# Algoritamski pristup izradi bibliografija

**Vežić, Jakov Marin**

**Master's thesis / Diplomski rad**

**2016**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Zadar / Sveučilište u Zadru**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:162:161047

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-03-28**

Image not found or type unknown

*Repository / Repozitorij:*

University of Zadar Institutional Repository

zir.nsk.hr

Image not found or type unknown

Sveučilište u Zadru

Odjel za informacijske znanosti
Diplomski sveučilišni studij Informacijske znanosti - knjižničarstvo

**Jakov M. Vežić**

# Algoritamski pristup izradi bibliografija

**Diplomski rad**

Zadar, 2016.

Sveučilište u Zadru

Odjel za informacijske znanosti
Diplomski sveučilišni studij Informacijske znanosti - knjižničarstvo

# Algoritamski pristup izradi bibliografija

Diplomski rad

Student/ica:
Jakov Marin Vežić

Mentor/ica:
dr. sc. Krešimir Zauder

Zadar, 2016.




# Izjava o akademskoj čestitosti

Ja, **Jakov Marin Vežić,** ovime izjavljujem da je moj **diplomski** rad pod naslovom **Algoritamski pristup izradi bibliografija** rezultat mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na izvore i radove navedene u bilješkama i popisu literature. Ni jedan dio mojega rada nije napisan na nedopušten način, odnosno nije prepisan iz necitiranih radova i ne krši bilo čija autorska prava.

Izjavljujem da ni jedan dio ovoga rada nije iskorišten u kojem drugom radu pri bilo kojoj drugoj visokoškolskoj, znanstvenoj, obrazovnoj ili inoj ustanovi.

Sadržaj mojega rada u potpunosti odgovara sadržaju obranjenoga i nakon obrane uređenoga rada.

Zadar, 3. listopada 2016.

# Sadržaj

Sažetak ..... 4
1. Uvod ..... 5
2. Razvoj bibliografije kroz povijest ..... 7
2.1. Definicija bibliografije ..... 7
2.2. Bibliografija i knjižnične znanosti ..... 8
3. Bibliografska heuristika ..... 12
3.1. Podjela bibliografija ..... 12
3.2. Faze bibliografske heuristike ..... 14
4. Definicija sustava i prikaz srodnih alata ..... 16
4.1. Ciljevi sustava ..... 16
4.2. Ciljevi softverskog rješenja ..... 16
4.3. Opseg sustava ..... 17
4.4. Srodni sustavi i alati ..... 17
4.4.1. Zotero ..... 18
4.4.2. Mendeley ..... 18
4.4.3. EndNote ..... 18
4.4.4. Integrirani knjižnični sustavi ..... 19
5. Sustav za poluautomatsku izradu bibliografije ..... 20
5.1. Komponente sustava ..... 20
5.1.1. Formati ..... 21
5.1.2. Model ..... 22
5.1.3. Mapiranje bibliografskih formata ..... 24
5.1.4. Zbirka ..... 25
5.1.5. Citatni stilovi ..... 27
5.1.6. Bibliografija ..... 28
5.2. Primjena sustava ..... 30
5.2.1 Izrada modela ..... 30
5.2.2. Uvoz i mapiranje BibTeX zapisa ..... 31
5.2.3. Uvoz i mapiranje CSV zapisa ..... 33
5.2.4. Uvoz i mapiranje MARC zapisa ..... 34
5.2.5. Priprema zapisa u zbirci ..... 37
5.2.6. Grupiranje ..... 43
5.2.7. Filtriranje ..... 45
5.2.8. Koš za smeće ..... 45
5.2.9 Citatni stilovi i bibliografija ..... 45
6. Prikaz arhitekture softverskog rješenja ..... 53
6.1. Python kôd ..... 53

6.1.1. Model ..... 54
6.1.2. Mapiranje formata u model ..... 59
6.1.3. Zbirka ..... 59
6.1.4. Citatni stil ..... 61
6.1.5. Bibliografija ..... 62
6.2. Grafičko sučelje softvera ..... 64
6.2.1. Izrada modela ..... 65
6.2.2. Mapiranje formata u model ..... 67
6.2.3. Zbirka ..... 70
6.2.4. Priprema zapisa u zbirci ..... 74
6.2.5. Filtriranje ..... 76
6.2.6. Grupiranje ..... 76
6.2.7. Primjena postojećih citatnih stilova ..... 77
6.2.8. Izrada citatnih stilova ..... 78
6.2.9. Bibliografija ..... 79
7. Nedostatci i mogućnosti proširenja sustava ..... 81
8. Zaključak ..... 83
Abstract ..... 85
Popis literature ..... 86

**ZAHVALA**

Rad na 90 stranica u kojem se isprepliću knjižničarstvo, informacijske znanosti, dizajn sustava i programiranje nemoguće je napraviti bez pomoći osobe koja u malom prstu ima sve navedeno.

Stoga se neizmjerno zahvaljujem svom mentoru, **dr. sc. Krešimiru Zauderu**, za pomoć vezanu uz Python (i sve njegove klase, rječnike, liste, funkcije, itd.), za strpljenje u ispravljanju teksta, za koncentriranost u povezivanju svih detalja u radu te za konstantnu dostupnost i otklanjanje nedoumica, poteškoća i nejasnoća.

Process finished with exit code 0.

# Sažetak

Cilj diplomskog rada je algoritamskim pristupom ponuditi svestran i pristupačan sustav za poluautomatsku izradu bibliografija. Bibliografija je jedna od najstarijih djelatnosti knjižnične znanosti koja u današnjem digitalnom dobu nailazi na nove izazove radi mnoštva dostupnih bibliografskih informacija u raznim strukturiranim bibliografskim zapisima te kroz *web* sučelja baza podataka. Bibliografski podaci danas se javljaju u raznim formatima, čime se stvara problem pri izradi bibliografije, odnosno otežava pristupačna i brza izrada bibliografija. Uz navedeno, strukturirani digitalni podaci namijenjeni su prvenstveno kako bi bili strojno iskoristivi što pretpostavlja algoritamsko baratanje istima te dodatan razlog za temu diplomskog rada. Predloženi sustav za poluautomatsku izradu bibliografija uključuje mogućnosti: čitanja raznih formata za razmjenu bibliografskih metapodataka kao što su specijalizirani (npr. MARC, BibTeX, RIS) i općeniti (npr. XML, JSON) formati za razmjenu podataka, filtriranja učitanih zapisa prema nizu parametara (npr. godina izdavanja, izdavač, autor), grupiranja istih, definiranja stila navođenja te izvoza bibliografije u razne formate kao što su HTML, TXT, PDF i DOCX. Rad će također pružiti pregled relevantnih tema iz područja bibliografije kao i osvrt na važnost te prednosti i mane algoritamskog pristupa izradi bibliografija. Diplomski rad predstavlja napredak u suvremenoj bibliografskoj djelatnosti s naglaskom na korištenju i adaptaciji novih tehnologija kao i mogućnosti izrade svestranih računalnih alata koji su prilagođeni lokalnoj tradiciji i potrebama.

**Ključne riječi**: bibliografija, algoritamski pristup, bibliografska heuristika, softver, bibliografski metapodaci

# 1. Uvod

Bibliografija, jedna od najstarijih disciplina knjižnične i informacijske znanosti, nailazi na nove izazove u današnjem umreženom digitalnom svijetu. Glavni razlozi tome su povećana dostupnost bibliografskih informacija i nove paradigme rada sa strukturiranim podacima koji doprinose kompleksnosti već složenih bibliografskih procesa. Izrada bibliografije i procesi bibliografske heuristike ovise o nizu čimbenika: kvaliteti i strukturi bibliografskih podataka, vrstama bibliografije, načinom izrade te lokalnom kontekstu i namjeni. Bibliografski podaci javljaju se u raznim strukturiranim i nestrukturiranim oblicima i formatima, dok njihova kvaliteta može značajno varirati ovisno o izvoru. Mnoštvo bibliografskih formata stvara problem pri izradi bibliografije iz razloga što se podaci iz svih formata moraju strukturalno i semantički ujednačiti te pripremiti za prikaz u bibliografiji, a ručna priprema takvih podataka dugotrajan je i iscrpan posao. Sam proces izrade bibliografije može se razlikovati ovisno o njenoj vrsti, a u slučaju serijskih bibliografija nikada nije gotov jer se bibliografija stalno nadopunjuje. Potencijalno rješenje navedenih problema može se pronaći u automatskom generiranju bibliografije. Međutim, takav pristup sklon je pogreškama, posebno ako je riječ o podacima loše kvalitete, a često je i ograničen na stvaranje enumerativnih bibliografija te se ne može uklopiti u lokalni kontekst bibliografije, kao ni potrebama potencijalnih korisnika. Iz navedenoga je vidljivo kako je najbolje rješenje problema olakšavanja i ubrzavanja izrade bibliografije **poluautomatski pristup**, gdje izrada bibliografija sadrži automatske elemente rukovanja podacima, ali sa značajnim intervencijama bibliografa.

Praktičan ishod ovog rada je ponuditi svestrano i pristupačno softversko rješenje za poluautomatsku izradu bibliografija, što znači prikazati proces poluautomatske izrade bilo koje vrste bibliografije za bilo koji lokalni kontekst. Navedeni sustav mora biti dovoljno apstraktan da je iz njega moguće razviti alate raznih oblika (softverski program, mrežnu aplikaciju, itd.), a dovoljno detaljan da odgovori na već iskazane probleme izrade bibliografije u današnjem digitalnom dobu.

Sustav stoga mora omogućiti:

- primjerenu slobodu izrade bibliografije
- čitanje raznih formata za razmjenu bibliografskih metapodataka
- definiranje vlastitog modela podataka
- ručno dodavanje vlastitih zapisa
- filtriranje i grupiranje zapisa
- definiranje stila navođenja za bibliografiju

- izvoz bibliografija u razne formate.

Time bi navedeni sustav predstavljao napredak u polju bibliografije i bibliografske heuristike. Kao dokaz valjanosti sustava, u radu će se prikazati i softversko rješenje temeljeno na predloženom sustavu, kao i primjer izrade bibliografije slijedeći logiku sustava te prikaz izrade bibliografije u softverskom rješenju.

Rad je podijeljen u sedam poglavlja. U poglavlju „2. Razvoj bibliografije kroz povijest“ daje se pregled razvoja bibliografije kroz povijest, što uključuje razvoj značenja definicije bibliografije, kao i uloge bibliografije u kontekstu knjižničnih i informacijskih znanosti. U poglavlju „3. Bibliografska heuristika“ definiraju se i objašnjavaju bibliografska heuristika i njene faze te prikazuju razne podjele bibliografije. U poglavlju „4. Definicija sustava i prikaz srodnih alata“ definiraju se ciljevi, svrha i opseg istraživanja te se prikazuju prethodna istraživanja i alati najbliži temi ovog rada. Peto poglavlje sadrži prikaz predloženog sustava, kao i primjer primjene sustava na izradu stvarne bibliografije, dok šesto poglavlje opisuje softversko rješenje kao dokaz valjanosti sustava, također kroz stvarni primjer izrade bibliografije. U sedmom i posljednjem poglavlju raspravlja se o nedostatcima algoritamskog pristupa, ali i o mogućnostima napretka i proširenja predloženog sustava te daljnjih istraživanja.

# 2. Razvoj bibliografije kroz povijest

Iako se bibliografija, kao umijeće popisivanja i opisivanja dokumenta, javlja još u starom vijeku u mezopotamskim kulturama, Egiptu te antičkoj Grčkoj i Rimu, značajniji razvoj bibliografije bio je moguć tek od pojave tiskane knjige 1450. godine čime je omogućena lakša i brža distribucija knjiga. Suvremena bibliografija tako ima korijene u praktičarima i teoretičarima tog vremena kao što su Johann Trithemius i Konrad Gesner. Gesner svojim djelom *Bibliotheca Universalis* (1545.g.) pokušava popisati sve znanstvene publikacije u svijetu u obliku bibliografije. U tom razdoblju bibliografija se označavala terminima poput *catalogus, elenchus, index, nomenclator* i *bibliotheca*. Čest termin bio je i *historia literaria* kojeg je prvi upotrijebio Francis Bacon 1605. godine, koji je smatrao da je vrijednost bibliografije u usmjeravanju čitatelja prema korištenju literature.[1]

Sam termin „bibliografija" pojavio se tek 1663. godine, iako tada nije označavao popis knjiga, već logično vođenje prema njihovu pronalaženju. U 18. stoljeću, bibliografija označava „umijeće davanja informacija o knjigama"[2], dok se u 19. stoljeću pojavljuju tri inovacije značajne za razvoj bibliografske teorije i prakse, a to su: knjižnična kataložna pravila, serijski popisi tekućih sadržaja te formalna istraživanja knjiga kao fizičkih objekata.[3]

Od 19. stoljeća, bibliografija se uglavnom definira slično kao danas. Međutim, danas postoje dvije različite discipline koje se označavaju kao „bibliografija". Bibliografija tako može biti „popisivanje knjiga, poredanih u skladu s određenim sustavom", ali i „učenje o knjigama kao materijalnim objektima, tj. proučavanje materijala od kojega su knjige napravljene i načina na koji se stvaraju".[4] Ovaj rad usmjerit će se na prvu disciplinu, odnosno na razvijanje sustava za izradu popisa poredanog u skladu s određenim sustavom. Međutim, navedena definicija proizlazi još iz 19. stoljeća stoga je potrebno, u kontekstu digitalnog doba, ispitati koje adaptacije koncepta su potrebne kako bi bibliografija zadržala svoj značaj i bila iskoristiva u današnjem svijetu.

## 2.1. Definicija bibliografije

Pojavom časopisa i drugih tiskanih publikacija u definicijama bibliografije naziv „knjiga" zamijenjen je nazivom „tiskani rad". Ipak, danas je vjerojatno najprikladnija i

[1] Usp. Pehar, Franjo. Od statističke bibliografije do bibliometrije. Povijest razvoja kvantitativnog pristupa istraživanju pisane riječi. // Libellarium 3, 1(2010), str. 1-28.
[2] Aparac-Gazivoda, Tatjana. Teorijske osnove knjižnične znanosti. Zagreb : Filozofski fakultet Sveučilišta u Zagrebu; Zavod za informacijske studije Odsjeka za informacijske znanosti, 1993. str. 54.
[3] Usp. Pehar, Franjo. Nav. dj.
[4] Aparac-Gazivoda, Tatjana. Nav. dj. str. 54.

najtočnija definicija prema kojoj je bibliografija „stručno-znanstvena djelatnost, koja sabire, vrednuje, odabire, sadržajno analizira i opisuje tiskane ili na drugi način umnožene, javnosti namijenjene tekstove – bibliografske jedinice – pa te opisane klasificira, uređuje i obično u obliku **uređenih popisa** i publicira s namjerom da pruži informacije o literaturi, a time i pomagalo za stručni rad“.[5] Iz ove definicije jasno je što je bibliografija te koji je joj cilj, svrha, pa čak i postupci pri izradi. Također se ističe kako je svaki izvor opisan u bibliografiji **bibliografska jedinica** te kako bibliografija uključuje i druge vrste građe („na drugi način umnožene“) osim knjiga i tiskanih publikacija. Bibliografija se stoga smatra i alatom za organizaciju informacija.[6]

**Cilj je bibliografije** ponuditi kratke prikaze svakog dokumenta, odnosno bibliografske jedinice koje identificiraju dokument (u slučaju određenih vrsta bibliografija i opisuju dokument), a koje se okupljaju prema određenim sličnostima.[7] Bibliografske jedinice izričito identificiraju izdanja djela („pojavne oblike“ prema Uvjetima za funkcionalnost bibliografskih zapisa[8]), a ne sama djela ili jedinice građe.[9]

## 2.2. Bibliografija i knjižnične znanosti

Bibliografiju su autori poput G. Leyh, J. A. F. Schmidt, J. Kirchner i G. Leidinger smatrali sastavnim dijelom knjižnične znanosti, dok autori V. Kubatov i J. Janos bibliografiju, uz knjižničarstvo, smatraju začetkom informacijskih znanosti, ističući kako je teorija bibliografije bila sustav propisa i pretpostavki proizašlih iz praktičnih potreba te kako se bibliografija usmjerila na „popis, opis, sistematizaciju i analizu tiskane građe, na pripremanje popisa te propagandu“[10] što naposljetku olakšava korisnicima snalaženje u mnoštvu literature. Shera i Egan govore o bibliografiji kao najvećoj mogućoj društvenoj koristi od zabilježenog ljudskog iskustva jer usmjerava bibliografske jedinice svim korisnicima, ali i napominju kako postoje različita gledišta o mjestu bibliografije u procesu razmjene znanja: makrokozmičko gledište govori kako je bibliografija jedan od instrumenata za komunikaciju među ljudima, dok

[5] Tadić, Katica. O bibliografiji s osvrtom na bibliografsku heuristiku. // Izazovi pisane baštine: zbornik radova u povodu 75. obljetnice života Aleksandra Stipčevića. Osijek: Filozofski fakultet, 2005. str. 164.
[6] Usp. Rubin, Richard E. Foundations of Library and Information Science. [s.l.] : Facet publishing, 2016. str. 326.
[7] Usp. White, Howard D. Publication and Bibliographic Statements. // Interpretations of Reference and Bibliographic Work / White, Howard D; Bates, Marcia J; Wilson, Patrick. Norwood, New Jersey: Ablex Publishing, [s.a]. str. 85-86.
[8] Uvjeti za funkcionalnost bibliografskih zapisa : završni izvještaj / IFLA-ina Studijska skupina za uvjete za funkcionalnost bibliografskih zapisa. Zagreb : Hrvatsko knjižničarsko društvo, 2004. str. 21-24.
[9] Isto. str. 107.
[10] Aparac-Gazivoda, Tatjana. Nav. dj. str. 55.

mikrokozmičko gledište opisuje bibliografiju kao alat kojim zadovoljavamo specifične potrebe ograničenog broja osoba s istim ili sličnim interesima.[11]

Krajem 19. stoljeća, teorija bibliografije počela se baviti i bibliografskom kontrolom, iako su neki autori smatrali da je svrha bibliografije ustanoviti razlike između primjeraka objavljenih djela, a ne obavljati bibliografski nadzor. P. Otlet i H. LaFontaine također su se bavili bibliografijom, točnije idejom „svjetske bibliografije“, iz čega su razvili pojam „dokumentacija“ kojem su pridavali više značenja, od kojih se jedan odnosio na postojeći korpus literature koji se odnosi na određeni predmet. Kasnije su, za potrebe uređivanja kataloga dokumenata razvili i Univerzalnu decimalnu klasifikaciju. Bitno je također napomenuti kako su Otlet i LaFontaine skup navedenih disciplina opisivali kao bibliografske znanosti.[12] Mihajlov, Giljarevski, Wellisch i Foskett tumače informatiku kao nastavak bibliografije i knjižnične znanosti, ali smatraju kako ih je potrebno preispitati radi boljeg razumijevanja strukture, svojstava i zakonitosti znanstvenih informacija[13] Danas se informacijske znanosti, s knjižničarskog gledišta, na izvedbenoj razini posebno zanimaju za razvoj odnosa između čovjeka i stroja.[14]

Bibliografija se u kontekstu knjižnica često uspoređuje i povezuje s katalogom. Razlika između bibliografije i kataloga je ta što je bibliografija u odnosu na katalog selektivnija, sveobuhvatnija i sustavnija. Razlika je također u tome što knjižnični katalog popisuje publikacije koje se nalaze u fondu knjižnice, odnosno koje knjižnica posjeduje, dok bibliografija opisuje objavljene publikacije neovisno o njihovoj trenutnoj lokaciji.[15] [16] Ipak, neki autori knjižnični katalog smatraju vrstom bibliografije,[17] odnosno jednom od bibliografija „na veliko“ (*wholesale*), u što uključuju i nacionalne bibliografije.[18]

Povezanost bibliografije i knjižnica također se pronalazi u činjenici što knjižnice moraju svojim korisnicima osiguravati bibliografske izvore koji se odabiru i uključuju u referentne zbirke, čime se stvara veza između knjižnice i njenih korisnika. Knjižničari također često sudjeluju u samoj izradi bibliografija kvalitetnim odabirom građe iz određenog područja te

[11] Usp. Isto.
[12] Usp. Aparac-Gazivoda, Tatjana. Nav. dj. str. 54-56.
[13] Usp. Isto. str. 59.
[14] Usp. Isto. str. 61.
[15] Usp. Tadić, Katica. Nav. dj. str. 166.
[16] Usp. Soergel, Dagobert. Organizing information: Principles of Data Base and Retreival Systems. San Diego [etc.] : Academic Press, 1985. str. 207-208.
[17] Usp. Bates, Marcia J. Rigorous Systematic Bibliography. // Interpretations of Reference and Bibliographic Work / White, Howard D; Bates, Marcia J; Wilson, Patrick. Norwood, New Jersey: Ablex Publishing, [s.a]. str. 119.
[18] Usp. Wilson, Patrick. Pragmatic Bibliography. // Interpretations of Reference and Bibliographic Work / White, Howard D; Bates, Marcia J; Wilson, Patrick. Norwood, New Jersey: Ablex Publishing, [s.a]. str. 239-240.

pouzdanim i dosljednim opisom bibliografskih jedinica. Knjižnice i bibliografije maksimalno se približavaju upravo u opisu bibliografske jedinice, a posebno u odnosu na nove načine predstavljanja bibliografija u mrežnom okruženju.[19] Bibliografije se u velikoj količini upotrebljavaju upravo u knjižnicama, kao sekundarne publikacije. Knjižnice su također do nedavno bile jedino mjesto gdje se korisniku mogla ponuditi i sekundarna informacija (podatak o dokumentu) i pristup samom dokumentu, što se promijenilo pojavom mrežnih publikacija i izvora. Ipak, u području humanističkih i društvenih znanosti još uvijek prevladavaju tiskani izvori, uglavnom dostupni u knjižnicama.[20]

Dora Sečić bibliografiju u kontekstu knjižnica klasificira kao bibliografski izvor ili izvor za pronalaženje dokumenata, kao dio konvencionalnih izvora informacija. U smislu razine izvora informacija, bibliografije svrstava u sekundarne izvore (u što spadaju kazala, katalozi i sl.), a bibliografije bibliografija u tercijarne izvore (enciklopedije, priručnici i sl.).[21] Bibliografije mogu često biti i odgovor na same informacijske upite korisnika, pogotovo one vezane uz specifičnu temu. Iako korisnici danas sami mogu pretraživati izvore informacija na mreži, bibliografije nude gotov, pomno odabran i uređen popis bibliografskih jedinica relevantnih korisniku, što u suvremenom svijetu postaje sve važnije, ali stvara i veći izazov pri izradi bibliografije. Iz tog razloga, knjižničari moraju biti upoznati sa što više bibliografija kako bi korisnicima mogli ponuditi kvalitetan izvor informacija.[22]

William Katz navodi osam kriterija za vrednovanje korisnosti i kvalitete bibliografije, a to su[23]:

1. **Svrha** – bibliografija mora odgovarati stvarnoj potrebi a ne ponavljati već dostupne informacije, tema bibliografije mora jasno biti navedena u njenom naslovu ili predgovoru
2. **Obujam** – bibliografija mora biti što potpunija u odnosu na njenu svrhu
3. **Metodologija** – metoda sastavljanja bibliografije mora biti jasno navedena, a izvori moraju biti napisani u standardnom bibliografskom stilu koristeći osnovne elemente bibliografskih jedinica
4. **Organizacija** – bibliografija mora biti jasno organizirana, a po mogućnosti i sadržavati kazala i popise skraćenica
5. **Anotacije i sažetci** – ukoliko sadrži deskriptivne podatke, oni moraju biti jasni i informativni

[19] Usp. Tadić, Katica. Nav. dj. str. 166.
[20] Usp. Sečić, Dora. Informacijska služba u knjižnici. Lokve: Benja, 2006. str. 17.
[21] Usp. Isto. str. 63-64.
[22] Usp. Katz, William A. Introduction to reference work. Boston [etc.] : McGraw Hill, 2002. str. 75
[23] Isto. str. 80-81.

6. **Bibliografske jedinice** – moraju sadržavati sve informacije potrebne za identifikaciju dokumenta

7. **Važeći dokumenti** – dokumenti navedeni u bibliografiji moraju biti recentni, ukoliko je to svrha bibliografije

8. **Preciznost** – informacije u svim bibliografskim jedinicama moraju biti točni, a ukoliko je to moguće, treba dopustiti i način ispravljanja pogreški nakon izdavanja bibliografije

# 3. Bibliografska heuristika

Bibliografska heuristika skupni je naziv za faze kroz koje bibliograf prolazi pri izradi bibliografije. One uglavnom uključuju selekciju gradiva, način formiranja i razvrstavanja bibliografskih jedinica te stvaranje bibliografskog niza.[24] Bibliografska jedinica može se razlikovati od područja do područja. Tako se u nekim područjima bibliografskom jedinicom može smatrati samo knjiga (odnosno njezin bibliografski opis), dok su u drugim područjima to članci, rezultati eksperimenta (npr. fizika) i sl. Stoga se može zaključiti da se bibliografske jedinice razlikuju po načinu pisanja u bibliografijama ovisno o području koje bibliografija pokriva. Također, određeni radovi iz područja poput fizike nerijetko imaju velik broj autora, čime se potencijalno povećava i prikaz bibliografske jedinice u bibliografiji.[25] Pri izradi bibliografije često se postavljaju pravila koja omogućuju dosljednu primjenu i ujednačenost bibliografskih jedinica pri popisivanju i razvrstavanju gradiva. Ta pravila mogu proizlaziti iz citatnih stilova ili mogu biti napravljena u potpunosti za potrebe jedne bibliografije.[26]

## 3.1. Podjela bibliografija

Postoji niz načina podjela bibliografija. Opće ili međunarodne bibliografije opisuju publikaciju za određeno vremensko razdoblje, bez obzira na mjesto, jezik i struku, dok specijalne bibliografije opisuju publikacije određene struke ili određene teme.

Prema mjestu nastajanja publikacija, bibliografije se dijele na nacionalne, regionalne i mjesne. Nacionalne bibliografije popisuju sve publikacije nastale u državi, ali i sva djela pisaca toga naroda kao i publikacije koje govore o toj državi. Regionalne i mjesne poštuju ista pravila kao nacionalne, ali su manjeg zemljopisnog opsega.

Prema vremenu nastajanja, bibliografije mogu biti retrospektivne ili tekuće. Tekuće bibliografije serijske su publikacije jer se tijekom vremena nadopunjuju, dok su retrospektivne bibliografije omeđene publikacije.

Prema načinu opisa, bibliografije su popisne, opisne i analitičke (anotirane). Popisne (enumerativne) bibliografije navode samo najnužnije elemente bibliografske jedinice (primjerice nacionalne bibliografije[27]), opisne (deskriptivne) popisuju i određene nestandardne

[24] Usp. Tadić, Katica. Nav. dj. str. 167.
[25] Usp. Borgman, Christine L. Od Gutenbergova izuma do globalnoga informacijskog povezivanja: pristup informaciji u umreženom svijetu. Lokve; Zadar : Naklada Benja; Gradska knjižnica Zadar, 2002. str. 86
[26] Usp. Tadić, Katica. Nav. dj. str. 168.
[27] Rubin, Richard E. Nav. dj. str. 326.

elemente poput mjesta tiska ili vrste papira, dok analitičke objašnjavaju i analiziraju sadržaj bibliografske jedinice.[28]

Patrick Wilson bibliografije dijeli na već spomenute bibliografije „na veliko“ i pragmatične bibliografije. Bibliografije „na veliko“ opisuje kao velike, iznimno organizirane i kontinuirane publikacije i baze podataka, dok pragmatične bibliografije opisuje kao rezultat rada jedne osobe na vrlo specifičnoj i ograničenoj temi, bilo znanstvenoj ili praktičnoj.[29]

Bibliografije je moguće podijeliti i prema disciplinarnim pristupima: njemački pristup polazi od bibliografskih popisa te predstavlja knjigu i dokumentaciju kao dijelove informacijske znanosti, britanski pristup smatra da je bibliografija područje usmjereno istraživanju knjiga (bibliologija), dok francuski pristup bibliografiju percipira kao kulturalnu i kognitivnu snagu.[30] Danas se bibliografija približava i nekim metodološkim znanostima kao što su matematika, statistika, logika itd. Najčešći primjer je bibliometrija, primjena bibliografskih i matematičkih metoda kako bi se došlo do odgovora na pitanja kojima se tradicionalna bibliografija ne bavi.[31]

Bibliografije se također mogu podijeliti na analogne (tiskane) i digitalne. Za razliku od tiskanih publikacija, digitalne bibliografije često se stvaraju automatski za određenu potrebu, npr. prikazivanje rezultata pretraživanja. One mogu biti statični, ali i dinamični prikazi koji dopuštaju *ad hoc* promjene u sortiranju, grupiranju i filtriranju bibliografskih jedinica, čime je dinamična digitalna bibliografija sličnija aplikaciji nego statičnom dokumentu.

Pojavom računala i računalnih sustava nastaje još jedna podjela bibliografija, a to je podjela prema načinu izrade. Stoga danas možemo reći da, prema načinu izrade, bibliografija može biti ručno, poluautomatski i automatski izrađena.

Ručno izrađena bibliografija je bibliografija u kojoj je svaka bibliografska jedinica ručno napisana, čime se najviše izbjegavaju pogreške kao što su krivi podaci, duplikacija podataka i ponavljanje bibliografskih jedinica. Ipak, ovaj pristup nije primjeren za *ad hoc* izradu bibliografija ili za bibliografije s velikim brojem bibliografskih jedinica. Potrebno je također napomenuti kako „ručna“ izrada ne isključuje upotrebu računala tijekom procesa izrade bibliografije, već jednostavno znači da se ne koriste algoritmi koji bi automatski selektirali, pripremali i stvarali bibliografske jedinice.

Automatski izrađena bibliografija izrađena je u potpunosti sa softverskim algoritmima, najčešće za jasno definiran skup publikacija za koje postoje gotovi strukturirani digitalni

[28] Usp. Tadić, Katica. Nav. dj. str. 169.
[29] Usp. Wilson, Patrick. Nav. dj. str. 239-240.
[30] Usp. Pehar, Franjo. Nav. dj. str. 5
[31] Usp. Isto.

podaci. Ovakva bibliografija se stvara uglavnom bez ikakvog nadzora i u sklopu drugog, većeg sustava (npr. *online* sučelja za bibliografske baze podataka). Također, automatski izrađena bibliografija uglavnom stvara enumerativne bibliografije te je najsklonija pogreškama zbog svoje potpuno automatske prirode.

Poluautomatski izrađena bibliografija, na koju će se ovaj rad fokusirati, odnosi se na bibliografiju koja je izrađena automatski sa softverskim algoritmima, ali koja ostavlja mogućnost ručnih intervencija tijekom određenih faza bibliografske heuristike. Automatsko generiranje bibliografskih jedinica tako se može nadopuniti s ručnim unosom koji može omogućiti, između ostalog, dodatnu provjeru vrijednosti polja ili izradu vlastitih polja kako bi se stvorila deskriptivna bibliografija. Iako se navedeni koncept može činiti sličan softverima za upravljanje referenci (*reference management software*) kao što su Zotero, Mendeley, JabRef, EndNote i sl., oni su uglavnom usmjereni na stvaranje kratkih, isključivo enumerativnih, bibliografija za potrebe osobnog znanstvenog izdavaštva, za razliku od rješenja predloženog u ovom radu, koje se usmjerava na profesionalnu izradu bibliografija svih vrsta i rješavanje drugih problema kao što su provjera valjanosti polja i vrijednosti.

Navedene situacije i podjele stvaraju kompleksnu situaciju za bibliografije, s nizom načina izrade, medija za objavljivanje i namjena korištenja.

## 3.2. Faze bibliografske heuristike

Izrada bibliografije može se prikazati slijedom sedam faza, koje zajedno čine bibliografsku heuristiku. Te su faze[32]:

1. Sastavljanje temeljnog programa rada
2. Proučavanje prethodnog znanja o predmetu kojim će se bibliograf baviti
3. Pronalaženje gradiva (bibliografskih jedinica) u svim raspoloživim izvorima
4. **Popisivanje bibliografskih jedinica i njihovo razvrstavanje**
5. **Klasificiranje gradiva**
6. **Stvaranje bibliografskog niza, konačne cjeline, bez obzira na njihovo oblikovanje u abecednom, kronološkom ili stručnom poretku**
7. Oblikovanje kazala, kao važnog instrumenta bibliografije te drugih pratećih tekstova važnih za bibliografiju

Neke od navedenih faza manje su ili više važne ovisno o vrsti bibliografije koja se izrađuje, ali u teorijskom smislu faze se odnose na svaku bibliografiju.[33] Ovaj rad usmjerit će se na

[32] Tadić, Katica. Nav. dj. str. 167.
[33] Usp. Isto.

četvrtu, petu i šestu fazu te za navedene faze predložiti sustav za poluautomatsku izradu bibliografija.

# 4. Definicija sustava i prikaz srodnih alata

Cilj je ovog rada predložiti sustav koji omogućava svestranu, prilagodljivu i proširivu platformu za izradu svih vrsta bibliografija te odgovara potrebama stručne, poluautomatske, izrade bibliografija.

## 4.1. Ciljevi sustava

Ciljevi koje sustav mora ispuniti su omogućavanje:

1. Primjerene slobode nad izradom bibliografije
2. Čitanja raznih formata za razmjenu bibliografskih metapodataka
3. Definiranja vlastitog modela podataka
4. Dodavanja vlastitih zapisa
5. Filtriranja učitanih zapisa
6. Grupiranja zapisa
7. Definiranja stila navođenja za bibliografiju
8. Izvoza bibliografije u razne formate

Bibliografski podaci mogu biti sadržani u nizu različitih formata, dok im kvaliteta također može varirati. Problemi koji se stoga mogu javiti pri izradi sustava nisu vezani samo uz rukovanje strukturiranim podacima, već i uz standardizaciju i provjeru valjanosti polja i vrijednosti. Također, izrada bibliografije ovisi o kontekstu (vrsta bibliografije i namjena) što znači da će se bibliografska heuristika i korištenje alata temeljenih na sustavu mijenjati ovisno o kontekstu izrade.

## 4.2. Ciljevi softverskog rješenja

Kako bi se dokazala valjanost opisanog algoritamskog pristupa, izradit će se softversko rješenje izvedeno iz predloženog sustava, koje će služiti kao dokaz koncepta (*proof of concept*). Softver bi stoga trebao omogućiti:

1. Učitavanje strukturiranih bibliografskih podataka (MARC, BibTex, CSV)
2. Zapisivanje konačne bibliografije u zadani format (HTML, TXT, PDF)
3. Mapiranje metapodataka definirano od strane bibliografa
4. Standardizaciju i validaciju podataka
5. Grafičko upravljanje
6. Tijek rada u grafičkom sučelju za poluautomatsku izradu bibliografija, što uključuje izvještaje i mogućnost odgovora od strane bibliografa

7. Mogućnost daljnjeg širenja i poboljšavanja softvera

Kako bi ispunio navedene ciljeve, softver će sadržavati nekoliko komponenti te će biti izrađen na način da se buduće nadogradnje, poboljšanja i promjene mogu lako i jednostavno integrirati.

Treba napomenuti kako softver osim izvoza bibliografije dozvoljava i ugrađivanje mogućnosti izvoza u strukturiranim formatima, ali kako to nije cilj ovog rada, navedena mogućnost nije integrirana.

## 4.3. Opseg sustava

U ovom radu istražit će se mogućnost algoritamskog pristupa za popisivanje bibliografskih jedinica i njihovo razvrstavanje, klasificiranje gradiva i stvaranje konačnog bibliografskog niza (koraci 4 do 6 u ranije navedenoj bibliografskoj heuristici). Sustav neće pokrivati korake bibliografske heuristike koji uključuju pronalaženje gradiva, izradu kazala, izradu popisa skraćenica i sl.

Očekuje se da predloženi sustav i iz njega izveden softver ponudi svestranu poluautomatsku izradu bibliografija koju je moguće primijeniti u raznim kontekstima, odnosno koja je prilagodljiva lokalnim tradicijama i potrebama. Sustav stoga sadrži algoritme (općenite postupke za sustavno rješavanje pojedinačnih zadataka[34]) poput algoritma za grupiranje, filtriranje i sl. Time navedeni sustav predstavlja napredak u polju bibliografije i bibliografske heuristike.

## 4.4. Srodni sustavi i alati

Iako nisu izričito vezani, sustavi za pohranu i dohvat podataka (*Information Storage and Retrieval Systems*) najbliži su u svom konceptu ranije navedenim ciljevima sustava. Oni uključuju faze poput skupljanja podataka, formatiranja podataka, uvoza podataka, obrade podataka, usporedbe podataka te prikaza podataka. Autori poput Dagoberta Soergela u detalje govore o kvalitetnom dizajnu sustava i shema za rukovanje raznim vrstama podataka, posebno o procesu koji opisuje korake od uvoza jedne vrsta formata podataka do izvoza u drugoj vrsti, s obradom podataka integriranom unutar procesa.[35]

Osim općenitih istraživanja obrade bibliografskih podataka i njihove prezentacije, vjerojatno najbliže ciljevima softverskog rješenja definiranim za ovaj rad dolaze softveri za

[34] Algoritam. Hrvatska enciklopedija. URL: http://www.enciklopedija.hr/Natuknica.aspx?ID=1718 (29.8.2016.)
[35] Usp. Soergel, Dagobert. Nav. dj. str. 137-159.

upravljanje bibliografskim bilješkama (*Reference Management Software*) poput softvera Zotero, Mendeley i EndNote. Ipak, navedeni softveri ne ispunjavaju sve ili čak većinu navedenih ciljeva, ali će biti opisani radi usporedbe sa sustavom i softverskim rješenjem u ovom radu.

### 4.4.1. Zotero

Zotero je dostupan kao proširenje za mrežni preglednik Firefox te programe za obradu teksta Word i LibreOffice, ali i kao zasebni program za operacijski sustav Windows. U kontekstu bibliografije, Zotero nudi mogućnost izvoza zbirke kao „bibliografije" u nizu citatnih stilova. Međutim, rezultat izvoza ne zadovoljava definiciju bibliografije iz razloga što se bibliografske jedinice ne mogu okupiti prema nekom parametru, drugim riječima, bibliografske jedinice nemoguće je grupirati. Zotero stoga zapravo stvara popis bibliografskih jedinica s glavnim ciljem izrade popisa literature za znanstvene radove. Takav popis moguće je izvesti u HTML i RDF formatu.[36] Također, za razliku od sustava i softverskog rješenja koji će biti prikazani u ovom radu, Zotero ne dopušta definiranje vlastitih polja, odnosno stvaranje vlastitog modela podataka, već je bibliograf prisiljen koristiti polja zadana programom.[37]

### 4.4.2. Mendeley

Mendeley je program usmjeren na upravljanje PDF datotekama, a dostupan je kao program za operacijske sustave Windows, Linux i Android, ali i kao mrežna aplikacija. U kontekstu bibliografije i bibliografskih jedinica, Mendeley nudi prikaz zbirke bibliografskih jedinica prema nizu citatnih stilova, ali ne i izvoz istih. Od formata, Mendeley podržava BibTeX, RIS i EndNote XML. Izvoz bibliografije nije moguć, kao ni izrada vlastitih polja i modela ili provedba dodatnih funkcija za analizu i upravljanje zapisima. Instalacijom dodatka za MS Word, moguće je kopirati bibliografske jedinice prema određenom citatnom stilu u MS Word, ali opet bez okupljanja bibliografskih jedinica u grupe, što čini stvarnu bibliografiju.

### 4.4.3. EndNote

EndNote je softver dostupan kao program za operacijske sustave MacOS i Windows, kao aplikacija za iPad te kao mrežna aplikacija. EndNote pruža mogućnosti pretraživanja baza

[36] Zotero. URL: https://www.zotero.org/download/ (29.8.2016.)
[37] Sarić, Ivana; Magdić, Antonio; Essert, Mario. ZOTERO – program otvorenog kôda za upravljanje bibliografskim bilješkama. // Vjesnik bibliotekara Hrvatske 54, 1/2(2011). str. 158-171. URL: http://hrcak.srce.hr/80472 (29.8.2016.)

podataka, uvoza i organizacije bibliografskih jedinica te podršku za najčešće korištene citatne stilove. Kao u Mendeley-u, iz MS Word-a je moguće stvarati citate i popis literature. EndNote također omogućava organizaciju PDF datoteka.[38] Međutim, kao što je slučaj u prethodna dva programa, ne sadrži mogućnosti grupiranja prema zadanim poljima, skupno manipuliranje zapisima u zbirci ili izradu vlastitih modela podataka.

### 4.4.4. Integrirani knjižnični sustavi

Može se pretpostaviti i da neki integrirani knjižnični sustavi posjeduju mogućnost izvoza zapisa u obliku bibliografije. Međutim, kako je velik broj integriranih knjižničnih sustava komercijalan i zatvorenog kôda, nemoguće je sa sigurnošću reći da je takva mogućnost prisutna. Ipak, kako primarna namjena takvih sustava nije izrada bibliografija, pitanje je posjeduju li naprednije mogućnosti poluautomatske izrade bibliografija. Koha, integrirani knjižnični sustav otvorenog kôda, posjeduje mogućnost izvoza zapisa u bibliografskim formatima kao što su MARC i XML[39], ali ne i mogućnost automatske ili poluautomatske izrade bibliografije.

Integrirani knjižnični sustavi su također, poput bibliografskih baza podataka, orijentirani na kontroliran unos, a zatim potencijalno i izvoz podataka iz njihovih sustava, što znači da u bibliografiju nije moguće dodati zapise iz drugih izvora, što je jedna od glavnih značajki sustava (samim time i softverskog rješenja) koji će biti predstavljeni ovom radu. Mnoge bibliografske baze podataka sadrže mogućnost izvoza odabranih zapisa u nizu formata i prema najčešće korištenim citatnim stilovima. Ipak, takve mogućnosti su često vrlo osnovne i ne odgovaraju na ciljeve sustava i softverskog rješenja koji su postavljeni u ovom radu.

[38] EndNote product details. URL: http://endnote.com/product-details (12.9.2016.)
[39] Export Bibliographic and Holdings Data (MARC Export). URL: http://manual.koha-community.org/3.2/en/exportbibs.html (29.8.2016.)

# 5. Sustav za poluautomatsku izradu bibliografije

Kako bi udovoljio jednom od navedenih ciljeva (omogućiti svestranu, prilagodljivu i proširivu platformu), sustav ne smije ovisiti o formatu iz kojeg će se unositi podaci, što znači da mora biti dovoljno apstraktan da prihvaća što veći broj strukturiranih podataka. Također, s obzirom na različite lokalne potrebe i tradicije, nemoguće je predvidjeti koji podaci će biti potrebni za svaku specifičnu bibliografiju. Iz tog razloga, sustav mora **omogućiti bibliografu definiranje vlastitog modela podataka** u kojeg će se strukturirani podaci uvoziti mapiranjem. Mapiranje se definira kao „operacija koja povezuje svaki element određenog skupa s jednim ili više elemenata drugog skupa“.[40] **Podaci** ujednačeni kroz **model** (mapirani iz više strukturiranih podataka u jedinstveni model) stvaraju **zbirku** na kojoj bibliograf može dodatno raditi (brisati i dodavati zapise, mijenjati ih, grupirati, filtrirati, itd.). Kada je zadovoljan stanjem zbirke, bibliograf će iz njenih zapisa generirati **bibliografiju** prema određenom **citatnom stilu** te po potrebi dodavati, mijenjati i brisati bibliografske jedinice.

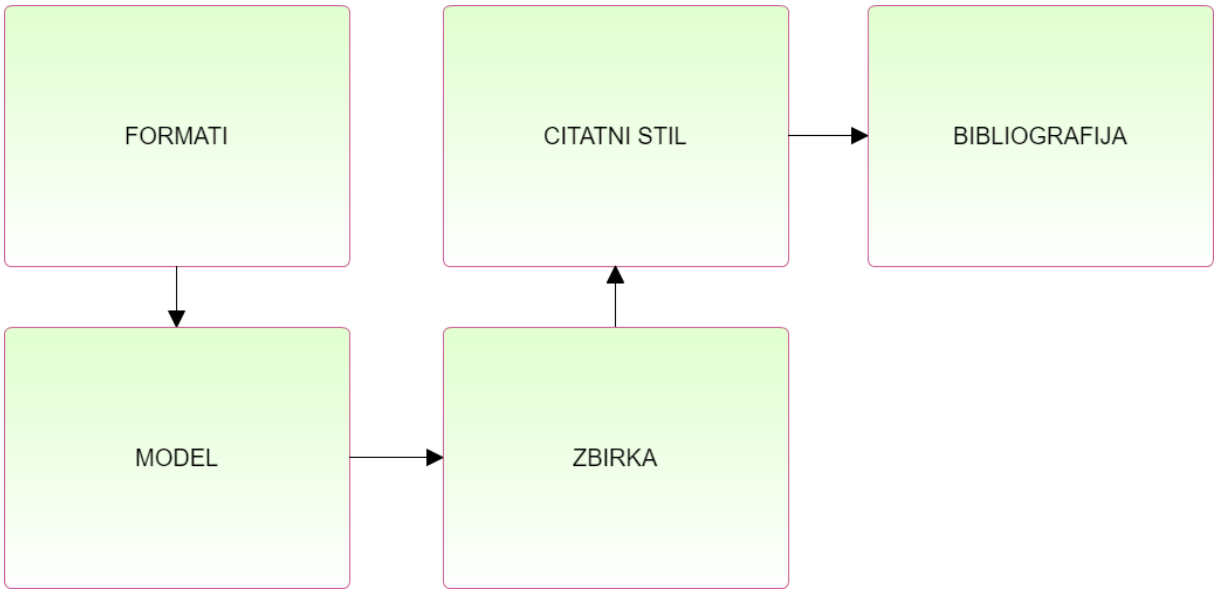
## 5.1. Komponente sustava

Sustav se sastoji od pet glavnih komponenti:

1. **Formati** – dio sustava koji rukuje raznim strukturiranim bibliografskim formatima i njihovim uvozom u zbirku te mapiranjem u modele podataka koje definira bibliograf
2. **Model** – označava model podataka koje definira bibliograf, a koji sadrži deklaraciju i opis polja te definira vrste publikacija kroz upotrebu tih polja
3. **Zbirka** – svi zapisi koje bibliograf unese ručno ili uvozom iz strukturiranih bibliografskih formata (1. komponenta) sadržani su u zbirci na koju se može primjenjivati niz algoritama poput algoritama za pripremu podataka te algoritama za grupiranje, filtriranje, i sl.
4. **Citatni stil** – označava stil navođenja koji će se primijeniti na zbirku kako bi se izradila bibliografija
5. **Bibliografija** – označava posljednju fazu u izradi bibliografije, a sadrži i mogućnost završnog ručnog uređivanja te dodavanja i brisanja bibliografskih jedinica

Navedeni sustav najjednostavnije se može prikazati na način prikazan u grafičkom prikazu 1.

[40] Mapping. URL: http://www.oxforddictionaries.com/definition/english/mapping (29.8.2016.)




*Slika 1. Dijagram osnovnog prikaza sustava*

### 5.1.1. Formati

Strukturirani digitalni podaci mogu biti sadržani u nizu formata, a prototip softverskog rješenja predloženog u ovom radu podržava formate BibTeX, MARC i CSV sa mogućnošću proširenja na dodatne formate.

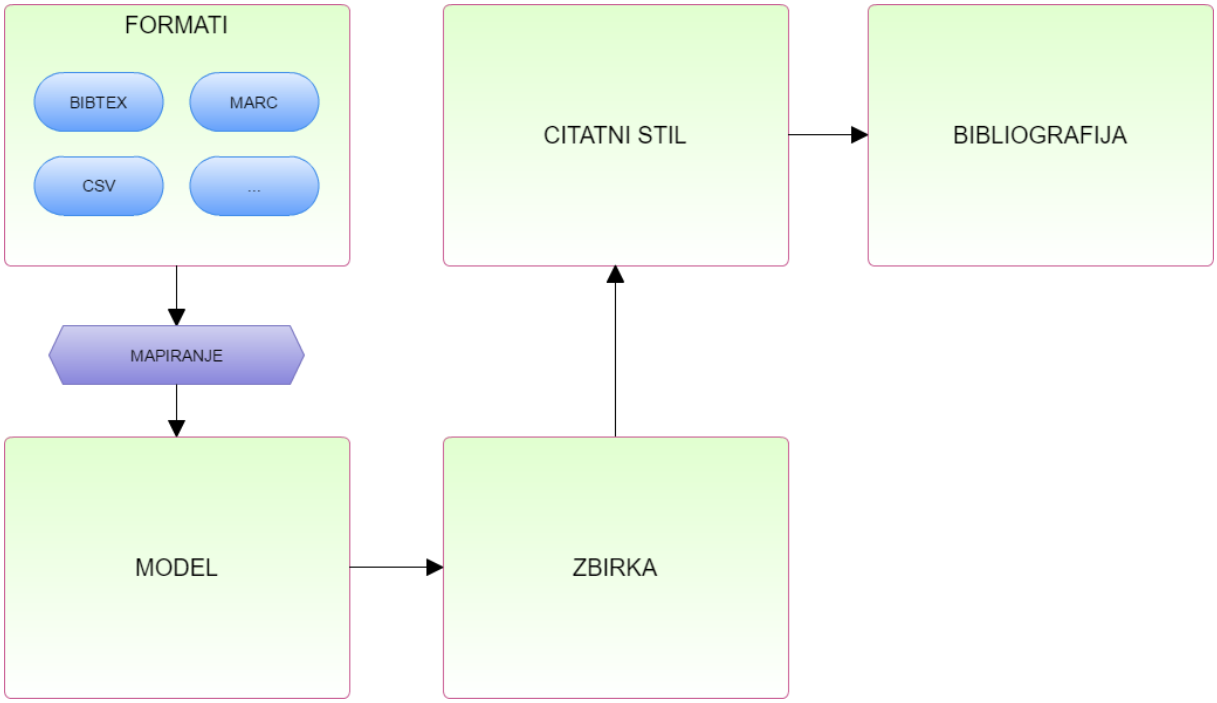
BibTeX je format kojem je prvotna namjena bila korištenje u kombinaciji sa sustavom za slaganje sloga (*typesetting*) LaTeX, a danas služi i za razmjenu bibliografskih podataka. BibTeX je datotečni format čistog teksta kojeg se može čitati i pisati u minimalnom softveru za uređivanje teksta.[41]

MARC je strojno čitljiv format namijenjen knjižnicama za stvaranje i razmjenu kataložnih zapisa.[42] Trenutno u knjižnicama prevladava nekoliko izvedenica MARC formata kao što su MARC21[43] i UNIMARC.[44] MARC je potpuno strukturirani format, s definiranim poljima i pravilima o sadržaju tih polja, dok BibTeX nudi nekoliko gotovih polja za potrebe korisnika ali dopušta i definiranje vlastitih polja.

CSV (*Comma-Seperated Values*) podržava arbitrarne modele podataka i bilo koje nazive polja, a jedino pravilo je da ona moraju biti odvojena određenim graničnikom, koji ne mora nužno biti zarez (polja u CSV datotekama mogu primjerice biti odvojena tabulatorom). CSV je također čest format za razmjenu tabličnih podataka među različitim softverskim

[41] BibTeX.URL: http://www.bibtex.org/About/ (30.8.2016.)
[42] Usp. Understanding MARC Bibliographic. URL: http://www.loc.gov/marc/umb/um01to06.html (30.8.2016.)
[43] MARC Standards. URL: https://www.loc.gov/marc/ (30.8.2016.)
[44] UNIMARC formats and related documentation. URL: http://www.ifla.org/publications/unimarc-formats-and-related-documentation (30.8.2016.)

sustavima. Struktura navedenih formata bit će detaljnije prikazana u poglavlju 5.2. Formati su sadržani u istoimenoj komponenti sustava, kako je prikazano na grafičkom prikazu 2.




*Slika 2. Dijagram mapiranja formata u model izrađen od strane korisnika*

### 5.1.2. Model

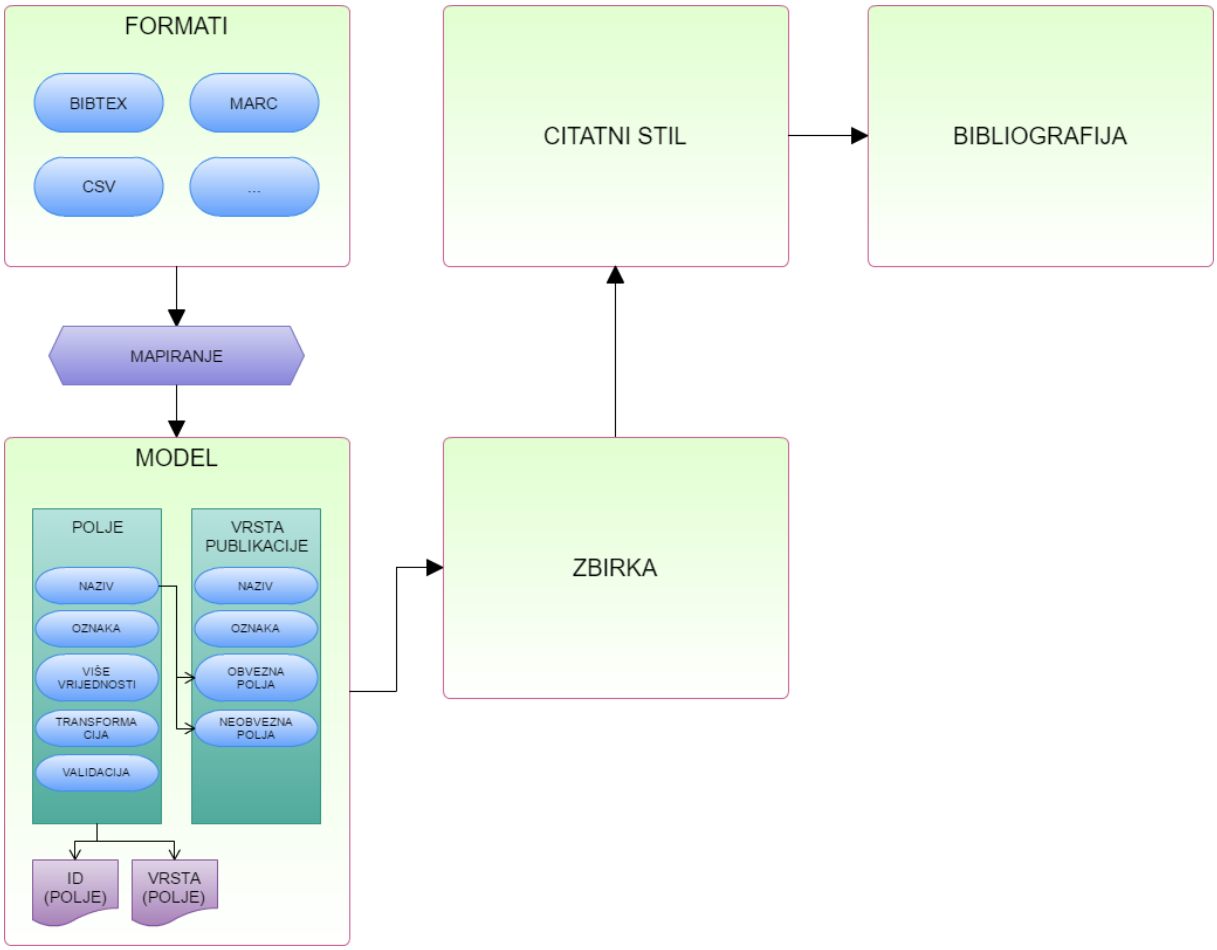
Model se u sustavu ne odnosi na neki specifični model podataka, već označava skup polja i vrsti publikacija koje bibliograf može napraviti i okupiti. Polja modela definira bibliograf, a polje može sadržavati jednu ili više vrijednosti. Polje ima atribute (informacije o svojstvu): naziv polja, oznaka polja, vrstu vrijednosti koje polje dozvoljava, te može i ne mora dozvoljavati unos više vrijednosti. Polja se izrađuju ovisno o potrebama bibliografa i bibliografije koju izrađuje. Često polje može, na primjer, biti „autori“. U tom slučaju, polje će izgledati kako je prikazano u tablici 1.

*Tablica 1. Primjer definiranja polja "Autori publikacije" u sustavu*

| | |
|---|---|
| Ime polja | auths |
| Oznaka polja | Autori publikacije |
| Vrsta polja | *String* |
| Više vrijednosti | Da |

Naziv polja najčešće je interni identifikator polja koji služi za komuniciranje s ostalim poljima i algoritmima unutar sustava. Oznaka polja koristi se u prikazima zapisa, a može se koristiti i u samoj bibliografiji ukoliko bibliograf procjeni da je to potrebno. Čest primjer za prikaz oznake polja su ključne riječi, gdje se navodi i oznaka i vrijednost polja, npr. „Ključne riječi: bibliografija, algoritam, …“. Vrsta polja najčešće sadrži vrste vrijednosti tekst (*string*) ili cijeli broj (*integer*). Primjerice, za polje godine izdavanja poželjno je da se vrsta polja definira kao „cijeli broj“ kako bi se na vrijednostima tih polja mogle provoditi matematičke funkcije i algoritmi specifični za brojčane vrijednosti. Atribut „više vrijednosti“ binarna je varijabla (*boolean*) koja smije biti točna ili netočna, što znači da polje može primati ili više vrijednosti ili jednu. Ovaj atribut iznimno je bitan za sustav, jer raščlanjuje vrijednosti na najmanje jedinice i time omogućuje grupiranje, pravilno navođenje bibliografskih jedinica, itd. Na primjer, iz strukturiranih podataka poput BibTeX-a moguće je dobiti sljedeću (jednu) vrijednost za autore: „Gordon Dunsire and Michael Buckland“. Međutim, s tom vrijednošću nije moguće grupirati bibliografske jedinice po jednom od navedenih autora niti pravilno citirati (u slučaju da se, primjerice, citira samo prvi autor) jer su oba autora sadržana u jednoj vrijednosti. Stoga je potrebno vrijednost odvojiti tako da polje autori sadrži niz vrijednosti („Gordon Dunsire“ i „Michael Buckland“), što je omogućeno navedenim atributom polja „više vrijednosti“.

Vrste publikacija se za potrebe ovog sustava definiraju kroz skupove obveznih i neobveznih polja. Atributi vrste su ime vrste, naziv vrste, obvezna polja i neobvezna polja. Bibliograf tako može za određenu vrstu publikacije definirati obvezna polja u kojima uvijek mora biti vrijednosti u svakom zapisu. Na primjer, za vrstu „članak u časopisu“ moguće je definirati da je „naslov časopisa“ obvezno polje. Kako svaki model može sadržavati više vrsti, za njih je moguće napraviti i zasebne sintakse citatnog stila (za svaku vrstu publikacije po jedna, što je čest slučaj u često korištenim citatnim stilovima poput MLA, APA, Chicago, itd.). Način na koji su polja i vrste publikacije uvršteni u model može se vidjeti u grafičkom prikazu 3. Obvezna gotova polja identifikatora i vrste publikacije bit će objašnjena u poglavlju 5.2.1.




*Slika 3. Dijagram sustava s detaljnim prikazom komponente modela*

### 5.1.3. Mapiranje bibliografskih formata

Kako svaki bibliografski strukturirani format sadrži svoje specifičnosti i načine zapisivanja podataka, nije moguće napraviti univerzalno mapiranje neovisno o formatu, što znači da je za svaki format potrebno imati zasebna pravila mapiranja. Međutim, kako sustav ne koristi zadani model, već ih definira bibliograf, bibliografu je potrebno omogućiti mapiranje u izrađeni model. Dio predloženog sustava koji se bavi mapiranjem čita polja formata te pita bibliografa u koja polja izrađenog modela želi mapirati polja izvornog formata. Format zapisa koji će bibliograf učitati može se prepoznati na više načina: preko datotečne ekstenzije, analizom zapisa (*parsing*) ili nekom drugom metodom. Način na koji će sustav prepoznati format stoga nije iznimno važan, već je važno proslijediti sustavu uputu koje mapiranje treba koristiti. Također, kako bi uopće bilo moguće mapirati iz bibliografskih formata u korisnički model, potrebno je prvo izraditi ili odabrati postojeći model. Stoga predloženi sustav može započeti iz dvije točke: odabirom postojećeg modela ili izradom novog modela. U slučaju izrade novog modela, bibliograf može definirati mapiranja za formate u taj model, ili može početi

ručno dodavati zapise bez definiranja mapiranja (u slučaju da neće uvoziti podatke iz bibliografskih formata). U slučaju odabira postojećeg modela, bibliograf može odmah uvoziti iz bibliografskih formata ukoliko su mapiranja za taj model definirana ili također početi ručno dodavati bibliografske jedinice u zbirku. Mapiranje je prikazano kao algoritam u grafičkim prikazima 2. i 3.

### 5.1.4. Zbirka

U kontekstu predloženog sustava, zbirku se može definirati kao skup bibliografskih zapisa, dodanih ili uvezenih prema modelu u upotrebi. Unutar jedne zbirke, svi bibliografski zapisi moraju koristiti jedan model, što znači i poštivanje svih pravila tog modela kao što su obvezna polja, mogućnosti sadržavanja više vrijednosti u poljima, itd. Pojedinačni zapisi mogu se brisati (premjestiti u „koš za smeće“) ukoliko bibliograf odluči da ih neće uvrstiti u bibliografiju, ali i vratiti iz koša za smeće u zbirku, u slučaju da bibliograf ipak želi uključiti neki zapis koji je ranije izbrisao. U trenutku kada bibliograf radi u zbirci, moguće je dodati vlastite zapise prema modelu u upotrebi. Zapisima se može i skupno manipulirati (*batch edit*), što je i važan dio sustava koji će bibliograf vjerojatno često iskoristiti. Tako je cijelu zbirku moguće grupirati i filtrirati (prema određenom polju iz modela) ili spremiti i učitati stanje zbirke (*save state*).

U slučaju više provedenih algoritama nad zbirkom poput višestrukih filtriranja i grupiranja, bibliograf može poželjeti vratiti stanje zbirke iz određenog koraka ili čak potpuno ukloniti parametre filtriranja i grupiranja. Iz tog razloga, sustav predviđa privremeno spremljena stanja zbirke i mogućnost potpunog uklanjanja parametara filtriranja i grupiranja.

Zbirku je stoga moguće spremiti na dva načina: privremeno ili trajno. Privremeno spremljena zbirka čuva stanja zbirke na kojoj bibliograf trenutno radi poput podataka o filtriranju i grupiranju, a sprema se automatski prije provedbe svakog algoritma. Trajno spremljena zbirka također čuva podatke o filtriranju i grupiranju (slika 4.). Prednost trajno spremljene zbirke je njena dostupnost, a nedostatak je brzina pohrane i učitavanja (u slučaju softverske implementacije).

„Dodatne funkcije“ u sustavu označavaju niz algoritama za pripremu zapisa u zbirci, a mogu se primjenjivati na sve zapise u zbirci ili samo na odabrane zapise. Tako je, u softverskim rješenjima izrađenim prema sustavu, moguće imati algoritme poput:

- Inverzija prema nizu znakova

  Algoritam koji je najkorisniji za inverziju autora koji su napisani u obliku „Prezime, Ime“. Ukoliko ih bibliograf želi u obliku „Ime Prezime“, može pokrenuti inverziju prema znaku zarez (odnosno nizu zarez-razmak).

- Raščlanjivanje u više vrijednosti prema nizu znakova

  U slučaju kada je određeni niz vrijednosti napisan kao jedna vrijednost, npr. „Nakladnik 1 i Nakladnik 2 i Nakladnik 3“, a bibliograf ih želi imati odvojene, izvršit će ovaj algoritam i razdvojiti vrijednost sa nizom znakova „razmak-i-razmak“ („ i “).

- Uklanjanje znakova iz vrijednosti

  Koristi se u slučaju da podaci u bibliografskim zapisima sadrže neželjene znakove, poput razmaka i interpunkcija prije i poslije same vrijednosti ili nasumičnih znakova poput znaka dolara ($) i sl. što se može dogoditi prilikom uvoza iz loše strukturiranih podataka.

- Zamjena niza znakova

  Algoritam kojim je moguće zamijeniti niz znakova poput riječi, brojeva i sl. Primjerice, ukoliko je bibliograf saznao da je u nekim zapisima godina izdavanja pogrešno navedena kao „20013“, navedenu vrijednosti može zamijeniti sa „2013“.

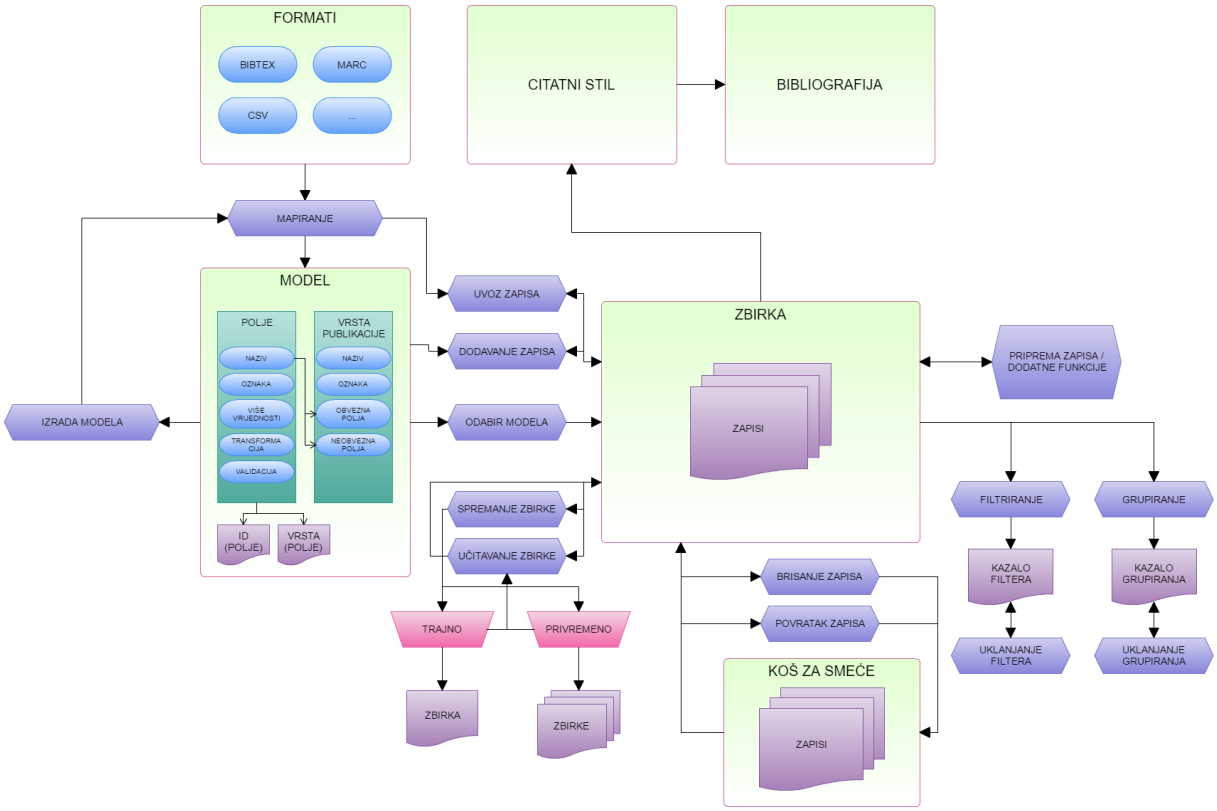
- Promjena velikih i malih slova

  Tekstualne vrijednosti mijenja u jednu od sljedećih:

  - Sva velika slova (PRIMJER VRIJEDNOSTI)
  - Prvo veliko slovo (Primjer vrijednosti)
  - Svako prvo slovo veliko (Primjer Vrijednosti)
  - Sva mala slova (primjer vrijednosti)

  Ovaj algoritam koristan je za standardiziranje izgleda bibliografije, npr. u slučaju naziva časopisa ili naslova publikacije.

Navedeni algoritmi dostupni su u softverskom rješenju prikazanom u ovom radu, ali moguće je dodati i niz drugih koji utječu na zapise u zbirci. Implementacije ovih algoritama moguće je skupno shvatiti kao bibliografsku „alatnu kutiju“. Način izrade zbirke te svi algoritmi vezani uz zbirku u sustavu su vidljivi na grafičkom prikazu 4.



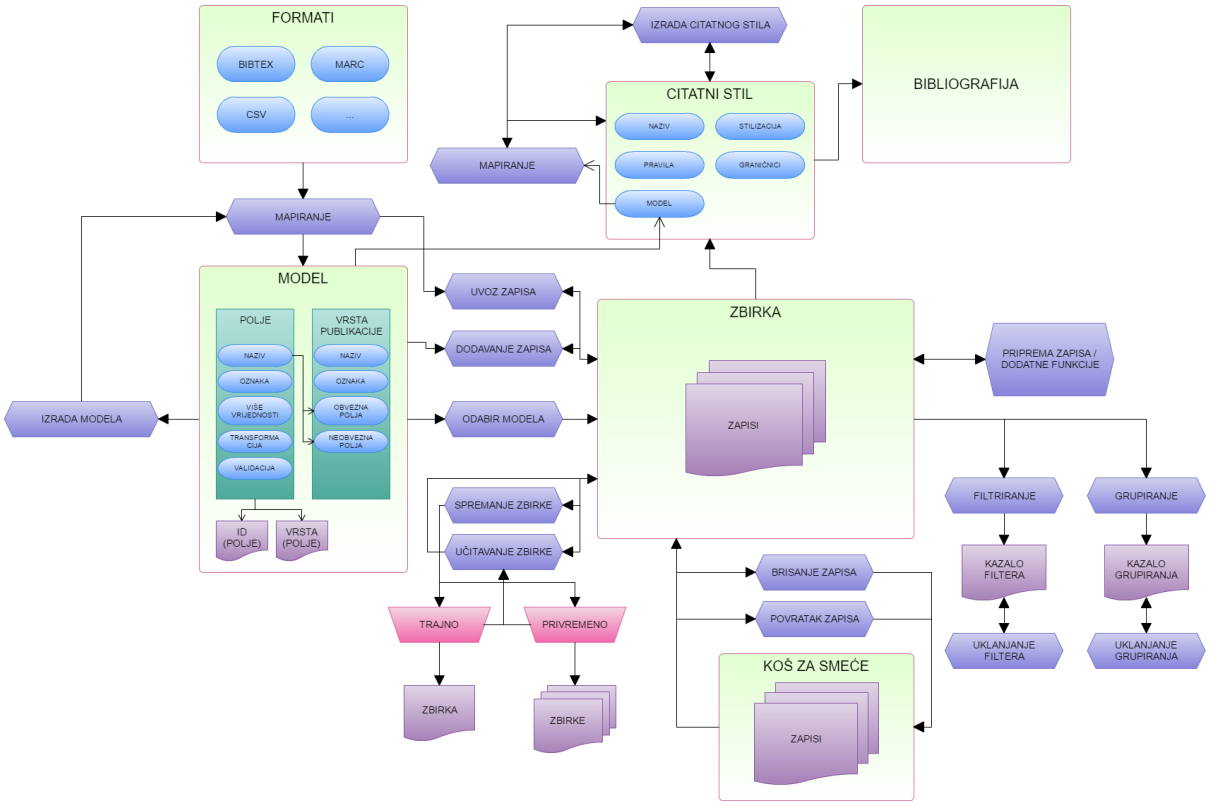

*Slika 4. Dijagram s detaljnim prikazom komponente zbirke*

### 5.1.5. Citatni stilovi

Iako se citatni stilovi najčešće spominju u kontekstu referenci u znanstvenim radovima, isti se mogu koristiti i za stiliziranje bibliografskih jedinica u bibliografijama. Neki od najpoznatijih citatnih stilova su Chicago stil, MLA (Modern Language Assocation), APA (American Psychological Assocation) i dr. Svaki citatni stil sadrži svoj niz pravila prema kojem se pišu reference, ili u slučaju bibliografija, bibliografske jedinice. Pravila citatnih stilova često su sintaktički kompleksna s velikim brojem uvjeta i slučajeva ovisno o tome koji podaci su raspoloživi za citiranje. Problem primjene citatnih stilova na postojeće zapise može se riješiti na više načina, upotrebom različitih algoritama. U poglavlju 5.2.9. bit će prikazana dva moguća rješenja: matrice uvjeta i blokovi sintakse.

Osim postojećih citatnih stilova, bibliograf se može odlučiti napraviti vlastiti stil koji bi odgovarao potrebama određene bibliografije. Takva mogućnost predviđena je u sustavu, a softversko rješenje bit će prikazano u poglavljima 6.1.4. i 6.2.8. Također, u ovom dijelu sustava dolazi do značajne upotrebe već spomenutih vrsta publikacije u modelu. Naime, većina citatnih stilova razlikuje sintaksu za npr. knjigu, članak u časopisu, poglavlje u zborniku, mrežno mjesto, itd.

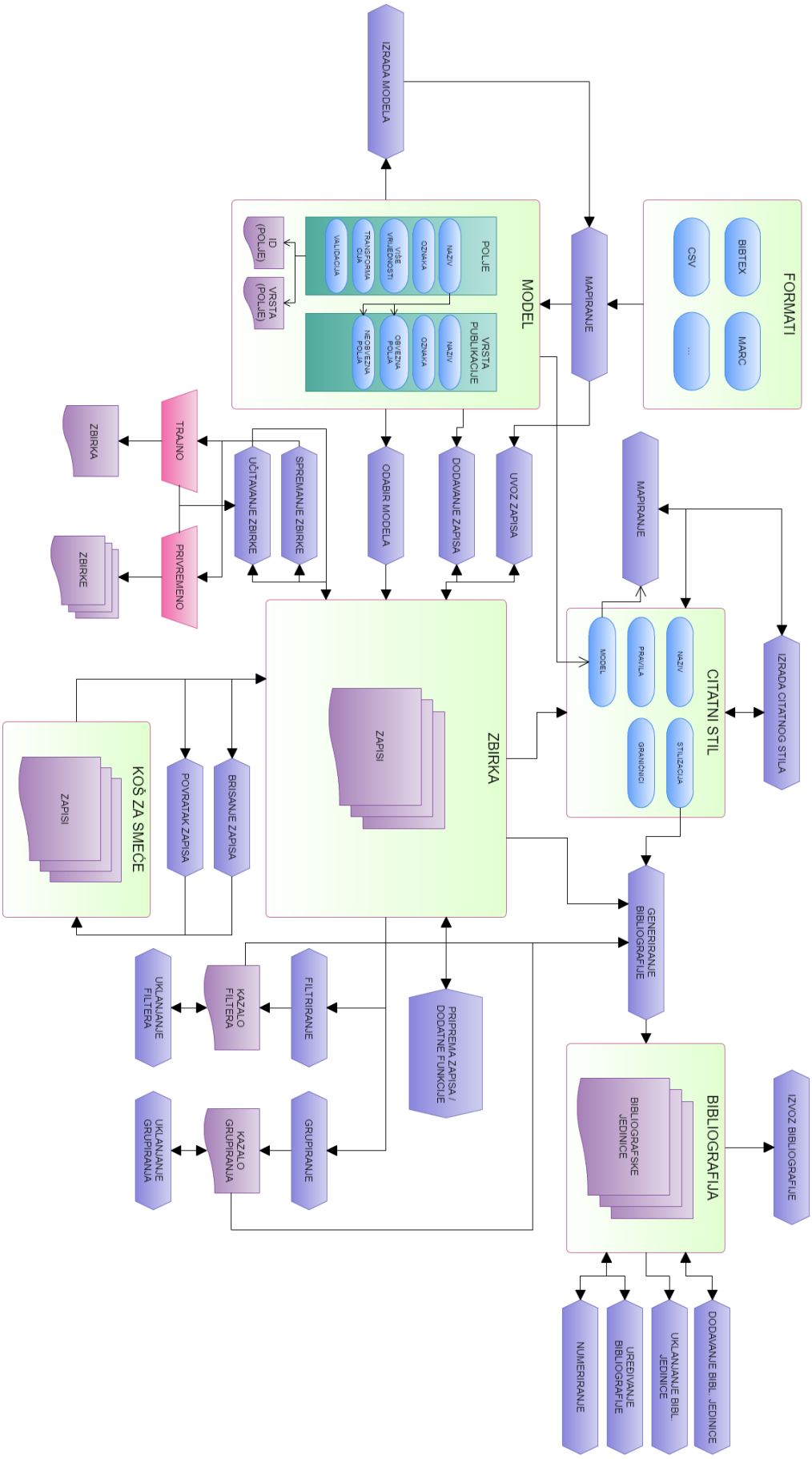
Kako bibliograf u fazi izrade modela definira vlastita polja i vrste publikacija, matrica ili blok sintakse citatnog stila ne može prepoznati koja polja i vrste publikacije modela odgovaraju poljima i vrstama citatnog stila. Iz tog razloga, kao i u dijelu sustava koji se bavi formatima, **citatni stilovi također sadrže funkciju mapiranja u modele** (grafički prikaz 5.).




*Slika 5. Dijagram s detaljnim prikazom komponente citatni stil*

### 5.1.6. Bibliografija

S obavljenim prethodnim koracima i postupcima u sustavu, moguća je izrada konačne bibliografije. Komponenta bibliografije tako uzima podatke o bibliografskim jedinicama, grupiranju, filtriranju i odabrani citatni stil te izrađuje bibliografiju u određenom formatu, npr. u HTML-u, što je i osnovi cilj sustava (grafički prikaz 6.). Za bibliografiju su u sustavu također predviđeni algoritmi poput ručnog dodavanja bibliografskih jedinica i uređivanja istih, iako je u primjeni sustava na softversko rješenje, kao što je slučaj u ovom radu, moguće navedene algoritme objediniti u jedan – uređivanje cjelokupne bibliografije. U sustav je moguće, iako nije nužno, dodati i algoritme poput grafičkog oblikovanja bibliografije.




*Slika 6. Dijagram potpunog sustava sa svim pripadajućim komponentama i algoritmima*

## 5.2. Primjena sustava

U nastavku teksta prikazan je primjer koji uključuje sve faze u procesu kako bi se detaljnije prikazao rad u predloženom sustavu.

### 5.2.1 Izrada modela

Prvi korak u ovom primjeru je izrada modela koji će se koristiti tijekom primjera (algoritam „izrada“ vezan uz komponentu modela na slici 6.). Model može imati i svoj naziv, u slučaju do bibliograf izradi više modela. Model koji će se koristiti u ovom primjeru zvat će se „Jednostavni model“. Jednostavni model sadržavat će polja navedena u tablici 2.

*Tablica 2. Polja jednostavnog modela*

| Naziv | Oznaka | Više vrijednosti |
|---|---|---|
| jednostavni_autori | Autori | Da |
| jednostavni_naslov | Naslov publikacije | Ne |
| jednostavni_izvornik | Naslov izvornika | Ne |
| jednostavni_godina | Godina izdavanja | Ne |
| jednostavni_nakladnik | Nakladnik | Da |

Kao što se može vidjeti iz tablice 2., u jednostavnom modelu više vrijednosti mogu sadržavati samo polja autora i nakladnika. Osim polja navedenih u tablici 2., svaki model obvezno sadrži i polja identifikatora i vrste publikacije. Identifikator je jedinstveni broj kojeg svakom zapisu dodjeljuje sam sustav, a polje vrste publikacije sadrži naziv vrste definirane u modelu. U ovom primjeru definirat će se dvije vrste publikacija, prikazane u tablici 3.

*Tablica 3. Vrste publikacija jednostavnog modela*

| Naziv | Oznaka | Obvezna polja | Neobvezna polja |
|---|---|---|---|
| vrsta_knjiga | Knjiga | jednostavni_naslov | jednostavni_autori<br>jednostavni_godina<br>jednostavni_nakladnik |
| vrsta_clanak | Članak u časopisu | jednostavni_autori<br>jednostavni_naslov<br>jednostavni_izvornik | jednostavni_godina<br>jednostavni_nakladnik |

Vrste su svojevrsni skupovi polja, raščlanjeni u obvezna i neobvezna polja. Nije nužno da svaka vrsta sadrži sva polja, pa tako vrsta „knjiga“ uopće ne sadrži polje „naslov izvornika“, bilo u obveznim ili neobveznim poljima, jer joj ono nije potrebno (u slučaju dodavanja vrste poput „poglavlje u knjizi“, polje naslova izvornika moglo bi se uključiti te bi služilo istoj svrsi). Sama polja sadržana kao obvezna i neobvezna polja u vrstama zapravo su pokazivači (*pointer*) na izvorna polja gdje su sadržani podaci o oznaci i dopuštanju više vrijednosti. Sličan sustav nalazimo u relacijskim bazama podataka, ali i u knjižnični katalozima (pristupnice).

Kako bi se omogućio uvoz podataka iz formata poput BibTeX-a, MARC-a i CSV-a u jednostavni model, potrebno je prvo definirati koja polja u navedenim formatima odgovaraju poljima u jednostavnom modelu, što znači da je potrebno definirati mapiranja. Za formate je često potrebno mapirati i vrste publikacija.

### 5.2.2. Uvoz i mapiranje BibTeX zapisa

Ekvivalenti polja i vrsta publikacija jednostavnog modela u BibTeX-u navedeni su u tablicama 4. i 5. čime se i stvara tablica mapiranja prema kojoj će se navedena polja i vrste publikacija pri uvozu mapirati u izrađeni jednostavni model.

*Tablica 4. Mapiranje vrsti publikacija iz BibTeX formata u jednostavni model*

| BibTeX vrsta publikacije | Jednostavni model (vrijednost vrste publikacije) |
|---|---|
| @article | vrsta_clanak |
| @book | vrsta_knjiga |

*Tablica 5. Mapiranje polja iz BibTeX formata u jednostavni model*

| BibTeX polje | Jednostavni model |
|---|---|
| author | jednostavni_autori |
| title | jednostavni_naslov |
| journal | jednostavni_izvornik |
| year | jednostavni_godina |
| publisher | jednostavni_nakladnik |

Pretpostavimo da uvozimo podatke jednog zapisa iz datoteke *zapis.bib* (BibTeX ekstenzija). Sadržaj datoteke je sljedeći:

```
@article{1355095420130601,
Author = {Ghani, A.A.A. and Parizi, R.M.},
Journal = {Journal of Software},
Keywords = {aspect-oriented programming, program testing,
software fault tolerance},
Number = {6},
Pages = {1281 - 1300},
Title = {Aspect-oriented program testing: an annotated
bibliography.},
Volume = {8},
URL =
{http://search.ebscohost.com/login.aspx?direct=true&db=inh&
AN=13550954&lang=hr&site=ehost-live},
Year = {2013},
}
```

Ovdje je riječ o vrsti „članak u časopisu“, u BibTeX-u označeno sa @article, što je ujedno i otvarajući element zapisa. Slijed brojeva 1355095420130601 identifikator je zapisa, a sve nakon toga napisano je u obliku „polje = {vrijednost}“. Prema zadanoj tablici mapiranja polja, sva polja i vrijednosti unutar njih bit će mapirani u jednostavni model. Nakon kratkog postupka izrade zbirke, temeljene na jednostavnom modelu, moguće je uvesti navedeni zapis, koji će nakon mapiranja u jednostavni model izgledati kako je navedeno u tablici 6.

*Tablica 6. Zapis uvezen i mapiran iz BibTeX formata u jednostavni model*

| | |
|---|---|
| jednostavni_autori | Ghani, A.A.A. and Parizi, R.M. |
| jednostavni_naslov | Aspect-oriented program testing: an annotated bibliography. |
| jednostavni_izvornik | Journal of Software |
| jednostavni_godina | 2013 |
| jednostavni_nakladnik | |

Iz tablice 6. može se vidjeti i primjer ranije spomenutog pisanja dva autora kao jedne vrijednosti – „Ghani, A.A.A. and Parizi, R.M.“. Kasnije će se na ovom primjeru prikazati raščlanjivanje u dvije vrijednosti kako bi se jedinice mogle okupiti prema autoru. Također, pod poljem naslova publikacije, moguće je vidjeti još jedan potencijalni problem, a to je točka na kraju naslova. Iako se može činiti kao sitnica, navedena točka (i drugi slični znakovi sadržani unutar vrijednosti polja) mogu stvoriti problem pri kasnijoj primjeni citatnog stila. Tako će se,

u slučaju da se u citatnom stilu na kraju naslova publikacije navodi točka, stvoriti dvije točke. Kao i s autorima, navedeni problem rješava se primjenom raznih gotovih algoritama na zbirci.

### 5.2.3. Uvoz i mapiranje CSV zapisa

Zapisi u formatu CSV (*Comma-seperated value*) razlikuju se po tome što polja nisu izričito definirana kao u ostalim formatima. Jedino pravilo CSV-a jest da su polja, a samim time i vrijednosti polja, odvojeni zarezom ili nekim drugim znakom. To znači da nije moguće primijeniti jednak način mapiranja kao za BibTeX, iz razloga što nije moguće znati koja polja će CSV sadržavati. Tako u jednom CSV-u autori mogu biti označeni s „AU“, dok se u drugom mogu pojaviti pod „Authors“. Zbog takvih specifičnosti, ali i specifičnosti ostalih formata, svako mapiranje mora se izraditi zasebno u odnosu na format, ali na način da se svako mapiranje može primijeniti na bilo koji model koji bibliograf izradi.

Slijedi primjer CSV datoteke koja je preuzeta iz Web of Scienc baze podataka. CSV preuzet s baza podataka često je razgraničen tabulatorima umjesto zareza iz razloga što autori, ali i druga polja, najčešće sadržavaju zareze.

| PT | AU | TI | SO | PY |
|---|---|---|---|---|
| J | Parizi, RM; Ghani, AAA; Lee, SP | Automated test generation technique for aspectual features in AspectJ | INFORMATION AND SOFTWARE TECHNOLOGY | 2015 |

Bibliograf često može pronaći specifikaciju polja CSV-a, a kod velikog broja polja može i prepoznati koja polja predstavljaju zaglavlja prema njihovim vrijednostima. U ovom slučaju „AU“ su autori, „TI“ je naslov, „SO“ je naslov časopisa, „PY“ godina izdanja, dok oznaka „PT“ označava vrstu publikacije[45], koja je u navedenom zapisu članak u časopisu (J – Journal). Stoga će bibliograf za svaku od tih oznaka odabrati u koja polja modela će ih mapirati (nije nužno da mapira sve oznake već samo one koje su mu potrebne). U slučaju jednostavnog modela, mapiranje polja obavit će se na način naveden u tablici 7, a mapiranje vrste publikacije obavit će se prema kombinaciji polja PT i njene vrijednosti, kako je prikazano u tablici 8.

[45] Web of Science Fields Tags. URL: https://images.webofknowledge.com/WOK46/help/WOS/h_fieldtags.html (12.9.2016.)

*Tablica 7. Mapiranje polja iz CSV formata u jednostavni model*

| | |
|---|---|
| AU | jednostavni_autori |
| TI | jednostavni_naslov |
| SO | jednostavni_izvornik |
| PY | jednostavni_godina |

*Tablica 8. Mapiranje vrsti publikacija iz CSV formata u jednostavni model*

| Vrijednost polja „PT“ u CSV-u | Vrsta publikacije u jednostavnom modelu |
|---|---|
| J | vrsta_clanak |
| B | vrsta_knjiga |

Nakon mapiranja u jednostavni model i uvoza u zbirku, zapis će izgledati kako je prikazano u tablici 9.

*Tablica 9. Zapis uvezen i mapiran iz CSV formata u jednostavni model*

| | |
|---|---|
| jednostavni_autori | Parizi, RM; Ghani, AAA; Lee, SP |
| jednostavni_naslov | Automated test generation technique for aspectual features in AspectJ |
| jednostavni_izvornik | INFORMATION AND SOFTWARE TECHNOLOGY |
| jednostavni_godina | 2015 |
| jednostavni_nakladnik | |

U odnosu na prethodni zapis uvezen iz BibTeX formata, mogu se vidjeti dvije razlike: autori su razgraničeni sa točkom i zarezom, za razliku od „and“, a inicijali imena autora nisu odvojeni točkom već su slova spojena kao da čine jednu riječ (Parizi, RM za razliku od Parizi, R.M.). Još jedna specifičnost zapisa iz CSV datoteke je naslov časopisa, koji je napisan u potpunosti velikim slovima. Sve navedene razlike bit će moguće ujednačiti tijekom upravljanja zbirkom.

### 5.2.4. Uvoz i mapiranje MARC zapisa

Gotovo identična struktura MARC formata (oznaka zapisa, kazalo, kodirani podaci, potpolja) u svim njegovim varijantama dozvoljava upotrebu istog algoritma mapiranja na više

varijanti, kao što su MARC21 i UNIMARC. Navedena mogućnost postiže se uglavnom upotrebom kazala MARC zapisa, u kojem su zapisane sve informacije o poziciji i dužini polja u zapisu. Stoga je potrebno jednostavno izraditi željeno mapiranje iz određenog MARC potpolja u polje modela. U ovom poglavlju, kao primjer će se koristiti UNIMARC zapis, dok će se u poglavlju 6.2. koristiti MARC21 zapisi kako bi se bolje prikazala mogućnost softvera vezana uz pripremu i skupno uređivanje zapisa (pri uklanjanju neželjenih ISBD interpunkcija iz zapisa).

UNIMARC polja unaprijed su definirana globalnim standardom te su poput CSV-a namijenjena strojnom čitanju. Slijedi primjer jednog UNIMARC zapisa, koji je radi čitljivosti dodatno podijeljen i pojednostavljen:

```
00000nam0#2200000# #45##
001                 111227077
005                 20140509182100.1
010   ##    $a      0-671-02423-X
100   ##    $a      20140509d1999#######y0hrvy0191####ba
101   0#    $a      eng
102   ##    $a      US
105   ##    $a      y#######000ad
200   1#    $a      Bag of bones
            $e      [a novel]
            $f      Stephen King
205   ##    $a      1st Pocket Books
210   ##    $a      New York [etc.]
            $c      Pocket Books
            $d      1999
215   ##    $a      731 str.
            $d      17 cm
312   ##    $a      Podnasl. preuzet s 1. om. lista
675   ##    $a      821.111(73)-31
700   #1    $3      930217001
            $a      King
            $b      Stephen
            $c      književnik
801   #0    $a      HR
            $b      GKZD
            $g      HR PPIAK
980   ##    $b      C
981   ##    $z      STR
```

```
990    ##    $a    820-3 KIN b
991    ##    $b    DM63/2014
992    ##    $a    BIBLIOBUS
```

Svako UNIMARC polje sastoji se od oznake polja, indikatora, oznake potpolja i vrijednosti. U gore navedenom primjeru, naslov je kao vrijednost sadržan u polju 200, potpolju $a, te ga je moguće putem tog potpolja i mapirati. Isto vrijedi i za ostala polja. Vrste publikacija moguće je mapirati na više načina: iz oznake zapisa (6 – vrsta zapisa i 7 – bibliografska razina), provjerom upotrebe polja 461 do 464 (skup, podskup, dio, analitički dio) ili provjerom potpolja 100$a (vrsta godine izdavanja).[46] U ovom primjeru, zapisi će biti mapirani prema oznaci zapisa, točnije 7. mjestu oznake zapisa koje označava bibliografsku razinu. Ukoliko je bibliografska razina „m“ (omeđena publikacija) zapis će biti mapiran u vrstu publikacije „knjiga“ u jednostavnom modelu, dok će bibliografska razina „a“ (analitička jedinica – sastavnica) biti mapirana u vrstu publikacije „članak u časopisu“. Mapiranje polja gore navedenog zapisa u jednostavni model navedeno je u tablici 10, a mapiranje vrsta publikacija u tablici 11.

*Tablica 10. Mapiranje polja iz UNIMARC formata u jednostavni model*

| | |
|---|---|
| 200 $f | jednostavni_autori |
| 200 $a | jednostavni_naslov |
| 210 $d | jednostavni_godina |
| 210 $c | jednostavni_nakladnik |

*Tablica 11. Mapiranje vrsti publikacija iz UNIMARC formata u jednostavni model*

| Bibliografska razina | Vrsta publikacije u jednostavnom modelu |
|---|---|
| m | vrsta_knjiga |
| a | vrsta_clanak |

Kako je u slučaju ovog zapisa riječ o knjizi, vidljivo je kako se upotrebljava polje nakladnika, za razliku od prethodna dva slučaja, ali zato nema polja naslova izvornika. Informacije o autoru osim potpolja 200$f moguće je mapirati i iz potpolja 700$a i 700$b

[46] UNIMARC: bibliografski format. Zagreb: Hrvatsko knjižničarsko društvo; Zadar, Sveučilište u Zadru, 2009. Str. 15., str. 24. i str. 99.

(potpolje $a za prezime i $b za ime). U tom slučaju svako ponavljanje polja 700 uzimat će se odmah kao zasebna vrijednost (iz razloga što polje „jednostavni_autori“ može sadržavati više vrijednosti), ali potrebno je i spojiti potpolja $a i $b u željeni oblik autorova imena i prezimena. Tako bi se dva potpolja u UNIMARC-u spojila u jedno polje modela (npr. 700$a i 700$b u „jednostavni_autori“). Međutim, za ovaj primjer prikazat će se mapiranje potpolja 200$f zbog jednostavnosti te iz razloga što prethodni zapisi u CSV-u i BibTeX-u ne odvajaju ime i prezime autora. Stoga zapis uvezen iz UNIMARC-a u jednostavnom modelu izgleda kako je navedeno u tablici 12.

*Tablica 12. Zapis uvezen i mapiran iz UNIMARC formata u jednostavni model*

| | |
|---|---|
| jednostavni_autori | Stephen King |
| jednostavni_naslov | Bag of bones |
| jednostavni_godina | 1999 |
| jednostavni_nakladnik | Pocket Books |

### 5.2.5. Priprema zapisa u zbirci

Sa zadnjim primjerom uvoza tri su zapisa u zbirci temeljenoj na izrađenom jednostavnom modelu. Podaci o mapiranju u jednostavni model dio su metapodataka tog modela. Tako uvozom iz formata za koji su već definirane tablice mapiranja (u ovom slučaju MARC, BibTeX i CSV) sustav bibliografske zapise iz tih formata prosljeđuje direktno u zbirku, prema modelu u upotrebi (u ovom primjeru jednostavnom modelu), što znači da nije nužno pri svakom uvozu definirati mapiranje. Potrebno je istaknuti kako svaki zapis, osim polja navedenih tijekom mapiranja sadrži i polje identifikatora te polje vrste publikacije (u ovom slučaju to može biti knjiga ili članak).

Kako oba zapisa vrste „članak u časopisu“ i jedini zapis vrste „knjiga“ sadrže sva obvezna polja, provjera prisutnosti obveznih polja neće vratiti nikakve pogreške. U slučaju pogreške, odnosno ukoliko neki zapis ne sadrži sva obvezna polja, bibliografu se može ponuditi da navedeni zapis izbriše ili unese vrijednosti za sva obvezna polja. Moguća je još jedna provjera valjanosti, a to je provjera valjanosti vrste polja. Kao što je ranije naglašeno, svako polje je tipično ili *string* ili *integer*, odnosno može sadržavati kombinaciju znakova ili samo cijele brojeve. Ova provjera korisna je u slučaju, primjerice, polja godine izdavanja. U jednostavnom modelu kojim se služi ovaj primjer, jedino polje koje je izričito naglašeno kao *integer* je upravo godina izdavanja. Provjerom valjanosti polja svih zapisa, zapisi će opet biti

valjani (godine su u zapisima zapisane kao „2013“, „2015“ i „1999“). Nevaljana vrijednost za polje godine izdavanja bila bi s točkom na kraju („2013.“) što može biti čest slučaj, posebno u MARC21 zapisima (iz razloga što MARC21 pravila nalažu primjenu ISBD interpunkcija u zapisima)[47][48], što će biti detaljnije prikazano u poglavlju 6.2. Kod provjere valjanosti polja, točnije u slučaju polja koja dopuštaju samo cijele brojeve, sustav (a samim time i softversko rješenje) može automatski ili sa prethodnim dopuštenjem bibliografa ukloniti sve znakove koji nisu brojevi. Alternative su i ručno mijenjanje vrijednosti polja, uklanjanje vrijednosti ukoliko godina izdavanja nije obvezno polje te uklanjanje cijelog zapisa ukoliko godina izdavanja jest obvezno polje. Ako u kontekstu neke bibliografije nije poželjno da se godina ograniči na cijeli broj, bibliograf može stvoriti model u kojem je dopušten širi slučaj vrijednosti za polje „godina izdavanja“. Navedeno je podrobnije opisano u poglavljima 6.1.1. i 6.2.1.

Prije primjene algoritama poput grupiranja, autore je potrebno razdvojiti u zasebne vrijednosti. Polje autora trenutno u zbirci sadrži sljedeće vrijednosti:

1. Ghani, A.A.A. and Parizi, R.M.
2. Parizi, RM; Ghani, AAA; Lee, SP
3. Stephen King

Kao što se može vidjeti, autori su sadržani u jednoj vrijednosti za svaki zapis. Također, razdvojeni su različitim slijedom znakova (u prvom zapisu s „and“, u drugom sa točkom i zarezom), a problem stvaraju i inicijali imena (u prvom zapisu odvojeni točkama, u drugom potpuno spojeni). Sve navedene razlike potrebno je ujednačiti, a autore razdvojiti u zasebne vrijednosti, tako da vrijednosti izgledaju kako je prikazano u tablici 13.

*Tablica 13. Ciljani način prikaza imena i prezimena autora u zbirci*

| | |
|---|---|
| Zapis 1 | Ghani, AAA<br>Parizi, RM |
| Zapis 2 | Parizi, RM<br>Ghani, AAA<br>Lee, SP |
| Zapis 3 | King, S |

[47] 300 - Physical Description (R). URL: https://www.loc.gov/marc/bibliographic/bd300.html (29.8.2016.)
[48] ISBD punctuation in the MARC 21 Bibliographic Format. URL: https://www.loc.gov/marc/marbi/2010/2010-dp01.html (29.8.2016.)

Navedeni oblici samo su primjer, na bibliografu ostaje kako će točno oblikovati imena i prezimena autora. Moguće je i razdvojiti sve inicijale imena točkama ili ostaviti puna imena gdje su ona dostupna (za razliku od skraćivanja na, primjerice, „King, S“).

Kako bi se došlo do vrijednosti navedenih u tablici 13., potrebno je prvo provesti razdvajanje. Algoritmom razdvajanja moguće je razdvojiti sve vrijednosti u svim poljima „jednostavni_autori“ po nizu znakova ' and ' (razmak-and-razmak). Time će se razdvojiti autori u prvom zapisu, što je prikazano u tablici 14.

*Tablica 14. Razdvajanje jedne vrijednosti polja autora u dvije, prema nizu znakova „and“*

| Izvorna vrijednost | Vrijednosti nakon razdvajanja |
|---|---|
| Ghani, A.A.A. and Parizi, R.M. | Ghani, A.A.A. |
| | Parizi, R.M. |

Isti algoritam potrebno je ponoviti s razdvajanjem po znakovima '; ' (točka zarez-razmak). Tako se u drugom zapisu dolazi do vrijednosti prikazanih u tablici 15.

*Tablica 15. Razdvajanje jedne vrijednosti polja autora u tri, prema nizu znakova ";"*

| Izvorna vrijednost | Vrijednosti nakon razdvajanja |
|---|---|
| Parizi, RM; Ghani, AAA; Lee, SP | Parizi, RM |
| | Ghani, AAA |
| | Lee, SP |

Za zadnji zapis nije potrebno provesti nikakvo razdvajanje jer sadrži samo jednog autora, tj. jednu vrijednost u polju „jednostavni_autori“. Potrebno je i napomenuti kako nije nužno provoditi razdvajanje na svakom zapisu zasebno, već se algoritam može primijeniti na sve zapise. Tako bi, u slučaju da primjer sadrži više zapisa u kojima su autori napisani na način „Autor 1 and Autor 2“ algoritam razdvajanja po znakovima „and“ razdvojio i te vrijednosti zapisa, ali kako bi primjer bio jednostavniji, prikazana su samo tri različita zapisa.

Ipak, kako bi se svi zapisi uskladili, na zadnjem zapisu, odnosno na polju autora, provest će se inverzija i skraćivanje imena. Algoritmu inverzije potrebno je navesti kojim nizom znakova će razdvojiti novi slijed imena i prezimena. Tako se inverzijom sa znakovima ', ' (zarez-razmak) u zadnjem zapisu dolazi do vrijednosti navedenih u tablici 16.

*Tablica 16. Inverzija autorova imena i prezimena*

| Izvorna vrijednost | Vrijednosti nakon inverzije |
|---|---|
| Stephen King | King, Stephen |

Prethodni zapisi ostat će neizmijenjeni jer su autori već u obliku prezime-zarez-razmak-ime, što algoritam prepoznaje. Za usklađivanje autora potrebno je još provesti skraćivanje imena na inicijale (bez znaka za razgraničavanje) na jednom zapisu, kako je navedeno u tablici 17. Postupak skraćivanja imena na inicijale ovdje je ponuđen samo kao primjer mogućeg algoritma, odnosno nije uključen u softversko rješenje prikazano u ovom radu.

*Tablica 17. Skraćivanje autorova imena na inicijale*

| Izvorna vrijednost | Vrijednosti nakon skraćivanja |
|---|---|
| King, Stephen | King, S |

Na isti način moguće je skratiti i prezime, ali u ovom primjeru to se neće koristiti. Posljednji korak usklađivanja je uklanjanje znakova točke u inicijalima imena autora. Ovaj algoritam opet se provodi na svim autorima, ali kako su inicijali odvojeni točkama samo u prvom zapisu, tako će se u tom zapisu dogoditi promjena navedena u tablici 18. Točke u inicijalima moguće je dodati i prilikom citiranja, a puna imena sačuvati, ovisno o korisničkim željama. Slučaj u ovom primjeru zadan je unaprijed jer je moguć za gotovo sva imena autora.

*Tablica 18. Uklanjanje znakova točke iz inicijala autora*

| Izvorna vrijednost | Vrijednost nakon uklanjanja |
|---|---|
| Ghani, A.A.A. | Ghani, AAA |
| Parizi, R.M. | Parizi, RM |

Time su imena i prezimena te način oblikovanja autora usklađeni, ali postoji još polja koje je potrebno uskladiti prije prelaska na grupiranje i izradu bibliografije. Jedno takvo polje je naslov izvornika, što se odnosi na prva dva zapisa. Vrijednosti tih polja prikazana su u tablici 19.

*Tablica 19. Vrijednosti polja naslova izvornika prije usklađivanja*

| | |
|---|---|
| Zapis 1 | Journal of Software |
| Zapis 2 | INFORMATION AND SOFTWARE TECHNOLOGY |

Svaku riječ osim pridjeva u naslovu bilo koje vrste u engleskom jeziku uobičajeno je pisati velikim početnim slovom, a takav algoritam moguće je provesti i na navedenim naslovima časopisa, što je spomenuto u prethodnom poglavlju 5.1.4. kao algoritam promjene velikih i malih slova. Provedbom navedenog algoritma prvi naslov izvornika ostaje nepromijenjen, dok se drugi iz „INFORMATION AND SOFTWARE TECHNOLOGY“ pretvara u „Information and Software Technology“, čime se mijenjaju vrijednosti polja naslova izvornika u navedena dva zapisa kako je prikazano u tablici 20.

*Tablica 20. Primjena funkcije promjene velikih i malih slova nad vrijednostima polja naslova izvornika*

| Izvorna vrijednost | Vrijednost nakon promjene |
|---|---|
| Journal of Software | Journal of Software |
| INFORMATION AND SOFTWARE TECHNOLOGY | Information and Software Technology |

Problem s promjenom velikih i malih slova javlja se u akronimima i nazivima koji sadrže kombinaciju velikih i malih slova. Primjerice, ukoliko naslov sadrži riječ „BibTeX“, promjenom u rečenični stil (prvo slovo prve riječi velikim slovom) ta riječ postaje „bibtex“ što nije pravilno pisanje te riječi. Sličan problem javlja se i s akronimima, primjerice za akronim „MARC“, kojeg algoritam može pogrešno pretvoriti u „marc“, posebno ako je riječ o naslovu u verzalu. Rješenje se može pronaći u metodama računalne obrade prirodnog jezika, što je moguće uključiti u sustav (kao dio dodatnih funkcija), a samim time i u izvedeno softversko rješenje. Međutim, zbog kompleksnosti uključivanja takvog sustava te činjenice kako to nije cilj ovog rada, takvo rješenje nije uključeno u softver prikazan u ovom radu.

Posljednje ujednačavanje odnosi se na naslove publikacija. Svi naslovi trenutno su napisani velikim slovom prve riječi, što je prihvatljivo. Međutim, naslov jedinice prvog zapisa završava s točkom, dok u ostala dva naslova to nije slučaj. Stoga je potrebno provesti funkciju koja uklanja znak točku iz vrijednosti polja naslova, čime će se dobiti naslov naveden u tablici 21. ujednačen s naslovima u ostalim zapisima.

*Tablica 21. Uklanjanje znaka točke iz naslova članka*

| Izvorna vrijednost | Vrijednost nakon promjene |
|---|---|
| Aspect-oriented program testing: an annotated bibliography. | Aspect-oriented program testing: an annotated bibliography |

Time je završeno ujednačavanje vrijednosti u svim zapisima u ovom primjeru, čime se omogućava primjena drugih algoritama poput grupiranja i filtriranja. Nakon ujednačavanja, sva tri zapisa izgledaju kako je navedeno u tablici 22.

*Tablica 22. Zapisi u zbirci nakon usklađivanja*

| Zapis br. 1 | |
|---|---|
| Id | 1 |
| Type | vrsta_clanak |
| jednostavni_naslov | Aspect-oriented program testing: an annotated bibliography |
| jednostavni_autori | Ghani, AAA |
| | Parizi, RM |
| jednostavni_izvornik | Journal of Software |
| jednostavni_godina | 2013 |
| jednostavni_nakladnik | |
| Zapis br. 2 | |
| Id | 2 |
| Type | vrsta_clanak |
| jednostavni_naslov | Automated test generation technique for aspectual features in AspectJ |
| jednostavni_autori | Parizi, RM |
| | Ghani, AAA |
| | Lee, SP |
| jednostavni_izvornik | Information and Software Technology |
| jednostavni_godina | 2015 |
| jednostavni_nakladnik | |
| Zapis br. 3 | |
| Id | 3 |

| Type | vrsta_knjiga |
|---|---|
| jednostavni_naslov | Bag of bones |
| jednostavni_autori | King, S |
| jednostavni_godina | 1999 |
| jednostavni_nakladnik | Pocket Books |

### 5.2.6. Grupiranje

Grupiranje je obvezna faza u izradi bibliografije što se jasno iskazuje u definicijama bibliografije navedenim u ranijim poglavljima. Nazivi za grupiranje u definicijama bibliografija su različiti, ali uvijek prisutni. Neke definicije grupiranje opisuju kao redanje u skladu s određenim sustavom, a neke ih opisuju kao klasifikaciju. Grupiranje se može opisati kao okupljanje bibliografskih jedinica po određenom parametru dostupnom u bibliografskim podacima. Tako je moguće grupirati po npr. autoru, godini izdavanja, mjestu izdavanja, itd. Ukoliko su takvi podaci dostupni, moguće je grupiranje i po „nekonvencionalnim“ parametrima. Tako je u slučaju bibliografije koja sadrži bibliografske jedinice iz područja kemije moguće grupirati jedinice po kemijskom spoju o kojem bibliografska jedinica govori. Svaka grupacija može sadržavati bibliografske jedinice ili druge grupacije (grupiranje na više razina).

U slučaju zbirke u ovom primjeru, algoritam grupiranja (npr. po autoru) u sustavu prolazi kroz svaku bibliografsku jedinicu i stvara kazalo (u obliku ključ-vrijednost) u kojem je ključ svaki autor, a vrijednost je popis identifikatora zapisa u kojima se taj autor javlja. Takvo kazalo u ovom primjeru izgledalo bi kako slijedi:

**Ghani, AAA:** 1, 2
**King, S:** 3
**Lee, SP:** 2
**Parizi, RM:** 1, 2

Drugim riječima, kada bi se identifikatori zapisa zamijenili s naslovima publikacija tih zapisa, navedeno kazalo bi odredilo način snalaženja u bibliografiji, odnosno povećalo efikasnost ciljane upotrebe. Iz kazala grupiranja vidljivo je okupljanje prema parametru, a na taj način se može i prikazati cijela zbirka:

**Ghani, AAA**

Aspect-oriented program testing: an annotated bibliography

Automated test generation technique for aspectual features in AspectJ

**King, S**

Bag of bones

**Lee, SP**

Automated test generation technique for aspectual features in AspectJ

**Parizi, RM**

Aspect-oriented program testing: an annotated bibliography

Automated test generation technique for aspectual features in AspectJ

Grupirati je moguće po bilo kojem polju modela kojeg je definirao bibliograf. U višerazinskom grupiranju, grupirati je moguće po bilo kojem polju osim polja po kojim su grupirane grupe na višim razinama. Kada bi se u navedeni primjer grupiranja dodalo grupiranje po godini izdavanja kao grupiranje druge razine, rezultat bi bio sljedeći:

**Ghani, AAA**

**2013**

Aspect-oriented program testing: an annotated bibliography

**2015**

Automated test generation technique for aspectual features in AspectJ

**King, S**

**1999**

Bag of bones

**Lee, SP**

**2015**

Automated test generation technique for aspectual features in AspectJ

**Parizi, RM**

**2013**

Aspect-oriented program testing: an annotated bibliography

**2015**

Automated test generation technique for aspectual features in AspectJ

Potrebno je ponovno istaknuti kako rezultat algoritma grupiranja nije bibliografija, već kazalo identifikatora zapisa grupiranih prema određenim parametrima. Takvo kazalo kasnije se može koristiti pri izradi same bibliografije. Važna prednost digitalne sredine je što se parametri

za grupiranje mogu definirati *ad hoc* čime se omogućuje lako strukturiranje istih bibliografskih jedinica za različite informacijske potrebe.

### 5.2.7. Filtriranje

Osim grupiranja, sustavom su predviđeni i drugi algoritmi, a jedan od njih je filtriranje. Filtriranje koristi zadani polje-vrijednost par te vraća sve zapise koji odgovaraju korisničkom upitu. Tako bi filtriranjem zbirke iz ovog primjera prema polju godina (jednostavna_godina) i vrijednosti „2013“ rezultat bio članak „Aspect-oriented program testing: an annotated bibliography“, filtriranjem po polju autori (jednostavni_autori) i vrijednosti „Parizi, RM“ rezultat bi bili članci „Aspect-oriented program testing: an annotated bibliography“ te „Automated test generation technique for aspectual features in AspectJ“, i tako dalje. Na filtriranoj zbirci moguće je provoditi dodatna filtriranja i druge algoritme, npr. grupiranje.

### 5.2.8. Koš za smeće

U slučaju da je potrebno neke zapise maknuti iz zbirke, odnosno ne uvrstiti u bibliografiju kao bibliografske jedinice, njih je moguće izbrisati. Izbrisani zapisi ne brišu se zauvijek već se spremaju u tzv. koš za smeće, što je zapravo sekundarna zbirka (sa istim svojstvima kao izvorna) koja sadrži zapise koje je bibliograf izbrisao. Brisanje je moguće kombinirati i sa filtriranjem, primjerice, ukoliko bibliograf iz zbirke želi izbaciti sve zapise kojima je godina izdavanja manja od 2005. Prvi korak u tom slučaju bi bio filtriranje prema zapisima gdje je godina izdavanja manja od 2005, a drugi korak brisanje svih zapisa dobivenih filtriranjem (označavanjem svih dobivenih zapisa), te naposljetku uklanjanje parametara filtriranja, čime se dolazi do zbirke sa zapisima novijim od 2005. godine.

Izbrisane zapise moguće je i vratiti u zbirku. Osim toga, koš za smeće tako može poslužiti i kao privremeni spremnik za razne potrebe bibliografa, primjerice za odvajanje zapisa koje bibliograf želi ručno mijenjati i sl.

### 5.2.9 Citatni stilovi i bibliografija

Bibliografija se, u kontekstu sustava, odnosi na skup bibliografskih jedinica izvedenih iz zapisa zbirke koje mogu biti grupirane prema određenim parametrima i oblikovane prema nekom citatnom stilu. Bibliografija osim dobavljanja podataka iz zbirke ni na koji način ne smije utjecati na same zapise u zbirci što znači da ne smije mijenjati, brisati ili dodavati polja i

vrijednosti polja zapisa. U tom smislu, bibliografija je nakon generiranja samostalni element u sustavu.

#### *5.2.9.1. Primjena postojećih citatnih stilova*

Citatni stilovi poput APA, MLA, itd. u kontekstu ovog sustava smatraju se „postojećim citatnim stilovima“, a njihova se primjena na zbirku, odnosno zapise u zbirci, razlikuje od primjene novih citatnih stilova koje izradi sam bibliograf. Za primjenu stila citiranja stoga je važno da sustav formira navod iz zapisa prema nekom citatnom stilu te tako izradi bibliografsku jedinicu i uključi je u bibliografiju.

Međutim, problem u primjeni postojećih citatnih stilova stvara činjenica da bibliograf u svom modelu može imati bilo koja polja, a ne nužno polja koja postojeći citati stilovi zahtijevaju. Samim time, algoritam primjene citatnog stila ne može znati koja polja treba koristiti za stvaranje bibliografske jedinice. Nadalje, citatni stilovi oslanjaju se na vrste publikacija pa se tako za, primjerice, članak u časopisu koriste pravila citiranja različita od onih za citiranje omeđenih publikacija. To stvara dodatan problem iz razloga što bibliograf u modelu također definira vlastite vrste publikacija.

Situaciju dodatno kompliciraju i sama pravila većine postojećih citatnih stilova, koja su izrazito kompleksna, a u kojima se često za svaki specifični slučaj (npr. jedinica ima podatak o autoru, nema podatak o godini te ima podatak o nakladniku) koristi različita sintaksa koja uključuje prateće i vodeće znakove, interpunkcije ili nizove znakova pri pisanju podataka.

Problem primjene postojećih citatnih stilova može se riješiti na više načina, jedan od kojih je kombinacija matrice uvjeta i mapiranja polja modela u polja citatnog stila. Primjer dijela matrice za APA citatni stil prikazan je u tablici 23. Matrica uvjeta je u slučaju većine postojećih citatnih stilova iznimno složena, ali njena upotreba osigurava točnu i doslijednu primjenu citatnih stilova na zapise.

Matrica se sastoji od niza da/ne (*boolean*) uvjeta, koji oslikavaju dostupnost podataka u zapisu na kojeg je potrebno primijeniti citatni stil, te od sintakse napisane specifično za taj uvjet. Primjerice, ukoliko određeni zapis sadrži informacije o naslovu knjige, mjestu izdavanja, godini izdavanja i nakladniku, ali ne sadrži informaciju o autorima, ili je knjiga anonimno djelo, primijenit će se sintaksa iz označenog retka u tablici 23.

*Tablica 23. Primjer dijela matrica za APA citatni stil (za knjige)*

| Autori? | Naslov knjige? | Mjesto izdavanja? | Godina izdavanja? | Nakladnik? | Sintaksa |
|---|---|---|---|---|---|
| Da | Da | Da | Da | Da | Autori. (Godina izdavanja). *Naslov knjige*. Mjesto izdavanja, Nakladnik. |
| **Ne** | **Da** | **Da** | **Da** | **Da** | **(Godina izdavanja). *Naslov knjige*. Mjesto izdavanja, Nakladnik.** |
| Ne | Da | Ne | Da | Ne | (Godina izdavanja). *Naslov knjige*. |
| Da | Da | Ne | Ne | Da | Autori. *Naslov knjige*. Nakladnik. |

U slučaju APA citatnog stila iz ovog primjera, naslov knjige piše se u kurzivu, što se u softverskim implementacijama sustava može označiti HTML oznakama za kurziv, „<i“> i „</i>“. Sve navedeno između te dvije oznake prikazat će se u kurzivu.

Drugo rješenje za primjenu postojećeg citatnog stila je kombinacija upotrebe blokova sintakse, redoslijeda tih blokova i mapiranja citatnog stila u model. Za razliku od matrice uvjeta, blokovi sintakse mogu se napisati brže, ali nisu potpuno vjerodostojni. Tablica blokova sintakse sastoji se od dva stupca: u lijevom stupcu nalaze se nazivi polja citatnog stila, a u desnom blok sintakse za to polje. Primjer blokova sintakse prikazan je u tablici 24.

*Tablica 24. Blokovi sintakse za APA citatni stil za knjigu*

| Naziv polja | Blok sintakse |
|---|---|
| Naslov knjige | *Naslov knjige*. |
| Autori | Autori . |
| Godina izdavanja | (Godina izdavanja). |
| Nakladnik | Nakladnik. |
| Mjesto izdavanja | Mjesto izdavanja, |

Redoslijed blokova, primjerice za knjigu u APA citatnom stilu, je sljedeći: Autori, godina izdavanja, naslov knjige, mjesto izdavanja, država, nakladnik. Kako polje „država“ u jednostavnom modelu iz ovog primjera ne postoji, algoritam bi navedeno polje (i cijeli blok sintakse) izbacio iz cjelokupne sintakse, čime bi se dobila slijedeća sintaksa: **Autori. (Godina izdavanja).** ***Naslov knjige*****. Mjesto izdavanja, Nakladnik.**

Upotrebom blokova sintakse ili matrice uvjeta sprječavaju se prazni elementi koji bi se ukazali primjenom najjednostavnije zamjene *placeholder*-a. Primjerice, u nedostatku informacije o godini izdavanja (u APA citatnom stilu za knjigu), algoritam koji ne upotrebljava blokove sintakse ili matricu uvjeta, pogrešno bi stvorio sljedeću sintaksu: **Autori. ().** ***Naslov knjige*****. Mjesto izdavanja, Nakladnik.**, što svakako nije poželjno za izradu bibliografije.

Sama matrica uvjeta ili blokovi sintakse nisu dovoljni za primjenu citatnog stila na zbirku iz ovog primjera, iz razloga što sustav nije u mogućnosti sam prepoznati da su, npr. „*Naslov knjige*“ i „jednostavni_naslov“ isto polje. Stoga je potrebno napraviti i tablicu mapiranja te mapirati sva polja (koja bibliograf želi koristiti u bibliografiji) u polja citatnog stila. Dio primjera tablice mapiranja citatnog stila prikazan je u tablici 25.

*Tablica 25. Primjer mapiranja citatnog stila u jednostavni model*

| Polje citatnog stila | Polje jednostavnog modela |
|---|---|
| Autori | jednostavni_autori |
| Godina izdavanja | jednostavni_godina |
| Naslov knjige | jednostavni_naslov |

Primjenom ili matrice uvjeta ili blokova sintakse te mapiranjem polja omogućena je primjena APA citatnog stila na zbirku iz primjera. Primjenom citatnog stila[49] dolazi se do pregleda bibliografije koja se odmah može i izvesti u nekom zadanom formatu:

**Ghani, AAA**

**2013**

Ghani, AAA & Parizi, RM. (2013). Aspect-oriented program testing: an annotated bibliography. *Journal of Software*.

**2015**

[49] U slučaju članaka u časopisu, APA citatni stil nalaže da se u kurzivu piše naslov izvornika, a ne naslov publikacije.

Parizi, RM, Ghani, AAA & Lee, SP. Automated test generation technique for aspectual features in AspectJ. *Information and Software Technology*.

**King, S**

**1999**

King, S. (1999). *Bag of Bones*. Pocket Books.

**Lee, SP**

**2015**

Parizi, RM, Ghani, AAA & Lee, SP. Automated test generation technique for aspectual features in AspectJ. Information and Software Technology.

**Parizi, RM**

**2013**

Ghani, AAA & Parizi, RM. (2013). Aspect-oriented program testing: an annotated bibliography. *Journal of Software*.

**2015**

Parizi, RM, Ghani, AAA & Lee, SP. Automated test generation technique for aspectual features in AspectJ. Information and Software Technology.

#### *5.2.9.2. Izrada citatnih stilova*

Osim primjene postojećih citatnih stilova, nerijetko je za bibliografiju potrebno izraditi posebni citatni stil. Kao što je već rečeno u poglavlju 2.1., bibliografija ne mora koristiti opće poznate citatne stilove, ali bibliografske jedinice moraju biti ispisane na način da su elementi i polja bibliografske jedinice jasni i ujednačeni. Sustav stoga predviđa izradu citatnog stila od strane bibliografa, i to na način da se za svaku vrstu publikacije definiranu u modelu može izraditi posebna sintaksa citiranja. Algoritam izrade citatnog stila prikazan u ovom primjeru relativno je jednostavan, ali moguće je izraditi i kompleksniji. Citatni stil izrađuje se na način da se uzimaju polja modela te dodaju posebni ostali statični dijelovi teksta navoda. Pri primjeni citatnog stila na zbirku, sustav prepoznaje oznake polja u citatnom stilu i zamjenjuje ih vrijednostima polja u modelu za svaku bibliografsku jedinicu. Na primjer, sintaksa novog citatnog stila za vrstu „članak u časopisu“ može izgledati kako je prikazano:

**„{jednostavni_naslov}“ u časopisu „{jednostavni_izvornik}“ / {jednostavni_autori} ({jednostavni_godina})**

Sustav prepoznaje polja po vitičastim zagradama te primjenjuje vrijednosti polja u zbirci na njih, a sve izvan vitičastih zagrada (kao što je kosa crta i oble zagrade) primjenom stila ostaje netaknuto. Međutim, kako polje autori može sadržavati više vrijednosti, potrebno je definirati i graničnike tih vrijednosti. Sustavom je predviđeno definiranje globalnih graničnika i graničnika zadnje vrijednosti.

Tako se u ovom primjeru može definirati da se svako polje koje sadržava više vrijednosti razgraniči sa točka-zarezom i razmakom, a posljednja vrijednost sa „ i “ (razmak-i-razmak). Primjenom novodefiniranog citatnog stila na jedan od zapisa (vrste „članak u časopisu“) dolazi se do sljedeće bibliografske jedinice:

„Automated test generation technique for aspectual features in AspectJ“ u časopisu „Information and Software Technology“ / Ghani, AAA; Parizi, RM i Lee, SP (2015)

Sličan stil moguće je izraditi i za vrstu „knjiga“, gdje je moguće uvrstiti i polje nakladnika koje u vrsti „članak u časopisu“ nije bilo dostupno:

**„{jednostavni_naslov}“ / {jednostavni_autori}. {jednostavni_nakladnik} ({jednostavni_godina})**

Kako vrsta „knjiga“ ne sadrži polje izvornika, ono nije napisano u sintaksi citatnog stila za tu vrstu. Primjenom navedenog citatnog stila za knjigu na jedini zapis te vrste u zbirci, dolazi se do sljedeće bibliografske jedinice:

„Bag of bones“ / King, S. Pocket Books (1999)

#### *5.2.9.3. Generiranje bibliografije*

Sustav će pri primjeni citatnog stila na zbirku prepoznati o kojoj vrsti publikacije je riječ te primijeniti pripadajuću sintaksu. Nakon primjene novoizrađenog citatnog stila na cijelu zbirku, bibliografija izgleda kako slijedi:

„Aspect-oriented program testing: an annotated bibliography“ u časopisu „Journal of Software“ / Ghani, AAA i Parizi, RM (2013)

„Automated test generation technique for aspectual features in AspectJ“ u časopisu „Information and Software Technology“ / Ghani, AAA; Parizi, RM i Lee, SP (2015)

„Bag of bones“ / King, S. Pocket Books (1999)

Kako bi bibliografija bila potpuna, potrebno je primijeniti i prethodne uvjete grupiranja (primjerice po autoru):

**Ghani, AAA**

„Aspect-oriented program testing: an annotated bibliography“ u časopisu „Journal of Software“ / Ghani, AAA i Parizi, RM (2013)

„Automated test generation technique for aspectual features in AspectJ“ u časopisu „Information and Software Technology“ / Ghani, AAA; Parizi, RM i Lee, SP (2015)

**King, S**

„Bag of bones“ / King, S. Pocket Books (1999)

**Lee, SP**

„Automated test generation technique for aspectual features in AspectJ“ u časopisu „Information and Software Technology“ / Ghani, AAA; Parizi, RM i Lee, SP (2015)

**Parizi, RM**

„Aspect-oriented program testing: an annotated bibliography“ u časopisu „Journal of Software“ / Ghani, AAA i Parizi, RM (2013)

„Automated test generation technique for aspectual features in AspectJ“ u časopisu „Information and Software Technology“ / Ghani, AAA; Parizi, RM i Lee, SP (2015)

U slučaju velikog broja bibliografskih jedinica po svakoj grupaciji, bibliografske jedinice poželjno je i numerirati, što je algoritam vezan izričito uz bibliografiju, a ne uz zbirku. Numerirati je moguće globalno, ali i zasebno (lokalno) unutar svake grupacije (numeriranje se resetira sa svakom grupacijom). Primjenom algoritma globalnog numeriranja na bibliografiju u ovom primjeru, rezultat je gotov proizvod koji zadovoljava definiciju bibliografije:

**Ghani, AAA**

1. „Aspect-oriented program testing: an annotated bibliography“ u časopisu „Journal of Software“ / Ghani, AAA i Parizi, RM (2013)
2. „Automated test generation technique for aspectual features in AspectJ“ u časopisu „Information and Software Technology“ / Ghani, AAA; Parizi, RM i Lee, SP (2015)

**King, S**

3. „Bag of bones“ / King, S. Pocket Books (1999)

**Lee, SP**

4. „Automated test generation technique for aspectual features in AspectJ“ u časopisu „Information and Software Technology“ / Ghani, AAA; Parizi, RM i Lee, SP (2015)

**Parizi, RM**

5. „Aspect-oriented program testing: an annotated bibliography“ u časopisu „Journal of Software“ / Ghani, AAA i Parizi, RM (2013)
6. „Automated test generation technique for aspectual features in AspectJ“ u časopisu „Information and Software Technology“ / Ghani, AAA; Parizi, RM i Lee, SP (2015)

Navedenu bibliografiju moguće je izvesti u raznim formatima, a specifični podržani formati ovise o softverskoj implementaciji opisanog sustava.

# 6. Prikaz arhitekture softverskog rješenja

U nastavku slijedi prikaz softverskog prototipa sustava, opisanog u prethodnom poglavlju, koji je implementiran u programskom jeziku Python. Softversko rješenje će zadovoljavati ciljeve već spomenute u poglavlju 4.2:

1. Učitavanje strukturiranih bibliografskih podataka (MARC, BibTex, CSV)
2. Zapisivanje konačne bibliografije u zadani format (HTML, PDF, TXT)
3. Mapiranje metapodataka
4. Standardizaciju i validaciju podataka
5. Grafičko upravljanje
6. Tijek rada u grafičkom sučelju za poluautomatsku izradu bibliografija, što uključuje izvještaje i mogućnost odgovora od strane bibliografa
7. Mogućnost daljnjeg širenja i poboljšavanja softvera

Prva tri navedena cilja jednostavno su ocrtavanje sustava na softversko rješenje, dok su zadnja tri cilja svojevrsno proširenje i dodana vrijednost softverskog rješenja, kako bi isto bilo upotrebljivije i kvalitetnije za korištenje.

U sklopu ovog rada, kao softversko rješenje izrađena je aplikacija, odnosno izvršni računalni program (*executable*) koji implementira sve komponente i algoritme sustava. Aplikacija je napisana u programskom jeziku Python, verziji 3.5.1.[50] Grafičko sučelje aplikacije izrađeno je u PyQt jeziku, verziji 5.[51] Za izradu konačne izvršne datoteke (kako bi se izbjeglo korištenje Python prevoditelja) koristio se PyInstaller, koji sve Python datoteke skuplja u jednu izvršnu (.exe) datoteku.[52]

## 6.1. Python kôd

Python je više-paradigmatski programski jezik opće upotrebe koji dopušta proceduralno, objektno i funkcijsko programiranje, a često se koristi za upravljanje, analizu i rukovanje velikim brojem podataka.[53] Iz tog razloga, Python je odabran kao programski jezik za izradu softverskog rješenja, koje će rukovati potencijalno velikim brojem zapisa i bibliografskih jedinica. Objektno-orijentirani aspekt poželjan je zbog mogućnost izrada klasa, odnosno mogućnosti apstrakcije kôda prema komponentama sustava (model, citatni stil, zbirka, bibliografija). U kontekstu Pythona, svi navedeni objekti u kôdu su predstavljeni kao klase.

[50] Python 3.5.1. URL: https://www.python.org/downloads/release/python-351/ (30.8.2016.)
[51] PyQt5. URL: https://www.riverbankcomputing.com/software/pyqt/download5 (30.8.2016.)
[52] PyInstaller. URL: http://www.pyinstaller.org/ (30.8.2016.)
[53] Why Python for Big Data? | Continuum. URL: https://www.continuum.io/why-python (30.8.2016.)

Python podržava i module, odnosno datoteke koje sadrže konstante, funkcije i klase koje implementiraju proširenja jezika koje se mogu ponovno koristiti među projektima. To znači da kôd u većini slučajeva nije potrebno ponavljati, već se jedan modul može pozivati na drugi. Pomoću modula ostvaruje se jedan od ciljeva softverskog rješenja, mogućnost daljnjeg širenja i poboljšavanja softvera.

Prije prikaza grafičkog sučelja softverskog rješenja, potrebno je objasniti način na koji je softver programiran slijedeći logiku sustava prikazanog u prethodnom poglavlju. Stoga će u ovom poglavlju biti prikazani dijelovi kôda koji prikazuju detalje ove implementacije sustava.

### 6.1.1. Model

Kao što je ranije prikazano, model se sastoji od polja i vrsti publikacija. Oni se u Pythonu iskazuju kao klase, pa tako klasa „polje modela“ izgleda kako je prikazano u ispisu kôda 1.

```
1 class ModelField:
2     def __init__(self, name, label, is_many=False, transform=noTransform,
validate=alwaysValid):
3         self.name = name
4         self.label = label
5         self._transform = transform
6         self._validate = validate
7         self.is_many = is_many
```

*Ispis kôda 1. Klasa ModelField za polja u modelu*

Klasa polja u ovom slučaju naziva se *ModelField*, te se u njoj definiraju svi atributi polja modela. Deklaracija „def __init__“ označava sve postupke koje će Python provesti kada se klasa instancira (izradi stvarno polje modela, primjerice „naslov publikacije“). Bibliograf je stoga dužan pri svakom definiranju polja navesti barem naziv i oznaku polja, kao što je već spomenuto u prethodnom poglavlju. Atribut „is_many“ označava mogućnost polja da primi više vrijednosti, a u slučaju da nije izričito navedeno, kao zadana vrijednost uzima se „False“, tj. mogućnost primanja samo jedne vrijednosti. Atribut „is_many“ je binaran pa ga je moguće definirati samo sa „True“ ili „False“. Način na koji Python (samim time i aplikacija) bilježi više vrijednosti u jedno polje je lista (*list*). Primjerice, u slučaju više autora, oni su u kôdu spremljeni na sljedeći način:

```
authors = ["Parizi, RM", "Ghani, AAA", "Lee, SP"]
```

Potrebno je napomenuti kako činjenica da u ovom slučaju polje sadrži više autora nije važna, već je važno da polje omogućava zapisivanje više vrijednosti. U slučaju da takvo polje sadrži samo jednog autora, on će svejedno biti sadržan u listi, na sljedeći način:

```
authors = ["King, S"]
```

Atributi *transform* i *validate* navedeni u klasi *ModelField* odnose se na funkcije. Kako je bibliograf u mogućnosti izabrati ograničavanje vrste vrijednosti polja samo na brojeve ili na kombinaciju znakova, funkcija *transform* omogućava pokušaj pretvorbe vrste podatka *string* (znakovi) u *integer* (cijeli brojevi) i obrnuto. Tako će funkcija, ukoliko je ciljano polje vrste *integer*, uspješno pretvoriti *string* „2016“, koji iz niza razloga u strukturiranim podacima može biti zabilježen kao običan tekst, u broj 2016. Međutim, u slučaju da vrijednost sadrži i znakove, npr. „2016.“ (s točkom), funkcija *transform* neće pretvoriti vrijednost u broj, već će pitati bibliografa želi li navedenu vrijednost u potpunosti ukloniti ili izmijeniti. S obzirom na mogući velik broj zapisa s kojim bibliograf može baratati prilikom izrade bibliografije, funkcija *transform* nije obvezna pri uvozu zapisa i dodavanju zapisa u zbirku. Umjesto obveznog pokušaja transformacije, bibliografu se nudi mogućnost provesti transformaciju onda kada mu to odgovara, što je u skladu s ciljem sustava da bibliografu omogući dovoljnu slobodu pri izradi bibliografije. Nadalje, kako postoji mogućnost da pri uvozu nekih strukturiranih podataka (npr. više od tisuću zapisa) većina polja bude pogrešne vrste (*string/integer*), prikladnije je bibliografu ostaviti mogućnost skupljanja svih zapisa koji su mu potrebni, a koje će naknadno provjeravati, transformirati i provoditi dodatne funkcije.

Funkcija *validate* za razliku od funkcije *transform* samo provjerava sadrži li određeno polje odgovarajuću vrstu vrijednosti (*string/integer*), bez pokušaja transformiranja. *Validate* stoga vraća vrijednosti *True* ili *False*, koje se mogu koristiti ili kao informacija bibliografu ili za prijedlog primjene dodatnih funkcija. Na primjer, provedbom funkcije *validate* na polju godine izdavanja koja je definirana kao cijeli broj, a kojoj je vrijednost „2016.“ (sa točkom), funkcija će vratiti *False*, što znači da je u tom slučaju bibliografu moguće ponuditi funkciju *transform* ili neku drugu funkciju.

Ukratko, razlika između funkcija *transform* i *validate* je u tome što funkcija *transform* pokušava promijeniti vrijednost polja u zadanu vrstu vrijednosti, dok funkcija *validate* provjerava zadovoljava li vrijednost u određenom polju zadanu vrstu vrijednosti i ostala svojstva polja (dopuštanje unosa više vrijednosti i sl.).

Klasa *ModelField*, međutim, samo opisuje kakve atribute polje mora imati te način na koji se polje ponaša u cjelokupnom modelu. Za izradu specifičnih polja u modelu koje će bibliograf definirati, klasu *ModelField* potrebno je **„instancirati“, što znači stvoriti stvarne instance (objekte) koje poštuju pravila klase.** U slučaju jednostavnog modela iz primjera u

prethodnom poglavlju, potrebno je instancirati pet polja (navedenih u tablici 2.) i držati se njima zadanih svojstava. Polja se instanciraju na način da se pozove klasa te ispišu svi potrebni atributi (*name, label, is_many, transform, validate*). Funkcije *transform* i *validate* dodjeljuju se kao delegati instancama klase *ModelField* sa zadanim vrijednostima *noTransform*, odnosno *alwaysValid*. Bibliograf po potrebi može iskoristiti ili vlastite ili priložene funkcije. U ovom primjeru, funkciji *transform* proslijedit će se vrste podataka *str* i *int* (*string* i *integer*) koje će služiti kao informacija o vrsti vrijednosti polja . Navedenih pet polja u Pythonu se instancira na način prikazan u ispisu kôda 2.

```
1 autori = ModelField("jednostavni_autori", "Autori", True, str,
validate=isString)
2 naslov = ModelField("jednostavni_naslov", "Naslov publikacije", False,
str, validate=isString)
3 izvornik = ModelField("jednostavni_izvornik", "Naslov izvornika", False,
str, validate=isString)
4 godina = ModelField("jednostavni_godina", "Godina izdavanja", False, int,
validate=isInt)
5 nakladnik = ModelField("jednostavni_nakladnik", "Nakladnik", True, str,
validate=isString)
```

*Ispis kôda 2. Instanciranje polja jednsotavnog modela*

Svaki od navedenih vrijednosti atributa instanci polja prati strog redoslijed atributa navedenih u klasi *ModelField*. Tako u slučaju autora „jednostavni_autori“ odgovara atributu *name*, „Autori“ odgovara atributu *label*, „True“ odgovara atributu *is_many*, a „str“ atributu *transform* (funkcija pokušaja transformacije budućih vrijednosti tog polja u vrstu *string*). Atribut *validate* u ovom slučaju sadrži vrijednosti poput „isString“, što se odnosi na priloženu funkciju „isString“ u modulu s dodatnim funkcijama. Navedena funkcija jednostavno služi za provjeru vrste vrijednosti, odnosno provjerava je li vrijednost nekog polja vrste *string*. Time se dokazuje mogućnost širenja i prilagođavanja softvera jer bibliograf može u istom modulu izraditi vlastite, možda i kompliciranije, funkcije za provjeru te pri instanciranju polja navesti te funkcije umjesto funkcija priloženih u aplikaciji.

Klasa *ModelType* odnosi se na vrste publikacija, koje sadrže skupove obveznih i neobveznih polja. Klasa *ModelType* nešto je jednostavnija od klase *ModelField*, ali za softver jednako važna. Atributi klase *ModelType* su naziv, oznaka, obvezna polja, neobvezna polja.

```
1 class ModelType:
2     def __init__(self, name, label, mandatory_atts, optional_atts):
3         self.name = name
4         self.label = label
5         self.mandatory_atts = mandatory_atts
6         self.optional_atts = optional_atts
```

*Ispis kôda 3. Klasa ModelType za vrste publikacija u modelu*

Za razliku od klase *ModelField*, u klasi *ModelType* sve atribute potrebno je navesti pri instanciranju. Atributi *mandatory_atts* i *optional_atts* primaju liste instanciranih polja kao vrijednosti. Iz tog razloga potrebno je prvo instancirati polja, a zatim vrste publikacija. Pri instanciranju klase *ModelType*, za zapisivanje obveznih i neobveznih polja u kôdu se koriste nazivi **instance** (u ovom primjeru „autori“, „naslov“, „izvornik“, itd.), a ne nazivi ili oznake samog polja.

```
1 knjiga = ModelType("vrsta_knjiga", "Knjiga", [naslov], [autori, godina,
nakladnik])
2 clanak = ModelType("vrsta_clanak", "Članak u časopisu", [autori, naslov,
izvornik], [godina, nakladnik])
```

*Ispis kôda 4. Instanciranje vrsti publikacija jednostavnog modela*

U softverskom rješenju navedeni nazivi instanci zapravo se ne koriste, već softver sam prati instance polja i vrste publikacija. Ovaj primjer koristi nazive instanci radi čitljivosti i jednostavnijeg prikaza kôda. Prije prelaska na izradu samog modela, međutim, potrebno je instancirati još dva polja, koja su obvezna bez obzira na vrstu publikacija. To su polja identifikatora i polje same vrste publikacije.

```
1 id = ModelField("id", "ID", False, int)
2 type = ModelField("type", "Vrsta", False, str)
```

*Ispis kôda 5. Instanciranje obveznih polja identifikatora i vrste*

Način na koji će ova dva polja funkcionirati u samom zapisu je sljedeći: identifikator se dodjeljuje automatski pri svakom uvozu ili dodavanju zapisa u zbirku. Identifikator stoga mora biti cijeli broj (*int*), ali bibliografu je dozvoljeno da napiše vlastitu oznaku za polje identifikatora, pa tako polje „id“ može imati oznake „Identifikator“, „ID“, „Identifier“, itd. Polje vrste publikacije jednostavno sadrži naziv vrste publikacije kojoj određeni zapis pripada, pa će tako u zbirci moguća vrijednost polja „Vrsta publikacije“, odnosno „type“, biti „vrsta_clanak“ ili „vrsta_knjiga“. Ova dva polja služe uglavnom za funkcije unutar softvera, kao što su grupiranje, provjera ispunjenih obveznih polja, primjena citatnog stila i sl.

Instanciranjem svih polja i vrsti publikacija, stvoreni su preduvjeti za izradu samog modela. Klasa *DataModel* odnosi se na model podataka bibliografa, kako je navedeno i u sustavu, što je ujedno i najvažniji dio kako sustava, tako i softverskog rješenja. Klasa *DataModel* opisuje kako se ponaša bilo koji korisnički model, pa je potrebno detaljno objasniti dio kôda koji se na nju odnosi.

```
1  class DataModel:
2      def __init__(self, name, data_types):
3          self.name = name
4          self.data_types = data_types
5          self.fields = [id, type]
6          self.types = {}
7          for datatype in self.data_types:
8              self.types.update({datatype.label: datatype})
9              for mandatory in datatype.mand:
10                 if mandatory not in self.fields:
11                     self.fields.append(mandatory)
12             for optional in datatype.opt:
13                 if optional not in self.fields:
14                     self.fields.append(optional)
```

*Ispis kôda 6. Klasa DataModel*

Kao što je slučaj u prethodne dvije klase, klasa *DataModel* sadrži funkciju instanciranja, pri čemu se definira naziv modela i prosljeđuju instance vrsti publikacija u obliku liste. Iz liste instanci vrsta publikacija, klasa prepoznaje sva polja koja su sadržana u atributima, te ujedno stvara rječnik vrsti publikacija. Točan način na koji se kroz kôd klase to provodi je sljedeći:

1. Naziv modela definira bibliograf (3. red, ispis kôda 6.)
2. Vrste publikacija također navodi bibliograf, kao listu, pri instanciranju modela, što aplikacija koristi za izvlačenje polja (4. red, ispis kôda 6.)
3. Stvara se prazna lista (koja će služiti za spremanje instanci polja) te prazan rječnik (za spremanje instanci vrsti publikacija). Listi polja (*fields*) dodaju se polja identifikatora i vrste publikacije koja moraju biti definirana u trenutku kada se model instancira. U suprotnom, model nije potpun. (5. i 6. red, ispis kôda 6.)
4. *For* petlja provodi sljedeće radnje, za svaku vrstu publikacije u listi vrsta (7. do 14. red):
    a. U rječnik *types* dodaje oznaku vrste publikacije i instancu vrste (8. red)
    b. Za svako obvezno polje vrste publikacije dodaje instancu tog polja u listu *fields*, ukoliko to polje već nije u listi (9. do 11. red)
    c. Za svako neobvezno polje vrste publikacije dodaje instancu tog polja u listu *fields*, ukoliko to polje već nije u listi (12. do 14. red)

Instanciranje modela zatim se provodi na jednostavan način kao i prethodno instanciranje polja i vrsti publikacija:

```
jednostavni_model = DataModel("Jednostavni model", [knjiga, clanak])
```

„Jednostavni model“ naziv je instance klase *DataModel* (*name*), dok se lista koja sadrži instance vrsti „knjiga“ i „clanak“ šalje atributu *data_types* klase *DataModel*. Time je

jednostavni model iz primjera prethodnog poglavlja spreman za korištenje. U kôdu je dostupna i funkcija *finalize*, kao dio klase *DataModel*, koja provjerava ispunjava li instancirani model sve uvjete klase *DataModel*, npr. sadrži li polja identifikatora i vrste, sadrži li barem jednu vrstu publikacije, itd.

### 6.1.2. Mapiranje formata u model

Prema primjeru iz prethodnog poglavlja, slijedi uvoz podataka iz formata BibTeX, CSV i MARC. Međutim, kako je kôd koji se bavi mapiranjem, standardiziranjem i uvozom podataka iznimno velik, ovdje će biti prikazani samo najvažniji dijelovi kôda bitni za razumijevanje aplikacije. Tablice mapiranja formata u model podataka koji definira bibliograf u Pythonu se pišu u rječnicima, pa bi tako mapiranje UNIMARC-a u jednostavni model izgledalo kako je prikazano u ispisu kôda 7.

```
1 UNIMARC_field_mapper = {
2     "200$f": "jednostavni_autori",
3     "200$a": "jednostavni_naslov",
4     "210$d": "jednostavni_godina",
5     "210$c": "jednostavni_nakladnik"
6 }
```

*Ispis kôda 7. Definiranje mapiranja UNIMARC-jednostavni model*

Rječnici vrijede i za druge formate, što znači da format koji se mapira mora imati prepoznatljivo polje iz kojeg će *parseri* (raščlanjivači) moći preuzeti vrijednosti polja iz bibliografskih formata u polja zapisa napravljenih prema zadanim modelu.

### 6.1.3. Zbirka

Klasa *Collection* odnosi se na zbirku, a u odnosu na klasu *DataModel* znatno je jednostavnija.

```
1 class Collection:
2     def __init__(self, model: DataModel):
3         if not model.finalized:
4             raise Exception('Model not finalized')
5         self.model = model
6         self._data = {}
7         self._recyclebin = {}
8         self._current_id = 0
9         self._saves = {}
10        self._grouping_index = {}
11        self._filtering_index = []
12        self._selected_items = []
13        self._view_field = „id“
```

*Ispis kôda 8. Klasa Collection*

Zbirka se uvijek instancira **nakon** modela, jer je za izradu zbirke potrebno navesti model koji će zbirka koristiti. Razlog pisanja parametra „model: DataModel“ umjesto samo „model“ u funkciji instanciranja jest uputa korisniku da je tu varijablu poželjno povezati na instancu modela, a ne neke druge klase poput polja i sl. Najvažniji dio klase *Collection* je „self._data = {}“, što pri instanciranju zbirke stvara prazni rječnik u kojeg se **unose zapisi napravljeni prema modelu u upotrebi**. Zbirka se stoga vrlo jednostavno instancira sljedećim kôdom:

```
zbirka = Collection(jednostavni_model)
```

Pravilno dodavanje zapisa u zbirku kôdom jest preko rječnika, na način da se svaki zapis zapiše kao jedan rječnik. Primjer je prikazan u ispisu kôda 9.

```
1 zapis = {
2     "jednostavni_autori": ["Ghani, AAA", "Parizi, RM"],
3     "jednostavni_naslov": "Aspect-oriented program testing: an annotated
bibliography",
4     "jednostavni_izvornik": "Journal of Software",
5     "jednostavni_godina": 2013,
6     "type": "vrsta_clanak"
7 }
```

*Ispis kôda 9. Instanciranje jednog zapisa koji odgovara pravilima jednostavnog modela*

U ovom slučaju, zapis je potpuno spreman i važeći za unos u zbirku: svaki autor je zasebna vrijednost, godina izdavanja je vrste *integer*, a ne *string*, a sva obvezna polja su prisutna. Sam postupak unosa u zbirku provodi funkcija *addItem*, koja je dio klase *Collection*, a prikazana je u ispisu kôda 10.

```
1 def addItem (self, _validate=True, _transform=True, **item):
2     id = item.get(self.model.id.name)
3     if id is None:
4         id = self.getId()
5         item[self.model.id.name] = id
6     else:
7         if id in self._data:
8             raise Exception("Duplicate ID.")
9     if _transform:
10        self.transformItem(item)
11    if _validate:
12        self.validateItem(item)
13    self._data[id] = item
```

*Ispis kôda 10. Funkcija dodavanja zapisa u zbirku*

Funkcija unosa zapisa u zbirku, *addItem*, kao jedini obvezni parametar prima sam zapis (označeno sa „**item“). Pri unosu je moguće i izričito naglasiti je li potrebno navedeni zapis provesti kroz funkcije validacije i transformacije (zadana vrijednost nalaže da se te funkcije provedu). Prethodni zapis, kao što je već rečeno, prolazi „ispit“ obje funkcije s obzirom da

poštuje sva pravila modela. Nadalje, s obzirom da u zapisu nije naveden identifikator, funkcija *addItem* poziva funkciju *getId*, koja je također dio klase *Collection* te sama generira vlastiti identifikator. Funkcija sadrži i mogućnost otkrivanja duplikata identifikatora, kako ne bi došlo do neispravnosti sa zapisima u zbirci.

Pozivom funkcije *addItem* instance zbirke te navođenjem prethodno navedenog zapisa kao jedinog parametra, zapis može ući u zbirku:

```
zbirka.addItem(**zapis)
```

### 6.1.4. Citatni stil

Svaki citatni stil, slijedeći logiku softvera, također je instanca klase. U ovoj implementaciji, klasa citatnog stila izgleda kako je prikazano u ispisu kôda 11.

```
1 class CitationStyle:
2     def __init__(self, name, types, rules, stylizer=simpleStylizer,
many_sep="; ", last_sep=" and ", custom=False):
3         self.name = name
4         self.types = types
5         self.rules = rules
6         self.stylizer = stylizer
7         self.many_sep = many_sep
8         self.last_sep = last_sep
9         self.custom = custom
```

*Ispis kôda 11. Klasa CitationStyle*

U klasi *CitationStyle*, kojom se opisuje svaki citatni stil koji će aplikacija koristiti, navodi se: naziv citatnog stila (*name*), vrste publikacija za citatni stil (*types*, kao lista), sintaksa citatnog stila (*rules*, također kao lista koja odgovara vrstama parametra *types*), funkcija za primjenu citatnog stila na zapise (*stylizer*), graničnik za više vrijednosti (*many_sep*) te graničnik za zadnju vrijednost (*last_sep*). Potrebno je napomenuti kako u slučaju ovog softverskog rješenja navedena klasa najbolje funkcionira u kombinaciji sa zadanom funkcijom primjene citatnog stila na zapise, što ne znači da nije moguće instancirati postojeće citatne stilove poput MLA, APA, itd., već samo da je njihova izrada otežana. Funkcija *simpleStylizer* navedena u parametru *stylizer* identična je onoj navedenoj u poglavlju 5.2.9.2. Za instanciranje klase *CitationStyle* stoga je potrebno navesti samo pravila citiranja, odnosno sintaksu citatnog stila. Kako su u ovom primjeru instancirane dvije vrste publikacije, potrebno je napisati dvije sintakse, kao i u prethodnom poglavlju, za svaku vrstu publikacije. Tako će se dvije sintakse iz prethodnog poglavlja u kôdu definirati na način prikazan u ispisu kôda 12.:

```
1 sintaksa_knjiga = "„{jednostavni_naslov}“ u časopisu
„{jednostavni_izvornik}“ / {jednostavni_autori} ({jednostavni_godina})"
2 sintaksa_clanak = "„{jednostavni_naslov}“ / {jednostavni_autori}.
{jednostavni_nakladnik} ({jednostavni_godina})"
```
*Ispis kôda 12. Instanciranje sintaksi citatnog stila*

Sada je moguće instancirati i citatni stil. Kako citatni stil koristi hrvatski jezik („u časopisu“), kao paramater *last_sep* (graničnik za zadnju vrijednosti) koristit će se „i“ umjesto „and“. Parametri *stylizer*, *many_sep* neće se navesti već će se koristiti zadane vrijednosti klase (*simpleStylizer* i točka-zarez).

```
    stil = CitationStyle("Jednostavni citatni stil", [knjiga, clanak],
            [sintaksa_knjiga, sintaksa_clanak], last_sep=" i ")
```

### 6.1.5. Bibliografija

Instanciranjem citatnog stila stvoreni su preduvjeti za izradu, odnosno generiranje bibliografije. Klasa za bibliografiju najjednostavnija je klasa od navedenih, a sadrži samo praznu listu pri instanciranju, koja služi za spremanje bibliografskih jedinica (3. red u ispisu koda 13.).

```
1 class Bibliography:
2     def __init__(self):
3         self._bibl = []
```
*Ispis kôda 13. Klasa bibliografije*

Funkcija instanciranja bibliografije ne prima nikakve parametre, iz razloga što bibliografija, u teoriji, može sadržavati bilo kakve bibliografske jedinice te se, kako je ranije rečeno, nakon instanciranja može smatrati gotovo samostalnim objektom unutar sustava i softverskog rješenja. Bibliografija se stoga instancira na sljedeći način:

```
nova_bibliografija = Bibliography()
```

U klasi *Bibliography* nalazi se i funkcija za generiranje bibliografije iz postojeće zbirke, prema zadanom citatnom stilu.

```
1 def generateBibliography (self, collection, cit_style):
2     stylized = cit_style.stylizer(collection, cit_style)
3     self._bibl = stylized
```
*Ispis kôda 14. Funkcija generiranja bibliografije iz zbirke prema definiranom citatnom stilu*

Funkcija *generateBibliography* dohvaća zapise iz zbirke, na njima primjenjuje citatni stil kojeg bibliograf odredi te tako stvara bibliografske jedinice, koje unosi u bibliografiju

(točnije u listu *self._bibl* navedenu ranije). U slučaju da postoje podaci o grupiranju zbirke, funkcija citiranja kao zaglavlja koristi parametre grupiranja (npr. u slučaju grupiranja prema autoru, zaglavlje će biti svaki jedinstveni autor u zbirci). U protivnom, funkcija jednostavno izlistava bibliografske jedinice abecednim redoslijedom. Funkcija *generateBibliography* može se provesti nad instanciranom bibliografijom na sljedeći način:

```
nova_bibliografija.generateBibliography(zbirka, stil)
```

Instancu modela nije potrebno navoditi, jer se citatni stil kao instanca već odnosi samo na jedan specifični model.

Funkcija *generateBibliography* nije dio funkcije za instanciranje bibliografije iz razloga što se bibliografu želi ostaviti mogućnost vlastitog unosa proizvoljnih bibliografskih jedinica, objasnidbenih tekstova i sl. Ručni unos proizvoljnih bibliografskih jedinica moguć je s funkcijom *addItem*, koja je također dio klase *Bibliography*, prikazanom u ispisu kôda 15.

```
1 def addItem(self, item, position, toBeginning=True):
2     for key, value in self._bibl.items():
3         if key == position:
4             if toBeginning is True:
5                 value.insert(0, item)
6             else:
7                 value.append(item)
```

*Ispis kôda 15. Funkcija dodavanja bibliografske jedinice u postojeću bibliografiju*

Funkciji se navodi bibliografska jedinica koju bibliograf želi uključiti u bibliografiju i grupacija pod kojom će je navesti, kao i specifična uputa treba li bibliografska jedinica biti na početku ili na kraju grupacije, odnosno prije ili nakon bibliografskih jedinica prisutnih u toj grupaciji. Tako bi, primjerice, u slučaju da bibliograf želi dodati napomenu za određenog autora (ukoliko je bibliografija grupirana po autorima), funkciju proveo na sljedeći način:

```
nova_bibliografija.addItem("Autor piše i pod pseudonimom XY", "Ghani, AAA",
                           True)
```

Time će se napomena o autoru uvrstiti na sam početak grupacije. Ukoliko bibliograf želi dodati stvarne bibliografske jedinice, koje iz nekog razloga neće biti navedene prema određenom citatnom stilu, ali će ipak biti abecedno poredane s ostalim bibliografskim jedinicama, takvu jedinicu može dodati na već naveden način, ali nakon dodavanja ručno provesti funkcije abecednog redanja i (po potrebi) numeriranja.

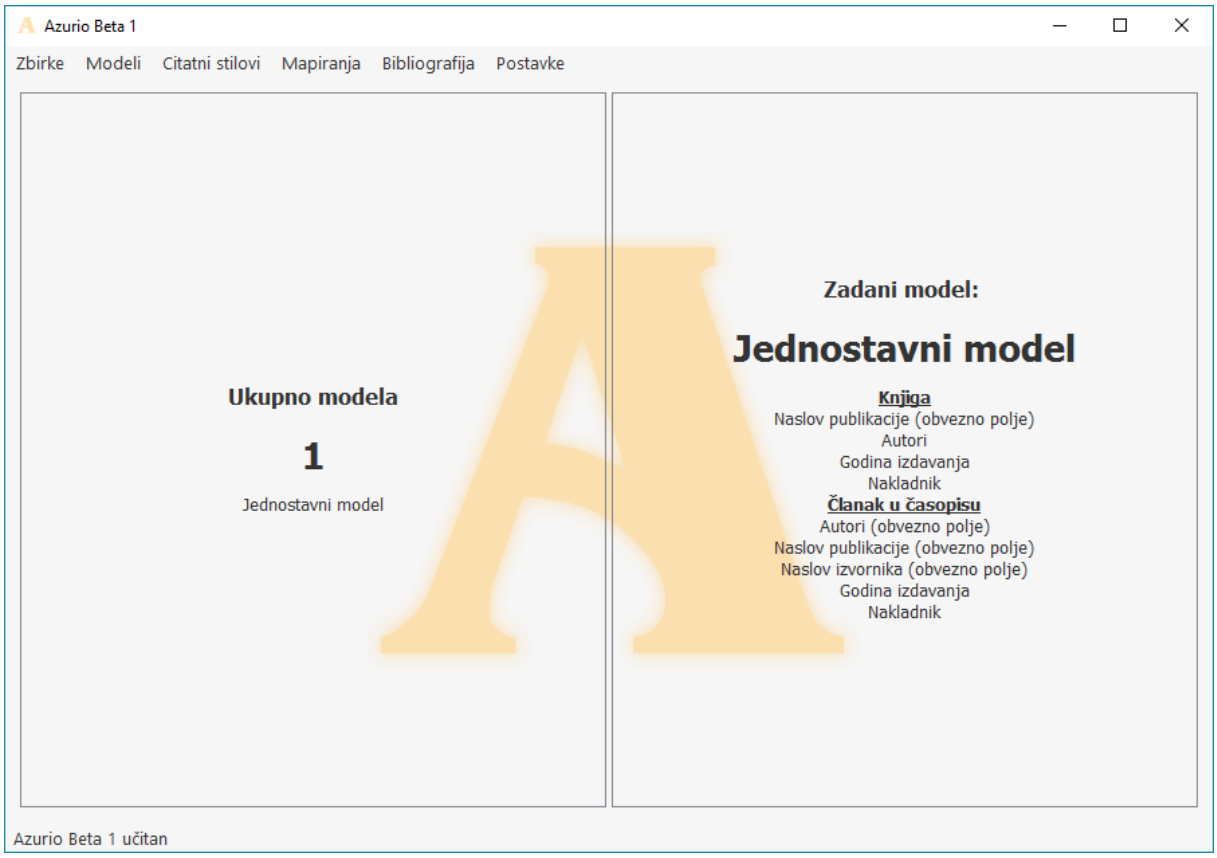
Osim navedenih funkcija, klasa *Bibliography* sadrži i funkcije koje upravljaju izvozom bibliografije u posebnim formatima, kao što su *prepareAsHtml*, *prepareAsTxt*, itd., dok samim zapisivanjem upravljaju funkcije *saveFunction* i *SaveToFile*. Kako je softversko rješenje proširivo, u teoriji je moguće dodati podršku za neograničen broj različitih formata.

## 6.2. Grafičko sučelje softvera

Grafičko sučelje (*Graphical User Interface*) za softversko rješenje napravljeno je u PyQt mapiranju na Qt sustav izrade grafičkih sučelja. U kombinaciji s Python jezikom, softver se tako može jednostavno i brzo prilagoditi za razne operacijske sustave (Windows, Linux, OS X). Za softversko rješenje primijenjena je peta verzija Qt sustava.

Implementacija grafičkog sučelja na postojeći Python kôd izvodi se na način da se sve klase, funkcije, instanciranja, itd. povezuju s dijaloškim okvirima, tipkama i drugim elementima grafičkog sučelja.

Pri pokretanju softvera, pokreće se zaslon dobrodošlice koji prikazuje popis i ukupan broj modela koje je bibliograf izradio te polja i vrste publikacije zadanog modela. Ukoliko bibliograf nije izradio nijedan model, na zaslonu dobrodošlice prikazuju se kratke upute za rad u grafičkom sučelju. Zaslon dobrodošlice stoga služi kao informacija o modelima u softverskom rješenju. Tijekom rada u grafičkom sučelju, bibliografu je uvijek dostupan glavni izbornik koji sadrži izbornike za: zbirke, modele, citatne stilove, mapiranja i bibliografiju. Osim navedenog, dostupne su i postavke, gdje je moguće mijenjati izgled i jezik prikaza grafičkog sučelja. Izbornik „Zbirke“ nudi mogućnosti izrade nove zbirke (prema određenom modelu), trajnog spremanja zbirke te učitavanja zbirke s diska. U izborniku „Modeli“ dostupne su mogućnosti izrade novog modela, izrade novog modela iz postojećeg modela te odabira zadanog modela. Odabir zadanog modela ubrzava rad u grafičkom sučelju ukoliko se bibliograf u većini vremena služi samo jednim od dostupnih modela. Tako se u padajućim izbornicima gdje je moguć odabir modela uvijek prvo nudi zadani model. Izbornik „Citatni stilovi“ pruža mogućnosti izrade novog citatnog stila, uređivanja postojećeg citatnog stila te odabira zadanog citatnog stila. Odabir zadanog citatnog stila slične je namjene kao odabir zadanog modela. Izbornik „Mapiranja“ nudi pristup mapiranjima formata i citatnih stilova u izrađene modele.



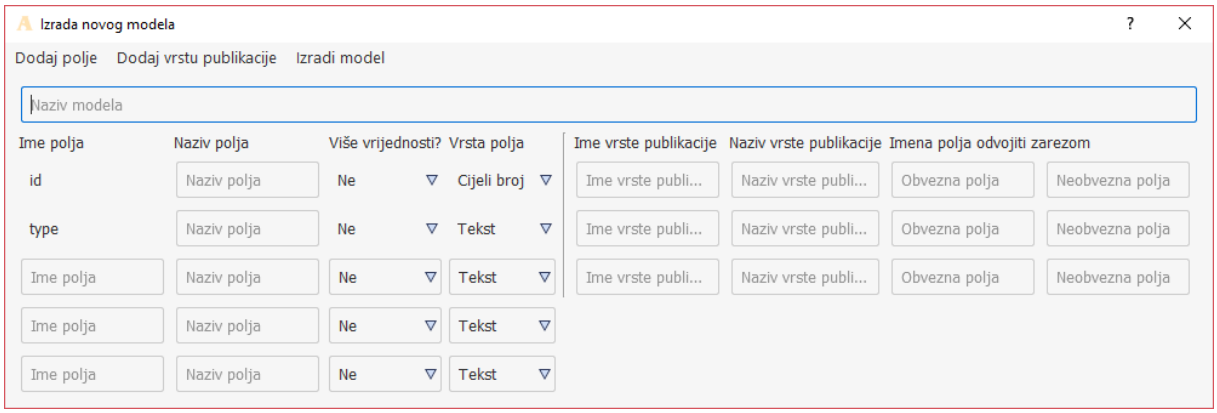

*Slika 7. Zaslon dobrodošlice (s već definiranim modelom)*

Tijek rada u grafičkom sučelju sličan je onom prikazanom u prethodna dva primjera. Jedina razlika u primjeru koji slijedi korištenje je većeg broja zapisa, kako bi se bolje prikazale mogućnosti softverskog rješenja. Stoga će se kao izvori zapisa za sljedeći primjer koristiti tri datoteke s ukupno 90 zapisa: 30 zapisa iz BibTeX datoteke preuzete iz EBSCOHost baze podataka, 30 zapisa iz CSV datoteke preuzete iz Web of Science baze podataka te 30 MARC21 zapisa preuzetih Z39.50 protokolom iz baze podataka Kongresne knjižnice.[54]

### 6.2.1. Izrada modela

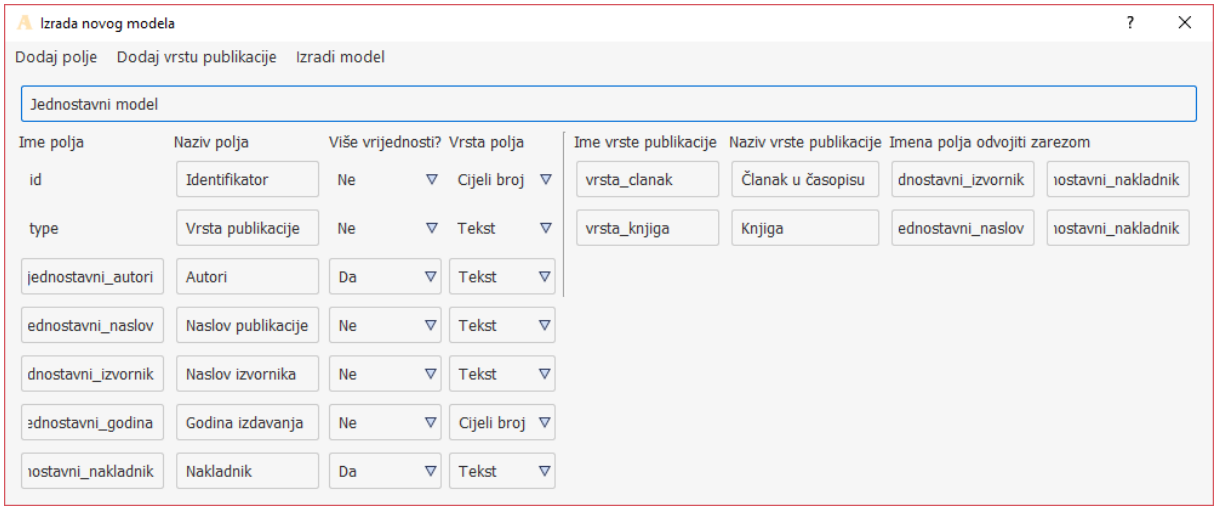
Kao u prethodnim primjerima, prvi korak koji je potrebno napraviti jest izrada modela, što je mogućnost dostupna u izborniku „modeli“. Pritiskom na tipku „Izrada novog modela“ otvara se dijaloški okvir u kojem bibliograf može dodavati i definirati polja modela i vrste publikacija modela.

[54] Za preuzimanje MARC21 zapisa putem Z39.50 protokola koristio se softver MarcEdit 6, dostupan na sljedećoj adresi: http://marcedit.reeset.net/downloads




*Slika 8. Prozor za izradu novog modela*

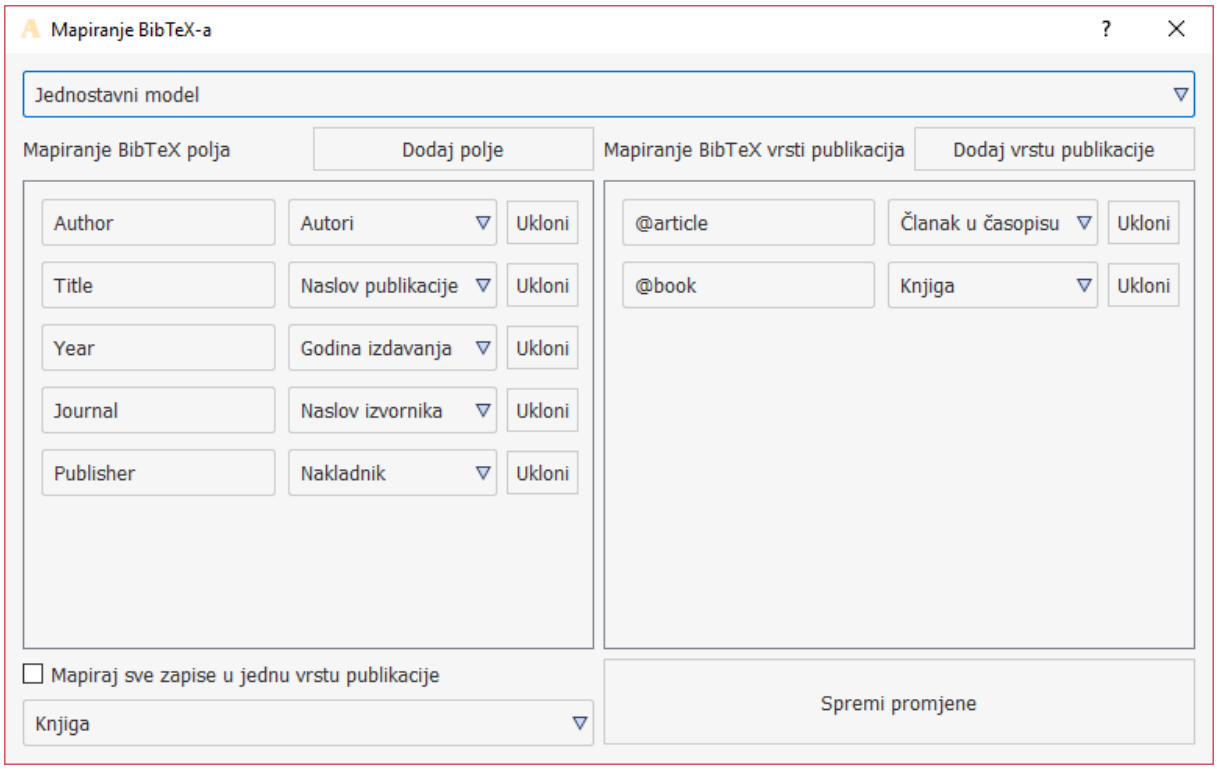
Na slici 8. vidljiva su dva obvezna polja identifikatora i vrste zapisa, kojima je moguće izmijeniti samo naziv. Ostala polja definiraju se upisivanjem imena, naziva, odabirom mogućnosti unosa više vrijednosti te odabirom vrste polja. Pritiskom na tipku „dodaj polje“ u izborniku prozora za izradu modela moguće je dodati još jedno polje (u teoriji je moguće dodati beskonačan broj polja). Vrste publikacije definiraju se upisivanjem imena i naziva vrste te imena obveznih i neobveznih polja odvojenih zarezom u za to predviđenim tekstualnim okvirima. Kao i kod polja, moguće je dodati beskonačan broj vrsti publikacija. Naziv modela upisuje se u tekstualni okvir odmah ispod izbornika. Redoslijed definiranja polja, vrsti publikacija i naziva modela u grafičkom sučelju nije važan, već je jedino važno da podaci definirani u prozoru budu valjani u trenutku kada bibliograf stisne tipku „izradi model“ u izborniku prozora. U tom trenutku, softversko rješenje provodi provjeru valjanosti cijelog modela te ukoliko je rezultat pozitivan stvara (instancira) model.




*Slika 9. Definiranje "jednostavnog modela" u prozoru za izradu novog modela*

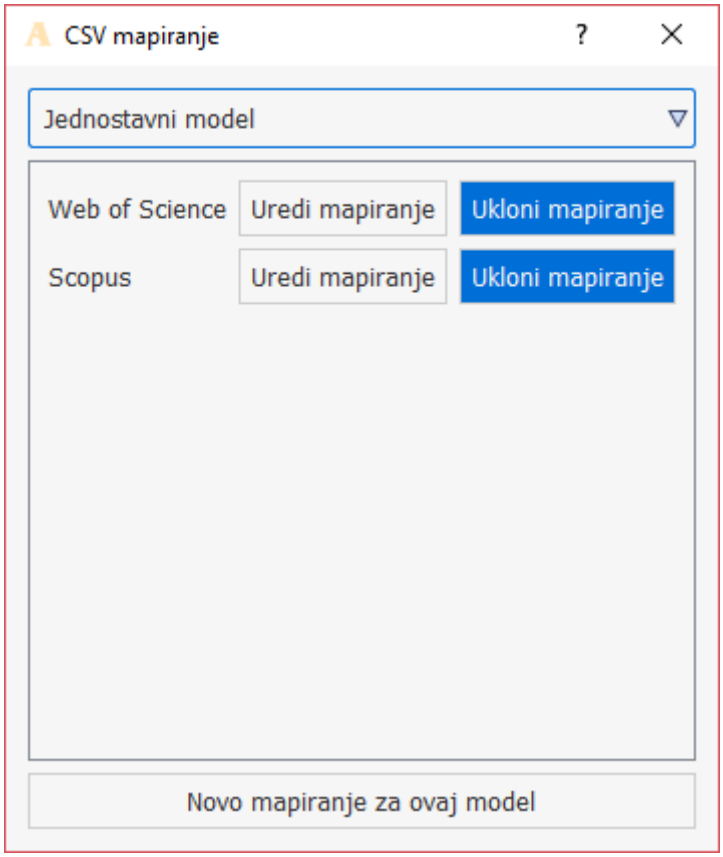
### 6.2.2. Mapiranje formata u model

Sljedeći korak u ovom primjeru definiranje je mapiranja za BibTeX, CSV i MARC21 u „jednostavni model“. Kao što je ranije spomenuto, u izborniku „Mapiranja“, dostupne su opcije za svaki od podržanih formata. Pritiskom na jednog od njih, otvara se posebni dijaloški okvir za mapiranje tog formata u odabrani model (odabirom iz postojećih modela u padajućem izborniku). U tekstualni okvir kod svakog polja upisuje se polje BibTeX-a, a odabir odgovarajućih polja modela obavlja se preko padajućih izbornika. U slučaju da određeno polje iz BibTeX-a bibliograf uopće nema namjeru koristiti, niti je njegov ekvivalent dostupan u modelu kojeg je bibliograf izradio, uklonit će navedeno polje iz mapiranja pritiskom na tipku „ukloni“ dostupno uz svako polje. U slučaju vrsti publikacija, bibliograf sam upisuje koja vrijednost mapiranog polja vrste publikacije BibTeX-a odgovara vrsti publikacije modela. Osim mapiranja svake vrste publikacije zasebno, dostupna je i mogućnost mapiranja svih zapisa tog formata u jednu vrstu publikacije modela. Odabirom korištenja „jednostavnog modela“, mapiranjem svih potrebnih polja i definiranjem mapiranja vrsti publikacija, pritiskom na tipku „spremi promjene“ tablica mapiranja sprema se na disk u obliku JSON datoteke, kako pri svakom pokretanju softvera ne bi bilo potrebno ponovno izraditi isto mapiranje.




*Slika 10. Mapiranje BibTeX-a u jednostavni model*

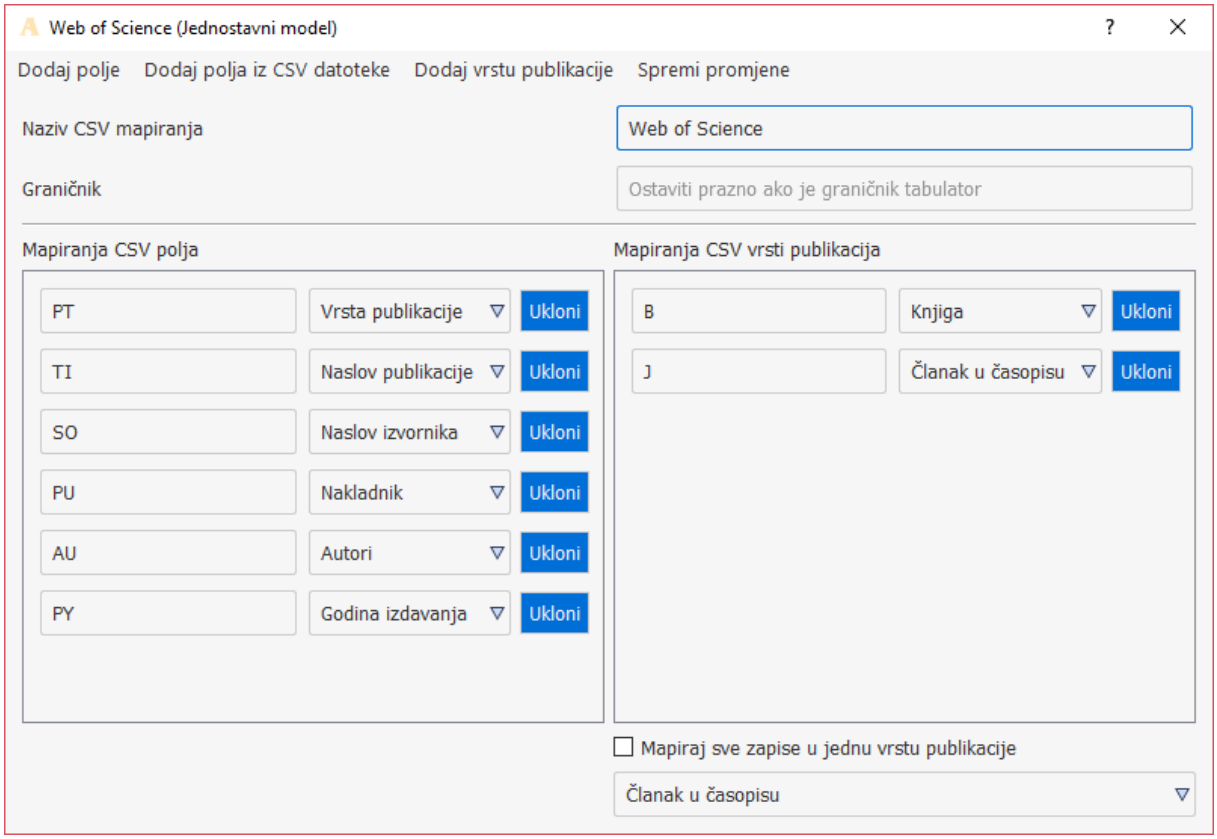
U slučaju CSV formata, svaka CSV datoteka smije imati bilo koje nazive polja, što znači da je bibliografu potrebno ponuditi mogućnost definiranja više mapiranja CSV formata za jedan model. Iz tog razloga, odabirom mogućnosti mapiranja CSV formata u glavnom izborniku, otvara se dijaloški okvir u kojem se iz padajućeg izbornika bira željeni model te prema tom modelu ispisuju dostupna mapiranja. Primjerice, kako različite baze podataka mogu proizvesti različite CSV datoteke, potrebno je definirati zasebna mapiranja. Na slici 11. tako su prikazana dva mapiranja za jednostavni model, a moguće je dodati još mapiranja pritiskom na tipku „novo mapiranje za ovaj model“. Zasebna mapiranja moguće je urediti ili ukloniti.




*Slika 11. Popis dostupnih CSV mapiranja*

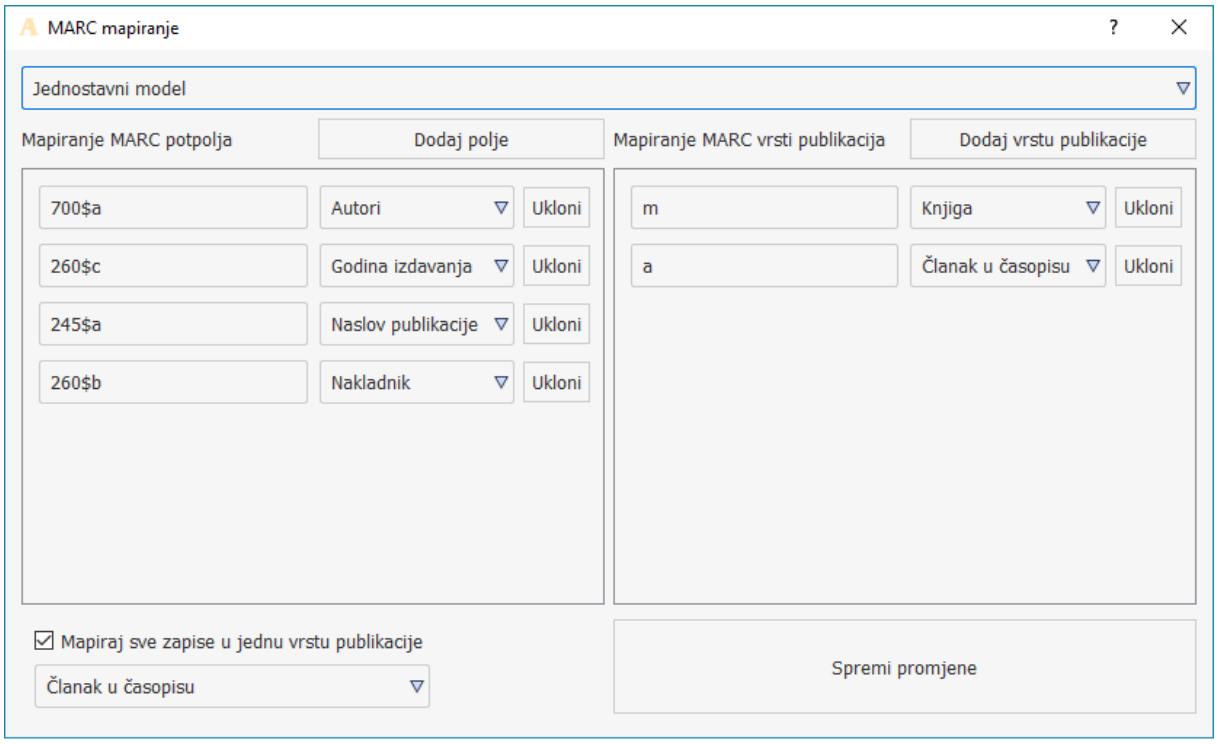
Dodavanjem novog mapiranja otvara se novi dijaloški okvir gdje je moguće definirati naziv mapiranja, CSV graničnik te dodavati i upisivati CSV polja i vrste publikacija i njihova odgovarajuća polja i vrste publikacije odabranog modela. Kao i kod BibTeX-a, moguće je sve zapise mapirati u jednu vrstu publikacije modela ukoliko je to potrebno. Posebnost kod mapiranja CSV formata u ovom slučaju je što je moguće dodati polja iz dostupne CSV datoteke, što je mogućnost prisutna u izborniku prozora za svako mapiranje, kao što je vidljivo na slici 12. Navedena mogućnost prisutna je kako bibliograf ne bi upisivao svako polje ručno. Na slici

12. prikazano je mapiranje CSV formata „Web of Science“ u jednostavni model, koje će se aktivirati uvozom ranije spomenute CSV datoteke s 30 zapisa.




*Slika 12. Mapiranje CSV formata (Web of Science)*

Dijaloški okvir za mapiranje MARC formata sličan je onom za mapiranje BibTeX-a. Bibliograf može dodati neograničen broj potpolja MARC-a i odabrati njihov ekvivalent u vlastitom modelu. Vrste publikacija, kao što je spomenuto u poglavlju 5.2.4., u teoriji je moguće mapirati iz bilo kojeg potpolja, ali u prototipu softvera nudi se mapiranje preko bibliografske razine u oznaci zapisa. Softver nudi jedan prozor za mapiranje MARC-a iz razloga što je preko navedenoga moguće mapirati bilo koje MARC varijante. Primjerice, ukoliko će bibliograf koristiti UNIMARC, naslov publikacije mapirat će preko potpolja 200$a, dok će u MARC-u 21 mapirati potpolje 245$a, a funkcija mapiranja MARC-a pronaći će vrijednost za to potpolje, bez obzira na to o kojoj MARC varijanti je riječ. U ovom primjeru, mapirat će se zapisi iz MARC21 formata, kako je prikazano na slici 13.

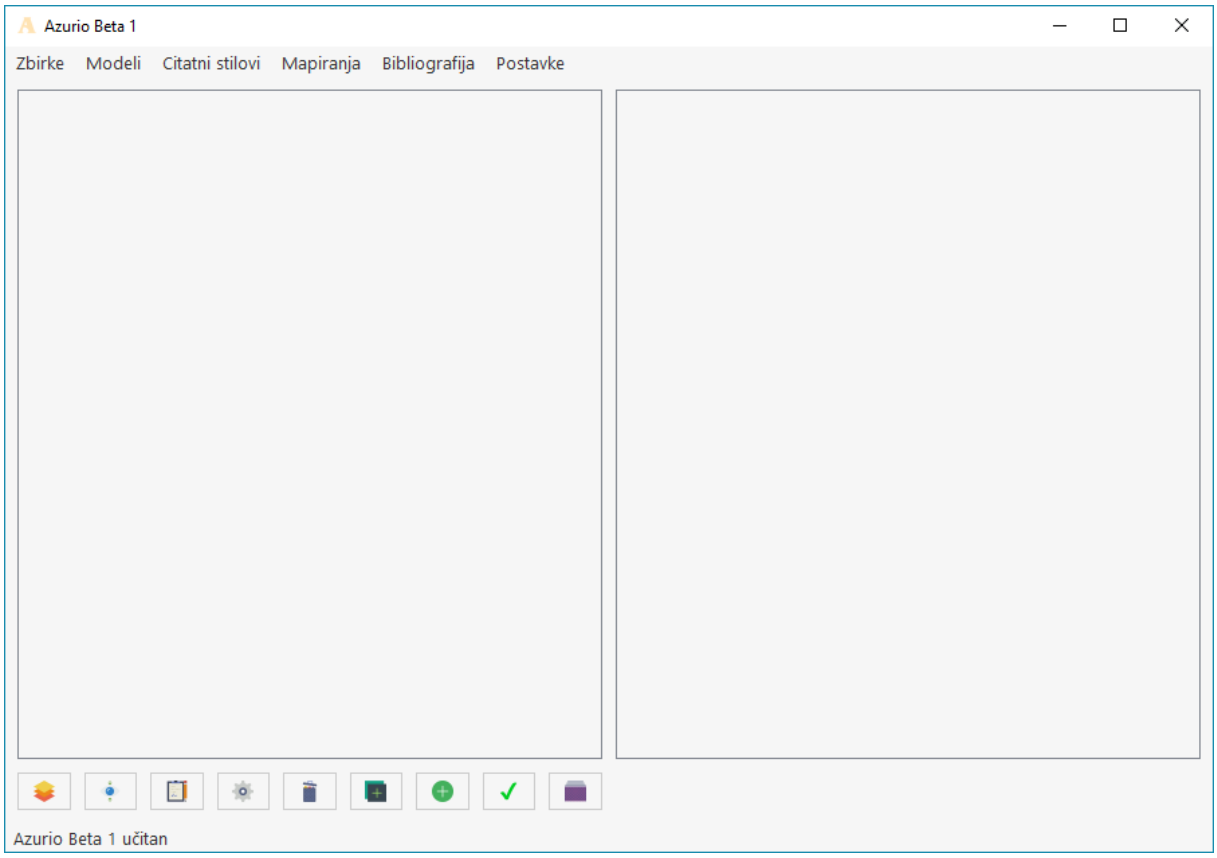



*Slika 13. Mapiranje MARC formata (MARC21 na slici)*

Funkcije mapiranja napisane u Pythonu tako nisu izričito vidljive bibliografu pri radu u grafičkom sučelju, već se pokreću svakim uvozom datoteke pripadajućeg formata. Ipak, to ne znači da softver nije u mogućnosti proširiti podršku na dodatne formate. Dodavanje podrške za dodatne formate moglo bi se implementirati ili običnim nadogradnjama softvera (novim verzijama) ili u obliku *plugin*-ova („proširenja“) koji bi sadržavali više Python funkcija i datoteka okupljenih u jedan Python paket, kojeg bi bibliograf preko grafičkog sučelja učitao kao običnu datoteku i time dodao podršku za format koji mu je potreban.

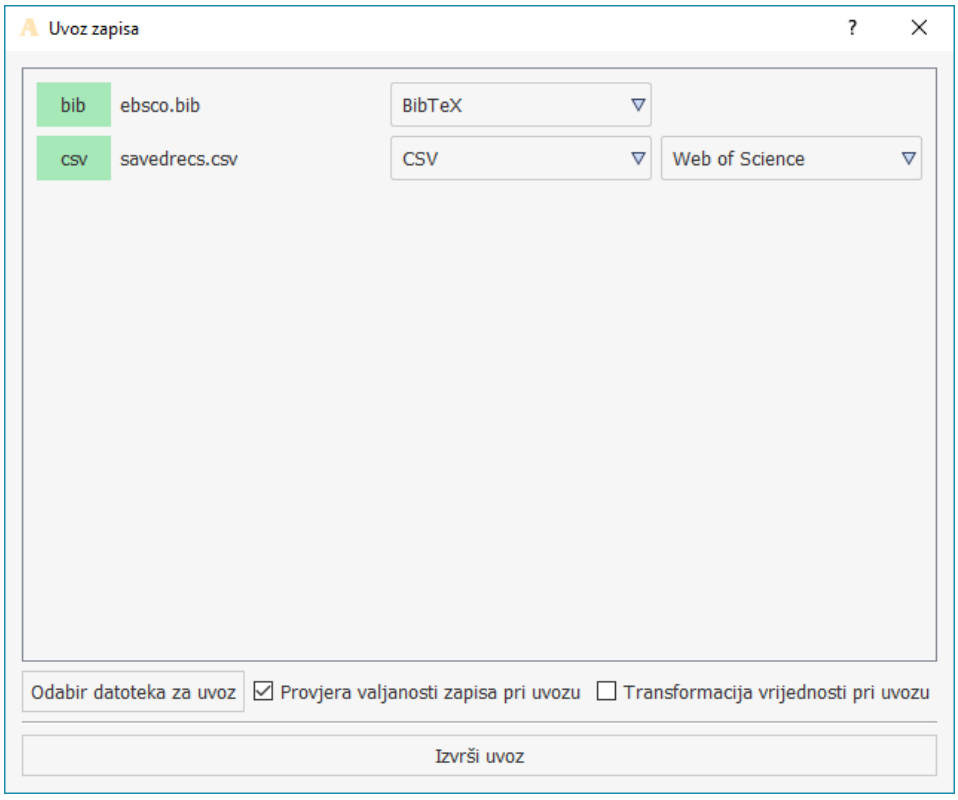
### 6.2.3. Zbirka

Izradom modela i mapiranja za sva tri formata, bibliograf može neometano izraditi zbirku i u nju unositi zapise. Odabir modela iz glavnog izbornika, na kojem će se zbirka temeljiti, otvara se novi zaslon koji zamjenjuje zaslon dobrodošlice. Novi zaslon podijeljen je u dva dijela: s lijeve strane dostupni su potvrdni okviri (*check box*) za svaki zapis u zbirci (ukoliko zbirka ima zapisa), a s desne strane dostupan je detaljan prikaz odabranih zapisa, kao i mogućnosti uređivanja i brisanja svakog zapisa te brisanja svih odabranih zapisa. Ispod popisa zapisa dostupne su tipke za razne funkcije koje utječu na zbirku, poput grupiranja, filtriranja, koša za smeće, dodavanja novog zapisa, uvoza zapisa, itd.




*Slika 14. Glavni prozor za upravljanje zbirkom (prazna instancirana zbirka)*

Zapisi iz datoteka s bibliografskim podacima uvoze se pritiskom na tipku za uvoz zapisa, čime se otvara novi prozor za uvoz zapisa (slika 15.). U prozoru sa uvoz zapisa bibliograf može dodavati datoteke s diska čije zapise želi uvrstiti u zbirku. Dodavanjem datoteke za uvoz, softver prepoznaje ekstenzije podržanih formata (BibTeX, MARC i CSV) te automatski prilagođava padajući izbornik za odabir formata datoteke. U slučaju kada ekstenzija nije prepoznata, tj. nije „podržana“, bibliograf odabirom iz padajućeg izbornika može softveru reći koje mapiranje treba koristiti za svaku datoteku. Prije provedbe samog uvoza datoteka, moguće je i izričito navesti hoće li se nad uvezenim zapisima provesti funkcije provjere valjanosti (*validate*) i transformacije (*transform*).

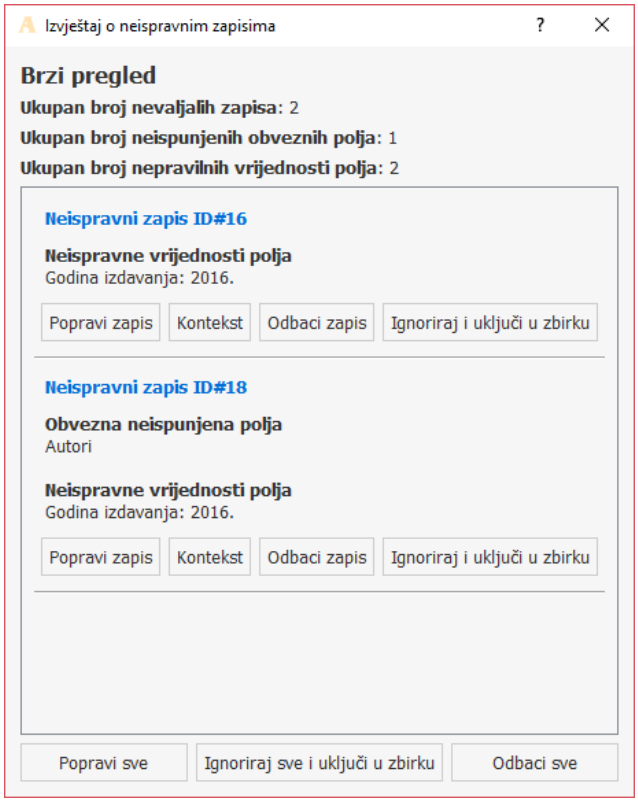



*Slika 15. Uvoz zapisa*

S obzirom da su zapisi sadržani u *loc.mrc* datoteci MARC21 formata, unaprijed se može pretpostaviti kako bi njihov uvoz, s validacijom, vratio niz pogreški, većinom vezanih uz neispravno polje godine izdavanja (koje sadrži ISBD interpunkcije i druge oznake). Stoga će se prvo uvesti BibTeX i CSV datoteke, s validacijom. Uvozom navedenih datoteka u zbirku koja koristi jednostavni model, softver, odnosno funkcija provjere valjanosti, nailazi na dva problematična zapisa: jedan zapis ima neispravno napisanu godinu izdavanja (točka na kraju godine), dok u drugom zapisu nisu navedeni autori (obvezno polje) te također ima neispravno napisanu godinu izdavanja (slika 16.).

Svaki zapis moguće je pokušati popraviti, pritiskom na tipku „popravi zapis“, čime se otvara dijaloški okvir za uređivanje vrijednosti neispravnih i neupisanih obveznih polja. Tipka „kontekst“ otvara dijaloški okvir koji prikazuje cijeli zapis kako bi bibliograf detaljnije mogao vidjeti o kojem je zapisu riječ. Svaki zapis moguće je i potpuno odbaciti, što znači da neće biti uvršten u zbirku, a moguće je i ignorirati upozorenja o neispravnim i neispunjenim obveznim poljima te zapis uključiti u zbirku. Odabirom te mogućnosti bibliografa se upozorava kako zbog neispravnih polja može doći do problema u radu sa zbirkom. Razlog zbog kojeg se korisniku nudi mogućnost ignoriranja upozorenja jest pružanje mogućnosti bibliografu da uveze još zapisa i zatim provjeri valjanost cijele zbirke. U tom slučaju, provjera valjanosti tijekom uvoza

bibliografu služi samo kao informativno sredstvo o broju zapisa sa neispravnim vrijednostima polja i neispunjenim obveznim poljima.



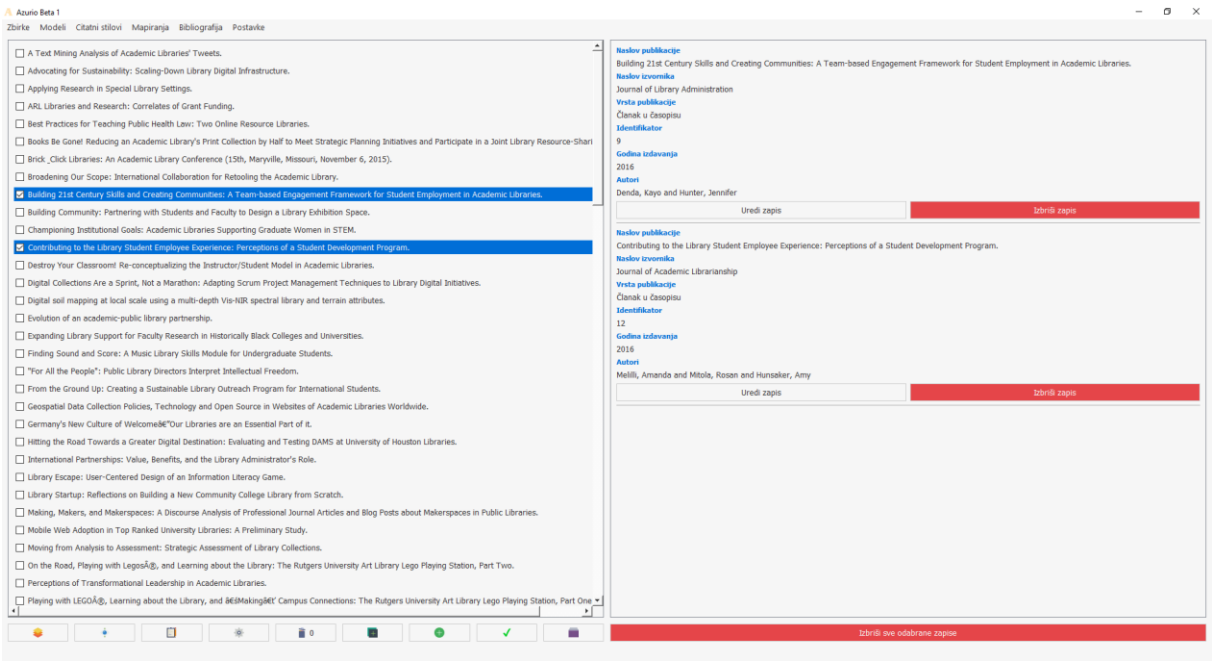

*Slika 16. Izvještaj o neispravnim uvezenim zapisima*

U ovom primjeru, jedan zapis (koji ne sadrži podatke o autorima) je izbrisan, dok je drugi ručno uređen na način da su nedozvoljeni znakovi uklonjeni.

MARC21 zapise u ovom primjeru poželjno je uvesti bez validacije, zatim na njima provesti skupno uređivanje vrijednosti polja godine izdavanja (uklanjanje svih znakova osim brojeva).

Uvezeni zapisi spremaju se u zbirku i prikazuju na lijevom dijelu glavnog prozora, u popisu zapisa. Pri prvom korištenju zbirke, kao zadani glavni prikaz zapisa prikazuje se identifikator, iz razloga što su jedino identifikator i vrsta publikacije obvezna polja, što znači da softver ne može znati koje je polje za bibliografa glavno identifikacijsko sredstvo. U slučaju da bibliograf kao glavni prikaz zapisa želi koristiti neko drugo polje zapisa, ta mogućnost je dostupna u postavkama prikaza zbirke. Kao glavni prikaz zapisa moguće je koristiti bilo koje polje modela. Navedena funkcija ni na koji način ne utječe na samo stanje zbirke, već je dostupna samo kako bi bibliograf lakše upravljao zbirkom.

Zapise u glavnom prozoru moguće je višestruko označavati te na označenim zapisima primjenjivati funkcije poput uređivanja, brisanja i pripreme podataka.

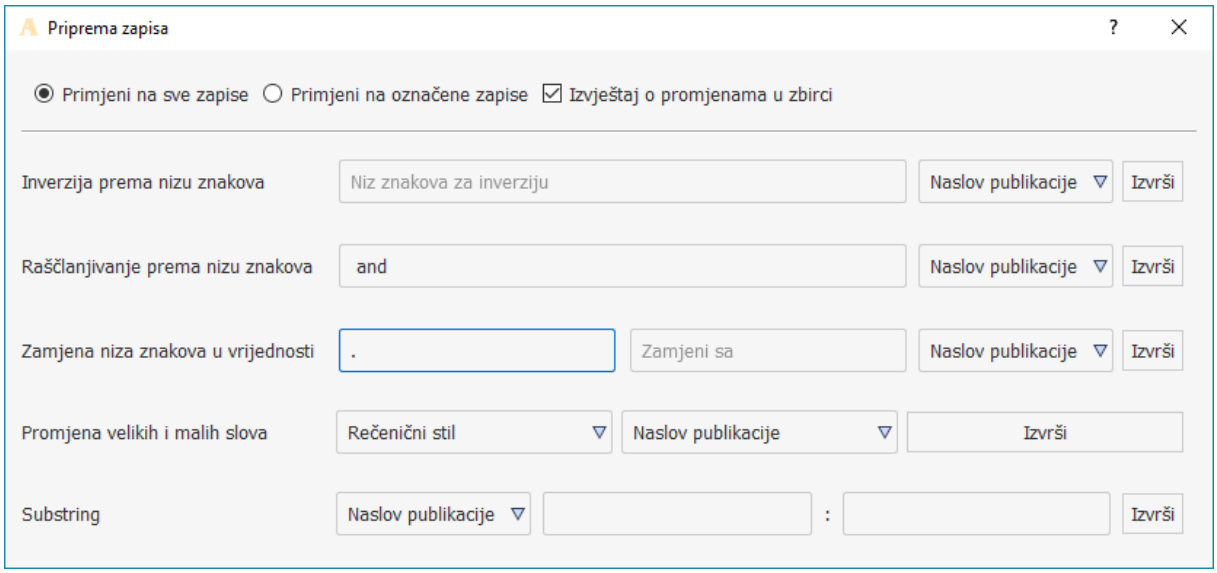



*Slika 17. Glavni prozor za upravljanje zbirkom, s uvezenim zapisima i dvama označenim zapisima*

### 6.2.4. Priprema zapisa u zbirci

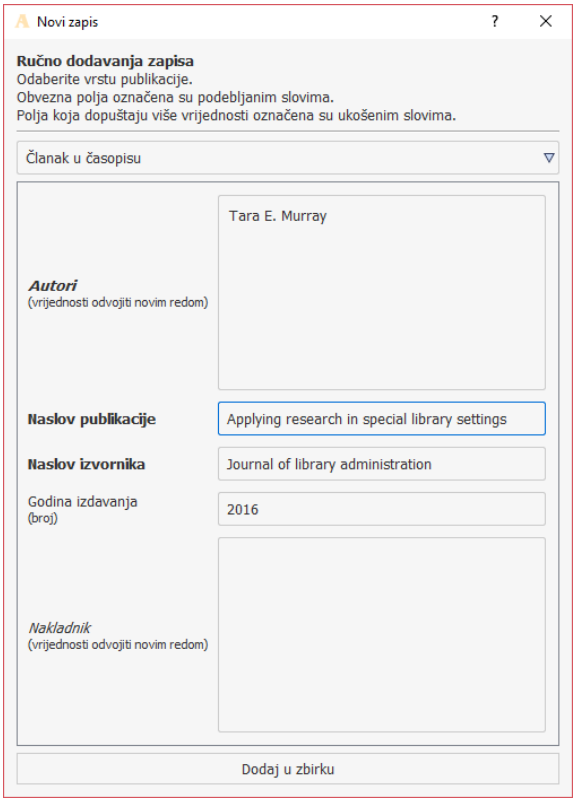
Već sada moguće je izraditi bibliografiju mapiranjem postojećih citatnih stilova i generiranjem bibliografije, no kako bi bibliografija bila kvalitetnija potrebno je provesti dodatne korake. Pregledom zapisa u prikazu zbirke, bibliograf može naići na neujednačenosti u načinu pisanja vrijednosti polja poput autora i naslova izvornika, objašnjene i u prethodnim poglavljima. Takve vrijednosti nije potrebno ručno mijenjati jednu po jednu, već su za to predviđene gotove funkcije, dostupne pritiskom na tipku za pripremu zapisa. Priprema zapisa je zasebni dijaloški okvir u kojem je dostupan niz funkcija koje mogu skupno mijenjati više zapisa odjednom (*batch edit*), prema parametrima koje im zada bibliograf.

U ovom primjeru, autori u svim zapisima bit će odvojeni u zasebne vrijednosti (prema parametrima kao što su *and* i *;* (točka-zarez). Zapisi dobiveni iz MARC21 formata već imaju odvojene autore jer su isti sadržani u ponovljenim poljima 700 (potpolje $a). Za naslove publikacija primijenjena je promjena velikih i malih slova u rečenični stil (Prvo slovo prve riječi veliko), a za naslove izvornika u stil engleskih naslova (Svaka Riječ Velikim Slovom). Također, za zapise dobivene iz MARC21 formata, potrebno je provesti uklanjanje znakova ISBD interpunkcije nad poljima naslova publikacije, nakladnika, itd., što je moguće s funkcijom „zamjena niza znakova u vrijednosti“ (slika 18.)




*Slika 18. Prozor za pripremu i skupno upravljanje zapisima*

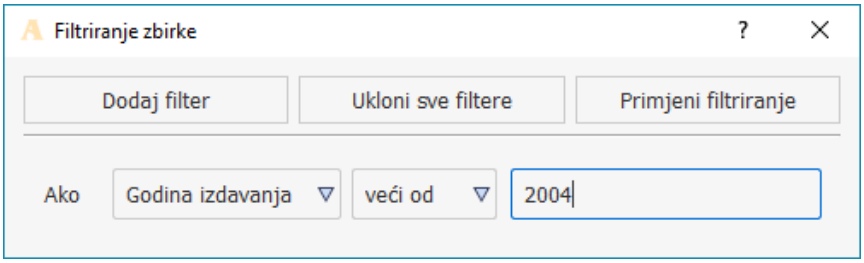
Kao što je spomenuto u prethodnim poglavljima, bibliograf nije ograničen samo na dodavanje zapisa iz bibliografskih formata. Vlastite, ručno unesene zapise, bibliograf može dodati pritiskom na tipku za ručno dodavanje zapisa na glavnom zaslonu. Time se otvara dijaloški okvir (slika 19.) u kojeg bibliograf unosi vrijednosti za polja definirana modelom u upotrebi. Unesene vrijednosti polja u ovom slučaju **moraju** biti valjane te sva obvezna polja moraju biti ispunjena. U protivnom, bibliograf neće moći izvršiti dodavanje zapisa u zbirku.




*Slika 19. Dijaloški okvir za ručno dodavanja zapisa prema jednostavnom modelu*

### 6.2.5. Filtriranje

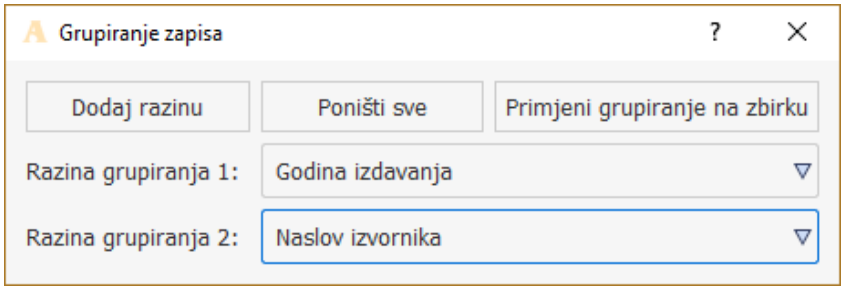
Mogućnost filtriranja u većini slučajeva bibliografu će služiti za sužavanje opsega bibliografije. Primjerice, ukoliko bibliograf u bibliografiji želi imati samo bibliografske jedinice kojima je godina izdavanja od 2005. nadalje, tada će prema tom parametru provesti funkciju filtriranja. Pritiskom na tipku za filtriranje na glavnom zaslonu otvara se dijaloški okvir koji sadrži padajući izbornik sa svim poljima modela u upotrebi, tekstualnim poljem za unos vrijednosti (ukoliko je riječ o brojčanom polju i padajući izbornik za operatore „manji od“, „veći od“ i „jednako“), te tipkom za dodavanje dodatnih uvjeta filtriranja (slika 20.). Za primjenu već spomenutog parametra, odabrat će se „godina izdavanja“, „veći od“ i upisati „2004“ (jer uvjet „veći od 2005“ ne uključuje i 2005. godinu).



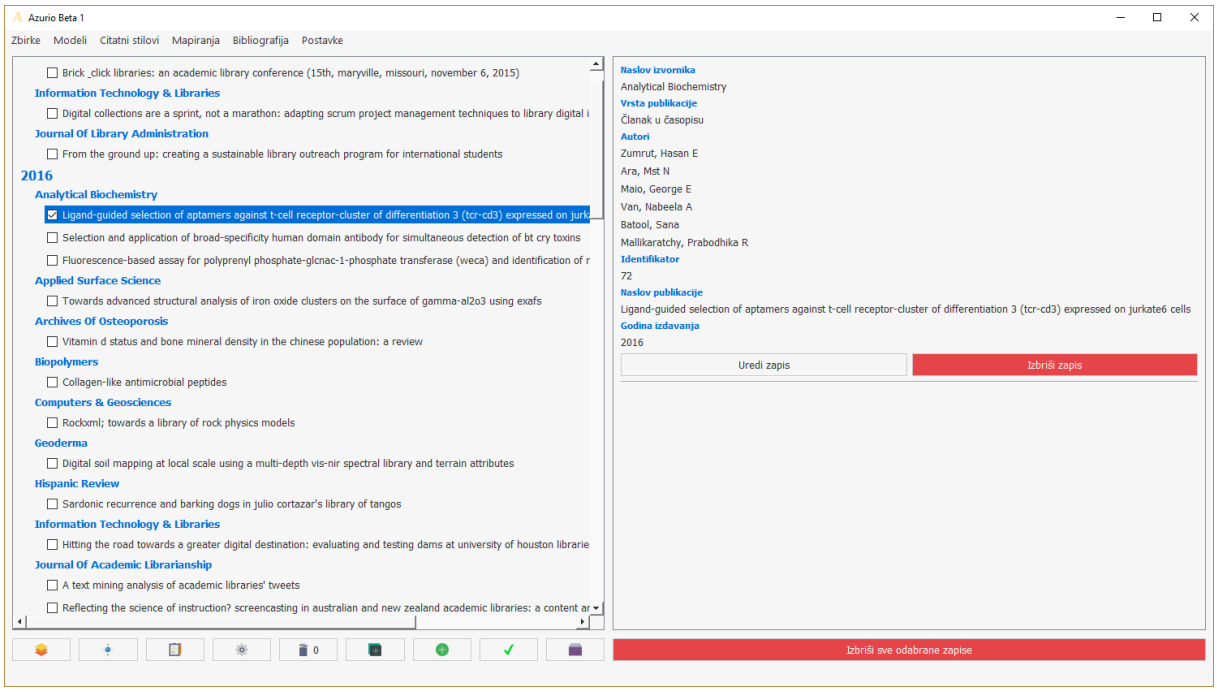

*Slika 20. Dijaloški okvir za filtriranje zbirke*

### 6.2.6. Grupiranje

Grupiranje zapisa u zbirci moguće je pritiskom na tipku za grupiranje na glavnom zaslonu. Dijaloški okvir za grupiranje sadrži samo padajući izbornik za izbor polja prema kojem će se zbirka (samim time i bibliografija) grupirati i mogućnost dodavanja dodatnih razina grupiranja. Zbirku iz ovog primjera tako je moguće, primjerice, grupirati prema godini izdavanja zatim prema naslovu izvornika (slika 21.), čime se mijenja prikaz zbirke na glavnom zaslonu (slika 22.).




*Slika 21. Dijaloški okvir za grupiranje zapisa*

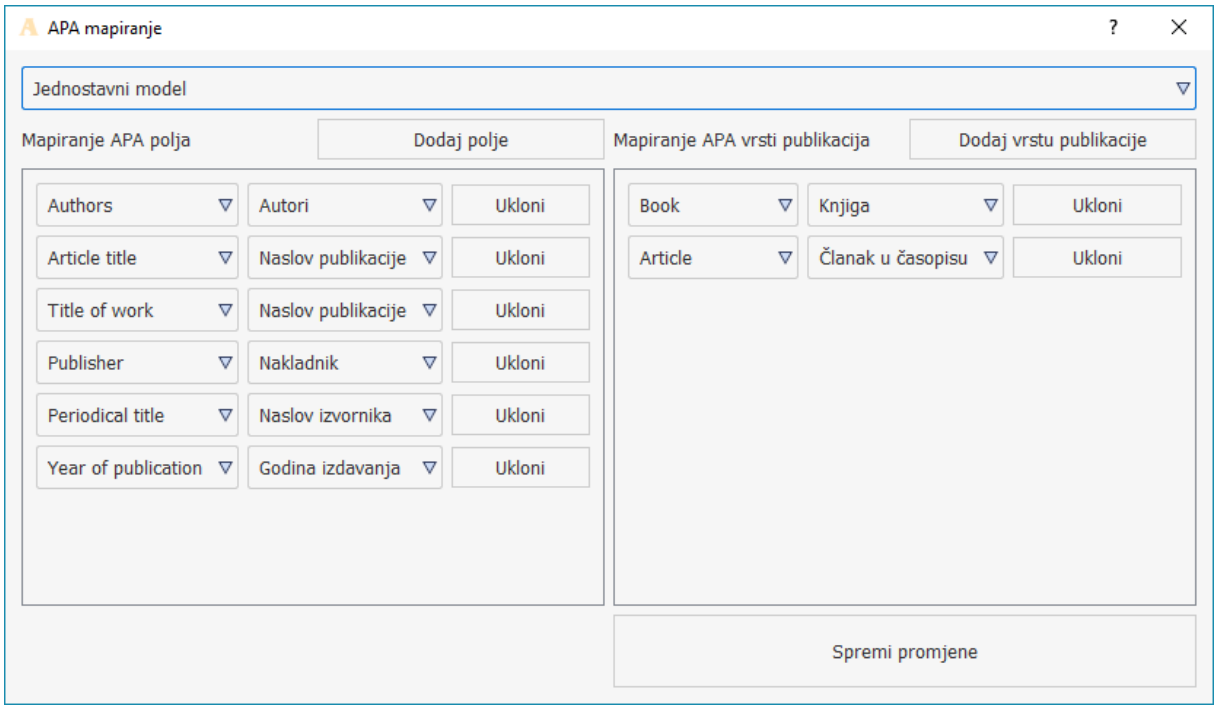



*Slika 22. Rezultat grupiranja u prozoru za upravljanje zbirkom*

Filtriranje i grupiranje ne mijenjaju izvornu zbirku ni na koji način, već se bilježe parametri tih funkcija koje je moguće ukloniti pritiskom na tipke „Poništi sve“ i „Ukloni sve filtere“ u dijaloškim okvirima filtriranja i grupiranja (slika 20. i 21.).

### 6.2.7. Primjena postojećih citatnih stilova

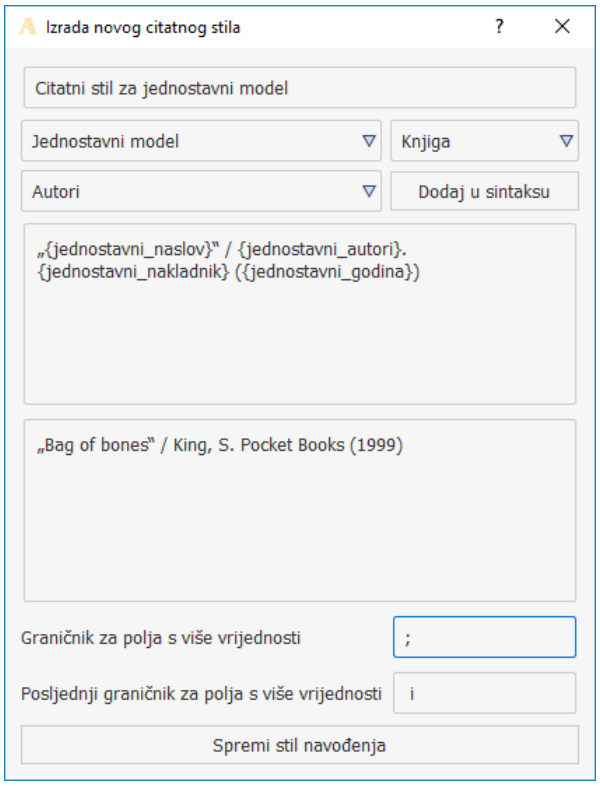
Za primjenu postojećeg citatnog stila (u prototipu softvera dostupan je APA stil), potrebno je mapirati polja modela u polja citatnog stila, što je moguće odabirom citatnog stila iz izbornika „mapiranja“ u glavnom izborniku. Pritiskom na opciju „APA“ pod izbornikom „mapiranje citatnih stilova“ otvara se prozor za mapiranje sličan mapiranju bibliografskih formata (slika 23.). Prozor se sastoji od odabira modela iz padajućeg izbornika (zadan je model koji je trenutno u upotrebi) te od popisa polja citatnog stila s lijeve strane i mogućnosti odabira odgovarajućih polja modela s desne strane. Nakon odabira odgovarajućih polja i vrsta publikacija, pritiskom na tipku „spremi promjene“ citatni stil je spreman za upotrebu s odabranim modelom.




*Slika 23. Mapiranje APA citatnog stila u jednostavni model*

### 6.2.8. Izrada citatnih stilova

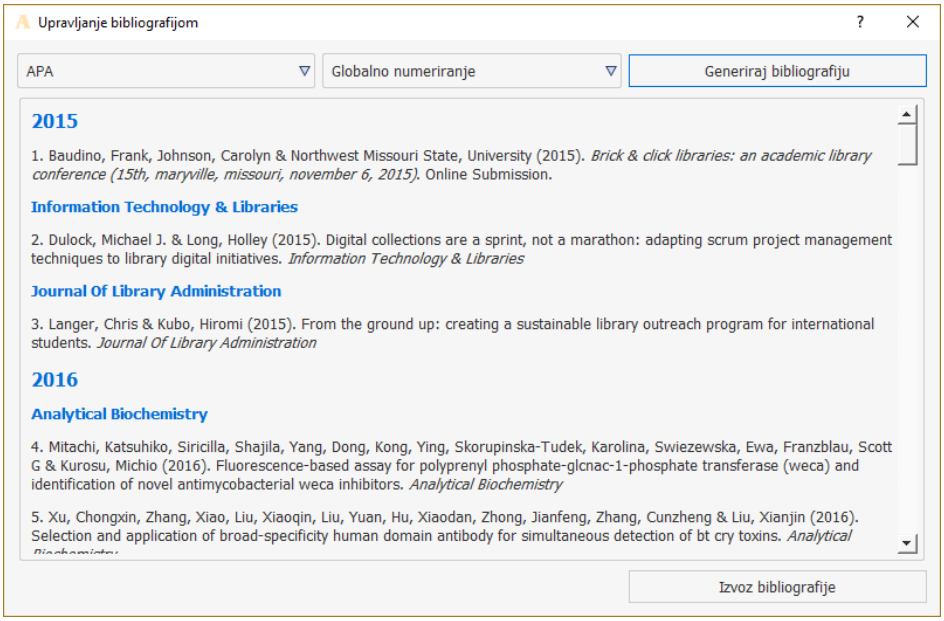
Ukoliko bibliograf želi izraditi potpuno novi citatni stil, tijek rada se razlikuje od mapiranja postojećeg citatnog stila. Pritiskom na naredbu „novi citatni stil“ u izborniku „citatni stilovi“ otvara se prozor za izradu citatnog stila, koji se sastoji od: padajućeg izbornika za odabir modela na kojeg će se citatni stil primijeniti, padajućeg izbornika za odabir vrste publikacije, padajućeg izbornika s poljima tog modela, tipkom za dodavanje polja u citatni stil, tekstualnog okvira sintakse citatnog stila, prikaza primjera bibliografske jedinice prema citatnom stilu, tekstualnog okvira za unos graničnika u slučaju više vrijednosti, tekstualnog okvira za unos posljednjeg graničnika u slučaju više vrijednosti te tekstualnog okvira za unos naziva citatnog stila. Za izradu citatnog stila potrebno je unijeti polja (ili ih ručno napisati) u prikaz sintakse citatnog stila te napisati odgovarajuće graničnike. Za ovaj primjer koristi se citatni stil definiran u poglavlju 5.2.9.2., koji će se nazvati „Citatni stil za jednostavni model“ (slika 24.).




*Slika 24. Dijaloški okvir za izradu novog citatnog stila*

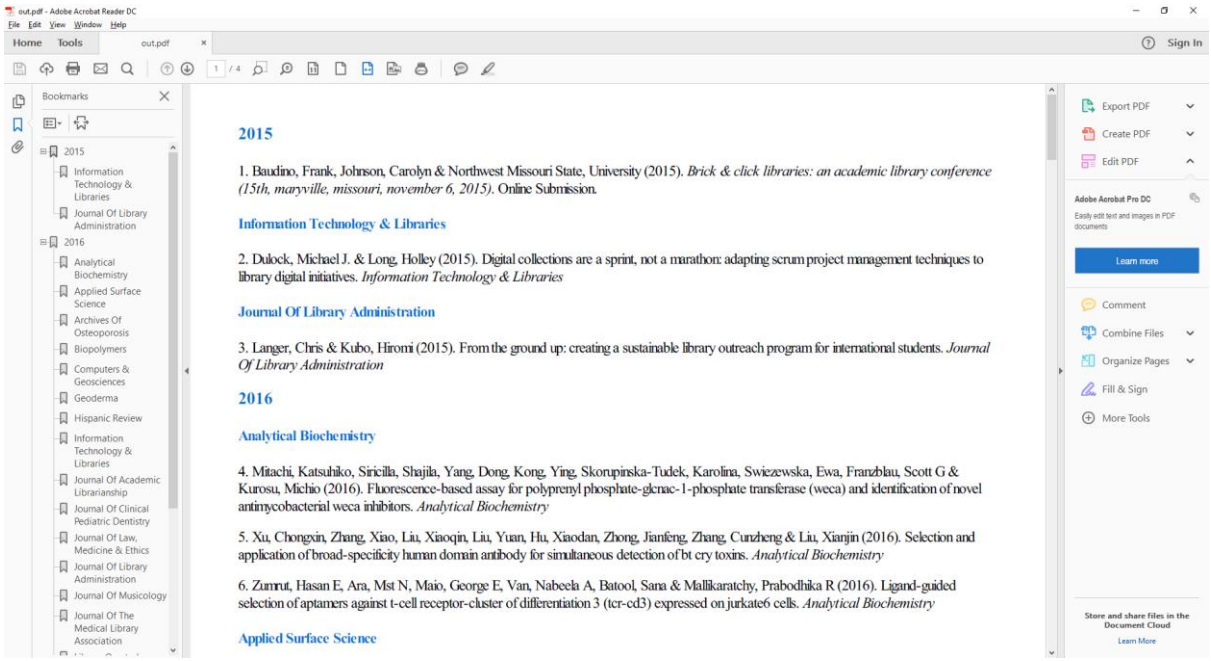
### 6.2.9. Bibliografija

Bibliografiju je sada moguće generirati iz zbirke prema APA citatnom stilu i prema citatnom stilu za jednostavni model, odabirom željenog citatnog stila i pritiskom na tipku „generiraj bibliografiju" u prozoru za upravljanje bibliografijom. Softver tako uzima zbirku, njene podatke o grupiranju i filtriranju te primjenjuje citatni stil za model u upotrebi i generira bibliografiju (slika 25.).




*Slika 25. Prozor za upravljanje bibliografijom*

Bibliografiju je u prototipu softvera moguće u potpunosti slobodno uređivati kao običan tekst (brisati zaglavlja, dodavati vlastite bibl. jedinice i sl.). Pritiskom na tipku „izvoz bibliografije“, otvara se standardni prozor za spremanje datoteka te se nudi mogućnost izbora formata za izvoz. Potrebno je napomenuti kako je, kao i u slučaju bibliografskih formata, u softver moguće dodati podršku za izvoz u nizu drugih formata zahvaljujući modularnosti softvera. Na slici 26. vidljiv je gotov proizvod: bibliografija zapisa grupiranih prema godini izdavanja i naslovu izvornika, spremljena u PDF formatu.




*Slika 26. Bibliografija izvezena u PDF formatu*

# 7. Nedostatci i mogućnosti proširenja sustava

Potencijalni nedostatak sustava je u tome što ne podržava trenutnu izradu bibliografije iz više zbirki s različitim modelima. U trenutnom sustavu, bibliografija se može izraditi isključivo iz jedne zbirke napravljene prema određenom modelu. Međutim, navedeno ograničenje se prema definiciji bibliografije i ne mora smatrati nedostatkom, iz razloga što je jedno od značajki bibliografije usklađenost njenih bibliografskih jedinica, što bi bilo teško izvedivo s više zbirki koje koriste različite modele.

Također, još jednim nedostatkom može se smatrati nemogućnost postavljanja višerazinskih vrijednosti, odnosno potpolja. Najbolji primjer za višerazinsku vrijednost vjerojatno je pripadnost autora instituciji (eng. *affiliation*), gdje je „pripadnost“ potpolje polja „autor“, a u slučaju više autora, svaki autor ima svoju pripadnost. U sustavu, a samim time i softverskom rješenju, podrška za višerazinske vrijednosti je odbačena iz razloga što bi njeno uključivanje značilo značajnu preinaku i **dodatnu kompleksnost** drugih dijelova sustava poput mapiranja i izrade citatnih stilova te same bibliografije. Stoga je jasno kako uključivanje ili odbacivanje podrške za višerazinske vrijednosti ima svoje prednosti i nedostatke. S podrškom za višerazinske vrijednosti, bibliografu je omogućena veća sloboda izrade bibliografije, ali pod cijenu veće kompleksnosti izrade modela, zbirki i citatnih stilova. Bez podrške za višerazinske vrijednosti, bibliograf je ograničen na jednu razinu vrijednosti, ali proces izrade bibliografije postaje jednostavniji i brži.

**Ravnoteža između pružanja slobode bibliografa i jednostavnosti rada** vidljiva je i u samom softverskom rješenju. Na primjer, u softverskom rješenju prikazanom u ovom radu jedina obvezna polja u svakom modelu kojeg bibliograf izradi su identifikator i vrsta publikacije. Time se bibliografu daje velika sloboda izrade modela i mapiranja bibliografskih formata. Međutim, bez dodatnih obveznih polja, mapiranje, ali i izrada citatnih stilova, također se otežavaju. Kada bi softversko rješenje zahtijevalo da obvezna polja svakog modela budu i, primjerice, naslov publikacije, autori, itd., softver bi imao automatske matrice mapiranja za svaki format u navedena polja, čime bi se ubrzao rad u softveru. S druge strane, bibliograf bi u tom slučaju bio zakinut za slobodu izbacivanja navedenih polja ukoliko mu ona nisu potrebna, kao i za slobodu mapiranja.

Zbog naslijeđenih kompleksnosti i različitosti citatnih stilova i strukturiranih bibliografskih formata, bibliografu se u samom softverskom rješenju ne može ponuditi izrada kompliciranijeg citatnog stila (poput MLA ili APA) ili omogućiti vlastita izrada podrške za neki strukturirani format koji već nije podržan. Takve nadogradnje moguće je izvesti samo

primjenom Pythona ili nekog drugog programskog jezika, ovisno o softverskom rješenju. Ipak, valja napomenuti kako je, zahvaljujući modularnosti sustava i samog softvera, dodavanje podrške za nove citatne stilove i bibliografske formate relativno jednostavno i brzo. Na primjer, podršku za Dublin Core moguće je napraviti u roku od jednog do dva dana, uključujući i grafičko sučelje za mapiranje istog u definirani model. Također, kao što je već spomenuto, moguće je izraditi i složeniji modul za izradu novih citatnih stilova, s više uvjeta (primjerice s uvjetom „ukoliko zapis sadrži autore, ne navodi urednike“) čime bi se problem kompleksnosti citatnih stilova mogao riješiti. Međutim, kako takav modul nije cilj ovog rada, u softversko rješenje uključen je jednostavniji način izrade citatnih stilova.

Softversko rješenje temeljeno na sustavu predstavljenom u ovom radu također može biti poboljšano s nizom dodatnih mogućnosti, primjerice integriranjem Z39.50 protokola preko kojeg bi se uvozili MARC zapisi ili učitavanjem i mapiranjem podataka preko ISBN-a, DOI-a i sl. Također, blagom promjenom sustava bilo bi moguće i izvesti zbirku kao dinamični HTML, čime bi se mogla stvoriti dinamična bibliografija za prikaz na mreži, na kojoj bi posjetitelji mogli sami grupirati i filtrirati bibliografiju.

# 8. Zaključak

U radu je dokazana primjena algoritamskog pristupa na izradu bibliografija, pri čemu je prikazan predloženi sustav iz kojeg je, kao dokaz koncepta (*proof of concept*), izvedeno softversko rješenje. Sustav doprinosi praktičnom radu pri izradi bibliografije, ali i teorijskom razumijevanju i razvoju prema značajkama i potrebama digitalnog okruženja.

Iako postoji više definicija bibliografija, u suvremenom okruženju bibliografija se najbolje definira kao „stručno znanstvena djelatnost, koja sabire, vrednuje, odabire, sadržajno analizira i opisuje tiskane ili na drugi način umnožene, javnosti namijenjene tekstove – bibliografske jedinice – pa te opisane klasificira, uređuje i obično u obliku uređenih popisa i publicira s namjerom da pruži informacije o literaturi, a time i pomagalo za stručni rad“[55]. Bibliografije se prema vrsti mogu podijeliti na više načina: prema mjestu nastajanja, prema vremenu nastanka, prema načinu opisa, itd.

Sustav i softversko rješenje predstavljeni u radu odgovaraju na potrebe izrade svih vrsta bibliografija. Izrada bibliografije može se podijeliti na više koraka, dok je sustav predstavljen u ovom radu usmjeren na tri koraka: popisivanje bibliografskih jedinica i njihovo razvrstavanje, klasificiranje gradiva te stvaranje bibliografskog niza kao konačne cjeline. Cilj rada stoga je bio izraditi sustav primjenjiv na sve vrste bibliografija koji omogućava polu-automatizaciju navedena tri koraka.

Sustav se sastoji od pet glavnih komponenti: formata, modela, zbirke, citatnih stilova i bibliografije. U komponenti formata definirani su svi podržani formati te njihova mapiranja za modele. U modelu bibliograf definira vlastita polja potrebna za krajnju bibliografiju te mapira model u bibliografske formate kako bi mogao uvesti zapise. Zbirka označava skup zapisa, uvezenih ili ručno dodanih, na koje se može primjenjivati niz funkcija poput grupiranja, filtriranja, provjere podataka i sl. Citatni stilovi primjenjuju se na zbirku preko modela, a bibliograf može izrađivati i vlastite citatne stilove. Bibliografija se odnosi na konačni proizvod, a generira se iz zbirke primjenjujući određeni citatni stil.

Kako bi se dokazala valjanost sustava, izrađen je softver napravljen prema samom sustavu. Softversko rješenje napisano je u Pythonu, a ciljevi softverskog rješenja su, slijedeći logiku sustava, omogućiti učitavanje strukturiranih bibliografskih podataka, zapisivanje konačne bibliografije u čitljivi format, mapiranje metapodataka, standardizaciju i potvrđivanje podataka, grafičko sučelje, tijek rada za poluautomatsku izradu bibliografije te daljnje širenje i poboljšavanje softvera.

[55] Tadić, Katica. Nav. dj. str. 164

Softversko rješenje uspješno je dokazalo valjanost sustava i njegovu primjenjivost na razne vrste bibliografija. Osim toga, prostor za poboljšanje softvera je otvoren zahvaljujući njegovoj modularnosti pa je tako moguće integrirati razne funkcije za olakšavanje dodavanja zapisa u zbirke kao što su integracija Z39.50 protokola ili učitavanje podataka preko ISBN-a i DOI-a.

Ipak, određeni nedostatci postoje kako u sustavu tako i u softveru iz razloga da se olakšala izrada bibliografije ili da se bibliografu ponudi veća sloboda. Jedan takav nedostatak je nemogućnost postavljanja višerazinskih vrijednosti poput pripadnosti autora instituciji. Kompliciranosti sustava pridodaje raznolikost strukturiranih formata i kompleksnost postojećih citatnih stilova. Međutim, navedene probleme moguće je riješiti određenim promjenama u sustavu, što se može smatrati potencijalom za daljnja istraživanja.

# Algorithmic approach to contemporary bibliography generation

## Abstract

Aim of the master thesis is to offer a versatile and accessible system for semi-automatic bibliography generation using algorithmic approach. Bibliography is one of the oldest activities of library science that is encountering new challenges in today's digital age as a result of a multitude of available bibliographic information in various structured bibliographic records and database web interfaces. Bibliographic data can be represented in many formats, which in turn creates problems during the process of creating a bibliography, making it difficult to approach bibliography creation in a fast and accessible way. In addition, structured digital data are meant primarily for machine reading which presupposes algorithmic handling, thus producing another motive for the master thesis. The proposed system for semi-automatic bibliography generation includes features for: reading various formats of bibliographic metadata, including specialized (MARC, BibTeX, RIS, etc.) and general (XML, JSON, etc.) formats; filtering of records according to user-specified parameters (e.g. date of publication, publisher, author, etc.); grouping of records according to user-specified parameters; defining a citation style and exporting the bibliography in formats such as HTML, TXT, PDF or DOCX. The thesis also discusses relevant bibliography topics and reviews the importance as well as advantages and disadvantages of algorithmic approach to bibliography generation. The master thesis represents an improvement in contemporary activity on bibliography with an emphasis on use and adaptation of new technologies as well as a possibility of creating versatile software tools tailored to local needs and tradition.

**Keywords**: bibliography, algorithmic approach, bibliography heuristics, software, bibliographic metadata

# Popis literature


1. 300 - Physical Description (R). URL: https://www.loc.gov/marc/bibliographic/bd300.html (29.8.2016.)
2. Algoritam. Hrvatska enciklopedija. URL: http://www.enciklopedija.hr/Natuknica.aspx?ID=1718 (29.8.2016.)
3. Aparac-Gazivoda, Tatjana. Teorijske osnove knjižnične znanosti. Zagreb : Filozofski fakultet Sveučilišta u Zagrebu; Zavod za informacijske studije Odsjeka za informacijske znanosti, 1993.
4. Bates, Marcia J. Rigorous Systematic Bibliography. // Interpretations of Reference and Bibliographic Work / White, Howard D; Bates, Marcia J; Wilson, Patrick. Norwood, New Jersey: Ablex Publishing, [s.a].
5. BibTeX.URL: http://www.bibtex.org/About/ (30.8.2016.)
6. Borgman, Christine L. Od Gutenbergova izuma do globalnoga informacijskog povezivanja: pristup informaciji u umreženom svijetu. Lokve; Zadar : Naklada Benja; Gradska knjižnica Zadar, 2002.
7. EndNote product details. URL: http://endnote.com/product-details (12.9.2016.)
8. Export Bibliographic and Holdings Data (MARC Export). URL: http://manual.koha-community.org/3.2/en/exportbibs.html (29.8.2016.)
9. ISBD punctuation in the MARC 21 Bibliographic Format. URL: https://www.loc.gov/marc/marbi/2010/2010-dp01.html (29.8.2016.)
10. Katz, William A. Introduction to reference work. Boston [etc.] : McGraw Hill, 2002.
11. Mapping. URL: http://www.oxforddictionaries.com/definition/english/mapping (29.8.2016.)
12. MARC Standards. URL: https://www.loc.gov/marc/ (30.8.2016.)
13. Pehar, Franjo. Od statističke bibliografije do bibliometrije. Povijest razvoja kvantitativnog pristupa istraživanju pisane riječi. // Libellarium 3, 1(2010), str. 1-28.
14. Rubin, Richard E. Foundations of Library and Information Science. [s.l.] : Facet publishing, 2016.
15. Sarić, Ivana; Magdić, Antonio; Essert, Mario. ZOTERO – program otvorenog kôda za upravljanje bibliografskim bilješkama. // Vjesnik bibliotekara Hrvatske 54, 1/2(2011). str. 158-171. URL: http://hrcak.srce.hr/80472 (29.8.2016.)
16. Sečić, Dora. Informacijska služba u knjižnici. Lokve: Benja, 2006.

17. Soergel, Dagobert. Organizing information: Principles of Data Base and Retreival Systems. San Diego [etc.] : Academic Press, 1985.
18. Tadić, Katica. O bibliografiji s osvrtom na bibliografsku heuristiku. // Izazovi pisane baštine: zbornik radova u povodu 75. obljetnice života Aleksandra Stipčevića. Osijek: Filozofski fakultet, 2005.
19. Understanding MARC Bibliographic. URL: http://www.loc.gov/marc/umb/um01to06.html (30.8.2016.)
20. UNIMARC formats and related documentation. URL: http://www.ifla.org/publications/unimarc-formats-and-related-documentation (30.8.2016.)
21. UNIMARC: bibliografski format. Zagreb: Hrvatsko knjižničarsko društvo; Zadar, Sveučilište u Zadru, 2009.
22. Uvjeti za funkcionalnost bibliografskih zapisa : završni izvještaj / IFLA-ina Studijska skupina za uvjete za funkcionalnost bibliografskih zapisa. Zagreb : Hrvatsko knjižničarsko društvo, 2004.
23. White, Howard D. Publication and Bibliographic Statements. // Interpretations of Reference and Bibliographic Work / White, Howard D; Bates, Marcia J; Wilson, Patrick. Norwood, New Jersey: Ablex Publishing, [s.a].
24. Wilson, Patrick. Pragmatic Bibliography. // Interpretations of Reference and Bibliographic Work / White, Howard D; Bates, Marcia J; Wilson, Patrick. Norwood, New Jersey: Ablex Publishing, [s.a].
25. Zotero. URL: https://www.zotero.org/download/ (29.8.2016.)