifically using the *XGB Regressor*, to predict precise global mean sea level changes. Using cleaned and analyzed data, the model can incorporate various factors affecting sea levels and forecast its changes based on historical patterns with a very low error rate.

Keywords: global warming, global mean sea level, machine learning, time series

1. УВОД

Глобално загревање представља трајни и непрекидни пораст површинске температуре Земље. Његово основно порекло лежи у људским активностима, укључујући сагоревање фосилних горива, крчење шума и испуштање значајних количина штетних гасова у атмосферу, као што су угљен-диоксид и метан. Повећање глобалних температура већ је довело до пораста нивоа мора и чешћих и озбиљнијих природних катастрофа. Штавише, глобално загревање оставља дубок траг на светске океане и мора. Како температуре расту, океанске струје се померају, а морска вода се загрева и закисељује због повећаног нивоа угљен-диоксида. Ове промене доводе до значајних поремећаја у морским екосистемима, узрокујући катастрофалне последице за бројне водене врсте.

Овај рад истражује утицај различитих фактора на месечне флукуације нивоа мора и њихове временске обрасце. Скуп података обухвата низ међусобно повезаних фактора који су од суштинске важности за глобално загревање, посебно за пораст нивоа мора. Ови чиниоци обухватају температуру, стопе топљења глечера, густину океана, салинитет, нивое угљен-диоксида и друге. Сви они су повезани заједничком временском ознаком, омогућавајући свеобухватно истраживање њихових међусобних односа и њиховог заједничког утицаја на глобално загревање.

Да би се постигао циљ прецизног предвиђања пораста нивоа мора, било је неопходно користити различите аналитичке и предиктивне методе користећи тренутно доступне податке. Сходно томе, извршено је прикупљање ограничених, али релевантних података из различитих извора који се тичу нивоа мора и фактора који су му допринели. Затим су примењене технике *data wrangling*-а за сређивање и организацију података. Детаљна анализа, користећи статистичке

НАПОМЕНА:

Овај рад произтекао је из мастер рада чији ментор је била др Јелена Сливка, ванр. проф.

методе, служи да би се идентификовали обрасци и трендови у подацима. На крају, спроведена је истраживачка анализа како би се истражиле потенцијално интересантне варијабле које доприносе променама нивоа мора.

Скуп података искоришћен је за решавање проблема временских серија који се односи на тачно превиђање средњег глобалног нивоа мора (енг. *Global Mean Sea Level*, *GMSL*). Упркос малом обиму података, примена анализе временских серија се показала као веома ефикасан алат за прецизно предвиђање. Користећи добро структурисане и анализиране податке, модел укључује различите факторе који утичу на пораст нивоа мора и предвиђа његове будуће промене на основу историјских образаца, са изузетно ниском маргином грешке. На пример, модел показује изузетну тачност у предвиђању пада нивоа мора, постижући апсолутну грешку од 0.03 и грешку предвиђања од 0.5 за *GMSL* између августа и септембра 2013. Вреди напоменути да модел лакше предвиђа опадање у односу на пораст нивоа мора.

У наредном поглављу ће бити представљена већ постојећа, слична решења овом раду. У поглављу 3, описана је имплементација система, док је у поглављу 4 приказана дискусија о резултатима. На крају, у последњем поглављу, представљен је закључак рада.

2. ПРЕТХОДНА РЕШЕЊА

У раду [1], вршена је реконструкција глобалног нивоа мора користећи по податке добијених мерењем плиме и осеке у периоду од 1870. до 2004. године. Резултати до којих су научници дошли указује на пораст нивоа мора и то од 195 мм и стопу пораста у 20. веку од $1,7 \pm 0,3$ мм годишње. Главни недостаци овог рада су да се рад ослања искључиво на податке о плими и осеки који лако могу бити подложни променама, као и то да нису покушали да открију допринос глечера и ледених површина на регионални образац пораста нивоа мора, што признају и сами аутори. Такође, иако је познато убрзање *GMSL*-а, оно се тешко може открити због малог броја података.

У раду [2], анализиран је раст глобалне температуре на Земљи користећи неколико главних алгоритама машинског учења над прикупљеним подацима. Модели су креирани анализом односа између температуре и фактора као што су концентрације угљен-диоксида, азот-оксида и метана. Искоришћена је линеарна интерполација како би се попуниле недостајуће вредности. Коришћена је осмострука унакрсна валидација за тражење одговарајућих хипер-параметара модела и упоређене су перформансе четири различита алгорита машинског учења: *Linear Regression*, *Lasso*, *SVR* и *Random Forest*. Као најбољи алгоритам се показао *Random Forest* и издвојио угљен-диоксид, као главни фактор који доприноси повећању температуре, а потом га прате метан и азот-оксид.

У раду [3], извршена је анализа термичке експанзије у регионима отвореног океана која доводи до промена у приобалним нивоима мора, а како би се одредила тенденција повећања/смањења нивоа мора. Коришћен је модел регресије Гаусовог процеса и рекурентна

неуронска мрежа. Скуп података датира између 1993. и 2018. године. Утврђено је да промене температуре низ водени стуб до 700 м дубине утичу на промене варијабилности регионалног обалног нивоа мора. У раду је показано да велике температурне флукуације утичу на промену нивоа мора за већину региона, ипак представљене су и мане, а то је да би се додатно могла укључити анализа кретања копна, промена опсега плиме и промена леда.

У раду [4], приказана је улога модела временске серије у процесу предвиђања уз анализу података временских серија користећи записе о месечном средњем нивоу мора од јануара 1978. до октобра 2020. године на Великом Острву у Луизијани. Коришћен је *SARIMA* модел за предвиђање будућих тачака у временској серији. Модел је показао линеарни узлазни тренд повећања нивоа мора уз константну стопу раста у краткорочном периоду. Такође, у раду је коришћена и вишеслојна перцептрон неуронска мрежа. Резултати рада указују да коришћење модела временских серија могу пружити одличан приступ за предикцију средњег глобалног нивоа мора.

3. МЕТОД

У наредним поглављима су описани коришћени скупови података, примењене технике њихове обраде ради креирања коначног скупа, као и начин креирања предиктивног модела временске серије.

3.1. Прикупљање и обрада података

Прикупљање података подразумевало је прегледање јавно доступних скупова података, прикупљаних из различитих извора, а који садрже употребљива обележја и податке о факторима који утичу на пораст нивоа мора. Издвојено је пет скупова сирових података, који су потом засебно процесирани у даљим корацима.

У скупу података који садржи информације о укупној површини морског леда [6], прикупљани су подаци у периоду од 26.10.1978. до 31.5.2019. године. Иницијалном анализом података, установљено је да су мерења конзистентна само на месечном нивоу и да скуп података садржи 26 354 редова. Обележја која чине овај скуп су: датум мерења, укупна површина морског леда, недостајуће вредности мерења, извор мерења и северна и јужна Земљина сфера. Укупна површина морског леда и датум мерења су обележја од значаја.

Наредни обрађивани скуп података садржи информације о узорцима воде сакупљане из океана између 1959. и 2020. године [5]. Иницијални скуп података садржи 337 792 реда и доста дуплицираних вредности. Емпијским путем је закључено да би обележја, чија је стандардна девијација мања од 3, требало уклонити из скупа. Неки од атрибута скупа су: датум прикупљања, дубина океана на ком је извршено мерење, концентрације кисеоника, нитрата, нитрита и силиката. Током анализе, утврђено је да велике дубине немају значај када је у питању регионални пораст нивоа мора и, по угледу на овај закључак, су узети у обзир само подаци прикупљени на дубинама до 200 метара.

У трећем скупу података [7], забележене су температуре од чак 1750. до 2015. Године. Приликом прегледања скупа, закључено је да су вредности датума мерења, просечних температура копна, као и температуре копна и океана једине од интереса у овом раду. Очишћени скуп података садржи 3 192 реда.

У четвртном раду [8], поред датума у ком је извршено мерење, налази и податак о просечној молској фракцији угљен-диоксида одређеног из дневног просека. Подаци су прикупљени од 1958. до 2018. године.

Последњи, најбитнији, скуп података [9] садржи промену нивоа мора мерену у периоду између 1880. до 2013. године и оно представља циљно обележје.

3.2. Data wrangling

Овај процес укључује технике манипулације над прикупљеним подацима, како би скуп података био што чистији и сачињен од међусобно усклађених података спремних за моделовање временских серија.

Анализом свих временских обележја, емпиријским путем је одабрано да ће се прикупљени скупови података агрегирати на основу месеца у ком је забележено мерење и да ће највећа искоришћеност прикупљених података бити уколико се одабере временска одредница од 1969. до 2013. године.

У скупу података [6] извршено је груписање података путем израчунавања средње вредности како би подаци били употребљени на основу месеца мерења. Обзиром да скуп података садржи податке од 1978. године, емпиријским путем, испробавањем неколико техника додавања недостајућих вредности, одлучено је да оне буду попуњене коришћењем *Curve Fitting* математичке функције.

У скупу података са узорцима воде [5], установљено је да постоји 3080 различитих вредности за датум, од укупно 236 431 редова. Извршено је груписање података коришћењем *median* методе и тиме су дупликати уклоњени, а подаци о температури воде, концентрацији кисеоника, силиката и нитрата су задржани за одређени дан. Због великог броја недостајућих вредности, извршена је интерполација. Емпиријским путем и испробавањем неколико типова интерполације, дошло се до закључка да полиноминална функција даје најбоље резултате.

Остали скупови података, осим уклањања датума који су ван одабраног опсега, нису захтевали детаљније трансформације.

3.3. Истраживачка анализа података

Истраживачка анализа је искоришћена за откривање образаца понашања података у сваком од испитаних скупова, уз визуелни приказ корисних обележја и њихових вредности.

Употребом неколико библиотека *python* програмског језика, вршена је провера одступајућих вредности у скупу података о површини морског леда [6]. С обзиром на то да график није приказао одступнике, долази се до закључка да скуп података садржи конзистентну дистрибуцију, без присуства екстремних вредности. Анализом скупа података о водама [5], идентификоване су тачке које значајно одступају од

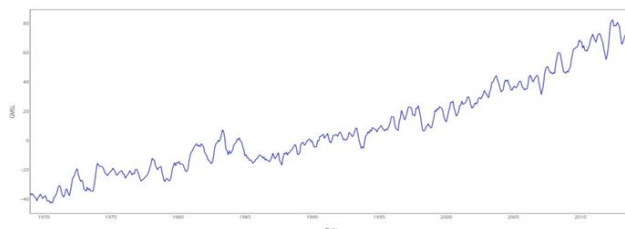
очекиване дистрибуције и ове вредности су изостављене из скупа да би се смањио лошији утицај на тачност предиктивног модела временских серија.

У свим осталим скуповима података, поновљени су исти кораци за детекцију аномалија и одступајућих вредности међу подацима и установљено је да нема екстремних вредности.

Након извршених свих корака претпроцесирања, скупови података су спремни за заједничку агрегацију на основу датума мерења. Резултујући скуп података садржи следеће атрибуте:

- *Date* – датум мерења изражен у формату година - месец,
- *Extent* - укупна површина морског леда изражена у јединици 10^6 km^2 ,
- *WaterTemp* – температура воде изражена у целзијусима,
- *O2ml* – засићеност воде кисеоником,
- *SiO3* – концентрација силиката,
- *NO3* – концентрација нитрата,
- *LandAverageTemperature* – глобална просечна температура копна изражена у целзијусима,
- *LandAndOceanAverageTemperature* - глобална просечна температура копна и океана изражена у целзијусима,
- *CO2* - просечна месечна молска фракција CO_2 и
- *GMSL* – средњи глобални ниво мора.

На слици 1, дат је графички приказ временске серије који приказује еволуцију средњег глобалног нивоа мора током времена. Уочен је тренд инкременталног пораста нивоа мора током посматраног периода који указује кретање навише, са благим сезонским осцилацијама.



Слика 1. Графички приказ временских серија *GMSL-a*

3.4. Архитектура решења

За предикцију средњег глобалног нивоа мора, коришћен је модел временских серија. *GMSL* означава циљно обележје коју модел настоји да предвиди, док се сва остала обележја користе као предиктивне карактеристике за помоћ при обучавању.

Улаз у модел додатно чине хронолошке тачке посматрања. Модел је у стању да предвиђа будуће вредности на основу претходних зависних корака. У овом раду, на основу *K* корака, предвиђа се (*K*+1)-ви корак.

ForecasterAutoreg је употребљен за предикцију модела, при чему је за регресор одабран *XGB Regressor* алгоритам. Приликом обучавања, узете вредности за *XGB Regressor* су биле [50, 100, 500] за *n_estimators*, [3, 5, 6, 10] за *max_depth* и [0.01, 0.1] за *learning_rate*. За временске кораке кашњења, модел временске серије је приликом обучавања издвојио да

низ корака [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] даје најбоље резултате.

4. РЕЗУЛТАТИ И ДИСКУСИЈА

Резултујући претпроцесирани скуп података подељен је на три подскупа: обучавајући, валидациони и тест скуп. За метрику процене учинка предвиђања средњег глобалног нивоа мора коришћена је средња апсолутна грешка (енг. *Mean Square Error*) података. Поред тога, коришћено је и тестирање уназад. У склопу тестирања уназад, урађена симулација предвиђања се одвија за сваки претходни месец са кораком један, уз процену предиктивних способности модела укључујући и тест скуп података.

Најбоље перформансе модела су показале да је вредност метрике средње апсолутне вредности грешке 70.8. Метрика тачности је приликом тестирања уназад имала вредност 56.3. Ова вредност представља средњу квадратну грешку предвиђања модела.

Анализирањем добијених резултата може се закључити да модел греши при предвиђању већих растућих вредности средњег глобалног нивоа мора. При мањим и опадајућим вредностима *GMSL*-а, модел има изузетно добре перформансе. У табели 1, приказана су три најбоља предвиђана месеца.

Датум	<i>GMSL</i>	Предвиђени <i>GMSL</i>	Апсолутна грешка
2013-08	71.6	71.57	0.03
2013-06	71.5	72	0.5
2013-07	71.2	72.4	1.2

Табела 1. Вредности најбољих перформанси модела

Модел временских серија је као факторе који највише утичу на предиктивне способности издвојио: корак временског кашњења од 1 до 10, дебљину леда, температуру воде, ниво кисеоника, ниво силиката, ниво нитрата, температуру копна и копна и океана и на крају количину угљен-диоксида. Извођењем закључка из свих сличних радова и ове студије, може се доћи до претпоставке да је таком времена пораст температуре имао утицај на дебљину леда, а који је потом имао утицај на повећање средњег глобалног нивоа мора. Управо су време, дебљина леда и температура воде, фактори који највише доприносе крајњим резултатима излаза из модела.

5. ЗАКЉУЧАК

У овом раду представљен је систем за предвиђање будућег пораста средњег глобалног нивоа мора, уз испитивање различитих фактора који на њега утичу. Мотивација за овај рад произилази из дубоког утицаја глобалног загревања на океане и мора широм света.

На почетку је циљ био прикупити што већи број скупова података из различитих извора. Потом је извршена обрада података с обзиром на то да је постојало доста неупотребљивих вредности и мерења која не би могла да се уклопе са осталим подацима из преосталих скупова података.

Наредни корак претпроцесирања података био је *data wrangling* током ког су подаци трансформисани на начин на који би се могли агрегирати по месецу мерења и додатно су уклоњени шумови и попуњене недостајуће вредности. Истраживачка анализа података помогла је да се отклоне одступници. За предвиђање будућих промена *GMSL*-а, коришћен је модел временске серије, при чему је за регресор одабран *XGB Regressor* алгоритам.

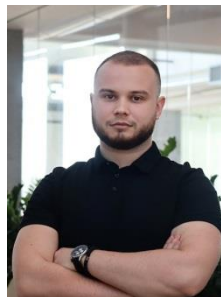
Средња апсолутна грешка приказује да модел греши са вредношћу 56.3. Најбоље предвиђани датум мерења јесте август 2013. године и за њега модел греши са вредношћу од само 0.03.

За даља унапређења, могу се истражити перформансе модела обученог над засебним скуповима података. Могуће је и поново обучити модел само са атрибутима за које се показало да највише утичу на његове предиктивне способности. Осим коришћења *XGB* регресора, требало би испитати и перформансе других алгоритама.

6. ЛИТЕРАТУРА

- [1] John A. Church, Neil J. White, 2006: A 20th century acceleration in global sea-level rise.
- [2] Harvey Zheng, vol. 7, no. 3 2018: Analysis of Global Warming Using Machine Learning.
- [3] Veronica Nieves, Christina Radin, Gustau Camps-Valls, 2021: Predicting regional coastal sea level changes with machine learning
- [4] Yeong Nain Chi, 2022: Time Series Modeling and Forecasting of Monthly Mean Sea Level (1978 – 2020): SARIMA and Multilayer Perceptron Neural Network
- [5] <https://www.kaggle.com/datasets/mathsian/water-temperature>
- [6] <https://www.kaggle.com/datasets/nsidcorg/daily-sea-ice-extent-data>
- [7] <https://www.kaggle.com/datasets/berkeleyearth/climate-change-earth-surface-temperature-data?select=GlobalTemperatures.csv>
- [8] <https://datahub.io/core/co2-ppm>
- [9] <https://datahub.io/core/sea-level-rise>

Кратка биографија:



Стефан Арађанин рођен је 1998. године у Кикинди. Основне академске студије завршио је 2021. године на Факултету техничких наука у Новом Саду, на ком брани и мастер рад 2023. године из области Електротехнике и рачунарства.
контакт: aradjanin.stefan07@gmail.com

ПЕРЗИСТЕНЦИЈА NEW WAVE АЛАТА ЗА УПРАВЉАЊЕ РАДНИМ ТОКОВИМА У РЕЛАЦИОНУ БАЗУ ПОДАТАКА**THE PERSISTENCE OF THE NEW WAVE WORKFLOW ENGINE IN A RELATIONAL DATABASE**

Матија Мијалковић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *Тема овог рада јесте перзистенција NewWave алата за управљање пословним процесима у релациону базу података. С обзиром да је у изрази NewWave система коришћен Pharo програмски језик, перзистенција се имплементира уз помоћ Pharo објектно-релационог мапера под називом GLORP (Generic Lightweight Object Relational Persistence), у комбинацији са PostgreSQL базом података. Имплементација самог решења може се поделити на две целине. Прва се односи на перзистенцију модела пословних процеса, док се друга фокусира на перзистенцију појединачних инстанци извршавања пословних процеса.*

Кључне речи: *NewWave, Pharo, пословни процеси, релационе базе података, Glorp*

Abstract – *The topic of this paper is the persistence of the NewWave tool for managing business processes into a relational database. Given that the Pharo programming language was used to implement the NewWave system, persistence is implemented with the help of the Pharo object-relational mapper called GLORP (Generic Lightweight Object Relational Persistence), in combination with the PostgreSQL database. The implementation of the solution itself can be divided into two parts. The first refers to the persistence of business process models, while the second focuses on the persistence of individual instances of business process execution.*

Keywords: *NewWave, Pharo, business processes, relational database, Glorp*

1. УВОД

Имајући у виду да су производи и услуге које се нуде на тржишту најчешће резултат обављања одређених пословних процеса, као и то да се све већи број поменутих процеса реализује кроз информационе системе, не чуди да је концепт управљања пословним процесима у фокусу не само стручњака из области менаџмента, већ и оних из области информационих технологија.

Бројне предности, као што су повећана ефикасност и продуктивност, смањење трошкова и унапређење квалитета услуга, главна су мотивација за изучавање и примену ове дисциплине у пословним окружењима.

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији ментор је био др Мирослав Зарић ванр. проф.

Системи за управљање пословним процесима су софтверски алати чија је основна улога пружање платформе за моделовање, аутоматизацију, извршавање и надгледање процеса. Један од кључних фактора за успешно функционисање ових система, као и код већине модерних апликација је перзистенција података. Под перзистенцијом се подразумева дуготрајно чување података, тако да су они доступни и након заустављања апликације у којој су креирани. Да би то било могуће, користе се трајна меморијска складишта (*non-volatile*) и постоји више приступа како се чување у поменутих складиштима имплементира. Један од најпопуларнијих приступа за чување података јесте коришћење релационих (*SQL*) база података. Тема овог рада јесте управо имплементација перзистенције података у релациону базу за већ постојећи систем за управљање пословним процесима под називом *NewWave*.

2. ПОСЛОВНИ ПРОЦЕСИ, РЕЛАЦИОНЕ БАЗЕ ПОДАТАКА, PHARO ПРОГРАМСКИ ЈЕЗИК

Крајем прошлог века дошло је до промене фокуса приликом развоја пословних информационих система са података на пословне процесе, што је за последицу имало настанак процесно оријентисаних/свесних информационих система (*process aware information systems*)[1].

Alter [2] дефинише информациони систем као посебан тип радног система који користи информационе технологије како би прихватио, преносио, чувао, претраживао, манипулисао и приказивао информације, и на тај начин пружао подршку једном или више радних система.

Уколико је информациони систем свестан постојања процеса и самим тим повезује информациону технологију са пословним процесима, онда се тај систем сматра процесно оријентисаним/свесним. Процес се у овом контексту дефинише као начин да нека организација организује посао и ресурсе (људе, опрему, информације) тако да постигне одређене пословне циљеве [3].

Weske [4] дефинише пословни процес као скуп активности које се координисано извршавају у одговарајућем организационом и технолошком окружењу. Извршавањем ових активности постиже се пословни циљ. Сваки пословни процес извршава једна организација, али он може интераговати са пословним процесима других организација.

Управљање пословним процесима не односи се само на репрезентацију пословног процеса већ обухвата и концепте, методе и технике за подршку дизајну, администрацији, конфигурисању, извршавању и анализи пословних процеса [4].

Осим традиционалног, мануелног управљања пословним процесима од стране запослених неке компаније, извршавањем пословних процеса може се управљати и софтверски кроз системе за управљање пословним процесима. Ови системи се дефинишу као генерички софтверски системи који, вођени експлицитном репрезентацијом пословног процеса, обезбеђују његово координисано извршавање [4].

На основу претходних дефиниција може се закључити да је за управљање пословним процесима најважнија експлицитна дефиниција модела пословног процеса којом се обухватају како саме активности процеса, тако и ограничења приликом њиховог извршавања.

Дистинкцију између модела пословног процеса и инстанце пословног процеса *Weske* дефинише на следећи начин: Модел пословног процеса састоји се од скупа модела активности и ограничења за њихово извршавање, док инстанца пословног процеса представља један конкретан случај извршавања дефинисаног модела у пословном окружењу компаније. Инстанца пословног процеса састоји се од инстанци појединачних активности [4].

Радни токови (*Workflow*) су посебна врста пословних процеса у којима постоји интензивна интеракција између корисника и апликације. Коалиција за управљање радним токовима (*Workflow Management Coalition – WfMC*) дефинише радни ток (радни процес) (*Workflow*) као аутоматизацију пословног процеса, делимично или у целости, коришћењем рачунара [5].

Коначно, систем за управљање радним токовима је софтверски систем који у целости дефинише, управља и извршава радне токове коришћењем одговарајућег софтвера чији је редослед извршавање вођен репрезентацијом пословне логике радног тока [5].

База података је организована колекција структурираних података који се складиште електронски. За приступ и управљање овим подацима користе се системи за управљање базама података (*Database management system – DBMS*). Ови системи представљају посредничку компоненту између базе података и крајњих корисника или програма и омогућавају добављање, измену и управљање подацима, то јест њиховом организацијом и оптимизацијом [6].

Најчешће коришћени тип базе података јесте релациона база података, заснована на релационом моделу података. Релациони модел података први је описао британски информатичар *Edgar F. Codd* у раду „*A Relational Model of Data for Large Shared Data Banks*“ [9]. Идеја овог модела јесте та да се подаци организују у форми међусобно повезаних табела, сачињених од редова и колона [7]. Табелама су представљени ентитети или везе између ентитета, редови представљају појаве (инстанце) ентитета или веза, а колонама су представљени њихови атрибути.

Овде треба напоменути да су табеле логичке структуре и да не осликавају физичку организацију података [8].

Најпознатији примери система за управљање релационим базама података (*Relational Database Management System – RDBMS*) су [7]: *Oracle*, *MySQL*, *PostgreSQL*, *MariaDB* итд. У овом раду коришћен је *PostgreSQL RDBMS*.

PostgreSQL [10] је моћан систем за управљање базама података, отвореног кода, који користи и проширује *SQL* језик и пружа широк спектар опција за безбедно складиштење података и за скалирање компликованих обрада података.

Pharo је чист објектно-оријентисани програмски језик отвореног кода, са динамичком доделом типова који подржава интерактиван приступ програмирању и заснован је на *Smalltalk-80* језику. Настао је 2008. године када се одвојио као посебна грана *Squeak* пројекта (модерне имплементације *Smalltalk-80* програмског језика) [11]. Четири основне компоненте *Pharo* система су: виртуелна машина, слика, фајл са изменама и фајл са изворним кодом.

С обзиром да је *Pharo* окружење објектно, то јест да су све инстале (појаве) доменских ентитета у форми објеката, а да су са друге стране инстанце (појаве) ентитета у релационим базама у форми редова (торки) у табелама, неопходно је да се пре перзистенције *Pharo* објеката у базу података изврши објектно-релационо мапирање (*Object – Relational Mapping – ORM*).

Glorp (*Generic Lightweight Object Relational Persistence*) је *Pharo* објектно-релациони мапер који подржава широк спектар могућности (управљање трансакцијама, формирање упита ка бази без коришћења *SQL*, писање једноставних и комплексних упита коришћењем *Pharo* синтаксе итд.) којима се поменути проблеми приликом мапирања мање или више редукују [12].

3. NEW WAVE

NewWave је алат отвореног кода за извршавање радних токова имплементиран у *Pharo* програмском језику. Циљ овог пројекта јесте креирање проширеног система за управљање радним токовима који омогућава лако дефинисање и извршавање пословне логике уз тренутно пружање повратних информација и једноставну интеграцију са већ постојећим решењима [13].

NewWave је и даље у фази развоја и тренутно од образаца за контролу тока подржава секвенцијално извршавање, ексклузивно и инклузивно гранање, паралелно гранање и спајање, као и различите типове задатака и догађаја. Структура пројекта је модулarna што значи да се могу користити подскупови доступних функционалности у складу са тренутним потребама. Осим у форми угњеженог система за извршавање радних токова, *NewWave* се може покренути и у форми засебног, самосталног сервиса којем се приступа екстерно [14].

NewWave систем подржава следеће типове радних токова [14]:

- Интерактивни радни токови у којима се радни задаци извршавају од стране једног или више корисника којима су додељени
- Делимично или потпуно аутоматизовани радни токови у којима се извршавање додељује некој софтверској компоненти или сервису

Сама имплементација *NewWave* апликације у различитим аспектима ослања се на екстерне пројекте, што не чуди с обзиром да је један од главних фокуса овог пројекта адаптивност и интеграција са већ постојећим библиотекама и алатима из *Pharo* екосистема.

4. ИМПЛЕМЕНТАЦИЈА И РЕЗУЛТАТИ

NewWave пројекат је подељен на више пакета којима су обухваћене класе које припадају истим функционалним целинама. Како је идеја овог рада перзистенција модела пословних процеса и њихових покренутих инстанци на почетку треба одредити које од постојећих *NewWave* класа се заправо користе приликом моделовања и инстанцирања процеса, а које, са друге стране, представљају додатне сервисе и функционалности у систему. Прва фаза имплементације односи се на перзистенцију класа које представљају градивне елементе модела пословних процеса, али и класа које описују пословну (корисничку) организацију. У другој, уједно и финалној, фази имплементације, перзистира се подкуп класа којима се описују покренуте инстанце процеса.

Опис објектно-релационог мапирања за све класе од интереса реализује се кроз систем дескриптора који се имплементира као поткласа *DescriptorSystem* класе. Овај систем садржи скуп метода на основу којих се генеришу *Glorp* метаподаци потребни за перзистенцију доменских објеката. Методе система дескриптора деле се у три групе.

Прву групу чине методе које описују моделе класа (*class models*) кроз опис њихових атрибута и именују се по обрасцу *classModelForSomeClass*.

Друга група обухвата методе којима се описује структура релационе базе кроз опис њених табела (*table models*). Методе из ове групе именују се по обрасцу *tableForTABLENAME*. Трећа, уједно и последња група садржи методе које описују мапирање између класа и табела релационе базе (*descriptors*) и добијају називе пратећи форму *descriptorForSomeClass*.

Класе свих градивних елемената за креирање модела пословних процеса у *NewWave* пројекту наслеђују основну класу *NWElement*, а најзначајније међу њима су *NWNode* (основни градивни елемент пословног процеса, чвор), *NWSequence* (веза између два чвора), *NWDataObject* (доменски објекат произвољне структуре) и *NWProcess* (сам модел пословног процеса). Описивањем њихове перзистенције као и перзистенције њихових поткласа (активности, догађаји, гранања), али и класа које описују корисничку организацију (корисник, група, припадност групи) реализује се део имплементације везан за перзистенцију градивних елемената модела пословних процеса. Комплетирањем ове етапе,

омогућено је да се произвољни модели пословних процеса, након креирања, сачувају у базу података и по потреби из ње читају како би се извршавали или даље модификовали. Међутим, поред перзистенције модела пословних процеса, потребно је перзистирати и појединачне инстанце ових процеса које настају њиховим покретањем. Чувањем покренутих инстанци и њиховог тренутног стања, може се, након читања из базе, наставити извршавање пословног процеса са истог места где је исти био заустављен пре складиштења у базу података.

Извршавањем пословних процеса, у *NewWave* пројекту, руководи посебна група класа под називом *NewWave-Engine*. Понашање свих активности и догађаја дефинише се кроз одговарајуће класе које припадају овој групи, а осим класа за опис понашања, групом *NewWave-Engine* обухваћене су и класе које имплементирају инфраструктуру за покретање и извршавање пословних процеса. Најважнија међу њима, *WaveEngine*, представља сервис за извршавање радних токова и у себи садржи листу инстанци процеса које чекају на извршавање или се већ извршавају. Инстанца процеса, представљена класом *NWProcessHandler*, садржи једну главну јединицу за извршавање радних токова (*NWExecutor*), а може садржати и додатне подизвршиоце (*NWSubExecutor*) у зависности од самог модела пословног процеса. Током извршавања пословног процеса управља класа *NWFlowHandler*, док је инстанца извршавања чвора из модела пословног процеса представљена класом *NWActivation*.

WaveEngine сервис и класе које описују понашања елемената пословних процеса, немају стање (*stateless*) па се сходно томе њихови објекти (инстанце) не перзистирају. За остале наведене класе, међутим, треба описати објектно-релационо мапирање кроз одговарајуће методе у систему дескриптора. На тај начин се комплетира и друга, финална фаза имплементације којом се омогућује перзистенција покренутих инстанци пословних процеса.

За демонстрацију имплементираних решења, коришћени су примери пословних процеса који најбоље осликавају тренутне могућности.

Перзистенција модела пословног процеса представљена је на примерима аутоматизованих процеса који комбинују скриптне задатке (*NWScriptTask*) са ексклузивним (*XOR*) и паралелним (*AND*) гранањем.

С друге стране, перзистенција инстанце пословног процеса демонстрирана је на примеру процеса који поред аутоматских садржи и корисничке задатке, а чија основна идеја јесте имплементација поједностављене пословне логике поступка запошљавања. Присуство корисничких задатака, пружа могућност анализе стања покренуте инстанце пре сваког следећег корисничког задатка у радном току, јер се у том тренутку извршавање процеса привремено зауставља и за наставак је потребна корисничка интервенција. Ажурирањем стања инстанце процеса, у бази података, током описаних прекида, обезбеђује се несметан наставак извршавања процеса (из последњег перзистираних стања) чак и у

случају да тренутни сервис за извршавање радних токова престане да постоји. Успешним наставком претходно заустављене инстанце процеса, унутар новог сервиса за извршавање радних токова (*Wave-Engine*), потврђена је функционалност имплементiranог решења и у погледу перзистенције појединачних покренутих инстанци модела пословних процеса.

5. ЗАКЉУЧАК

На почетку овог рада представљени су теоријски концепти на које се ослања имплементација, након чега је уследио опис модела и архитектуре *NewWave* система. У наредним поглављима детаљно је описан ток имплементације читавог решења, као и приказ резултата на примеру неколико одабраних пословних процеса.

На самом крају треба предложити и смернице за даље унапређење тренутног решења. Најзначајније унапређење може се остварити на пољу аутоматизације процеса перзистенције. Наиме, читав опис перзистенције *NewWave* система имплементиран је мануелно креирањем појединачних описа за сваку класу која се перзистира. У случају било какве измене неке од ових класа, неопходна је и промена описа њене перзистенције што може бити заморно и подложно грешкама, или чак потпуно занемарено.

Једно од могућих решења јесте генерисање *Glorp* мета-описа перзистенције на основу мета-описа доменских класа, генерисаног коришћењем неког од доступних алата *Pharo* екосистема (нпр. *Magritte* [15]). На овај начин ће се обезбедити конзистенција између тренутног стања доменских класа и описа њиховог објектно-релационог мапирања, а читав поступак перзистенције ће бити аутоматизован и упрошћен.

6. ЛИТЕРАТУРА

- [1] M. Dummas, W.M.P. van der Aalst, A.H.M. ter Hofstede, “*Process-Aware Information Systems*”, John Wiley & Sons, Inc., Hoboken, New Jersey, 2005.
- [2] S. Alter, “*Defining information systems as work systems: implications for the IS field*”, *European Journal of Information Systems* 17, 448-469, 2008. <https://doi.org/10.1057/ejis.2008.37> (приступљено у августу 2023.)

- [3] A. Sharp, P. McDermott, “*Workflow Modeling: Tools for Process Improvement and Application Development, Second Edition*”, Artech, 2008.
- [4] M. Weske, “*Business Process Management – Concepts, Languages, Architectures*”, Springer Science & Business Media, Berlin, 2007.
- [5] D. Hollingsworth, “*The Workflow Reference Model*”, Workflow Management Coalition, 1995.
- [6] <https://www.oracle.com/database/what-is-database/> (приступљено у августу 2023.)
- [7] <https://cloud.google.com/learn/what-is-a-relational-database> (приступљено у августу 2023.)
- [8] <https://www.oracle.com/database/what-is-a-relational-database/> (приступљено у августу 2023.)
- [9] E.F. Codd, “*A Relational Model of Data for Large Shared Data Banks*”, *Communications of the ACM, Classics*, 13(6), 377–87, 1970. doi: <https://doi.org/10.1145/362384.362685> (приступљено у септембру 2023.)
- [10] <https://www.postgresql.org/about/> (приступљено у августу 2023.)
- [11] A.P. Black, S. Ducasse, O. Nierstrasz, D. Pollet, D. Cassou, M. Denker, Open Textbook Library, “*Pharo by example*”, Square Bracket Associates, [Kehrsatz] Switzerland, 2009. <https://open.umn.edu/opentextbooks/textbooks/314> (приступљено у августу 2023.)
- [12] E. Maringolo, N. Pratt, R. Withney, “*Object-Relational Persistence With Glorp*”, 2018.
- [13] S. Kaplar, M. Zarić, G. Milosavljević, “*NewWave Workflow Engine*”, International Workshop on Smalltalk Technologies, Koln, Germany, 2019.
- [14] S. Kaplar, M. Zarić, S. Ducasse, “*NewWave: Workflow engine*”, *Science of Computer Programming*, Volume 203, 2020.
- [15] <https://github.com/magritte-metamodel/magritte.git> (приступљено у августу 2023.)

Kratka biografija:



Матија Мијалковић рођен је у Пироту 1999. г. Студијски програм Рачунарство и аутоматика на Факултету техничких наука, уписао је 2017. г. Дипломирао је 2021. г., а затим уписао мастер академске студије из исте области.
контакт: matija.mijalkovic@gmail.com

МИГРАЦИЈА МИКРОСЕРВИСНЕ АПЛИКАЦИЈЕ СА РЕЛАЦИОНОГ МОДЕЛА ПОДАТАКА У НЕРЕЛАЦИОНИ МОДЕЛ КОРИШЋЕЊЕМ *SERVERLESS* AWS СЕРВИСА

MIGRATION FROM MICROSERVICES APPLICATION ARCHITECTURE WITH RELATIONAL DATABASES TO AWS SERVERLESS BASED SOLUTION WITH NON-RELATIONAL DATABASE

Сенка Шоић, Факултет техничких наука, Нови Сад

Област – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – У раду је приказана конверзија веб апликације са микросервисном архитектуром и релационим моделом података у апликацију нерелационог модела података која се ослања на AWS сервисе.

Анализирана је микросервисна архитектура наспрам *serverless* архитектуре. Описани су коришћени AWS сервиси. Анализирани су и описани кораци миграције у *serverless* систем.

Кључне речи: Микросервиси, рачунарство у облаку, *Serverless*, AWS, *DynamoDB*, *AppSync*, *Lambda* функције

Abstract – This paper presents conversion of web application with microservices architecture and relational data model into non-relational data model application which is relying on AWS services. It analyses microservices architecture compared to *serverless* architecture. It describes used AWS services. It analyses and describes migration steps into *serverless* system.

Keywords: *Microservices*, *Cloud computing*, *Serverless*, AWS, *DynamoDB*, *AppSync*, *Lambda functions*.

1. УВОД

Микросервисна архитектура је чест избор за архитектуру система приликом развоја апликација. Разлог за то је то што се индивидуални микросервиси могу независно модификовати, испоручивати и скалирати. Као резултат, у микросервисној архитектури, развој апликације и управљање инфраструктуром су уско повезани.

Ипак, са повећаном популарношћу *serverless* парадигме и њених предности, попут изостајања потребе управљања инфраструктуром, као и модела плаћања у коме се наплаћују само искоришћени ресурси, долази до раста интереса за рефакторисањем апликација базираним на микросервисима и миграције у *serverless* архитектуру. Један од најпознатијих *cloud* провајдера – AWS нуди бројне сервисе који се могу искористити за изградњу *serverless* апликација.

НАПОМЕНА:

Овај рад произтекао је из мастер рада чији ментор је био др Мирослав Зарић, ванр. проф.

У склопу овог рада миграција је описана на примеру једног система који поседује микросервисну архитектуру. У питању је систем за изнајмљивање аутомобила (*Rent-a-Car*), који се састоји из неколико микросервиса, корисничке (тзв. фронтенд) апликације и релационог модела података који је изграђен у *MySQL* бази података.

Као резултат овог рада добијено је *serverless* решење, чији је модел података изграђен у нерелационој *NoSql* бази података (*DynamoDB*), коришћењем *AppSync*, *Cognito* и *Lambda* сервиса.

2. ТЕОРИЈСКЕ ОСНОВЕ

2.1. Микросервисна архитектура

Микросервисна архитектура је стил архитектуре софтверских система у ком је апликација подељена на мале компоненте које су самосталне, лабаво повезане независно испоручиве и скалабилне. Те мале компоненте се називају микросервиси и типично имају следеће карактеристике [1]:

- Имају своју комбинацију технологија (енгл. *tech stack*), библиотека и зависности
- Обично имају своју базу података
- Програмери могу да раде на њима одвојено, без утицаја на рад других тимова
- Међусобно најчешће комуницирају преко *REST API*-ја, стриминга догађаја или посредника у преносу порука (енгл. *message broker*)

Микросервиси су настали из потребе да се олакша развој и скалирање апликација када су оне постајале огромне и претешке за сналажење у бази кода, одржавање и руковање [1].

2.2 Рачунарство у облаку

Концепт рачунарства у облаку (енгл. *Cloud computing*) може се објаснити као технологија која омогућава коришћење различитих информатичких услуга на физички удаљеним серверима уз помоћ наше мрежне инфраструктуре и одговарајућих интернет протокола. *Cloud* технологија се заснива на томе да сви подаци који су неопходни кориснику (биле то апликације, документи, хардвер или нешто друго) буду доступни

у сваком тренутку, те *cloud* на неки начин представља испоруку услуга уместо самог производа [2].

Постоје три основна типа рачунарства у облаку [3]:

1. Инфраструктура као сервис - Типично омогућава приступ и управљање: мрежним ресурсима, рачунарима (виртуелним или на наменском хардверу) и месту за складиштење података.

2. Платформа као сервис - Уклања потребу за управљањем комплетном инфраструктуром већ омогућава фокусирање само на испоручивање и управљање самом апликацијом, јер се брига о заузимању ресурса, одржавању софтвера и сл. препушта *cloud* провајдеру.

3. Софтвер као сервис - Пружа комплетан производ који одржава провајдер. У овом случају се не мора размишљати о томе како одржавати сервис или о његовој инфраструктури, већ само о томе на који начин ће овај софтвер бити коришћен.

2.3 Serverless computing

Рачунарство без сервера (енгл. *serverless computing*) је *cloud* парадигма која омогућава програмерима да изграђују апликације и сервисе без потребе да обезбеђују и управљају њиховим серверима. У *serverless* архитектури комплетном позадинском инфраструктуром управља *cloud* провајдер, и једино што програмери раде је писање и испоручивање кода. *Serverless* архитектура је вођена догађајима (енгл. *event-driven*) и може бити иницирана извршавањем разних догађаја, попут акција корисника или промене података. Такође је високо скалабилна, јер *cloud* провајдер може аутоматски покренути нове инстанце кода као одговор на повећан саобраћај [4].

2.4 Микросервиси или Serverless, коју архитектуру одабрати?

Овде нема једног тачног одговора. Све зависи од случаја коришћења као и неких субјективних преференци.

Serverless архитектура је добар избор ако нам је потребна брза апликација вођена догађајима. Такође, ако нам је потребно да брзо дођемо до радне верзије апликације, без велике потребе или жеље за управљањем инфраструктуром, онда је ово добар приступ, јер омогућава брзу изградњу лаганих, флексибилних апликација које могу бити проширене или ажуриране релативно брзо.

Међутим, за комплексније и дуготрајније процесирање података, микросервиси су вероватно прихватљивије решење, како се инфраструктура и *tech stack* могу адаптирати по потреби [4].

2.5 Амазон веб сервиси

Амазон веб сервиси (енгл. *Amazon Web Services* – *AWS*) су обимна, стално развијајућа *cloud* платформа коју пружа Амазон. Укључује мешавину инфраструк-

тура као сервис, платформа као сервис и софтвер као сервис понуда. *AWS* сервиси нуде организационе алате, попут рачунарске снаге, складишта база података и сервиса за испоручивање садржаја [5].

2.6 DynamoDB

Amazon DynamoDB је потпуно управљана, нерелациона (енгл. *noSQL*) база података која омогућава брзе и предвидиве перформансе и конзистентну скалабилност. Основне компоненте [6]:

Табела - *DynamoDB* складишти податке у табелама. Табела је колекција података (ставки).

Ставка - Група атрибута која се може јединствено идентификовати унутар ставки једне табеле, пандан редовима у неким релационим системима база података.

Атрибут - Свака ставка се састоји од једног или више атрибута. Атрибут је фундаментални елемент података, нешто што нема потребе да се даље раставља. Атрибути у *DynamoDB* табели се могу поредити са колонама у релационим системима база података, са разликом да се осим примарног кључа унапред не дефинишу називи и типови атрибута

Примарни кључ – Јединствено идентификује сваку ставку табеле, тако да ниједна ставка не може имати исти примарни кључ. Постоје два различита типа примарних кључева:

1. Партициони кључ - Једноставни примарни кључ, састоји се од једног атрибута познатог као партициони кључ

2. Композитни примарни кључ - Састоји се од два атрибута: партициони кључ и кључ за сортирање (тзв. сорт кључ)

У табели која има партициони кључ и кључ за сортирање могуће је да више ставки има исту вредност партиционог кључа, али такве ставке морају имати јединствену вредност сорт кључа, како би могле бити јединствено идентификоване. Све ставке са истом вредношћу партиционог кључа су складиштене заједно, у истој партицији, сортиране по вредности сорт кључа.

Секундарни индекс - Секундарни индекс нам омогућава да вршимо упите над базом користећи алтернативни кључ, поред коришћења примарног кључа. Постоје:

1. Глобални секундарни индекси (партициони кључ и сорт кључ се могу разликовати од оних у табели)

2. Локални секундарни индекси (исти партициони кључ, али различит сорт кључ)

DynamoDB Streams – опционо својство „хватања“ догађаја модификације података у *DynamoDB* табелама. Подаци о овим догађајима се појављују у реду у приближно реалном времену, у редоследу у

ком су се догађаји десили. Сваки догађај је представљен записом (енгл. *stream record*) и он се креира приликом догађаја додавања, ажурирања или брисања ставке из табеле.

Могуће је користити *DynamoDB Streams* заједно са *AWS Lambda* функцијама, да се креирају окидачи (енгл. *trigger*) – код који се извршава аутоматски када год се догађај од интереса појави у *stream*-у, што је и искоришћено у овом раду.

2.7 GraphQL

GraphQL је језик за креирање упита и окружење за извршавање тих упита над постојећим подацима, користећи унапред дефинисану шему типова података. Обезбеђује комплетан и разумљив опис података, даје клијентској страни моћ да тражи и добије тачно оно што јој је потребно од података, и омогућава добављање више ресурса у једном захтеву праћењем референци између њих. Главни концепти су: **шема** (сет типова који у потпуности описује сет података над којим је могуће извршавати упите), **тип** (тип објекта који је могуће додати), **упит** (служи за читање података), **мутација** (служи за ажурирање података) [7].

2.8 AppSync

AWS AppSync је потпуно управљан *GraphQL API* сервис који, у правом *serverless* облику, омогућава креирање *API*-ја без брига о томе како га хостовати [8].

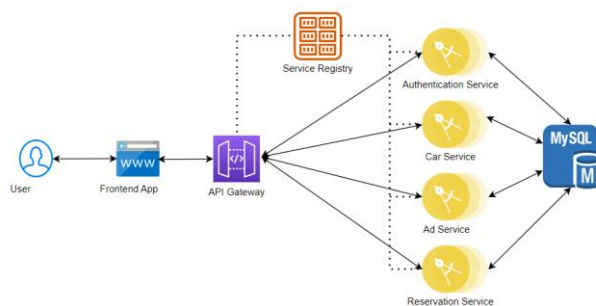
3. АРХИТЕКТУРА МИКРОСЕРВИСНОГ СИСТЕМА

Rent-a-Car систем се састоји од корисничке (фронтенд) апликације која комуницира са микросервисима преко *Zuul API Gateway*-а (Слика 1) Систем је направљен тако да подржи кључне функционалности:

- Регистрација и пријављивање корисника
- Креирање аутомобила који ће бити повезан са одређеним корисником и за који ће он моћи да креира огласе за изнајмљивање
- Креирање огласа за изнајмљивање аутомобила
- Претраживање и филтрирање доступних огласа
- Изнајмљивање огласа

На основу ових функционалности и главних ентитета у систему произашла су четири микросервиса:

- Аутентификациони микросервис – чува податке о корисницима и задужен је за ауторизацију пристиглих захтева
- Сервис аутомобила – чува информације о аутомобилима у систему
- Сервис огласа – чува податке о огласима у систему
- Сервис резервација – бави се чувањем креираних захтева за резервације и самих резервација



Слика 1 - Дијаграм архитектуре *Rent-a-Car* апликације

4. ИМПЛЕМЕНТАЦИЈА

4.1 Миграција базе из релационе у нерелациону

Применом познатих концепата и стратегија моделовања *DynamoDB* табеле, а полазећи од потребних шаблона приступа које апликација треба да задовољи, добијен је следећи модел:

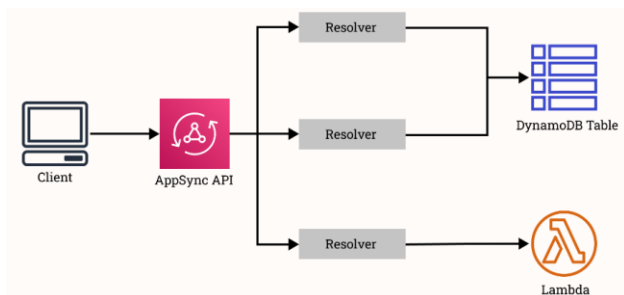
Главна табела апликације је табела чији партициони кључ представља *ID* корисника, сорт кључ представља *ID* Огласа, Аутомобила или Резервације, а сваки *ID* ће започињати префиксом *AD#*, *CAR#* или *RES#* како би се приликом претраживања могла разликовати ова три ентитета. На овај начин сви огласи, аутомобили и резервације једног корисника су груписани у истој партицији, што је врло корисно, узимајући у обзир карактеристичне шаблоне приступа које треба задовољити, а који су везани за добављање поменутих података за једног корисника.

Веза између Огласа и Аутомобила је представљена дуплирањем података за Аутомобил у сваком Огласу, како би им се могло директно приступити у сваком огласу, без додатних референцирања или упита, а из разлога што је учесталост измене података Аутомобила занемарљива. Што се тиче Огласа и Резервација, за њихову везу је могуће искористити глобални секундарни индекс. Наиме, у ставку Резервације је само потребно додати атрибут *ID* Огласа, који ће се искористити као партициони кључ глобалног секундарног индекса, а сорт кључ остаје исти као у главној табели.

Да би се подржало претраживање Огласа по локацији, направљен је још један глобални индекс чији ће партициони кључ бити локација огласа, а сорт кључ остаје исти.

4.2 Коришћење AppSync-а

Да би се брзо и једноставно могао направити *API* који подржава све основне операције над *DynamoDB* моделом, искоришћен је *AppSync* сервис. Уз помоћ њега креирана је *GraphQL* шема података која је потом повезана са *DynamoDB* табелом која представља њен извор података, а да би се функционалност комплетирали креирани су потребни резолвери *GraphQL* модела, који представљају спону између *GraphQL* захтева и одговора са *DynamoDB* табелом.



Слика 2 – Гرافички приказ AppSync комуникације [8]

4.3 Коришћење Lambda функције и DynamoDB Stream-a

Остаје још једна функционалност која није покривена, а то је само резервисање огласа. У *Rent-a-Car* систему је постојала улога администратора, који је ручно одобравао или одбијао пристигле захтеве, како би се избегли проблеми са конкурентношћу.

Овде је тај задатак аутоматизован коришћењем Lambda функције и DynamoDB Stream-a. Наиме, пошто је могуће „хватати“ догађаје у DynamoDB табели, уз помоћ DynamoDB Stream-a над табелом и Lambda функције коју иницирају догађаји у stream-у, то је искоришћено да би омогућили аутоматско одобравање или одбијање резервације на основу тренутне доступности огласа. Дакле, када корисник направи захтев за резервацију, иницира се извршавање Lambda функције која проверава доступност огласа и на основу тог резултата одобрава или одбија захтев за резервацију.

5. ЗАКЉУЧАК

Описаном миграцијом микросервисне апликације у овом раду извршена је трансформација у *serverless* апликацију са нерелационим моделом података. Добијена је апликација која има једноставни нерелациони модел који је ефективно смештен у једну табелу, уз два глобална секундарна индекса. Тиме су искоришћене бројне погодности које нуди DynamoDB, у виду брзих и предвидивих перформанси као и високе скалабилности.

На миграцију микросервисне архитектуре на *serverless* се одлучује све више компанија, услед бројних предности које *serverless* архитектура доноси, али пре него што се одлучи на такав корак потребно је добро размислити о томе шта она носи са собом. Сам процес је доста сложен и умногоме зависи од тренутне архитектуре, циљева и ограничења система. Познавање пословних процеса и тачно уочавање шаблона коришћења које је потребно подржати, те успешно моделовање базе података је кључна тачка успешне миграције.

Лоше постављен модел, који касније захтева промене може донети више штете него користи. Зато је први корак моделовања података уствари и најважнији корак миграције, и потребно му је поклонити највише пажње. Даље је веома важно идентификовати компоненте и функционалности које могу бити мигриране у *serverless* функције или сервисе, као и рефакторисати код и податке да би се уклопили и прилагодили *serverless* моделу.

6. ЛИТЕРАТУРА

- [1] What are Microservices?
<https://cmcglobal.com.vn/cloud-migration/what-are-microservices-microservices-in-cloud-computing-explained>
- [2] Рачунарство у облацима – Cloud computing
<https://www.industrija.rs/vesti/clanak/racunarstvo-u-oblacima-cloud-computing>
- [3] Types of cloud computing
<https://aws.amazon.com/what-is-cloud-computing>
- [4] Serverless vs Microservices – Which Architecture to Choose?
<https://www.solutelabs.com/blog/serverless-vs-microservices>
- [5] Amazon Web Services (AWS)
<https://www.techtarget.com/searchaws/definition/Amazon-Web-Services>
- [6] What is Amazon DynamoDB?
<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html>
- [7] <https://graphql.org/>
- [8] What is AWS AppSync <https://sst.dev/chapters/what-is-aws-appsync.html>

Кратка биографија:



Сенка Шојић рођена је 1997. године у Врбасу. Завршила је основне академске студије 2020. године на Факултету техничких наука у Новом Саду. Уписала је мастер студије исте године, студијски програм Рачунарство и аутоматика - Електронско пословање.

WEB APLIKACIJA ZA PRETRAGU WINDOWS LOGOVA**WEB APPLICATION FOR SEARCHING WINDOWS EVENT LOGS**Anđela Čičković, *Fakultet tehničkih nauka, Novi Sad***Oblast – RAČUNARSTVO I AUTOMATIKA**

Kratak sadržaj – Rad pruža pregled teorijskih osnova u oblasti logova događaja na Windows operativnom sistemu. Takođe, u okviru ovog rada implementirana je i detaljno opisana aplikacija za prikupljanje, pretragu i filtriranje logova događaja unutar lokalne mreže.

Ključne reči: Log događaja, Windows, prikupljanje, pretraga, filtriranje

Abstract – This paper provides an overview of the theoretical foundations about event logging on Windows operating system. An application for collecting, searching and filtering event logs in local network has been implemented and described as part of this paperwork, as well.

Keywords: Event log, Windows, collecting, searching, filtering

1. UVOD

Ubrzani razvoj računarskih sistema i svjetska zavisnost od tehnologije u današnje vrijeme nose sa sobom značajan napredak, ali i brojne izazove. Računarski sistemi postali su kompleksniji, obimniji i brži, pa se i broj procesa koji se odvijaju u njima eksponencijalno povećao. Samim tim, stvorila se potreba za efikasnijim metodama praćenja i upravljanja sistemima, pa je logovanje događaja (engl. *event logging*) u sistemu postalo izuzetno važno [1].

Logovi predstavljaju važan resurs u savremenim računarskim sistemima. Oni vjerno prikazuju redoslijed aktivnosti u radu sistema, odnosno, prikazuju istoriju događaja koja može pomoći da se isprati bilo koja aktivnost u sistemu [2]. Logovi vrše evidentiranje informacija o neuspjelim pokušajima pristupa sistemu, neovlašćenim radnjama, greškama, upozorenjima, uspješnim radnjama ili bilo kojoj sumnjivoj aktivnosti koja bi mogla ugroziti sistem [3]. Kroz logove se može vršiti nadzor rada sistema, što dalje omogućuje sprovođenje analize, optimizacije, te ranog otkrivanja potencijalnih problema i bezbjednosnih rizika [4]. Uslijed složenosti i obima današnjih sistema, količina logova koja se generiše brzo raste, pa efikasno upravljanje logovima i analiza istih predstavljaju veliki izazov. Tu na scenu stupaju aplikacije za pretragu i filtriranje logova. Ove aplikacije omogućavaju sistemskim administratorima da efikasno pretražuju, filtriraju i analiziraju logove, što im

omogućava brzo reagovanje na potencijalne probleme i sigurnosne prijetnje i rizike [5].

Kao dio ovog rada izvršeno je istraživanje i analiza procesa logovanja događaja na računarima koji koriste Windows operativni sistem. Takođe, implementirano je softversko rješenje za prikaz, pretragu i filtriranje logova sa bilo kojeg računara u lokalnoj mreži.

2. WINDOWS EVENT LOGS

Već u uvodnom poglavlju su navedeni neki od najznačajnijih pojmova koji će se višestruko pominjati dalje u radu. Zbog toga, bitno je na samom početku razumjeti svaki od njih, njihovu međusobnu povezanost i namjenu.

Događaj (engl. *event*) predstavlja promjenu stanja sistema, pri čemu neki događaji predstavljaju redovne sistemskih aktivnosti (na primjer: uspješna prijava korisnika), dok drugi mogu da predstavljaju nepredviđene promjene koje su se desile u sistemu (na primjer: neuspješna instalacija drajvera) [6].

Evidencija događaja omogućava bilježenje svih događaja u operativnom sistemu, kako bi se na taj način osigurala kontrola pristupa i sigurnost sistema. Događaji se bilježe u **logove događaja** (engl. *event logs*) [7].

Logovi događaja, u daljem tekstu samo logovi, su fundamentalni fajlovi operativnog sistema, koji u svakom trenutku pružaju mogućnost uvida u sve promjene stanja sistema [8].

Počevši od *Microsoft Windows Vista* verzije operativnog sistema, *Microsoft* koristi *Windows Event Log* sistem za logovanje, koji događaje dijeli u dvije kategorije:

1. Windows logs – kategorija namijenjena čuvanju događaja koji se odnose na cijeli sistem. Unutar ove kategorije razlikuje se pet tipova događaja:

- **Application** – događaji na raspolaganju korisničkim aplikacijama, kako bi bilježile događaje od značaja za cijeli sistem;

- **Security** – događaji vezani za bezbjednost sistema i upotrebu sistemskih resursa;

- **System** – sistemski događaji, odnosno događaji u različitim dijelovima operativnog sistema;

- **Setup** – događaji nastali prilikom instalacije *Windows* operativnog sistema;

- **Forwarded Events** – događaji koji su prosljeđeni sa drugih servera, ukoliko je prosljeđivanje događaja konfigurisano.

2. Application and Services logs – kategorija logova namijenjena čuvanju događaja izazvanim aplikacijama ili

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Goran Sladić, red. prof.

komponentama aplikacije, a koji se ne odnose na cijeli sistem [9].

Tema ovog rada jesu *Windows* logovi. Stoga, u daljem tekstu će se podrazumijevati da, kada se pominju logovi, to se odnosi na prethodno opisane tipove *Windows* logova, bez *Forwarded events* tipa.

Kada je u pitanju format zapisa logova koristi se **EVTX** (*Windows Extensible Markup Language (XML) Event Log*) format. Ovaj format podrazumijeva čuvanje podataka u *XML* formatu, što omogućava visok nivo detaljnosti prilikom pregledanja logova od strane *third party* softvera. Za opisivanje logova koriste se *property*-ji, a najznačajniji i najčešći u upotrebi, predstavljeni su u tabeli 1.

Tabela 1. *Property*-ji kojima se opisuju logovi

Naziv <i>property</i> -ja	Opis
Source	Naziv softvera (program ili komponenta sistema) koji je zabilježio događaj.
Event ID	Broj koji jedinstveno identifikuje tip događaja
Level	Nivo ozbiljnosti/važnosti događaja
User	Naziv korisnika u čije ime se događaj desio
Operational Code	Numerička vrijednost koja identifikuje aktivnost ili specifičan dio aktivnosti koji se odvijao u trenutku događaja
Log	Naziv fajla (dnevnika događaja) u kome je događaj zabilježen
Task category	Identifikuje kategoriju aktivnosti <i>event publisher</i> -a
Keywords	Skup kategorija ili oznaka koje se mogu koristiti za filtriranje ili traženje događaja
Computer	Naziv računara na kome se događaj desio
Date and Time	Datum i vrijeme nastanka događaja u dnevniku događaja

Kada je u pitanju nivo ozbiljnosti događaja pomenut u tabeli 1, razlikuje se pet različitih nivoa, a to su:

- **Information** – obavještenje, identifikuje promjenu u aplikaciji ili nekoj komponenti aplikacije, kao na primjer uspješan završetak neke operacije, pokretanje servisa, kreiranje resursa i slično.
- **Verbose** - detaljno obavještenje, koristi se kod događaja koji imaju detaljne informacije o sistemskoj ili aplikativnoj aktivnosti.
- **Warning** – upozorenje, ukazuje na neočekivanu aktivnost (potencijalni problem) koja bi mogla rezultirati ozbiljnijim problemom ako se ostavi bez nadzora.
- **Error** – greška, ukazuje na problem koji se dogodio i koji može uticati na funkcionalnosti van aplikacije ili komponente koje su prouzrokovale događaj.

- **Critical** – kritična greška, ukazuje na problem koji se desio i od koga se aplikacija ili komponenta na kojoj se isti dogodio ne mogu automatski oporaviti.

3. WINDOWS EVENT VIEWER

Windows Event Viewer jeste alat koji dolazi zajedno sa *Windows* operativnim sistemom i nudi različite funkcionalnosti kojima se obezbjeđuje praćenje i analiza logova. Ključne funkcionalnosti ovog alata su: pregled logova, filtriranje logova, pregled detalja događaja, kreiranje i primjena filtera, pregled logova sa udaljenih računara, i slično. Ovaj alat pruža jasan i pregledan prikaz logova na računaru i daje mogućnost praćenja performansi računara. Međutim, kao najveći nedostatak ovog alata smatra se kompleksnost prilikom kreiranja filtera, posebno za korisnike koji nemaju dovoljno znanja o radu sistema i logovima događaja. Kako bi se kreirao filter, alat zahtjeva od korisnika poznavanje pravila, strukture i formata *XPath* izraza, kao i ograničenja koje ima *Windows Event Log* vezano za iste.

4. MODEL SISTEMA ZA PRETRAGU LOGOVA

Ovaj sistem treba da omogući korisnicima bogat korisnički interfejs, tako da, brzo i lako mogu imati uvid u logove ne samo jednog, nego svih računara u lokalnoj mreži. Pored toga, ovaj sistem treba da omogući korisnicima jednostavnu i efikasnu pretragu i filtriranje logova, bez potrebe za poznavanjem strukture, formata i ograničenja *XPath query*-ja i sličnih tehničkih ograničenja.

4.1. Aktivnosti korisnika sistema

U sistemu za prikaz i pretragu logova postoji tačno jedan tip korisnika koji može da vrši sljedeće aktivnosti:

1. Korisnik koji pristupa sistemu može da pregleda i odabere jedan od dostupnih računara u mreži za koji želi da prikaže logove.
2. Korisnik ima tabelarni prikaz logova za odabrani računar u mreži.
3. Korisnik može da filtrira podatke prikazane u tabeli, po datumu i vremenu, po tipu događaja, po identifikatoru događaja, po ključnim riječima, po korisnicima, po izvoru događaja, te po nivou ozbiljnosti događaja.
4. Korisnik ima mogućnost da koristi opciju napredne pretrage, te na taj način kreira dodatne uslove za filtriranje logova, koji nisu dostupni kroz standardne opcije filtera.
5. U sklopu tabelarnog prikaza logova, korisnik može da odabere bilo koji log i prikaže detalje o istom.
6. Korisnik ima opciju da odštampa logove koji se trenutno prikazuju u tabeli.
7. Takođe, korisnik može da preuzme logove prikazane u tabeli u *CSV* formatu.
8. Dodatno, korisnik može da konfiguriše izgled tabele, može da mijenja kolone koje se prikazuju, te da mijenja kompaktnost, odnosno visinu redova.

4.2. Arhitektura sistema

Osnovni dijelovi sistema su jedna klijentska i jedna serverska aplikacija.

Serverska aplikacija se može posmatrati kao agent za prikupljanje logova sa ostalih računara u lokalnoj mreži. Po zahtjevu klijenta, server uspostavlja sesiju i prikuplja podatke sa odgovarajućeg računara u mreži.

Klijentska aplikacija predstavlja korisnički interfejs za uspostavljanje komunikacije sa serverom i obavljanje svih aktivnosti opisanih u prethodnom odjeljku (4.1).

5. IMPLEMENTACIJA

5.1. Serverska aplikacija

Serverska aplikacija napisana je u C# programskom jeziku, upotrebom .NET 6 radnog okruženja (engl. *framework*). Sastoji se od dva projekta, *Host* i *Logic*. *Host* (5.1.1.) predstavlja REST servis, zadužen za uspostavljanje komunikacije sa klijentom. Drugi projekat, *Logic*, zadužen je za realizaciju poslovne logike sistema, koja je u najvećoj mjeri implementirana u servisima, *NetworkDeviceService* (5.1.2.) i *EventLogService* (5.1.3.).

5.1.1. Host

Host osluškuje zahtjeve klijenta na četiri *endpoint*-a, koji se koriste za:

1. dobavljanje logova (*url*: „*api/EventLogs*“)
2. filtriranje logova (*url*: „*api/EventLogs/filter*“)
3. dobavljanje opcija za filtriranje logova (*url*: „*api/EventLogs/filterOptions/{eventType}*“)
4. dobavljanje dostupnih računara u lokalnoj mreži (*url*: „*api/NetworkDevice/availableHosts*“)

Zahtjevi pristigli na ove *endpoint*-e dalje se proslijeđuju odgovarajućim servisima koji predstavljaju dio *Logic* projekta (5.1.2.). Kada je u pitanju autentifikacija, aplikacija je podešena tako da autentifikuje korisnika koristeći *Negotiate SSP* (*Security Support Provider*).

5.1.2. Logic – NetworkDeviceService

NetworkDeviceService vrši analizu mreže i vraća informacije o dostupnim računarima u mreži. Analiza mreže vrši se uz pomoć alata *nmap* (*Network Mapper*). To je alat otvorenog koda, namijenjen za brzo skeniranje velikih mreža i reviziju sigurnosti (engl. *security auditing*). Iako se najčešće koristi za reviziju sigurnosti, ovaj alat je našao primjenu kod mnogih sistemskih i mrežnih administratora za svakodnevne, rutinske zadatke, kao što su popisivanje mrežnog inventara i nadgledanje *uptime*-a računara i servisa. Rezultat *nmap* skeniranja predstavlja popis skeniranih uređaja sa dodatnim informacijama, pri čemu količina i format dodatnih informacija zavise od proslijeđenih parametara. U slučaju ove aplikacije, osim dostupnih računara u mreži, nijedna druga informacija nije neophodna, pa se ni ne pozivaju komande za njihovo generisanje.

5.1.2. Logic - EventLogService

EventLogService namijenjen je rukovanju logovima. Ovaj servis uspostavlja komunikaciju sa ostalim računarima u mreži sa kojih je neophodno prikupiti podatke. Komunikacija između ovih računara odvija se upotrebom

klasa, koje će biti opisane dalje u tekstu, a koje se nalaze unutar *System.Diagnostics.Eventing.Reader namespace*-a.

EventLogQuery predstavlja upit, koji definiše koji log podaci treba da budu vraćeni kao rezultat operacije čitanja logova. Koristeći ovu klasu moguće je postaviti različite filtere i uslove pretrage.

Prilikom kreiranja objekta ove klase, proslijeđuju se dva parametra. Prvi parametar može da bude ili putanja do fajla gdje se nalaze logovi nad kojima je potrebno izvršiti upit ili naziv sistemskog log fajla nad kojim je potrebno izvršiti upit. Pomoću drugog parametra, enumeracije *PathType*, definiše se da li je prvi parametar putanja do fajla (*FilePath*) ili naziv fajla (*LogName*). U slučaju ovog rješenja upiti se vrše isključivo and sistemskim log fajlovima. Prilikom kreiranja upita, moguće je proslijediti i treći parameter, koji predstavlja *query* string, na osnovu koga je moguće izvršiti filtriranje upita.

EventLogSession predstavlja klasu uz pomoć koje se uspostavlja sesija kako bi se izvršio pristup *Event Log* servisu na lokalnom računaru ili računaru u mreži. Prilikom kreiranja objekta ove klase, kao parametar se proslijeđuje naziv računara sa koga je potrebno pročitati logove. Po *default*-u se koristi *Windows* autentifikacija prilikom pristupanja udaljenom računaru, odnosno, šalju se korisničko ime i lozinka korisnika koji poziva ovu metodu.

EventLogRecord jeste klasa kojom se predstavljaju logovi događaja. Svaki objekat ove klase nosi podatke o vremenu kada je log zabilježen, nazivu računara na kojem je zapisan događaj, identifikacionom broju korisnika, identifikacionom broju događaja i još mnoge druge. Osnovne metode ove klase su *FormatDescription()*, koja vraća opis događaja u formatu prikazanom u sistemskom logu događaja i *ToXml()* koja vraća *xml* zapis događaja. Obzirom da ova klasa sadrži veliki broj *property*-ja, objekti ove klase se prije slanja klijentu mapiraju na objekte klase *EventLogDto* upotrebom *Automapper*-a.

EventLogReader klasa omogućuje čitanje logova. Kao rezultat operacije čitanja vraća se objekat klase *EventRecord*. Prilikom kreiranja objekta klase *EventLogReader* konstruktoru se proslijeđuje kreirani upit (objekat klase *EventLogQuery*).

Pored ovih klasa, kako bi se implementirale neophodne funkcionalnosti, kreirane su dvije pomoćne klase i to *EventLogQueryBuilder* i *EventRecordExtensions*.

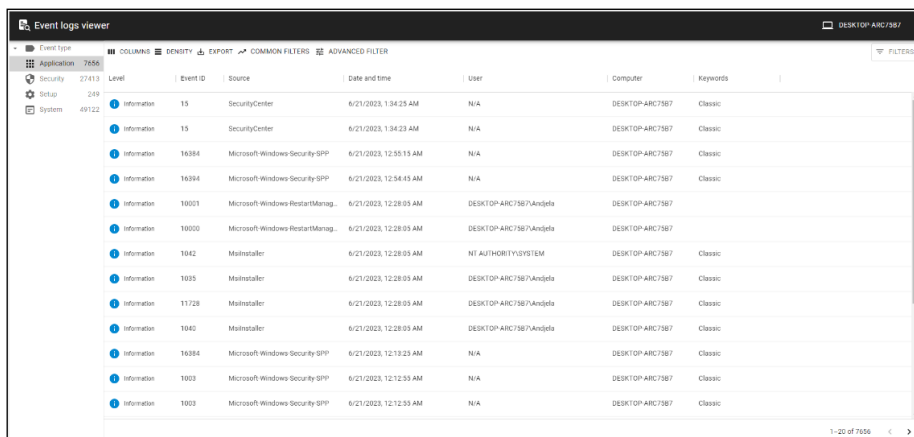
EventLogQueryBuilder klasa koristi se za izgradnju prethodno pomenutog *query* stringa (upita). Kreiranje upita vrši se na osnovu pravila za kreiranje *XML* upita uz pomoć *XPath* izraza.

EventRecordExtensions klasa sarži *extension* metodu *MatchesAll* za klasu *EventRecord*. Ova metoda koristi se da provjeri da li objekat klase zadovoljava kriterijume napredne pretrage. Potreba da se implementira ova metoda, proistekla je istovremeno kao rezultat zahtjeva za implementacijom napredne pretrage sa jedne strane, i ograničenja kada su u pitanju *XPath* upiti, sa druge strane.

5.2. Klijentska aplikacija

Klijentska aplikacija implementirana u *React-u* predstavlja alat uz pomoć koga korisnik može brzo, lako i jednostavno da pregleda i filtrira logove sa računara

unutar mreže. Sve funkcionalnosti korisnik obavlja na jednoj stranici, izgrađenoj od više komponenti. Na slici 1. prikazana je stranica koja se prikazuje korisniku odmah nakon pokretanja aplikacije.



The screenshot shows a web application titled "Event logs viewer". It features a sidebar on the left with a tree view containing "Application" (7656), "Security" (27413), "Setup" (249), and "System" (49122). The main area displays a table of log entries with columns: Level, Event ID, Source, Date and time, User, Computer, and Keywords. The table contains 15 rows of log data, including events from SecurityCenter, Microsoft Windows Security SPP, and Microsoft Windows RestartManager.

Level	Event ID	Source	Date and time	User	Computer	Keywords
Information	15	SecurityCenter	6/21/2023, 1:34:25 AM	N/A	DESKTOP-ARC7367	Classic
Information	15	SecurityCenter	6/21/2023, 1:34:23 AM	N/A	DESKTOP-ARC7367	Classic
Information	16384	Microsoft Windows Security SPP	6/21/2023, 12:55:15 AM	N/A	DESKTOP-ARC7367	Classic
Information	16384	Microsoft Windows Security SPP	6/21/2023, 12:54:45 AM	N/A	DESKTOP-ARC7367	Classic
Information	18001	Microsoft Windows RestartManager	6/21/2023, 12:28:05 AM	DESKTOP-ARC7367\Andjela	DESKTOP-ARC7367	Classic
Information	18000	Microsoft Windows RestartManager	6/21/2023, 12:28:05 AM	DESKTOP-ARC7367\Andjela	DESKTOP-ARC7367	Classic
Information	1842	MailInstaller	6/21/2023, 12:28:05 AM	NT AUTHORITY\SYSTEM	DESKTOP-ARC7367	Classic
Information	1835	MailInstaller	6/21/2023, 12:28:05 AM	DESKTOP-ARC7367\Andjela	DESKTOP-ARC7367	Classic
Information	17728	MailInstaller	6/21/2023, 12:28:05 AM	DESKTOP-ARC7367\Andjela	DESKTOP-ARC7367	Classic
Information	1840	MailInstaller	6/21/2023, 12:28:05 AM	DESKTOP-ARC7367\Andjela	DESKTOP-ARC7367	Classic
Information	16384	Microsoft Windows Security SPP	6/21/2023, 12:13:23 AM	N/A	DESKTOP-ARC7367	Classic
Information	18003	Microsoft Windows Security SPP	6/21/2023, 12:12:55 AM	N/A	DESKTOP-ARC7367	Classic
Information	18003	Microsoft Windows Security SPP	6/21/2023, 12:12:55 AM	N/A	DESKTOP-ARC7367	Classic

Slika 1 Početna stranica aplikacije

Centralni dio stranice predstavlja tabela sa log podacima. Po default-u, u tabeli se prikazuju *Application* logovi računara na kome je pokrenuta serverska aplikacija.

Na lijevoj strani se nalazi meni u okviru koga je moguće promijeniti tip događaja (*EventType*) čiji se logovi prikazuju.

U gornjem desnom uglu *header-a*, ispisan je naziv računara čiji logovi se prikazuju u tabeli, a klikom na isti otvara se padajući meni u okviru koga je moguće promijeniti *host* računar.

Takođe tabela ima i svoj *toolbar*, koji ima sljedeće opcije: promjena kolona koje se prikazuju u tabeli (*COLUMNS*), promjena kompaktnosti redova (*DENSITY*), *export* sadržaja tabele (*EXPORT*), brzo filtriranje (*COMMON FILTERS*), te opcija naprednog filtriranja (*ADVANCED FILTER*)

6. ZAKLJUČAK

U radu je opisana struktura i osnovna namjena *Windows* logova događaja. Implementirano je web rješenje koje omogućuje prikaz, pretragu i filtriranje logova, sa bilo kojeg računara u mreži. Sve prethodno navedeno omogućeno je korisniku bez potrebe za promjenom okruženja, instalacijom dodatnog softvera ili prelaskom sa jednog računara na drugi.

Na ovaj način se značajno ubrzava proces analize i otkrivanja problema, što su ključne aktivnosti u procesu povećanja efikasnosti i dostupnosti računarskih sistema. Neki od mogućih koraka u daljem razvoju i unaprjeđenju rješenja bi mogli biti:

- Omogućiti korisniku da grupiše uslove napredne pretrage uz pomoć logičkih operatora "and" i "or", te uvesti mogućnost negiranja nekog uslova uz pomoć operatora "not".
- Na korisničkom interfejsu dodati grafički prikaz statistika i izvještaja za logove.
- Omogućiti korisniku *import* i *export* filtera;
- Omogućiti dvosmjernu komunikaciju, odnosno, omogućiti *real-time* osvježavanje logova u tabeli ili bar uvesti sistem notifikacija.

7. LITERATURA

- [1] Nicoleta Stanciu, "Importance of event log management to ensure information system security", Academy of Economic Studies, Bucharest, 2013.
- [2] Zhuangbin Chen, Jinyang Liu Wenwei Gu, Yuxin Su, Jieming Zhu, Yongqiang Yang, Michael R. Lyu "Deep Learning-based System Log Analysis for Anomaly Detection", Januray 2022.
- [3] GSEC Practical Assignment, "Importance of Event Logging", SANS institute, 2003
- [4] <https://www.blumira.com/what-are-event-logs-and-why-do-they-matter/> (pristupljeno u junu 2023.)
- [5] Marcello Cinque, Raffaele Della Corte, Antonio Pecchia, "Contextual filtering and prioritization of computer application logs for security situational awareness", 2020
- [6] Risto Vaarandi, "Tools and techniques for event log analysis", Tallin Universitz of Technology, 2005
- [7] Lei Zeng, Yang Xiao, Hui Chen, Bo Sun and Wenlin Han, "Computer operating system logging and security issues: a survey", 2016
- [8] Vanja M. Korać, "Digitalna forenzika u funkciji zaštite informacionog sistema baziranog na Linux i Windows platformama", Univerzitet u Beogradu, 2014
- [9] <https://www.odseknis.akademijanisanis.edu.rs/wp-content/plugins/vtspredmeti/uploads/1585731723ARM%20Predavanje%2010%202016.pdf?script=lat> (pristupljeno u junu 2023.)

Kratka biografija:



Andela Čičković rođena je u Trebinju 1997. god. Master rad na Fakultetu tehničkih nauka iz oblasti Računarstva i automatike – Informaciona bezbjednost odbranila je 2023.god.

kontakt: andjela.cickovic@gmail.com

IMPLEMENTACIJA SISTEMA ZA PLAĆANJE NA AWS INFRASTRUKTURI

IMPLEMENTATION OF A PAYMENT SYSTEM ON AWS INFRASTRUCTURE

Monika Erdeg, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U radu su opisane komponente i arhitektura sistema za online plaćanje karticama. Analizirana je mogućnost za eksploataciju sistema na elementima računarstva u sklopu AWS infrastrukture. Predstavljena je automatizacija procesa raspoređivanja softvera i konfiguracije infrastrukture.

Ključne reči: *Elektronsko plaćanje, CI/CD, GitHub akcije, IaC, Terraform, AWS*

Abstract – *This paper presents the implementation of a payment system. Deployment processes are discussed. The paper also provides a detail explanation of the AWS infrastructure setup, which is used for hosting the application. Finally, it discloses infrastructure automation and proposes a solution for Infrastructure as Code.*

Keywords: *Electronic payment, CI/CD, GitHub actions, IaC, Terraform, AWS*

1. UVOD

U današnjem digitalnom dobu, online kupovina postala je neizostavan deo svakodnevnog života. Sa sve većim brojem korisnika koji se oslanjaju na digitalne platforme za obavljanje transakcija, sigurnost i stabilnost sistema za plaćanje postali od suštinskog značaja. AWS (Amazon Web Services) infrastruktura pruža mogućnosti za simulaciju i implementaciju bankarskog sistema za plaćanje karticama tokom online kupovine.

U savremenom poslovnom okruženju, implementacija skalabilne infrastrukture zahteva ne samo tehničko razumevanje već i duboko poznavanje raznih aspekata upotrebe servisa i kompromisa koje je potrebno napraviti radi postizanja optimalnih rezultata. AWS infrastruktura omogućava simulaciju raznih scenarija, analizu performansi i testiranje sigurnosnih mehanizama.

U ovom radu će biti istraženo kako AWS može da se iskoristi za simulaciju i implementaciju sistema za plaćanje karticama. Fokus će biti na analizi različitih komponenti infrastrukture, kao i na implementaciji mehanizama raspoređivanja, kako aplikacije, tako i infrastrukture, kako bi se osigurala stabilnost i skalabilnost sistema. Takođe, biće stavljen akcenat na povezivanje lokalnih (on-premise) komponenti sa sistemom koji se izvršava u oblaku.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Goran Sladić, red. prof.

Kroz detaljno istraživanje i praktične primere, razmotriće se kako AWS pruža rešenja koja su ključna za uspešno obavljanje online transakcija u današnjem digitalnom ekosistemu.

2. SISTEM ZA PLAĆANJE KARTICAMA PRILIKOM ONLINE KUPOVINE

Koncentrator plaćanja predstavlja sistem kojim upravlja entitet čiji poslovni model je da posreduje između različitih servisa za plaćanje i klijenata koji žele da podrže plaćanje, ali ne žele da brinu o bezbednosti datih funkcija i njihovom održavanju.

Koncentrator ima API koji očekuje podatke o uplati. Otvara se stranica gde se bira kojim servisom plaćanja kupac želi da plati svoju kupovinu.

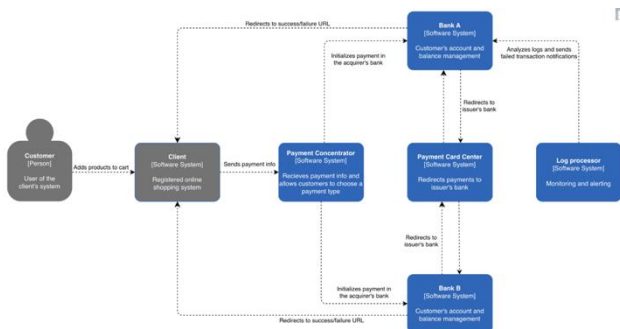
Plaćanje putem platne kartice podrazumeva protokol komunikacije između različitih učesnika:

- Kupac, koji vrši onlajn kupovinu i ima otvoren račun u banci,
- Prodavac, koji prodaje proizvode putem veb-prodavnice i poseduje račun u banci,
- Banka prodavca (Acquirer), koja pruža servis za onlajn plaćanje, gde kupac unosi podatke o svojoj platnoj kartici,
- Centar za platne kartice (PCC - Payment Card Center) – posreduje u međubankarskoj komunikaciji ovog tipa, te prihvata zahtev za transakciju od banke prodavca i prosleđuje ga banki kupca,
- Banka kupca (Issuer) – proverava stanje računa kupca i odobrava transakciju ukoliko postoje raspoloživa sredstva.

2.1. Pregled osnovnih komponenti sistema

Delovi sistema koji su deo implementacije koncentratora plaćanja, prikazanih na slici 1, su sledeći:

- Payment Concentrator - koncentrator plaćanja
- Bank A - banka u kojoj su registrovani neki od klijenata koncentratora (kupci/prodavci)
- Bank B - banka u kojoj su registrovani neki od klijenata koncentratora (kupci/prodavci)
- Payment Card Center - centar za platne kartice
- Log processor – sistem za analizu log-ova



Slika 1. C1 dijagram sistema

Od navedenih komponenti sistema, jedan deo (koncentrator plaćanja, centar za platne kartice i banke) se izvršava u oblaku, a drugi deo (log procesor) se izvršava lokalno (engl. *on-premise*). Potrebno je povezati ova dva dela sistema, kako bi mogla da komuniciraju i razmenjuju podatke sigurnim putem.

Aplikacije su pisane u Spring Boot radnom okviru. Za bazu podataka je korišten PostgreSQL servis za skladištenje i upravljanje podacima.

3. AMAZON WEB SERVICES

Amazon Web Services (AWS) predstavlja vodećeg pružaoca usluga za računarstvo u oblaku koji nudi širok spektar infrastrukturnih resursa i servisa kao usluga. AWS je deo Amazon.com kompanije i omogućava korisnicima pristup skalabilnim i fleksibilnim resursima u oblaku kako bi implementirali, testirali i pustili u produkciju svoje aplikacije i IT infrastrukture.

3.1 IAM (Identity and Access Management)

AWS Identity and Access Management (IAM) je servis koji nudi AWS, koji omogućava upravljanje korisničkim identitetima i njihovim pristupom AWS resursima. IAM omogućava sigurno kontrolisanje pristupa različitim AWS servisima i resursima, čime se obezbeđuje poverljivost i integritet podataka u AWS okruženju [1].

3.2 VPC (Virtual Private Cloud)

VPC (virtuelna privatna mreža) je servis AWS-a koji omogućava kreiranje privatne mreže u okviru AWS oblaka. Ova mreža je potpuno izolovana od drugih VPC-ova i resursa, čime se obezbeđuje visok nivo privatnosti i bezbednosti za aplikacije i podatke. Kroz jednostavne konfiguracije, omogućeno je prilagođavanje postavki VPC-a. To uključuje definisanje IP adresa za podmreže, kreiranje ruta i konfigurisanje mrežnih pristupnih kontrola (Network Access Control List) i sigurnosnih grupa [2].

3.3 EC2 (Elastic Cloud Compute)

Elastic Cloud Compute (EC2) je AWS servis koji omogućava korisnicima da iznajme virtualne servere u oblaku i pokreću svoje aplikacije na tim serverima. U okviru EC2 servisa, postoji nekoliko mogućnosti za izbor tipa servera: on-demand instance, spot instance, rezervisane instance (reserved instances) i dodeljeni serveri (dedicated hosts) [3].

3.4 ECS (Elastic Container Service)

Amazon Elastic Container Service (ECS) je AWS-ov servis koji omogućava upravljanje kontejnerizovanim

aplikacijama. Pomoću ECS-a moguće je jednostavno pokrenuti, zaustaviti i skalirati kontejnere koji se izvršavaju u datom okruženju [4].

3.4.1 ECS Anywhere

ECS Anywhere je servis koji pruža ECS, koji omogućava pokretanje i upravljanje kontejnerima u lokalnom okruženju, van AWS infrastrukture u oblaku. Ova usluga proširuje funkcionalnosti ECS-a i omogućava korišćenje istog koncepta i alata za upravljanje kontejnerima kako u oblaku, tako i u lokalnom okruženju [5].

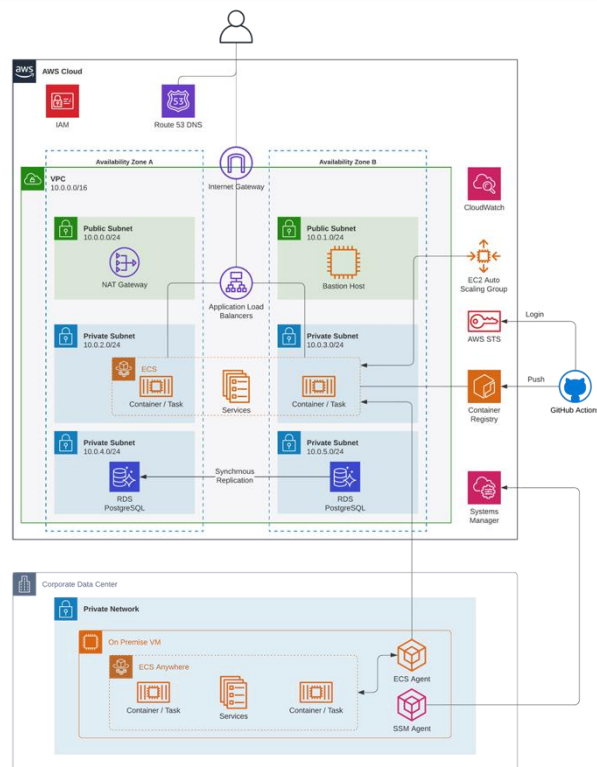
3.5 RDS (Relational Database Service)

Amazon Relational Database Service (RDS) je servis AWS-a koji omogućava lakše upravljanje relacionim bazama podataka u oblaku. RDS podržava različite popularne baze podataka, kao što su MySQL, PostgreSQL, Oracle, SQL Server, MariaDB i Amazon Aurora [6].

3.6 Route53

Amazon Route 53 je web servis za upravljanje DNS-om (Domain Name System). Omogućava korisnicima da registruju i upravljaju imenima domena, pridružuju ih AWS resursima i rutiraju korisničke zahteve ka pravim AWS servisima ili drugim infrastrukturnim resursima. Doprinosi poboljšanju performansi, dostupnosti i sigurnosti aplikacija i web sajtova [7].

4. INFRASTRUKTURA SISTEMA ZA PLAĆANJE



Slika 2. Dijagram AWS infrastrukture sistema

Na slici 2 je prikazan dijagram AWS infrastrukture datog sistema. Radi jednostavnosti prikaza, sve komponente su prikazane singularno (baze podataka, *load balancer*-i, ECS servisi i GitHub akcije nisu prikazani za svaku od aplikacija, već zbirno).

5. INFRASTRUKTURA KAO KOD

Infrastruktura kao kod (Infrastructure as Code) je proces kreiranja i upravljanja infrastrukturom putem koda, umesto putem manualnih procesa.

Sa IaC-om, kreiraju se konfiguracioni fajlovi koji sadrže specifikacije infrastrukture, što olakšava održavanje i distribuciju konfiguracija. Ovaj pristup takođe garantuje kreiranje istog okruženja pri svakom pokretanju. Kroz pretapanje konfiguracija u kod, kao i njihovo dokumentovanje, IaC doprinosi izbegavanju nedozvoljenih ili ad-hoc promena koje nisu dokumentovane [8].

Verzija kontrole je važan deo IaC-a, te bi konfiguracioni fajlovi trebali biti pod kontrolom izvornog koda (source control-om), kao i svaki drugi fajl sa softverskim izvornim kodom. Implementacija infrastrukture kao koda donosi i mogućnost dekompozicije infrastrukture na modularne komponente, koje mogu biti kombinovane na različite načine [8].

Automatizacija infrastrukture putem IaC-a, znači da programeri ne moraju ručno upravljati serverima, operativnim sistemima, skladištem i drugim infrastrukturnim komponentama prilikom razvoja aplikacija. Pisanjem infrastrukture putem koda dobija se šablon za konfigurisanje iste, koji može da se iskoristi za replikaciju na nova okruženja u roku od nekoliko minuta [8].

5.1 Prednosti IaC-a

Prednosti IaC-a se ogledaju u sledećim stavkama [8]:

- Smanjenje troškova: Obezbeđivanje infrastrukture je nekada zahtevalo dosta vremena i bilo je skupo. Sa IaC-om, moguće je automatski upravljati infrastrukturom, što smanjuje troškove i potrebu za ručnim intervencijama.
- Povećanje brzine implementacija: IaC omogućava brže i češće izmene infrastrukture. Ona može biti brže kreirana, skalirana i uklonjena, što podržava agilni razvoj softvera i brze iteracije.
- Smanjenje grešaka: Korišćenjem IaC-a, minimiziraju se greške koje se mogu pojaviti pri ručnom konfigurisanju infrastrukture. Konfiguracija je dosledna i tačno definisana u kodu, što eliminiše slučajne greške i smanjuje verovatnoću ljudskih grešaka.
- Pобољшanje doslednosti infrastrukture: IaC obezbeđuje doslednost u konfiguraciji infrastrukture na svim okruženjima. Infrastrukturni resursi se konfiguriraju na isti način u različitim okruženjima, što pomaže u održavanju konzistentnosti i smanjuje razlike u konfiguraciji.
- Eliminisanje odstupanja u konfiguraciji: IaC osigurava da infrastruktura bude u skladu sa definisanim specifikacijama. Svako odstupanje od željenog stanja može biti automatski ispravljeno, što eliminiše konfiguracione promene koje nisu dokumentovane i nepredviđene probleme.

5.2 Terraform

Alati za infrastrukturu kao kod (IaC) omogućavaju upravljanje infrastrukturom putem konfiguracionih fajlova, umesto putem grafičkog korisničkog interfejsa. IaC omogućava postavljanje i upravljanje infrastrukturom na siguran, dosledan i ponovljiv način, definišući konfiguracije resursa koje se mogu verzionisati, ponovo koristiti i deliti [9].

Terraform je alat za infrastrukturu kao kod kompanije HashiCorp. Omogućava definisanje resursa i infrastrukture u ljudski čitljivim, deklarativnim konfiguracionim fajlovima i upravlja životnim ciklusom infrastrukture.

Korišćenje Terraforma ima nekoliko prednosti u odnosu na ručno upravljanje infrastrukturom [9]:

- Terraform može upravljati infrastrukturom na više platformi.
- Jezik konfiguracije, koji je lako čitljiv, omogućava brzo pisanje infrastrukturnog koda.
- Stanje (state) Terraform-a omogućava praćenje promena resursa tokom implementacije.
- Konfiguracije se mogu čuvati u sistemu za verzionisanje, kako bi bilo omogućeno bezbedno sarađivanje na infrastrukturi.

Za implementaciju infrastrukture putem Terraform-a, neophodni su sledeći koraci:

1. Definicija obima (Scope) - Identifikacija projektna infrastrukture.
2. Kodiranje (Author) - Pisanje konfiguracije infrastrukture.
3. Inicijalizacija (Initialize) - Instaliranje alata koje Terraform koristi za upravljanje infrastrukturom.
4. Planiranje (Plan) - Pregledanje promena koje će Terraform napraviti da bi se uskladila konfiguracija.
5. Implementacija (Apply) - Sprovođenje planiranih promena.
6. Uništavanje (Destroy) - Brisanje svih resursa.

6. CI/CD

CI/CD (Continuous Integration/Continuous Delivery) pipeline automatizuje proces isporuke softvera. Pipeline obuhvata proces izgradnje (build-a) koda, izvršavanja testova (CI) i objavljivanje (deployment) nove verzije aplikacije (CD) [10].

Automatizovani proces doprinosi eliminaciji grešaka, koje bi mogle da se potkradu prilikom manualnog izvršavanja procesa, pruža češće povratne informacije razvojnim timovima i omogućava kvalitetniju isporuku proizvoda [10].

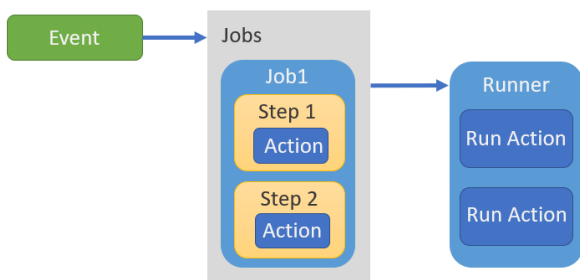
Implementacija CI/CD pipeline-a omogućava automatsku izgradnju, testiranje i isporuku softvera, što značajno povećava efikasnost razvoja.

Ovaj pristup eliminiše potrebu za ručnim koracima, smanjuje greške koje mogu nastati usled prisustva ljudskog faktora i omogućava bržu isporuku softvera. Time se ne samo poboljšava produktivnost, već i obezbeđuje kvalitetniji softver i kraće vreme do izlaska na tržište (time to market) [10].

6.1 GitHub akcije

GitHub Actions je platforma za kontinuiranu integraciju i isporuku (CI/CD) koja omogućava automatizaciju izgradnje, testiranja i isporuke aplikacija. Korisnici mogu kreirati radne tokove koji će se automatski pokretati pri određenim događajima u repozitorijumu, kao što su otvaranje "issue"-a ili "pull request"-a. Ova platforma nudi virtualne mašine za izvršavanje radnih tokova, a takođe omogućava i korišćenje sopstvenih mašina (self-hosted runners) u korisničkoj infrastrukturi [11].

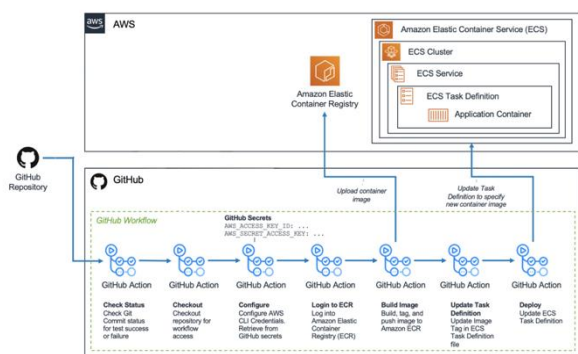
GitHub Actions se sastoji od nekoliko ključnih komponenti. Radni tokovi se konfigurišu da se pokreću na osnovu određenih događaja u repozitorijumu. Oni sadrže jedan ili više poslova (jobs), koji se mogu izvršavati sekvencijalno ili paralelno. Svaki posao se izvršava unutar sopstvenog virtuelnog mašinskog izvršioca (runner) ili kontejnera i sastoji se od više koraka (steps), kao što je prikazano na slici 3. Koraci mogu izvršavati prilagođene skripte ili akcije (actions), koje su ponovljivi dodaci koji pojednostavljaju proces radnog toka [11].



Slika 3. Struktura GitHub akcije [12]

6.1.1 Radni tok sistema za plaćanje

Sledeći dijagram (slika 4) predstavlja pregled osnovnih komponenti radnog toka aplikacije. Ova arhitektura predstavlja potpuni CI/CD proces koji koristi GitHub radni tok za automatsku koordinaciju izgradnje, testiranja i isporuke aplikacije na ECS (*Elastic Container Service*) za svaku izmenu u repozitorijumu.



Slika 4. Referentni radni tok [13]

7. ZAKLJUČAK

U radu je opisan sistem za online plaćane karticama, odnosno, simulacija bankarskog sistema za plaćanje. Softver se kao servis integriše sa aplikacijama koje žele da naplate svoje proizvode putem online transakcije. Omogućuje brzo i jednostavno monetizovanje bilo koje aplikacije.

Istražena je integracija lokalnih (on-premise) mašina sa AWS ekosistemom putem ECS Anywhere servisa. Opisana je registracija mašina i konfiguracija ove postavke, uz rezultate i prednosti datog servisa.

Naglasak je stavljen na mehanizme raspoređivanja i infrastrukturu sistema. Opisani su CI/CD radni tokovi, koji su implementirani korišćenjem GitHub akcija, kao i AWS servisi koji su korišteni za hosting aplikacija. Infrastruktura je napisana putem Terraform IaC alata. Detaljno je dokumentovana konfiguracija celog sistema.

Integracija koncentratora plaćanja sa pravim bankama bi bio prvi sledeći korak u daljem razvoju sistema. Dodavanje novih metoda plaćanja bi takođe moglo da bude jedno od unapređenja.

8. LITERATURA

- [1] <https://docs.aws.amazon.com/iam/index.html>
- [2] <https://docs.aws.amazon.com/vpc/>
- [3] <https://docs.aws.amazon.com/ec2/>
- [4] <https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html>
- [5] <https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs-anywhere.html>
- [6] <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html>
- [7] <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/Welcome.html>
- [8] <https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac>
- [9] <https://developer.hashicorp.com/terraform/tutorials/aws-get-started/infrastructure-as-code>
- [10] <https://semaphoreci.com/blog/cicd-pipeline#what-is-a-ci-cd-pipeline>
- [11] <https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions>
- [12] <https://ithelp.ithome.com.tw/articles/10262377>
- [13] <https://aws.amazon.com/blogs/containers/create-a-ci-cd-pipeline-for-amazon-ecs-with-github-actions-and-aws-codebuild-tests/>

Kratka biografija:



Monika Erdeg rođena je u Novom Sadu 1995. god. Fakultet tehničkih nauka u Novom Sadu upisala je 2014. godine. Završila je osnovne akademske studije 2018. godine, na smeru Softversko inženjerstvo i informacione tehnologije.

kontakt: monika.erdeg@gmail.com

FIZIČKI MODEL SISTEMA ZA AKTIVNO POTISKIVANJE BUKE

PHYSICAL MODEL OF AN ACTIVE NOISE CANCELLATION SYSTEM

Sava Njegovan, *Fakultet tehničkih nauka, Novi Sad*

Oblast – DIGITALNA OBRADA SIGNALA

Kratak sadržaj – *Ovaj rad istražuje mogućnost aktivnog potiskivanja buke (ANC) putem primene digitalne obrade signala u kombinaciji sa feedforward upravljanjem koristeći adaptivni filter. Eksperimentalna postavka simulira klimatizacioni sistem, dok se implementirani algoritam fokusira na generisanje signala za potiskivanje periodične buke koju je potrebno prethodno identifikovati.*

Ključne reči: ANC sistemi, Digitalna obrada signala, Adaptivni filteri

Abstract – *This paper explores the possibility of active noise cancellation (ANC) through the application of digital signal processing in combination with feedforward control using an adaptive filter. The experimental setup simulates an air conditioning system, while the implemented algorithm focuses on generating a signal for attenuating periodic noise that needs to be identified beforehand.*

Keywords: ANC systems, Digital Signal Processing, Adaptive filters

1. UVOD

Koncept aktivnog potiskivanja buke (ANC - Active Noise Control) datira još iz 1930-ih godina, ali zbog tehničkih ograničenja, praktična primena ovih sistema nije bila moguća sve do devedesetih godina prošlog veka [1]. Upravo u tom periodu, razvoj kontrolnih algoritama i unapređenje računarske moći digitalnih signalnih procesora omogućili su brz razvoj ovih sistema. Danas se ANC sistemi široko primenjuju i postali su neizostavan deo naše svakodnevnice, prisutni u raznim oblastima i na tržištu [2].

U okviru ovog istraživanja, glavni fokus usmeren je na eksperimentalnoj verifikaciji sposobnosti ANC sistema da potisne buku putem primene tehnika digitalne obrade signala (DSP) i korišćenja adaptivnog filtera pri upravljanju. Eksperimentalna postavka prikazana u ovom radu simulira klimatizacioni sistem, čest izvor buke u prostorima gde ljudi borave, a implementirani algoritam je namenjen za jednokanalne sisteme sa upravljanjem u otvorenoj sprezi. Cilj algoritma jeste identifikovati neželjenu buku i generisati signal koji će ju potisnuti ili u idealnom slučaju potpuno eliminisati.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Nikola Jorgovanović, red. prof.

2. METOD

U realizovanom fizičkom sistemu klimatizacionu cev simulira PVC cev dužine 101 cm i prečnika 10,5 cm, dok buku koju proizvodi ventilator oponaša signal koji se generiše na zvučniku. Zvučnik je postavljen na jednom kraju cevi, dok je na drugom kraju cevi potrebno potisnuti buku.

Vrednost frekvencije odabiranja postavljena je na $f_s=8\text{kHz}$, uprkos činjenici da se za audio sisteme najčešće koristi vrednost od 44.1kHz. Frekvencija odabiranja je namerno redukovana kako bi se smanjili računarski zahtevi, s obzirom da se algoritmi za potiskivanje buke realizuju na standardnom PC računaru.

2.1. Oprema za realizaciju ANC sistema

Opremu čine dva mikrofona: RODE NT-USB Mini (Referentni mikrofoni) i Blue Yeti (Mikrofon greške). Oba mikrofona su sa računarom povezani putem USB kablova. Zvučnik Logitech Z120 je korišten kao ANC zvučnik, odnosno za generisanje signala za potiskivanje buke.

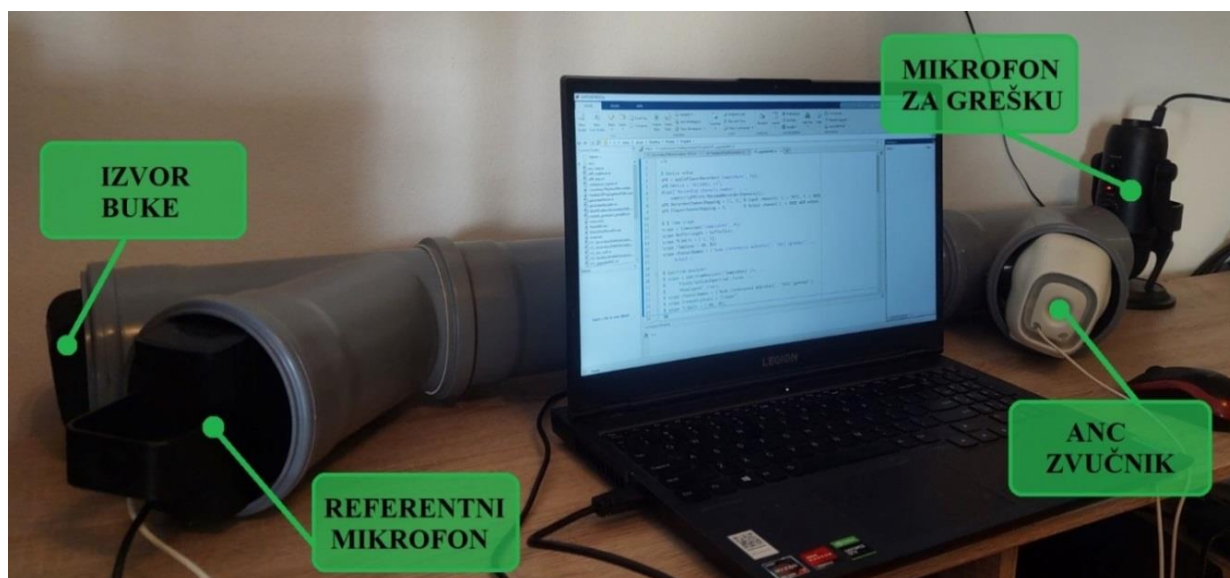
2.1. Postavka eksperimenta

Buka na izvoru se snima referentnim mikrofonom, dok se buka na izlazu sistema, tj. na drugom kraju cevi snima mikrofonom za grešku. Cilj algoritma je generisanje zvuka na ANC zvučniku koji će potisnuti buku na mestu gde je postavljen mikrofoni za grešku koji samim tim daje informaciju i o efektivnosti algoritma. Postavka sistema je prikazana na slici 1.

Softversko rešenje je implementirano u MATLAB okruženju, a za identifikaciju sistema i generisanje signala koji treba da potisne buku korišten je *FxLMS* algoritam [2].

2.3. FxLMS algoritam

FxLMS (Filtered-x LMS) algoritam je poboljšana verzija klasičnog LMS (Least Mean Square) algoritma, posebno prilagođena za primenu u ANC sistemima. Oba ova algoritma se koriste za obradu signala i koriste LMS adaptivni filter. LMS filter prilagođava svoje koeficijente u realnom vremenu kako bi smanjio kvadratnu grešku između željenog i izlaznog signala. Na ulaz ovog filtera se dovodi ulazni signal koji treba obraditi. Pored ulaznog signala LMS algoritam zahteva i referentni tj. željeni signal. LMS filter primenjuje koeficijente na ulazni signal kako bi generisao izlazni signal. Ovaj izlazni signal se zatim upoređuje sa referentnim signalom kako bi se izračunala greška.



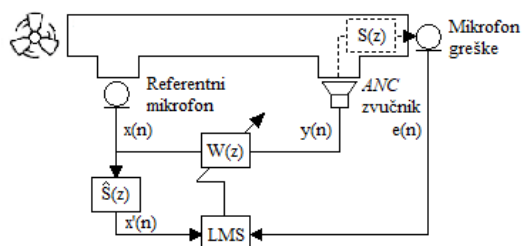
Slika 1. Fizički model ANC sistema

Ključna karakteristika *LMS* algoritma je njegova sposobnost da se adaptira. Na osnovu izračunate greške, algoritam koristi gradijentni pristup kako bi prilagodio koeficijente filtera. Koraci obrade signala, izračunavanja greške i prilagođavanja koeficijenata se ponavljaju iterativno u realnom vremenu. Algoritam se kontinuirano prilagođava kako bi smanjio grešku i približio izlazni signal referentnom signalu.

FxLMS algoritam je jedna od najčešćih tehnika adaptivnog filtriranja koja se koristi za realizaciju ANC sistema [3]. Pored digitalnog adaptivnog *LMS* filtera $W(z)$ čiji se koeficijenti računaju u realnom vremenu tokom rada sistema, ovaj algoritam uzima u obzir postojanje akustičnog odziva koji obuhvata prostiranje zvuka od ANC zvučnika do mikrofona za grešku. Upravo uzimanje u obzir ovog sistema, koji je na slici 2 označen sa $S(z)$, predstavlja unapređenje *FxLMS* u odnosu na *LMS* algoritam. U *embedded* sistemima koji se koriste za ANC, sistem $S(z)$ pored akustičnog odziva od zvučnika do mikrofona greške obuhvata hardverske komponente kao što su *D/A* konvertor, audio pojačavač, zvučnik, mikروفon greške, anti-aliasing filter i *A/D* konvertor [2]. U odnosu na *embedded* sistem koji je namenjen za rad u realnom vremenu, kod standardnog *PC* računara hardverska arhitektura je znatno kompleksnija, a operativni sistemi nisu optimizovani za brzu obradu. Fizički model realizovan u ovom radu, sa softverskim rešenjem implementiranim u *MATLAB*-u na *PC* računaru, iz navedenih razloga ograničava performanse ANC sistema i prouzrokuje veće kašnjenje što može znatno uticati na efikasnost algoritma.

Da bi se *FxLMS* algoritam mogao implementirati prethodno je neophodno odraditi *offline* identifikaciju sistema $S(z)$ i u ovom radu je za njegovu estimaciju korišten *FIR* filter. Identifikacija ovog sistema se vrši pomoću prethodno navedenog *LMS* adaptivnog filtera koji iz iteracije u iteraciju izračunava procenjene koeficijente filtera sve dok se koeficijenti ne ustale. Proces identifikacije pomoću *LMS* adaptivnog filtera se vrši generisanjem belog šuma sa ANC zvučnika. Generisani beli šum predstavlja ulazni signal u filter i za

identifikaciju se koristi zato što ima konstantu spektralnu gustinu na svim frekvencijama [4]. Za referentni signal se koristi signal snimljen sa mikrofona greške, a cilj je minimizovati grešku tj. razliku izlaznog signala iz estimiranog filtera i referentnog signala. U 3. odeljku prikazani su rezultati verifikacije identifikacije sistema $S(z)$ putem upoređivanja impulsnog odziva stvarnog i procenjenog sistema. Kada je proces identifikacije sistema $S(z)$ završen, dobijeni koeficijenti filtera $\hat{S}(z)$ se čuvaju kako bi *FxLMS* algoritam bio spreman za primenu.

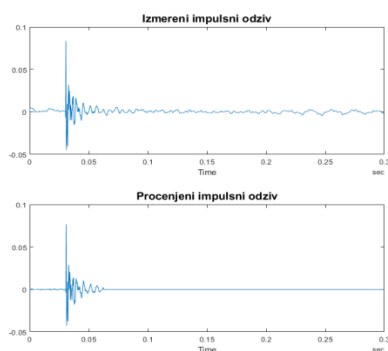


Slika 2. *FxLMS* algoritam

3. REZULTATI RADA

Pre nego što je aktiviran ANC sistem, sproveden je proces *offline* identifikacije sistema $S(z)$. Radi provere tačnosti ovog procesa identifikacije, izvršeno je poređenje impulsnog odziva stvarnog sistema sa impulsnim odzivom procenjenog filtera $\hat{S}(z)$. Rezultati dobijeni tokom ovog procesa verifikacije sistema $S(z)$ prikazani su na slici 3.

Ventilatori u klimatizacionim sistemima su izvor periodične buke, pa je i ovaj model testiran nad takvim signalima. ANC sistem počinje da podešava koeficijente adaptivnog filtera nakon 3,5 sekunde od puštanja buke i pokretanja programa. Pored toga, nakon što algoritam konvergira, tj. podesi koeficijente filtera tako da se buka potisne, iterativno ažuriranje koeficijenata se zaustavlja i koeficijenti filtera se zaključavaju. Trenutak zaključavanja koeficijenata u implementiranom algoritmu zavisi od vrednosti signala greške, a optimalna vrednost je izabrana nakon više testiranja.



Slika 3. Poređenje impulsnih odziva sistema $S(z)$ i $\hat{S}(z)$

Na slici 4 prikazan je rezultat potiskivanja periodične buke na 800Hz u vremenskom domenu. Plavom bojom je prikazan signal sa referentnog mikrofona, a crvenom signal koji registruje mikrofona za grešku. Ovo je ujedno i jedan od najboljih rezultata koji su dobijeni pri testiranju.

Na slici 5(a) prikazan je amplitudski spektar signala pre potiskivanja, dok je na slici 5(b) prikazan amplitudski spektar signala posle potiskivanja. Crvenom bojom je označen spektar signala sa mikrofona greške.

Uočava se da je komponenta na frekvenciji 800 Hz smanjena sa -11 dB na -37 dB nakon primene potiskivanja.

4. DISKUSIJA

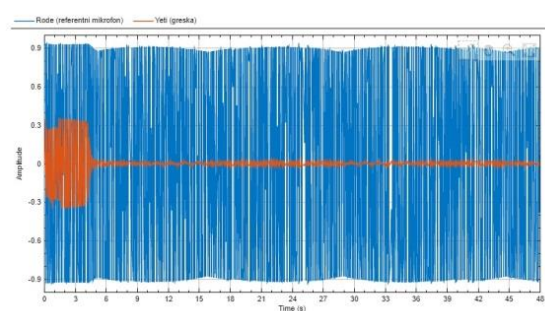
Testiranje sistema za aktivno potiskivanje buke na periodičnim signalima pokazalo je odlične rezultate za određene periodične signale. Sistem je efikasno smanjio prisutnost buke u tim uslovima, potvrđujući teorijsku osnovu sistema. Naredni korak istraživanja treba da se usredsredi na proširenje funkcionalnosti sistema kako bi se omogućila uspešna implementacija i testiranje na aperiodičnim signalima. Ovakav pristup će omogućiti sistem da efikasno reaguje na širok spektar signala.

Tokom implementacije, identifikovane su određene prepreke koje su uticale na performanse sistema. Upotreba opreme koja nije prvobitno namenjena za ANC prouzrokovala je povećano kašnjenje i ograničenja u prilagođavanju parametara sistema. Jedan od primetnih problema tokom analize rezultata je bila prisutnost distorzija harmonika što se i vidi na spektru signala prikazanom na slici 5.

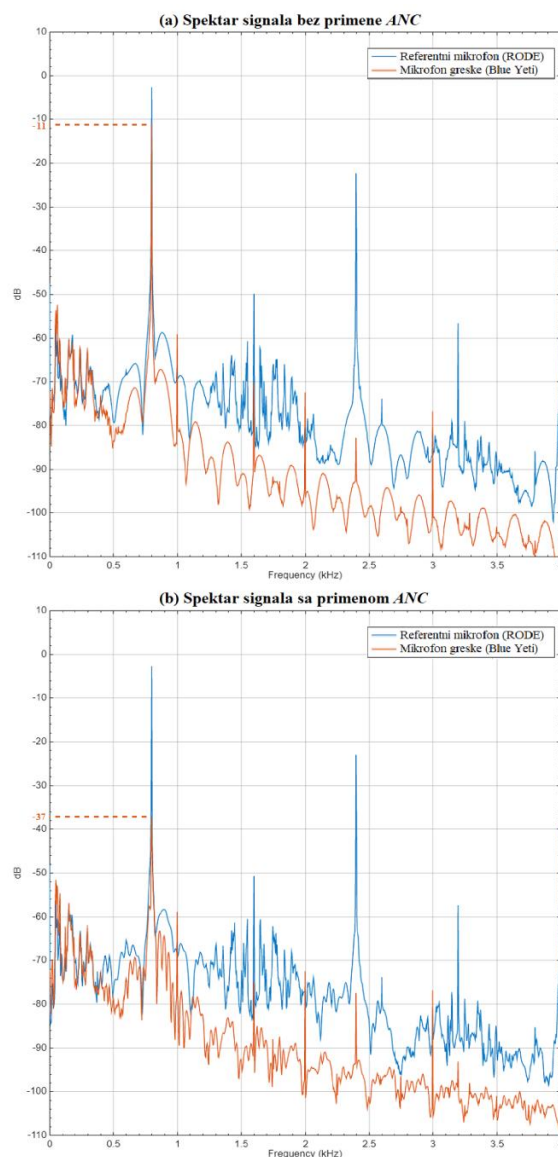
Ovo ističe važnost kvalitetne i adekvatne opreme za uspešno funkcionisanje ANC sistema, kao i potrebu za dodatnim istraživanjem kako bi se otklonili ovi problemi i poboljšala celokupna efikasnost sistema.

Što se tiče $FXLMS$ algoritma koji je korišćen za adaptivno upravljanje u ANC sistemu, njegova primena se pokazala adekvatnom.

Međutim, dalja optimizacija i prilagođavanje ovog algoritma može doneti dodatne koristi u pogledu performansi i stabilnosti sistema.



Slika 4. Rezultat potiskivanja periodičnog signala sa komponentom na 800Hz



Slika 5. Spektar signala bez i sa primenom ANC-a

5. ZAKLJUČAK

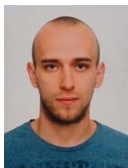
Cilj ovog rada je bilo pokazati mogućnost potiskivanja buke primenom *feedforward FxLMS* algoritma na realnom fizičkom modelu što je i postignuto. Ovaj master rad predstavlja nastavak istraživanja iz diplomskog rada koji se fokusirao na teorijskom razumevanju i simulaciji sistema za aktivno potiskivanje buke.

Pokazano je da praktična implementacija donosi sa sobom izazove koji se ne pojavljuju tokom simulacija, te je tokom ovog rada bilo potrebno razviti rešenja za realne probleme i tehničke prepreke.

6. LITERATURA

- [1] C.H.Hansen, "*Understanding Active Noise Cancellation*", London, Spon Press, 2001.
- [2] S.M.Kuo, "*Active Noise Control: A Tutorial Review*", Proceedings of the IEEE, vol. 87, no. 6, pp. 943-973, June 1999.
- [3] K.-C.Chen, C.-Y.Chang, S.M.Kuo, "*Active noise control in a duct to cancel broadband noise*", IOP Conference Series: Materials Science and Engineering, vol. 237, no. 1, 2017.
- [4] S.M.Kuo, "*Adaptive active noise control systems: algorithms and digital signal processing (DSP) implementations*", Wiley-Interscience; 1st edition (February 8, 1996)

Kratka biografija:



Sava Njegovan rođen je u Prnjavoru 1999. god. Master rad na Fakultetu tehničkih nauka iz oblasti Automatika i upravljanje sistemima – Biomedicinsko inženjerstvo odbranio je 2023.god.
kontakt: savanjegovan9@gmail.com

**ПРЕДИКЦИЈА ПОРАСТА НИВОА МОРА КОРИШЋЕЊЕМ АЛГОРИТАМА
МАШИНСКОГ УЧЕЊА****PREDICTION OF SEA LEVEL RISE USING MACHINE LEARNING ALGORITHMS**Стефан Савић, Јелена Сливка, *Факултет техничких наука, Нови Сад***Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО**

Кратак садржај – Због утицаја које има на планету и на људске животе, глобално загревање је тема која је постала веома значајна у последњих неколико деценија. Пораст нивоа мора представља једну од најозбиљнијих последица глобалног загревања. Главни циљ овог рада је његова предикција. Најпре је вршено прикупљање релевантних података из различитих извора. Након тога, вршено је претпроцесирање и формирање коначног скупа података који представља улаз у систем. Над претпроцесираним подацима тренинг скупа извршено је обучавање више модела међу које спада метод потпорних вектора, Наивни Бајесов модел, метод случајне шуме, Bagging и XGBoost, као и модели временских серија. Сваки од модела је за излаз имао вероватноћу да ли се десило пораст нивоа мора у односу на претходни месец, док су временске серије предвиђале тачне вредности средњег глобалног нивоа мора. Изведен је закључак да главни утицај на пораст нивоа мора представља емисија штетних гасова тј. угљен-диоксида и температуре, како на површини воде, тако и на копну.

Кључне речи: GMSL; глобално загревање; ниво мора; XGBoost; временске серије

Abstract – Due to the impact it has on the planet and human lives, global warming is a topic that has become highly significant in the last few decades. Rising sea levels represent one of the most serious consequences of global warming. The main objective of this study is its prediction. First, relevant data was collected from various sources and preprocessed. Multiple models were trained on the preprocessed training set - Support Vector Machine, Naive Bayes model, Random Forest, Bagging, XGBoost, and time series models. Each model outputs the probability of a sea level rise compared to the previous month, while the time series predicted the exact values of the average global sea level occurring. The conclusion is that the primary influence on the sea level rise is the emission of harmful gases (i.e., carbon dioxide) and temperature on the water surface and land.

Keywords: GMSL; global warming; sea level; XGBoost; time series

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији ментор је била др Јелена Сливка, ванр. проф.

1. УВОД

Глобално загревање, тј., постепени пораст просечне температуре површине Земље, тема је која је постала веома значајна у последњих неколико деценија због утицаја које има на планету и на наше животе. Оно представља озбиљан изазов за човечанство, са потенцијалом да проузрокује низ штетних последица. Једна од најозбиљнијих последица је пораст нивоа мора услед топљења леда са полова и глечера. Ово може имати озбиљне ефекте на приобална подручја и острва, која су изложена ризику од ерозије и поплава.

Проучавање пораста нивоа мора и анализа фактора који томе доприносе од суштинског су значаја за одрживи развој приобалних заједница и за будућност човечанства. Додатни мотив за бављење овим проблемом је и жеља да се спрече штетне последице по људску економију, друштво и здравље, као и потреба да се очува природа и биодиверзитет.

У овом раду представљена су два решења за одређивање пораста нивоа мора:

1. Први проблем је постављен као проблем бинарне класификације – циљ је био предвидети да ли се десило повећање нивоа мора у односу на претходни месец или не. Такође, одређивани су фактори који су највише допринели повећању нивоа мора. Овај проблем решаван је обуком различитих модела машинског учења (метод потпорних вектора, Наивни Бајесов модел, метод случајне шуме, XGBoost и Bagging), при чему су као улазни подаци коришћени фактори који утичу на глобално загревање.
2. Други решавани проблем је предикција тачне промене средњег глобалног нивоа мора (GMSL). Овај проблем је решаван применом временских серија са кораком од месец дана.

Као најбољи модел за први проблем показао се XGBoost, где је вредност F мере за предикцију да се није десило повећање нивоа мора у односу на претходни месец, износила 0.52, а да јесте 0.62. Показало се да највећи утицај на пораст нивоа мора имају кисеоник, који је саставни део воде, нитрат, као и температура копна.

Примена модела временских серија се показала као успешан приступ у добијању жељеног предвиђања нивоа мора. Након предвиђања тачне вредности нивоа мора, поред временских корака, фактори који највише доприносе крајњим резултатима јесу дебљина леда, температура и молекули кисеоника у води, што је

слично показао и *XGBoost* модел. Уколико би се поседовали подаци на дневном, а не месечном нивоу, претпоставка је да би ова предвиђања постала тачнија. У наредном поглављу, биће приказана сродна истраживања. Затим, у трећем поглављу, биће представљена методологија коришћена при решавању овог проблема, као и спроведени експерименти. Након тога, четврто поглавље ће чинити резултати и дискусија, док ће у петом поглављу бити изведен закључак о самом раду и његовом проблему.

2. ПРЕТХОДНА РЕШЕЊА

У свом раду [1], Балогун и Адебиси интегрисали су широк спектар океанско-атмосферских варијабли ради предвиђања варијација нивоа мора помоћу неуронских мрежа. Компарација модела обучених на океанским и атмосферским варијаблама показује да атмосферски процеси имају већи утицај на предвиђање модела. Ипак, боља предвиђања модела добијају се комбинацијом океанско-атмосферских варијабли. Атмосферски подаци који се користе укључују падавине, облачност и брзину ветра, док подаци о мору укључују температуру, салинитет и густину површине мора. С обзиром на релевантност варијабли и експерименталну поставку, рад [1] се показао као значајан за ово истраживање.

Зхенг [2] је проучавао однос између температуре и других потенцијалних фактора попут CO₂, N₂O, CH₄. Користио је конструкцију статистичких модела базираних на великој количини климатских података и многобројне алгоритме машинског учења као што су: метод случајне шуме, линеарна и регресија вектора подршке и LASSO. Линеарна интерполација коришћена је за усклађивање и ефикаснију обраду података. Током процеса обуке, коришћена је унакрсна валидација за тражење одговарајућих хипер-параметара модела. Потом, упоређена су три различита алгоритма машинског учења (метод случајне шуме, LASSO и SVR). Метод случајне шуме се издвојио као најпрецизнији и он сугерише да највећи утицај на промену температуре има CO₂. По узору на рад [2], одлучено је да један од атрибута буде управо CO₂, а метод случајне шуме један од коришћених алгоритама. У складу са препорукама аутора, у овом раду примењен је и *XGBoost*.

С обзиром да напредне статистичке анализе, укључујући методе машинског учења, дају значајан увид у промену нивоа мора, узет је рад Ниевеса и сарадника [3] као релевантан за ово истраживање. Уз помоћ машинског учења, главни циљ је одредити склоност ка повећању и смањењу нивоа мора у годинама које предстоје. Примењен је модел регресије Гаусовог процеса (GP) и рекурентна неуронска мрежа са јединицама дуготрајне меморије. Вршена је анализа утицаја отвореног океана на обалу, уз промену просечне температуре до 700 метара дубине. Аутори су предложили да би скуп података требало проширити и другим параметрима, као што је промена дебљине леда, салинитет и утицај глобалног загревања на површинске температуре. Сходно томе, наведени предлози су узети у обзир и у овом раду.

Рад [4] разматра примену техника машинског учења на проблеме регресије који се најчешће уочавају при анализи временских серија података. Регионалне варијације у порасту нивоа мора су знатне и требале би се узети у разматрање при планирању будућег пораста нивоа мора. Из тог разлога, примењени су историјски подаци о нивоу мора са мерача плиме и осеке који се налазе на многобројним местима распрострањеним дуж шведске обале. Методе машинског учења које су коришћене представљају три различите вештачке неуронске мреже и вишеструку линеарну регресију и оне су универзалне, те се могу лако применити и на податке са других локација. Доказано је да су свеукупне перформансе ових алгоритама машинског учења задовољавајуће, често и боље од перформанси много скупљих нумеричких модела океана.

3. МЕТОД

У наредним поглављима описан је начин добијања коначног скупа података, начин на који су модели креирани и експерименти над испробаним моделима за предикцију.

3.1. Скуп података

С обзиром да су фактори који утичу на глобално загревање разнолики, било је потребно прикупити податке из различитих извора. У циљу добијања прецизнијих резултата, сваки од прикупљених скупова података најпре је детаљно анализиран и процесиран. Услед великог броја података, уочено је да постоји значајан број оних који су невалидни, непотпуни или захтевају додатну обраду.

Након завршеног претпроцесирања, спајањем претходно добијених скупова података и издвајањем релевантних атрибута, формиран је коначан скуп. Подаци су груписани на основу датума мерења на месечном нивоу, док *GMSL* представља циљно обележје на основу кога је формирана класа - *IsGMSLIncreased*. Поред ових података, као релевантни атрибути издвојили су се и:

- *Extent* - укупна површина морског леда изражена у јединици 10⁶ km²,
- *WaterTemp* - температура воде у °C,
- *O2ml* - zasiћеност воде кисеоником,
- *SiO3* - концентрација силиката,
- *NO3* - концентрација нитрата,
- *LandAvgTemp* - просечна температура,
- *LandAndOceanAvgTemp* - просечна температура копна и мора за дан мерења и
- *CO2* - просечна концентрација угљен-диоксида на месечном нивоу.

3.2. Генерисање модела за предикцију утицаја глобалног загревања на пораста нивоа мора

Циљ овог рада огледа се у проналажењу модела који дају најбоље резултате за дати проблем. Ради њиховог проналажења, за предвиђање су коришћени сви алгоритми споменути у релевантној литератури.

Прво су испробани Наивни Бајес и метод потпорних вектора, а затим метод случајне шуме и *Bagging* са и

без оптимизације хипер-параметара. Највише времена посвећено је *XGBoost* моделу и његовој оптимизацији.

Као улаз у систем коришћен је претходно претпроцесиран скуп података. С друге стране, излаз из система чинили су обучени модели на основу којих се закључује да ли се десило повећање нивоа мора.

Код обучавања модела временских серија, улаз у систем састојао се такође од претпроцесираних података коначног скупа. За разлику од претходно обучених модела, излаз из овог система представља обучен модел који пружа информацију о тачном повећању средњег глобалног нивоа мора - *GMSL*. Модел врши предвиђање будућих тачака података на основу историјских података, а као временски корак узет је интервал од месец дана.

3.3. Подешавање хипер-параметара модела машинског учења

Ради постизања најбољих перформанси модела над коначним скупом података, било је неопходно прецизно подесити вредности хипер-параметара. Процес подешавања хипер-параметара укључује испробавање њихових различитих вредности, обуку модела за сваки од њих и избор најбољих вредности.

Најпре, испробан је Наивни Бајесов модел. С обзиром да је релативно једноставан класификациони алгоритам, он нема велик број хипер-параметара у поређењу са сложенијим алгоритмима, па самим тим за овај модел није вршено додатно подешавање.

Затим, за метод потпорних вектора подешен је само тип функције језгра - *RBF*. Она је корисна у ситуацијама када је однос између карактеристика и ознака класа сложен и не може се ефикасно одвојити линеарном хипер-равном у оригиналном простору обележја. Самим тим, њеним избором омогућене су флексибилније и нелинеарне границе одлучивања.

Следећи испробан алгоритам био је метод случајне шуме - *RandomForestClassifier*. Прво је тестиран без подешавања хипер-параметара, а затим су испробане њихове различите комбинације како би открило који од њих дају најбоље резултате. Ради проналажења оптималних вредности хипер-параметара, коришћен је *RandomizedSearchCV*. Модел је пружао најбоље вредности када је *n_estimators* подешен на 100, *max_features* на *sqrt* и *max_depth* на 10.

Након тога, испробан је и *BaggingClassifier*. Оптимизација хипер-параметара вршена је истраживањем различитих вредности и упоређивањем њихових резултата. Прво је као базни естиматор дефинисан *DecisionTreeClassifier*, а у случају када је *n_estimators* подешен на 95, *max_samples* на 0.9 и *random_state* на 9, *BaggingClassifier*-а је давао најбоље резултате.

На крају, испробан је и *XGBoost*. За проналажење оптималних вредности параметара коришћен је *RandomizedSearchCV*. За естиматор је подешен *XGBRegressor*, док је сваком од хипер-параметара додељен низ могућих вредности. Оптимални хипер-параметри уз које *XGBoost* даје најбоље предикције постигнути су подешавањем: *max_depth* на 3, *min_child_weight* на 6, *learning_rate* на 0.01, *eta* на 2,

subsample на 1, *colsample_bytree* на 0.8, *objective* на *binary:logistic* и *eval_metric* на *mae*.

Додатно, како би се предвидео тачан пораст глобалне средње вредности нивоа мора, испробан је модел временских серија.

За предвиђање је коришћен *ForecasterAutoreg*, при чему је за регресор одабран *XGBRegressor*. Емпијским путем изведен је закључак да предвиђање са кораком 8 даје најбоље резултате. У ситуацији када је *n_estimators* подешен на 500, *max_depth* на 10, *learning_rate* на 0.1 и *lags* на низ вредности од 1 до 10, модел је пружао најпрецизније предикције.

4. РЕЗУЛТАТИ И ДИСКУСИЈА

Наивни Бајес и метод потпорних вектора били су први тренирани модели. Њихова евалуација показала је идентичан учинак тачности, а он је износио 54%. Резултати су приказани у табели 4.1, где 0 показује да се пораст нивоа мора није десило, а 1 да јесте.

За оба наведена случаја, израчуната је прецизност, одзив и *F* мера. Из добијених резултата може се уочити да, услед глобалног раста нивоа мора, модели боље предвиђају повећање нивоа мора у односу на претходни месец него опадање.

Резултати	Прецизност	Одзив	F мера
0	0.47	0.41	0.4
1	0.60	0.65	0.6

Табела 4.1. Резултати евалуације за Наивног Бајеса и SVM.

У *XGBoost* је следећи тренирани модел чија је основна средња апсолутна грешка износила 50%. Грешка је смањена за 2% при првој итерацији модела и то без оптимизације параметара, али добијени резултати су прилично лошији у односу на претходна два модела.

Оптимизација хипер-параметара модела, допринела је побољшању предикције пораста нивоа мора и то за 10%. Ови резултати представљени су у табели 4.2.

Резултати	Прецизност	Одзив	F мера
0	0.51	0.53	0.52
1	0.64	0.61	0.62

Табела 4.2. Резултати евалуације за *XGBoost* са оптимизацијом хипер-параметара.

Уз помоћ *XGBoost*-а, закључено је да највећи утицај на пораст нивоа мора имају кисеоник, који је саставни део воде, нитрата (NO_3), као и температура копна.

Метод случајне шуме без оптимизације хипер-параметара није допринео бољим резултатима. Међутим, након њиховог подешавања, добијено је најбоље предвиђање за ситуацију када се посматра смањење нивоа мора и то је приказано у табели 4.3.

Резултати	Прецизност	Одзив	F мера
0	0.60	0.53	0.57
1	0.62	0.68	0.65

Табела 4.3. Резултати евалуације за метод случајне шуме са оптимизацијом хипер-параметара.

Последњи модел за који су разматрани резултати предикције је *Bagging*. Метод случајне шуме и овај метод имали су идентичне резултате без оптимизације хипер-параметара, док су резултати добијени након подешавања приказани су у табели 4.4.

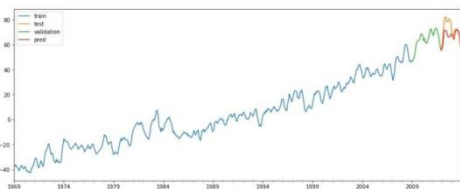
Резултати	Прецизност	Одзив	F мера
0	0.62	0.49	0.55
1	0.61	0.72	0.67

Табела 4.4. Резултати евалуације за *Bagging* са оптимизацијом хипер-параметара.

Након проучених, углавном лоших резултата свих обучених модела, изведен је закључак да сваки од њих даје приближно сличне резултате без обзира на подешавање хипер-параметара. Уз помоћ *XGBoost*-а, показано је да најзначајнији атрибути подударају са кључним факторима глобалног загревања.

Узимајући у обзир да претходни модели нису дали најбоље резултате, употребљене су временске серије које су допринеле бољим резултатима и анализама.

Фактори који највише утичу на коначне резултате су: временски кораци, дебљина леда, температура и молекули кисеоника у води. Ове вредности се скоро поклапају са оним добијеним од стране *XGBoost* модела. На слици 2, дат је приказ оригиналних и предвиђених вредности. Примећено је да модел лакше предвиђа опадање у односу на пораст нивоа мора.



Слика 2. Предикција временских серија приказана црвеном линијом.

Након анализе добијених резултата, уочено је да се нагли скок средње вредности нивоа мора догодио у години која је у односу на претходне имала рекордно високе температуре. У том периоду, модел је правио највише грешака, али тиме је уједно и потврђено да је температура најважнији фактор пораста GMSL-а. Међутим, за предикцију смањења нивоа мора, овај модел се показао као врло успешан.

5. ЗАКЉУЧАК

Услед све већег глобалног загревања и интересовања људи за ту тему, овај рад се бавио предвиђањем пораста нивоа мора.

Прикупљени су релевантни подаци из различитих извора где је сваки од скупова података имао заједнички атрибут који представља датум мерења на

месечном нивоу. Извршено је њихово претпроцесирање и од таквих података формиран је коначан скуп. Први приступ се базирао на предикцији да ли се десио пораст нивоа мора у односу на претходни месец и који фактори су томе највише допринели. Модели машинског учења употребљени у овом приступу су: Наивни Бајес, метод потпорних вектора, метод случајне шуме, *Bagging* и *XGBoost*. Над коначним скупом података, упоређене су перформансе сваког од ових модела, док је за евалуацију тачности изабрана F мера. Примећено је да је након оптимизације хипер-параметара сваки модел давао приближне резултате, док се као најбољи издвојио *XGBoost*. Други приступ предвиђа тачне вредности средњег глобалног нивоа мора употребом временских серија. Оне су поред осталих фактора који утичу на предикцију, користиле и временски корак. За разлику од првог приступа, коришћење временских серија се испоставило као прецизније и корисније за даље унапређивање у циљу добијања бољих решења.

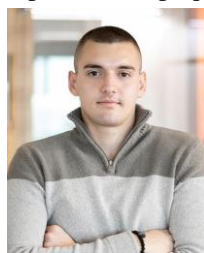
Један од недостатака овог рада лежи у ограниченом броју интегрисаних скупова података који имају различите нивое временске агрегације и не садрже константно мерене вредности. Уколико би се континуирано спроводио тренд акумулације података, претпоставка је да би модели били способнији да остваре прецизније прогнозе.

У циљу добијања бољих резултата, овај рад може бити проширен додатним факторима који утичу на прецизност модела. Такође, требало би посветити више времена анализи фактора за које модел предвиђа да највише утичу на пораст нивоа мора. Додатно, претпоставка је да би се поновним обучавање модела са подацима прикупљеним додатним мерењима и коришћењем временских серија са дневним кораком предвиђања, добили прецизнији резултати.

6. ЛИТЕРАТУРА

- [1] Al. Balogun, N. Adebisi, "Geomatics, Natural Hazards and Risk", vol. 12, 2021.
- [2] Harvey Zheng, "Analysis of Global Warming Using Machine Learning", vol. 7, no. 3 2018.
- [3] Veronica Nieves, Christina Radin, Gustau Camps-Valls, "Predicting regional coastal sea level changes with machine learning", 2021.
- [4] Magnus Hieronymos, Jenny Hieronymos, Frederik Hieronymos, "On the Application of Machine Learning Techniques to Regression Problems in Sea Level Studies", Sep. 2019.

Кратка биографија:



Стефан Савић рођен је 1998. год. у Београду. Основне академске студије завршио је 2021. год. на Факултету техничких наука, на ком брани и мастер рад 2023. године из области Електротехнике и рачунарства.

Контакт: stefan.savic98@gmail.com

U realizaciji Zbornika radova Fakulteta tehničkih nauka u toku 2022. godine učestvovali su sledeći recenzenti:

Aco Antić	Dragana	Marija Silađi	Nemanja Kašiković
Aleksandar	Konstantinović	Marinko Maslarić	Nemanja Sremčev
Anđelković	Dragana Šarac	Marko Lazić	Nemanja Tasić
Aleksandar Kovačević	Dragoljub Šević	Marko Marković	Nenad Grahovac
Aleksandar	Drago Žarković	Marko Todorov	Nenad Simeunović
Kupusinac	Duško Bekut	Marko Vekić	Nikola Vojnović
Aleksandar Ristić	Đorđe Vukelić	Maša Bukurov	Petar Mirković
Aleksandar Selakov	Goran Jeftenić	Matija Stipić	Platon Sovilj
Aleksandra Radulović	Goran Savić	Mijodrag Milošević	Radivoje Dinulović
Aleksandra Pešterac	Goran Sladić	Milan Delić	Radomir Kojić
Andraš Anderla	Goran Švenda	Milan Gavrić	Ratko Obradović
Andrija Rašeta	Goran Tepić	Milan Marinković	Sandra Dedijer
Atila Zelić	Gordan Stojić	Milan Mirković	Saša Medić
Bojan Batinić	Gordana Ostojić	Milan Rapajić	Slavica Mitrović
Bojan Matić	Igor Dejanović	Milan Rackov	Senka Bajić
Bojan Tepavčević	Igor Peško	Milan Trivunić	Slobodan Morača
Borislav Savković	Iva Šiđanin	Milan Vidaković	Slobodan Šupić
Branislav Atlagić	Ivana Mihajlović	Milena Krklješ	Srđan Popov
Branislav Stevanov	Igor Maraš	Milica Kostreš	Srđan Vukmirović
Branka Nakomčić	Ivan Prokić	Milica Miličić	Stevan Gostojić
Branko Milosavljević	Ivana Katić	Miloš Simić	Stevan Grabić
Branko Škorić	Ivana Maraš	Milovan Lazarević	Stevan Milisavljević
Damir Đaković	Ivana Miškeljin	Milja Simeunović	Stevan Stankovski
Danijela Ćirić	Ivana Vasiljević	Miodrag Milutinov	Strahil Gušavac
Danijela Gračanin	Jasmina Dražić	Miodrag Žigić	Svetlana Bačkalić
Danijela Lalić	Jelena Atanacković	Mirjana Malešev	Svetlana Nikoličić
Darko Čapko	Jeličić	Miroslav Zarić	Tamara Čeranić
Darko Reba	Jelena Borocki	Mirko Borisov	Veran Vasić
Darko Stefanović	Jelena Demko Rihter	Mirko Raković	Vesna Stojaković
Dejan Ecet	Jelena Ivetić	Miro Govedarica	Višnja Žugić
Dejan Lukić	Jelena Radonić	Miroslav Kljajić	Vladimir Ilić
Dejan Reljić	Jelena Slivka	Miroslav Popović	Vladimir Katić
Dejan Jerkan	Jelena Spajić	Miroslav Zarić	Vladimir Mučenski
Dejan Movrin	Kalman Babković	Mitar Jcanović	Vlastimir Radonjanin
Dejan Ubavin	Lazar Kovačević	Mladen Tomić	Vuk Bogdanović
Dejana Nedučin	Lidija Krstanović	Mladen Radišić	Vuk Vranjkovic
Dragan Ivanović	Ljiljana Popović	Nataša Samardžić	Zdravko Tešić
Dragan Ivetić	Ljubica Duđak	Nebojša Brkljač	Zoran Čepić
Dragan Jovanović	Magdolna Pal	Nebojša Radović	Zoran Jeličić
Dragan Pejić	Maja Turk Sekulić	Nebojša Ralević	Zoran Papić
Dragan Ružić	Maja Petrović	Neda Milić Keresteš	Željen Trpovski