|  |  |  |
| --- | --- | --- |
|  | UNIVERZITET U NOVOM SADU  PRIRODNO-MATEMATIČKI FAKULTET  DEPARTMAN ZA MATEMATIKU I INFORMATIKU |  |

**Indeksiranje i pretraživanje naučno-istraživačkih časopisa otvorenog pristupa**

- Master rad -

**Mentor: Student:**

**dr Miloš Savić Jovan Vučetić**

**Novi Sad, 2020. godine**

***Predgovor***

Informacije predstavljaju najvredniji resurs modernog društva. Prikupljanjem i organizacijom podataka nastaju informacije, pomoću kojih se formira znanje potrebno za izvršavanje različitih radnji korisnih čoveku, zbog čega je pristup dobrom izvoru informacija veoma važan. Nastanak i razvoj interneta doneo je čoveku mnoga olakšanja u izvršavanju svakodnevnih zadataka. Radnje koje čovek najčešće izvršava korišćenjem interneta su svakako komunikacija i pretraživanje informacija. Korišćenjem savremenih tehnologija čovek je danas u mogućnosti da veoma lako pristupi velikom broju informacija iz raznih oblasti. Internet je od svog nastanka do danas postao najveća baza znanja poznata ljudskoj istoriji, te danas sadrži neverovatno velike količine podataka u raznim oblicima, zbog čega razmenjivanje znanja ima tendenciju potpunog prelaska na internet. Podaci na internetu uglavnom se nalaze u nestruktuiranoj formi u kojoj nisu naročito značajni čoveku. Takođe, kako je broj podataka dostupnih na internetu prevelik za obradu od strane čoveka, veoma je važno postojanje sistema koji ove nestruktuirane podatke obrađuju i od njih formiraju informacije te omogućavaju pretraživanje istih, kako bi čovek mogao pristupiti željenim informacijama na brz i jednostavan način. Zbog ove činjenice, korišćenje interneta u modernom svetu je gotovo nezamisliv bez raznih sistema za web pretraživanje. Korišćenjem ovih sistema čovek ima mogućnost brzog i jednostavnog pristupa informacija iz raznih domena dostupnih na internetu. Sistemi web pretraživanja mogu imati opštu namenu. Najbolji primeri ovakvih pretraživača su Bing ili Google koji imaju mogućnost pretraživanja svih dostupnih podataka na internetu, uz mnoge dodatne opcije koje ograničavaju pretraživanje prema različitim kriterijumima, kako bi omogućile bolje korisničko iskustvo. Takođe postoje i mnogi sistemi pretraživanja koji su domenski određeni, odnosno fokusirani na pretraživanje informacija iz određene oblasti. Moderni sistemi pretrage nastali su kao plod oblasti informatike koja se naziva pretraživanje informacija.

Pretraživanje informacija podrazumeva pronalazak podataka koji se nalaze u nestruktuiranoj formi, iz velikih kolekcija podataka, na osnovu određenog korisničkog upita. Ova oblast informatike nudi mnoge tehnike za brzu računarsku pretragu velikog obima podataka i njihovo grupisanje i rangiranje na osnovu važnosti za korisnika. Na ovim tehnikama se zasnivaju svi moderni pretraživači, što definiše i sam način korišćenja interneta danas.

U ovom radu predstavljeni su osnovni koncepti pretraživanja informacija kroz teorijsko poglavlje, te je kroz implementaciju manjeg sistema pretraživanja demonstrirana primena ovih koncepata.

Ovaj rad sastoji se iz ukupno četiri poglavlja:

* Uvod
* Dizajn i implementacija
* Demonstracija i testiranje
* Zaključak

U prvom poglavlju ovog rada opisani su teorijski osnovi nekih od glavnih tema kojima se pretraživanje informacija bavi, kao što su: prikupljanje nestruktuiranih podataka u vidu web stranica, skladištenje podataka u indeks fajlove, kao i pretraživanje indeks fajlova na osnovu korisničkog upita.

Pored teorijskog poglavlja, ovo istraživanje obuhvata i implementaciju web aplikacije za pretraživanje informacija o naučnim radovima objavljenih u časopisu otvorenog pristupa. U drugom poglavlju ovog rada biće prikazan dizajn i implementacija ove web aplikacije sa svim njenim značajnim delovima.

Treće poglavlje baviće se demonstracijom kreirane aplikacije, i prikazom svih njenih funkcionalnosti. Ovo poglavlje će objasniti način upotrebe aplikacije i prikazati neke od razultata značajnih za ovo istraživanje.

Zaključak ovog rada, kao i ideje za potencijalna proširenja, prikazani su u četvrtom, finalnom poglavlju.

*Ovom prilikom želim da izrazim veliku zahvalnost dr Milošu Saviću, na prenetnom znanju u toku školovanja, kao i na velikoj pomoći prilikom izrade ovoga rada.*

*Zahvaljujem se i porodici i prijateljima na velikoj podršci u dosadašnjem školovanju.*

Sadržaj

[1. Uvod 7](#_Toc36989902)

[1. Prikupljanje podataka 7](#_Toc36989903)

[2. Skladištenje podataka 10](#_Toc36989904)

[3. Pretraživanje podataka i njihovo rangiranje na osnovu korisničkog upita 11](#_Toc36989905)

[2. Dizajn i implementacija 14](#_Toc36989906)

[1. Podaci 15](#_Toc36989907)

[2. Arhitektura 16](#_Toc36989908)

[3. Prikupljanje podataka 18](#_Toc36989909)

[4. Skladištenje podataka 23](#_Toc36989910)

[5. Pretraživanje i rangiranje podataka 25](#_Toc36989911)

[6. Dodaci u pretraživanju podataka 28](#_Toc36989912)

[3. Demonstracija i testiranje 29](#_Toc36989913)

[4. Zaključak 38](#_Toc36989914)

## Uvod

Razvoj čovečanstva od njegovog nastanka do danas u potpunosti je zasnovan na prikupljenom znanju koje ljudi stiču, zbog čega znanje predstavlja jedan od osnovnih stubova čovečanstva. Zbog ove činjenice jasno je da je razmena znanja ključna stvar u opstanku čovečanstva, kao i u njegovom konstantnom napretku. Moderno društvo posmatra znanje kao najvredniji resurs, te je pristup dobrom izvoru informacija jedno od najvećih vrednosti u savremenom svetu. Zbog ovih činjenica čovečanstvo je u konstatnoj potrazi za pouzdanim i aktuelnim izvorom informacija. Korišćenjem interneta, kao najveće baze znanja u istoriji čovečanstva, čovek je danas u mogućnosti da u bilo kom momentu pristupi velikoj količini informacija iz bilo kog domena. Upravo zbog ogromne količine podataka dostupnih na internetu, dolazi do potrebe za razvojem raznih načina za pretragu podataka, kao i za distinkciju između podataka koji su relevantni za određenog korisnika od podataka koji to nisu. Pored ovoga, dodatni izazov prilikom sticanja znanja korišćenjem interneta, jeste što se podaci pronalaze na različitim lokacijama i u različitim oblicima, što otežava korisnicima formiranje znanja. Grana informatike koja se bavi ovim izazovima jeste pretraživanje informacija.

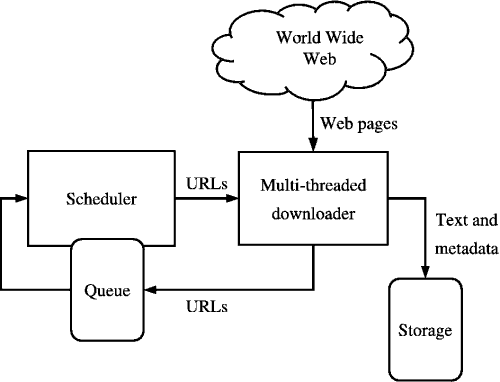
Pretraživanje informacija je grana informatike koja za cilj ima omogućavanje pretrage velike količine podataka i njihovo predstavljanje na način koji je relevantan i pogodan za korisnika. Preciznije određena definicija ove grane informatike data je u knjizi *Introduction to Information Retrieval* i ona glasi „Pretraživanje informacija predstavlja traženje materijala (najčešće dokumenata) nestruktuirane prirode (najčešće tekstualne) iz velike kolekcije (koja je najčešće pohranjena na računaru), koji zadovoljava potrebu za informacijom“[1]. Zbog konstantne potrebe za novim izvorima znanja, kao i zbog činjenice da se potraga za znanjem u modernom društvu obavlja sve češće putem interneta, pretraživanje informacija predstavlja jednu od najpopularnijih oblasti istraživanja. Sistemi za pretraživanje informacija nastali su kao produkt ove oblasti informatike, i danas su gotovo nezaobilazni prilikom korišćenja interneta. Kreiranje brzog i jednostavnog sistema za pretraživanje podataka koji se nalaze u nestruktuiranoj formi na webu predstavlja veliki izazov. Tokom procesa kreiranja ovakvog sistema za pretraživanje moguće je naići na tri glavna izazova:

* Prikupljanje i struktuiranje podataka
* Skladištenje prikupljenih i struktuiranih podataka
* Pretraživanje podataka i njihovo rangiranje na osnovu korisničkog upita

Svaki od ova tri izazova je potrebno savladati kako bi se uspešno kreirao sistem za pretraživanje informacija. U nastavku ovog poglavlja objašnjeni su pomenuti izazovi, kao i osnovni koncepti za prevazilaženje istih.

### Prikupljanje podataka

World wide web predstavlja najveći izvor znanja poznat ljudskoj istoriji, te se kao takav najčešće koristi za prikupljanje podataka svih vrsta i oblika. Podaci na webu se najčešće nalaze u nestruktuiranim oblicima, kao što su tekstualni članci predstavljeni u formi web stranica, koje za cilj imaju da podatke predstave korisniku na način koji će za čoveka biti intuitivan. Ipak kada je u pitanju rad sa velikim obimom podataka koje prikuplja i obrađuje računar, ovaj način predstavljanja podataka nije naročito pogodan. Ipak, kako struktuirani grupisani podaci nisu uvek dostupni, prikupljanje podataka sa web stranica često predstavlja jedini mogući izbor. Najčešća tehnika u prikupljanju podataka sa web stranica jeste korišćenje web crawler komponenti. Ove komponente koriste grafovsku strukturu world wide weba u cilju pretrage međusobno povezanih stranica bez ljudske interakcije. Suština web crawler komponente jeste pretraga html sadržaja web stranica iz kojeg je moguće prikupiti tražene podatke, kao i linkove ka daljim stanicama od interesa. Pretraga web sadržaja korišćenjem web crawlera može biti uopštena ili fokusirana na određeni opseg stranica. Opšta pretraga je proces koji najčešće vrše sistemi pretrage kao što su Google ili Bing. Jedna od najvažnijih osobina dobrog web pretraživača jeste rangiranje stranica prema njihovoj važnosti. Pionir ove oblasti bila je upravo kompanija Google koja je uvela inovacije u svet pretraživanja informacija korišćenjem Page Ranka, algoritma za određivanje „važnosti“ stranice na osnovu njenog položaja u usmerenom web grafu. Kada je u pitanju uopštena pretraga weba koju vrše sistemi za pretraživanje, web crawler komponenta analizira sve dostupne stranice na webu, te ih rangira prema važnosti. Ovaj proces se razlikuje kada je u pitanju usmerena pretraga. Ovakav tip pretrage fokusiran je na određeni domen znanja, kao što je skup web stranica u okviru nekog web sajta. Web crawler komponenta ima za cilj pristupanje skupu međusobno povezanih stranica korišćenjem nekog od algoritama za obilazak grafova, kao i prikupljanje značajnih podataka tokom njihovog obilaska. Na slici 1.1.1. moguće je videti pojednostavljenu arhitekturu web crawler komponente.



Slika 1.1.1. Arhitektura web crawler komponente [2]

Kao što je moguće videti na slici web crawler se sastoji iz nekoliko osnovnih delova. Ova komponenta najčešće koristi više procesa prilikom preuzimanja i analize html stranica kako bi se sama pretraga ubrzala. Deo koji vrši preuzimanje i analizu html sadržaja stranica na slici je označen kao „Multi-threaded downloader“. Kako bi se ovi procesi preuzimanja i skladištenja uspešno orkestrirali potrebana je komponenta koja ih pokreće i dodeljuje im linkove ka stranicama koje je potrebno posetiti, ova komponenta na slici je označena kao „Scheduler“. Takođe web crawler mora sadržati queue, odnosno kolekciju linkova ka stranicama koje je potrebno posetiti. Takođe, preporučeno je i korišćenje kolekcije stranica koje su već posećene kako bi se izbeglo posećivanje stranica koje su već posećene, što može izazvati i zaglavljivanje web crawlera u slučaju pojave konture u web grafu. Poslednji deo web crawler komponente jeste skladište podataka dobijenih za vreme analize prikupljenih web stranica.

Proces web pretraživanja započinje određivanjem skupa početnih stranica za obilazak. Prilikom ovog koraka je najvažnije pažljivo odabrati stranice iz kojih se putem linkova može stići do što većeg broja drugih stranica od značaja. Važnost ovog koraka se ogleda u tome da ukoliko se odabir početnih tačaka pretraživanja ne izvrši na adekvatan način, celokupno pretraživanje može biti neuspešno. Prilikom posete svake stranice, njen link se najpre prebacuje iz skupa neposećenih u skup posećenih stranica, nakon čega se sa stranice preuzimaju svi potrebni podaci. Poslednji korak koji se obavlja na svakoj stranici je izvlačenje linkova ka ostalim stranicama, i njihovo smeštanje u kolekciju neposećenih ukoliko stranice na tim linkovima već nisu posećene. Kraj procesa web pretraživanja nastupa u trenutku kada kolekcija neposećenih linkova više ne sarži ni jedan element, ili se ovaj proces prekine od strane orkestratora.

Kako bi web crawler komponenta imala svoju svrhu, veoma je važno postojanje segmenta za analizu html sadržaja web stranice, kao i parsiranje podataka sa iste. Prilikom posete svake od stranica komponenta zadužena za prikupljanje podataka izvršava tri koraka: pristupanje traženim html elementima, parsiranje vrednosti elemenata u podatke i struktuiranje i čuvanje podataka. Kada su u pitanju prva dva koraka, oni uveliko zavise od same html strukture stranice, što je veoma često najveći izazov prilikom implementacije komponenti za prikupljanje podataka. Kako bi pravilno prikupljanje podataka sa web stranice bilo moguće, sama stranica mora imati jasno definisanu strukturu. Takođe kako bi se omogućilo lakše preuzimanje podataka, web stranice često koriste identifikacione html atribute kao što su id ili data atributi, pomoću kojih web crawler komponente mogu direktno pristupiti željenim html elementima i njihovim vrednostima. Nakon prikupljanja traženih podataka sa stranice, poslednji korak jeste struktuiranje i čuvanje istih. Ovaj proces počinje dizajnom strukture podataka, koja zavisi od samog cilja istraživanja. Ove strukture podataka često se nazivaju i modeli. Modeli u svojoj strukturi sadrže atribute čije vrednosti bivaju određene na osnovu prikupljenih podataka iz prethodnih koraka. Prilikom kreiranja modela, potrebno je dodeliti vrednost svim atributima koji su dostupni i neophodni za dalje istraživanje. Nakon što su novi podaci prikupljeni i struktuirani u modele, najčešća praksa jeste njihovo skladištenje, kako bi se omogućio kasniji pristup bez potrebe za ponovnom analizom web sadržaja.

Kao što je moguće videti iz prethodnih nekoliko pasusa, upotreba web crawler komponenti je veoma prost ali efikasan način pravilnog prikupljanja podataka koji se nalaze u formatu web stranica. Korišćenjem pogodnosti koje ova tehnika prikupljanja podataka pruža, nestruktuirani podaci sa weba se na jednostavan način mogu prikupiti i struktuirati, kako bi bili spremni za skladištenje i dalju upotrebu.

### Skladištenje podataka

Prilikom implementacije sistema za pretraživanje informacija veoma je važno pronaći adekvatan način skladištenja podataka. Kada je implementacija ovakvih sistema u pitanju, korišćenje klasičnih relacionih baza podataka često nije najbolja opcija. Razlog ovome jeste činjenica da su korisnički upiti, kao i sami podaci vrlo često nestruktuirani i nepotpuni, te nisu pogodni za skladišta podataka kao što su relacione baze. Takođe kod sistema za pretraživanje informacija veoma je važno rangiranje rezultata prema važnosti za korisnika, što nije moguće korišćenjem baza podataka. Kako bi se ove prepreke zaobišle pretraživači često koriste indekse, specijalne strukture podataka koje omogućavaju brzo pretraživanje podataka. Ove strukture podataka omogućavaju skladištenje nestruktuiranih podataka kao što su tekstualni podaci, i njihovo pretraživanje na osnovu korisničkog upita, koji ne mora biti potpun niti tačan kako bi se pretraživanje uspešno izvršilo. Skladištenje podataka u indeks zasniva se na takozvanim indeks termovima, odnosno ključnim rečima. Prilikom indeksiranja podataka, nad njihovim tekstom je moguće vršiti različite transformacije koje mogu služiti optimizaciji pretraživanja i rangiranju rezultata. Ove transformacije mogu obuhvatati uklanjanje dijalekata, delova teksta koji služe za formatranje, veznika, kao i svođenje reči na njihov koren i slično. U slučaju izvršavanja ovakvih transformacija veoma je važno primeniti iste i na svaki korisnički upit, kako bi se pretraga zaista optimizovla. Rezultati pretrage indeks fajlova su dokumenti, koji se rangiraju prema važnosti za korisnički upit. Ovakav format skladištenja podataka u svetu savremenih tehnologija vrlo često je povezan sa mašinskim učenjem, koje za cilj ima unapređivanje korisničkog iskustva brzinom pretrage i relevantnošću vraćenih rezultata. Postizanje ovog cilja korišćenjem mašinskog učenja zasniva se na automatizovanoj klasifikaciji i klasterizaciji dokumenata, kao i automatizovanom izvlačenju podataka i njihovom rangiranju na osnovu korisničkog upita. Takođe oblast nauke s kojom se ovakav tip skladištenja podataka bavi jeste procesiranje prirodnih jezika. Upotreba ove oblasti zasniva se na sintaksnoj i semantičkoj analizi teksta koji se indeksira. Ova analiza omogućava premošćavanje određenih pojava u jezicima koje mogu negativno da utiču na pretragu, kao što su na primer sinonimi ili homonimi. Takođe ovakva vrsta analize omogućava autokorekciju eventualnih grešaka u rečima korisničkog upita, kao i dopunjavanje upita na osnovu konteksta pretrage.

Proces indeksiranja obuhvata prijem podataka za indeksiranje, kreiranje dokumenata, njegovo popunjavanje na osnovu ključnih reči i na kraju smeštanje kreiranih dokumenata u indeks. Prvi korak u indeksiranju započinje kada komponenta za indeksiranje dobije ulazni podatak ili skup podataka. Nakon toga sledi kreiranje dokumenta koji predstavlja strukturu za skladištenje podataka kada je u pitanju ovakav vid skladištenja. Ukoliko ulazni parametar indeksiranja predstavlja kolekciju podataka, najčešća praksa jeste kreiranje dokumenta za svaki od elemenata tog skupa. Sledeći korak jeste popunjavanje dokumenata na osnovu ulaznog parametra i ključnih reči. Prilikom popunjavanja dokumenta moguće je dodeliti jednu ili više ključnih reči na osnovu kojih je moguća kasnija pretraga. Svaka ključna reč uz sebe ima određene vrednosti koje mogu biti različitog tipa. Ovaj tip skladištenja veoma podseća na skladištenje u strukturama podataka kao što su mape. Nakon što se dokument kreira on biva smešten u indekser, koji za cilj ima kreiranje indeks fajlova i smeštanje dokumenata u iste. Indekser takođe u zavisnosti od svoje konfiguracije može vršiti različite operacije nad dokumentima u cilju optimizacije kasnije pretrage. Nakon što se indeks fajlovi kreiraju oni su otvoreni za pretraživanje dokumenata na osnovu prethodno definisanih ključnih reči.

### Pretraživanje podataka i njihovo rangiranje na osnovu korisničkog upita

Nakon što se indeks fajlovi kreiraju poslednji korak jeste omogućavanje pretrage podataka na osnovu korisničkog upita. Kada je pretraga podataka u pitanju najveći izazov jeste predviđanje relevantnosti dokumenta na osnovu korisničkog upita. Kroz istoriju pretraživanja informacija korišćene su razne vrste IR modela. Ovi modeli definišu kako su upiti i dokumenti struktuirani kao i na koji način se relevantnost dokumenata određuje.

Osnovni i najjednostavniji IR model koji je i danas u upotrebi jeste boolean model. Ovaj IR model se zasniva na binarnoj relevantnosti dokumenata. Ovo znači da prilikom pretrage dokumenata, svaki dokument može biti samo relevantan ili irelevantan za pretragu, što zavisi od korisničkog upita. Ovaj model omogućava konstrukciju korisničkog upita na osnovu termova, odnosno reči koje se pretražuju, kao i logičkih operatora **I, ILI** i **NE**. Korišćenjem termova i logičkih operatora korisniku se omogućava kreiranje korisničkog upita, na osnovu koga se dobija rezultat pretrage koji sadrži sve one dokumente koji zadovoljavaju taj upit, te se na taj način ostvaruje veoma efikasna pretraga podataka. Ipak uprkos svojoj efikasnosti, pretraživanje informacija korišćenjem boolean IR modela ima mnogobrojne nedostatke. Osnovni problem ovog modela pretraživanja jeste sama binarna relevatnost dokumenta, koja onemogućava da neki dokumenti budu relevantniji od drugih na osnovu upita. Iz ove činjenice takođe proizilazi da su svi relevantni dokumenti jednaki, te da korisnik nema uvid u to koji od ponuđenih dokumenata može biti od najveće koristi bez pregledanja kompletnog dokumenta. Pored ovoga, jedna od velikih mana ovog modela pretraživanja jeste činjenica da su rezultati pretrage često preveliki ili premali, što u oba slučaja uzrokuje loše korisničko iskustvo. Takođe još jedna od mana ovakvog pristupa jeste sam način predstavljanja upita u boolean modelu. Upiti u ovom IR modelu moraju se predstaviti kao logički izrazi, što zahteva određeno predznanje, te predstavlja prepreku mnogim neobučenim korisnicima. Zbog ove činjenice, odgovornost programera jeste da omogući korisnicima pretragu korišćenjem prirodnog jezika, koji potom biva preveden u logički izraz pre samog izvršenja pretrage. Zbog svih navedenih mana boolean modela, postoji potreba za novim, unapređenim IR modelom koji omogućava rangirano pretraživanje korišćenjem prirodnog jezika pri konstruisanju korisničkog upita.

Rangirano pretraživanje predstavlja unapređenu verziju prvobitnog boolean pretraživanja. Najznačajniji IR model rangiranog pretraživanja jeste vector-space model. Osnovni princip ovog modela jeste ocenjivanje relevantnosti dokumenta na osnovu korisničkog upita. Vrednost relevantnosti dokumenta u ovakvom načinu pretraživanja više nije binarna, već se može kvantitativno odrediti. Ova činjenica unapređuje korisničko iskustvo u mnogim sferama. Najvažnija prednost ovakvog pristupa jeste mogućnost sortiranja rezultata pretrage po njihovoj relevantnosti čime se omogućava da korisnik ima uvid u to koji dokumenti mogu imati najviše značaja za uneseni upit. Osim toga, ovaj pristup rešava i problem prevelikog broja vraćenih rezultata, tako što omogućava vraćanje tačno određenog broja najznačajnijih dokumenata. Takođe ovaj model pretrage zasnima se na korisničkom upitu koji se piše u prirodnom jeziku, što veoma unapređuje korisničko iskustvo, ali i uklanja odgovornost sa programera kada je u pitanju prevođenje upita u logički izraz.

Mera relevantnosti dokumenta u pretraživanju informacija često se naziva i skor. Određivanje ove mere predstavlja jedan od najvećih izazova kada je u pitanju rangirano pretraživanje informacija. Računanje skora vrši se za svaki dokument posebno na osnovu nivoa poklapanja sa unetim korisničkim upitom. Osnovni pristup kalkulaciji skora jeste upotreba statističke relevantnosti, odnosno korišćenje učestalosti pojave termova iz upita u samom dokumentu. Ova metrika naziva se term frekvencija i zasniva se na pretpostavci da je dokument u kome se često pojavljuju ključne reči iz upita gotovo sigurno važan za korisnika. Mana ovog pristupa jeste činjenica da značaj dokumenta linearno raste sa brojem pojavljivanja terma iz upita u posmatranom dokumentu. Ovo znači da će dokument u kome se term iz upita pojavljuje pet puta, biti pet puta važniji od dokumenta u kome se isti term pojavljuje samo jednom, što ne predstavlja naročito precizan rezultat pretrage. Zbog ove činjenice dolazi do potrebe za preciznijim pristupom, te se kao unapređeni način računanja skora koristi log frekvencija. Ovaj način zasniva se na tome da se skor dokumenta za term frekvenciju računa kao:

Ovaj način se sada može koristiti za računanje skora koji daje realniji prikaz relevantnosti. Kako bi se izračunao skor, najpre je potrebno izračunati težinu svakog terma iz upita i dokumenata koji se pretražuju. Težina terma predstavlja značaj određenog terma za pretragu koja se vrši, i ona se računa kao log frekvencija terma. Nakon što su težine svih termova izračunate, njihova suma daje skor za svaki od dokumenata.

Kako bi se rangiranje dokumenata prema nivou relevantnosti dodatno unapredilo, osim same frekvencije pojavljivanja nekog terma u dokumentu u obzir je moguće uzeti i dokument frekvenciju, odnosno broj dokumenata u celokupnoj kolekciji koji sadrže term iz upita. Ovaj pristup zasniva se na činjenici da su termovi koji se pojavljuju u malom broju dokumenata uglavnom informativniji, te su dokumenti koji sadrže ove termove najčešće relevantni za korisnika. Kako bi izračunati skor bio obrnuto srazmeran sa brojem pojavljivanja nekog terma na nivou čitave kolekcije dokumenata, koristi se inverzna dokument frekvencija, koja se računa pomoću formule:

Gde N predstavlja ukupan dokumenata, a predstavlja broj dokumenata koji sadrže term t. S obzirom da se prilikom računanja skora, računa i težina reči u upitu, upotreba inverzne dokument frekvencije ne utiče na rezultate pretrage ukoliko se korisnički upit sastoji od samo jednog terma.

Kombinovanjem term frekvencije i inverzne dokument frekvencije dobija se najoptimalniji poznati način za računanje težine termova u korisničkom upitu i dokumentima. Ova formula glasi:

Gde je term frekvencija terma t u dokumentu d, N broj dokumenata, a broj dokumenata koji sadrže term t.

Nakon računanja težine svakog od termova svaki od dokumenata, kao i upit moguće je predstaviti pomoću vektora sačinjenog izračunatih težina. Ukoliko se i upit i dokument predstave na ovaj način, važnost dokumenta za zadati upit moguće je izračunati kao sličnost između ova dva vektora. Osnovni pristup računanju sličnosti ova dva vektora jeste računanje njihove inverzne distance. Ipak, ovaj način nije naročito pogodan zbog činjenice da Euklidova distanca među vektorima različite dužine uvek ima veliku vrednost. Bolja alternativa ovom pristupu jeste korišćenje ugla među vektorima, za računanje nivoa preklapanja između vektora upita i dokumenta. Kako bi ova mera bila srazmerna sličnosti vektora, koristi se kosinusna sličnost, čija vrednost raste sa opadanjem ugla između dva vektora. Kako bi se ovo računanje dodatno optimizovalo, vektori mogu biti normalizovani po svojoj dužini. Ova normalizacija vrši se tako što se svaka vrednost u vektoru podeli sa brojem tačaka tog vektora. Vršenjem ove operacije, dokumentima različitih dužina se obezbeđuje mogućnost poređenja.

Računanje kosinusne sličnost vektora upita i vektora dokumenta normalizovanih prema dužini vrši se pomoću formule:

Gde predstavlja vektor upita, vektor dokumenta, i njihove dužine respektivno, a predstavlja vektorski proizvod. Ako se u obzir uzme da su težine svih termova veće ili jednake sa 0, izvodi se zaključak da će kosinusna sličnost uvek biti . Nakon računanja kosinusne sličnosti između upita i dokumenata, moguće je vratiti zadati broj dokumenata sa najvećom sličnošću.

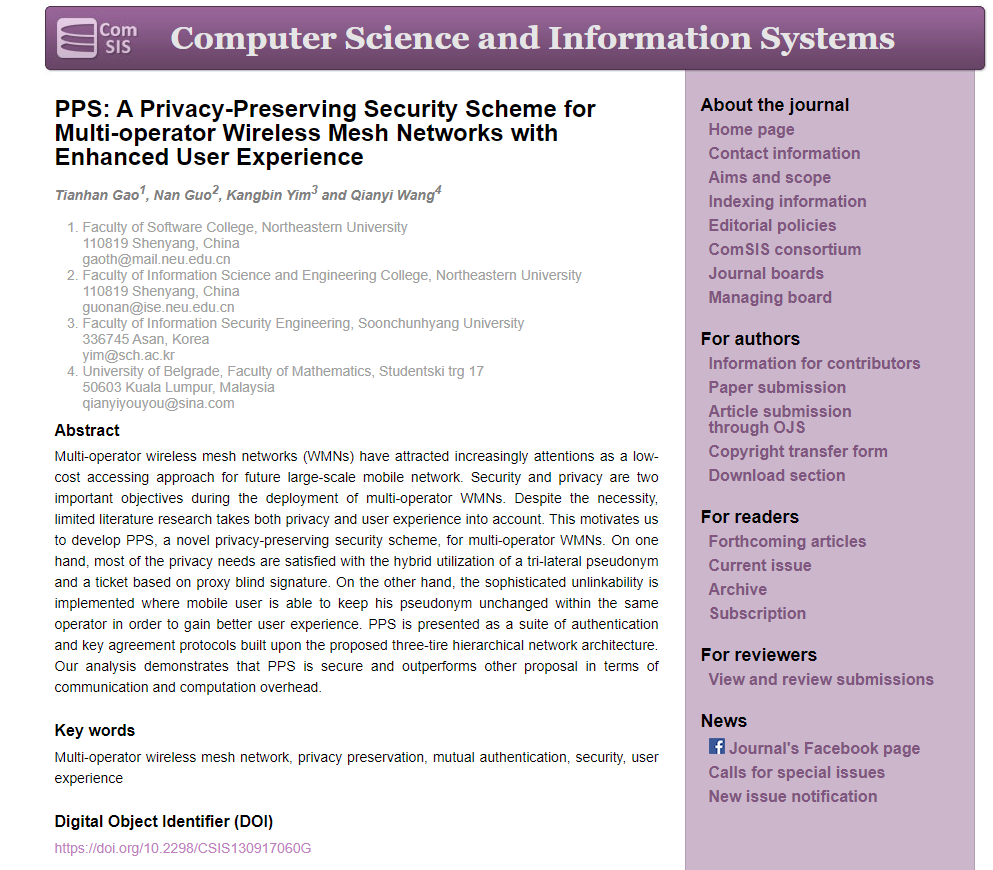
## Dizajn i implementacija

Kao što je pomenuto u prethodnom poglavlju, sistemi pretraživanja informacija mogu biti opšti ili usmereni na određenu oblast. Kako bi se dokazao koncept ovakvog pretraživanja, u okviru ovog rada implementirana je web aplikacija za pretraživanje naučnih radova iz oblasti računarskih nauka, objavljenih u časopisima otvorenog pristupa. [PRIMER RANIJIH RADOVA NA OVU TEMU]

Ovaj sistem kreiran je iz dve glavne komponente: servera, koji je implementiran kao aplikacioni programski interfejs (API) i klijentske web aplikacije koja konzumira serverski API. Za potrebe implementacije serverske strane, korišćeno je Java razvojno okruženje Spring Boot. Glavni razlog korišćenja ovog razvojnog okruženja je njegova jednostavnost, kao i činjenica da je baziran na Java programskom jeziku, što omogućava korišćenje velikog broja biblioteka otvorenog pristupa zasnovanih na tom programskom jeziku, među kojima je i Lucene, biblioteka za pretraživanje informacija korišćena za potrebe ovog rada. Prilikom implementacije klijentske aplikacije korišćen je Angular 8, TypeScript razvojno okruženje. Odabir ove tehnologije je veoma prirodan ako se u obzir uzme da su web aplikacije današnjeg doba izrazito okrenute modernim razvojnim okruženjima za kreiranje klijetskih web aplikacija, kao što su Angular, React, VueJS i slični. Angular je zbog svoje jednostavnosti i veoma kvalitetno napisane dokumentacije veoma često prvi izbor prilikom kreiranja modernih single page aplikacija (SPA). U nastavku ovog poglavlja biće opisani podaci prikupljeni za potrebe ove aplikacije, high-level arhitektura ovog sistema, kao i implementacija procesa za pretraživanje podataka na webu, skladištenje sakupljenih podataka kao i njihovo pretraživanje i rangiranje na osnovu korisničkog upita. Pored ovoga, u toku poglavlja biće opisana i implementacija nekih dodatnih funkcionalnosti koje ova aplikacija nudi.

### Podaci

Prvi korak pri izradi sistema za pretraživanje svakako jeste određivanje podataka koji će biti prikupljeni, struktuirani i dostupni za pretraživanje. Za potrebe ovog istraživanja korišćeni su podaci prikupljeni iz arhive časopisa *„Computer Science and Information Systems“ (ComSIS)* [3]. ComSIS je naučni časopis otvorenog tipa koji objavljuje originalne naučne radove kako teorijskog, tako i komercijalnog odnosno industrijskog tipa, usmerenih primarno na računarske nauke. Kako ovaj časopis ne nudi otvoren pristup skladištu podataka u vidu aplikacionog programskog interfejsa, jedini način pristupa podacima o naučnim radovima objavljenim u ovom časopisu jeste korišćenje web crawler komponente, koja je opisana kasnije u toku ovog poglavlja. Arhiva ovog naučnog časopisa predstavljena je kao web stranica, koja u svom sadržaju poseduje linkove ka drugim stranicama koje predstavljaju brojeve časopisa. Stranice brojeva časopisa u svom sadržaju poseduju linkove ka svim naučnim radovima objavljenim u tom broju časopisa. Sama web stranica naučnog rada prikazana je na slici (2.1.1). Kao što je moguće videti na slici web stranica na kojoj je predstavljen naučni rad, sadrži potake o naslovu rada, autorima, apstraktu kao i ključnim rečima u radu. Pored ovoga, stranica sadrži podatke i o godini u kojoj je rad objavljen, kao i link za preuzimanje kompletnog rada u vidu PDF dokumenta. Svi ovi podaci korišćeni su u fazi indeksiranja podataka o naučnim radovima.

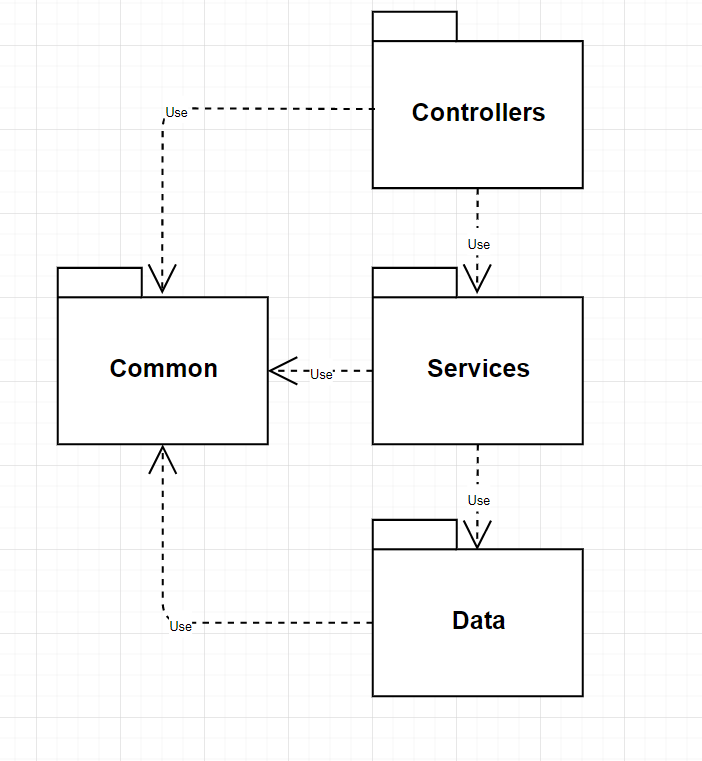


Slika 2.1.1. Primer izgleda stranice naučnog rada u ComSIS časopisu

### Arhitektura

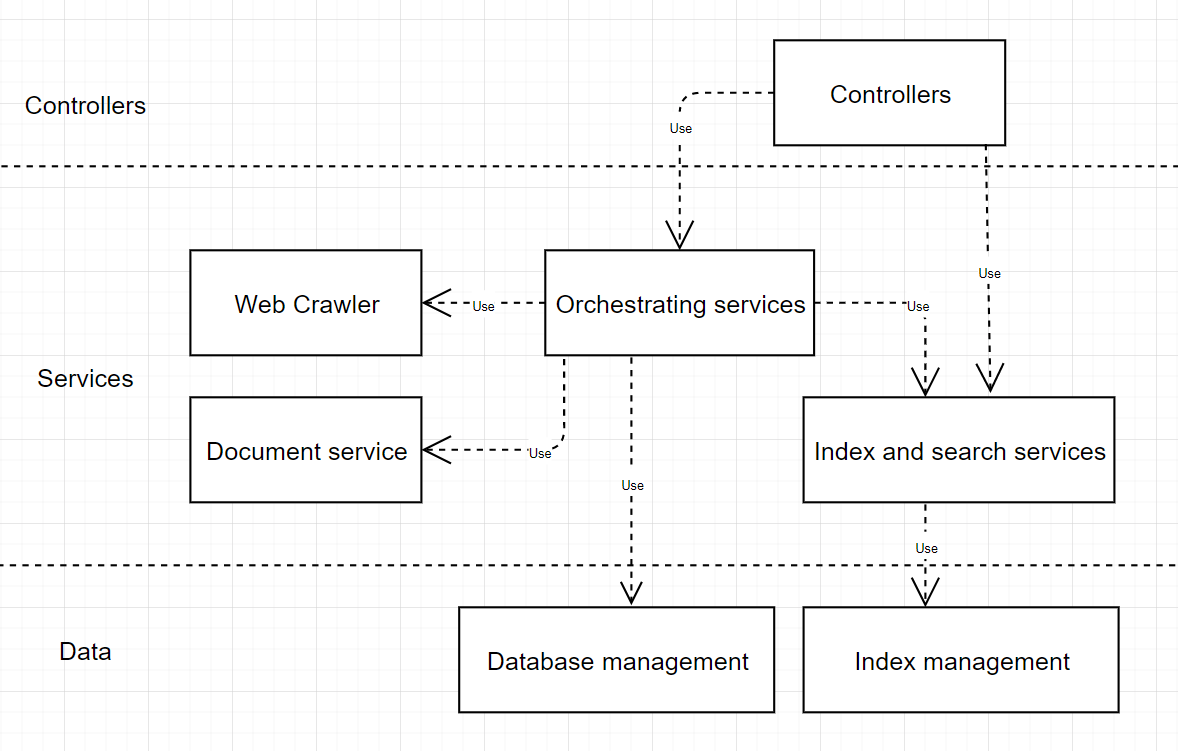
Kao što je pomenuto u ranijem delu ovog poglavlja, aplikacija kreirana u sklopu ovog rada se može podeliti na dve odvojene celine: klijentsku i serversku. Arhitektura klijentskog dela aplikacije je prilično trivijalna, ako se u obzir uzme činjenica da se kompletan rad aplikacije zasniva na serverskom delu, te da klijentska aplikacija samo šalje zahteve ka serveru i prikazuje dobijene rezultate. Takođe, klijentska aplikacije ne sadrži nikakvu logiku za pretraživanje informacije, zbog čega ovaj deo aplikacije neće biti detaljno opisan u sklopu ovog rada.

Serverski deo ovog sistema pretrage, kreiran je kao aplikacioni programski interfejs (API). Arhitekturu ove komponente moguće je jasno razdeliti na tri subkomponente: kontroler, servis i komponenta za rad sa podacima (Slika 2.2.1).



Slika 2.2.1. High-level arhitektura web api komponente

Svaka od pomenutih komponenti ima određen skup odgovornosti. Takođe kao što je moguće videti na slici, svaka od komponenti oslanja se i na komponentu „*Common*“ koja sadrži infrastrukturalne klase koje olakšavaju pisanje i održavanje koda. Komponente sa slike moguće je podeliti i na manje celine od kojih svaka ima sopstvene odgovornosti. Ova arhitekura ilustrovana je na slici 2.2.2, i na njoj je moguće uočiti podelu kompletnog servera na tri sloja: sloj kontrolera, sloj servisa i sloj podataka.



Slika 2.2.2. Slojevi unutar serverskog dela aplikacije

Sloj kontrolera predstavlja ulaznu tačku web api komponente, te kao jedinu odgovornost ima prosleđivanje korisničkog poziva određenom servisu i obezbeđivanje povratne informacije korisniku. S obzirom da ni jedan od kontrolera iz ove komponente ne sadrži nikakvu biznis logiku, niti bilo kakve delove koda ključne za ovaj rad, klase unutar ove komponente neće biti detaljno opisane u sklopu ovog rada.

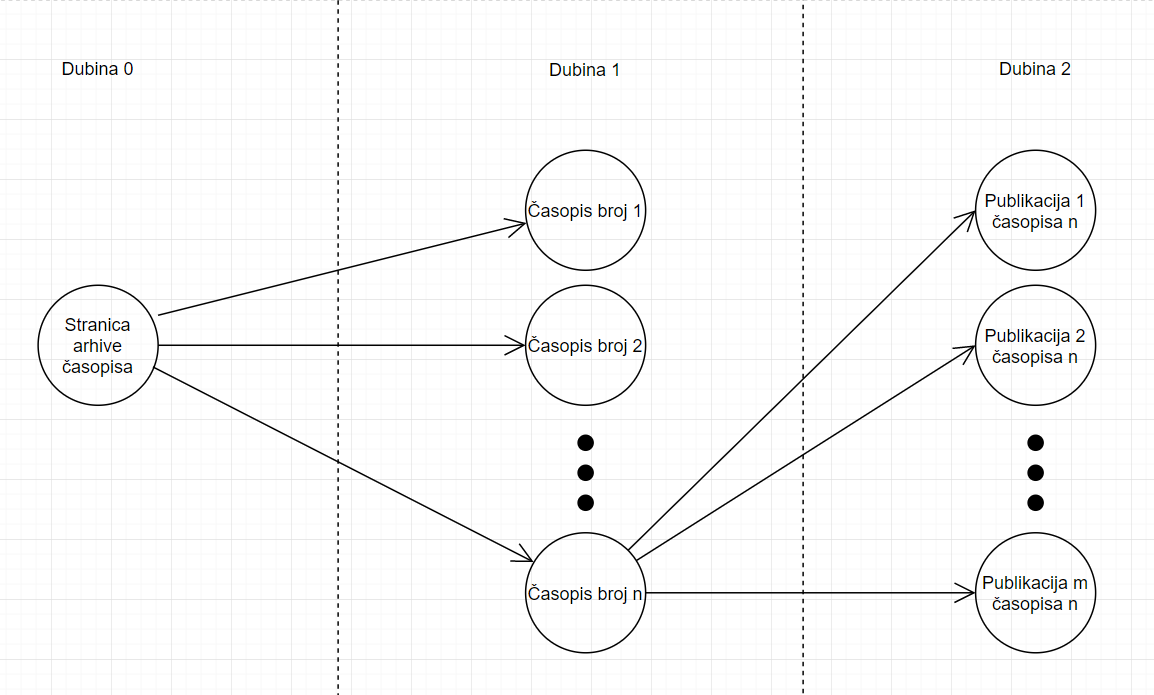
Sloj servisa je zasigurno najsloženiji sloj serverskog dela aplikacije. Komponenta servisnog nivoa koja poseduje najveći broj odgovornosti je svakako orkestrator. Orkestrator se sastoji od dva servisa: *PublicationService* i *AuthorService,* koji su zaduženi za prikupljanje i čuvanje podataka o publikacijama i autorima. Pri inicijalnom pokretanju aplikacije vrši se poziv metoda *loadPublications* klase *PublicationService.* Ovaj metod pokreće niz operacija čijim se izvršavanjem podaci o publikacijama i njihovim autorima preuzimaju i čuvaju u PostgreSQL bazi podataka. Celokupan proces prikupljanja podataka i njihovog čuvanja u bazu podataka opisan je nešto kasnije tokom ovog poglavlja. Pored metoda za učitavanje i čuvanje podataka, *PublicationService* sadrži i metod *indexPublications* koji za cilj ima preuzimanje prikupljenih podataka o publikacijama iz baze podataka, njihovo dopunjavanje i smeštanje u indeks fajlove. Druga značajna klasa orkestratora je *AuthorService* koja sadrži metod za smeštanje podataka o autoru u bazu, i metod za indeksiranje podataka o autorima. Prvi od ova dva metoda poziva se od strane klase *PublicationService* prilikom prikupljanja i čuvanja podataka u bazi. Razlog tome jeste činjenica da časopis ComSIS ne sadrži stranice o autorima radova, već se podaci o autorima dobavljaju isključivo iz podataka o naučnom radu, zbog čega je u cilju dobavljanja podataka o autoru najpre potrebno učitati podatke o naučnom radu.

Pored orkestratora, servisni nivo sadrži još i web crawler komponentu, koja je zadužena za prikupljanje podataka sa web stranica, kao i *DocumentService* koji je zadužen za preuzimanje PDF dokumenata o publikacijama sa web stranica i ekstraktovanje tekstualnog sadržaja iz njih, kako bi se omogućilo indeksiranje kompletnih naučnih radova. Poslednja komponenta servisnog sloja jesu servisi za indeksiranje i pretragu podataka o naučnim radovima i njihovim autorima. Svaka od pomenute tri komponente, biće detaljnije opisana kasnije u toku ovog poglavlja.

Kao što je navedeno u teorijskom uvodu ovog rada, u cilju kreiranja sistema za pretraživanje informacija potrebno je obezbediti načine za prikupljanje podataka, skladištenje podataka kao i njihovo pretraživanje i rangiranje. U nastavku ovog poglavlja biće opisani načini za prevazilaženje izazova u svakom od ovih koraka.

### Prikupljanje podataka

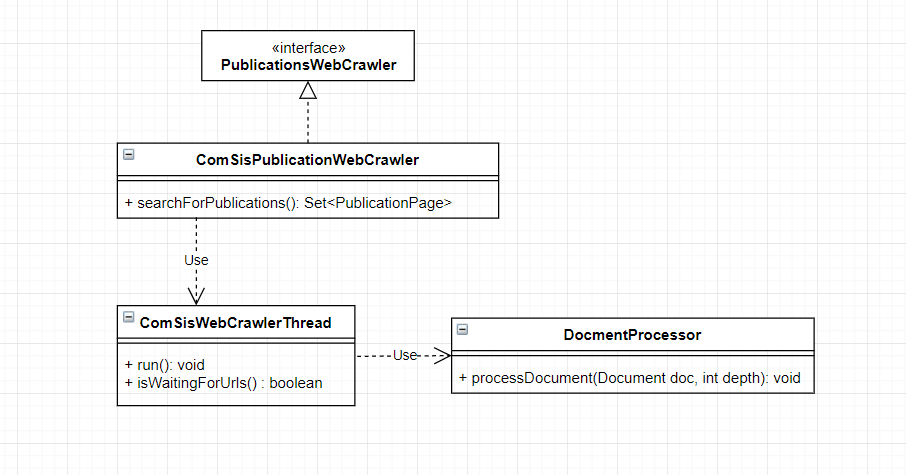
Početna tačka ovog istraživanja jeste prikupljanje podataka o naučnim radovima. Kao što je rečeno u ranijem delu rada, za potrebe ovog rada podaci su prikupljani iz arhive časopisa ComSIS, te je u cilju njihovog prikupljanja korišćena web crawler komponenta. Kako bi ovu komponentu bilo moguće implementirati najpre je potrebno utvrditi samu strukturu web sajta, kao i utvrditi skup početnih stranica sa kojih se pretraga može izvršavati. Nakon kraće analize ComSIS web sajta, moguće je uvideti jasnu i pravilnu strukturu linkovanja prikazanu na slici 2.3.1.



Slika 2.3.1. Struktura web sajta časopisa ComSIS

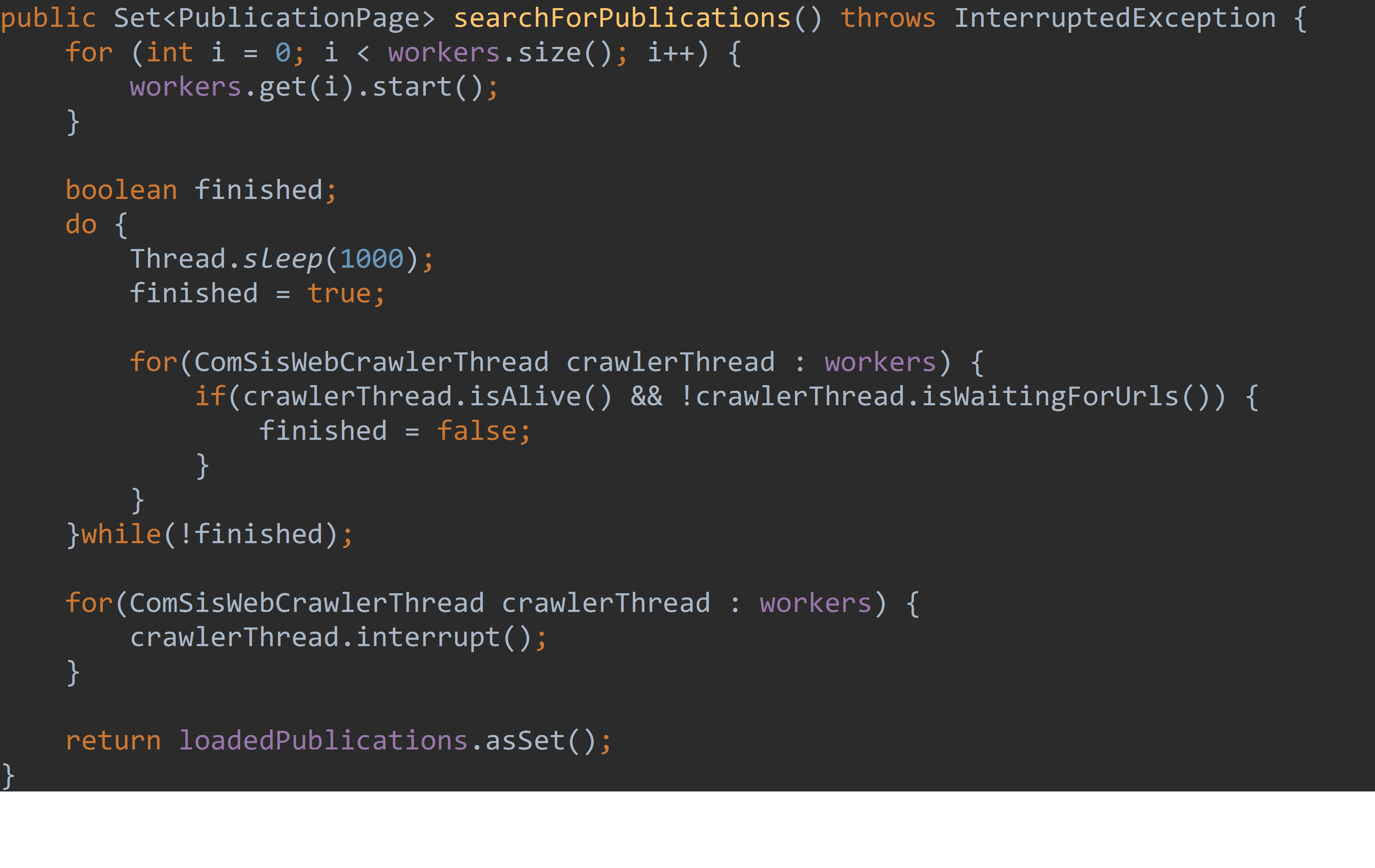
Nakon osvrta na graf strukture sajta nameće se jasan zaključak da je najbolja početna tačka za komponentu pretraživanja stranica arhive časopisa, jer je sa te stranice putem linkova moguće pristupiti svim ostalim stranicama od interesa. Ova struktura takođe omogućava relativno jednostavan proces samog posećivanja stranica, korišćenjem nekog od algoritama za obilazak usmerenih grafova. Za potrebe ovog istraživanja korišćeno je pretraživanje grafa u širinu (BFS). Iako ovakav način pretrage nema značajne prednosti u odnosu na pretragu grafa po dubini, ovo je elegantan način da se web stranice ovog sajta pretražuju po dubini, odnosno od stranice arhive ka stranicama koje predstavljaju publikacije.

Nakon analize grafovske strukture sajta, moguće je implementirati web crawler komponentu. Sama struktura web crawler komponente prikazana je na slici 2.3.2.



Slika 2.3.2. Struktura web crawler komponente

Kao što je moguće videti na dijagramu kalasa sa slike, konstruisan je interfejs *PublicationsWebCrawler* namenjen za korišćenje od strane eksternih komponenti. Klasa *ComSisPublicationWebCrawler* predstavlja implementaciju ovog interfejsa, i kako je propisano interfejsom sadrži samo jedan metod *searchForPublications* koja za cilj ima inicijalizaciju procesa za pretragu i njihovo upravljanje. Prilikom vršenja ovog istraživanja klasa *ComSisPublicationWebCrawler* konfigurisana je da inicijalizuje deset procesa pretraživanja koji se paralelno izvršavaju. Prilikom inicijalizacije ova klasa kreira instance skupa posećenih stranica, skupa parsiranih publikacija kao i red opsluživanja stranica koje tek trebaju biti posećene. Ova tri polja su veoma važna zbog toga što se prosleđuju svim procesima pretraživanja koji ih potom koriste za skladištenje izvučenih podataka i određivanje završetka pretrage. S obzirom na činjenicu da su ova tri polja deljena na nivou više procesa, veoma je važno da strukture podataka koje se koriste budu sinhronizovane, odnosno da se onemogući pristup strukturi od strane više procesa istovremeno. Klasa *ComSisPublicationWebCrawler* startuje deset instanci klase *ComSisWebCrawlerThread* i pokreće ih svake sekunde. Razlog zbog koga se nakon svake iteracije proces zaustavlja na jednu sekundu je zbog usporavanja slanja http zahteva. Ovaj korak je veoma važan jer u slučaju kreiranja prevelikog broja zahteva u kratkom vremenskom periodu, često dolazi do preopterećenja servera. Pored toga, mnoge web aplikacije ovakvo ponašanje identifikuju kao napad, što rezultuje blokiranjem zahteva sa određene IP adrese. Takođe nakon svake od iteracija potrebno je proveriti da li je proces pretrage završen. Ovaj podatak se računa tako što program iterira kroz sve instance klase *ComSisWebCrawlerThread* i proverava da li je pretraga za njih završena. Ukoliko su svi procesi pretrage završeni, skup prikupljenih modela vraća se korisniku. Implementacija ovog procesa prikazana je na listingu 2.3.3.



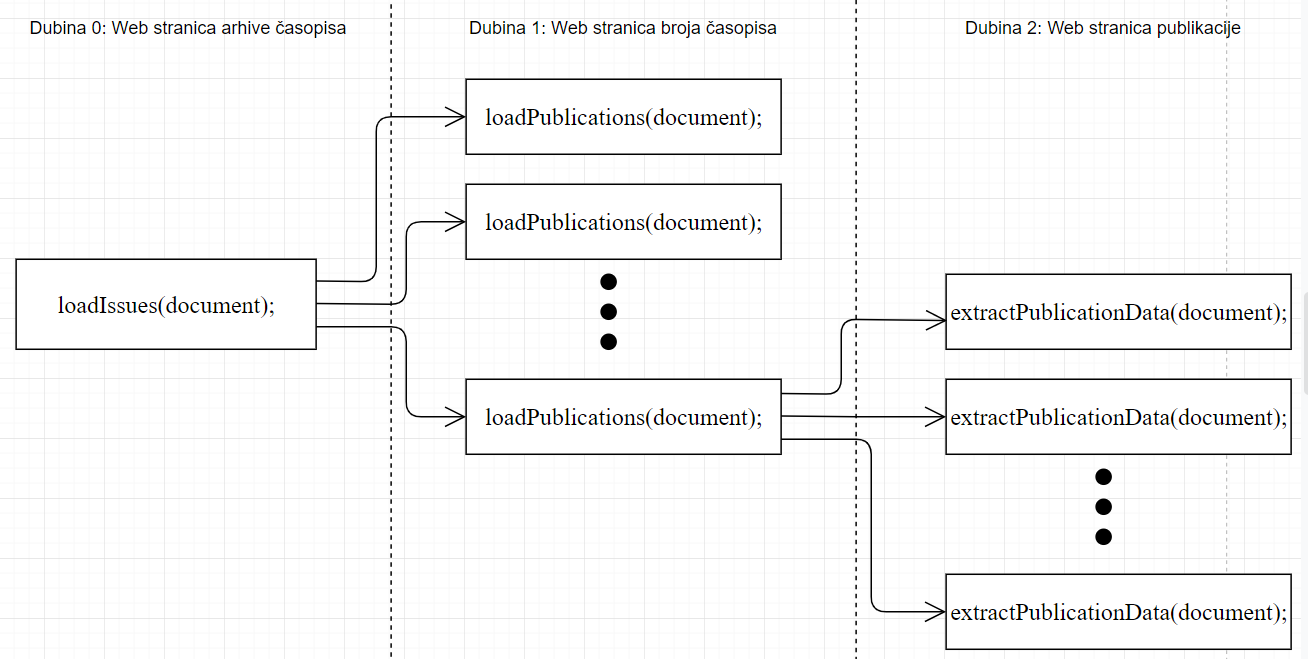
Listing 2.3.3. Upravljanje web crawler procesima

Centralna klasa web crawler komponente jeste *ComSisWebCrawlerThread.* Ova klasa nasleđuje je naslednica klase *java.lang.Thread*, što omogućava objektima ove klase da izvršavaju kod paralelno sa drugim procesima istog ili drugog tipa. Ovakav pristup je veoma pogodan kada je u pitanju pretraživanje web sajtova koji često mogu biti veoma obimni, stoga se korišćenjem paralelne obrade stranica značajno skraćuje vreme pretraživanja sajta. Klasa *ComSisWebCrawlerThread*  sadrži *run* metod koji pokreće proces učitavanja web stranica i njhovo analiziranje i parsiranje. Prilikom poziva ovog metoda proces ulazi u beskonačnu petlju i na taj način omogućava da se proces izvršava dokle god ne bude prekinut od strane klase koja ga je pokrenula. U svakoj iteraciji ove petlje objekat najpre proverava red opsluživanja koji sadrži objekte klase *WebPageData*. Svaki od ovih objekata sadrži link ka stranici koja do sada nije posećena, kao i dubinu u graf strukturi na kojoj se ta stranica nalazi. Ukoliko ovaj red nije prazan sa njega će biti preuzet prvi objekat klase *WebPageData*, i na osnovu njegovog linka biće upućen zahtev za preuzimanje html sadržaja web stranice.

Za potrebe ovog istraživanja preuzimanje html sadržaja vršeno je pomoću javine biblioteke Jsoup. Ova biblioteka je izrazito pogodna za rad sa html sadržajem stranica, jer omogućava selektovanje i parsiranje html elemenata na osnovu css selektora, koje podseća na korišćenje JQuery JavaScript biblioteke, što ovu biblioteku čini vrlo intuitivnom za korišćenje. Pored ovoga Jsoup obezbeđuje sdk koji se u Springboot aplikaciju može dodati na jednostavan način. Glavna klasa ove biblioteke takođe nosi naziv *Jsoup,* i ona omogućava preuzimanje html sadržaja u formi objekta clase *Document*, koja je takođe deo ove biblioteke.

Poslednji korak pri svakom posećivanju stranice jeste procesiranje dokumenta, koje izvršava klasa *DocumentProcessor* na osnovu prosleđenog objekta klase Document, kao i nivoa na kome se nalazi stranica koja je učitana. Kako bi se omogućilo prekidanje procesa nakon završetka pretrage, *ComSisWebCrawlerThread* sadrži metod *isWaitingForUrls*, koji vraća pozitivnu boolean vrednost ukoliko proces pokušava da dobavi sledeći objekat klase *WebPageData* iz reda opsluživanja.

*DocumentProcessor* je klasa zadužena za procesiranje html dokumenata. Ova klasa sadrži statički metod *processDocument* koji kao ulazne parametre prima html dokument, kao i nivo na kome se taj dokument nalazi. Na osnovu prosleđenog nivoa procesor odlučuje o načinu parsiranja dokumenta. Ilustracija ovog procesa odlučivanja prikazana je na slici 2.3.4.



Slika 2.3.4. Procesiranje jsoup Document objekta na osnovu nivoa web stranice

Kao što je moguće videti na slici klasa *DocumentProcessor* sadrži tri metoda za parsiranje html dokumenta. Metod *loadIssues* koristi se prilikom obrade web stranice arhive časopisa. Ovaj metod sa html dokumenata prikuplja sve linkove koji odgovaraju formatu linkova ka brojevima časopisa, i potom na osnovu njih kreira nove instance klase *WebPageData* koje se potom smeštaju u red linkova koji još nisu posećeni. Metod *loadPublications* poziva se kada je dubina na kojoj se stranica nalazi jednaka broju jedan. Ovaj metod se izvršava skoro potpuno isto kao i prethodni. Prilikom pozivanja ovog metoda, sa stranice se parsiraju svi linkovi koji odgovaraju formatu linkova ka stranicama sa naučnim radovima, i na osnovu njih se kao i u prošlom slučaju kreiraju novi objekti tipa *WebPageData* koji se smeštaju u red opsluživanja. Kada je dubina stranice jednaka 2, unutar klase *DocumentProcessor* poziva se metod *extractPublicationData.* Prilikom poziva ovog metoda prvi korak koji se izvršava jeste pronalaženje vrednosti unutar html elemenata na osnovu unapred dodeljenih selektora, nastalim analizom html sadržaja stranica na kojima se nalaze publikacije. Sledeći korak koji se izvršava u okviru ovog metoda jeste kreiranje objekta klase *PublicationPage* na osnovu prikupljenih atributa, nakon čega se kreirani objekat smešta u kolekciju prikupljenih modela deljenu na nivou svih *ComSisWebCrawlerThread*  procesa.

Nakon završetka pretraživanja web stranica svi web crawler procesi bivaju prekinuti od strane objekta klase *ComSisPublicationWebCrawler.* Poslednji korak glavnog procesa jeste vraćanje kolekcije sa objektima klase *PublicationPage,* popunjene modelima prikupljenih sa web stranica. Klasa *PublicationPage* sadrži neprilagođene podatke o naučnom radu, koji se moraju prilagoditi ili dopuniti kako bi bili pogodni za kasnije čuvanje i indeksiranje. Nakon što klasa *PublicationService* od web crawler komponente dobije skup objekata *PublicationPage*, svaki se dodatno obrađuje i pretvara u instancu klase *PublicationData.* Glavni razlog za ovu konverziju jeste činjenica da se podaci sakupljeni prilikom procesa pretrage ne nalaze u formatu koji je pogodan za skladištenje. Najbolji primer ovoga jeste sekcija autora kao i sekcija sa afilijacijama i email adresama autora. Kao što je moguće videti na slici 2.3.5, spisak autora je odvojen od sekcije sa njihovim afilijacijama, te je prostom selekcijom html elemenata stranice nemoguće povezati autore sa odovarajućom afilijacijom.



Slika 2.3.5. Primer spiska autora naučnog rada i njihovih afilijacija

Kako bi se ovaj problem prevazišao korišćena je klasa *AuthorsPageDataParser* koja najpre otklanja ostatke html tagova iz samih vrednosti nakon čega na osnovu spiska autora i njihovih referenci, za svakog autora kreira servisni model koji sadrži podatke o imenu autora, njegovoj email adresi i njegovoj afilijaciji. Drugi proces dopunjavanja podataka jeste konstruisanje url-a za preuzimanje dokumenta kombinovanjem adrese sajta dobijene iz konfiguracije i relativne putanje za preuzimanje dobijene sa stranice prilikom prikupljanja podataka.

Nakon što su svi podaci uspešno prikupljeni i parsirani sledi njihovo čuvanje u bazu podataka. Prilikom implementacije ovog projekta korišćena je PosgreSQL baza podataka. Ova baza sastoji se od dva glavna entiteta: publikacija i autor, i povezne tabele između ova dva entiteta koja povezuje publikacije sa njenim autorima. Glavni razlog korišćenja baze podataka umesto direktnog indeksiranja jeste mogućnost dobavljanja podataka iz više različitih izvora pre samog procesa indeksiranja, kao i skladištenje podataka u slučaju potrebe za ponovnim indeksiranjem podataka.

### Skladištenje podataka

Drugi izazov pri implementaciji sistema za pretraživanje jeste obezbeđivanje adekvatnog načina za skladištenje podataka. Kao što je objašnjeno u teorijskom uvodu ovog rada, najbolji pristup rešavanju ovog problema jeste korišćenje indeks fajlova. Ovaj pristup omogućava brzu, rangiranu pretragu zasnovanu na korisničkom upitu, kojeg je moguće pisati u prirodnom jeziku. Servisni sloj sistema za pretraživanje informacija, kreiranog za potrebe ovog rada sadrži komponentu za indeksiranje i pretragu indeks fajlova. Za potrebe indeksiranja podataka, ovaj deo sistema sadrži dve klase *PublicationIndexService* i *AuthorIndexService,* zadužene za indeskiranje podataka o naučnim radovima i autorima respektivno. Ovi servisi se pokreću od strane orkestratora, koji im inicijalno prosleđuje sve prikupljene podatke iz baze podataka. Važno je napomenuti da ovi servisi ne obavljaju samo indeksiranje, već validiraju ulazne podatke, kreiraju indeks modele i prosleđuju rezultujuću kolekciju modela komponenti za rad sa indeksima koja se nalazi na sloju podataka. Ova komponenta sadrži dve glavne klase koje za odgovornost imaju kreiranje indeks fajlova: *PublicationIndexer* i *AuthorIndexer.* Kako je proces indeksiranja publikacija i njihovih autora gotovo isti, u toku toku ovog poglavlja detaljnije će biti opisan samo proces indeksiranja naučnih radova.

Proces indeksiranja podataka započinje nakon što metod *indexPublications* bude pozvan od strane orkestratora. Ovaj metod kao ulazni parametar prima kolekciju naučnih radova prikupljenih iz baze podataka. Na svakom od ulaznih naučnih radova vrši se validacija u sklopu koje se proverava da li nedostaje neki od podataka o naučnom radu, te se u slučaju da neki od podataka nedosaje posmatrana publikacija izbacuje iz kolekcije, kako bi se u indeks fajlovima nalazili samo naučni radovi sa svim popunjenim podacima. Nakon što se validacija uspešno izvrši, na osnovu validnih publikacija kreiraju se indeks modeli. Ovi modeli podakte o publikaciji predstavljaju na način koji je pogodniji za kreiranje dokumenata koji se smeštaju u indeks fajlove. Nakon što su modeli uspešno kreirani i prikupljeni u listu, aplikacija kreira novu instancu klase *PublicationIndexer* nad kojom se potom poziva metod *indexPublications* kome se kao ulazni parametar prosleđuje lista prikupljenih indeks modela.

Klasa *PublicationIndexer* predstavlja centralnu klasu kada je u pitanju skladištenje podataka o naučnim radovima u indeks fajlove. Kako bi se sam rad sa indeks fajlovima omogućio, za potrebe ovog rada korišćena je Lucene biblioteka za pretragu. Lucene je open-source biblioteka za pretraživanje informacija, koja omogućava lako i brzo dodavanje funkcionalnosti pretraživanja informacija u sve vrste aplikacija. Moderno vreme i velika količina podataka doneli su brojne biblioteke ovog tipa, kao što su Elasticsearch, Google Enterprise Search, Coveo i drugi. Glavni razlog za odabir Lucene biblioteke jeste veoma dobra dokumentacija kao i veliki broj naučnih radova i knjiga sa uputstvima za korišćenje ove biblioteke. Funckionalnosti koje Lucene pruža pri čuvanju podataka jeste kreiranje indeks fajlova na zadatoj putanji, izvršavanje analize ulaznog texta i njihovo smeštanje u kreirane indeks fajlove. Glavna klasa biblioteke Lucene za indeksiranje jeste *IndexWriter.* Konstruktor ove klase kao parametre prihvata objekat klase *Directory* koja predstavlja direktorijum u koji će indeks fajlovi biti smešteni. Pored ovoga, konstruktor klase *IndexWrite* kao parametar prima i konfiguracioni objekat koji određuje način analiziranja teksta, polisu brisanja dokumenata i slično. Nakon što je instanca klase *IndexWriter* uspešno kreirana, njoj se dodeljuju dokumenti koji se potom smeštaju u kreirane indeks fajlove.

Nakon što metod *indexPublications* bude pozvan sa ulaznom kolekcijom indeks modela za svaki od modela se kreira posebna instanca klase *org.apache.lucene.document.Document* koja se potom popunjava vrednostima iz indeks modela publikacije. Ovaj proces prikazan je na listingu 2.4.1.



Slika 2.4.1. Priprema dokumenta za indeksiranje

Metod *populateDocumentWithPublicationData* kao parametre prihvata indeks model publikacije i objekat klase *Document* koji predstavlja dokument koji će biti smešten u indeks fajl. Dokumentu se dodeljuju vrednosti pozivom metoda *add* nad instacom dokumenta, koja kao ulazne parametre prihvata naziv polja koje će dokument sadržati, vrednost tog polja, kao i enum koji označava da li se vrednost polja takođe čuva u indeksu. Kako je moguće videti na listingu 2.4.1. polja koja se čuvaju u indeks fajlovima su naslov, id, apstrakt, godina objavljivanja, ključne reči, tekst publikacije kao i njeni autori. Sva ova polja u kasnijoj fazi ovog rada biće dostupna za pretragu. Moguće je uočiti da se prilikom dodavanja tekstualnog sadržaja publikacije koristi *DocumentService* kako bi se dobavio sadržaj. Ovaj pristup odabran je umesto skladištenja tekstualnog sadržaja unutar indeks modela jeste činjenica da bi zbog velikog broja ulaznih modela došlo do nedostatka memorije za čuvanje kompletnog tekstualnog sadržaja. Takođe, na listingu je moguće videti i da je spisak autora publikacije serijalizovan, što omogućava pretragu naučnih radova po imenima autora, kao i po njihovim email adresama ili afilijacijama.



Slika 2.4.2. Indeksiranje podataka o naučnim radovima

Nakon što se dokument popuni od strane ovog metoda, on biva dodat u instancu klase *IndexWriter* koja ga potom upisuje u kreirani indeks fajl. Ovaj proces se završava kada se dokumenti kreiraju za sve zadate modele i nakon što budu smešteni u indeks fajlove. Celokupan process indeksiranja podataka prikazan je na listingu 2.4.2. Nakon što se proces indeksiranja uspešno završi, veoma je važno zatvoriti instancu klase *IndexWriter* kako bi se oslobodili indeks fajlovi i na taj način se omogućilo kasnije čitanje iz njih. Nakon što se metod *indexPublications* uspešno izvrši indeks fajlovi za pretragu naučnih radova su sačuvani i spremni za pretragu.

### Pretraživanje i rangiranje podataka

Sledeći korak u implementaciji ove aplikacije jeste kreiranje komponente za rangirano pretraživanje podataka. Kako bi se podaci o naučnim radovima i njihovim autorima uspešno pretražile, aplikacija pruža brojne metode kontrolera koji klijentu omogućavaju da uspešno dobavi podatke na osnovu zadatog upita. Svaki od metoda kontrolera odgovara tačno jednom polju pretrage, i svaki od njih prima dva ulazna parametra, od kojih prvi predstavlja sam upit u obliku teksta, dok trugi predstavlja maksimalni broj rezultata pretrage. Drugi parametar ovih metoda je opcioni te će se ukoliko klijent ne pruži ovaj podatak iskoristiti podrazumevana vrednost uzeta iz konfiguracije. Za potrebe pretrage naučnih radova koriste se metodi klase *PublicationSearchController,* dok se za potrebe pretrage autora koristi *AuthorSearchController.* Ove klase kao povratnu vrednost imaju kolekciju publikacija odnosno autora poređanu prema nivou sličnosti sa zadatim upitom. Kako je proces pretrage za naučne radove i autore gotovo isti, u toku ovog poglavlja detaljno će biti opisana samo implementacija procesa pretrage naučnih radova na osnovu korisničkog upita.

Nakon što se neki od metoda objekta klase *PublicationSearchController* pozove, on delegira poziv ka instanci klase *PublicationSearchService* koja sadrži metode za pripremu upita za pretragu, kao i mapiranje dobijenih rezultata na odgovarajuće modele koji će biti čitljivi klijentu. Nakon što se uputi poziv metodu klase *PublicationSearchService,* interno se poziva metod *searchPublications,* kome se kao parametri prosleđuju upit, maksimalni broj rezultata pretrage i polje po kome pretraga treba da se vrši. Ovaj metod potom delegira poziv instanci klase *PublicationIndexSarcher,* te vraća dobijene rezultate pretrage.

*PublicationIndexSarcher* i *AuthorIndexSarcher* su dve centralne klase za pretraživanje podataka iz indeks fajlova. Ove klase se nalaze u sloju podataka, i pretražuju indeks fajlove korišćenjem Lucene biblioteke. Prilikom inicijalizacije ove klase, u njenom konstruktoru se kreira instaca klase *IndexSearcher* na osnovu putanje do direktorijuma u kome se nalaze indeks fajlovi. Ovaj objekat kasnije se koristi za pretraživanje podata iz indeks fajlova. Glavni metod klase *PublicationIndexSarcher* jeste metod *search* koji kao ulazne parametre prihvata tekstualni upit, naziv polja koje se pretražuje, kao i maksimalni broj vraćenih rezultata.



Listing 2.5.1. Pretraga dokumenata iz indeks fajlova

Kompletan proces pretrage podataka iz indeks fajlova moguće je videti na listingu 2.5.1. Iz priloženog listinga može se videti da je prvi korak u pretraživanju kreiranje objekta klase *Query*. Ova klasa je deo Lucene biblioteke i kreira se korišćenjem instanceklase *QueryParser,* čijem se konstruktoru prosleđuje naziv polja za pretragu, kao i objekat koji označana način analiziranja teksta, te se nad ovom instancom poziva metod *parse* kome se prosleđuje tekstualni upit dobijen od strane klijenta. Nakon što se objekat klase *Query* uspešno kreira, on se zajedno sa maksimalnim brojem željenih rezultata prosleđuje metodu *search* klase *IndexSearcher*. Povratna vrednost ove pretrage je niz objekata klase *ScoreDoc* koja sadrži dokument koji je pronađen prilikom pretrage, kao i njegovu vrednost skora. Lucene biblioteka podrazumevano vraća niz ovih objekata prema sortiran prema vrednosti skora. Maksimalan broj elemenata ovog niza odgovara broju koji je prosleđen prilikom kreiranja *Query* objekta. Nakon obavljene pretrage i dobijanja rezultata, svaki od dobijenih elemenata parsira se u indeks model. Kao što je moguće videti na listingu 2.5.2 algoritam za parsiranje indeks modela na osnovu dokumenta je inverzan algoritmu za kreiranje dokumenta na osnovu indeks modela. Takođe, na izvorni kod prikazan na listingu pokazuje da se prilikom parsiranja liste autora poziva metod getAutorsList, unutar koga se u cilju dobijanja kompletne liste autora publikacije vrši deserijalizacija teksta sačuvanog prilikom indeksiranja podataka. Na ovaj način se ostvaruje to da se podaci ne izgube prilikom pretrage, nego se vraćaju u istom formatu u kome su bili i pre samog čuvanja u indeks fajlove.



Listing 2.5.2. Formiranje indeks modela na osnovu pronađenog dokumenta

Nakon što su modeli uspešno kreirani sledi formiranje liste dobijenih modela i vraćanje rezultujuće liste. Kao što je to bio slučaj i pri procesu indeksiranja, veoma je važno da se na kraju procesa pretraživanja zatvori instanca klase za pretragu indeks fajlova, kako bi se fajlovi oslobodili za kasniju ponovnu pretragu. Krajnji korak ove pretrage jeste mapiranje servisnih modela publikacija u modele koji će biti čitljivi za klijenta, njihova serijalizacija i vraćanje http odgovora koji u svom sadržaju poseduje niz rezultujućih publikacija u json formatu.

### Dodaci u pretraživanju podataka

Pored osnovnog pretraživanja na osnovu korisničkog upita, aplikacija kreirana za potrebe ovog rada omogućava i dodatke u pretrazi podataka, kao i računanje osnovnih statistika nad prikupljenim podacima. Ove operacije predstavljaju dokaz o konceptu kombinovanja pretrage podataka iz indeks fajlova sa drugim vrstama pretrage, kao i različitim vrstama organizovanja dobijenih rezultata pretrage.

Jedna od najvažnijih dodatnih funkcionalnosti koje aplikacija nudi, jeste pretraga svih objavljenih radova za odabranog autora. Veoma je važno napomenuti da se ova funkcionalnost ne ograničava samo na naučne radove objavljene u ComSIS časopisu, već se prikupljaju radovi bez obzira na mesto njihovog objavljivanja. Kako bi se ova funkcionalnost ostvarila, za pretragu je potrebno koristiti servis koji ima pristup ovakvim podacima, te je za ove potrebe korišćena baza podataka pod nazivom „**dblp Computer Science Bibliography**”. Dblp predstavlja najveću poznatu bazu sa podacima o naučnim radovima iz područja računarskih nauka i njihovim autorima, sa gotovo 5 miliona naučnih radova i 2.5 miliona poznatih autora. Ovo skladište podataka javno je dostupno korišćenjem dblp web servisa koji nudi endpointe za pretragu naučnih radova, kao i pretragu autora. Za potrebe pretrage dblp servisa aplikacija kreirana za potrebe ovog rada sadrži servisnu klasu *DblpService*.



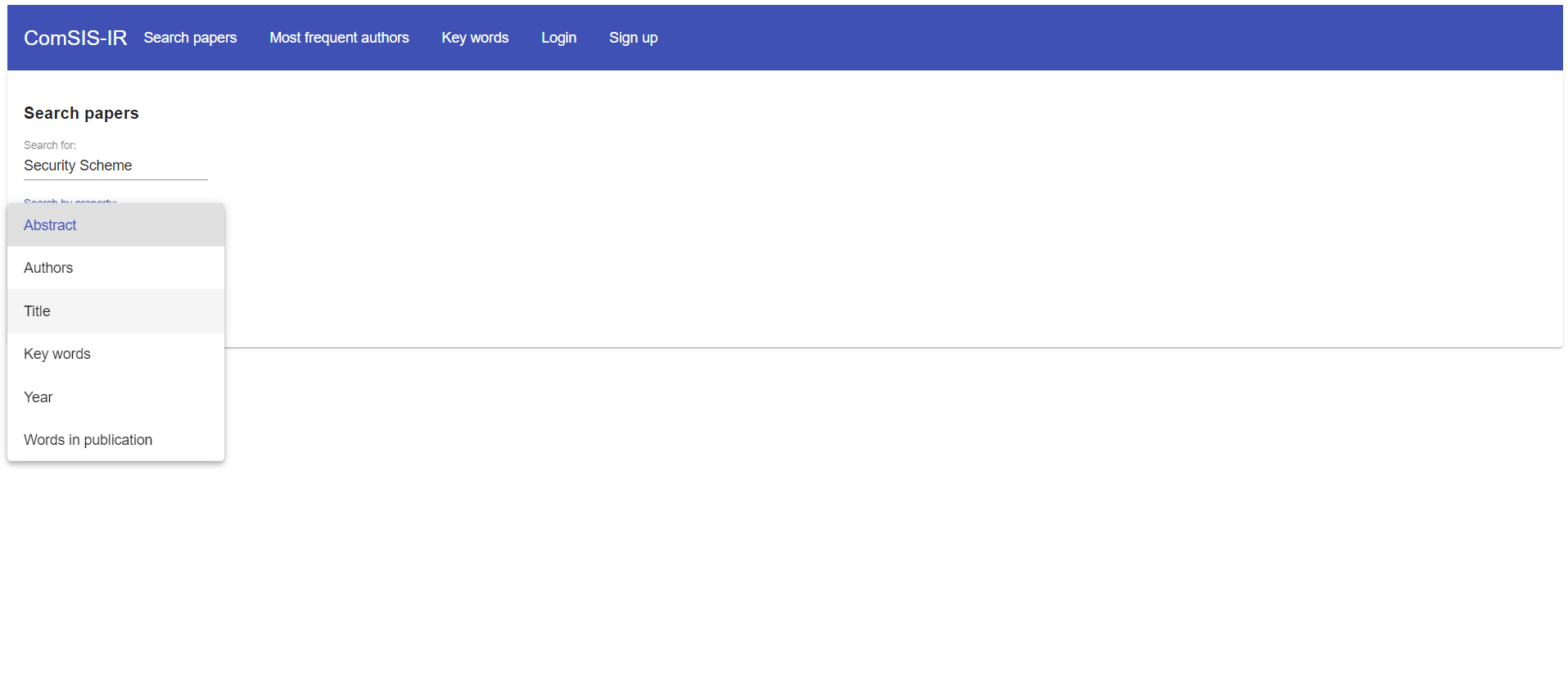
Listing 2.5.1. Dobavljanje podataka o publikacijama korišćenjem dblp web servisa

Glavni metod klase *DblpService* je *findPublicationsForAuthor* koji kao ulazni parametar prihvata ime autora i kao odgovor vraća kolekciju publikacija za zadatog autora. Kompletan rad ovog metoda prikazan je na listingu 2.6.1. Kako bi se poziv dblp web servisu izveo na pravilan način, unutar metoda *findPublicationsForAuthor* kreira se objekat klase *DblpSearchPreferences* koji određuje osobine zahteva koji će biti upućen ka dblp servisu. Nakon što je ovaj objekat uspešno kreiran, on se prosleđuje *buildRequestUrl* klase *DblpRequestBuilder* ova klasa omogućava da se na osnovu prosleđenog objekta konstruiše url za http get zahtev koji se potom upućuje na dblp web servis. Nakon što se ovaj zahtev obradi i vrati pronađene rezultate, oni bivaju deserijalizovani u niz objekata klase *DblpPublication* koji se potom vraća klijentu aplikacije. Ova funkcionalnost se koristi od strane klijentske aplikacije kako bi se za autora definisanog prikupljanjem podataka iz ComSIS časopisa preuzeli podaci o radovima objavljenim na drugim mestima, te se na taj način proširuju dostupni podaci, bez potrebe za dodatnim web pretraživanjem.

## Demonstracija i testiranje

Kako bi se dokazao rad aplikacije implementirane za potrebe ovog rada, treće poglavlje se bavi prikazom izgleda aplikacije i njenog rada. Kao što je rečeno u prethodnom poglavlju za potrebe kreiranja ponašanja klijentske strane aplikacije korišćeno je Javascript razvojno okruženje Angular, dok je za izgled aplikacije zadužena Angular biblioteka pod nazivom „Angular Material“.

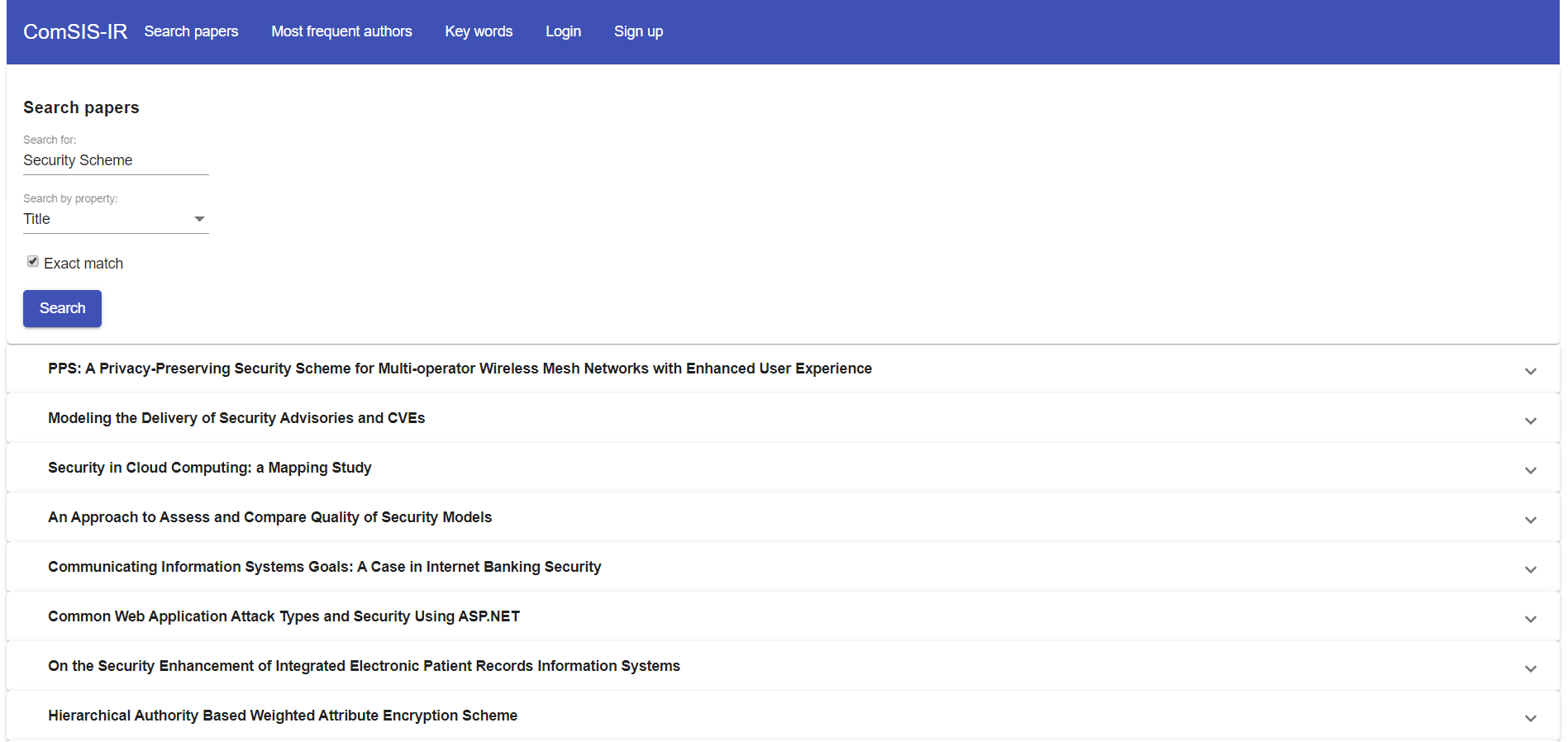
Najvažniji korisnički slučaj jeste pretraživanje publikacija prema korisničkom upitu. Ovu funkcionalnost moguće je pokrenuti klikom na dugme „Search Papers“ u navigacionoj sekciji stranice. Klikom na ovo dugme prikazuje se meni za pretragu naučnih radova, prikazan na slici 3.1.1.



Slika 3.1.1. Stranica za pretraživanje naučnih radova

Kako bi se izvršila pretraga naučnih radova potrebno je uneti ključne reči za pretragu. Takođe je potrebno odabrati polje po kome će se pretraga vršiti. Polja po kojima je moguća pretraga su: apstrakt rada, autori, naslov, ključne reči, godina objavljivanja rada kao i reči u samom tekstu naučnog rada. Pored ovoga, korisnički grafički interfejs nudi i opciju „exact match“ koja označava da li su se reči iz upita moraju pojaviti u dokumentu u tačnom obliku. Ukoliko se ova opcija eksplicitno ne promeni ona je podrazumevano odabrana. Nakon što je forma za pretraživanje popunjena, zahtev za pretragu moguće je poslati klikom na dugme „search“. Kao što je moguće videti na slici 3.1.1 za potrebe ove demonstracije izvršena je pretraga fraze „Security Scheme“. Kao polje za pretraživanje iskorišćen je naslov rada, te je opcija „exact match“ odabrana. Nakon klika na dugme „search“ serverski deo aplikacije započinje pretragu nad indeks fajlovima i vraća dobijene rezultate koji se potom izlistavaju u ovkiru stranice za pretragu.

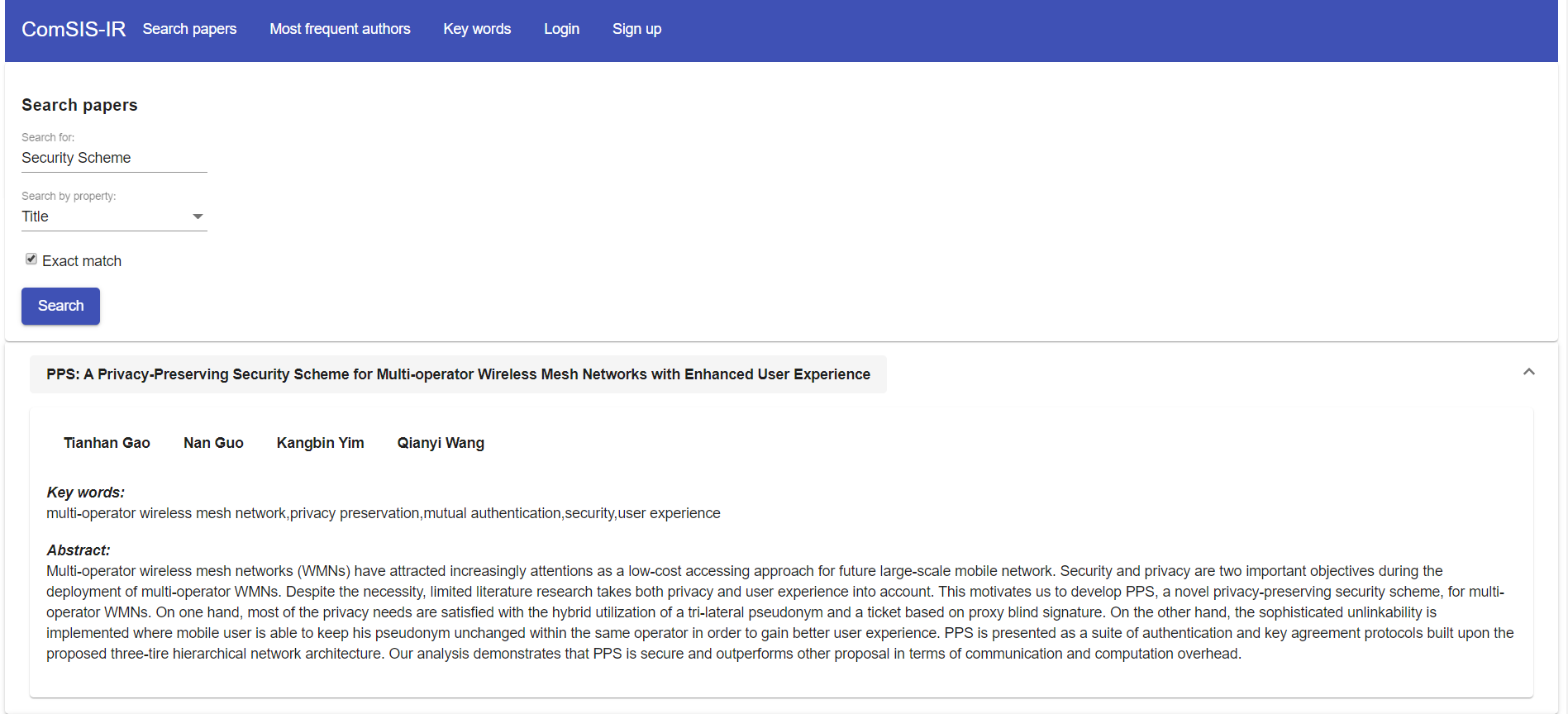
Slika 3.1.2. Lista naučnih radova pronađenih na osnovu zadatih parametara pretrage



Na slici 3.1.2 je moguće videti listu rezultata pretrage fraze „Security Scheme“ u naslovu naučnog rada. Ukoliko se obrati pažnja na rezultate pretrage moguće je videti da samo prvi naučni rad iz liste sadrži obe reči fraze pretrage u svom naslovu. Razlog ovoga jeste činjenica da u kolekciji prikupljenih radova zaista postoji samo jedan naučni rad koji sadrži obe reči u svom naslovu. Ipak, kako postoje mnogi naučni radovi koji u svom naslovu sadrže jednu od dve reči iz upita, postoji verovatnoća da će neki od radova biti značajan za pretragu. Ovo predstavlja glavnu prednost koju nudi sistem koji za rangirano pretraživanje u odnosu na klasično pretraživanje korišćenjem baze podataka. Ipak važno je napomenuti da je naučni rad koji sadrži obe reči iz upita u svom naslovu prvi na listi rezultata zbog najvećeg nivoa preklapanja sa korisničkim upitom, odnosno zbog najvećeg izračunatog skora.

Kao što je moguće videti na slici 3.1.2 svaki od rezultujućih naučnih radova nalazi se u zasebnoj sekciji. Ove sekcije se mogu proširiti klikom na ikonicu strelice, čime se prikazuje kompletan panel sa sažetim podacima o naučnom radu, što je prikazano na slici 3.1.3.

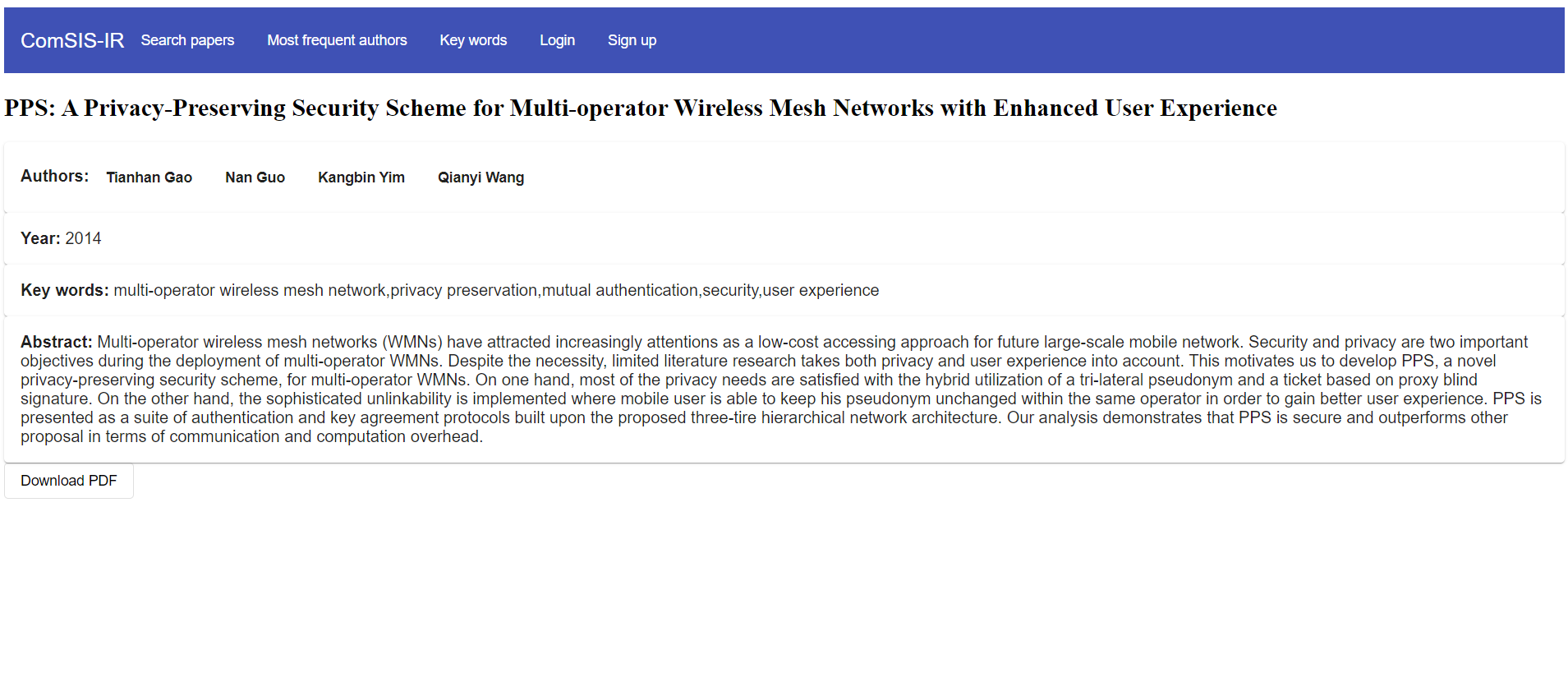
Slika 3.1.3. Panel sa detaljima o naučnom radu u sklopu stranice za pretraživanje



Nakon što se panel sa detaljima naučnog rada proširi moguće je videti da on pored samog naslova rada sadrži i druge podatke, kao što su autori rada, ključne reči i apstrakt rada. U sklopu panela sa detaljima o naučnom radu, naslov rada predstavlja link koji vodi do stranice sa detaljnijim prikazom naučnog rada (Slika 3.1.4), dok imena autora rada predstavljaju linkove ka stranicama sa detaljnim prikazom podataka o autoru naučnog rada.

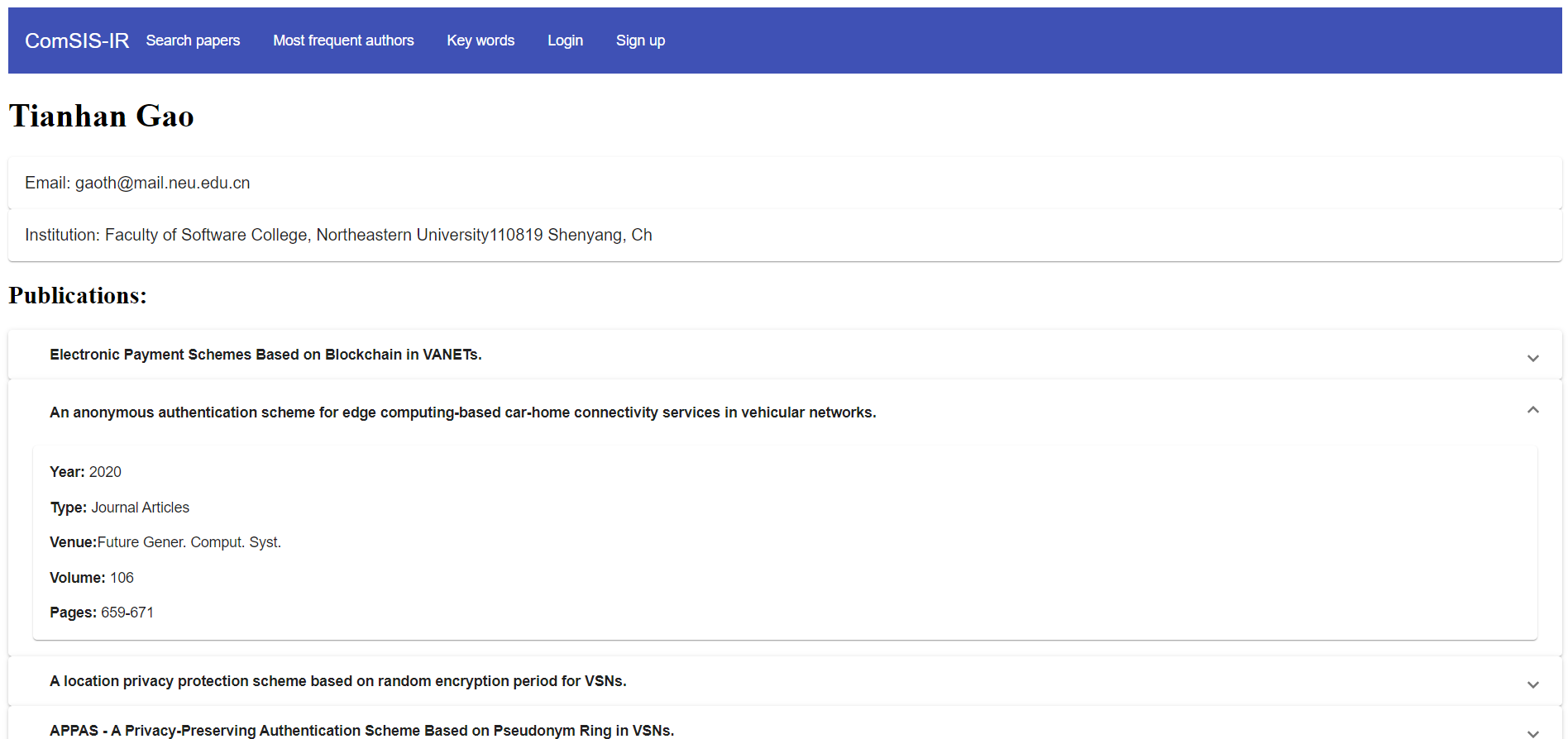
Panel sa sažetim podacima o naučnom radu omogućava korisniku površni pregled dokumenta u svrhu brzog i jednostavnog donošenja odluke o važnosti samog dokumenta za izvršenu pretragu. Ovime se korisniku omogućava izbegavanje menjanja konteksta u svrhu detaljnog pregleda potencijalno nevažnog dokumenta, te se na taj način ostvaruje bolje korisničko iskustvo.

Slika 3.1.4. Stranica sa detaljima o naučnom radu



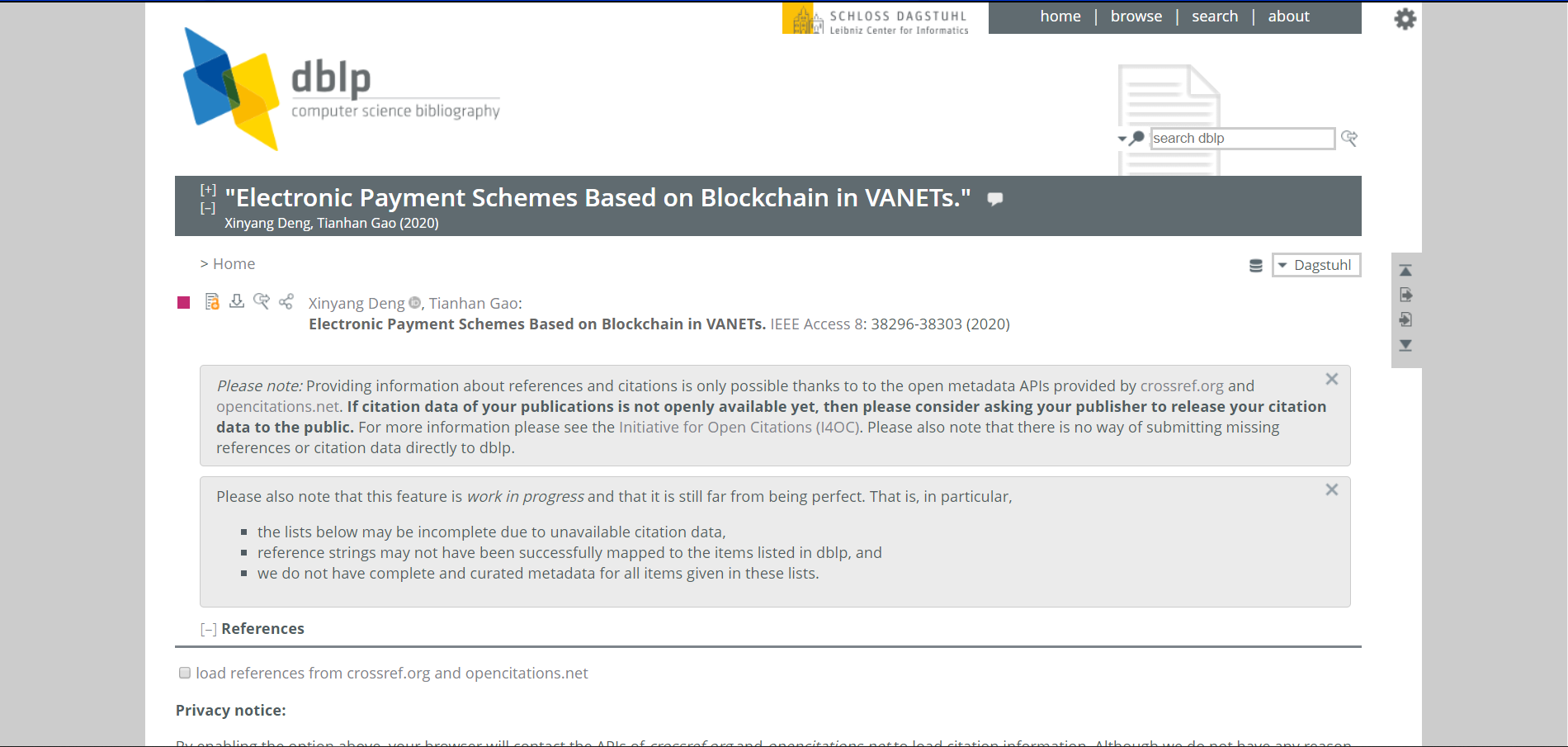
Na slici 3.1.4 prikazan je izgled stranice za detaljni pregled naučnog rada. Ova stranica omogućava korisniku pregled svih podataka o naučnom radu prikupljenih sa web stranica nučnog časopisa, kao što su autori rada, godina objave rada, ključne reči i apstrakt rada. Pored ovoga aplikacija nudi i mogućnost preuzimanja kompletnog naučnog rada u obliku pdf dokumenta, sa zvaničnog ComSIS sajta. Na ovaj način korisniku se omogućava da pregleda detalje naučnog rada i preuzme njegov sadržaj ukoliko ga smatra važnim na osnovu ponuđenih podataka. Važno je dodati da imena autora na ovoj stranici predstavljaju linkove ka stranicama koje sadrže detaljne podatke o autorima naučnog rada. Za potrebe demonstracije rada ove aplikacije, otvorena je stranica sa detaljima jednog od autora naučnog rada sa slike 3.1.4. klikom na link sa njegovim imenom.

Slika 3.1.5. Stranica sa detaljima autoru



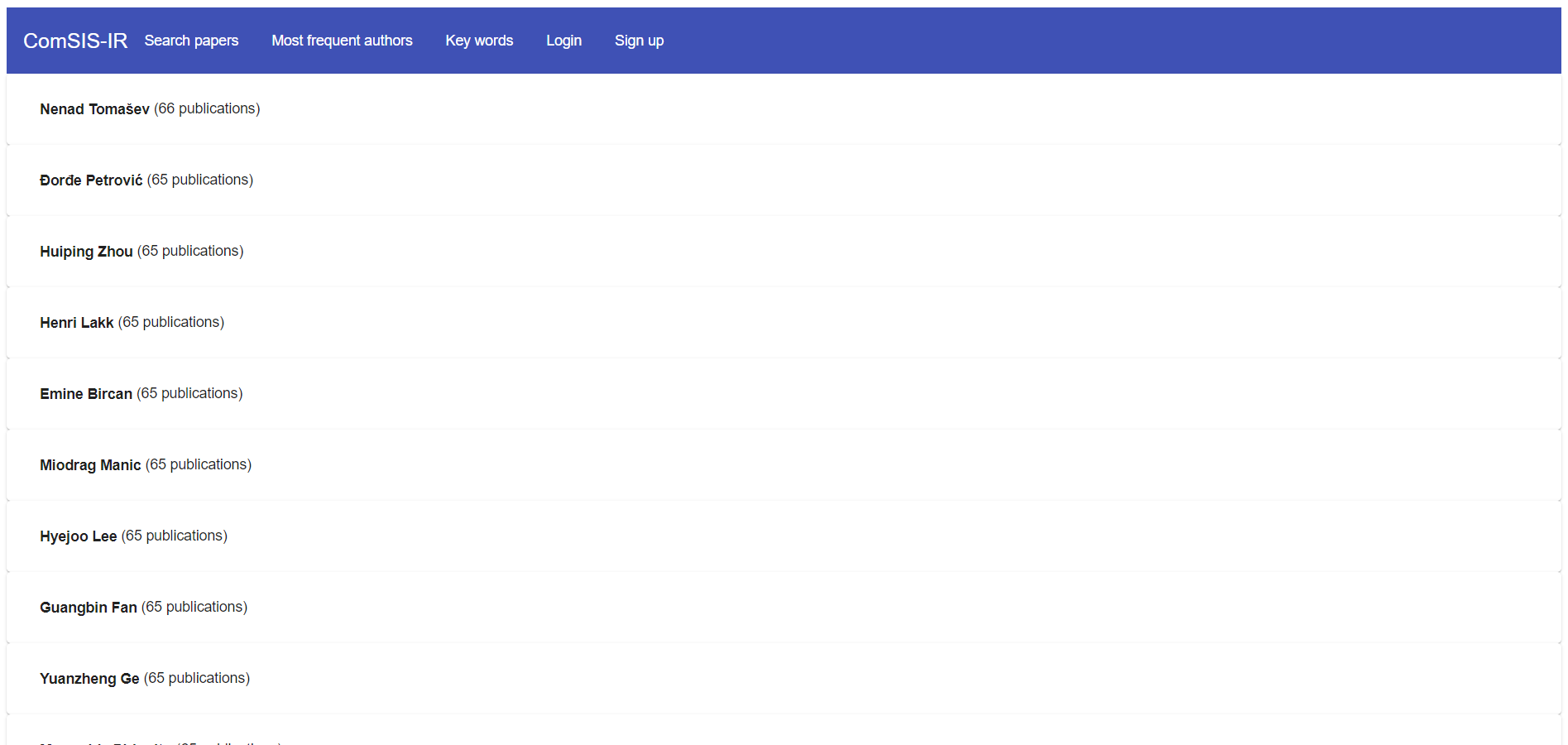
Na slici 3.1.5 prikazan je izgled stranice sa detaljima autora. Pored standardnih podataka o autoru kao što su ime, email adresa i institucija u kojoj naučnik deluje, ova stranica nudi i podatke o naučnim radovima na kojima je autor radio. Pretraga naučnih radova za određenog autora vrši se korišćenjem web servisa dblp, te je lista vraćenih rezultata prikazana na stranici sortirana opadajuće prema godini objavljivanja naučnog rada. Svaki od naučnih radova sa ove stranice prikazan je u zasebnoj sekciji koja je proširiva klikom da ikonicu strelice u desnom uglu sekcije. Proširivanjem sekcije sa naučnim radom na uvid se dobijaju detalji o publikaciji kao što je godina objave, tip i drugo. Takođe, klikom na naslov naučnog rada moguće je otvoriti zvaničnu dblp stranicu sa svim detaljima o tom naučnom radu. Na ovoj stranici moguće je pristupiti svim podacima o naučnom radu, kao i njegovim autorima prikupljenih od strane dblp organizacije. Izgled ove stranice moguće je videti na slici 3.1.6.

Slika 3.1.6. Dblp stranica sa detaljima o naučnom radu

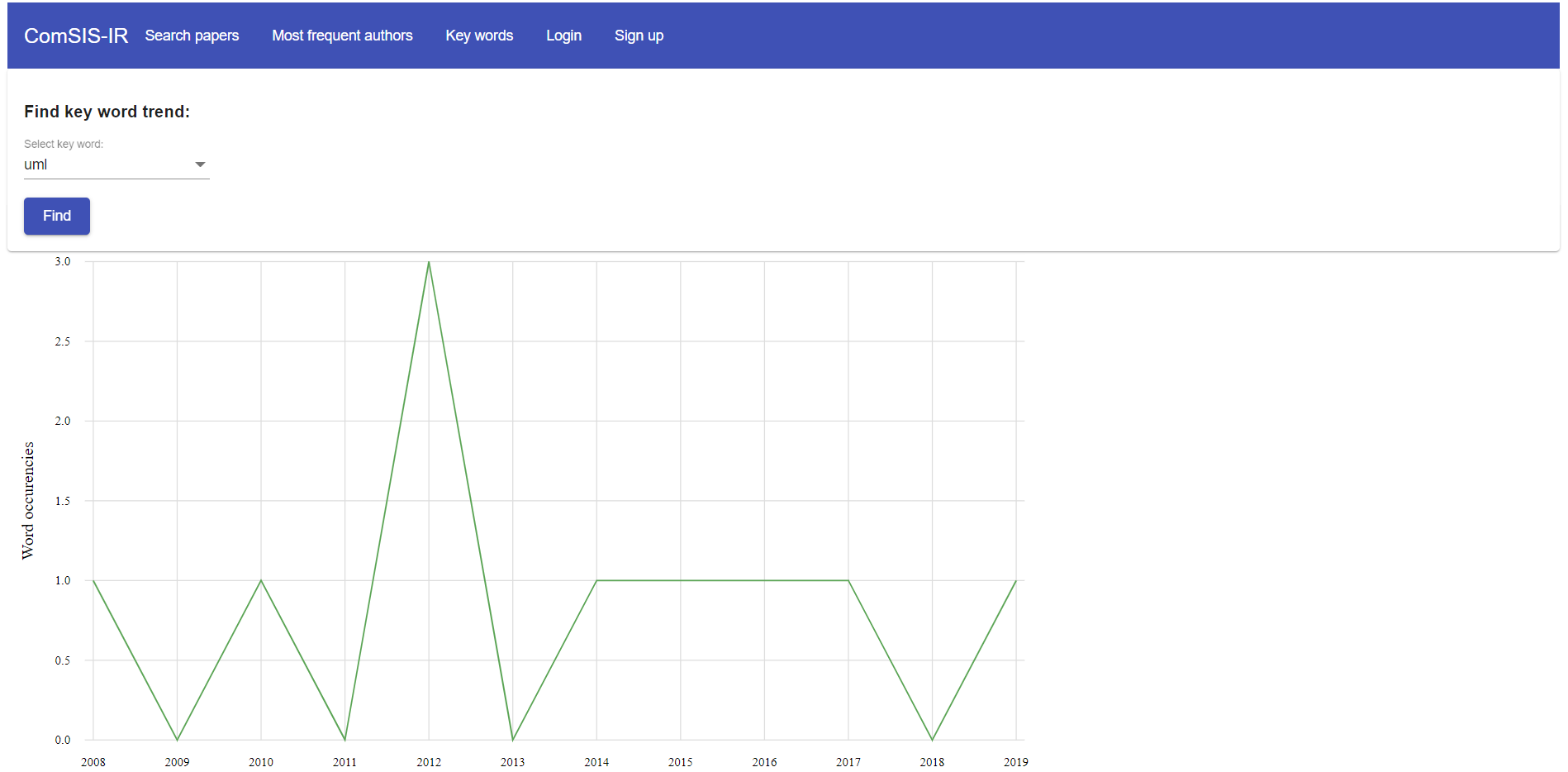


Pored mogućnosti pretraživanja naučnih radova na osnovu korisničkog upita, ova aplikacija nudi i nekoliko dodatnih funkcionalnosti. Jedna od ovih funkcionalnosti jeste pronalaženje autora sa najvećim brojem objavljenih publikacija. Ovoj funkcionalnosti moguće je pristupiti pritiskom na dugme „Most frequent authors“, koje se nalazi u sklopu navigacione komponente na vrhu stranice. Klikom na ovo dugme otvara se stranica sa spiskom autora, prikazana na slici 3.1.7. Prilikom otvaranja ove stranice, serverskom delu aplikacije šalje se zahtev za dobavljanje svih autora prikupljenih naučnih radova, poređanih prema broju radova na kojima su autori radili. Rezultat ove pretrage takođe je moguće videti na slici 3.1.7. Važno je napomenuti da se broj radova na kome su autori radili odnosi samo na radove koji su objavljeni u ComSIS naučnom časopisu, odnosno radove koji se nalaze u indeks fajlovima. Klikom na ime autora korisnik će biti redirektovan na stranicu sa detaljima o autoru, koja je prikazana ranije u ovom poglavlju.

Slika 3.1.7. Stranica sa listom najčešćih autora



Još jedna od funkcionalnosti koje aplikacija nudi jeste pregled ključnih reči koje se pojavljuju u prikupljenim naučnim radovima. Ova funkcionalnost daje korisniku grafički prikaz trenda zadate teme iz oblasti računarskih nauka. Klikom na dugme „Key words“ lociranog u navigacionom meniju, moguće je otvoriti stranicu prikazanu na slici 3.1.8.

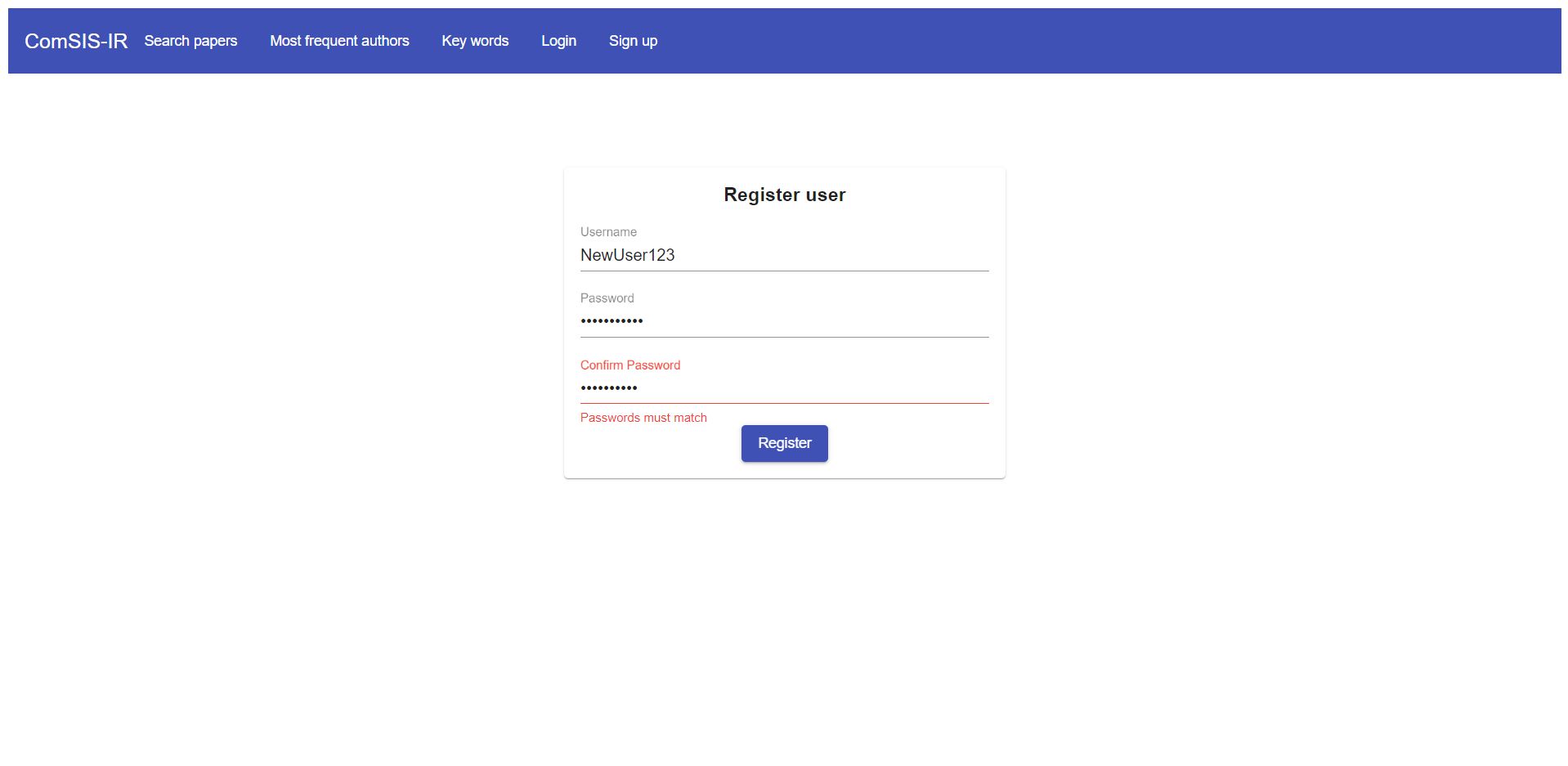


Slika 3.1.8. Stranica sa podacima o najčešćim ključnim rečima

Prilikom otvaranja ove stranice, sa servera su učitane sve ključne reči koje se pojavljuju u naučnim radovima lociranim u skladištu podataka. Lista ovih reči vraćena je sa servera sortirana opadajuće prema broju dokumenata u kojima se ove reči pojavljuju, odnosno prema dokument frekvenciji ovih reči. Nakon što su ključne reči pronađene i poređane prema svojoj učestalosti, one se smeštaju u element za selekciju. Odabirom ključne reči i klikom na dugme „Find“ na server se šalje novi zahtev, kojim se pokreće operacija za računanje broj pojava zadate ključne reči prema godinama objavljivanja rada. Nakon što se ovo računanje uspešno izvrši, server vraća ogovor na osnovu koga se genriše grafik funkcije broja pojavljivanja ključne reči u radu prema godini objave, gde početna i krajnja godina na grafiku predstavljaju godine kada se ključna reč prvi odnosno poslednji put pojavila u prikupljenim naučnim radovima.

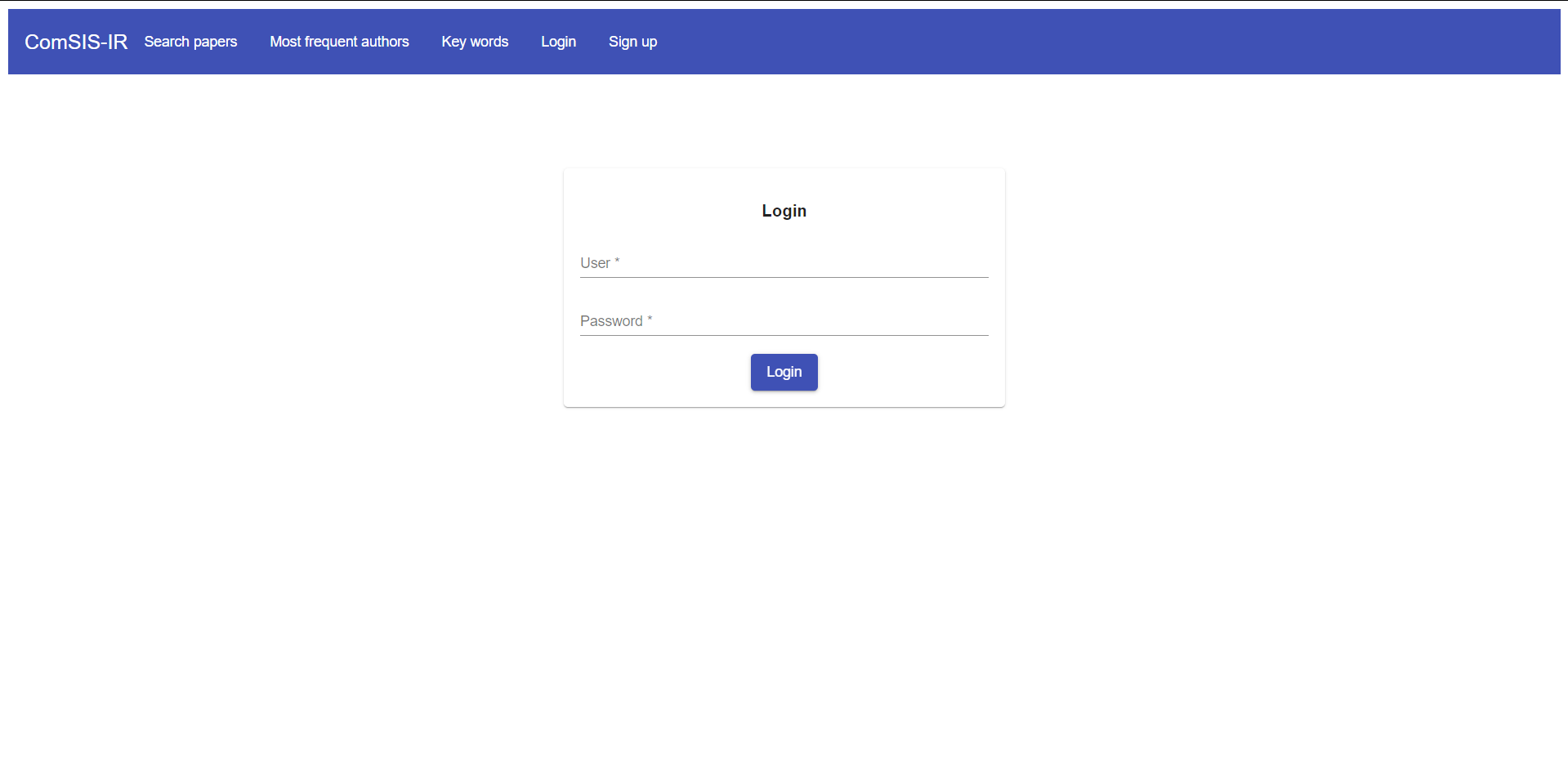
Funkcionalnosti koje su opisane u dosadašnjem toku rada, moguće je izvesti bez bilo kakve autentikacije. Ipak, aplikacija nudi i neke od funkcionalnosti koje su dostupne samo registrovanim korisnicima. Klikom na dugme „Sign up“ navigacionog menija, moguće je doći na stranicu za registraciju korisnika, koja je prikazana na slici 3.1.9.

Slika 3.1.9. Stranica za registraciju korisnika



Kao što je moguće vidti na slici, u svrhu registrovanja korisnik je obavezan da unese svoje korisničko ime, koje pritom mora biti jedinstveno, kao i korisničku lozinku. Pored ovoga korisnik je obavezan i da ponovi unetu lozinku, kako ne bi došlo do unošenja pogrešne lozinke. Važno je napomenti i da se prilikom unošenja imena i lozinke vrši validacija, te da će u slučaju pogrešno unetih podataka korisniku biti prikazana poruka o grešci. Nakon što su podaci uspešno uneti, pritiskom na dugme „Register“, šalje se zahtev za registraciju korisnika, pri čijem izvršenju se ispisuje odgovarajuća poruka i vrši redirekcija na stranicu za prijavu korisnika, koja je prikazana na slici 3.1.10.

Slika 3.1.10. Stranica za prijavljivanje korisnika

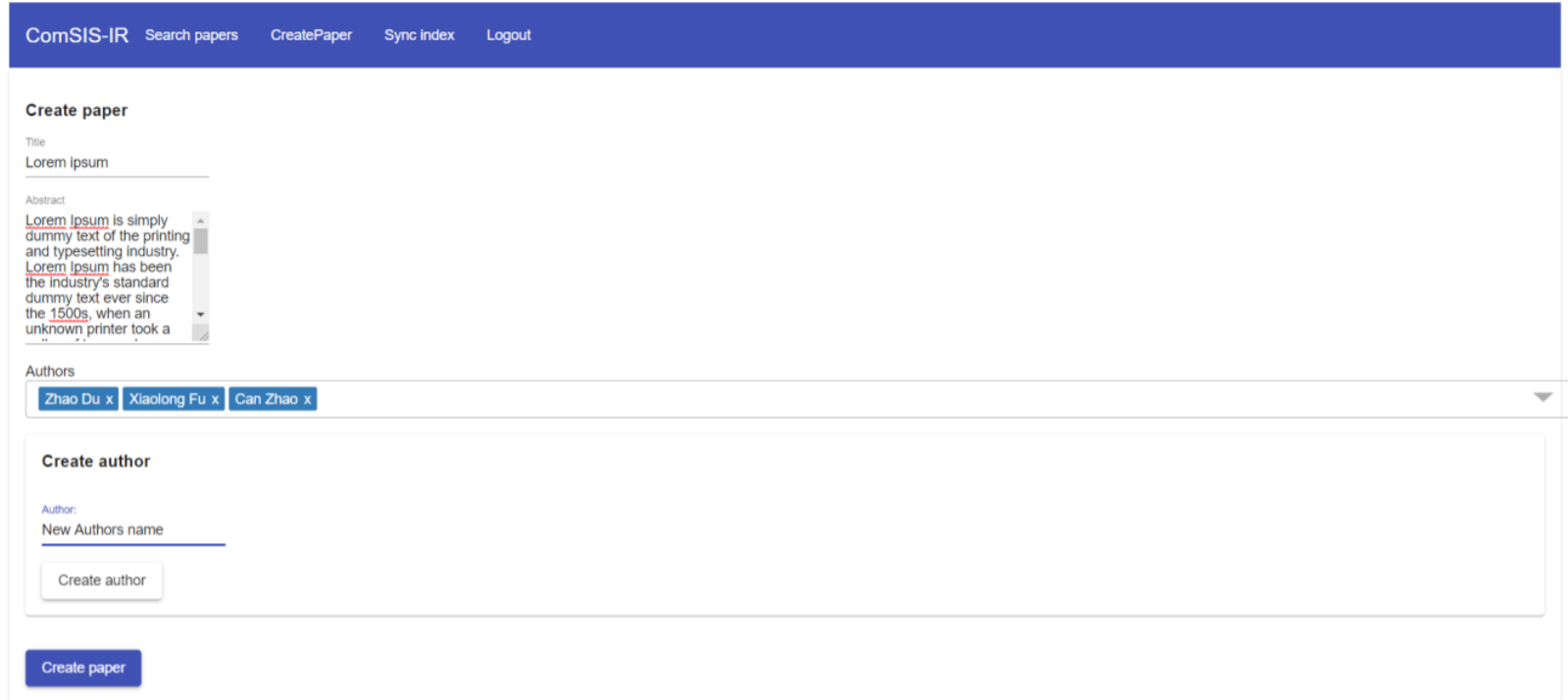


Stranica za prijavljivanje korisnika sadrži jednostavnu formu koja pri svom popunjavanju i predaji klikom na dugme „Login“ šalje zahtev za prijavu korisnika. U slučaju pogrešno unetih podataka, na ekranu se prikazuje odgovarajuća poruka. Nakon uspešnog prijavljivanja, korisnik dobija mogućnost za izvršavanje dve dodatne operacije.

Jedna od funkcionalnosti koje se omogućavaju korisniku jeste sinhronizovanje indeksa, odnosno preuzimanje kompletne kolekcije podataka iz PostgreSQL baze podataka, i ponovno kreiranje indeks fajlova na osnovu ovih podataka. Ova operacija je potrebna kako bi indeks fajlovi uvek posedovali najsvežije dostupne podatke, u slučaju da je u bazi podataka došlo do bilo kakvih izmena.

Druga funkcionalnost koju aplikacija nudi prijavljenim korisnicima jeste unošenje novih publikacija u sistem pretrage. Klikom na dugme „Create paper“ moguće je otvoriti stranicu za dodavanje novih publikacija (Slika 3.1.11). Ova stranica sadrži formu za unos podataka o naučnom radu, kao što su naslov rada, godina objave, apstrakt i drugo. Pored ovoga forma za unos novog naučnog rada sadrži i mogućnost za dodavanje kompletnog naučnog rada u obliku pdf dokumenta. Takođe, prilikom unosa podataka o autorima rada, korisnik ima mogućnost odabira nekog od postojećih autora, kao i unos podataka o autoru koji još ne postoji u skladištu podataka. Kako bi se podaci o autorima uspešno uneli, podatke o autoru je potrebno uneti u okviru sekcije pod nazivom „Create author“, nakon čega se podaci potvrđuju klikom na dugme „Create author“. Nakon što su podaci o autoru uspešno uneti, ime autora će postati dostupno za selekciju u okviru liste dostupnih autora. Kompletiranje unosa podataka o naučnom radu vrši se klikom na dugme „Create paper“

Slika 3.1.11. Stranica za dodavanje nove publikacije



Nakon što su podaci o naučnom radu uspešno uneti, kako bi oni postali dostupni za pretragu, najpre je potrebno izvršiti sinhronizaciju, odnosno ponovno indeksiranje podataka iz baze. Ovo je ujedno i poslednja funkcionalnost koju prva verzija aplikacije za pretragu naučnih radova nudi, te se ovom funkcionalnošću zaključuje demonstracija rada aplikacije kao i ovo poglavlje.

## Zaključak

Tokom prethodna tri poglavlja prikazani su osnovni teorijski koncepti pretraživanja informacija, osnovne razlike ovog pristupa u odnosu na korišćenje standardnih relacionih baza podataka, kao i osnovne pogodnosti koje ovakav pristup pretraživanju nudi. Objašnjena je i tehnika za pravilno rangiranje podataka prilikom pretraživanja, koja za cilj ima postizanje što boljeg korisničkog iskustva. Pored ovoga, ovaj rad nudi dizajn i implementaciju aplikacije za indeksiranje i pretraživanje naučno-istraživačkih časopisa otvorenog pristupa, koja služi kao dokaz koncepta ovakvog pristupa u pretraživanju informacija. Drugo poglavlje ovog rada pružilo je uvid u arhirekturu ove aplikacije, kao i uvid u izvorni kod ključnih delova ove aplikacije. Tokom trećeg poglavlja ovog rada predstavljene su funkcionalnosti koje ova aplikacija nudi, kao i kompletan proces korišćenja ove aplikacije.

Kao sledeći korak u razvoju ovog koncepta, svakako se najpre nameće proširenje baze podataka obradom većeg broja naučno-istraživačkih časopisa. Ovaj korak bi pre svega zahtevao pronalazak odgovarajućih časopisa otvorenog pristupa, ali i dodatnu analizu u cilju osposobljavanja web crawler komponente za prikupljanje i analizu web stranica drugačijih oblika i struktura.

Proširenje koje bi zasigurno ostvarilo veći uticaj jeste uvođenje „Relevance feedback“ koncepta. Ovaj koncept zasniva se na pretrazi koja kao svoj rezultat ima inicijalnu kolekciju dokumenata. Ova kolekcija potom biva ocenjena od strane korisnika, tako što korisnik određuje važne odnosno nevažne dokumente unutar kolekcije. Nakon ovog ocenjivanja, sistem na osnovu korisničkih ocena računa novu, unapređenu kolekciju rezultata koja je značajnija za korisnika. Jedan od glavnih pristupa ovakvom načinu pretrage jeste korišćenje Rocchio algoritma za povratnu informaciju o važnosti dokumenata u vector-space modelu. Ovaj algoritam zasniva se na korišćenju pozitivnih i negativnih ocena rezultata pretrage pri računanju skora, čime se postiže oblikovanje rezultata pretrage na način koji odgovara korisniku. Uvođenjem ovog principa, moguće je dodatno unaprediti brzinu i preciznost rangiranog pretraživanja informacija.

# Biografija

Jovan Vučetić rođen je u Somboru, 31. 05. 1995. godine. Osnovnu školu „Matko Vuković“ završio je 2010. godine, a potom upisao srednju tehničku školu „Ivan Sarić“ u Subotici. Tehničku školu završio je 2014. godine odličnim uspehom, kada upisuje Prirodno-matematički fakultet u Novom Sadu, na departmanu za matematiku i informatiku, smer diplomirani informatičar. Diplomirao u septembru 2018. godine. Nosilac je majstorskog zvanja u džudou, 1. DAN.

**UNIVERZITET U NOVOM SADU**

**PRIRODNO MATEMATIČKI FAKULTET**

**KLJUČNA DOKUMENTACIJSKA INFORMACIJA**

Redni broj:

**RBR**

Identifikacioni broj:

**IBR**

Tip dokumentacije: Monografska dokumentacija

**TD**

Tip zapisa: Tekstualni štampani materijal

**TZ**

Vrsta rada: Master rad

**VR**

Autor: Jovan Vučetić

**AU**

Mentor: dr Miloš Savić

**MN**

Naslov rada: Indeksiranje i pretraživanje naučno-istraživačkih časopisa otvorenog pristupa

**NR**

Jezik publikacije: srpski(latinica)

**JP**

Jezik izvoda: s/en

**JI**

Zemlja publikovanja: Republika Srbija

**ZP**

Uže geografsko područje: Vojvodina

**UGP**

Godina: 2020.

**GO**

Izdavač: Autorski reprint

**IZ**

Mesto i adresa: Novi Sad, Trg Dositeja Obradovića 4

**MA**

Fizički opis rada: (4/51/6/8/3/21/10)

(broj poglavlja/strana/lit.citata/tabela/slika/grafika/priloga)

**FO**

Naučna oblast: Računarske nauke

**NO**

Naučna disciplina: Pretraživanje informacija

**ND**

Predmetne obrednica,

Ključne reči: k-core dekompozicija, Batagelj-Zaveršnik algoritam, socijalne mreže

**PO**

**UKD**

Važna napomena:

**VN**

Čuva se:

**ČU**

Izvod: Rad se bavi implementacijom aplikacije za k-core dekompoziciju težinskih i netežinskih socijalnih mreža korišćenjem Batagelj-Zaveršnik algoritma za računanje shell index-a čvorova u grafu.

**IZ**

Datum prihvatanja teme

od NN veća: 15. 05. 2018.

**DP**

Članovi komisije:

(Naučni stepen/ime i prezime/zvanje/fakultet)

**KO**

Predsednik: dr Vladimir Kurbalija, vanredni profesor Prirodno-matematičkog fakulteta u Novom Sadu

Član: dr Miloš Savić, docent Prirodno-matematičkog fakulteta u Novom Sadu

Član: dr Aleksandra Klašnja Milićević, docent Prirodno-matematičkog fakulteta u Novom Sadu

**UNIVERSITY OF NOVI SAD**

**FACULTY OF NATURAL SCIENCES & MATHEMATICS**

**KEY WORDS DOCUMENTATION**

Accession number:

**ANO**

Identification number:

**INO**

Document type: Monograph documentation

**DT**

Type of record: Textual printed material

**TR**

Contents code: Graduation thesis

**CC**

Author: Jovan Vucetic

**AU**

Mentor: Dr Milos Savic

**MN**

Title: K-core decomposition of weighted and unweighted social networks

**TI**

Language of text: Serbian (Latin)

**LT**

Language of abstract: en/s

**LT**

Country of publication: Republic of Serbia

**CP**

Locality of publication: Vojvodina

**LP**

Publication year: 2018

**PY**

Publisher: Author’s reprint

**PU**

Publ. place: Novi Sad, Trd D. Obradovića 4

**PP**

Physical description: (4/51/6/8/3/21/10)

**PD**

Scientific field: Computer Science

**SF**

Scientific discipline: Social Networks

**SD**

Subject Key words: K-core decomposition, Batagelj-Zaversnik algorithm, social networks

**SKW**

**UC**

Holding data:

**HD**

Note:

**N**

Abstract: This thesis decribes implementation of application for k-core decomposition of weighted and unweighted social networks, by using Batagelj-Zaversnik algorithm for computing shell index of vertices in graph.

**AB**

Accepted on Scientific board on: 5/15/2018

**AS**

Defended:

**DE**

Thesis Defend board:

**DB**

President: Dr Vladimir Kurbalija, associate professor, Faculty of Sciences, Novi Sad

Member: Dr Milos Savic, assistant professor, Faculty of Sciences, Novi Sad

Member: Dr Aleksandra Klasnja Milicevic, assistant professor, Faculty of Sciences,Novi Sad