SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

MREŽNI PROGRAM ZA TVORBU I OBRADBU TEHNIČKIH RJEČNIKA

Juraj Benić, Jakov Topić

Zagreb, 2015.

Ovaj rad izrađen je na Fakultetu strojarstva i brodogradnje, na Zavodu za strojarsku automatiku i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2014./2015.

Mentor:
Prof.dr.sc.Mario Essert

Studenti:
Juraj Benić, Jakov Topić

Zagreb, 2015.

Sadržaj

Popis slika

Popis tablica

1 UVOD

Rječnik je leksikografsko djelo koje sadrži popis riječi i izraza nekog jezika (ili više jezika) izabranih, raspoređenih i objašnjenih po nekom načelu, te poredanih abecednim ili tematskim redoslijedom.
Elektronički rječnik je rječnik u digitalnom obliku, a ako je uz to dohvatljiv i preko mreže (inerneta), zovemo ga **on-line rječnik**. Postoje različite vrste rječnika s obzirom na sadržaj (opći, terminološki, višejezični i sl.), ustrojbeno načelo (abecedarij, čestotni rječnici, inverzni/obrtimični, konceptualni i sl.) i tehničku izvedbu (lokalni, višekorisnički, integracijski). Najjednostavniji je onaj u kojem pripadno programsko rješenje samo pretražuje bazu i na temelju unešene riječi daje njenu definiciju i eventualno, popratni sadržaj.

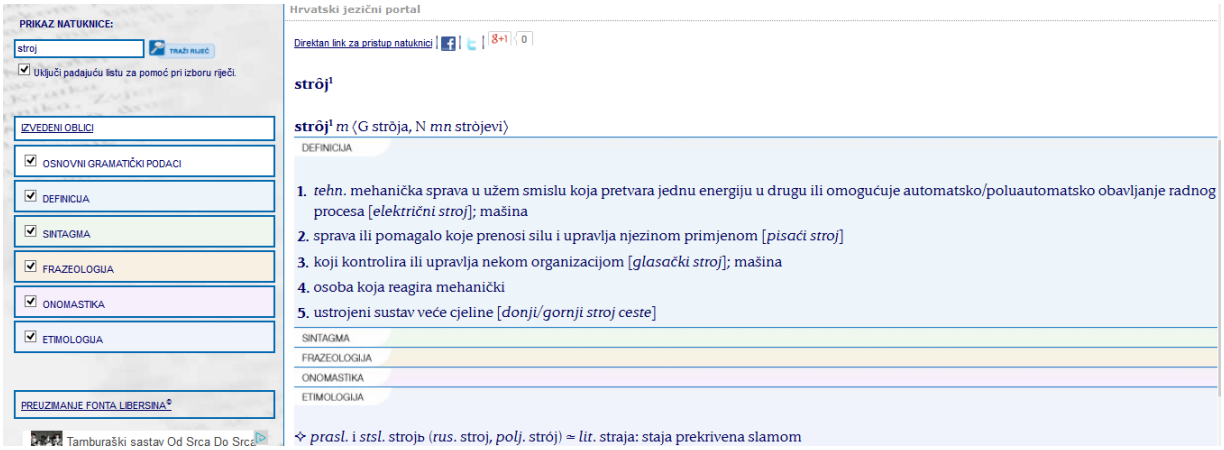
U osnovi se razlikuju *jezični rječnici* i *predmetni rječnici*, kao što su *enciklopedija* i *leksikon*. Jezični rječnik se prvotno bavi jezikom, odnosno leksičkim jedinicama jezika i svim njihovim jezičnim osobinama. Jezični rječnik nerijetko nastoji, barem implicitno, odgovoriti na potrebu utvrđivanja leksičke norme danoga jezika. Tu istu zadaću imaju i *strukovni* ili *terminološki rječnici*.

Rječnička makrostruktura sastoji se od popisa natuknica ili lema u rječniku (abecedariju). Rječnička mikrostruktura sastoji se od natukničkog članka i natuknice ili leme. Natukнички članak često sadržava gramatičke podatke o riječi, stilske odrednice ili oznake, podatke o etimologiji ili porijeklu riječi, definiciju, frazeološke, kolokacijske, te sintagmatske konstrukcije.

Iako su se donedavno rječnici izdavali najčešće u obliku tiskanih knjiga, a zatim prelaze na CD-izdanja, u ovom radu bavit ćemo se isključivo *on-line rječnicima*, tj. nakon pregleda postojećeg stanja u Hrvatskoj ponuditi jedno svoje on-line rješenje koje u zamisli i funkcionalnosti nadilazi postojeća.

1.1 Postojeća rješenje u Hrvatskoj

Najpoznatiji *jezični rječnik* u Hrvatskoj je **Hrvatski jezični portal** (slika 1) i može se dohvatiti na adresi: `http://hjp.novi-liber.hr/`



Slika 1: Hrvatski jezični portal

Hrvatski jezični portal (HJP) rezultat je zajedničkog projekta *Novog Libera* i

Srca. Projekte i poslovanje Novog Libera i Hrvatskog jezičnog portala od 2015. godine preuzela je nakladnička kuća *Znanje*, jedna od najstarijih izdavačkih kuća u Hrvatskoj s dugogodišnjom tradicijom u nakladničkoj i tiskarskoj djelatnosti.

Rječnička baza sastoji se od 116.516 osnovnih riječi (natuknica), od toga 67.049 imenica, 15.699 glagola, 20.154 pridjeva, 7.017 priloga, 111 prijedloga, 72 veznika, 152 broja, 102 zamjenice, 98 čestica i 302 uzvika. Uz osnovne riječi i njihove definicije bazu sačinjavaju i primjeri (oko 60.000), sintagmatski izrazi (oko 18.000) i frazeološki izrazi (oko 10.000). Porijeklo riječi objašnjeno je u zoni etimologije, a porijeklo osobnih imena i prezimena (antroponimi) te geografskih imena (toponima) objašnjeno je u zoni onomastike.

Kako se iz slike vidi, rječnički članak podijeljen je na šest modula (osnovni gramatički podaci, definicija, sintagma, frazeologija, onomastika i etimologija), koje je moguće isključivati i uključivati s pomoću izbornika na lijevoj strani zaslona.

Najpoznatiji *terminološki rječnik* u Hrvatskoj je **Struna** (slika 2) na adresi `http://struna.ihjj.hr`. Struna je terminološka baza hrvatskoga strukovnog nazivlja u kojoj se sustavno prikuplja, stvara, obrađuje i tumači nazivlje različitih struka radi okupljanja i usklađivanja nazivlja na hrvatskome jeziku. Nositelj projekta je *Institut za hrvatski jezik i jezikoslovlje*, a voditeljica *dr.sc. Maja Bratanić*.

Slika 2: Struna

Kako se na *Struni* može pročitati "nazivlje je ključ stručnoga i znanstvenoga sporazumijevanja. Rad na nazivlju stoga je iznimno važan i veoma zahtjevan posao

koji je u nas donedavno bio prilično zanemaren. Osnivanjem *Strune* stvorene su temeljne pretpostavke za njegov razvoj na nacionalnoj razini. Ovim se programom izgrađuje terminološka infrastruktura kakvom već raspolažu gotovo sve europske države. Izgradnja takve infrastrukture osobito je važna za hrvatski jezik otkad je postao službenim jezikom *Europske unije*. Normiranje nazivlja neodgodiv je i dalekosežan zadatak."

Projekt izgradnje hrvatskoga strukovnog nazivlja pokrenut je na inicijativu *Vijeća za normu hrvatskoga standardnog jezika*, a *Hrvatska zaklada za znanost* financijski ga je podupirala od 2008. do 2103. godine.

U *Struni* ravnopravno surađuju stručnjaci jezikoslovne struke sa stručnjacima drugih struka. *Institut za hrvatski jezik i jezikoslovlje* kao nacionalni koordinator pruža terminološku, kroatističku i tehničku podršku.

U *Strunu* je dosad uključeno dvadeset različitih struka, a javnosti su dostupna nazivlja osamnaest struka: arheologija, anatomija i fiziologija, antropologija, brodostrojarstvo, fizika , geomatematika, građevinarstvo, hidraulika i pneumatika, kartografija i geoinformatika, kemija, korozija i zaštita materijala, matematika, polimerstvo, pomorstvo, pravo EU-a, stomatologija, strojni elementi te zrakoplovstvo. U tijeku je obradba nazivlja iz područja fitomedicine i muzikologije.

Na koncu opisa postojećeg stanja on-line rječnika u Hrvatskoj, ne smije se zaobići i on-line **Wiki-rječnik**, tzv. **Wječnik** (sastavljen od dvije riječi wiki i rječnik) koji kontinuirano i djelotovorno stvara rječnik slobodnog sadržaja (slika 3), dostupan na više od 150 jezika: `http://hr.wiktionary.org/wiki/hrvatski`

Slika 3: Wječnik

Dakako, **Wječnik** je povezan s najvećom on-line enciklopedijom (**wikipedijom**) `http://wiki.open.hr/l10n/rjecnik.html`, koja je u detak godina postojanja djelovala na većinu nacionalnih enciklopedija (npr. Enciklopedije *Brittanice*) da iz klasičnog pređu u on-line izdanja.

Postoje i pokušaji paralelnog stvaranja posebne hrvatske wikipedije: `http://`

`wikinfo.org/w/Hrvatski/index.php/Glavna_stranica` ali ona zasad nema osobit značaj.

S obzirom da je ovaj rad usmjeren stvaranju rješenja koje je najbliže, ali funkcionalno posve različito, od postojeće *Strune*, u idućem potpoglavlju obradit će se ustroj računalne baze i pretraživanja, te potom ukazati na nedostatke projekta, koji bi se s novim rješenjem mogli otkloniti.

1.2 Ustroj "*Struna*" terminološke baze

U *Struni* se, kako piše na portalu, u trenutku otvaranja za javnost nalazilo oko 30.000 hrvatskih naziva, njihove istovrijednice na engleskome jeziku i na nekim drugim europskim jezicima.

Iako zapis svakoga naziva može sadržavati mnogo elemenata, najčešće se (kod nalaženja riječi) nalaze samo podebljani elementi iz ovog popisa:

1. **naziv na hrvatskome jeziku**
2. **jezičnu odrednicu**
3. skraćeni oblik naziva
4. **definiciju**
5. kontekst
6. predloženi naziv (IHJJ)
7. istoznačnice (sinonimi), tj. nazivi na hrvatskome jeziku s oznakom normativnoga statusa naziva (dopušteni naziv, predloženi naziv, nepreporučeni naziv, zastarjeli naziv, žargonizam)
8. istovrijednice (ekvivalente), tj. nazive na stranim jezicima
9. suprotnice (antonimi)
10. kratice na hrvatskome i/ili stranome jeziku
11. podređeni nazivi
12. simbol
13. jednadžba
14. formula
15. poveznica
16. napomena
17. privitak

18. **razredba** (oznaka znanstvenoga polja i grane te projekta kojemu naziv pripada)

19. vrela (naziva, definicije i konteksta)

Znanstvena i umjetnička područja raspoređena su prema razredbi iz *Pravilnika Nacionalnoga vijeća za znanost* objavljenoga 2009. godine.

Cilj je terminološkoga opisa normiranje naziva, što znači da se izborom i prikazom naziva u bazi preporučuje uporaba najprihvatljivijega naziva za određeni pojam. Pritom se popisuju i drugi supostojeći nazivi za isti pojam, no svakomu se pridjeljuje njegov normativni status.

Razlikuje se naziv i njegove *istoznačnice*, koje mogu biti označene kao: *dopušteni naziv*, *predloženi naziv*, *nepreporučeni naziv*, *zastarjeli naziv* i *žargonizam*.

Zadaća je *Instituta za hrvatski jezik i jezikoslovlje* predložiti jezično najprihvatljiviji naziv koji se najbolje uklapa u sustav hrvatskoga standardnog jezika, ali i u sustav nazivlja struke o kojoj je riječ. To je u pravilu i naziv od kojega se najlakše tvore značenjski povezani nazivi.

Značenje svakoga naziva opisuje se definicijom koja objašnjava razliku između pojma pridruženoga nazivu i drugih povezanih pojmova.

Pojmovni se sustav gradi i navođenjem podređenih naziva. Primjer uporabe naziva u kontekstu uvijek je izvorni navod iz stručne literature, a navodi se u rubrici *kontekst*.

U zapisu svakoga naziva navode se i *istovrijednice* na drugim jezicima, u pravilu uvijek na engleskome, a često i na drugim europskim jezicima. Tako se osigurava međujezična razmjenjivost terminoloških podataka.

Uz naziv se, ako postoje, prikazuju i kratice, a u skladu s prirodom naziva može se navesti i simbol, formula, jednadžba, poveznica te grafički prilog.

Vrela naziva, definicije i konteksta se obvezatno navodi uz kontekst, dok se uz naziv navodi onda kad se želi naglasiti da je naziv povezan s određenim autorom ili vrelom, ili da ga je skovao sam obrađivač. Vrela za definicije navode se kad je definicija iz vrela doslovno preuzeta ili je tek neznatno preuređena.

Nastojalo se da uporaba tražilice *Strune* bude intuitivna i razumljiva, prikaz naziva pregledan te da korisnik u *Struni* lako i brzo nađe traženi naziv.

1.3 Nedostatci projekta "*Struna*"

Svaki pionirski posao nužno trpi brojne nedostatke, koji se čestom uporabom brzo i lako otkriju. Tako će, na primjer, korisnik, tražeći pojam '*anoda*' dobiti višeznačnost istog pojma, koji su obradili ljudi (stručnjaci) iz različitih područja:

Tablica 1: Različito značenje pojma anoda u raznim područjima

Pojam	Značenje	Područje
anoda	negativno nabijena elektroda	kemija
anoda	pozitivna elektroda u elektrolitskome članku	elektrotehnika
anoda	elektroda na kojoj prevladava anodna reakcija	kemijsko inženjerstvo
anoda	elektroda koja ima veći električni potencijal	fizika
inertna anoda	anoda koja nije podložna anodnomu otapanju i anodnoj koroziji	kemijsko inženjerstvo
netopljiva anoda	anoda koja se primjenjuje u sustavu katodne zaštite s narinutom strujom i koja se tijekom rada sustava ne troši znatno	kemijsko inženjerstvo
pomoćna anoda	dodatna anoda koja se upotrebljava tijekom elektronanošenja radi postizanja željene raspodjele debljine prevlake	strojarstvo

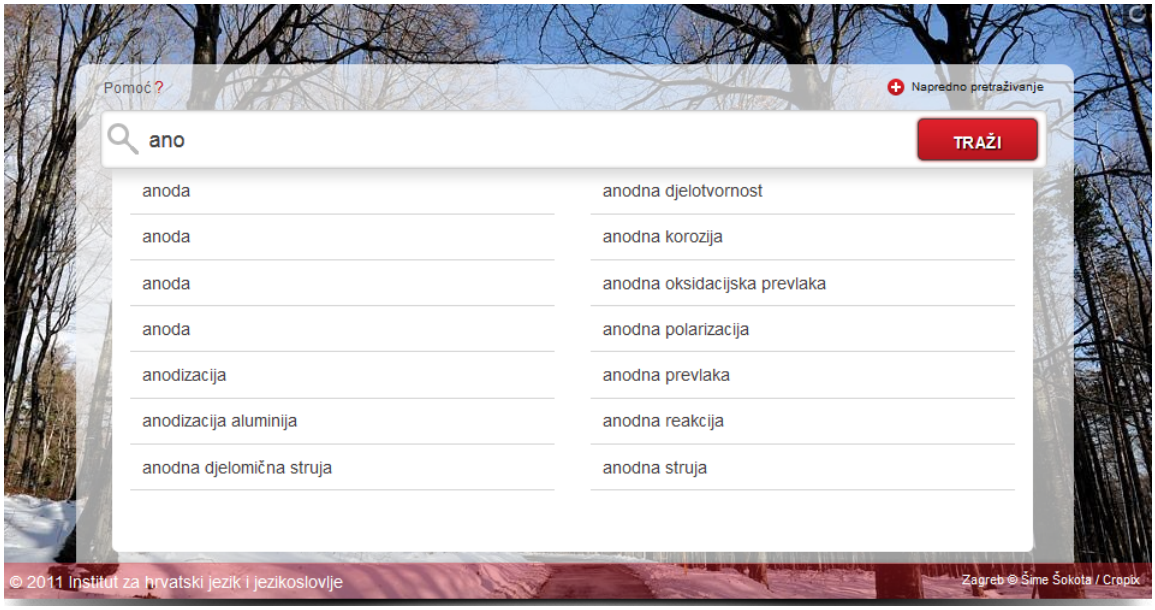
U ovom slučaju, vrlo teško je korisniku, na temelju gornje tablice ili klikom na `http://struna.ihjj.hr/search-do/?q=anoda&naziv=1&polje=0#container`, odgovoriti na sljedeća pitanja:

- je li anoda pozitivna ili negativna elektroda?
- što je to 'anodna reakcija', 'anodno otapanje' ili 'anodna korozija'? (problem poznat kao *'definitio in se'* na latinskom.)
- što znači '**veći** električni potencijal' i 'ne troši **znatno**'? (kvantifikatorski problemi)
- postoje li 'topljive anode' i 'glavne anode' kad postoji 'netopljiva' i 'pomoćna' anoda?

No, ne radi se samo o poteškoćama (ne)znanja i semantičke naravi. Još teži je problem što korisnik ne može doći do neke definicije, ako *unaprijed ne zna natuknicu riječi* (a ako je zna, onda mu definicija vjerojatno niti ne treba). Problem se donekle rješava tako što korisnik pogađa početna slova tražene riječi. Naime, upisom nekoliko slova u obrazac, pojavljuje se popis riječi s istim početnim slovima.

Hoće li se tražena riječ nalaziti među njima, teško je odgovoriti (možda baš počinje s nekim drugim početnim slovima).

Iako je predviđeno da svaka riječ ima svoju jezičnu odrednicu, u većini slučaja ona izostaje, tj. nije poznata temeljna gramatička vrsta (imenica, pridjev, glagol) riječi, a također niti njeni pripadni gramatički oblici (riječi dobivene morfološkim promjenama - deklinacijama ili konjugacijama). To znači da se informacija u "*Struni*"



Slika 4: Pogađanje ili znanje (ne/pouzdanost)?

ne može koristiti za prepoznavanje stručnog nazivlja u stručnim tekstovima, jer se tamo riječi nalaze u različitim gramatičkim oblicima. S tim nedostatkom povezana je i svaka daljnja i sve potrebnija mogućnost kvantitavnih analiza stručnog teksta, npr. otkrivanja čestotnosti nazivlja, kolokacija, uporaba konkordancija i slično.

Veliki nedostatak "Strune" je da ne postoji povezanost riječi u definiciji s nazivima (natuknicama) u bazi. Ako se u definiciji pojavljuju riječi koje se nalaze u bazi, korisnik ih ne može vizualno otkriti, niti na njih 'kliknuti mišem' (ne postoje sveze), nego pokušavati unijeti/utipkati u isti ulazni obrazac, ne bi li našao neku od njih. Čest problem stručnog nazivlja je baš u tomu, što definicija može biti isto tako nerazumljiva kao i natuknica, pa zahtijeva niz iteracija dok se ne dobije/otkrije pravo znanje o pojmu.

Na koncu, centralizirano skupljanje informacije bez pravog on-line pristupa unošenja i ažuriranja informacije, te omogućavanja paralelnog rada brojnih instituta, obraznovnih ustanova i akademske zajednice predstavlja veliko, mukotrpno opterećenje - prije svega za ljude u IHJJ-u koji se brinu o terminološkoj bazi, a s druge strane za stalan rast i svladavanje strukovnih neologizama (novih riječi) koji se bez distribuiranog i paraleliziranog pristupa ne mogu kvalitetno riješiti.

S obzirom da ne postoji ime stručnjaka koji je definirao pojam, a definicije (kako vidimo) mogu bit kontradiktorne, informacija sa Strune je nepouzdana i otporna brzim promjenama.

Na temelju svega uočenog, pojavila se potreba sagledavanja novog pristupa i stvaranja on-line rješenja koje će uspješno odgovoriti na sve nabrojene nedostatke. Štoviše, to rješenje treba dati korisniku mogućnost da bez ikakve intervencije za zadani tekst, stroj sâm povuče i prikaže informaciju za sve riječi koje se u tekstu koji korisnik čita ili napiše. Sasvim je pritom svejedno jesu li to riječi iz nekog općeg on-line (standardnog rječnika), npr. jezičnog portala ili strukovnog, npr. "*Strune*". Današnje mrežne tehnologije to nam omogućuju.

2 LEKSIKON - DIO OPĆEG (MHJ) SUSTAVA

Ispunjenje svih prethodno definiranih zahtjeva u izgradnji on-line tehničkog rječnika ili leksikona, nije moguće bez nužne računalne infrastrukture koja pokriva one segmente koji su potrebni tvorbi i održavanju mrežnog rječnika, a nisu njegov sastavni dio. To se, na primjer, odnosi na program računalnog oblikoslovlja s kojim se tvore, generiraju, svi gramatički oblici neke riječi (dakako i tehničke riječi) na koje korisnik naiđe. Jednom obrađenu i spremljenu takvu riječ sa svim njenim gramatičkim oblicima, više nije potrebno obrađivati. S obzirom da se radi o mrežnom rješenju, taj posao mogu raditi brojni pojedinici, od stručnjaka do nestručnjaka, od instituta i akademske zajednice do škola i nastavnika, dapače i učenika. Problem koji se tu pojavljuje je točnost, ispravnost, dobivenih, odnosno spremljenih podataka. Hoće li tehnička rječnička baza uskoro biti puna 'informacijskog smeća'? Odgovor je: *ne*, jer je **MHJ** - *mrežni sustav hrvatskoga računalnog jezikoslovlja* zamišljen tako, da svaka osoba može imati svoj '*rječnik*' (i puno toga drugog, npr. korpus, morfosintaktički označene rečenicei sl.), a riječi iz njega može (ako želi) ponuditi za '*opći rječnik ili tehnički*' (opći korpus, opće stablene strukture i sl.). Osobe koje su okupljene/izabrane kao administratori u **MHJ** projektu postaju provjeritelji ispravnosti načinjenog/ponuđenog, a nisu više samo nužna i najčešće jedina radna snaga.

Odmah na početku treba spomenuti da se u ovakvom sustavu stvoren i održavan **on-line tehnički rječnik** ni po čemu (osim sadržajem) ne razlikuje od on-line humanističkog, umjetničkog, medicinskog ili općeg (jezikoslovnog) on-line rječnika. Semenatička domena neke riječi može biti opća ili terminološka.

Iako će se često uz pojam *rječnika* pojavljivati i pojam *leksikona*, odmah treba naglasiti da se pod pojmom *tehničkog rječnika* misli na *tehnički leksikon*, dok sâm *rječnik* može značiti i *abecedarij*, popis riječi bez njhove definicije i dodatni atributi, što je čest slučaj kod standardnih, općih rječnika.

U nastavku će prvo bit razrađena zamisao on-line rječnika, ustroj njegovih spremišta i općeg algoritma tvorbe, ažuriranja i dohvaćanja riječi, a potom mjesto rječnika u općem mrežnom sustavu u kojem korisnici na sličan način rade sa svim segmentima jezikoslovlja (oblikoslovlje/morfologija, sintaksa, semantika, korpus).

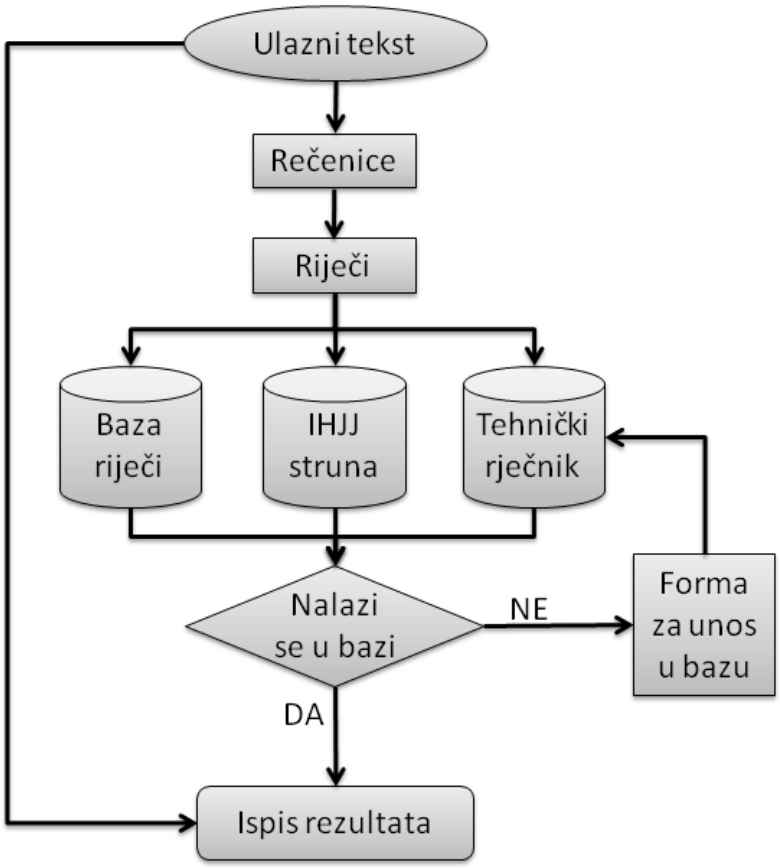
Na koncu će se pokazati uglavljenost ovih dijelova u budući *CroLLOD projekt*, za koji je u ovom radu načinjen temeljni algoritam tvorbe povezanih podataka (*tripleta*) i njihovo povezivanje u *Virtuso triplestore*-u.

2.1 Algoritam tvorbe i održavanja leksikona

Pretpostavimo postojanje ulaznog teksta, bilo da ga korisnik sam napiše ili na neki način dohvati (preko mreže ili iz digitalne knjige). Od sada nadalje o njemu ćemo govoriti kao o stručnom, *tehničkom tekstu* iz kojeg će se generirati tehnički rječnik, iako je program univerzalan i neovisan o terminološkom sadržaju teksta. U ovom pristupu, temelj je **stručni tekst**, jer se u njemu pokazuju riječi koje su već u općim ili terminološkim bazama pohranjene, zajedno sa svojim definicijama i atributima, a ako ih tamo nema, onda postoji mogućnost njihovog definiranja.

Prva zadaća koju je trebalo riješiti je napraviti program koji će tekst rastavljati u rečenice, a potom te rečenice u riječi.

Na slici 5 pokazana je opća ideja algoritma. Iz ulaznog teksta izvlače se rečenice, pa riječi iz svake od njih. Svaka riječ se potom provjerava s obzirom na tri baze: 'bazu riječi' koja sadrži standardne (opće) riječi, kao što je opisana HJP baza (`http://www.hjp.hr`), terminološku bazu (kao što je npr. '*Struna*' `http://struna.ihjj.hr`) i na koncu *Tehnički rječnik* u kojoj su spremljene samo one riječi kojih do tada nije bilo u navedenim dvjema. Važno je napomenuti, da ne postoji ograničenje na broj baza, on-line enciklopdija ili mrežnih repozitorija u kojima bi se riječ mogla provjeravati. U ovom koraku želi se samo naglasiti da se riječ provjerava prvo među općim riječima, zatim među riječima mrežnih rječnika koji su već načinjeni i dostupni i na koncu s obzirom na novi, strukturno drugačiji rječnik (*Tehnički rječnik* na slici) koji se svakodnevno usavršava i raste.



Slika 5: Tvorba i održavanje on-line tehničkog rječnika

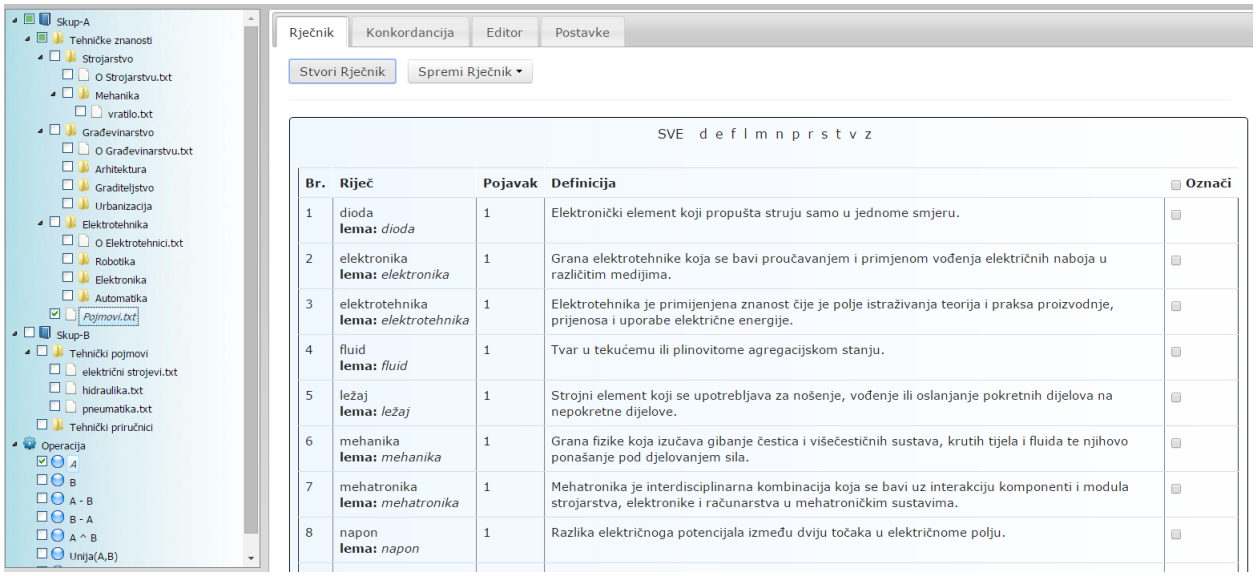
Ako pojam/riječ nije nigdje pronađena, korisniku (koji za to ima dopuštenja) će se ponuditi obrazac za unos pojma u *Tehnički rječnik*. Riječi koje nisu pronađene u bazama bit će u tekstu posebno označene. Klikom na takvu riječ, otvorit će se obrazac za unos.

Ako se pojam već nalazi u bazi nekog od rječnika (osim *Tehničkog*) riječ će biti posebno označena. Klikom na nju, pojavit će se definicija iz dotične baze i mogućnost (link/sveza) za skok na tu riječ u bazi (kako bi se pročitala sva informacija koja je za nju tamo spremljena).

Riječ koja je pronađena u *Tehničkom rječniku* bit će ispisana s brojem pridruženim rječniku i i klikom na nju korisnik će dobiti sve podatke koji su uz nju zadani, a ako je uz to i njen vlasnik, moći će je mijenjati (dakako, samo u svojoj bazi *Tehničkog rječnika*) ili zajedničkog, ako su administratori/(*editorial board*) to odobrili. Riječi u zajedničkoj bazi *Tehničkog rječnika*, odobrene od jezikoslovaca iz MHJ neće se moći mijenjati).

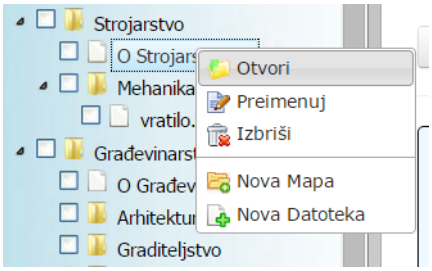
2.2 Ulazni tekst u stablu korpusa

Stručne tekstove moguće je pohraniti u hijerarhijskom stablu koje može imati po volji puno razina, bilo koje namjene (npr. po tematici, područjima i granama i sl.).



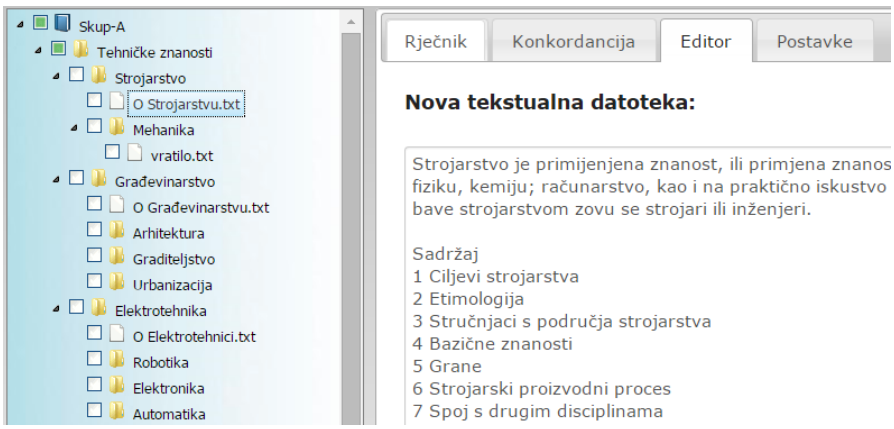
Slika 6: Stablo stručnih tekstova

Ovog časa nije važno na koji sve način korisnik može doći do stručnog teksta, bitno je da ga može pohraniti u stablo i smjestiti u bilo koju mapu sa željenim imenom. Svaki korisnik može svoje stablo generirati na svoj način i pohraniti tekstove koje on želi (s kojima se bavi). U idućem poglavlju bit će detaljnije opisane mogućnosti koje se iz tekstova s navedenim stablom mogu izvoditi (npr. tvorba rječnika, izvlačenje neologizama, usporedba stručnih tekstova, informacija o frekvenciji/čestotnosti riječi i sl.)



Slika 7: Tvorba mapa i datoteka

Izgled zamišljene forme (obrasca) može se vidjeti na slici 8.



Slika 8: Pretinac za unos ili dovlačenje teksta

2.3 Rastavljanje teksta na rečenice i riječi

Priprava teksta za dohvaćanje informacije iz njegovih rečenica postiže se nizom koraka:

1. Tekst se rastavi u rečenice.
2. Rečenice se rastavljaju u riječi i interpunkcijske znakove.

Rastavljanje teksta u rečenice i rečenica u riječi provodi se s dvjema funkcijama koje su zbog specifičnosti hrvatskoga jezika prilagođene iz poznatoga jezikoslovnog modula NLTK[9] (Natural Language ToolKit) za Python. Te funkcije su:

word_tokenize(tekst_rečenice) i *sent_tokenize(tekst_dokumenta)*.

Funkcija *sent_tokenize()* kao ulazni argument uzima tekst dokumenta, a kao rezultat vraća listu rečenica. Iz te liste rečenica uzima se pojedina rečenica kao ulazni argument za funkciju *word_tokenize()*, koja vraća listu riječi za tu rečenicu. Najveći problem koji je trebalo riješiti u prvoj funkciji bio je kako otkriti kraj rečenice. To nije moguće samo na temelju točke, uskličnika i upitnika, jer se isti znakovi (npr. točka) mogu pojavljivati i u drugim okolnostima, npr. kad su dio neke kratice (npr. prof. dr. sc.). Stoga je trebalo prikupiti više od 600 kratica koje su česte u hrvatskome jeziku, te ugraditi stanovitu heuristiku za slučaje koji nisu kratice, a ne mogu biti ni rečenice (sastavljene od riječi s malim brojem slova, kojih pritom nema u bazi).

2.4 Baze podataka za sve vrste rječnika

Baza riječi sadrži jednu tablicu, tablicu s riječima. U njoj je spremljena svaka riječ za sebe, a ne pojmovi kao u druge dvije baze. Svaka riječ je spremljena u bazu sa svojim gramatičkim oblikom, i jednim dodatnim svojstvom koji je u ovom slučaju

lema tj. osnovni oblik riječi iz koje je izvedena. Neke riječi mogu imati više gramatičkih oblika pa su oblici onda razdvojeni znakom "/", te pomoću jednostavnog regularnog izraza možemo pretražiti sve gramatičke oblike.

#	id	rijec	tag	dod1	dod2	dod3
157	157	osmjehnuste	gr0j	osmjehnuti	NULL	NULL
158	158	procvilite	gp2m/gm2m	procviljeti	NULL	NULL
159	159	ustajalijem	pmjd0nk/pmjl0nk/psjd0nk/psjl0nk	ustajao	NULL	NULL
160	160	centrirali	gr0m	centrirati	NULL	NULL
161	161	tornadima	immd0/imml0/immi0	tornado	NULL	NULL
162	162	brbljavoj	pzjd0op/pzjl0op	brbljav	NULL	NULL
163	163	brbljavom	pzji0op	brbljav	NULL	NULL
164	164	čelisticama	izmd0/izml0/izmi0	čelistica	NULL	NULL
165	165	tustima	pmmd0op/pmml0op/pmmi0op/psmd0op/psml0op/psmi0op/pzmd0op/pzml0op/pzmi0op/pmmd0np...	tust	NULL	NULL
166	166	jednogodišnjaci	immn0/immv0	jednogodišnjak	NULL	NULL

Slika 9: Baza riječi (Pinjatela, Zadar)

IHJJ struna baza se također sastoji od jedne tablice. U tablicu su spremljeni pojmovi hrvatskog strukovnog nazivlja sa svojim opisom, linkom koji pokazuje na IHJJ[18] stranicu gdje se pojam nalazi, te stupcima s osnovnom gramatikom (glagoli, imenice i nepoznato). U stupce glagol i imenice spremaju se riječi koje čine pojmove, a koji se nalaze u bazi riječi, a u stupac nepoznato se spremaju riječi iz pojma koje su nepoznate, tj. ne nalaze se u bazi riječi.

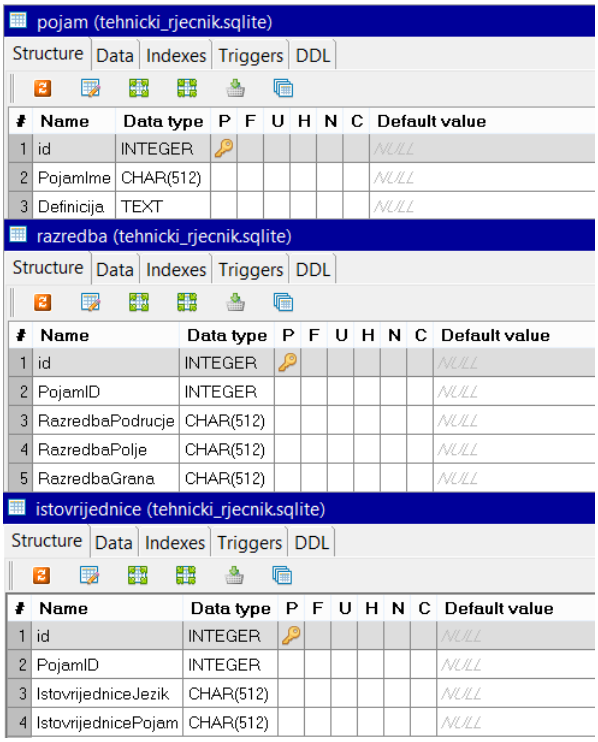
#	id	natuknica	opis	link	glagoli	imenice	nepoznato
1	1	abaksijalan	koji je nasuprot osi tijela	http://struna.ihjj.hr/naziv/abaksijalan/16717/#naziv	NULL	NULL	[u'abaksijalan']
2	2	abakterijski	koji je bez bakterija	http://struna.ihjj.hr/naziv/abakterijski/16780/#naziv	NULL	NULL	[u'abakterijski']
3	3	abapikalan	koji je suprotno od vrha zubnoga korijena	http://struna.ihjj.hr/naziv/abapikalan/16657/#naziv	NULL	NULL	[u'abapikalan']
4	4	aberacija	odstupanje od normalna rasta i razvoja	http://struna.ihjj.hr/naziv/aberacija/14186/#naziv	NULL	NULL	[u'aberacija']
5	5	aberantan	koji odstupa od normalna oblika ili smjera	http://struna.ihjj.hr/naziv/aberantan/16669/#naziv	NULL	NULL	[u'aberantan']
6	6	aberantna žlijezda slinovnica	žljezdano tkivo koje se razvija na neuobičajenim	http://struna.ihjj.hr/naziv/aberantna-zlijezda-slinovnica/15	NULL	NULL	[u'aberantna', u'slinovnica']
7	7	abfrakcija	patološki gubitak tvrdih zubnih tkiva prouzročen t	http://struna.ihjj.hr/naziv/abfrakcija/17303/#naziv	NULL	NULL	[u'abfrakcija']
8	8	ablacija	nestajanje leda ili snijega zbog otapanja i ispari	http://struna.ihjj.hr/naziv/ablacija/23019/#naziv	NULL	NULL	[u'ablacija']
9	9	aboralan	koji se nalazi dalje od usta	http://struna.ihjj.hr/naziv/aboralan/16664/#naziv	NULL	NULL	[u'aboralan']
10	10	abortivni zub	trajni zub prerano izgubljen zbog karijesa	http://struna.ihjj.hr/naziv/abortivni-zub/16704/#naziv	NULL	NULL	[u'abortivni']

Slika 10: Baza termina (*Struna*, IHJJ

Tehnički rječnik se sastoji od tri tablice. Glavna tablica se zove *pojam* i u nju se spremaju sljedeći podatci: *naziv pojma* i *definicija*. Ostale tablice se odnose na polja u obrascu/formi koja se mogu dodavati više puta (vidjeti sliku 12), pa tako imamo tablice razredba i istovrijednice. Tablica *razredba* se sastoji od *područja, polja* i *grane*. U tablici *istovrijednica* se nalaze polja jezik i pojma. Ostale tablice su povezane s glavnom tablicom preko *ID*-a pojma u glavnoj tablici, tj. u svaku od ostalih tablica dodano je još jedno polje naziva *pojamID* koje nam govori na koji na koji pojam se podatak odnosi.

2.5 Obrazac (forma) za definiciju pojma

Obrazac za unos/definiciju pojma sastoji se od sljedećih polja podataka:



Slika 11: Tehnički rječnik

1. **definicije** pojma,

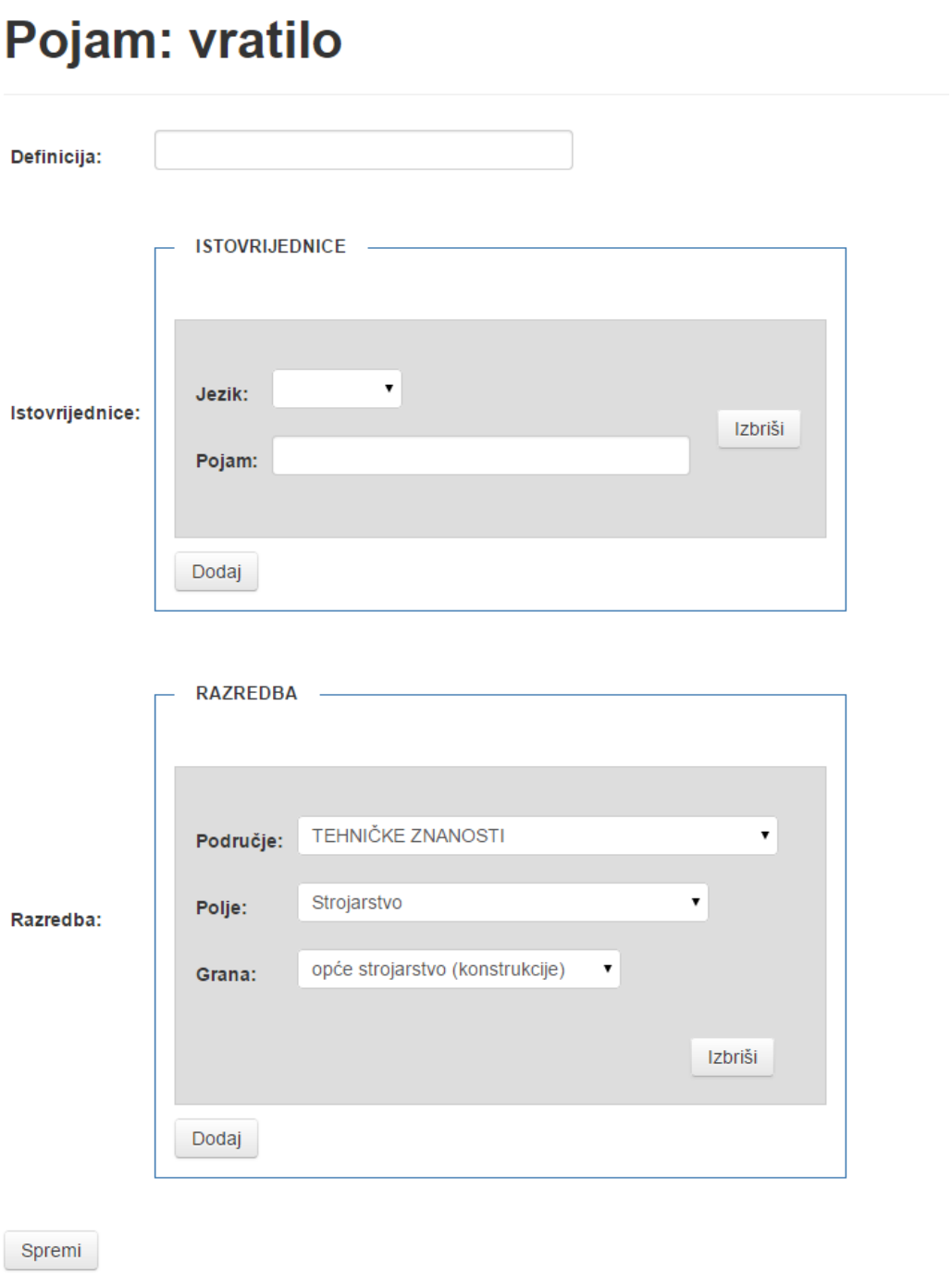
2. **istovrijednice**, tj. naziv toga pojma na hrvatskom i drugim jezicima; moguće je (preko gumba '*Dodaj*') definirati više naziva za isti pojam unutar jezika, pa se istoznačnice i istovrijednice na ovaj način sažimaju,

3. **razredba** određuje *glavno područje* znanosti (*prirodne, tehničke, humanističke*, ...), zatim *polje* kojem taj pojam pripada unutar npr. *tehničkog područja* (*strojarstvo, elektrotehnika, brodogradnja*, ...), te *grana* unutar *polja* npr. *strojarstva* (*opće strojarstvo (konstrukcije), procesno- energetsko, proizvodno strojarstvo* ...)

Obrazac ne dopušta da se jedan pojam unese više puta. Kada se u obrazac unese pojam koji već postoji, on će prvo popuniti sva polja koja su bila ispunjena prilikom njegovog zadnjeg spremanja u bazu, a potom će ažurirati samo novo unesene podatke. To vrijedi isključivo za prijavljenog korisnika koji ima to dopuštenje od strane administratora.

Izgled obrasca može se vidjeti na slici 12.

Ostali podatci/atributi (navedeni u potpoglavlju o ustroju Strune) se ne navode u obrascu, jer se izvlače iz vanjskih, prikladnijih baza (npr. gramatičkih oblika ili kontekst iz korpusa). Ulazni obrazac služi samo za prikupljanje specifičnosti, a ono opće ostaje na općem mjestu.



Slika 12: Obrazac za unos pojmova u bazu podataka

2.5.1 Podjela područja, polja i grane

Na temelju članka 115. stavka 5. Zakona o znanstvenoj djelatnosti i visokom obrazovanju[19] znanstvena i umjetnička područja su:

1. prirodne znanosti
2. tehničke znanosti
3. biomedicina i zdravstvo

4. biotehničke znanosti
5. društvene znanosti
6. humanističke znanosti
7. umjetničko područje
8. interdisciplinarna područja znanosti
9. interdisciplinarna područja umjetnosti

Znanstvena područja dijele se na znanstvena polja, a polja se dalje dijele na grane.

1. PODRUČJE PRIRODNIH ZNANOSTI

1. 1. Matematika
Grane:

1. 1. 1. algebra
1. 1. 2. geometrija i topologija
1. 1. 3. diskretna i kombinatorna matematika
1. 1. 4. matematička analiza
1. 1. 5. matematička logika i računarstvo
1. 1. 6. numerička matematika
1. 1. 7. primijenjena matematika i matematičko modeliranje
1. 1. 8. teorija vjerojatnosti i statistika
1. 1. 9. financijska i poslovna matematika
1. 1. 10. ostale matematičke discipline

1. 2. Fizika
Grane:

1. 2. 1. opća i klasična fizika
1. 2. 2. fizika elementarnih čestica i polja
1. 2. 3. nuklearna fizika
1. 2. 4. atomska i molekulska fizika
1. 2. 5. fizika kondenzirane tvari
1. 2. 6. astronomija i astrofizika
1. 2. 7. biofizika i medicinska fizika

1. 3. Geologija
Grane:

1. 3. 1. geologija i paleontologija
1. 3. 2. mineralogija i petrologija

1. 4. Kemija
Grane:

1. 4. 1. fizikalna kemija
1. 4. 2. teorijska kemija
1. 4. 3. analitička kemija
1. 4. 4. anorganska kemija
1. 4. 5. organska kemija
1. 4. 6. biokemija i medicinska kemija
1. 4. 7. primijenjena kemija

1. 5. Biologija
Grane:

1. 5. 1. biokemija i molekularna biologija
1. 5. 2. botanika
1. 5. 3. mikrobiologija
1. 5. 4. zoologija
1. 5. 5. ekologija
1. 5. 6. genetika, evolucija i filogenija
1. 5. 7. opća biologija

1. 6. Geofizika
Grane:

1. 6. 1. Meteorologija s klimatologijom
1. 6. 2. Fizička oceanografija
1. 6. 3. Seizmologija i fizika unutrašnjosti Zemlje
1. 6. 4. Ostale geofizičke discipline

1. 7. Interdisciplinarne prirodne znanosti
Grane:
1. 7. 1. metodike nastavnih predmeta prirodnih znanosti
1. 7. 2. znanost o moru
1. 7. 3. znanost o okolišu
1. 7. 4. znanost o zračenju

2. PODRUČJE TEHNIČKIH ZNANOSTI

2. 1. **Arhitektura i urbanizam**
Grane:
2. 1. 1. arhitektonsko projektiranje
2. 1. 2. urbanizam i prostorno planiranje
2. 1. 3. arhitektonske konstrukcije, fizika zgrade, materijali i tehnologija građenja
2. 1. 4. povijest i teorija arhitekture i zaštita graditeljskog naslijeđa
2. 1. 5. pejsažna arhitektura

2. 2. **Brodogradnja**
Grane:
2. 2. 1. konstrukcija plovnih i pučinskih objekata
2. 2. 2. hidromehanika plovnih i pučinskih objekata
2. 2. 3. osnivanje plovnih i pučinskih objekata
2. 2. 4. tehnologija gradnje i održavanje plovnih i pučinskih objekata

2. 3. **Elektrotehnika**
Grane:
2. 3. 1. elektroenergetika
2. 3. 2. elektrostrojarstvo
2. 3. 3. elektronika
2. 3. 4. telekomunikacije i informatika
2. 3. 5. radiokomunikacije
2. 3. 6. automatizacija i robotika

2. 4. **Geodezija**
Grane:
2. 4. 1. kartografija
2. 4. 2. fotogrametrija i daljinska istraživanja
2. 4. 3. pomorska, satelitska i fizikalna geodezija
2. 4. 4. primijenjena geodezija
2. 4. 5. geomatika

2. 5. **Građevinarstvo**
Grane:
2. 5. 1. geotehnika
2. 5. 2. nosive konstrukcije
2. 5. 3. hidrotehnika
2. 5. 4. prometnice
2. 5. 5. organizacija i tehnologija građenja

2. 6. **Grafička tehnologija**
Grana:
2. 6. 1. procesi grafičke reprodukcije

2. 7. **Kemijsko inženjerstvo**
Grane:
2. 7. 1. reakcijsko inženjerstvo
2. 7. 2. mehanički, toplinski i separacijski procesi
2. 7. 3. analiza, sinteza i vođenje kemijskih procesa
2. 7. 4. kemijsko inženjerstvo u razvoju materijala
2. 7. 5. zaštita okoliša u kemijskom inženjerstvu

2. 8. **Metalurgija**
Grane:
- 2. 8. 1. procesna metalurgija
- 2. 8. 2. mehanička metalurgija
- 2. 8. 3. fizička metalurgija

2. 9. **Računarstvo**
Grane:
- 2. 9. 1. arhitektura računalnih sustava
- 2. 9. 2. informacijski sustavi
- 2. 9. 3. obradba informacija
- 2. 9. 4. umjetna inteligencija
- 2. 9. 5. procesno računarstvo
- 2. 9. 6. programsko inženjerstvo

2. 10. **Rudarstvo, nafta i geološko inženjerstvo**
Grane:
- 2. 10. 1. rudarstvo
- 2. 10. 2. naftno rudarstvo
- 2. 10. 3. geološko inženjerstvo

2. 11. *Strojarstvo*
Grane:
- 2. 11. 1. opće strojarstvo (konstrukcije)
- 2. 11. 2. procesno energetsko strojarstvo
- 2. 11. 3. proizvodno strojarstvo
- 2. 11. 4. brodsko strojarstvo
- 2. 11. 5. precizno strojarstvo

2. 12. **Tehnologija prometa i transport**
Grane:
- 2. 12. 1. cestovni i željeznički promet
- 2. 12. 2. pomorski i riječni promet
- 2. 12. 3. poštansko-telekomunikacijski promet
- 2. 12. 4. zračni promet
- 2. 12. 5. inteligentni transportni sustavi i logistika

2. 13. **Tekstilna tehnologija**
Grane:
- 2. 13. 1. tekstilno-mehaničko inženjerstvo
- 2. 13. 2. tekstilna kemija
- 2. 13. 3. odjevna tehnologija
- 2. 13. 4. dizajn tekstila i odjeće

2. 14. **Zrakoplovstvo, raketna i svemirska tehnika**
Grane:
- 2. 14. 1. konstrukcija i osnivanje letjelica
- 2. 14. 2. zrakoplovne tehnologije i održavanje
- 2. 14. 3. vođenje i upravljanje letjelicama

2. 15. **Temeljne tehničke znanosti**
Grane:
- 2. 15. 1. automatika
- 2. 15. 2. energetika
- 2. 15. 3. materijali
- 2. 15. 4. mehanika fluida
- 2. 15. 5. organizacija rada i proizvodnje
- 2. 15. 6. tehnička mehanika (mehanika krutih i deformabilnih tijela)
- 2. 15. 7. termodinamika

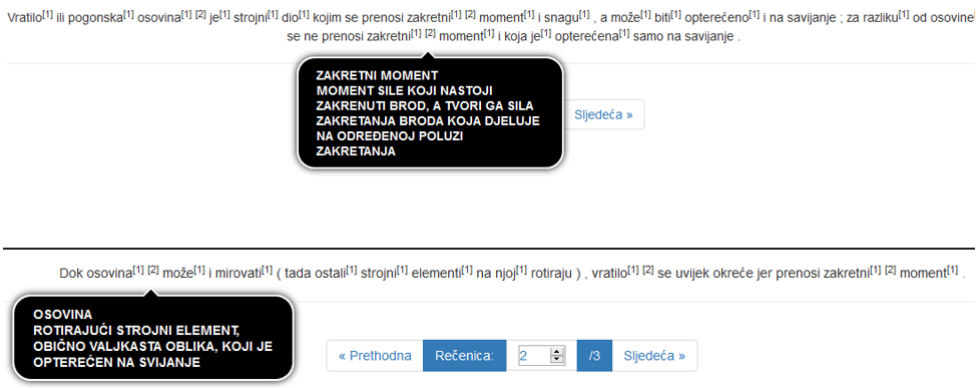
2. 16. **Interdisciplinarne tehničke znanosti**
Grane:
- 2. 16. 1. inženjerstvo okoliša
- 2. 16. 2. mikro i nanotehnologije

2.6 Prikaz rezultata

Ulazni tekst se ili upisuje (dovlači) ili izabire iz označenog stabla, što se vidi u *Lexicon* pretincu (tab) na slika 13.

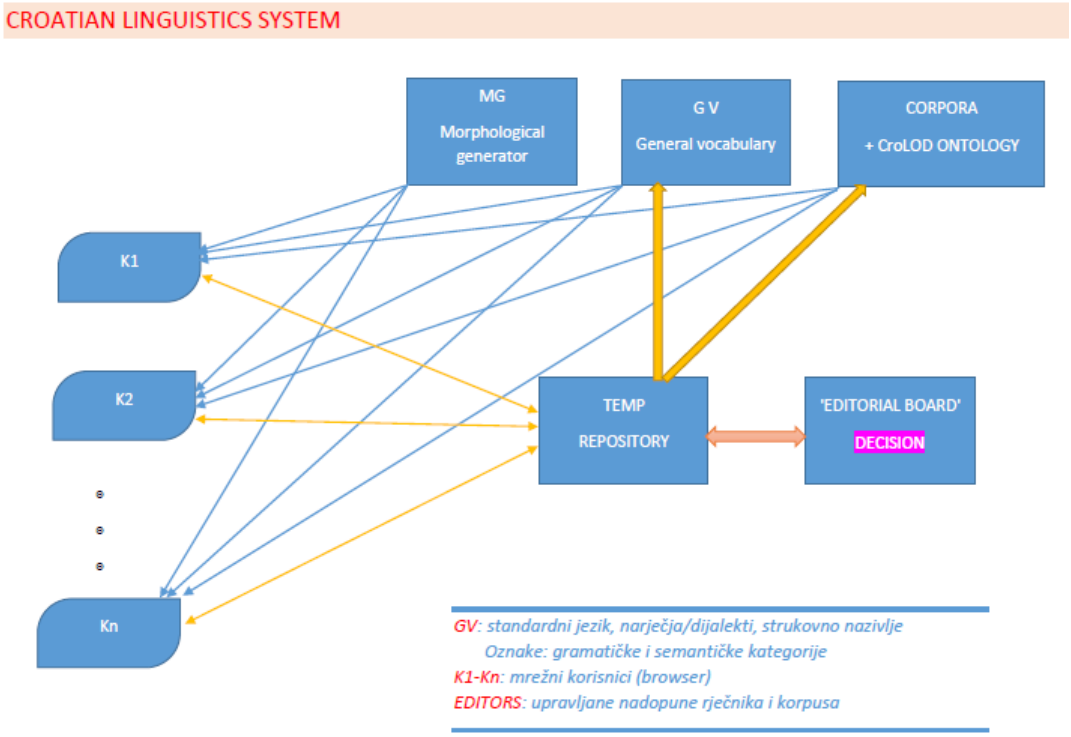
Slika 13: Ispis nakon obrade teksta

Za običnog korisnika, koji ne može mijenjati tehnički leksikon, pojavljuje se samo analizirana rečenica (slika 14 u kojoj se poviše prepoznatih riječi u rječnicima pojavljuje broj izvora, a dolaskom na njega (mišem) ispisuje definicija.
Ispod rečenice se nalazi navigacija na sljedeću, prethodnu i željenu rečenicu.



Slika 14: Ispis nakon korisnikovog uključivanja tooltip-a

2.7 Integrirani mrežni sustav



Slika 15: Opći model sustava

Opisani sustav za tvorbu i obradbu mrežnih rječnika samo je dio općeg, integriranog sustava koji prikazuje slika 15. S oznakom $K1$, $K2$, ... Kn prikazani su korisnici, svaki sa svojim mrežnim preglednikom (browserom), bilo gdje na svijetu. Svaki od njih može dohvaćati/koristiti MG - morfološki generator, GV - opći leksikon (standarnih i strukovnih riječi) i opći korpus (standarnih i strukovnih dokumenata, sintaksnih stabala, ontologija i dr.). Opisano dohvaćanje je jednosmjerno, korisnik ne može ništa unositi ili mijenjati u zajedničkim spremištima. Međutim, u $TEMP$ repozitoriju - korisničkoj bazi, svaki korisnik može spremati ili stvarati sve što mu je ponuđeno (leksikon, sintaksna stabla, ontologije) i omogućiti drugima da njegovo koriste. Hoće li to korisnikovo blago bit uključeno u zajednička spremišta, odlučuje '*Editorial board*' - stručnjaci, administratori, okupljeni oko ovog projekta.

Treba napomenuti, a što se iz slike 15 može vidjeti, da je veza s $TEMP$ repozitorijem za svakog korisnika dvosmjerna, što znači da se on uvijek može nastaviti služiti sa svojom informacijom koju je do tada spremio, bez obzira je li ona ili nije, pod budnim okom stručnjaka, prebačena u 'opće dobro'.

2.8 MHJ ⇒ CroLLOD ⇒ semantički oblak

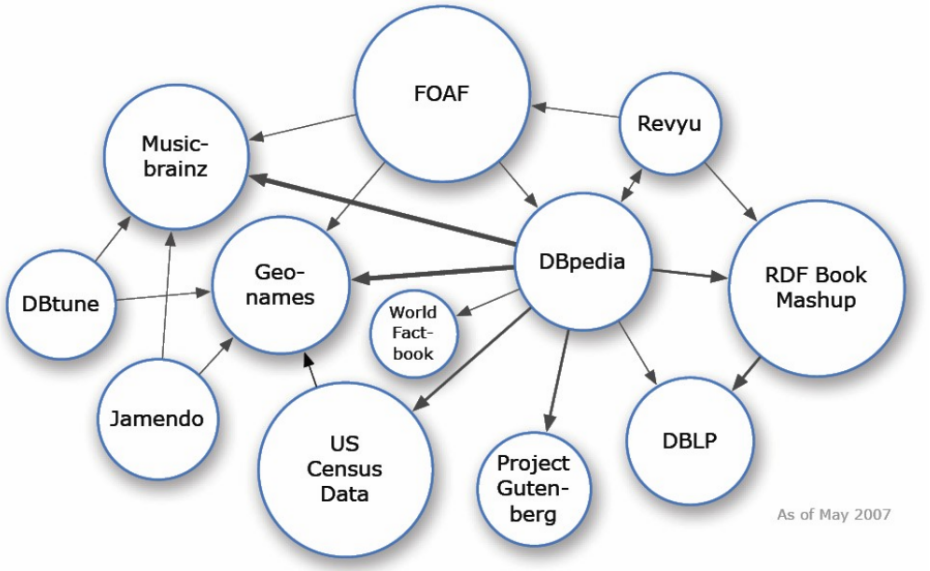
Povećanje količine podataka na internetu, donijelo je niz prednosti za čitavo čovječanstvo. Informacije za koje je nekada trebalo provesti mnogo vremena u knjižnici, uz telefon ili danima čekati da stignu poštom, sada su od nas udaljene tek nekoliko "klikova"

mišem. Međutim, podaci na mreži namijenjeni su ljudima, a čak i kada dolaze iz strukturiranih i dobro opisanih izvora, njihova struktura najčešće nije razumljiva računalima koja ga obrađuju.

U svom radu koji predstavlja početak semantičkog weba, Berners Lee iznosi ideju: *Semantički web* će donijeti strukturu sadržaju web stranica (u kojima informacija nije strukturirana), stvarajući okruženje u kojem softverski agenti putujući s jedne stranice na drugu mogu obaviti složene (sofisticirane) zadaće za korisnike.

Inicijativa *Linked Data* tj. *Povezanih podataka* polako stvara golemi globalni graf i transformira Semantički web iz niza nepovezanih "otoka" (kakav je i naš tehnički rječnik unutar mrežnog sustava), u kompaktni prostor podataka. Naziv *Linked Data* odnosi se na skup uputa i dobrih praksi za objavu i povezivanje strukturiranih podataka na webu. Globalni graf podataka povezanih u skladu s tim uputama naziva se *Oblak povezanih podataka*, (engl.*LODC, Linking Open Data Cloud*).

U početku se sastojao od 12 međusobno povezanih izvora (slika 16), koji i danas čine njegovu jezgru:

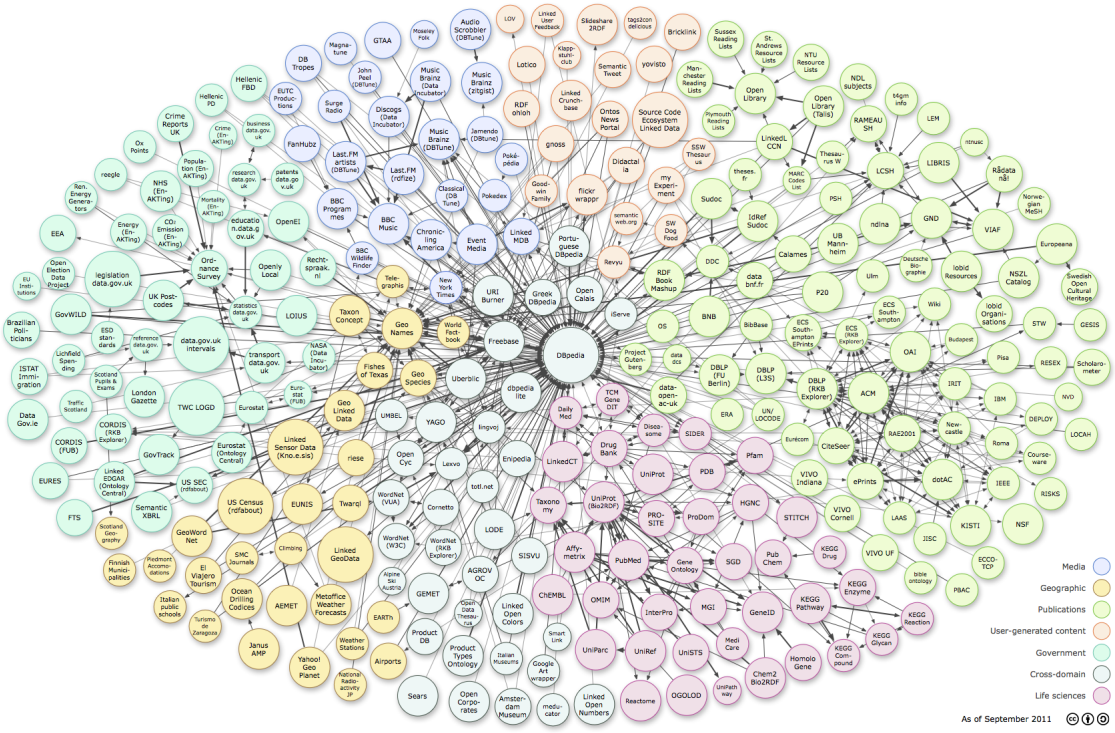


Slika 16: Oblak povezanih podataka - svibanj 2007. godine

Ubrzo im se, objavljujući svoje podatke u skladu s pravilima inicijative, pridružio velik broj organizacija i pojedinaca. Prema trenutnim procjenama, oblak sadrži oko 330 izvora s preko 31 milijarde trojki i 500 milijuna veza. Slika 17 prikazuje stanje oblaka podataka iz rujna 2011. godine. Podaci pokrivaju najrazličitija područja – od knjiga i znanstvenih publikacija, preko glazbe i filmova, pa do podataka o lijekovima ili strojarskim elementima:

Zamišljeni algoritam za tvorbu i obradbu tehničkih tekstova, proširio bi se uputama za semantičke odnose što bi se prikazalo slikom **??** dalo i opisalo na sljedeći način:

1. Prvo se iz teksta izvlače sve riječi uz pomoć regularnih izraza.



Slika 17: Oblak povezanih podataka - rujan 2011. godine

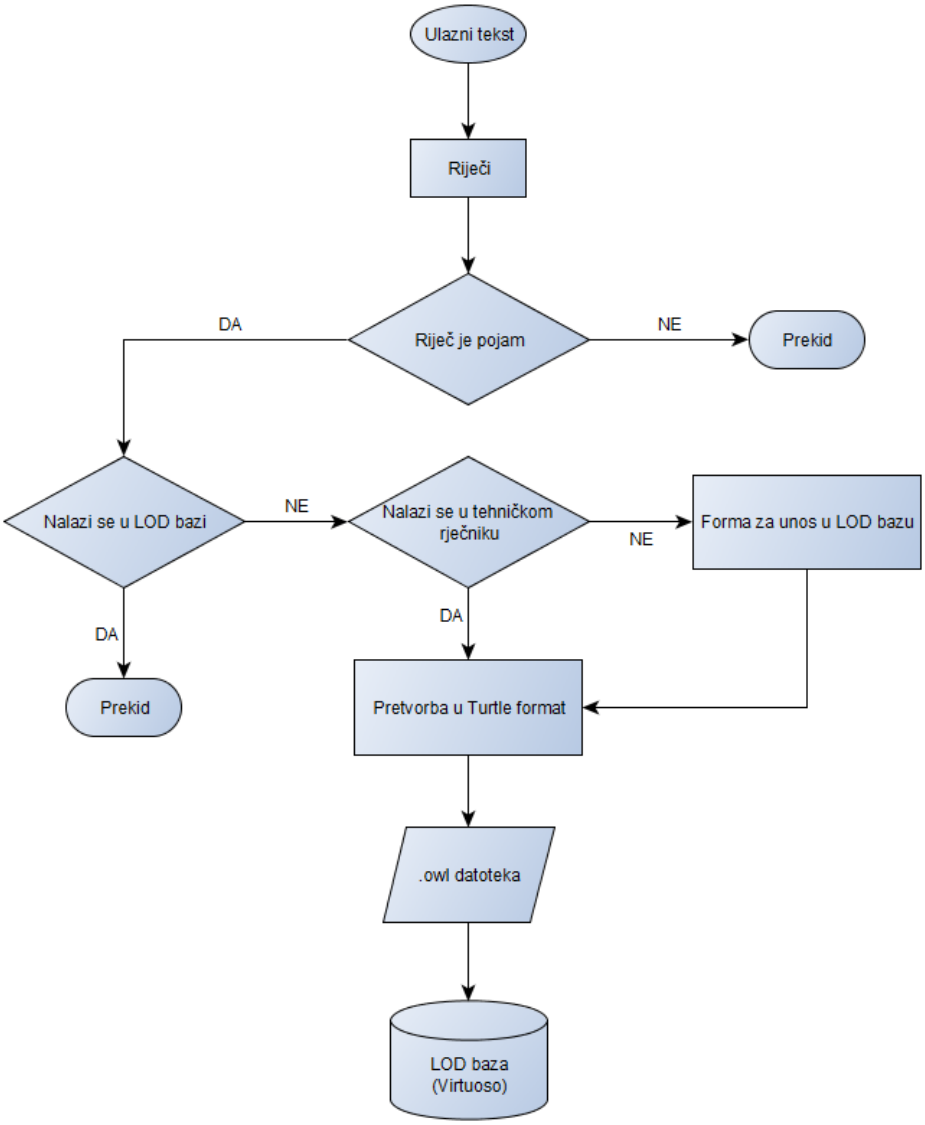
2. Zatim bi se iz tih riječi izdvajaju sve one koje predstavljaju pojam, tj. koje imaju neko značenje (nisu čestice, veznici, prilozi ili pridjevi).

3. Nakon toga, sljedi provjera je li određeni pojam već pripada *triplestore* bazi, pa ukoliko pripada, više se ne obrađuje.

4. Ako se ustanovi da pojam ne pripada *triplestore* bazi, slijedi provjera pripadnosti pojma tehničkom rječniku. Ako pojam pripada rječniku, vrši se pretvorba svih podataka koji su za njega u tehničkom rječniku pronađeni u "Turtle" format. Ako pak ne pripada, prikazaje se obrazac za unos podataka u *triplestore* bazu, te se nakon njenog popunjavanja načini pretvorba u '*Turtle*' format.

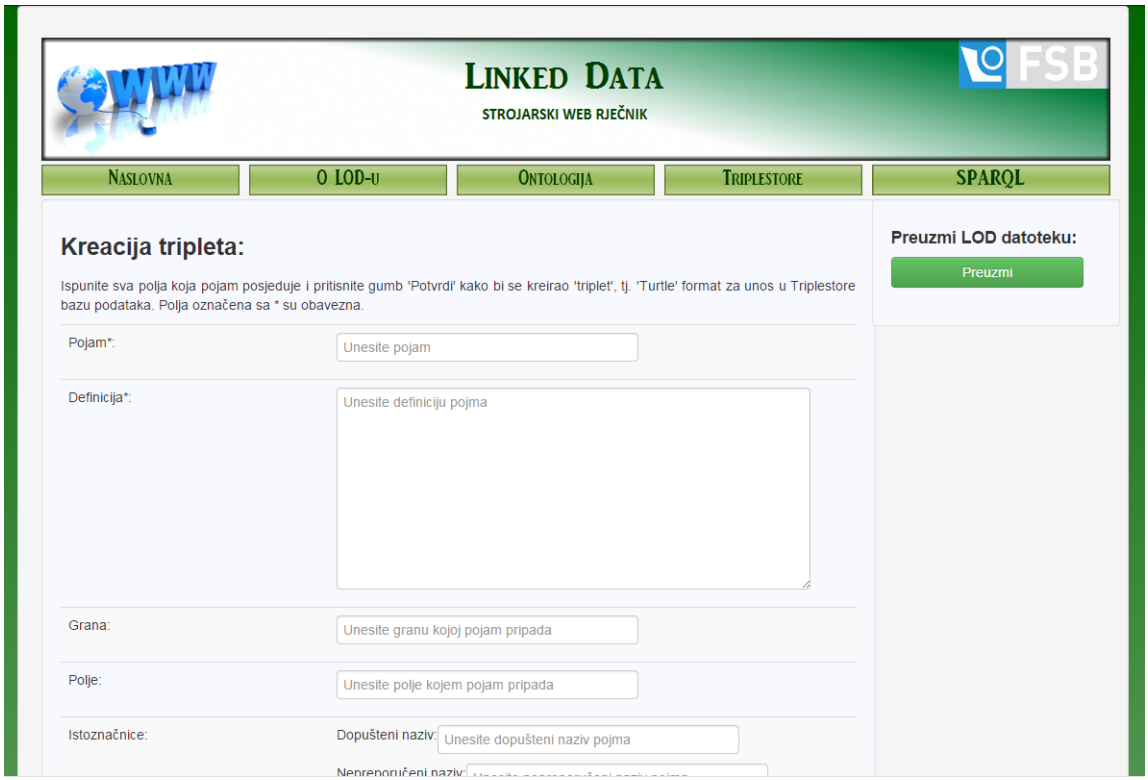
5. Na koncu se stvoreni tripleti ubacuju u datoteku pod nazivom '*Turtle.owl*', koja se može učitati iz bilo koje baze koja ima mogućnost čitanja '*Turtle.owl*' zapisa. U ovom slučaju, to je *triplestore* baza podataka koju posjeduje *OpenLink Virtuoso* univerzalni server.

Unutar mrežne stranice za povezane podatke, nalaze se: obrazac za tvorbu trojaca iz podataka unesenih u pojedina polja, gumb za preuzimanje "Turtle.owl" datoteke, koja sa svojim definiranim trojcima predstavlja ulazni podatak za *triplestore* bazu podataka, te dio pod nazivom "Obrada tehničkih tekstova", koji sadrži polje s tekstom za obradu unešenih tekstova (može sadržavati veliku količinu podataka, tj. pojmova). Izgled mrežne stranice prikazuju slika 19.

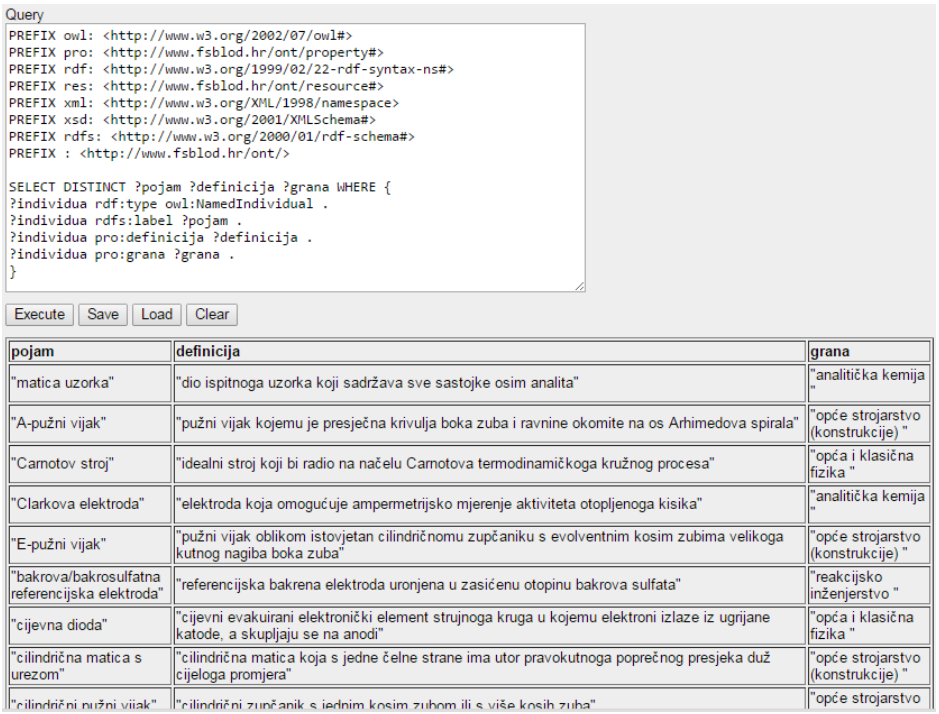


Slika 18: Algoritam za obradbu tehničkih tekstova kroz povezane podatke

SPARQL dio stranice sadrži obrazac koji se sastoji od polja za unos SPARQL naredbi za potragu povezanih podataka i gumba "Pretraži" s kojim pretraga počinje. Ispod tog obrasca pod dijelom "*Rezultati pretrage*", ispisuju se rezultati (za sada) samo u tabličnom obliku (slika 20).



Slika 19: Obrazac za stvaranje LOD trojaca



Slika 20: Traženje pojmova SPARQL-om

3 PROGRAMSKA TEHNOLOGIJA

Rješenje *Mrežnog programa za tvorbu i obradbu tehničkih rječnika* uključilo je više različitih, modernih tehnologija. Tu je u prvom redu Web2py tehnologija, zamišljena na MVC (*model-view-controller*) paradigmi i izvedena u *Python* programskom okruženju. To je značilo da su svi programi, moduli koji su razvijani, bili pisani u *Python* programskom jeziku, iako *Web2py* može pozivati i programe iz nekih drugih programskih jezika (*C++, Java*). Izmedju brojnih modula koje Python ima, najveća pozornost i uporaba posvećena je 're'-modulu za regularne izraze, s kojim se informacija iz rečenica 'čupa' i pretvara u *s-p-o* trojce (*subject-predicate-object triplets*). Trojci su temeljna struktura semantičkog weba, a temeljeni su na *RDF* modelu i njegovim tehnologijama izvedbe (*XML, Turtle, JSON, RDFS*) i programskom jeziku za stvaranje ontologija (*OWL*) i traženju informacija (*SPARQL*). U idućim potpoglavljima ukratko su opisane sve navedene tehnologije, kako bi se lakše čitali u njima realizirani programski odsječci (pogotovo u *Dodatku*).

3.1 O Web2py-u

Web2py je besplatni program za izradu web aplikacija. Napisan je u python-u i programira se s python-om. Massimo Di Pierro najzasluženiji je za njegovo stvaranje. Web2py je 2012. godine proglašen tehnologijom godine. Jednostavan je za korištenje i ne zahtijeva nikakvu instalaciju i dodatnu konfiguraciju.

Njegovo administracijsko sučelje (slika 21) omogućuje stvaranje nove aplikacije, brisanje stare aplikacija i instaliranje već gotovih aplikacija. Omogućuje takodjer jednostavan uvid u sve datoteke i rad s njima, te pregled baze i funkcija koje se koriste u aplikacijama.

Slika 21: Administracijsko sučelje u web2py-ju [1]

Svaka web2py aplikacija sastoji se od *modela* (datoteke koje sadrži opis prikaz

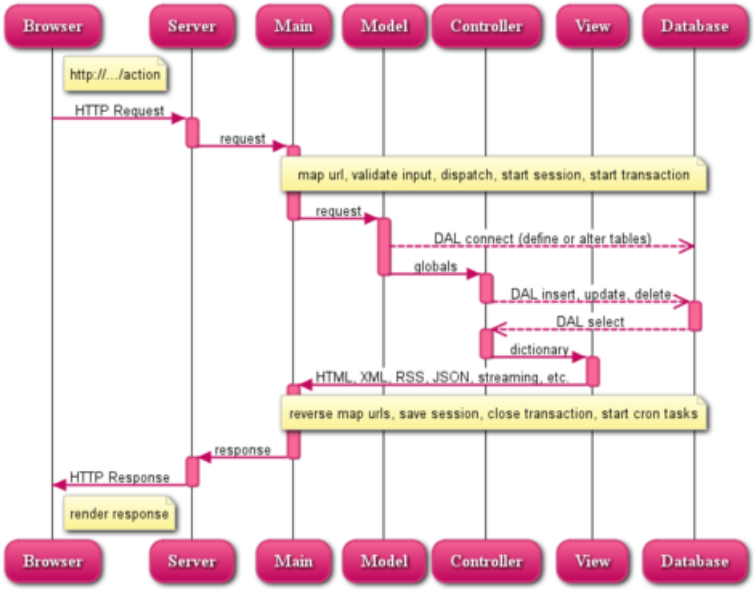
baze), *pogleda* (datoteke koje sadrže opis prikaza baze), *kontrolera* (datoteke koje sadrže opis poslovne logike i tijeka rada), *Cron Jobs*-a (zadatci koji moraju biti redovito izvršena u pozadini), *modula* (kolekcija klasa i funkcija) i *statičkih* datoteka (slike, skripte, ...).

3.1.1 Tok podataka

Tok podataka (slika 22) u web2py-u je sljedeći:

- HTTP zahtjev dolazi na web server (ugrađeni Rocket server ili drugačiji server spojen s web2py putem WSGI[1] ili nekog drugog adaptera). Web server obrađuje svaki zahtjev u svom thread-u (programskoj niti) i paralelno s ostalim zahtjevima.
- Zaglavlje HTTP zahtjeva raščlanjuje se i predaje dispečeru.
- Dispečer zatim odlučuje koja od instaliranih aplikacija će obraditi zahtjev i mapira *PATH_INFO* u URL koji poziva funkciju. Svaki URL odgovara jednoj funkciji.
- Zahtjevi koji uključuju datoteke iz static mape obrađuju se direktno, a velike datoteke su automatski poslane korisniku.
- Prije pozivanja akcije, događa se par stvari: ako zaglavlje zahtjeva sadrži session cookie za aplikaciju, vraća se session objekt. Ako ne sadrži, onda se kreira novi session te je time kreirana okolina za izvršavanje zahtjeva i u njoj se izvršavaju model.
- Konačno, izvršavaju se naredbe u controller-u u prije definiranoj okolini.
- Ako funkcija vraća string, on se vraća korisniku.
- Ako funkcija vraća iterabilan objekt, on se pomoću petlje vraća korisniku.
- Ako funkcija vraća rječnik web2py pokušava pronaći view za prikaz rječnika. View mora imati isto ime kao i funkcija i istu ekstenziju kao početna stranica
- Ako se uspješno izvrše sve korisničke naredbe transakcija se prihvaća.
- Ako se korisničke naredbe ne izvrše greška se sprema u ticket i korisniku se vraća ticket ID. Samo sistemski administrator može otvarati i čitati ticket. Ako mu to ne uspije, web2py pokušava otvoriti generički view. Cijeli korisnički kod se izvršava u jednoj transakciji ako nije drugačije specificirano.

[1]WSGI (Web Server Gateway Interface) je skup specifikacija koje opisuju kako web server komunicira s web aplikacijama, te kako više aplikacija obradi jedan zahtjev.



Slika 22: Tok zahtjeva u web2py-u [1]

3.1.2 Model

U model se tipično spremaju baze podataka i funkcije koje služe za rad s bazama podataka. Svaka baza i funkcija u modelu postaju globalna varijabla bez da ju se prethodno mora definirati globalnom.

Web2py dolazi s DAL ("Database Abstraction Layer") klasom koja služi za kreiranje tablica i njeno korištenje. DAL dinamički generira sql sintaksu u realnom vremenu pomoću određenih pravila tako da programer ne mora učiti različite sql sintakse za različite tipove baza podataka. Podržani tipovi baza podataka: SQLite, MySQL, Oracle, ...

DAL objekt funkcionira slično kao python-ov modul sqlite3, samo što ne moramo koristiti sql sintaksu već koristiti gotove metode objekta za kreiranje tablica i rad s njima. Prvo se stvara veze s bazom podataka ako ona postoji, a ako ne postoji onda se stvara baza i veza. Nakon što smo se spojili s bazom podataka koristimo metodu *define_table* za stvaranje tablice koja kao argument prima ime tablice i polja koja želimo stvoriti. Metoda sama automatski dodaje polje id. Svakom polju možemo definirati njegov tip i ograničenja.

```
1 baza=DAL('sqlite://moja_baza.db')
2 baza.define_table('moja_tablica',
3     Field('polje_1', 'string'),
4     Field('polje_2', 'integer', required=True))
```

Spremanje podataka vrši se pomoću metode *insert*. Prvo moramo pristupiti tablici u bazi pa onda koristimo metodu. Spremati podatke možemo na 3 načina:

1. kroz *for* petlju punimo bazu gdje metodi navodimo podatak koji želimo spremiti u to polje

```
polje_1=['kruske','jabuke']
polje_2=[4,3]
for i in xrange(len(polje_1)):
    baza.moja_tablica.insert(polje_1=polje_1[i],polje_2=
        polje_2[i])
```

2. koristeći rječnik čiji ključevi odgovaraju poljima u tablici

3. koristeći sql sintaksu koja je objašnjenja u poglavlju **??**

Vađenje podataka iz tablice radi se metodom *select*. Za vađenje podataka prvo moramo navesti objekt baze kojem u argumentu navodimo bazu, tablicu i stupac za koji se treba ispuniti uvjet te na to pozivamo metodu *select* koja može i ne mora primati argumente. Ako se u pozivu metode ne navedu argumenti to znači da ona iz tablice vadi sve podatke za ispunjeni uvjet, a ako joj se navedu argumenti onda iz baze za taj uvjet vadi samo navedena polja.

```
rows = baza(baza.moja_tablica.id>0).select(baza.moja_tablica.
    polje_1)
```

DAL nam nudi metodu *executesql* pomoću koje možemo koristiti sql sintaksu. Ona nam isto omogućava stvaranje index-a nad tablicom jer DAL objekt ne posjeduje metodu za stvaranje index-a na tablici.

```
db.executesql('SELECT * FROM moja_tablica;')
```

Pomoću *drop* metode možemo ispustiti sve tablice i svi podatci bit će izbrisani.

```
db.moja_tablica.drop()
```

3.1.3 Kontroler

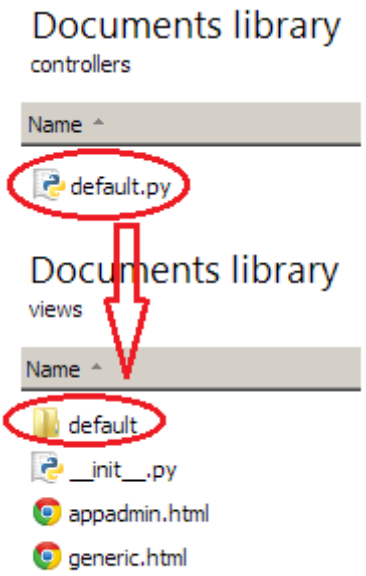
Svaki kontroler ima svoje ime i njegovo ime definira vezu između mape u view-u. To znači da ako imamo kontroler imenom *default* znači da u view-u moramo imati mapu imenom *default*, što je ilustrirano na slici 23.

Svaki kontroler se sastoji od funkcija. Razlikujemo tri glavne vrste funkcija:

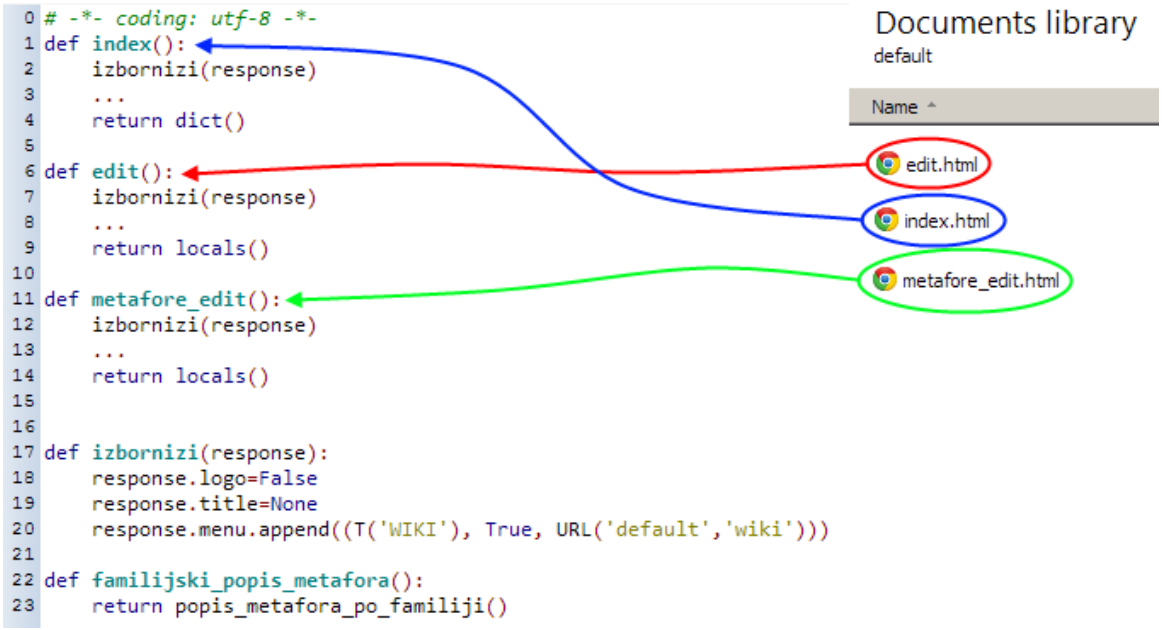
1. funkcije čije se ime veže na pojedinu stranicu aplikacije.

2. funkcije koje pozivamo unutar funkcija i služe nam skratiti kod.

3. funkcije koje se pozivaju pomoću *ajax-a*. One se ne vežu na ni jednu stranicu već se izvršavaju unutar te stranice, a vežu se na pojedini id nekog elementa gdje se nešto klikom želi promijeniti.



Slika 23: Veza između imena kontrolera i view-a



Slika 24: Veza između funkcija i odgovarajuće datoteke u view-u

Na slici 24 možemo vidjeti sve tri vrste funkcija i veze između funkcija koje se vežu na pojedinu stranicu.

Osim rječnika i string-a funkcija može vratiti lokalne varijable:

```
1 def index(): return locals()
```

preusmjeriti korisnika na drugu stranicu:

```
1 def index(): redirect(URL('druga_stranica'))
```

vratiti HTTP stranice koje govore o nekoj grešci:

```
def index(): raise HTTP(404)
```

vratiti "helper":

```
def index(): return FORM(INPUT(_name='test'))
```

ili rječnik koji sadrži "helper"

```
def index(): return dict(form=SQLFORM.factory(Field('name')).
    process())
```

Svaki "helper" prima iste argumente kao i element kojeg zamjenjuje u HTML kodu uz iznimku što kod "helper" moramo staviti _ (donju crticu) ispred argumenta jer bi se u protivnom njegovo ime miješalo s Python-ovim ključnim riječima kao što su id, class... Oni koji se ne podudaraju s Python-ovim ključnim riječima pišu se isto _ (donju crticu) zbog jednostavnosti.

```
polje_za_unos=INPUT(_name='ime', _type='text', _id='prvi',
    _style="color:red;")
```

3.1.4 Pregled (view)

U view-u se nalazi sav html kôd koji definira izgled stranice. Isto tako u view-u možemo pisati python-ov kod ali moramo naglasiti da se radi o Python-u, pa ga pišemo u vitičastim zagradama.

```
{{ python kod }}
```

Kod pisanja python-ovih naredba i view-u moramo obratiti pozornost na petlje i uvjete. Svaka petlja i uvjet mora imati svoj početak i kraj. To je u html kodu ne možemo riješiti pomoću tabova kako smo naučili u Python editoru-u već moramo staviti naredbu *pass* na kraju, koja interpreter-u označava kraj petlje ili uvjeta.

```
{{for i in xrange(1,11):}}
  <span>{{=i}}</span>
{{pass}}
```

Početak svake stranice u view-u je isti i započinje s naredbom *extend 'layout.html'* pomoću koje odabiremo osnovni izgled stranice, tj. poziciju meni-a, podjelu stranice na stupce i retke. Nakon toga može se, ali i ne mora, nalaziti naredba *block head* koja sve skripte i css kodove uključuje u head html stranice tako da se učitaju prije nego što se ostatak stranice učita. Ta naredba nije potrebna ako nemamo nikakve skripte i css te ako smo ih već uključili u layout-u. Na kraju dolazi *div* element s id-om *container* koji nije ništa drugo nego centrirani div element koji sadrži cijelu našu stranicu.

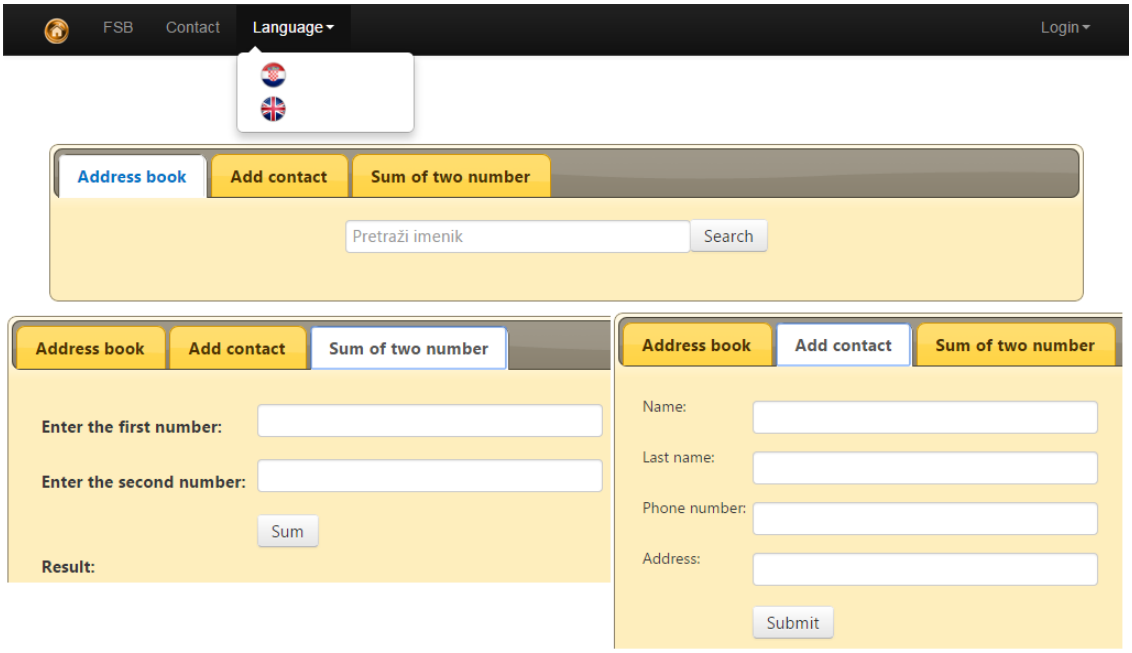
```
{{extend 'layout.html'}}

{{block head}}
  <!-- Ukljucivanje skripti i css -->
{{end}}
```

```
6
7 <div id="container">
8   <!-- Preostali html kod stranice -->
9 </div>
```

3.1.5 Primjer web2py aplikacije

Na slici 25 možemo vidjeti primjer *web2py* aplikacije koja je načinjena od prethodno navedenih primjera *model-pogled-kontrolera*.



Slika 25: Primjer web2py aplikacije

3.2 Regularni Izrazi

3.2.1 Uvod

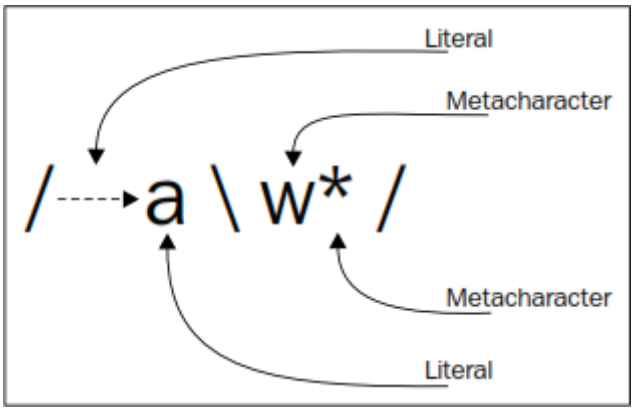
U računarstvu i informatici, regularni izraz ili "pravilni/ispravni izraz" - često i engleske skraćenice "regexp" ili "regex") jest niz znakova (zvanih *metaznakovi*, nadznakovi) koji opisujuju druge nizove znakova u skladu s određenim sintaksnim pravilima. Prvenstvena svrha regularnog izraza je opisivanje uzorka za pretraživnje nizova znakova. Koristeći ih, među ostalim primjenama, moguće je učiniti sljedeće:

- Provjeriti poštuje li unešeni niz znakova zadani uzorak, npr. može se s njim provjeriti je li vrijednost unesena u HTML obrazac valjana e-mail adresa.

- Pretražiti pojavljivanje uzorka u dijelu teksta; npr. provjeriti pojavljuje li se riječ "tlak" ili riječ "pritisak" u dokumentu, i to samo s jednim, a ne višestrukim, ispitivanjem.
- Izdvojiti određene dijelove teksta; npr. izvući poštanski broj iz neke adrese.
- Zamijeniti dijelove teksta; npr. promijeniti sve pojave riječi "tlak" ili "pritisak" s riječju "pressure".
- Podijeliti veći tekst u manje dijelove, npr. razdijeliti tekst na bilo kojem mjestu gdje se pojavljuje točka, zarez ili znak za novi red.

3.2.2 Sintaksa

Svaki stariji programer (onaj koji je upoznao prve operacijske sustave, npr. *DOS - disc operating systems*) bez sumnje se barem nekada koristio regularnim izrazom, iako možda nije znao da se o njemu radi. Često bi u konzoli operativnog sustava koristio zvjezdicu (*) ili upitnik (?) za pronalaženje nekih datoteka. Tako će, na primjer, uzorak koji sadrži upitnik kao što je "file?.xml" u naredbi "dir file?.xml" naći datoteke s nazivom file1.xml, file2.xml i file3.xml (ako dakako postoje u mapi/folderu u kojem naredba djeluje), ali neće ispisati datoteku s nazivom file99.xml, koja se tamo može nalaziti. Razlog je što metaznak '?' označuje niti jedan ili samo jedan znak, a ne dva ili više znakova, pa će se podudarati samo s onim imenima datoteka koje imaju niz znakova 'file' iza kojih slijedi samo jedan (bilo koji) znak ili da ga uopće nema i na koncu znakovi '.xml'.
U prethodnom izrazu, pojavljuju se dvije vrste komponente: **literal** ('file' i '.xml') i **metaznakovi** (? ili *). Slika (26) prikazuje primjer regularnog izraza u kojem se jasno vidi razlika između literala i metaznakova:



Slika 26: Regex - korištenje literala i metaznakova

Treba napomenuti kako su regularni izrazi koji se koriste u programskim jezicima mnogo moćniji od jednostavnih izraza koje obično pronalazimo u naredbenoj liniji temeljnog operativnog sustava, ali i jedni i drugi dijele istu definiciju: *"Regularni izraz je uzorak teksta koji se sastoji od običnih znakova (primjerice, slova **a** do **z** ili*

*brojeva **0** do **9**) i posebnih znakova poznatih kao metaznakova te opisuje one nizove znakova koji će mu odgovarati kada se primijeni u nekom tekstu."*

3.2.3 Literali

Literali su najjednostavniji oblici uzorka u regularnim izrazima, oni će jednostavno uspjeti svaki put kad se utvrdi da je literal pronađen. Ako se primijeni regularni izraz " *programa* " kao uzorak za traženje, u sljedećem tekstu (stringu), naći će se jedno podudaranje:

"Programiranje je umjetnost i umijeće u stvaranju programa za računala."

Međutim, dobit će se više rezultata, ako se na isti tekst primijeni regularni izraz "*dioda*":

"Svjetleća dioda je poluvodička dioda koja emitira svjetlo čim se kroz nju pusti električna energija."

Metaznakovi se često pojavljuju zajedno s literalima u istom regularnom izrazu. To može unijeti pomutnju, ako se ne uoče i razumiju pravila i razlikovanje običnih znakova od metaznakova. Na primjer, primijeni li se izraz "(simbol N)" za pretraživanje sljedećeg teksta, može se vidjeti kako zagrade nisu uključene u rezultat:

"SI jedinica za silu je newton (simbol N)."

To se događa zato što su zagrade *metaznakovi*, a oni imaju posebno značenje, posebnu interpretaciju u regularnim izrazima. Međutim, metaznakovi se mogu koristiti kao literali, kao obični znakovi, ali onda je u Python programskom kôdu potrebno:

- ispred metaznaka umetnuti lijevu kosu crtu (engl. *backslash*).
- ili koristiti metodu *re.escape* za izbjegavanje ne-alfanumeričkih znakova koji se mogu pojaviti u izrazu.

U regularnim izrazima postoji ukupno dvanaest *metaznakova* koje treba izbjegavati, ako se planiraju koristiti sa svojim doslovnim značenjenjem. To su:

- lijeva kosa crta "\"
- krović "^"
- točka "."
- znak dolara "$"
- povezni 'ILI' znak "—"
- upitnik "?"
- zvjezdica "*"
- znak plusa "+"
- otvorene zagrade "("

- zatvorene zagrade ")"
- otvorene uglaste zagrade "["
- zatvorene uglaste zagrade "]"
- otvorene vitičaste zagrade "{"
- zatvorene vitičaste zagrade "}"

U nekim slučajevima, sustavi regularnih izraza učiniti će sve kako bi razumjeli da li ti metaznakovi trebaju imati doslovno značenje, čak i ako nisu izbjegnuti; primjerice otvorena vitičasta zagrada "{" biti će tretirana kao metaznak, samo ako iza nje slijedi broj koji ukazuje na ponavljanje.

3.2.4 Klase znakova

Razredi ili *klase znakova*, (također poznate kao "*skupovi znakova*") omogućuju raspolaganje s jednim od definiranih znakova unutar skupa. Za definiranje razreda znakova, prvo trebamo napisati metaznak "[", a zatim sve znakove iz skupa koji se mogu pojaviti, te na koncu zatvoriti skup s metaznakom "]". Npr. definirani regularni izraz "filt[ae]r" će se podudarati s riječi "*filtar*", ali i "*filter*". Kao rezultat dobit će se sljedeće:

Električki filtar (ili filter) je elektronički sklop čija je funkcija da na određeni način promijeni karakteristiku frekvencijskog spektra ulaznog signala."

Može se također koristiti i raspon znakova. To se postiže uporabom simbola crtice (-) između dvaju (srodnih) znakova; npr. za izraz koji će se podudarati s bilo kojim malim slovom može se koristiti skup znakova [a-z]. Isto tako za podudaranje s bilo kojim jednoznamenkastim brojem može se definirati skup znakova [0-9]. Rasponi skupova znakova mogu se kombinirati tako da znak koji se obrađuje ima mogućnost zadovoljavati samo jedan od skupa raspona, a da pri tome nije potrebno nikakvo posebno odvajanje. Npr. ako se želi podudaranje s bilo kojim malim ili velikim alfanumeričkim znakom, može se koristiti izraz "[0-9a-zA-Z]", ili alternativni zapis "[0-9[a-z[A-Z]]]".

Postoji i još jedna mogućnost - *negacija raspona*. Možemo izokrenuti značenje skupa znakova postavljanjem metaznaka "^" odmah nakon metaznaka "[". Ako imamo skup znakova kao što je [0-9] koji označava bilo koji jednoznamenkasti broj, takav negirani skup [^0-9] podudarat će se sa svime što nije znamenka.

Upotreba znakovnih skupova vrlo brzo je pokazala kako su neki od njih vrlo korisni i umjesto da se stalno prepisuju, dodijeljena im je kratica, prečac. Trenutačno postoji velik broj predefiniranih razreda znakova što omogućuje izrazima koji ih koriste da budu puno čitljiviji i uredniji. Ovi znakovi nisu korisni samo kao dobro poznati prečaci za tipične skupove znakova, već također imaju i različita značenja u različitim kontekstima. Znakovna klasa "\w", koja odgovara bilo kojem alfanumeričkom znaku, podudarat će se s drugačijim skupom znakova ovisno o konfiguriranom lokalnom jeziku (eng. "*configured locale*") i "Unicode" podršci. Sljedeća tablica prikazuje predefinirane klase znakove podržane u Python-u:

Element	Opis (za regex s defaultnim flag-om)
.	Ovaj element odgovara bilo kojem znaku osim newline \n
\d	Ovaj element odgovara bilo kojoj decimalnoj znamenki; to je ekvivalent za klasu [0-9]
\D	Ovaj element odgovara bilo kojem znaku koji nije decimalna znamenka; to je ekvivalent za klasu [^0-9]
\s	Ovaj element odgovara bilo kojem znaku razmaka; to je ekvivalent za klasu [\t\n\r\f\v]
\S	Ovaj element odgovara bilo kojem znaku koji nije znak razmaka; to je ekvivalent za klasu [^\t\n\r\f\v]
\w	Ovaj element odgovara bilo kojem alfanumeričkom znaku; to je ekvivalent za klasu [a-zA-Z0-9_]
\W	Ovaj element odgovara bilo kojem znaku koji nije alfanumerički znak; to je ekvivalent za klasu [^a-zA-Z0-9_]

Tablica 2: Predefinirane klase znakova

3.2.5 Izbor

Izbor ili alternacija između dva ili više elemenata iz skupa regularnih izraza, postiže se s pomoću simbola "—". Npr. definiranje regularnog izraza kojim se želi podudaranje s riječi "*nizak*" ili riječi "*srednji*", željeni rezultat postiže se izrazom "nizak — srednji". Na isti se način, izraz može proširiti novim vrijednostima iz skupa koje proširuju regularni izraz, npr. "nizak — srednji — visok".

Prilikom korištenja alternacije u složenijim regularnim izrazima, moguće je alternaciju smjestiti unutar zagrada '( )' kako bi se izbor proveo samo unutar njih, a ne na cijelom izrazu. Npr. ako se napiše pogrešni izraz "*Pritisak: nizak — visok*" očekivat će se podudaranje s "*Pritisak: nizak*" ili "*Pritisak: visok*". Međutim, podudaranje je u ovom slučaju: "*Pritisak: nizak*" ili "*visok*", jer je alternacija primjenjena na cijeli regularni izraz, umjesto samo na *"nizak — visok*" dio. Ispravan regularni izraz treba, za takvu namjenu, biti: "*Pritisak: (nizak — visok)*".

3.2.6 Kvantifikatori

Kvantifikatori su brojevni mehanizmi s kojima se metaznakovi ili skupovi znakova mogu ponavljati. Osim na znakove i skupove znakova, kvantifikatori se mogu primjeniti i na grupe. Simbole kvantifikatora, te njihov opis, sadrži sljedeća tablica:

Neka se kao primjer uzme zadatak u kojemu je potrebno definirati regularni izraz koji će se podudarati s telefonskim brojem koji može bit zapisan u bilo kojem od sljedeća tri formata: *123-456-789*, *123 654 879*, ili *987654321*. Može se npr. konstruirati ovakav regularni izraz: "\d+[-\s]?\d+[-\s]?\d+". On će se podudarati s telefonskim brojem samo ako se na početku stringa nalazi jedna ili više znamenki, iza kojih može ali i ne mora slijediti crtica "-" ili prazan prostor, pa onda opet isto, te na koncu mora biti jedna ili više znamenki. Ovaj regularni izraz podudarao bi se sa svim gore navedenim formatima telefonskog broja, ali samo ako se ispred njega ne bi nalazili neki drugi brojevi. Oni bi pokvarili naše traženje. Zato je potrebno

Simbol	Ime	Kvantifikacija prethodnog znaka
?	Question mark	Uvjetno (0 ili 1 ponavljanje)
*	Asterisk	Nula ili više puta
+	Plus	Jednom ili više puta
{m,n}	Curly braces	Između n i m puta
{n}	Curly braces	Točno n puta
{,n}	Curly braces	Najviše n puta
{n,}	Curly braces	Minimalno n puta

Tablica 3: Kvantifikatori

finije podesiti regularni izraz, tako da krajnje lijevi skup znamenki može sadržavati maksimalno do tri znaka, dok ostatak brojevnih skupina treba sadržavati točno tri znamenke. Tako uređeni regularni izraz izgledao bi ovako "\d{1,3}[-\s]?\d{3}[-\s]?\d{3}". On će davati ispravno rješenje u bilo kojem slučaju.

3.3 IZVLAČENJE INFORMACIJE IZ TEKSTA

Da bi smo mogli izvlačiti informacije iz teksta moramo proći kroz sljedeće korake:

1. Rastaviti tekst na rečenice i riječi
2. Pridružiti riječi s njihovim gramatičkim oblicima iz **baze riječi**
3. Sastaviti uzorke i izraze
4. Pretražiti tekst uz pomoć uzoraka i izraza

3.3.1 Pridruživanje gramatičkih oblika riječima

Nakon što je načinjena lista riječi iz nekoga zadanog teksta, moramo za te riječi pronaći njihove tagove u **bazi riječi** i spremiti ih jednoznačnim pridruživanjem, stvarajući na taj način listu tagova. Riječ iz liste riječi i pripadni tag/ovi u listi tagova imaju isti indeks. Algoritam koji to radi prikazan je u Algoritmu 1.

```
1  class Greska(Exception):
2    pass
3
4  class Baza():
5    def __init__(self, path):
6      baza_connect=sqlite3.connect(path)
7      self.baza=baza_connect.cursor()
8      self.lista_stupaca=['rijec','tag','dod1','dod2','dod3']
9
10   def pretrazi(self, rijec, stupac):
11     if stupac in self.lista_stupaca:
12       trazi="SELECT "+stupac+" FROM Rijeci WHERE rijec='"+rijec+
           "';"
13       self.baza.execute(trazi)
```

```
14        pod=self.baza.fetchall()
15        if pod:
16          return pod[0][0]
17        else:
18          return None
19      else:
20        raise Greska('NE POSTOJI STUPAC '+stupac)
21
22
23  lista_obradeno=[ ]
24  rijeci=word_tokenize(recenica) #lista riječi iz rečenice
25
26  for r in rijeci: #za sve riječi u listi
27    podatak=baza.pretrazi(r.lower(),stupac) #pretraživanje baze za
          pojedinu riječ i njenog svojstva
28    if podatak is None: #ako riječ nije u bazi dodaj u listu #
29      lista_obradeno.append('#')
30    else:
31      lista_obradeno.append(podatak) #inače spremi tag
32
33  print lista_obradeno
```

Primjer 1: Algoritam za pridruživanje tagova iz baze podataka

S istim programom/algoritmom mogu se pridruživati i ostali tagovi /sintaksni ili semantički/ iz baze, ako su spremljeni za tu riječ.Uzmimo npr. rečenicu Uz našu kuću raste zeleni bršljan već treću godinu. za koju želimo naći tagove za pojedine riječi.

```
[[u'Uz', u'na\u0161u', u'ku\u0107u', u'raste', u'zeleni', u'br\u0161ljan', u've\u0107',
 u'tre\u0107u', u'godinu', u'.'], [u'saj', u'imjd0/imjv0/imjl0', u'izja0', u'imjv0/gp3j'
, u'izjg0/izjd0/izjv0/izjl0/izji0/pmjv0op/pmmn0op/pmmv0op/pmjn0np/pmja0np/pmjv0np
/pmmn0np/pmmv0np/gp3j/gm2j', u'imjn0/imja0', u'vus/vn', '#', u'izja0', u'q.']]
```

Slika 27: Prikaz tagova za danu rečenicu

Na slici 27 vidimo da neke riječi mogu imati više tagova odvojenih znakom „/", što znači da jedna riječ u nekom kontekstu može biti imenica, glagol ili pridjev ili da se isto piše u različitim padežima, licima, brojevima. Kao što možemo vidjeti, riječ raste ima tag „imjv0/gp3j" koji govori da riječ raste može biti prezentski oblik glagola rasti sa značenjem 'postajati većim ili višim...' ili vokativ imenice rast sa značenjem 'povećavanje jedinke...' Znak „# " govori nam da ta riječ ne postoji u bazi podataka, u ovom slučaju riječ treću nije spremljena u bazu podataka.

3.3.2 Sastavljanje uzoraka i izraza

Sastavljanje uzoraka i izraza temelji se na regularnim izrazima (eng. regular expression). To je način zadavanja općih oblika preko programskih kratica, kako bi

se iz teksta (niza znakova, eng. string) mogla izvući struktura, a ne samo znakovi. Tipičan je primjer dohvaćanje e-adrese iz nekog teksta. Iako adresa može biti gotovo beskonačno, sve one imaju istu strukturu (ne samo u znaku @ koji odjeljuje ime osobe od računalne domene). Moguće je s pomoću regularnih izraza (npr. `\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b`) napisati program koji će iz teksta izvlačiti bilo koju e-adresu. Pisanje regularnih izraza nije jednostavno, ali jednom dobro napisani služe zauvijek, osim poboljšavanja njihova filtriranja, nije ih potrebno mijenjati. Za potrebe našega programa kojim izvlačimo bilo koju informaciju iz dokumenata nužno je dobro poznavanje kako problematike (sintakse hrvatskoga jezika), tako i regularnih izraza za pisanje uzoraka kojim se slovni i gramatički oblici dohvaćaju. Štoviše, povezivanje uzoraka u sintaksno-semantičku cjelinu također se provodi regularnim izrazima (druga razina izvlačenja informacije), pa je njihova uporaba nezaobilazna. Naš program kao ulazne argumente prima niz uzoraka i izraza kojima se onda djeluje na pojedinačne rečenice iz teksta (dokumenta).

3.3.3 Uzorci

Uz pomoć (s regularnim izrazima) napisanih uzoraka pretražujemo tekst na osnovi slova i gramatičkih obilježja riječi. Moguće je birati neko od ponuđenih svojstava riječi ili uz dodatne argumente definirati kontekst (okoliš) riječi. Svaki uzorak sastoji se od triju osnovnih argumenata i deset mogućih dodatnih argumenata koji se stavljaju po volji korisnika. Opći oblik uzorka izgleda ovako:

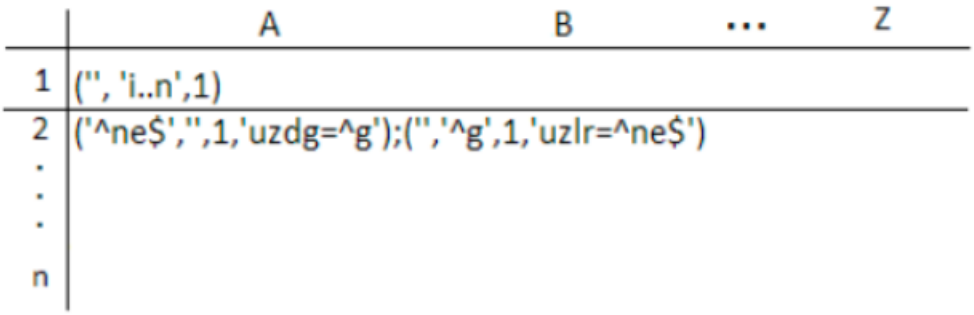
('Riječ', 'Gramatički oblik', Logika, 'Dodatni argumenti').

Prvi argument „Riječ" odnosi se na regularni izraz kojim pretražujemo riječi u nekom tekstu na temelju slovnih znakova. Drugi argument „Gramatički oblik" omogućuje nam da pretražujemo riječi po gramatičkoj osnovi, na primjer, da tražimo imenicu muškoga roda ili glagol u prezentu jednine ili zamjenicu itd. Treći argument „Logika" može imati vrijednost 1 ili 0, što znači da se uzima u obzir traženje i po slovnom i po gramatičkom obliku (1) ili bilo kojem od njih (0). Zadnji argument može imati više svojstava koja se vide u tablici 4.:

Tablica 4: Opis dodatnih svojstava uzoraka

Svojstva	Značenje
lg	lijevo od nađene riječi traži se gramatika
lr	lijevo od nađene riječi traži se riječ
dg	desno od nađene riječi traži se gramatika
dr	desno od nađene riječi traži se riječ
uzlg	neposredno lijevo uz nađenu riječ traži se gramatika
uzlr	neposredno lijevo uz nađenu riječ traži se riječ
uzdg	neposredno desno uz nađenu riječ traži se gramatika
uzdr	neposredno desno uz nađenu riječ traži se riječ
imag	traži se da u rečenici bude riječ sa zadanom gramatikom
imar	traži se da u rečenici bude tražena riječ

Slika 28. pokazuje kako se pravilno unose uzorci. U njima se mogu koristiti sve oznake regularnih izraza kako bi se ispravno definiralo što se želi naći u zadanom tekstu. Uzorci se odvajaju znakom ';' i navode u n-redaka. Svaki redak može imati najviše 26 uzoraka (označenih slovima engleske abecede radi lakše daljnje obrade u programu kada se pozivaju izrazi).



Slika 28: Prikaz zadavanja uzoraka

3.3.4 Izrazi

Sintaksno-semantički izraz sastoji se općenito od niza riječi (dohvaćenih na temelju njihova slovnog ili gramatičkog obilježja). Izraz se dohvaća iz rečenice na temelju regularnoga izraza kojim se povezuju gore opisani uzorci. Uzorak je jednoznačno opisan svojim položajem (u retku i stupcu), pa će 1A značiti prvi (A) uzorak iz prvoga retka, 3B će značiti drugi (B) uzorak iz trećega retka, 4C će značiti treći uzorak iz četvrtoga retka i slično.

Slijedi faza povezivanja uzoraka u izraze. Što će naš izraz tražiti, u ovom je času nevažno. Može se na primjer pokazati traženje službe riječi u rečenici (subjekta, objekta, predikata, atributa sročnog i nesročnog, priložnih oznaka i slično). Svaki od izraza može imati ime ili kraticu, da bi se kod rezultata jasno razabralo što je program uspio pronaći. Izrazi se sastavljaju tako da se prvo napiše naziv ili ime izraza, zatim stavimo znak dvotočke (':') iza čega slijedi kombinacija uzoraka označenih njihovim kraticama (1A, 3B i sl.), kako pokazuje slika 29.

```
S: 1A 1A
PG: 2A
PI: 3A
AS: 4A
```

Slika 29: Zadavanje izraza

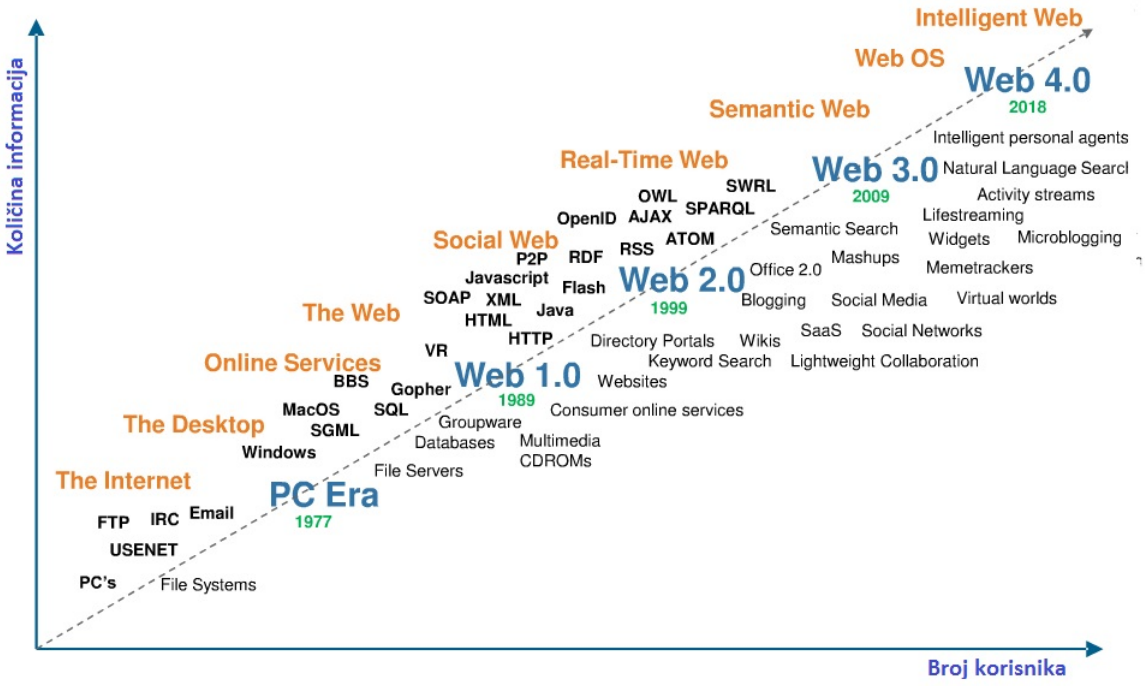
3.4 Semantički Web i povezani podaci

Semantički web predstavlja sljedeću evolucijsku stepenicu u razvoju World Wide Web-a (WWW). U proteklom razdoblju WWW je prošao ogroman put (od alata za razmjenu i vezivanje dokumenata unutar jednog istraživačkog centra do najpopularnije i najkorištenije usluge na Internetu). Utemeljen na nizu široko prihvaćenih standarda dostigao je neslućene razmjere. Ti vrlo važni standardi su *TCP/IP protoko*l na koji se nadovezuje HTTP (Hypertext transfer protocol) i jezik za opis stranice HTML (*Hypertext markup language*). Povijesno gledano, možemo reći kako je web trenutno u svojoj drugoj generacijskoj dobi.

3.4.1 Resurs, URI (Uniform Resource Identifier)

Uniform Resourse Identifier (URI) je jednoznačno određena adresa nekog izvora informacije (odnosno dokumenta) na web mreži, ili identifikator nekog izvora. Svrha korištenja URI-a je pristup (ili interakcija sa) izvorima informacija na webu korištenjem različitih protokola. Sintaksa (opći oblik pisanja) URI-a:

- URI shema (npr. http, ftp, mailto, URN, itd.)
- dvotočka
- dio koji označava određenu shemu koja se poziva



Slika 30: Razvitak globalne mreže - interneta

```
http://dbpedia.org/resource/Mechatronics
```

Valja napomenuti kako ovaj link ujedno predstavlja i URL (Uniform Resource Locator), jer klikom na njega dolazimo do stranice sa informacijama o određenom podatku, u ovom slučaju o mehatronici.

3.4.2 NameSpace (NS), Qname

U XML-u, imena elemenata definirana su od strane programera što često rezultira sukobom pri pokušaju miješanja XML dokumenata iz različitih XML aplikacija. Također se kod zapisivanja u tekstualne datoteke često pojavljuje nepreglednost zbog mnogo ponavljanja prvog dijela URI-a, odnosno domene. Kako bi izbjegli to ponavljanje i probleme sa sukobljavanjem te doprinijeli kompaktnosti i preglednosti zapisa u datotekama, uvodi se NameSpace. Uzmimo za primjer slijedeće URI-e:

```
http://dbpedia.org/resource/Mechatronics
http://dbpedia.org/resource/Robotics
http://dbpedia.org/resource/Mechanical_engineering
```

U navedenim zapisima možemo primjetiti zajednički dio http://dbpedia.org/resource/ koji predstavlja domenu, odnosno NameSpace, te različit dio koji zajedno s domenom čini URI. Kako bismo zamijenili NS sa prefiksom koristimo slijedeću sintaksu:

```
xmlns:db = http://dbpedia.org/resource/
```

Ovime smo definirali novo proizvoljno ime *db* za naš prefix i dodijelili mu NameSpace *http://dbpedia.org/resource/*. Sada možemo napisati gornje URI-je u skraćenom obliku, tj. pomoću prefiksa:

```
db:Mechatronics
db:Robotics
db:Mechanical_engineering
```

Gornji izrazi prestavljaju Qname koji predstavlja kompaktniju verziju URI-a.

3.4.3 XML (eXtensible Markup Language)

XML je označiteljski jezik za opisivanje dokumenata i podataka. Ideja je bila stvoriti jedan jezik koji će biti jednostavno čitljiv i ljudima i računalnim programima. Princip realizacije je vrlo jednostavan: odgovarajući sadržaj treba se uokviriti odgovarajućim oznakama koje ga opisuju i imaju poznato, ili lako shvatljivo značenje.

Pod izrazom "dokumenti i podaci" podrazumijevamo tekstne dokumente ili pak skupove podataka kakvi se obično pohranjuju u baze podataka. Promatrajući XML kao tekstni format, može se ustvrditi da je on potpuno neovisan o računalnoj platformi na kojoj se nalazi. Također je potpuno neovisan o operacijskom sustavu kojeg koristimo – XML je otvoreni standard čija je specifikacija javna i dostupna svima, tj. za njegovu standardizaciju brine se W3C (World Wide Web Consortium). Za korištenje XML-a nisu potrebne nikakve licence.

Oznake koje koristimo za opisivanje podataka nisu unaprijed definirani. XML standard jedino opisuje minimalni skup pravila koja dokument mora zadovoljavati. Korisnici XML-a moraju sami definirati dozvoljene oznake za označavanje. Danas je XML jezik vrlo raširen i koristi se za različite namjene: odvajanje podataka od prezentacije, razmjenu podataka, pohranu podataka, povećavanje dostupnosti podataka i izradu novih specijaliziranih jezika za označavanje.

XML je strukturiran kao hijerarhija oznaka/tagova. Postoje *root*, *child* i *parent* tagovi. Slijedi primjer XML zapisa:

```
<?xml version="1.0" encoding="UTF-8" ?>
<student jmbag="0152195996">
  <ime>Marko</ime>
  <prezime>Marković</prezime>
  <institucija>
    <sveučilište>Sveučilište u Zagrebu</sveučilište>
      <fakultet id="1234">
        <ime>Fakultet strojarstva i brodogranje</ime>
        <adresa>
          <ulica>Ivana Lucića</ulica>
          <broj>5</broj>
          <pbr>HR-10000</pbr>
```

```
          <grad>Zagreb</grad>
        </adresa>
      </fakultet>
  </institucija>
</student>
```

U gornjem primjeru uočavamo da se XML dokument sastoji od dva dijela. Prvi dio je prolog ili zaglavlje.

```
<?xml version="1.0" encoding="UTF-8" ?>
```

U njemu se navode podatci koji opisuju XML dokument kao što su verzija XML preporuke prema čijim pravilima je dokument napravljen i kodna stranica. Ako se ne navede ispravna kodna stranica, programi koji barataju s XML dokumentom kada naiđu na nestandardni znak (npr. naše slovo "č") javit će grešku.

Drugi dio je sadržaj dokumenta u kojem se nalazi korisni sadržaj omeđen XML oznakama.

```
<student>
  <ime>Marko</ime>
    ...
</student>
```

Svaki XML dokument mora imati jedan korjenski ili root element koji uokviruje kompletan sadržaj. Taj element opisuje XML dokument i npr. kaže "ovaj XML dokument predstavlja podatke o studentu". Unutar korjenskog elementa ugniježđeni su svi ostali.

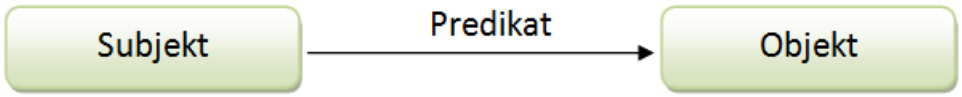
3.4.4 RDF (Resource Description Framework)

RDF je sintaksno neovisan, apstraktni model koji određuje standard o meta podacima (podaci o podacima) koji služe za opis resursa na webu. Kada u ovom obliku govorimo o meta podacima, njih isto tako uzimamo u najširem mogućem opsegu, ne ograničajući se ne prvotnu namjenu. RDF je standard kojeg je uspostavio i dalje razvija W3C te ga predlaže kao osnovu semantičkog weba, na temelju kojeg se izgrađuju svi ostali jezici semantičkog weba.

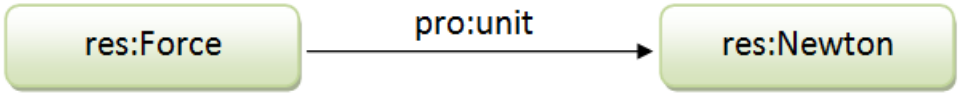
RDF se temelji na prijašnjim radovima koji su rezultirali modelima za prikaz imenovanih svojstava i njihovih vrijednosti. Temeljna konstrukcija povezanih podataka je RDF trojac oblika:

Kako se vidi (slika 31), RDF trojac se sastoji od *subjekta*, *predikata* i *objekta*. Isto kao što čovjek definira vezu između subjekta i objekta u rečenici preko predikata, tako je i u RDF trojcu definirana veza između dva web resursa (ili resursa i doslovne vrijednosti) predstavljena pomoću URI-a.
Primjer jednog RDF trojca koji daje vezu između *sile* i njezine *mjerne jedinice* prikazuje slika 32.

Iz gore navednog primjećuje se da se RDF data model sastoji od nekoliko članova:



Slika 31: Općeniti oblik RDF trojca



Slika 32: Primjer RDF trojca

1. Resurs (resource) može predstavljati bilo što, sve što posjeduje URI je resurs. To može biti HTML ili XML dokument, ili skup bilo kojih od navedenih dokumenata, cijela web stranica ili jedna osoba ... *URI* specifikacija pruža dovoljno proširivosti da možemo reći kako bilo što može posjedovati jedinstveni identifikator.

2. Atributi, još poznati pod drugim imenom kao svojstva (properties), predstavljaju specifičan aspekt nekog resursa. Svaki atribut posjeduje vlastito značenje koje dopušta određeni opseg vrijednosti ili može biti povezan samo s određenim resursima.

3. Vrijednosti koja može biti ili literal ili neki drugi resurs ili bilo koji drugi primitiv (atom) definiran XML-om.

4. Sama tvrdnja (statement) je spomenuta trojka koja se sastoji od trojca: *subjekta, predikata* i *objekta.*

3.4.5 RDF/XML

RDF/XML je sintaksa, definirana od strane W3C-a, za izražavanje RDF grafa kao XML dokumenta. RDF graf može se smatrati skupom staza u obliku čvorova: predikatna grana, čvor, predikatna grana, čvor, predikatna grana, ... čvor, koji pokrivaju cijeli graf. U RDF/XML-u one su pretvorene u sekvence elemenata unutar elemenata koji se izmjenjuju između elemenata za čvorove i predikatnih grana. To se naziva niz čvor/grana poveznica. Čvor na početku sekvence prelazi u krajnji vanjski (*outermost*) element, sljedeća predikatna grana prelazi u *child* element, i tako dalje.

Slijedi primjer RDF/XML zapisa:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:pro="http://dbpedia.org/property/">

<rdf:Description rdf:about="http://dbpedia.org/resource/Force">
  <pro:unit>Newton</pro:unit>
```

```
  <pro:symbol>F</pro:symbol>
  ...
</rdf:Description>

<rdf:Description rdf:about="http://dbpedia.org/resource/Power">
  <pro:unit rdf:resource="http://dbpedia.org/resource/Watt"/>
  <pro:symbol>P</pro:symbol>
  ...
</rdf:Description>
.
.
</rdf:RDF>
```

Kao što je vidljivo, na početku je definiran glavni tag *rdf:RDF*, te također da se radi o XML datoteci. Unutar glavnog čvora nalaze se RDF trojci, dok se u njegovim atributima mogu definirati prefiksi NS-a.

RDF trojac je strukturiran tako da je najprije definiran čvor koji predstavlja subjekt, s tagom *rdf:Description* koji označuje opis nečega, a kao rdf:about atribut tog taga, upisuje se subjekt pod vrijednost atributa unutar navodnika. Zatim unutar njega slijedi tag koji predstavlja predikat, npr. tag *pro:unit*, a on kao vrijednost atributa sadrži URI objekta *rdf:resource="http://dbpedia.org/resource/Watt* ili se između otvorenog i zavorenog dijela taga nalazi doslovna vrijednost objekta.

3.4.6 Turtle (Terse RDF Triple Language)

Turtle je format za izražavanje podataka u opisu RDF modela podataka sa sintaksom sličnom SPARQL. Zapravo, on nam pruža način za grupiranje tri URI-ja kako bi napravili trojku, te također pruža načine za skraćivanje takve informacije, npr. izlučivanje zajedničkih dijelova URI-ja pomoću prefiksa. Slijedi primjer definiranja prefiksa i najjednostavniji zapis jedne trojke u turtle formatu:

```
@prefix res:<http://dbpedia.org/resource/>
@prefix pro:<http://dbpedia.org/property/>
@prefix ont:<http://dbpedia.org/ontology/>

res:Force pro:unit res:Newton.
```

Često će se isti subjekt referencirati na veći broj predikata. Tada kreiranje *predicateObjectList-e* odgovara nizu predikata i objekata odijeljenih znakom točka-zarez ";" koji se nalaze na kraju svake trojke, dok se nakon svih ponovljenih nalazi točka. Taj zapis izražava niz RDF trojki s istim subjektom, te svaki predikat i objekt dodijeljen pojedinoj trojki. Sljedeća dva primjera predstavljaju ekvivalentan način zapisa trojki o sili:

```
res:Force pro:unit res:Newton;
          pro:isVector "True";
          pro:symbol "F".
```

```
res:Force pro:unit res:Newton.
res:Force pro:isVector "True".
res:Force pro:symbol "F".
```

Kao i sa predikatima, često se i objekti ponavljaju s istim subjektima i predikatima. Tada stvaranje *objectList-e* odgovara nizu objekta odvojenih znakom zareza "," koji se nalazi na kraju svake trojke, dok se na kraju svih ponovljenih isto nalazi točka. Slijedi primjer zapisa trojke sa zajedničkim subjektom i prekidatom:

```
res:Tensile_strength pro:unit res:Pascal_(unit),
                                "Mpa".
```

3.4.7 JSON-LD (JavaScript Object Notation for Linking Data)

JSON-LD je lagana sintaksa za serijalizaciju povezanih podataka u JSON-u. Njegov dizajn omogućuje postojećem JSON-u da se tumači kao povezani podatak s minimalnim izmjenama. Uz sve značajke koje pruža JSON, JSON-LD uvodi:

- univerzalni identifikatorski mehanizam za JSON objekte putem korištenja URI-ja.
- način da se ukloni neizvjesnost značenja ključeva koji se dijele između različitih JSON dokumenata njihovim mapiranjem u URI-je pomoću @context.
- mehanizam u kojem se vrijednost u JSON objektu može odnositi na JSON objekt na drugom mjestu na webu.
- mogućnost da bilježi stringove napisane u različitim jezicima.
- način za povezivanje vrste podataka s vrijednostima kao što su datum i vrijeme.
- Sposobnost izražavanja jednog ili više usmjerenih grafova, kao što su društvene mreže, u jednom dokumentu.

Općenito govoreći, model podataka korišten za JSON-LD je označen, usmjereni graf. Taj graf sadrži čvorove, koji su međusobno povezani. Čvor je obično podatak kao što je string, broj, datum ili vrijeme, ili URI. Postoji također i posebna klasa čvora koja se naziva prazan čvor, a obično se koristi za izražavanje podataka koji nema globalni identifikator kao što je URI. Prazni čvorovi su identificirani pomoću identifikatora praznog čvora. Ovaj jednostavan model podataka je nevjerojatno fleksibilan i moćan, te sposoban za modeliranje gotovo bilo koje vrste podataka.

3.5 Ontologijski jezik

U računalnoj znanosti i umjetnoj inteligenciji, ontologijski jezici su formalni jezici koji se koriste za izgradnju ontologija. Oni omogućuju kodiranje znanja o specifičnim domenama i često uključuju pravila rasuđivanja koja podržavaju obradu tog znanja. Jezici ontologije su obično deklaratitvni jezici, gotovo uvijek su generalizacije *frame* jezika, a obično se temelje na logici prvog reda ili opisnoj logici.
Ontologija nam služi za točno definiranje veza imeđu resursa, dok nam za povezivanje resursa služi RDF. Kako bi definirali semantičke odnose između resursa potrebna nam je RDF Schema.

3.5.1 RDFS (RDF Schema)

RDFS je semantičko proširenje RDF-a, tj. skup klasa s određenim značajkama RDF proširivog jezika koji pruža osnovne elemente za opis ontologija, namijenjenih za strukturiranje RDF resursa. Ovi resursi mogu se spremiti u trojke kako bi se do njih došlo pomoću SPARQL *query* jezika. Mnoge RDFS komponente uključene su u više izražajan Web Ontology Language (OWL).
RDFS konstrukti su RDFS klase, te pridružena svojstva izgrađena na ograničenom RDF rječniku.

Klase:

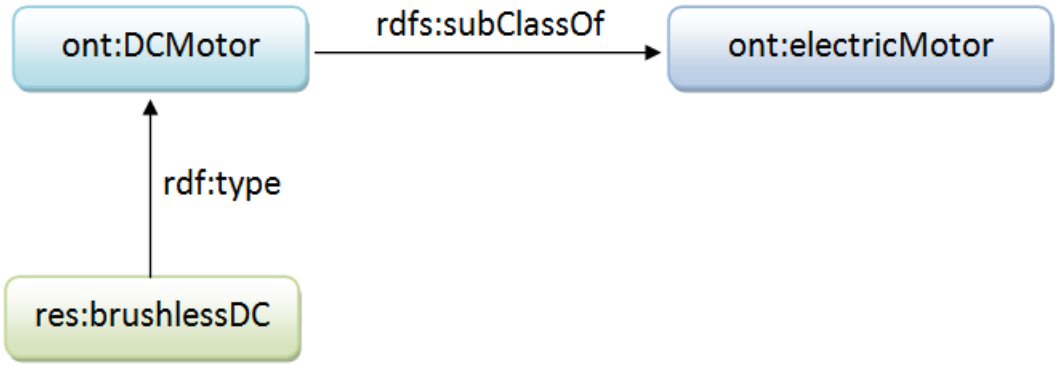
- **rdfs:Resource** je klasa svega. Sve stvari koje opisuje RDF su resursi.
- **rdfs:Class** - proglašava resurs kao klasu za druge resurse.
- **rdfs:Literal** - doslovne vrijednosti kao što su stringovi i brojevi, npr. vrijednosti nekog svojstva.
- **rdfs:Datatype** - klasa vrste podataka. rdfs:Datatype je ujedno instanca i podklasa od rdfs:Class. Svaka instanca od rdfs:Datatype je i podklasa od rdfs:Literal.
- **rdf:XMLLiteral** - klasa XML doslovne vrijednosti.
- **rdf:Property** - klasa svojstava.

Tipičan primjer *rdfs:Class* je *ont:DCMotor* u primjeru prikazanom na slici 33. Instanca od *ont:DCMotor* klase je resurs *res:brushlessDC* koji je povezan s njom pomoću *rdf:type* svojstva.

Ovaj zapis govorida je brushlessDC ujedno i električni motor, iako nisu direktno povezani.

Svojstva:

Svojstva su instance klase *rdf:Property* i opisuju odnos između resursa subjekta i resursa objekta, što znači da su kao takva ujedno i predikati.



Slika 33: Hijerarhijska struktura RDF trojca

- **rdfs:domain** - je instanca od rdf:Property koja se koristi za tvrdnju da su vrijednosti svojstva instance od jedne ili više klasa.
- **rdfs:range** - je instanca od rdf:Property koja se koristi za navod da bilo koji resurs koji ima dano svojstvo jest instanca jedne ili više klasa.
- **rdf:type** - je svojstvo koje se koristi za tvrdnju da je određeni resurs instanca neke klase.
- **rdfs:isSubClassOf** - omogućuje definiranje hijerarhije klasa.
- **rdf:subPropertyOf** - je instanca od rdf:Property koja se koristi za tvrdnju da su svi resursi povezani s jednim svojstvom također povezani i s nekim drugim.
- **rdf:label** - je instanca od rdf:Property koja se može koristiti za pružanje ljudski čitljive verzije imena resursa.
- **rdf:comment** - je instanca od rdf:Property koja se može koristiti za pružanje ljudski čitljivog opisa resursa.

3.5.2 OWL (Web Ontology Language)

OWL je ontološki jezik namijenjen opisu pojmova i odnosa između pojmova. Izrastao je na ideji semantičkog weba u kojemu informacija za web ima eksplicitno značenje, a ne samo oznake za prikaz. Implementacija ideje semantičkog weba ostvarena je kroz višerazinsku arhitekturu u kojoj su prve razine ostvarene s XML-om i RDFS-om. OWL predstavlja korak dalje, s ciljem da "formalno opiše značenje terminologije upotrebljene u web dokumentima". Teoretsku osnovu za OWL pružaju opisne logike - formalni jezici za predstavljanje znanja.

OWL je izgrađen na RDF-u i RDF Schemi te koristi RDF-ovu XML-baziranu sintaksu ali dodaje više vokabulara za opisivanje svojstava i klasa: između ostalog,

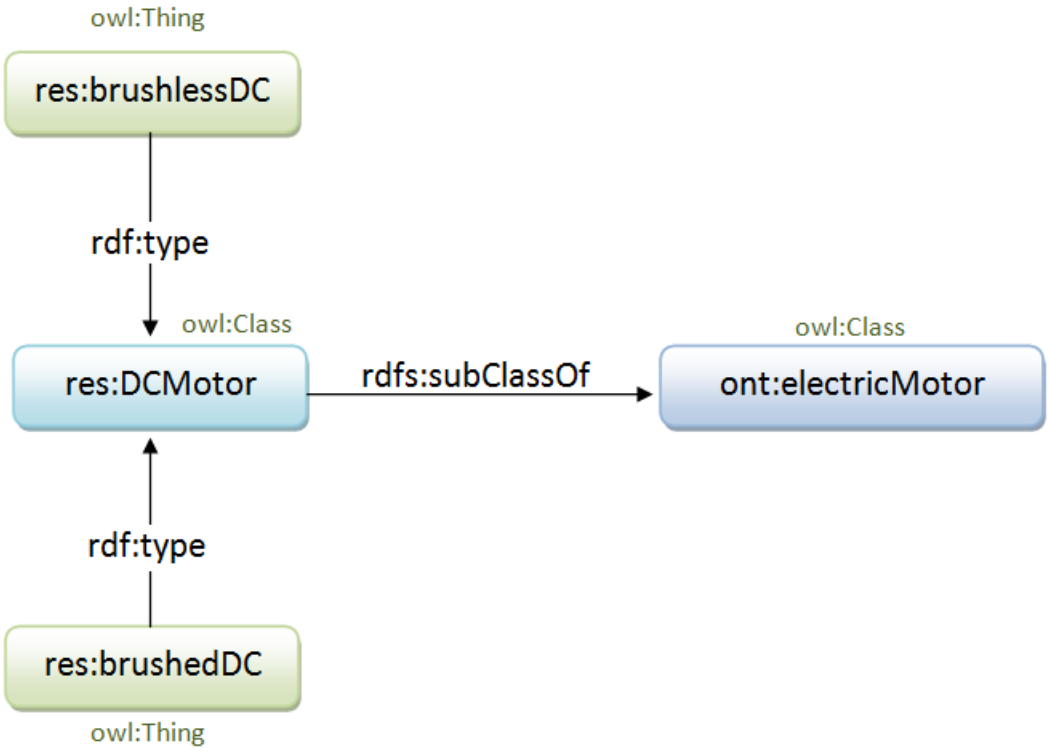
odnose između klasa (npr. "disjointness"), kardinalitet (npr. "točno jedan"), jednakosti, bogatije tipizacija svojstava, karakteristike svojstava (npr. "simetrija"), te numerirane klase. Globalni *NameSpace* OWL rječnika je *http://www.w3.org/2002/07/owl#*, a uobičajeni prefiks koji se dodjeljuje je *owl*.

Kako definirati da je *DCMotor* klasa te ujedno i podklasa od *electricMotor* u OWL dokumentu prikazuje sljedeći primjer:

```
<rdf:RDF xmlns:owl ="http://www.w3.org/2002/07/owl#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

  <owl:Class rdf:about="DCMotor">
    <rdfs:subClassOf rdf:resource="htpp://www.dbpedia.org/resource/
    electricMotor" />
    ...
  </owl:Class>

</rdf:RDF>
```



Slika 34: Prikaz grafa OWL zapisa

Osim definiranja klase mogu se opisati i njeni članovi, što su pojedinci/individue u domeni stvari (things). Slijedeći primjer definira pojedince *brushlessMotor* i *brushedMotor* kao tipove električnog motora *DCMotor*:

```
<owl:Thing rdf:ID="http://www.dbpedia.org/resource/brushlessMotor"/>
<owl:Thing rdf:ID="http://www.dbpedia.org/resource/brushedMotor"/>

<owl:Thing rdf:about="http://www.dbpedia.org/resource/brushlessMotor"/>
   <rdf:type rdf:resource="http://www.dbpedia.org/resource/DCMotor"/>
</owl:Thing>

<owl:Thing rdf:about="http://www.dbpedia.org/resource/brushedMotor"/>
   <rdf:type rdf:resource="http://www.dbpedia.org/resource/DCMotor"/>
</owl:Thing>
```

Najnoviji standard ontologijskog jezika je OWL2 koji je kompatibilan sa prošlom verzijom OWL-a i preporučen od strane W3C-a od 2009. godine.

3.6 Virtuoso, univerzalni server

Virtuoso je višenamjenski, više-protokolni mrežni poslužitelj kojeg je napravio *OpenLink Software*, a uključuje upravljanje podacima kao što su SQL, RDF, XML, Free Text, web aplikacije, LOD povezani podaci i ostalo.

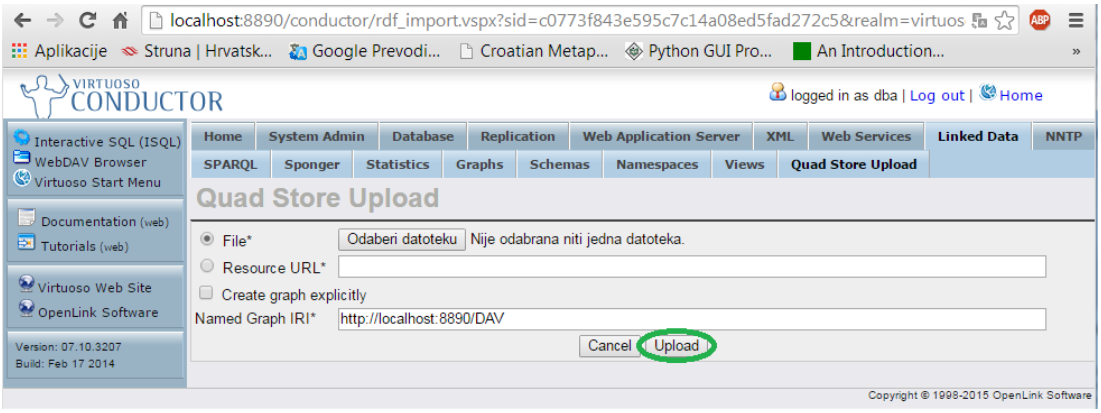
Virtuoso također sadrži ugrađenu mrežnu aplikaciju koja se naziva *Virtuoso Conductor* (slika 35), a pruža sučelje za upravljanje bazama podataka koje uobičajeno izvodi administrator (DBA, *engl. Database Administrator*):

Slika 35: *Virtuoso Conductor* sučelje

Unutar opsega ovog rada biti će korišten i objašnjen samo dio *Virtuosa* koji se tiče ubacivanja povezanih podataka u *triplestore* bazu podataka, te njihovog pretraživanja uz pomoć naredbi iz SPARQL editora.

3.6.1 Ubacivanje trojaca u *triplestore* bazu

Ubacivanje datoteka koje sadrže LOD trojce, tj. datoteka koje imaju ekstenziju bilo .rdf, .xml, .owl, te ostale koje zadovoljavaju jedan od formata standardnih načina serijalizacije kao što su RDF/XML, N-triples i Turtle, u *triplestore* bazu vrši se unutar *Linked Data* ⇒ *Quad Store Upload* sučelja pritiskom na gumb "Upload" (slika 36).



Slika 36: Prikaz Virtuoso Conductor, Quad Store Upload sučelja

3.6.2 SPARQL

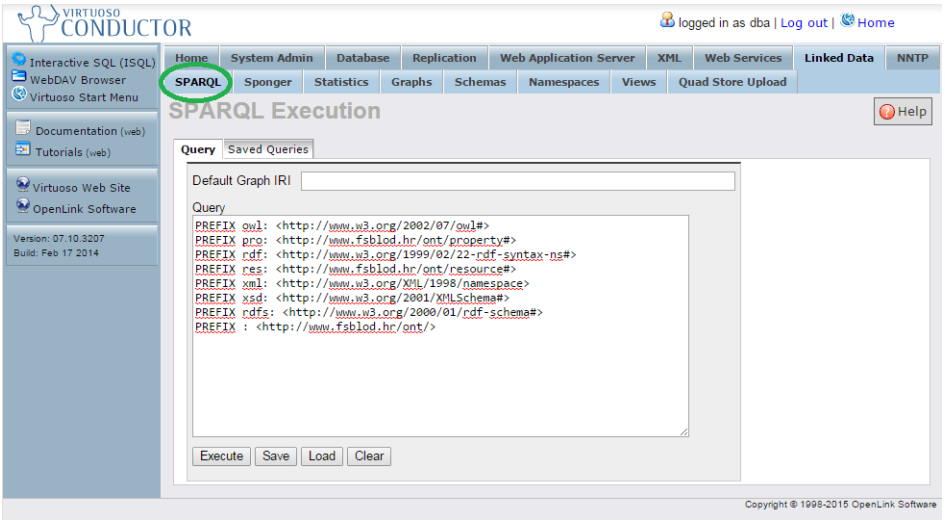
Nakon što je objašnjena osnovna ideja semantičkog weba i povezanih podataka, te njihova obilježja i način pohrane, potrebno je te podatke i isčitati iz baze. W3C je postavio SPARQL jezik kao standard za pretragu podataka u RDF modelu.

Pretraživanje baze vrši se unutar *Virtuoso-vog* SPARQL editora kojeg prikazuje slika 37. *SPARQL* je za semantički web, odnosno RDF baze podataka, ono što je *SQL* jezik za relacijske baze podataka. Jezik za pretragu podataka u RDF modelu, koji nije ograničen samo na baze podataka. S njim je moguće pretraživati bilo koji format zapisanih RDF tojaca, poput XML datoteka.

Kod upita putem SPARQL-a postoje varijable. One su namijenjene kako bi u njih spremali rezultat pretrage ili ih koristili samo privremeno kako bi filtrirali rezultate. Varijable se označavaju upitnikom ispred imena i nije ih potrebno deklarirati unaprijed.

Slično kao i u SQL-u, upit postavljamo sa kombinacijom naredbi SELECT i WHERE.

U SELECT dijelu upisujemo varijable koje želimo pronaći, odnosno varijable koje želimo zadržati kao rezultat. Važno je napomenuti kako u nastavku možemo deklarirati i dodatne varijable koje će nam pmoći u pretrazi, ali ako nisu spomenute u SELECT dijelu, one neće biti u rezultatima. Ako želimo zadržati sve varijable korištene u pretrazi, tada poput SQL-a možemo upisati zvjezdicu "*" umjesto varijabli.



Slika 37: Virtuoso Conductor, SPARQL editor

Query

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX pro: <http://www.fsblod.hr/ont/property#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX res: <http://www.fsblod.hr/ont/resource#>
PREFIX xml: <http://www.w3.org/XML/1998/namespace>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.fsblod.hr/ont/>

SELECT DISTINCT ?pojam ?definicija ?istoznacnice WHERE {
?individua rdf:type owl:NamedIndividual ;
           rdfs:label ?pojam ;
           pro:definicija ?definicija .
OPTIONAL { ?individua pro:istoznacnice ?istoznacnice . }
} ORDER BY ?pojam
```

Execute | Save | Load | Clear

pojam	definicija	istoznacnice
"A-pužni vijak"	"pužni vijak kojemu je presječna krivulja boka zuba i ravnine okomite na os Arhimedova spirala"	"A-puž , ZA-puž, ZA-pužni vijak"
"Carnotov stroj"	"idealni stroj koji bi radio na načelu Carnotova termodinamičkoga kružnog procesa"	
"Clarkova elektroda"	"elektroda koja omogućuje ampermetrijsko mjerenje aktiviteta otopljenoga kisika"	
"E-pužni vijak"	"pužni vijak oblikom istovjetan cilindričnomu zupčaniku s evolventnim kosim zubima velikoga kutnog nagiba boka zuba"	"E-puž , ZE-puž, ZE-pužni vijak"
"bakrova/bakrosulfatna referencijska elektroda"	"referencijska bakrena elektroda uronjena u zasićenu otopinu bakrova sulfata"	"bakar / bakar sulfat referentna elektroda"
"cijevna dioda"	"cijevni evakuirani elektronički element strujnoga kruga u kojemu elektroni izlaze iz ugrijane katode, a skupljaju se na anodi"	"vakuumska dioda"
"cilindrična matica s urezom"	"cilindrična matica koja s jedne čelne strane ima utor pravokutnoga poprečnog presjeka duž cijeloga promjera"	
"cilindrični pužni vijak"	"cilindrični zupčanik s jednim kosim zubom ili s više kosih zuba"	"cilindrični puž"
"dioda"	"elektronički element koji propušta struju samo u jednome smjeru"	
"dosjedni vijak"	"vijak kojemu je dio tijela u dosjedu s provrtom u kojemu se nalazi"	

Slika 38: SPARQL upit

WHERE predstavlja dio upita u kojeg se upisuju trojci s varijablama. Oni se pišu između vitičastih zagrada, "{" i "}", te kao u Turtle sintaksi, elementi trojaca, odnosno subjekt, objekt i predikat su odvojeni praznim mjestom, a na kraju linije upisuje se točka. URI se stavljaju između izlomljenih zagrada "<" i ">", doslovne vrijednosti u navodnike, a Qname se piše bez ikakvih znakova. Također, koriste

se točka-zarez ";" za ponavljanje subjekta (sa različitim predikatom i objektom) ili zarez "," za ponavljanje subjekta i predikata (sa različitim objektom). Umjesto nepoznatog elementa trojca upisuje se varijabla.

Također, slično kao i *Turtle* format, moguće je odrediti prefikse koji omogućuju kompaktniji prikaz trojaca. Određeni su prefiksi *res* i *pro*, kojima glavnu domenu predstavlja URI *http://www.fsblod.hr/ont/*, te ostali, nužni za definiranje veza i odnosa između resursa kao što su *owl. rdf*, *rdfs*, itd.

U sljedećem primjeru prikazan je SPARQL upit prema bazi podataka koja sadrži pojmove, zajedno s njihovim svojstvima, iz tehničkog rječnika.

Ukoliko se iz baze žele preuzeti svi pojmovi, njihove definicije i grane djelatnosti kojoj pripadaju, potrebno je napisati sljedeću SPARQL naredbu, prema slici 38.

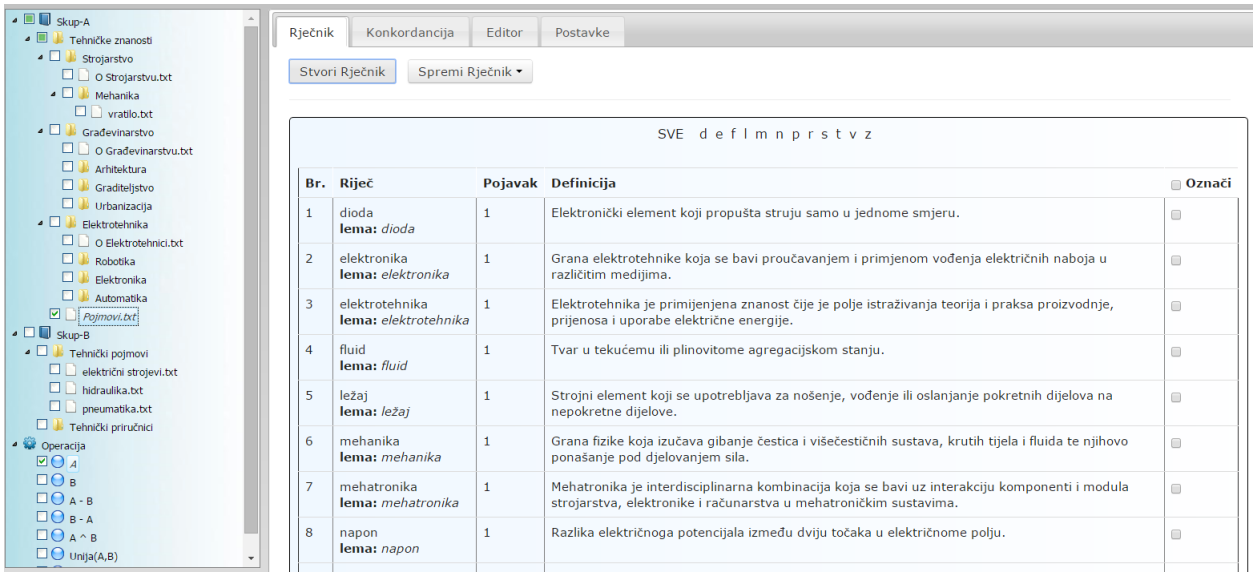
U SELECT dijelu su također navedene samo varijable ?pojam, ?definicija i ?grana jer nije potrebno ispisivati ostale (?individua) koje su služile samo za pretragu i sadrže URI-e. U prvom RDF trojcu upita u varijablu ?individua spremljene su sve vrijednosti koje s *owl:NamedIndividual* veže ontologija *rdf:type*, odnosno URI vrijednosti koje predstavljaju pojmove. Zatim su u drugom redu iz tih filtriranih pojmova izvučene njihove doslovne vrijednosti (*rdfs:label*), koje su spremljene u varijablu ?pojam. U trećem i četvrtom retku na isti način izvučena je definicija i grana (djelatnosti) pojedinih pojmova preko odgovarajuće ontologije (*pro:definicija i pro:grana*), te su također spremljene u varijable ?definicija i ?grana kako bi ih se moglo ispisati.

4 PROGRAMSKO GRAFIČKO SUČELJE

Programskom grafičko sučelje, kao što je bilo prikazano u drugom poglavlju, sastoji se od dva osnova dijela: stablene strukture s korpusom (trenutačno stručnih tekstova, ali za budućnost zamišljeno puno šire) i glavnih funkcija (izvedenih tabovima, pretincima) koje služe za željenu obradbu odabranih dokumenata.

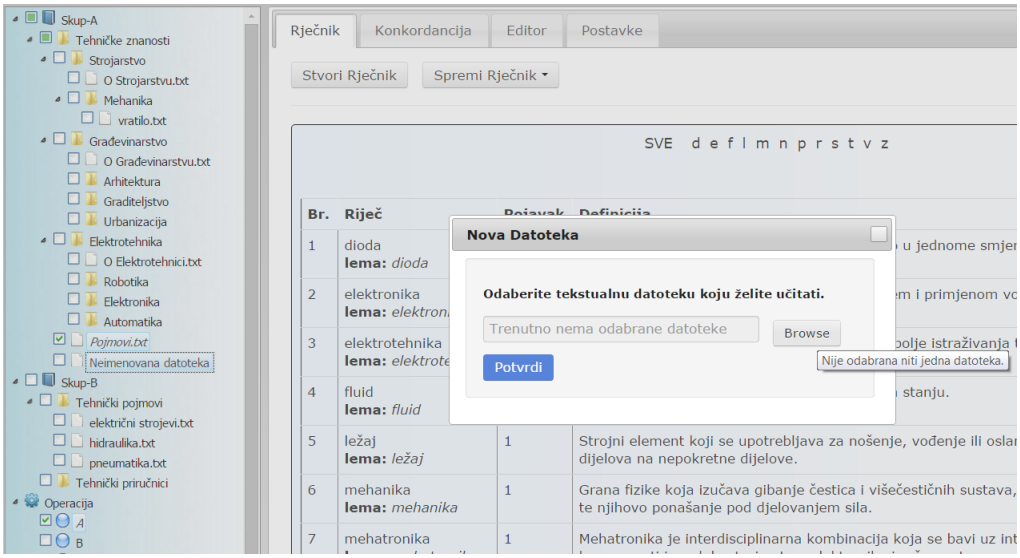
4.1 Stablena struktura

Stablena struktura, izgrađena na Fancy-tree Javascript modulu, zamišljena je kao vizualno spremište tekstova (korpusa) različite namjene. Tekstovi se spremaju u grane, a grane (mape, folderi) stvaraju po želji i gnijezde do bilo koje dubine. Odabir, označivanje neke grane više razine, automatski uključuje i sve njezine podrazine. U stablu se razlikuju 3 velike cjeline: skup A s granama i podgranama u koji se spremaju željeni dokumenti i skup B koji ima istu namjenu kao i A, a služi da bi se nad dokumentima iz oba skupa (A i B) mogle raditi različite pojedinačne i zajedničke logičke operacije. Na koncu, treća cjelina s imenom 'Operacija' služi upravo za takve operacije, traži, na primjer, razlike u skupovima riječi iz A i B odabranih dokumenata (to može biti djelotvorni način izvlačenja neologizama iz tekstova, novih stručnih riječi koje do tada nisu kategorizirane), pa presjek skupova riječi koje imaju i tekstovi iz A dijela i tekstovi iz B dijela, i slično.



Slika 39: Stablo stručnih tekstova

Stablo je osjetljivo na desni klik mišem, pri čemu se otvaraju padajuće izborne ponude. Njima se može otvoriti, preimenovati ili izbrisati označeni dokument, a može se također izgraditi nova mapa (eng. *folder*) ili nova datoteka (eng. *file*). Otvaranje nekog dokumenta na desni klik mišem i odabirom 'Otvori', taj tekst se otvara u pretincu '*Editor*' u kojem se može obrađivati (u ovom radu, obradba dokumenta nije načinjena, ali je predviđena u budućem razvitku programa).



Slika 40: Tvorba mapa i datoteka

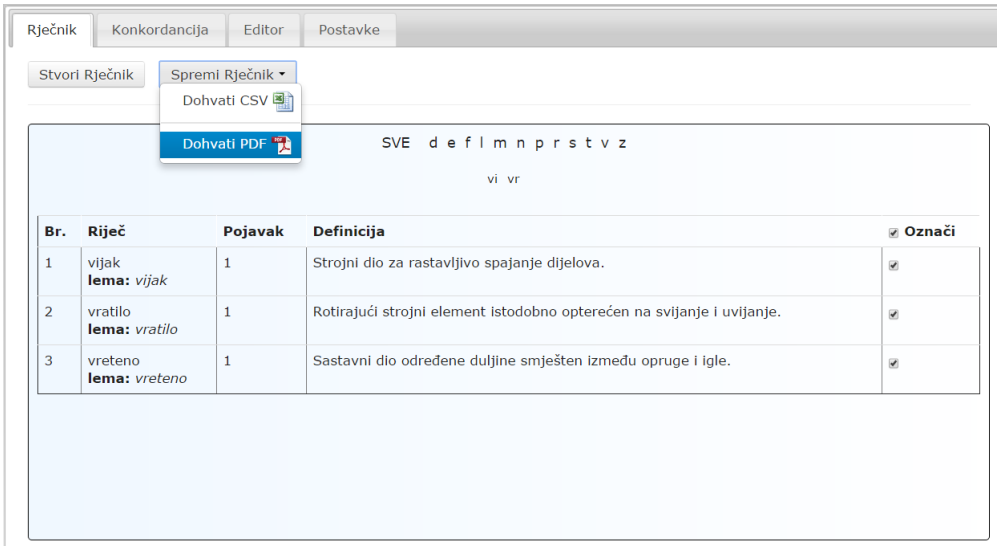
Kod brisanja mape i datoteke, korisniku se postavlja pitanje, je li to doista želi, kako bi se mogao predomisliti prije nepovratnih posljedica. Stvaranje nove mape/datoteke, jednostavan je postupak, u stablu se otvara obrazac u koji se samo upisuje ime koja se nakon toga stvara.

4.2 Glavne funkcije

Među glavnim funkcijama s desne strane u korisničkom sučelju nalaze se pretinci (eng. *tabs*) koji ostvaruju funkcije tvorbe *rječnika/abecedarija*, *konkordancije* u tekstovima, *editora* tekstova, tvorbe *leksikona* i *postavki* programa. Iako su funkcije više-manje intuitivne, u idućim potpoglavljima bit će detaljnije opisane.

4.2.1 Rječnik

Rječnik (abecedarij) tvori se iz jednog ili više odabranogih dokumenata. Klikom na 'Stvori rječnik' pokreće se program za tvorbu i ispis abecedarija. Njegov ispis na zaslon ovisan je o odabranim postavkama iz pretinca 'Postavke'. Tako će se na stranici pojaviti 200 prvih riječi iz abecedarija, ako je u postavkama odabrano baš 200 riječi (pretpostavljena vrijednost je 50 riječi), a ne više ili manje. Iduće stranice s riječima dobivaju se klikom na '>>> Sljedeća stranica' ili upisom broja za navigaciju apsolutne stranice. Moguća je također navigacija po slovu, a ako je i ona nepregledna i s prevelikom odazivom riječi, onda se može klikati na parove početnih slova, koji se pojavljuju ispod odabranog. Ovisno o postavkama u ispisu se mogu osim riječi pojavljivati i dodatna informacija (lema riječi, čestotnost riječi / broj pojavaka riječi u odabranom tekstu, definicija riječi iz leksikona).



Slika 41: Stvaranje PDF-a iz odabranih riječi

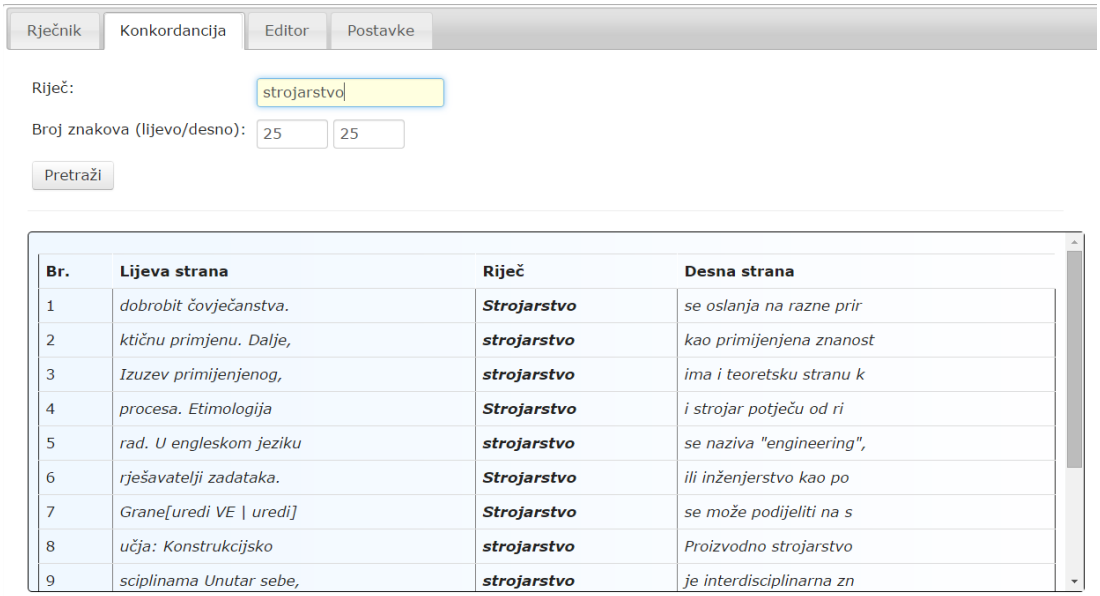
Rječnik/abecedarij se može dohvatiti u *CSV* i *PDF* formatu. U *CSV* formatu je pogodan za obradbu u *Excel* tablicama, a u *PDF* formatu za arhiviranje, trajnu pohranu ili ispis. U rječniku se pojedine riječi mogu označiti i odznačiti (eng. check/uncheck) kako bi se u *CSV* i *PDF-u pojavile samo željene.*

Br.	Riječ	Lema	Pojavak	Definicija
1	sila	sila	1	Uzrok promjene gibanja čestice ili tijela.
2	snaga	snaga	1	Brzina kojom se prenosi energija ili obavlja rad.
3	strojarstvo	strojarstvo	1	Grana tehnike koja se bavi konstruiranjem i proizvodnjom strojeva.
4	struja	struja	1	Usmjereno gibanje fluida, plina ili električnih naboja.
5	vratilo	vratilo	1	Rotirajući strojni element istodobno opterećen na svijanje i uvijanje.
6	vreteno	vreteno	1	Sastavni dio određene duljine smješten između opruge i igle.

Slika 42: Izgled PDF datoteke

4.2.2 Konkordancija i kontekst

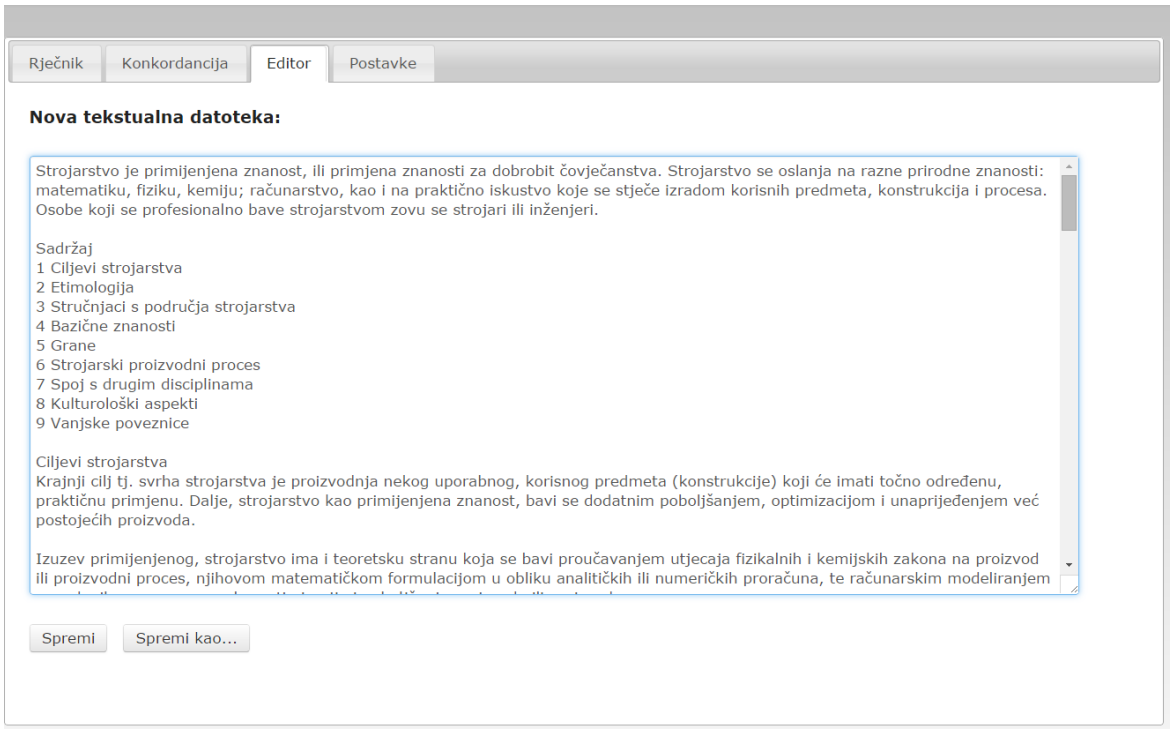
Konkordancija je funkcija koja služi za ispis okoliša (konteksta) za traženu riječ. Naime, dohvaćanje konteksta nekog stručnog pojma, važan je parametar u analizi stručnih tekstova. U ovoj, prvoj inačici ove funkcije, ispisuje se samo okoliš tražene riječi s obzirom na broj znakova slijeva i sdesna od nje (koji se može mijenjati). U budućim inačicama pretraga će se obogatiti dodatnim atributima (učestalim riječima uz traženu riječ - kolokacije, te morfo-sintaksnim i semantičkim atributima riječi).



Slika 43: Konkordancija - prikaz rezultata

4.2.3 Editor

Editor je funkcija u kojoj će korisnik moći mijenjati dokument spremljen u datoteci na stablu ili upisivati novi u slobodni obrazac, te ga potom spremati u datoteku u nekoj grani stabla. Trenutačno u editoru nisu uključene posebne funkcije nego samo one koje se tipkama na tipkovnici mogu postići, kao što je unos, mijenjanje i brisanje teksta.

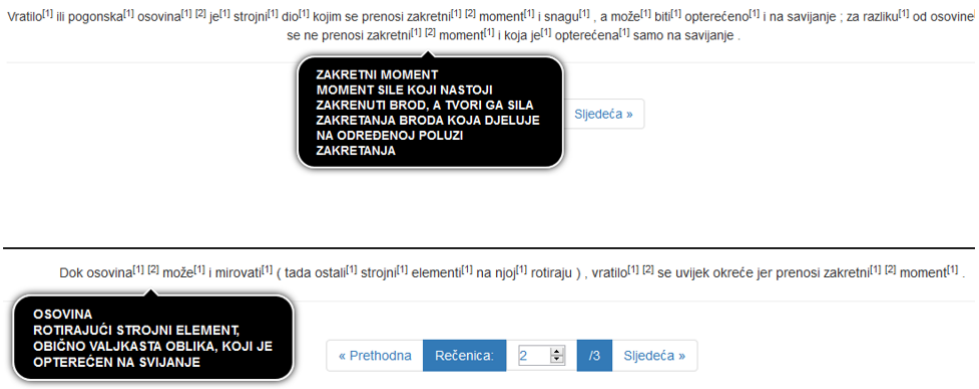


Slika 44: Editor - otvoreni tekst

4.2.4 Leksikon

Funkcija *Leksikon* temeljna je funkcija ovog programa, jer u odabranom tekstu slijedno ili navigacijski ispisuje njegove rečenice, s posebno označenim poznatim, a posebno nepoznatim pojmovima (riječima). Za nepoznate pojmove omogućuje otvaranje posebnog obrasca u koji se upisuje, definira, pojam/riječ.

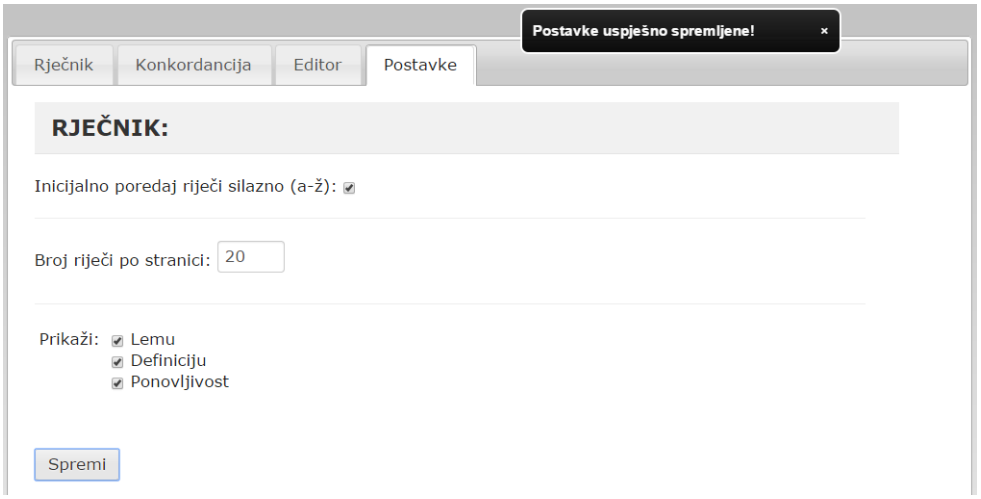
Slika 45: Leksikon - učitani tekst



Slika 46: Leksikon - prikaz rezultata

4.2.5 Postavke

U postavkama se odabiru parametri ispisa. Tako se može načiniti da ispisani *abecedarij* bude poredan od *A-Ž* ili od *Ž-A*. Mogu se uključiti ispisi čestotnosti, lema i definicija, a u budućim inačicama riječima će se pridružiti i gramatičke vrste (eng. POS, *part-of-speach*) koje se generiraju *mrežnim programom za on-line hrvatsko oblikoslovlje* i prikazom pripadnih gramatičkih atributa (rod, broj, padež, način, vrijeme, . . .).



Slika 47: Postavke

5 ZAKLJUČAK

Cilj ovog rada bio je osmisliti i načiniti (on-line) mrežnu aplikaciju koja će riješiti tvorbu i obradbu strukovnog nazivlja za tehničke znanosti (prvenstveno strojarstva), a proširivo i na sva druga područja znanosti (humanistiku, medicinu, arhitekturu i dr.). Rad se temelji na upoznavanju, a potom rješavanju svih nedostataka koje trenutačni rječnici ili terminološki repozitoriji u Hrvatskoj (npr. HJP, Struna, Wječnik) imaju, odnosno, projektiranju novog (programskog) mrežnog okvira (eng. framework), koji će stvarati tehničke i druge rječnike polu/automatskim izvlačenjem pojmova iz stručnih tekstova i istodobnom uporabom svih raspoloživih mrežnih resursa s javnim pristupom.

S funkcionalnog gledišta, rad na taj način proširuje mogućnosti postojećih mrežnih rječnika u Hrvatskoj: dodaje gramatičke oblike riječi, otkriva stručne riječi u nekom tekstu (napisanom ili dovučenom s mreže), ispisuje definicije i druge atribute stručnih riječi iz teksta koje je pronašao u nekom od mrežnih leksikona, omogućuje obradbe teksta (čestotnost, konkordancija, statistika, ...), daje mogućnost unosa riječi kojih u tehničkom leksikonu ili enciklopediji nema (a nalaze se u stručnom tekstu) i slično.

U radu je ustrojen i potpuni sustav mrežnih korisnika, tj. izvedena su hijerarhijska dopuštenja za više korisničkih skupina: administratori koji mogu izvršavati sve operacije i dodjeljivati dopuštenja korisnicima, skupine korisnika koji imaju pravo unosa novih riječi ili promjena u leksikonu, te korisnici koji imaju samo dopuštenja gledanja, ali ne i editiranja informacije.

Radom se također uspjelo obuhvatiti *Strunu* i slične rječnike, leksikone i enciklopedije, u zajedničku, kvalitetniju cjelinu, te olakšati održavanje i proširivanje novog mrežnog leksikona bilo kojih područja, polja i pripadnih grana.

Dakako, radi se u ovom trenutku samo o programu, mrežnom okviru, ali ne i sadržaju (riječima) takvog mrežnog leksikona.

Za korisnika koji se ne bavi leksikografskim poslom, cilj programa bio je omogućiti mu prepoznavanje stručnog/tehničkog nazivlja u nekom njegovom tekstu (napisanom ili dohvaćenom s mreže), ispis značenja (definicija) i atributa prepoznatih riječi, te povezivanje tih značenja s mrežnim izvorom koji daje više informacija. Za one koji se bave leksikografskim poslom, ili žele pomoći u tvorbi mrežnog leksikona iz osobne struke, cilj programa bio je omogućiti tvorbu onih stručnih riječi, pronadjenih u tekstu, koji do sada nisu obradjene.

Kako je najnoviji tehnologijski trend pretvaranje sveobuhvatne informacije u povezane otvorene podatke (eng. LOD-linked open data), glavna zamisao bila je načiniti alat kojim će se takvi (LOD) podaci automatski generirati i spremati u triplestore (Virtuoso) repozitorij. Zato je u radu veliki posao načinjen i za takvu pripravu i pokazano dohvaćanje trojaca (stručnog nazivlja) s pomoću SparQL naredbi. Ocjena rada je da su ciljevi u potpunosti postignuti, pa preostaje još podizanje načinjenog sustava na nacionalnu razinu kroz CARNet-ov web-servis. Dugoročan cilj je moguća uporaba ovog mrežnog leksikona, preko hrvatske akademske mreže, u svakoj školi, fakultetu ili institutu, kako bi u čuvanju i razvitku hrvatskoga jezikoslovlja mogli sudjelovati svi nastavnici/profesori i njihovi učenici/studenti.

1. Stvaranje korpusa stručnih dokumenata (stablo) - upisom ili dovlačenjem s

mreže

2. Odabir teksta/dokumenta iz stabla ili ručni upis novog teksta
3. Određivanje i spremanje postavki na kartici "Postavke"
4. Tvorba rječnika, abecedarija
5. Tvorba leksikona i njegovo dohvaaćanje u CSV ili PDF obliku
6. Rad s karticom "Konkordancija" za promatranje okoliša/konteksta rječi
7. Rad s karticom "Editor" za unos, promjenu i brisanje stručnog teksta
8. Rad s karticom "Lexicon" koja obrađuje tekst i daje definicije stručnih riječi ili prijavljenom korisniku omogućuje definiciju nepoznate riječi
9. Spremanje novog pojma u bazu podataka i njegovu pretvorbu u LOD trojac
10. Administracijsko sučelje daje mogućnosti stvaranja hijerarhijskih grupa korisnika s potpunim ili ograničenim dopuštenjima.

Rad se može provjeriti/demonstrirati i mrežno/online na: `https://jt195996.pythonanywhere.com/Test_FancyTree/default/index` (s obzirom na česte promjene servera, za najnoviju inačicu kontaktirati autore ili mentora). Rad je ostvaren s programskim (Python, HTML) kôdom od preko 3.500 linija.

LITERATURA

[1] Massimo Di Piero: *WEB2PY*

[2] Felix López, Victor Romero: *Mastering Python Regular Expressions*

[3] Marko Cundeković: *Povezani podaci u novom ustroju Interneta*

[4] Juraj Benić: *Izvlačenje strukovnog nazivlja iz strojarskih tekstova*

[5] Jakov Topić: *Izvlačenje povezanih podataka iz strojarskih dokumenata s mreže*

[6] Steven M. Schafer: *Web Standards Programmer's Reference : HTML, CSS, JavaScript, Perl, Python, and PHP*

[7] Rodolfo Delmonte: *Computational linguistic text processing: lexicon, grammar, parsing, and anaphora resolution*

[8] `https://www.python.org/`

[9] `http://www.nltk.org/`

[10] `http://protege.stanford.edu/`

[11] `http://linkeddata.org/`

[12] `http://virtuoso.openlinksw.com`

[13] `http://jquery.com/`

[14] `http://getbootstrap.com/2.3.2/javascript.html`

[15] `http://www.w3.org/standards/techs/`

[16] `www.zemris.fer.hr/predmeti/krep/Pavic.pdf`

[17] `http://intranet.fesb.hr/Portals/0/docs/nastava/kvalifikacijski/Karmen%20Klarin%20KDI%20v14.pdf`

[18] `http://struna.ihjj.hr/`

[19] `http://narodne-novine.nn.hr/clanci/sluzbeni/2009_09_118_2929.html`

[20] `http://en.wikipedia.org/wiki/World_Wide_Web`

[21] `http://en.wikipedia.org/wiki/Semantic_Web`

[22] `http://en.wikipedia.org/wiki/Web_Ontology_Language`

SAŽETAK

Autori: Juraj Benić i Jakov Topić

Naslov rada: MREŽNI PROGRAM ZA TVORBU I OBRADBU TEHNIČKIH RJEČNIKA

U ovom radu zamišljen je i realiziran složeni mrežni alat za stvaranje, obnavljanje/ažuriranje, obradbu i uporabu tehničkih rječnika različitih područja. Alat je dio projekta mrežnog hrvatskoga jezikoslovlja, koji obuhvaća on-line oblikoslovlje (morfologiju), skladnju (sintaksu) i semantiku, temeljene na vremensko-prostornom mrežnom korpusu standardne i stručne literature opće hrvatske kulturne baštine. Projekt (s podprojektima) predstavljen je u 2014. godini na Otvorenim vratima HAZU, a u 2015. godini na događanjima Matice Hrvatske i na 5. NSK festivalu hrvatskih digitalizacijskih projekata.

Poticaj za izgradnju ovog mrežnog programa za stručne leksikone došao je od hvale-vrijednog mrežnog projekta Struna (`http://struna.ihjj.hr`) s Instituta za hrvatski jezik i jezikoslovlje. Cilj rada je obuhvatiti Strunu i slične rječnike, leksikone i enciklopedije, u zajedničku, kvalitetniju cjelinu, te olakšati održavanje i proširivanje leksikona bilo kojih područja, polja i pripadnih grana.

S tehničke strane gledano, rad je izveden u 'web2py' MVC mrežnom okviru (engl. framework), s modulima pisanim u Python programskom jeziku. Skladištenje jezikoslovne informacije ostvareno je uz pomoć MySqL/SQLite relacijskih baza, te Virtuoso triplestore baze za povezane podatake.

S funkcionalnog gledišta, rad proširuje mogućnosti postojećih mrežnih rječnika ili repozitorija u Hrvatskoj: dodaje gramatičke oblike riječi, otkriva stručne riječi u nekom tekstu (napisanom ili dovučenom s mreže), ispisuje definicije i druge atribute stručnih riječi iz teksta koje je pronašao u nekom od mrežnih leksikona, omogućuje obradbe teksta (čestotnost, konkordancija, statistika, ...), daje mogućnost unosa riječi kojih u tehničkom leksikonu ili enciklopediji nema, a nalaze se u stručnom tekstu, i slično. Organiziran je i potpuni sustav mrežnih korisnika, tj. izvedena su hijerarhijska dopuštenja za više korisničkih skupina: administratori koji mogu izvršavati sve operacije i dodjeljivati dopuštenja korisnicima, skupine korisnika koji imaju pravo unosa novih riječi ili promjena u leksikonu, te korisnici koji imaju samo dopuštenja gledanja, ali ne i editiranja informacije.

U dodatku rada, kratko su opisani glavni algoritmi ovog alata i to za izvlačenje informacije iz teksta (uz pomoć regularnih izraza), algoritmi transformacije informacije iz računalne baze u triplestore bazu, te temeljni algoritmi za vizualizaciju korisničkog sučelja u web2py okolišu, od kojih su samo neki bili dio završnih radova autora (objavljenih na mreži).

Za funkcioniranje ovog mrežnog sustava bilo je potrebno napisati preko 3.500 linija programskog kôda.

Ključne riječi: *leksikon, tehnički rječnik, web2py, izvlačenje informacije, povezani podaci*

ABSTRACT

Autors: Juraj Benić i Jakov Topić

Title: NETWORK PROGRAM FOR FORMATION AND ANALYSIS OF TECHNICAL DICTIONARIES

This paper conceives and develops a complex network tool for creation, updating, analysis and usage of technical dictionaries from various areas. The tool is a part of a project of Croatian computational (network) linguistics, as it comprises the on-line morphology, syntax and semantics, which are based on a temporal and spatial corpus of the standardized and expert literature related to general Croatian cultural heritage. The project (with its sub-projects) was introduced in 2014 during the event of Open doors of Croatian Academy of Science and Art, and in 2015 during the happenings of Matica hrvatska and at the 5th festival of Croatian digitalization projects in the National University Library.

The incentive for the development of this network programme for terminological lexicons came from the praiseworthy network project Struna (`http://struna.ihjj.hr`) of the Institute for Croatian language and linguistics. The goal of the paper is to include the Struna and similar dictionaries, lexicons and encyclopaedias in a common, high-quality unit, and to ease the maintenance and expansion of a lexicon of any area, field or scientific branch.

Technically speaking, the paper was formed in the web2py MVC network framework, with modules written in Python programming language. The storage of linguistic data is generated by MySqL/SQLite relational database, and the Virtuoso triplestore database for related information.

From the functional point of view, the paper expands the possibilities of existing network dictionaries and repositories in Croatia: it adds the grammatical forms of a word, detects terms in a text (written on-line or dragged from the network), notes the definitions and other attributes of terms in a text found in one of the network lexicons, enables text analysis (frequency, concordance, statistics...), ensures the entry of words which are not in the terminological lexicon or encyclopaedia, and are in the professional text etc. A complete system of network users is organized, i.e. hierarchical solutions are derived for more user groups: the administrators can execute all the operations and assign permissions for other users; the groups of users which have the permission to enter new words in the lexicon or edit it; and the users who only have the permission to view, but not to edit the information.

In the paper's appendix there is a short description of the algorithms of this tool: those for information extraction from the text (using regular phrases), algorithms for information transformation from the computer database to a triplestore database, and the fundamental algorithms for the visualization of the user interface in web3py environment, some of which were parts of the final works of the authors (published on-line).

For this network system to function properly, it was necessary to write more than 3500 lines of programming code.

Key words: lexicon, terminology, web2py, information extraction, related data

Životopisi

Juraj Benić rođen je 15. travnja 1992. godine u Karlovcu. Nakon završene osnovne škole upisuje tehničku školu Karlovac. U drugom i trećem razredu srednje škole sudjeluje na državnom natjecanju iz tehničke mehanike, a u četvrtom razredu sudjeluje na državnom natjecanju iz dizajniranja računalom u CATIA-i. Maturirao je 2011. godine. Iste godine upisuje preddiplomski studij strojarstva na Fakultetu strojarstva i brodogradnje u Zagrebu. Preddiplomski studij mehatronike i robotike završio je 2015. godine, te iste godine upisuje diplomski studij. Aktivno se služi engleskim jezikom.

Jakov Topić rođen je 27. studenog 1990. godine u Zagrebu. Nakon završene osnovne škole Ljudevita Gaja u Zaprešiću upisuje strojarsko-tehnički smjer u srednjoj školi Frane Bošnjakovića u Zagrebu. Maturirao je 2009. godine. Iste godine upisuje Strojarski Fakultet u Slavonskom brodu, te nakon pola godine napušta školovanje. Sljedeće godine polaže Državnu maturu te 2011. godine upisuje preddiplomski studij brodogradnje na Fakultetu strojarstva i brodogradnje u Zagrebu. Nakon položene prve godine prebacuje se na studij strojarstva te upisuje smjer mehatronike i robotike. Preddiplomski studij mehatronike i robotike završio je 2015. godine, te iste godine upisuje diplomski studij. Aktivno se služi engleskim jezikom.

DODATAK

1. Pretraživanje teksta i dohvaćanje informacije
2. Funkcija za konverziju podataka u triplet (”Turtle - Terse RDF triple language ”) zapis
3. Funkcija unutar kontrolera za stvaranje rječnika

Pretraživanje teksta i dohvaćanje informacije

Konačno, nakon što su definirani uzorci i izrazi, pristupa se pozivu programa koji će pretražiti tekst i izvući željenu informaciju. Na primjer, u slici 48 . navedeni su uzorci za dohvaćanje predikata, i to kao pomoćnoga glagola biti uz njega glagolski pridjev radni, zatim uz riječ 'ne' koja stoji ispred glagola i glagola u prezentu, infinitivu, aoristu i imperativu.

```
('','^ob',1);('','^gr',1,'lg=^ob');('^ne$','',1,'uzdg=^g');('','^g',1,'uzlr=^ne$')
('','^gp',1);('','^gi',1);('','^gm',1);('','^ga',1)
```

Slika 48: Uzorci za pretraživanje teksta

Algoritam za pretraživanje teksta po uzorcima radi u koracima. Prvo rastavi rečenicu na riječi i zatim iz baze izvadi tagove za pojedinu riječ i spremi ih u listu. Nakon toga uzima riječ po riječ i za svaku od njih pretražuje poklapa li se riječ ili njezin tag sa zadanim. Ako se podudara, onda rezultat spremi u listu. Nakon što je program prošao kroz sve riječi u rečenici, počinje pretraživati sve mogućih kombinacije te rečenice, jer jedna riječ može biti imenica u nekom kontekstu, dok je u drugom glagol ili se ista riječ definira s dva različita uzorka, pa imamo dvije slovne oznake za nju. Nakon što se izračuna broj mogućih kombinacija, program pravu rečenicu zamjenjuje rečenicom sa slovnim oznakama i vraća listu tih rečenica. Kako to izgleda u programu, vidi se u Algoritmu 2.

```
1  def pretraz_recenice(self,recenica, uzorak, sl_oz, stupac):
2    """Pretrazuje svaku rijec u recenici za svaki zadani uzorak
3    Vraća listu svim mogućih kombinacija i listu pozicija u
       recenici"""
4      rijeci=word_tokenize(recenica)      #lista rijeci iz recenice
5      gram_oblik=self.iz_baze([r.lower() for r in rijeci], stupac)
           #gram_oblik za svaku rijec
6
7      pozicija=[] #pozicija rijeci u recenici
8      rez=[]; rez_2=[]     #rez_2 - nadjeno dodatno
9      for r in xrange(len(rijeci)):      #za sve rijeci u recenici,
           r -> pozicija rijeci u recenici
10       rez_1=[]     #rez_1 - nadjeno opcenito;
11         for u in xrange(len(uzorak)):   #za sve uzorke
12           for g in re.split(r'/',gram_oblik[r]): #za sve gram.
                 oblike rijeci
13             if self.trazi(rijeci[r].lower(),g,uzorak[u]) is
                   True: #ako je nadjeno nesto
14
15               if len(uzorak[u])>3:   #ako uzorak ima vise od
                     3 argumenta
16                 uzr=uzorak[u][3:len(uzorak[u])]
```

```
17                              pom=dodatni_argumenti(rijeci,gram_oblik,
                                    r,uzr) #pretrazivanje dodatnih
                                    argumenata
18                              if pom:
19                                rez_1.append(sl_oz[u])
20                          else:
21                            rez_1.append(sl_oz[u])
22          if rez_1:
23              pozicija.append(r)
24              rez.append(list(set(rez_1)))
25
26    return list(it.product(*rez)), pozicija, gram_oblik
```

Primjer 2: Funkcija koja pretražuje tekst po uzorcima

Nakon što program nađe podudaranje uzorka i riječi u tekstu, nju zamjenjuje s oznakom koju izraz ima, kao što je bilo prikazano u 3.5.2., u ovom slučaju to su oznake 1A, 1B, 1C, 1D, 2A, 2B, 2C i 2D, a za ostale riječi koje se ne traže program stavlja znak crtice ('–'), dok za riječi koje nema u bazi podataka stavlja znak povisilice ('#'). Nakon toga dolaze izrazi koji kombiniraju oznake kojima su zamijenjene pronađene riječi i onda na temelju novih podudaranja između riječi i oznaka dobivamo željeni podatak iz teksta. Za slučaj traženja predikata mogli bismo napisati sljedeće izraze, kako se vidi na slici 49.

```
PRED: ((1A)[-|#]*(1B))|(1C1D)|(1A)|(2A)|(2B)|(2C)|(2D)
```

Slika 49: Izraz za traženje predikata u tekstu

Algoritam za pretraživanje izraza radi po sljedećim koracima. Kao ulazni argument program prima listu rečenica u kojima su riječi zamijenjene slovnim oznakama. Funkcija prolazi kroz svaku rečenicu pojedinačno i na njoj isprobava izraze koje smo prethodno bili sastavili, a nakon što je nađeno podudaranje izraza i oznaka u novoj rečenici, program sprema mjesto podudaranja u rečenici zajedno s vrstom izraza. Nakon što je program prošao kroz sve rečenice, vraća listu sa svim tim rezultatima. Kako to izgleda u programskom kodu, može se vidjeti u algoritmu 3.

```
1  def trazenje_po_izr(self,nadjeno_uzr, izr,pozicije):
2      """Trazi koja kombinacija odgovara zadanom izrazu
3      nadjeno_uzr=['X1X0X2',...]  vraca koja kombinacija odgovara,
           pocetak, kraj i reg_izraz"""
4      vise_odv_rj=[];vise_sku_rj=[];jedno_rj=[]
5      for n in nadjeno_uzr:   #za sve kombinacije iz nadjeno_uzr
6          komb=''.join(n)
7          for i in re.split(r'\|',izr): #za svaki izraz iz
               kombinacije
8
9              if '[\d+\D]' in i:  #vise odvojenih rjesenje
```

```
10                  for g in re.finditer(i,komb):
11                      pom=[]
12                      for p in xrange(2,len(g.groups())+1):
13                          pom.append(self.poz_u_tekstu(pozicije,
                                komb,g.start(p)))
14                      if pom:
15                          vise_odv_rj.append((tuple(pom),i))
16
17              elif re.search(r'\(\d+\D\)',i): #jedno rjesenje
18                  for g in re.finditer(i,komb):
19                      jedno_rj.append((self.poz_u_tekstu(pozicije,
                            komb,g.start()),i))
20
21              else:   #vise spojenih rjesenje
22                  for g in re.finditer(i,komb):
23                      c=self.poz_u_tekstu_skupa(pozicije,komb,g.
                            start(), g.end())
24                      if self.niz(c) is True: vise_sku_rj.append((
                            c,i))
25
26      return sorted(list(set(jedno_rj))), sorted(list(set(
            vise_sku_rj))),sorted(list(set(tuple(i) for i in
            vise_odv_rj)))
```

Primjer 3: Funkcija za pretraživanje po izrazima

Nakon što je program pretražio rečenice po uzorcima i izrazima, moraju se još samo ispisati rezultati. Da bi se oni ispisali, prvo moramo dohvatiti riječi koje su se nalazile na mjestima na kojima su se slovne oznake poklopile s izrazima. Njih vadimo s pomoću rezultata koje vraća funkcija iz Tablice 9., tj. uz pomoć početka i kraja mjesta na kojem je program našao podudaranje sa slovnim oznakama. Kako je to realizirano u algoritmu, vidi se u algoritmu 4.

```
1  def rezultat(self,rec, uzr, izr, slovne_oz, objekti, stupac):
2          """rec->recenice, uzr->uzorak izr->izrazi, slovne_oz,
               stupac->po_cemu_se_baza_pretrazuje"""
3          rez=''
4          for r in xrange(len(rec)):
5              rijeci=word_tokenize(rec[r])
6              if self.provjera_uzoraka(uzr) is True:
7                  nadjeno_uzr,pozicije, gram_obl=self.
                       pretraz_recenice(rec[r],uzr,slovne_oz,stupac)
8
9                  for i in xrange(len(izr)):
10                     jedno_rj,vise_skupa,vise_odvojeno=__filtri__
                           (rijeci,*self.trazenje_po_izr(nadjeno_uzr
                           ,izr[i],pozicije))
11
12
```

```
13
14                      nema='Nema u bazi: '
15                      for z in xrange(len(gram_obl)):
16                          if gram_obl[z]=='#':
17                              nema+=rijeci[z]+', '
18
19                      if i==0:
20                          if nema=='Nema u bazi: ': rez+=rec[r]+'\
                                n'
21                          else: rez+=rec[r]+'\n\n'+nema+'\n\n'
22
23                      rez+= objekti[i]+'\n'
24
25                      for j in jedno_rj:
26                          if rijeci[j[0]]=='da': pass     #
                                izbacuje rijec da ako je sama kao
                                rješenje
27                          else:  rez+='\t'+rijeci[j[0]]+'\n'
28
29                      for j in vise_skupa:
30                          tekst=''
31                          for k in j[0]: tekst+=' '+rijeci[k]
32                          rez+='\t'+tekst+'\n'
33
34                      for j in vise_odvojeno:
35                          tekst=''
36                          for k in j[0]: tekst+=' '+rijeci[k]
37                          rez+='\t'+tekst+'\n'
38              rez+='\n'
39              rez+='-' * 30+'\n'
40          return rez
```

Primjer 4: Algoritam za ispis rezultata

Kao što možemo vidjeti, funkcija za ispis rezultata poziva funkciji iz algoritama 2 i 3 te neke dodatne vlastite funkcije za filtriranje podataka. Tek se njihovim povezivanjem stvara i prikazuje konačni rezultat.

Funkcija za konverziju podataka u triplet (”Turtle - Terse RDF triple language ”) zapis

```
def Convert_to_Turtle(podaci):
    """
    Kao argument uzima sve pronađene podatke pojma te iz njih stvara Turtle
    format, kojeg potom ubacuje u 'Turtle.owl' datoteku.
    """
    # pojam
    pojam = podaci['pojam']
    # gramatika
    vrsta_rijeci = podaci['gramatika']['vrsta']
    rod = podaci['gramatika']['rod']
    broj = podaci['gramatika']['broj']
    # definicija
    definicija = podaci['definicija']
    # istoznacnice
    istoznacnice = podaci['istoznacnice']
    (dopusteni_naziv, nepreporuceni_naziv, zargonizam) = (None, None, None)
    if istoznacnice != None:
        istoznacnice_list = istoznacnice.split('|')
        istoznacnice = ""
        for istoznacnica in istoznacnice_list:
            istozn = re.search(r"(?P<tip>.+):\s(?P<istoznacnica>.+)", istoznacnica)
            if istozn.group("tip") == "dopušteni naziv".decode("utf-8"):
                dopusteni_naziv = istozn.group("istoznacnica")
            elif istozn.group("tip") == " nepreporučeni naziv".decode("utf-8"):
                nepreporuceni_naziv = istozn.group("istoznacnica")
            elif istozn.group("tip") == "žargonizam".decode("utf-8"):
                zargonizam = istozn.group("istoznacnica")
            istoznacnice += istozn.group("istoznacnica") + ", "
        istoznacnice = istoznacnice[:-2]
    # istovrijednice
    istovrijednice = podaci['istovrijednice']
    (istovrijednica_en, istovrijednica_fr, istovrijednica_de, istovrijednica_slo) = ('', '', '', '')
    if istovrijednice != None:
        istovrijednice_list = istovrijednice.split('|')
        for istovrijednica in istovrijednice_list:
            istovr = re.search(r"(?P<jezik>\w+):\s(?P<istovrijednica>.+)", istovrijednica)
```

```
37              if istovr.group("jezik") == "engleski":
38                  istovrijednica_en = istovr.group("istovrijednica
                        ")
39                  istovrijednica_en = istovrijednica_en.strip()
40              elif istovr.group("jezik") == "francuski":
41                  istovrijednica_fr = istovr.group("istovrijednica
                        ")
42                  istovrijednica_fr = istovrijednica_fr.strip()
43              elif istovr.group("jezik") == "njemački".decode("utf
                    -8"):
44                  istovrijednica_de = istovr.group("istovrijednica
                        ")
45                  istovrijednica_de = istovrijednica_de.strip()
46              elif istovr.group("jezik") == "slovenski":
47                  istovrijednica_slo = istovr.group("
                        istovrijednica")
48                  istovrijednica_slo = istovrijednica_slo.strip()
49      # kratice
50      kratice = podaci['kratice']
51      (kratica_hr, kratica_en, kratica_de, kratica_fr) = ('', '',
            '', '')
52      if kratice != None:
53          kratice_list = kratice.split('|')
54          for kratica in kratice_list:
55              krat = re.search(r"(?P<jezik>\w+):\s(?P<kratica>.+)"
                    , kratica)
56              if krat.group("jezik") == "hrvatska":
57                  kratica_hr = krat.group("kratica")
58              elif krat.group("jezik") == "engleska":
59                  kratica_en = krat.group("kratica")
60              elif krat.group("jezik") == "francuska":
61                  kratica_fr = krat.group("kratica")
62              elif krat.group("jezik") == "njemačka".decode('utf-8
                    '):
63                  kratica_de = krat.group("kratica")
64      # napomena
65      napomena = podaci['napomena']
66      # razredba
67      razredba = podaci['razredba']
68      (polje, grana) = (None, None)
69      if razredba != None:
70          razredba_list = razredba.split('|')
71          for razredba in razredba_list:
72              razr = re.search(r"(?P<razredba>\w+):\s(?P<tekst>.+)
                    ", razredba)
73              try:
74                  if razr.group("razredba") == "polje":
75                      polje = razr.group("tekst")
76                  elif razr.group("razredba") == "grana":
```

```
77                      grana = razr.group("tekst")
78              except:
79                  pass
80      # podređeni nazivi
81      podredjeni_nazivi = podaci['podredjeni nazivi']
82      if podredjeni_nazivi != None:
83          podredjeni_nazivi_list = podredjeni_nazivi.split(" | ")
84          podredjeni_nazivi = ""
85          for i in range(len(podredjeni_nazivi_list)):
86              if i != len(podredjeni_nazivi_list)-1:
87                  podredjeni_nazivi += podredjeni_nazivi_list[i] +
                        ', '
88              else:
89                  podredjeni_nazivi += podredjeni_nazivi_list[i]
90      # kontekts
91      kontekst = podaci['kontekst']
92      # nadpojam
93      nadpojam = podaci['nadpojam']
94      # link pojma
95      pojam_link = podaci['pojam_link']
96
97      #================ stvaranje turtle zapisa ==================
98      resurs = createResource(pojam)
99      resurs_link = "###  http://www.fsblod.hr/ont/resource#" +
            resurs
100     # provjera da li triplet vec postoji u datoteci
101     link = "###  http://www.fsblod.hr/ont/resource#" + resurs
102     if re.search(link, tekst_dokumenta):
103         print "%s se već nalazi u dokumentu Turtle.owl!".decode(
                'utf-8') %pojam
104     else:
105         # kreiraj triplet
106         turtle = "\n\n###  http://www.fsblod.hr/ont/resource#" +
                resurs + "\n\n" +\
107                  "res:" + resurs + " rdf:type res:tehnicki , owl
                        :NamedIndividual ; \n\n" +\
108                  "rdfs:label " + "\"" + pojam + "\"" + " ;\n\n"
109         if definicija != None:
110             turtle += "pro:definicija " + "\"" + definicija + "
                    \"" + " ;\n\n"
111         if definicija != None:
112             turtle += "pro:definicija " + "\"" + definicija + "
                    \"" + " ;\n\n"
113         if istoznacnice != None:
114             turtle += "pro:istoznacnice " + "\"" + istoznacnice
                    + "\"" + " ;\n\n"
115         if dopusteni_naziv != None:
116             turtle += "pro:dopusteni_naziv " + "\"" +
                    dopusteni_naziv + "\"" + " ;\n\n"
```

```
117         if nepreporuceni_naziv != None:
118             turtle += "pro:nepreporuceni_naziv " + "\"" +
                    nepreporuceni_naziv + "\"" + " ;\n\n"
119         if zargonizam != None:
120             turtle += "pro:zargonizam " + "\"" + zargonizam + "
                    \"" + " ;\n\n"
121         if istovrijednice != None:
122             turtle += "pro:istovrijednice " + "\"" +
                    istovrijednica_en + "\"@en" + " , \"" +
                    istovrijednica_fr + "\"@fr"\
123              + " , \"" +istovrijednica_de + "\"@de" + " , \"" +
                     istovrijednica_slo + "\"@slo ; \n\n"
124         if kratice != None:
125             turtle += "pro:kratice " + "\"" + kratica_hr + "\"
                    @hr" + " , \"" + kratica_en + "\"@en"\
126              + " , \"" +kratica_de + "\"@de" + " , \"" +
                     kratica_fr + "\"@fr ; \n\n"
127         if napomena != None:
128             turtle += "pro:napomena " + "\"" + napomena + "\"" +
                     " ;\n\n"
129         if polje != None:
130             turtle += "pro:polje " + "\"" + polje + "\"" + " ;\n
                    \n"
131         if grana != None:
132             turtle += "pro:grana " + "\"" + grana + "\"" + " ;\n
                    \n"
133         if podredjeni_nazivi != None:
134             turtle += "pro:podredjeni_nazivi " + "\"" +
                    podredjeni_nazivi + "\"" + " ;\n\n"
135         if kontekst != None:
136             turtle += "pro:kontekst " + "\"" + kontekst + "\"" +
                     " ;\n\n"
137         turtle += "pro:nadpojam " + "\"" + nadpojam + "\"" + "
                ;\n\n" +\
138                   "pro:pojam_link " + "\"" + pojam_link+ "\"" +
                        " ;\n\n"
139         turtle = turtle[:-4]
140         turtle += '.'
141         # upisi triplet u datoteku
142         print "Resurs %s je dodan u Turtle.owl datoteku!".decode
                ('utf-8') %resurs
143         Turtle_doc.write(turtle.encode("utf-8"))
```

Primjer 5: Algoritam za konverziju podataka u triplet zapis

Funkcija unutar kontrolera za stvaranje rječnika

```
def Create_Dict():
    try:
        data = request.vars.ajaxdata
        titlelist = data.split(";")
        operacija = titlelist[-1]
        txtdatabase = db(db.textmain).select()
        Tekst_Skup_A = ""
        Tekst_Skup_B = ""
        for title in titlelist:
            for row in txtdatabase:
                if (row.ime == title and row.skup == "A"):
                    Tekst_Skup_A += str(row.tekst)
                if (row.ime == title and row.skup == "B"):
                    Tekst_Skup_B += str(row.tekst)
        if Tekst_Skup_A == "":
            Skup = "B"
        if Tekst_Skup_B == "":
            Skup = "A"
        if Tekst_Skup_A != "" and Tekst_Skup_B != "":
            Skup = "AB"
        Lista_Rijeci_Skup_A = []
        for word in word_tokenize(Tekst_Skup_A):
            rijec = convertUTF8(word.lower())
            if rijec[-1] in LISTA_ZNAKOVA:
                rijec = rijec[:-1]
            if re.match(r"[a-zA-ZšđčćžŠĐČĆŽ]+".decode("utf-8"),
                rijec):
                Lista_Rijeci_Skup_A.append(rijec)
        Lista_Rijeci_Skup_B = []
        for word in word_tokenize(Tekst_Skup_B):
            rijec = convertUTF8(word.lower())
            if rijec[-1] in LISTA_ZNAKOVA:
                rijec = rijec[:-1]
            if re.match(r"[a-zA-ZšđčćžŠĐČĆŽ]+".decode("utf-8"),
                rijec):
                Lista_Rijeci_Skup_B.append(rijec)
        # Operacija A
        if operacija == "A":
            Rezultat = list(set(Lista_Rijeci_Skup_A))
            Rezultat.sort(key=str.lower)
        # Operacija B
        elif operacija == "B":
            Rezultat = list(set(Lista_Rijeci_Skup_B))
            Rezultat.sort(key=str.lower)
        # Operacija A-B
        elif operacija == "A - B":
```

```
45              RazlikaA_B = list(set(Lista_Rijeci_Skup_A).
                    difference(Lista_Rijeci_Skup_B))
46              RazlikaA_B.sort(key=str.lower)
47              Rezultat = RazlikaA_B
48          # Operacija B-A
49          elif operacija == "B - A":
50              RazlikaB_A = list(set(Lista_Rijeci_Skup_B).
                    difference(Lista_Rijeci_Skup_A))
51              RazlikaB_A.sort(key=str.lower)
52              Rezultat = RazlikaB_A
53          # Operacija A^B
54          elif operacija == "A ^ B":
55              XOR = list(set(Lista_Rijeci_Skup_A).
                    symmetric_difference(Lista_Rijeci_Skup_B))
56              XOR.sort(key=str.lower)
57              Rezultat = XOR
58          # Operacija Presjek(A,B)
59          elif operacija == "Presjek(A,B)":
60              Presjek = list(set(Lista_Rijeci_Skup_A).intersection
                    (Lista_Rijeci_Skup_B))
61              Presjek.sort(key=str.lower)
62              Rezultat = Presjek
63          # Operacija Unija(A,B)
64          elif operacija == "Unija(A,B)":
65              Unija = list(set(Lista_Rijeci_Skup_A).union(
                    Lista_Rijeci_Skup_B))
66              Unija.sort(key=str.lower)
67              Rezultat = Unija
68          # Bez selektiranih tekstova ili operacije
69          else:
70              return "Selektirajte tekst(ove) i operaciju koju ž
                    elite izvršiti te pritisnite na gumb \"Stvori rje
                    čnik\"."
71          # Filtriranje Rezultata
72          Rezultat = [rijec.strip() for rijec in Rezultat]
73          # Kreacija Rjecnika
74          session.Rjecnik = {}
75          session.Rjecnik_Sub = {}
76          session.Podaci_Rjecnik = {}
77          if checkUTF8(Rezultat[0][0]) == False:
78              pocetno_slovo = Rezultat[0][0]
79          else:
80              pocetno_slovo = Rezultat[0][:2]
81          i = 0
82          for Rijec in Rezultat:
83              if checkUTF8(Rijec[:2]) == False:
84                  if Rijec[0] != pocetno_slovo:
85                      pocetno_slovo = Rijec[0]
86                      i = 0
```

```
87                else:
88                    if Rijec[:2] != pocetno_slovo:
89                        pocetno_slovo = Rijec[:2]
90                        i = 0
91                if i == 0:
92                    Lista = []
93                    i += 1
94                    for rijec in Rezultat:
95                        if checkUTF8(Rijec[:2]) == False:
96                            if rijec[0] == pocetno_slovo:
97                                Lista.append(rijec)
98                        else:
99                            if rijec[:2] == pocetno_slovo:
100                               Lista.append(rijec)
101                   session.Rjecnik[pocetno_slovo] = Lista
102                   # Dodatna podjela
103                   if checkUTF8(Lista[0][:2]) == False and
                         checkUTF8(Lista[0][1:3]) == False:
104                       pocetna_slova = Lista[0][:2]
105                   else:
106                       pocetna_slova = Lista[0][:3]
107                   j = 0
108                   for Podrijec in Lista:
109                       if len(Podrijec) >= 2:
110                           if checkUTF8(Podrijec[:2]) == False and
                                 checkUTF8(Podrijec[1:3]) == False:
111                               if Podrijec[:2] != pocetna_slova:
112                                   pocetna_slova = Podrijec[:2]
113                                   j = 0
114                           else:
115                               if Podrijec[:3] != pocetna_slova:
116                                   pocetna_slova = Podrijec[:3]
117                                   j = 0
118                           if j == 0:
119                               Podlista = []
120                               j += 1
121                               for podrijec in Lista:
122                                   if len(podrijec) >= 2:
123                                       if checkUTF8(podrijec[:2])
                                             == False and checkUTF8(
                                             podrijec[1:3]) == False:
124                                           if podrijec[:2] ==
                                                 pocetna_slova:
125                                               Podlista.append(
                                                     podrijec)
126                                       else:
127                                           if podrijec[:3] ==
                                                 pocetna_slova:
```

```
128                                                 Podlista.append(
                                                        podrijec)
129                                 session.Rjecnik_Sub[pocetna_slova] =
                                        Podlista
130         # Inicijalizacija lista slova i prikaza (a-z ili z-a)
131         if session.poredak == "checked":
132             poredak = False
133         else:
134             poredak = True
135         Lista_Slova = session.Rjecnik.keys()
136         Lista_Slova.sort(key=str.lower)
137         # Generiraj output
138         Output = "<div id=\"slova\"><a id=\"slovo\" href=\"#\"
               onclick=\"ResultShowAll()\">SVE</a>   
               ; "
139         br = 0
140         for slovo in Lista_Slova:
141             Output += "<a id=\"slovo\" href=\"#\" onclick=\"
                   FilterResult("+"'"+slovo+"'"+")\">"+slovo+"</a>&
                   nbsp; "
142         Output += "</div><br><div id=\"tablica\"><table border
               =\"1px\" class=\"table table-hover\"><tr><th>Br.</th
               ><th>Riječ</th>"
143         if session.prikaz_ponovljivost == "checked":
144             Output += "<th>Pojavak</th>"
145         if session.prikaz_definicija == "checked":
146             Output += "<th>Definicija</th>"
147         Output += "<th><input type=\"checkbox\" id=\"selectall\"
                onclick=\"SelectAll()\"> Označi</th></tr>"
148         Lista_Slova.sort(key=str.lower, reverse=poredak)
149         Lista_Rijeci = []
150         for slovo in Lista_Slova:
151             for value in session.Rjecnik[slovo]:
152                 Lista_Rijeci.append(value)
153                 if session.prikaz_ponovljivost == "checked":
154                     pojavak = GetWordAppearance(value, Skup,
                            Lista_Rijeci_Skup_A, Lista_Rijeci_Skup_B)
155                 else:
156                     pojavak = ""
157                 if session.prikaz_lema == "checked":
158                     try:
159                         lema = zamjena(value, "dod1")
160                         if type(lema) is not str:
161                             lema = lema.encode("utf-8")
162                     except:
163                         lema = "#"
164                 else:
165                     lema = "#"
166                 if session.prikaz_definicija == "checked":
```

```
167                     try:
168                         definicija = db_struna.pretrazi(value, "
                               opis")
169                         if type(definicija) is not str:
170                             definicija = definicija.encode("
                                   utf-8")
171                     except:
172                         definicija = "/"
173                 else:
174                     definicija = "/"
175                 session.Podaci_Rjecnik[value] = {'pojavak':
                       pojavak, 'lema':lema, 'definicija':definicija
                       , 'selekcija':False}
176                 br += 1
177                 if br < session.broj_rijeci_ps + 1:
178                     Output += "<tr><td>"+str(br)+"</td><td>"+
                           value
179                     if session.prikaz_lema == "checked":
180                         Output += "<br><b>lema: </b><i>%s</
                               i></td>" %lema
181                     else:
182                         Output += "</td>"
183                     if session.prikaz_ponovljivost == "checked":
184                         Output += "<td>"+str(pojavak)+"</td>"
185                     if session.prikaz_definicija == "checked":
186                         Output += "<td>"+definicija+"</td>"
187                     Output += "<td><input type=\"checkbox\"
                           class=\"checkbox\" name=\""+value+"\"
                           onclick=\"UpdateDictionary(event)\"></td
                           ></tr>"
188         Output += "</table>"
189         # Pagination
190         session.lista_kompletna = Lista_Rijeci
191         session.temp_lista = Lista_Rijeci
192         session.temp_podslova = []
193         pages = (len(Lista_Rijeci)+1)/session.broj_rijeci_ps
194         mod = (len(Lista_Rijeci)+1)%session.broj_rijeci_ps
195         if mod != 0:
196             pages += 1
197         if pages > 1:
198             Output += """<br><a href=\"#\" onclick=\"SetPage('
                   previous', %i, 'slovo')\"><<< Prethodna stranica
                   </a>  Stanica:<input type=\"number\"
199                         id="pagination" value=\"1\" style=\"
                               width:40px;margin-top:5px;\"
                               onchange=\"SetPage('set', %i, 'slovo
                               ')\">/%i
200                         <a href=\"#\" onclick=\"SetPage('next',
                               %i, 'slovo')\">Sljedeća stranica
```

```
                                >>></a></div>""" %(pages,pages,pages
                                ,pages)
201             else:
202                 Output += "</div>"
203             # Povrat funkcije
204             return Output
205         except:
206             response.flash = "Greška: operacija nije uspjela zbog
                    neispravne selekcije!"
207             return "Selektirajte tekst(ove) i operaciju koju želite
                    izvršiti te pritisnite na gumb \"Stvori rječnik\"."
```

Primjer 6: Algoritam za stvaranje rječnika