SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Branimir Štivić

IMPLEMENTACIJA APLIKACIJE ZA GENERIRANJE GRAFIČKOG PRIKAZA AKADEMSKIH DRUŠTVENIH MREŽA

ZAVRŠNI RAD

Varaždin, 2013.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Branimir Štivić
Redoviti student
Broj indeksa: 39078/10-R
Smjer: Informacijski sustavi
Preddiplomski studij

IMPLEMENTACIJA APLIKACIJE ZA GENERIRANJE GRAFIČKOG PRIKAZA AKADEMSKIH DRUŠTVENIH MREŽA

ZAVRŠNI RAD

Mentor:
Tihomir Orehovački, mag. inf.

Varaždin, 2013.

Sadržaj

Popis slika

1. Uvod

U ovome radu opisan je razvoj aplikacije za generiranje grafičkog prikaza akademskih društvenih mreža uz pomoć C# .NET i OpenGL tehnologije. Analiza društvenih mreža jedna je od ključnih tehnika u modernoj sociologiji, a koristi se u antropologiji, biologiji, informacijsko-komunikacijskim i organizacijskim znanostima, društvenoj psihologiji i sociolingvistici. Grafički prikaz podataka od presudne je važnosti za znanstveno-istraživački rad u navedenim granama znanosti, a kao što stara izreka kaže „Slika vrijedi više od tisuću riječi“ tako i grafički prikaz podataka može pomoći prilikom otkrivanja uzoraka i novih spoznaja koje ne bi bilo moguće otkriti običnim promatranjem podataka. Aplikacija koja je implementirana omogućava pretraživanje i prikaz akademskih društvenih mreža u Republici Hrvatskoj te je primjer praktične primjene grafičkog programiranja u svrhu izrade alata za analizu društvenih mreža. Aplikacija se najvećim dijelom usredotočuje na male i srednje egocentrične mreže sa naglaskom na traženog autora. Generiranim grafom moguće je manipulirati u realnom vremenu, što ujedno poboljšava kvalitetu interakcije s korisnikom. Uz opis koraka izrade aplikacije, opisane su i tehnologije koje su korištene prilikom izrade projekta, a implementacija aplikacije je podijeljena u tri faze razvoja. Prva faza se odnosi na prikupljanje i obradu podataka sa web mjesta projekta[1] *„Hrvatska znanstvena bibliografija“*. Druga faza sastoji se od izrade podatkovnog modula koji se koristi za pretraživanje baze autora i radova te kreiranje grafa sudjelovanja autora. Treća faza je ujedno i najzahtjevnija faza implementacije aplikacije te se sastoji od grafičkog prikaza generiranih podataka, implementacije manipulacije grafom akademske društvene mreže i realizacije upravljanja grafom u realnom vremenu.

[1] Web adresa *Hrvatska znanstvena bibliografije* je bib.irb.hr, a preuzimanje podataka izvršeno je u veljači, 2013.

2. Akademske društvene mreže

Proučavanje mreža ima dugu povijest u matematici i ostalim znanstvenim disciplinama. Još od Leonarda Eulera (18. st.) i njegovog poznatog problema „Mostova Königsberga“, znanstvenici se bave proučavanjem mreža. Euler je 1756. godine pružio dokaz za tada popularnu mozgalicu o postojanju puta kojim bi se prešlo svih sedam mostova Königsberga, a da se svaki most posjeti točno jednom. Problem je riješio uz pomoć grafova, a potom je takav put koji prolazi svim rubovima točno jednom nazvan Eulerov put (Biggs, Lloyd, & Wilson, 1976, str. 3). Newman i suradnici (2006) smatraju Eulerov dokaz prvim teoremom u sada već visoko razvijenom području diskretne matematike poznatom pod nazivom „teorija grafova“. To je područje koje je u protekla tri stoljeća postalo glavni matematički jezik za opisivanje svojstava mreža. Mreža je u njenoj najjednostavnijoj formi skup diskretnih elemenata (vrhova) i veza (rubova) koji povezuju elemente. Skup vrhova i rubova može prikazivati gotovo bilo što, a apstrakcijom detalja problema teorija grafova sposobna je opisati važna topološka svojstva s jasnoćom koja bi bila nemoguća zadržavanjem svih detalja (Newman, Barabási, & Watts, 2006, str. 1-4). Važnost mrežne strukture dokazuje i njezina uporaba kroz različite znanstvene discipline. Znanstvenici Jackson i Watts (2002) tvrde da je „Mrežna struktura bitna za određivanje ishoda velikog broja važnih socijalnih i ekonomskih veza. Na primjer, mreža ima ključnu ulogu u određivanju izmjene informacija“ (Jackson & Watts, 2002, str. 265-266).

Nadalje, pojmovi bitni za nastavak istraživanja su društvene mreže i specijalizirani oblik društvenih mreža pod nazivom akademske društvene mreže. „Društvene mreže mogu biti definirane u kontekstu sustava poput Facebook-a, koji su eksplicitno dizajnirani za društvenu interakciju ili kao ostale stranice poput Flickr-a, koje su dizajnirane za različite servise poput dijeljenja sadržaja, ali koje ujedno omogućavaju široku razinu društvene interakcije. Generalno govoreći, društvena mreža se definira kao mreža međudjelovanja ili odnosa gdje se čvorovi sastoje od sudionika, a rubovi od međudjelovanja ili odnosa između sudionika“ (Aggarwal, 2011, str. 1-2). Akademska društvena mreža je specijalizirani oblik društvene mreže, a sudionici u toj mreži su isključivo akademski građani čije se međudjelovanje svodi na koautorstvo i odabir referenci. Akademski građani kroz svoje publikacije formiraju određenu društvenu mrežu. Autori odabiru koautore i reference na radove drugih autora, što se razlikuje od ostalih popularnih društvenih mreža gdje sudionici dijele svoja mišljenja, poveznice ili fotografije (Fu, Song, & Chiu, 2013, str. 1-3). Autori povećavaju povezanost uz pomoć koautorstva s drugim autorima, a osobito kada surađuju s jače povezanim autorima. U akademskoj zajednici

uobičajeno je steći kratku impresiju o autorovom istraživanju uz pomoć jednostavne statistike njegovih publikacija.

Do publikacija je moguće doći preko raznih Internet servisa poput Google Scholar, ACM Digital Library, Agricola, ArXiv i dr. Google Scholar je tražilica znanstvenih radova koja pruža mogućnost pretraživanja brojnih stručnih podataka i izvora. ACM Digital Library je baza podataka koja sadrži cjelovite tekstove svih radova koji su izdani od strane ACM-a te bibliografske citate glavnih izdavača iz područja računarstva. ArXiv je servis koji nudi pristup velikom broju radova iz područja fizike, matematike, računalstva, kvantitativne biologije i statistike. Postoji veliki broj baza akademskih radova, a uz njih je bitno spomenuti i ostale servise koji pružaju različiti skup funkcionalnosti. ArnetMiner je servis koji pruža mogućnost pretraživanja, rudarenja i analize akademskih društvenih mreža. Navedeni servis prikuplja podatke s različitih izvora te tako izgrađuje profil za svakog autora. Važno je navesti da ArnetMiner omogućava vizualni prikaz statističkih podataka i grafa sudjelovanja traženog autora. ResearchGate je društvena mreža koja znanstvenicima pruža mogućnost učitavanja vlastitih radova te povezivanje i kolaboraciju s ostalim znanstvenicima. Mendeley je alat koji prvenstveno služi za upravljanje bibliografijom i citatima prilikom izrade znanstvenih radova, ali uz to omogućava pretraživanje radova i izradu javnih akademskih profila.

Za promatranje akademskih društvenih mreža potrebne su metrike koje će autorima pridružiti određene karakteristike. Postoje razne metrike promatranja akademskih društvenih mreža, a u nastavku su navedene tri grupe metrika koje su razvili Fu i suradnici (2013, str. 5-6).

U prvoj grupi nalaze se *Metrike temeljene na radovima*. U slučaju ove grupe metrika, svaki rad ima vrijednost definiranu putem određene metrike. Za prvu grupu Fu i suradnici (2013, str. 5-6) navode tri metrike:

a. *Broj citata (engl. Citation count, CC)*
b. *Balansirani broj citata (engl. Balanced citation count, BCC)*
c. *Vrijednost citata (engl. Citation Value, CV)*

Kod metrike *Broj citata*, autoru se pridjeljuje potpuni broj citiranja rada, dok se kod metrike *Balansirani broj citata* autoru pribraja broj citiranja podijeljen s brojem koautora na pojedinom radu. Metrika *Vrijednost citata* koristi iterativno izračunavanje citata između svih radova te broj citiranja dijeli na jednake dijelove između svih koautora na pojedinom radu.

Druga grupa su *Metrike temeljene na autorima*. U ovom slučaju metrike se računaju direktno pomoću veze autor - autor. U toj grupi metrika autori navode tri metrike:

a. *Utjecaj (engl. Influence)*
b. *Sljedbenici (engl. Followers)*
c. *Veze (engl. Connections)*

Metrika *Utjecaj* računa se tako da se svaki puta kada autor *i* citira rad autora *j*, vrijednost dodjeljuje autoru *j*, ali podijeljena na jednake dijelove između svih autora navedenog rada. Prilikom izračuna metrike *Sljedbenici* promatra se samo direktna veza između dva autora te se autoru dodjeljuje vrijednost samo jednom za istog autora, neovisno o broju radova u kojima su ta dva autora sudjelovala. Kod metrike *Veze,* autoru i njegovom koautoru dodjeljuje se jednaka vrijednost za svako koautorstvo na određenom radu.

Posljednja grupa su *Metrike temeljene na mjestima i autorima.* U ovoj kategoriji autori definiraju samo jednu metriku. Metrika *Izloženost (engl. Exposure)* računa se uz pomoć direktnih veza autor – autor, autor - mjesto, mjesto - autor i mjesto – mjesto. Svaki puta kada autor *i* napiše rad koji je objavljen u mjestu izdavanja *k,* distribuira svoj utjecaj mjestu izdavanja *k*. Kada mjesto izdavanja *k* objavi rad autora *i*, utjecaj se ravnomjerno raspodjeljuje između svih koautora na radu.

Iz priloženog možemo vidjeti da postoje različite metrike za analizu akademskih društvenih mreža. Ista predstavlja ključnu tehniku u modernoj sociologiji, a primjenjuje se i u antropologiji, biologiji, komunikacijskim, informacijskim i organizacijskim znanostima, društvenoj psihologiji i sociolingvistici. Analitičari većinom promatraju potpune mreže (sve veze između pojedinaca u određenoj populaciji) ili osobne mreže (još poznate i pod pojmom egocentrične mreže, a prikazuju mrežu povezanosti pojedinca sa njegovom okolinom). Razlika između potpunih mreža i osobnih mreža ponajviše ovisi o mogućnostima analitičara u prikupljanju podataka (Analysis, 2010, str. 2). Iz navedenog proizlazi da se analiza društvenih mreža primjenjuje u širokom krugu znanstvenih disciplina. Prilikom analize u središte promatranja se stavlja struktura odnosa između pojedinaca. Odnosi koji se mogu kretati od neformalnih poznanstava do bliskih veza označavaju se i mjere u svrhu razumijevanja međudjelovanja između različitih članova mreže (Serrat, 2009, str. 1-2).

Jako bitan dio analize društvenih mreža čini vizualni prikaz prikupljenih podataka. Vizualizacija mrežnih struktura uz pomoć dijagrama predstavlja najčešće korištenu tehniku prilikom analize. Takvi dijagrami pružaju kompleksnu sliku interakcije između pojedinaca ili organizacija (Hogan,

Carrasco, & Wellman, 2007, str. 1). Da su vizualizacije korisne za podizanje ljudske percepcije prilikom pronalaženja uzoraka, pokazuje i veliki broj uspješnih projekata na kojima su se koristile. Proces izrade i istraživanja vizualizacije mreže nije trivijalan, a zbog potrebe za analizom sve kompleksnijih mrežnih struktura te poboljšanja prikaza društvenih mreža pojavljuju se razni alati. Oni doprinose novim mogućnostima analize te je uvelike olakšavaju. Zbog poboljšanja istraživačkog procesa alati moraju omogućavati interakciju s grafom i analizu u realnom vremenu. Takvi alati ujedno moraju biti tehnički ispravni i vizualno atraktivni (Bastian, Heymann, & Jacomy, 2009, str. 1).

Razvojem računalne tehnologije stvara se mogućnost obrade, izračuna i prikaza sve većih grafova u realnom vremenu koji nadilaze čak i nekoliko milijuna čvorova. Za takav prikaz potrebni su moćni alati. Huisman & van Duijn (2003) u svojem radu navode i testiraju najpoznatiji komercijalni, ali i besplatni softver za analizu društvenih mreža. Neki od najpoznatijih alata za analizu društvenih mreža koji imaju mogućnost vizualne reprezentacije mreža su Agna, Blanche, InFlow, KliqFinder, MultiNet, NetDraw, NetMiner, Pajek, SNAFU, UCINET, Gephi, Graphi i dr.

2.1 Rudarenje akademskih društvenih mreža

U ovome poglavlju opisano je rudarenje akademskih društvenih mreža na primjeru projekta ArnetMiner[2]. Korisnicima servisa za rudarenje akademskih društvenih mreža u većini slučajeva nisu dovoljne samo osnovne informacije o autorima, stoga je potrebno dohvatiti podatke iz različitih izvora te ih pravilno strukturirati (Tang, Zhang, Yao, & Li, 2008, str. 1). ArnetMiner je servis koji prikuplja podatke iz različitih izvora te tako izgrađuje profil za svakog autora. Prikupljeni podaci se naknadno modeliraju u različite oblike prikaza i korisniku pružaju grafički prikaz istih. Prethodna istraživanja znanstvenika na projektu ArnetMiner poslužila su kao temelj funkcionalnosti aplikacije. Tang i suradnici (2008, str. 2) navode arhitekturu i glavne komponente podijeljene u kategorije ArnetMiner sustava:

1. **Izvlačenje**: *(engl. Extraction)* fokusira se na izvlačenje znanstvenih profila i radova s Web-a.
2. **Integracija:** *(engl. Integration)* integrira izvučene profile znanstvenika i radova po imenu autora. Prilikom integracije javlja se poseban problem dvosmislenosti imena (engl. the name ambiguity problem) za kojeg je potrebno razviti posebne algoritme. Integrirani podaci spremaju se u bazu podataka istraživačke mreže znanja (engl. Researcher Network Knowledge Base, RNKB**).**
3. **Spremište i pristup:** *(engl. Storage and Access)* pruža spremište i indeksiranje podataka u RNKB.
4. **Modeliranje:** *(engl. Modeling)* omogućava istovremeno modeliranje različitih vrsta podataka.
5. **Servisi pretraživanja:** *(engl. Search Services)* baziran na rezultatu modeliranja pruža veći broj servisa pretraživanja.

Faze izrade projekta podijeljene su po funkcionalnostima/modulima prema navedenim komponentama ArnetMiner sustava. Neke od navedenih komponenti nije bilo potrebno razvijati, dok je neke komponente bilo potrebno detaljnije razviti pa tako komponenta integracije, zbog izvlačenja podataka iz samo jednoga izvora, nije od značajne važnosti za završni rad. Međutim, funkcionalnost grafičkog prikaza bilo je potrebno detaljnije razviti.

[2] Sustav za izvlačenje i rudarenje akademskih društvenih mreža, http://arnetminer.org/

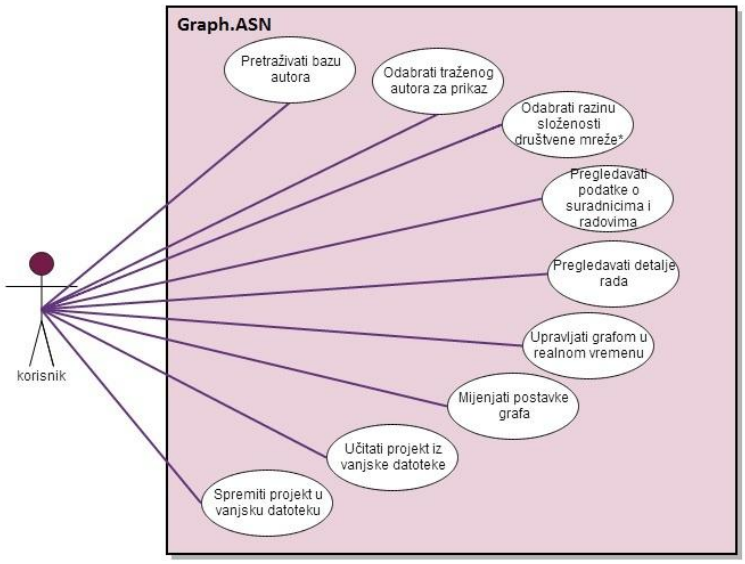
3. Opis aplikacije

Ovaj dio rada usmjeren je na praktičnu primjenu usvojenog znanja iz područja prikupljanja i obrade podataka, grafičkog programiranja i izrade softvera u cilju izrade aplikacije za generiranje grafičkog prikaza akademskih društvenih mreža temeljenom na modelu malih i srednjih egocentričnih društvenih mreža. Aplikacija omogućava pretraživanje baze autora i radova prema imenu i prezimenu autora s ciljem analize podataka akademske društvene mreže (grafički prikaz komunikacije i sudjelovanja autora, pregled svih sudjelovanja i radova te detaljan prikaz rada). Za razliku od popularnih stolnih aplikacija za prikazivanje i analizu grafova društvenih mreža, aplikacija omogućava generiranje podataka o povezanosti između traženog autora i njegovih koautora. Podaci potrebni za generiranje grafova nalaze se u lokalnoj bazi podataka. Kao izvor za izvlačenje podataka akademske društvene mreže, odabran je web projekt *„Hrvatska znanstvena bibliografija"*[3]. To je projekt online bibliografije s mogućnošću unosa i pretraživanja putem web sučelja, a nastao je u suradnji stručnjaka Ministarstva znanosti, obrazovanja i sporta i Knjižnice Instituta Ruđer Bošković. Aplikacija ima dostupnu lokalnu kopiju bibliografskih podataka (oko 660 000 bibliografskih unosa) i podataka o autorima (oko 330 000 autora) iz Republike Hrvatske koji su prikupljeni izvlačenjem iz web mjesta (engl. web scraping). Zbog ograničenih resursa i prihvatljivih performansi, projekt je usmjeren samo na radove i autore iz Republike Hrvatske. Aplikacija je razvijena na Windows platformi u programskom jeziku C#, a koristi OpenGL aplikacijsko programsko sučelje (engl. application programming interface, API) za generiranje grafičkog prikaza društvenih mreža. Unatoč tome što aplikacija sadrži i nestandardnu funkcionalnost alata za analizu društvenih mreža (aplikacija omogućava i generiranje podataka o društvenoj mreži po zadanom autoru iz baze podataka), uspješno su sačuvane zadovoljavajuće performanse uz primjenu višedretvenosti (engl. multithreading) i generiranja grafičkog prikaza uz pomoć grafičke procesorske jedinice (engl. graphics processing unit, GPU). Aplikacija uz mogućnost generiranja grafa povezanosti autora u društvenoj mreži omogućava i pohranu trenutnog projekta s postavkama grafičkog prikaza u vanjsku XML datoteku te naknadno učitavanje projekta iz vanjske datoteke, što znatno ubrzava rad aplikacije.

[3] Pristupano 23. srpnja, 2013. na adresi: http://bib.irb.hr/o_projektu

3.1 Opis funkcionalnosti aplikacije

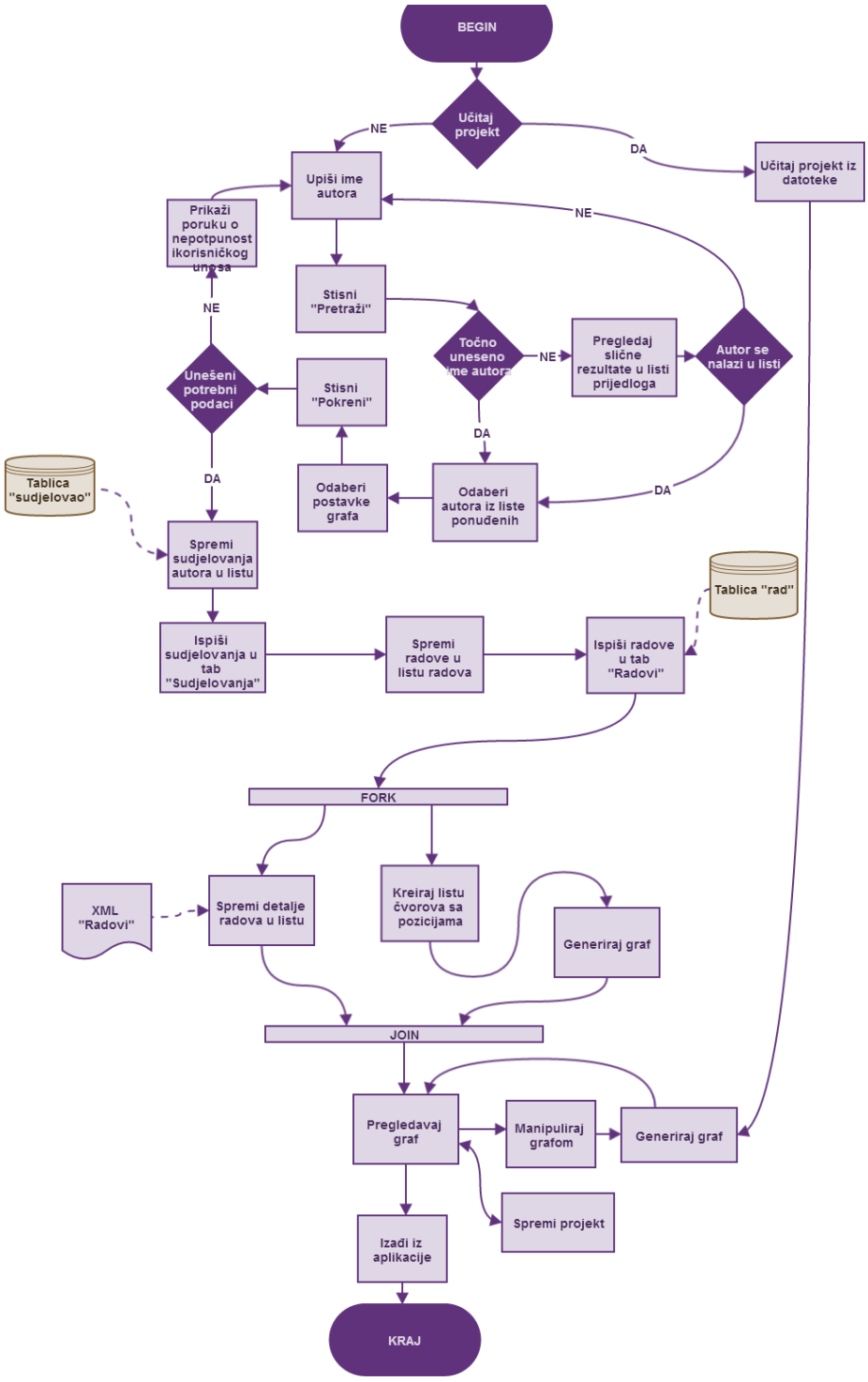
Ovo potpoglavlje sadrži opis svih funkcionalnosti aplikacije. Na slici 3.1 prikazan je dijagram slučajeva korištenja aplikacije koji je u nastavku teksta detaljno opisan. Na dijagramu korištenja nalaze se korisnik i aplikacija „graphASN“. Korisnik može pretraživati bazu podataka autora. Ukoliko korisnik unese pogrešno ime ili prezime autora, aplikacija vraća popis najsličnijih imena. Nakon uspješnog pronalaženja autora, korisnik može odabrati autora za kojeg želi prikazati graf. Nakon odabira autora, korisnik može pregledavati sve koautore i radove na kojima je traženi autor sudjelovao. Sa klikom na određeni rad prikazuju se detalji rada. Korisnik može mišem upravljati pozicijom i veličinom grafa u realnom vremenu, upravljati kvalitetom veza i čvorova, veličinom veza, čvorova i teksta. Za vizualizaciju grafa moguće je odabrati više tema prikaza u različitim bojama te je moguće odabrati više različitih algoritama pozicioniranja grafa u 3D prostoru. Naposljetku, projekt je moguće spremiti u vanjsku XML datoteku i naknadno ga učitati po želji.



Slika 3.1 Dijagram slučajeva korištenja

3.2 Dijagram toka

U ovom potpoglavlju opisan je način na koji aplikacija kreira vizualizaciju sudjelovanja autora i prikazuje detalje o traženom autoru. Na slici 3.2 nalazi se dijagram toka koji detaljnije prikazuje rad aplikacije.



Slika 3.2 Dijagram toka

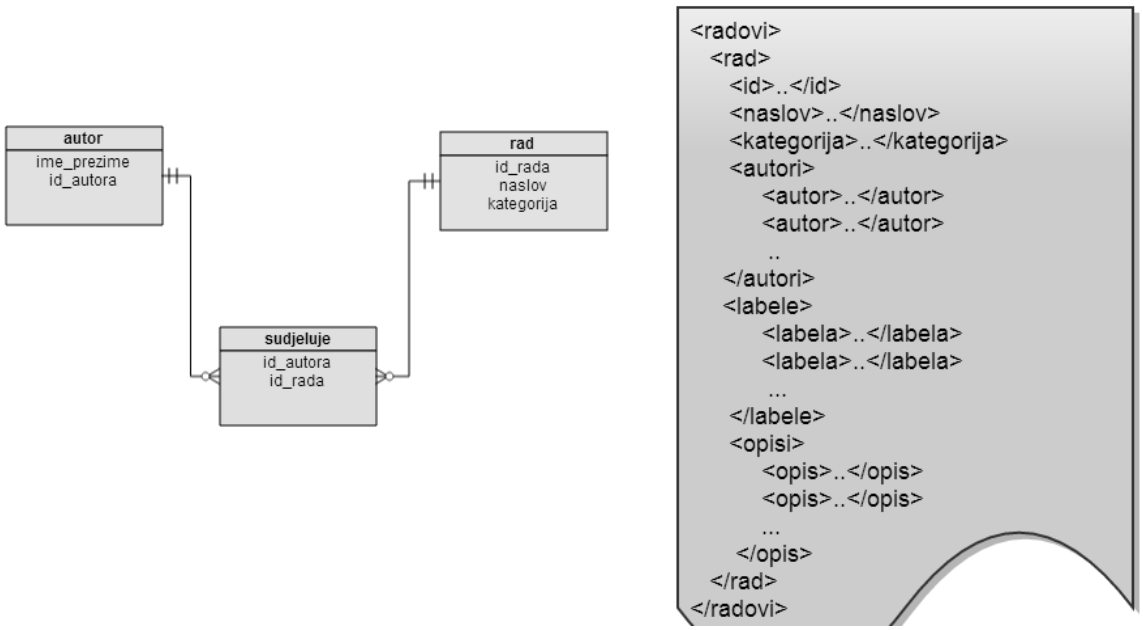
Na početku korištenja aplikacije korisnik može početi pretraživati autore ili učitati već gotov projekt iz XML datoteke. Ukoliko korisnik odluči generirati novi graf, prvo mora upisati ime autora. Ukoliko ime autora nije točno, aplikacija generira listu autora sa sličnim imenom i prezimenom, a u međuvremenu se korisnik obavještava o postojanju autora u bazi uz pomoć

bojanja polja za tekstualni unos. Nakon uspješnog odabira autora i postavki, uz pomoć tablice iz baze podataka „sudjelovao“ u listu se spremaju svi koautori odnosno suradnici autora za kojeg je izvršeno pretraživanje. Nakon pronalaska svih autora koji su surađivali s traženim autorom, imena svih pronađenih autora se ispisuju u karticu „Sudjelovanja“. Kartica „Sudjelovanja“ sadrži popis svih autora na grafu, broj koautorstava za svakog autora te autora prema kojem je pretraživano. Prilikom označavanja autora u kartici „Sudjelovao“, čvor označenog autora poprima drugu boju. Slijedi spremanje svih radova iz tablice „rad“ u listu. Radovi se nakon dohvaćanja prikazuju u kartici „Radovi“. Zbog poboljšanja performansi, izvršavanje aplikacije se nakon toga dijeli na dvije dodatne dretve. Prva dretva raščlanjuje (engl. parsing) XML dokument s detaljima svih radova i sprema u listu detalje svih radova odabranog autora. Ovaj postupak opterećuje procesor, a ovisno o brzini procesora traje u prosjeku od 60 do 90 sekundi zbog veličine XML datoteke (818 MB). Postupak parsiranja bilo je potrebno odvojiti u dretvu i tako dopustiti korisniku interakciju s korisničkim sučeljem[4]. Paralelno s prvom dretvom izvršava se i druga dretva koja je zadužena za kreiranje liste čvorova, generiranje pozicioniranja čvorova u 3D prostoru i početno generiranje grafa. Nakon generiranja grafa korisnik ima mogućnost pregledavanja grafa i interakcije u realnom vremenu. Nakon manipulacije grafom, poziva se ponovno generiranje grafa. Korisnik može spremiti projekt u vanjsku XML datoteku i izaći iz aplikacije. Prilikom učitavanja projekta svi podaci trenutnog projekta se brišu (ukoliko postoje) te se u liste objekata učitavaju podaci potrebni za rad aplikacije.

3.3 Opis vrsta podataka

Za funkcioniranje aplikacije potrebna je baza podataka i XML dokument čiju je struktura prikazana na slici 3.3. Strukturu baze podataka opisuje vrlo jednostavan ERA dijagram, dok strukturu XML dokumenta opisuju čvorovi. Zbog poboljšanja performansi, cijeli XML dokument s autorima i dio dokumenta s radovima pohranjen je u bazu podataka. Podaci ključni za rad aplikacije nalaze se u tri tablice koje imaju mali broj atributa. XML dokument „radovi“ sadrži varijabilni broj atributa. Tablica autor sadrži atribute „ime_prezime“ i „id_autora“. Povezana je vezom više-više s tablicom “rad“ koja ima atribute „id_rada“, „naslov“ i „kategoriju“. Detalji o obradi podataka nalaze se u potpoglavlju 5.1.

[4] U Windows Forms aplikacijama korisničko sučelje se pokreće s glavnom dretvom korisničkog sučelja. Ukoliko se iz dretve korisničkog sučelja izvršava vremenski zahtjevna operacija tada korisnik ne može imati fluidnu interakciju s korisničkim sučeljem sve dok se ta operacija ne izvrši do kraja.



Slika 3.3 ERA dijagram

4. Korištene tehnologije

U ovom poglavlju opisane su korištene tehnologije prilikom implementacije aplikacije. Za pojedine faze implementacije korištene su različite tehnologije. Tako je za prvu fazu korišten programski jezik Python i okvir Scrapy, koji služi za izvlačenje podataka. U drugoj fazi je korišten programski jezik C# i sustav za upravljanje bazom podataka SQLite. U trećoj fazi implementacije korišteni su OpenGL i TAO Framework.

4.1 Python-Scrapy

Python zbog svoje snage i fleksibilnosti predstavlja idealan izbor za rješavanje problema u različitim domenama. Uz navedeni programski jezik korišteno je radno okruženje (engl. framework) Scrapy. Ono služi za prikupljanje podataka (engl. screen scraping) i web pretraživanje (engl. web crawling). Kao što je navedeno na službenoj stranici, „Scrapy se može upotrebljavati za različite namjene od rudarenja podataka (engl. data mining) do nadgledanja (engl. monitoring) i automatiziranog testiranja (engl. automated testing)“ (Scrapy, 2013). Prilikom web pretraživanja podaci se izvlače s web stranica uz pomoć HTML-a, XPatha i Regex-a te se naposljetku spremaju u datoteku (JSON, XML ili CSV).

4.2 C# .NET

Za razvoj aplikacije odabran je programski jezik C#. Prema službenom opisu (Microsoft, 2013) C# je objektno-orijentirani programski jezik koji služi za razvoj aplikacija u .NET okruženju. Navedeni okvir predstavlja integralnu komponentu Windows platforme, a sastoji se od sustava virtualnog izvođenja pod nazivom Common Language Runtime (CLR) i zajedničkog skupa biblioteka klasa. „CLR je komercijalna implementacija Common Language Infrastructure međunarodnog standarda koji je podloga za kreiranje izvršnog i razvojnog okruženja u kojem jezici i biblioteke međusobno djeluju. Programski kod napisan u jeziku C# prevodi se u posrednički jezik (engl. Intermediate Language, IL) koji podliježe Common Language Infrastructure specifikaciji“ (Microsoft, 2013). Upravo zbog kvalitete i snage C# .NET radnog okruženja, navedena tehnologija korištena je prilikom izrade glavnog dijela aplikacije.

4.3 SQLite

SQLite je sustav za upravljanje bazom podataka, koji za razliku od ostalih sustava nema odvojeni serverski proces (Hipp, 2013). Datotečni format baze podataka neovisan je o platformi, što znači da je moguće kopirati datoteku baze podatka između 32-bitnih i 64-bitnih platformi (Hipp, 2013). SQLite je vrlo kompaktan i mali sustav za upravljanje bazom podataka veličine svega 500 KB koji je lako prenosiv i popularan kod razvoja mobilnih aplikacija. Razlozi zbog kojeg je odabran SQLite u svrhu upravljanja bazom podataka su jednostavnost, veličina i lakoća ažuriranja baze podataka. Za pristup bazi podataka razvijena je posebna klasa u programskom jeziku C# koja omogućava čitanje i pisanje u bazu podataka.

4.4 OpenGL

OpenGL je skraćenica za Open Graphics Library. Navedeno predstavlja „specifikaciju API-a za renderiranje većim dijelom 3D grafike. OpenGL implementacije su biblioteke koje implementiraju API definiran specifikacijom“ (The Khronos Group, 2013). OpenGL je odabran za izradu grafičkog modula aplikacije zbog svoje snage, podržanosti i vrlo opširne dokumentacije. Navedeni API također koristi GPU za prikazivanje grafičkog sadržaja baziran na hardverskom ubrzanju. Prema službenim stranicama Khronos grupe, „OpenGL je jedan od vodećih okruženja za razvoj prenosivih interaktivnih 2D i 3D grafičkih aplikacija. Još od njegovog predstavljanja 1992. godine, OpenGL je postao jedan od najkorištenijih i najpodržanijih 2D/3D grafičkih aplikacijskih programskih sučelja (engl. Application Programming Interface, API)“ (The Khronos Group, 2013). OpenGL je razvijen od strane Silicon Graphics Inc. i služio je za dizajn potpomognut računalom (engl. Computer-aided Design, CAD), virtualnu stvarnost, znanstvene i informacijske vizualizacije, simulacije letova i razvoj računalnih igara. Razvoj OpenGL-a preuzeo je neprofitni tehnološki konzorcij Khronos Group. Posljednja verzija je 4.4 i objavljena je 22. srpnja 2013. godine. OpenGL pruža programskim inženjerima pristup geometrijskim i slikovnim primitivama, listama prikaza, transformacijama modeliranja, osvjetljenju i teksturama, uglađivanju rubova (engl. anti-aliasing), prozirnosti (engl. transparency) i još velikom broju ostalih značajki (The Khronos Group, 2013).

4.5 OpenGL i C#

OpenGL je pisan u C jeziku i kao takav nije dostupan objektno orijentiranim jezicima poput C#. Da bi bilo moguće koristiti OpenGL API u C# jeziku, potrebno je implementirati tzv. omotač (engl. wrapper), koji će sve funkcionalnosti OpenGL-a prilagoditi objektno orijentiranom jeziku. Ne postoji službena potpora za OpenGL omotače, ali postoje vanjske programske komponente koje su razvili neovisni timovi. Neke od prednosti korištenja OpenGL API-a u programskom jeziku C# su brži razvoj osnovnog dijela aplikacije, lakše ugrađivanje korisničkih kontroli i objektni pristup razvoja, dok su glavni nedostaci izostanak službene potpore, loše performanse kod zahtjevnijih aplikacija, nepotpuna implementiranost omotača i manjak dokumentacije.

4.6 TAO Framework

TAO Framework je razvojno okruženje koje služi kao omotač (engl. wrapper) OpenGL API-a u objektno orijentiranom okruženju. Uz pomoć navedenog razvojnog okruženja moguće je razvijati Windows Forms aplikacije s ugrađenom OpenGL kontrolom. Čitavi okvir je veličine 33 MB i sastoji se od većeg broja pomoćnih biblioteka i primjera u obliku Visual Studio projekta. Pored toga, TAO Framework pruža omotač za OpenGL, FreeGlut, Lua, Glfw, Sdl, Cg, FFmpeg, FreeType, Platform.X11, Ode, DevIl, OpenAI, PhysFs, Platforms.Windows. Manjak dokumentacije može se pripisati prestanku razvoja projekta koji je nastavljen u obliku OpenTK eksperimentalne grane.

Neke od glavnih prednosti: jednostavna ugradnja u Visual Studio 2012 integrirano razvojno okruženje, lakoća razvoja Windows Forms aplikacija, implementacija omotača za FreeGlut i FreeType.

Nedostaci: Tao Framework projekt nije više aktivan, a posljednja verzija je izašla 2008. godine. Najveći nedostatak je manjak potpune implementacije biblioteke Tao.Platforms.Windows.Gdi. Nisu implementirane sve vrste formatiranja znakova (podržan je samo Gdi.ANSI_CHARSET) kod naredbe Gdi.CreateFont koja je ključna za izradu fonta kod renderiranja „Bitmap“ i „Outline“ fontova u OpenGL-u. To znači da grafički dio aplikacije ne podržava hrvatske znakove i ujedno je ogromni nedostatak okvira. Nije implementiran sustav obavijesti o pogreškama u razvoju OpenGL programskog koda, što u većini slučajeva rezultira rušenje aplikacije bez ikakve poruke o pogrješki.

5. Opis razvoja aplikacije

U ovome poglavlju opisan je razvoj aplikacije kroz faze. Prva faza je prikupljanje, obrada i spremanje podataka autora i radova s web mjesta bib.irb.hr. Druga faza je izrada podatkovnog modula, odnosno izrada okvira aplikacije i funkcionalnosti pretraživanja autora, izrada osnovnih klasa za manipulaciju podataka o autorima i radovima, kreiranje grafa sudjelovanja, prikazivanje podataka o autoru i radovima te spremanje i učitavanje projekta u vanjsku datoteku. Konačno, u trećoj je fazi ugrađen grafički kontekst OpenGL kontrole u formu aplikacije te su razvijeni algoritmi za pozicioniranje grafa, vizualno uređenje grafa, ispis naziva autora na grafu, izmjenu postavki grafa i manipulaciju grafom.

5.1 Prikupljanje i obrada podataka

Za izvor podataka akademske društvene mreže odabran je web projekt *„Hrvatska znanstvena bibliografija“* koji se nalazi na adresi http://bib.irb.hr. Postoji nekoliko glavnih čimbenika zbog kojih je odabran navedeni projekt znanstvene bibliografije. Glavni razlog je ograničenost resursa. Međunarodne znanstvene bibliografije imaju u svojoj bazi i nekoliko milijuna autora i radova, dok *„Hrvatska znanstvena bibliografija*“ u bazi ima oko 660 000 radova i 330 000 autora, što je i dalje jako velika količina podataka koju treba obraditi i pretražiti. Neki od ostalih razloga zbog kojih je odabran navedeni projekt su: uočene pravilnosti u radu web stranice, vrlo uredno organizirano i jednostavno web mjesto te verzija za ispis koja je jednostavna za izvlačenje podataka. Nakon izvlačenja, podatke je bilo potrebno raščlaniti i jedan dio unijeti u bazu podataka, dok je drugi dio zbog veličine trebalo optimizirati za čitanje iz XML datoteke. Nakon prikupljanja, pripreme i obrade podataka bilo je potrebno implementirati skriptu za unos podataka u bazu. Nakon završetka faze prikupljanja i obrade podataka bilo je moguće nastaviti s izradom glavne forme aplikacije, izradom podatkovnog modula, kreiranje grafa sudjelovanja te pretraživanja i ispisa podataka na glavnoj formi.

5.2 Izvlačenje

Kako bi se bolje upoznali sa pojmovima izvlačenje (engl. scraping) i web pretraživanje (engl. web crawling), u nastavku slijede opisi i definicije pojmova iz različitih izvora. Moody i suradnici (2003) navode pojam web izvlačenja u kontekstu izvršavanja automatiziranja upita koji preuzimaju rezultat formatiran u HTML obliku te ga raščlanjuju u strukturirani oblik. Najveći

problem kod web izvlačenja predstavlja izmjena strukture web mjesta s kojeg se preuzimaju HTML stranice. Ukoliko dođe do navedene izmjene, vrlo je vjerojatno da programski kod namijenjen web izvlačenju navedenog web mjesta neće biti djelotvoran (Moody & Palomino, 2003, str. 2). Chang i suradnici (2003, str. 2) objašnjavaju pojam izvlačenje informacija (engl. information extraction, IE) koji u kontekstu rudarenja Web-a označava automatizirani prijelaz ulaznih web stranica u strukturirane podatke. Najveći problem kod sofisticiranih aplikacija za rudarenje web mjesta predstavlja skupo održavanje zbog učestale izmjene strukture i prilagodbe različitim formatima web mjesta (Chang, Kayed, Girgis, & Shaalan, 2003, str. 2). Dakle, web izvlačenje možemo smatrati izvršavanjem neke vrste programskog koda koji služi za izvlačenje informacija (IE), kao ulaz prima stranice u HTML obliku, a kao izlaz vraća strukturirane podatke.

Boldi i suradnici (2003) definiraju pojam web pretraživanja (engl. web crawling) kao računalne programe koji služe za preuzimanje dokumenata s Interneta (Boldi, Codenotti, Santini, & Vigna, 2003, str. 1). Nadalje, Cothey (2004, str. 1) je definirao web pretraživač kao računalni program koji se kreće od čvora[5] do čvora uz pomoć prikupljanja hiperveza te tako određuje rubove web grafa. Alternativna definicija web pretraživanja (engl. web crawling) glasi: „Web pretraživanje je tehnika prikupljanja podataka koja podržava strukturnu i informacijsku analizu web grafova“ (Cothey, 2004, str. 1).

U nastavku ovoga potpoglavlja slijedi praktična primjena prethodno navedenih pojmova. Razvojno okruženje Scrapy služi za automatizirano pretraživanje i izvlačenje web mjesta. U kontekstu okruženja Scrapy potrebno je definirati takozvanu programsku funkcionalnost „spider“. Navedena funkcionalnost predstavlja implementacija automatiziranog web pretraživača u sklopu programskog jezika Python. Da bismo uspješno implementirali automatizirano pretraživanje web mjesta, potrebno je prvo uključiti potrebne biblioteke koje su navedene u sljedećem programskom kodu:

```
import re          #naredba za uključivanje biblioteka
from scrapy.spider import BaseSpider
from scrapy.selector import HtmlXPathSelector
from scrapy_bib.items import bibItem
```

Zatim definiramo klasu koja je tipa „BaseSpider“ (klasična implementacija pauka). Postavljamo ime pauka, domenu i početni URL. Kao što se može uočiti, URL za pojedinačni rad glasi

[5] čvor može biti veza na web stranicu ili dokument

„http://bib.irb.hr/lista-radova?autor=X“ gdje je „X“ jedinstveni identifikator autora, što uvelike olakšava implementaciju web pretraživača. Kao što je vidljivo u programskom kodu koji se nalazi u nastavku, uz pomoć „for“ petlje prolazi se kroz sve URL adrese i prikupljaju se svi autori koji postoje u bazi podataka.

```
class bibSpider(BaseSpider):
    name = 'bib_spider'
    allowed_domains = 'bib.irb.hr'   #adresa web mjesta
    start_urls = ['http://bib.irb.hr/lista-radova?autor=%d' % (i) for
i in range(1,400000)]
```

Prilikom web pretraživanja potrebno je raščlaniti preuzeti HTML dokument, a nakon raščlanjivanja potrebno je prikupljene podatke pravilno strukturirati. Izvlačenje podataka iz pojedinih HTML elementa implementira se uz pomoć Xpath-a, a programski kod koji se nalazi u nastavku služi za izvlačenje preuzete HTML stranice, njezino raščlanjivanje i naposljetku strukturiranje prikupljenih podataka.

```
def parse(self, response):
        Autori = []
        hxs = HtmlXPathSelector(response)
        Autor = bibItem()
#tražimo HTML element sa klasom naslov, te izvlačimo tekst prvog elementa
        Autor['ime_prezime']=
hxs.select("//*[@class='naslov']/text()").extract()[0]
#tražimo HTML element sa klasom naslov, te izvlačimo tekst drugog elementa
        Autor['id'] = hxs.select("//*[@class='naslov']/text()").extract()[1]
#tražimo sve href elemente koji su linkovi na radove traženog autora te ih
raščlanjujemo i uzimamo samo id rada koji se nalazi u linku
        Autor['rad'] = hxs.select("//*[contains(concat(' ', @href, ' '), '
prikazi-rad?&rad=')]/@href").extract()
        Autor['ime_prezime']= Autor['ime_prezime'].strip()
        Autor['id']= Autor['id'].strip('( )')
               for i in range(len(Autor['rad'])):
        Autor['rad'][i]=Autor['rad'][i].split('=')[1]
        Autori.append(Autor)

        return Autori
```

Implementirani programski kod za automatizirano pretraživanje web mjesta pokrećemo iz Python Shell-a. Zatim pauka pokrećemo iz Python Shell-a uz pomoć sljedeće naredbe:

```
scrapy crawl bib_spider -o scraped_data.xml -t xml
```

Postupak izvlačenja podataka o radovima vrlo je sličan prethodno opisanom postupku izvlačenja podataka o autorima. Jedina bitna razlika je ta što su detalji rada ispisani u tablici s varijabilnim brojem atributa te je stoga bilo potrebno atribute spremiti u dvije liste, gdje je prva lista naziv atributa, a druga lista vrijednost atributa. Bitno je naglasiti da prilikom izvlačenja podataka s web mjesta treba paziti na zagušenje servera s kojeg se preuzimaju HTML stranice. Scrapy posjeduje implementirane funkcionalnosti automatskog sprječavanja zagušenja servera, a naredbe koje se nalaze u nastavku automatski pauziraju preuzimanje HTML stranica ovisno o zagušenosti servera.

```
#naredba za uključivanje automatskog pauziranja preuzimanja
AUTOTHROTTLE_ENABLED=True
#trajanje stanke od 2 sekunde prilikom pokretanja programa
AUTOTHROTTLE_START_DELAY=2.0
#maksimalno trajanje stanke od 30 sekundi između preuzimanja HTML stranica(ovisno o zagušenosti servera funkcija „AUTOTHROTTLE“ će regulirati vrijeme stanke između 0 do 30 sekundi
AUTOTHROTTLE_MAX_DELAY=30.0
#naredba koja uključuje ispisivanje informacija o zagušenju servera
AUTOTHROTTLE_DEBUG=True
```

5.3 Raščlanjivanje

Nakon prikupljanja, podaci su se nalazili lokalno u dvije XML datoteke. Njihova veličina bila je izrazito velika te je manipulacija s istima bila ograničena. Datoteke zbog veličine nije bilo moguće otvoriti običnim čitačima teksta, kao ni raščlaniti uz pomoć DOM („pull“) raščlanjivača pa je implementiran SAX („push“) raščlanjivač koji slijedno prolazi dokumentom i ispituje određene uvjete poput naziva i tipa elementa. Raščlanjivanje je izvršeno uz pomoć programskog jezika C# u razvojnom okuženju Visual Studio 2012. Za implementaciju raščlanjivača korištena je klasa „XmlTextReader“ koja, za razliku od klase „XmlReader“, dokument čita slijedno. Nakon otvaranja datoteke, za svaki element u XML dokumentu bilo je potrebno provjeriti različite uvjete. Programski kod koji slijedi u nastavku prikazuje dio metode koja služi za raščlanjivanje XML dokumenta „radovi“. Čitanje XML čvorova izvršava se slijedno, te se prilikom svakog zaustavljanja na čvoru provjeravaju zadani uvjeti.

```
while (reader.Read())//petlja se izvršava sve dok postoje čvorovi za čitanje
            {   //uvjet je istinit ukoliko je pročitani element početni čvor
                if (reader.NodeType == XmlNodeType.Element)
                {
                   //uvjet je istinit ukoliko je naziv trenutnog čvora „kategorija“
                   if (reader.Name == "kategorija")
                   {
                       rad.Kategorija = reader.ReadString();
                   }
                   //uvjet je istinit ukoliko je naziv trenutnog čvora „opis_vrijednosti“
                   if (reader.Name == "opis_vrijednosti")
                   {
                       opis_vrijednosti = true;
                   }
…
```

5.4 Spremanje u bazu podataka

Zbog poboljšanja performansi prilikom pretraživanja određenog autora, podaci iz XML dokumenta „autori“ spremljeni su u bazu podatka. Indeksiranje pojedinih atributa u tablicama baze podataka uvelike je poboljšalo performanse pretraživanja. Nakon unošenja podataka o autorima i djelomičnih podataka o radovima u bazu podataka, čitanje velikog XML dokumenta „radovi“ više nije bilo nužno za glavnu funkcionalnost aplikacije. Za rad sa SQLite bazom podataka implementirana je singleton[6] instanca klase. Unošenje u bazu nakon čitanja iz XML datoteke izvršeno je uz pomoć sljedećeg programskog koda:

```
public static int UnesiRad(Radovi rad)
{
   string    sqlUpit    =    "INSERT    INTO    rad('id_rada','naslov',    'kategorija')
VALUES("+rad.IdRada+",'"+rad.Naziv+"', '"+rad.Kategorija+"');";
   int upit = BazaPodataka.Instance.IzvrsiUpit(sqlUpit);
   return upit;
}
public int IzvrsiUpit(string sqlUpit)
{
   SQLiteCommand command = new SQLiteCommand(sqlUpit, Connection);
   return command.ExecuteNonQuery();
}
```

[6] Uzorak dizajna koji omogućava instanciranje samo jednog objekta klase

Analogno prethodnom primjeru programskog koda za unos u bazu podataka, uneseni su podaci o autorima. Prilikom raščlanjivanja XML dokumenta „autori“, ujedno su uneseni i podaci o sudjelovanjima autora na pojedinim radovima uz pomoć sljedećeg programskog koda:

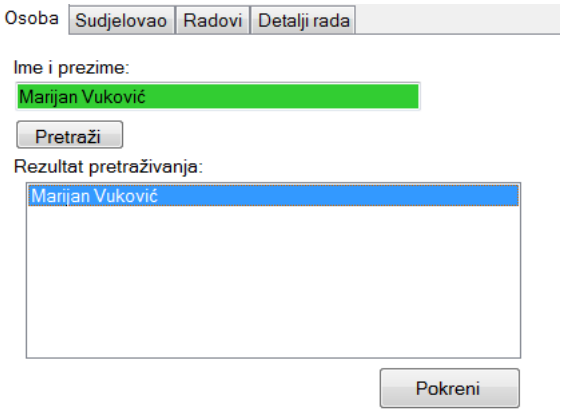
```
if(reader.Name =="autor" && (reader.NodeType == XmlNodeType.EndElement))
                    {
                         Autori autor = new Autori();
                         autor.IdAutora = int.Parse(id);
                         autor.ImePrezime = imePrezime;

                         //Prvo unosimo autora u bazu podataka
                         Autori.UnesiAutora(autor);

                         //Unosimo sve radove na kojima je surađivao autor
                         foreach (int rad in radovi)
                         {
                             Autori.UnesiSudjelovao(autor.IdAutora, rad);
                         }
                         //Console.WriteLine(id.ToString());
                         radovi.Clear();
                    }
```

6. Izrada podatkovnoga modula

U ovome poglavlju opisana je faza izrade podatkovnog modula koji omogućava pretraživanje autora i izradu grafa sudjelovanja. Nakon uspješnog unošenja podataka u bazu i priprema XML datoteke sa detaljima radova, slijedi izrada glavnog obrasca s mogućnošću pretraživanja autora te prikaz koautorstava i detalja radova. Implementaciji kreiranja grafa koautorstava prethodi izrada funkcionalnosti učitavanja relevantnih autora i radova u generičke liste. Naposljetku, opisana je implementacija spremanja projekta u vanjsku XML datoteku i učitavanja projekta iz vanjske datoteke.

6.1 Pretraživanje baze autora i radova

Zbog slabih performansi prilikom prvobitnog testnog dizajna spremišta podataka (dvije XML datoteke bez SQLite baze podataka), podaci o autorima, sudjelovanjima i radovima prebačeni su u SQLite bazu podataka koja uz pomoć indeksiranja višestruko ubrzava pretraživanje. Primjer uspješnog pretraživanja autora prikazan je na slici 6.1. Kada korisnik unese ime i prezime autora u polje za pretraživanje, pritiskom na gumb „Pretraži“ pokreće se metoda za pretraživanje tablice „autor“. Kao što se može vidjeti na slici 6.1, ukoliko traženi autor postoji u bazi podataka, tada polje za unos imena i prezimena autora mijenja boju u zelenu. Kada metoda za pretraživanje baze podataka završi s izvršavanjem, u „listBox“ kontrolu se ispisuju pronađeni autori. Korisnik je tada u mogućnosti označiti traženog autora ako on postoji, a ukoliko se traženi autor ne nalazi u „listBox“ kontroli, tada je potrebno ponoviti pretraživanje različitim unosom.

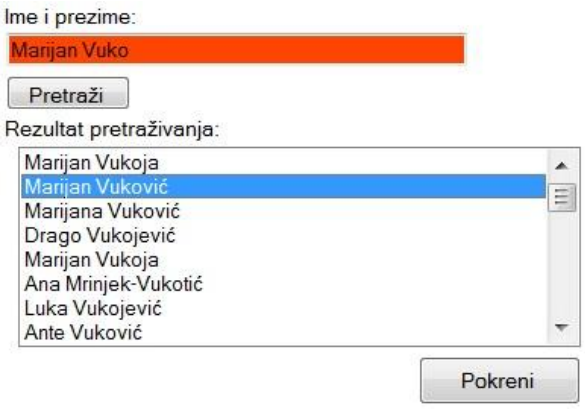


Slika 6.1 Primjer uspješnog pretraživanja autora

Kada u bazi podataka ne postoji traženi autor, poziva se metoda za ponovljeno pretraživanje autora koji imaju slično ime i prezime. Prilikom implementacije metode za ponovljeno pretraživanje, korišten je operator „LIKE“ koji omogućava pronalaženje sličnih rezultata. Dio programskog koda navedene metode nalazi se u nastavku.

```
//Ako neka vrijednost nedostaje
string sqlUpit = "SELECT * FROM autor WHERE ime_prezime LIKE '%" + nizIme[0] + "%' AND
ime_prezime LIKE '%" + nizIme[1] + "%';";
DbDataReader dr = BazaPodataka.Instance.DohvatiDataReader(sqlUpit);
while (dr.Read())
        {
          Autori autor = new Autori(dr);
          listaAutora.Add(autor);//spremamo autora u listu autora
        }
        dr.Close();     //Zatvaranje DataReader objekta.

//Tražimo sve autore s istim prezimenom
string sqlUpit2 = "SELECT * FROM autor WHERE ime_prezime LIKE '%" + nizIme[1] + "%';";
dr = BazaPodataka.Instance.DohvatiDataReader(sqlUpit2);
while (dr.Read())
      {
       Autori autor = new Autori(dr);
       listaAutora.Add(autor); //spremamo autora u listu autora
      }
      dr.Close();     //Zatvaranje DataReader objekta.
...
```

Po završetku izvršavanja prethodno opisane metode, „listBox“ kontrola se popunjava sa svim pronađenim rezultatima, a polje za tekstualni unos imena i prezimena autora poprima crvenu boju, koja je ujedno i poruka korisniku da se traženi autor ne nalazi u bazi podataka. Opisana situacija prikazana je na slici 6.2.



Slika 6.2 Pogrješka kod unosa imena autora

6.2 Kreiranje grafa sudjelovanja autora

Nakon pronalaska i odabira željenog autora iz baze podataka, korisnik pritiskom na gumb „Pokreni“ poziva metodu koja pokreće dvije dretve. Prilikom izvršavanja prve dretve pronalaze se svi koautori traženog autora te se zajedno sa brojem koautorstava svakog pojedinog autora zapisuju u „listView“ kontrolu na kartici „Sudjelovao“. Daljnjim izvršavanjem programskog koda prve dretve, u karticu „Radovi“ upisuju se nazivi svih radova na kojima je traženi autor sudjelovao. Kada je prethodna operacija završena, poziva se sljedeća dretva koja paralelno s prvom dretvom raščlanjuje XML dokument s radovima. Paralelnim radom navedenih dretvi izvršava se prebacivanje detalja radova u generičku listu te se generira graf sudjelovanja i pozicionira vrhove grafa u 3D prostoru[7]. U nastavku se nalaze relevantni dijelovi koda koji prikazuju implementaciju navedenih dretvi:

```
//Pokretanje prve dretve koja pronalazi sva sudjelovanja autora
dretvaDetaljiAutor = new Thread(new ThreadStart(DretvaDetaljiAutorWork));
dretvaDetaljiAutor.Start();
…
public void DretvaDetaljiAutorWork() //Dretva koja učitava sva sudjelovanja i radove
{
   toolStripStatusLabel1.Text = "Učitavam podatke o autoru";
   noviAutor = Autori.DohvatiSudjelovanja(idAutora); //dohvaćamo sva sudjelovanja

   foreach (Autori autor in noviAutor.SudjelovaoAutori)
   {
```

[7] Detaljnije opisano u poglavlju 6.3.

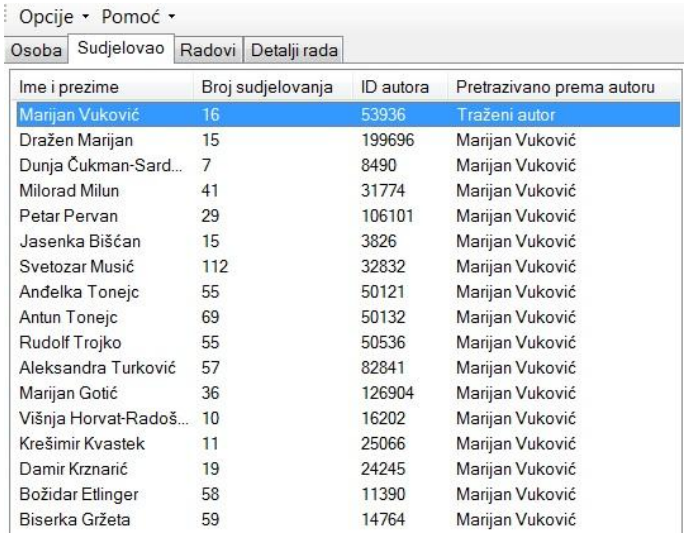
```
        //u ListBox dodajemo sve autore koji su sudjelovali sa traženim autorom
        lbSudjelovao.Items.Add(autor);
    }

    foreach (Radovi rad in noviAutor.SudjelovaoRadovi)
    {
        //u ListBox dodajemo sve radove na kojima je autor radio
        lbRadovi.Items.Add(rad);
    }
     …
    //pokrećemo dretvu koja parsira XML dokument s detaljima rada i učitava ga u listu
    dretvaDetaljiRad = new Thread(new ThreadStart(DretvaDetaljiRadWork));
    dretvaDetaljiRad.Start();

    //Pozivamo metodu koja generira čvorove, veličinu i veze između čvorova
    this.GenerirajCvorove();
```

Uz pomoć metode „GenerirajRazinuCvorova“ kreiraju se čvorove u grafu za svakog autora, ukoliko on već nije ranije dodan. Sve koordinate postavljaju se na v3f(0,0,0) te se dodjeljuje veličina ovisno o broju koautorstava. Veličina čvora direktno ovisi o broju koautorstava, što ujedno znači da su autori s većim brojem koautora prikazani većim čvorovima. Nakon što spremimo čvor u listu, provjeravamo postoji li veza između autora i njegovog koautora. Ukoliko navedena veza ne postoji, tada ju kreiramo. Naziv veze od čvora *A* do čvora *B*, koji je ujedno i jedinstveni identifikator, sastavlja se od jedinstvenog identifikatora čvora *A*, znaka '-' i jedinstvenog identifikatora čvora *B*. Na primjer, ako čvor *A* ima jedinstveni identifikator „23491“, a čvor *B* ima jedinstveni identifikator „12364“, tada naziv veze između čvora *A* i *B* glasi „23491-12364“. Navedeni algoritam pojednostavljuje naknadnu manipulaciju vezama, a spomenuti algoritam opisan je u nastavku.

```
…
//ako čvor postoji onda provjeravamo postoji li veza
string idVeze = autorSudjelovao.IdAutora.ToString() + "-" + autor.IdAutora.ToString();
postoji=false;
for (int i = 0; i < linkList.Count && postoji==false; i++)
{
    if (linkList[i].IdVeze.Equals(idVeze)) postoji = true;
}
//ako ne postoji veza između ta dva elementa
if (!postoji)
{
    idVeze = autor.IdAutora.ToString() + "-" + autorSudjelovao.IdAutora.ToString();
```

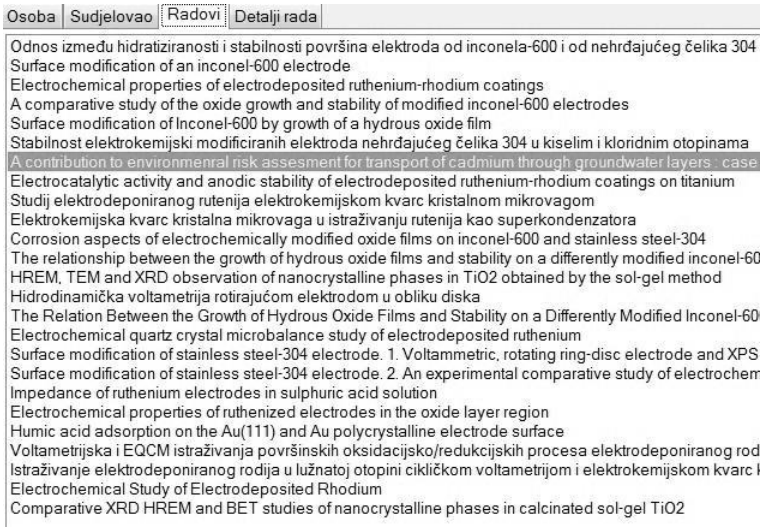
```
    Link veza = new Link();
    veza.IdVeze = idVeze;
    linkList.Add(veza);
 }
…
```

Nakon učitavanja sudjelovanja i radova u „listBox“ kontrole, korisnik može pregledavati sve koautore traženog autora. Na glavnoj formi nalaze se kartice „Osoba“, „Sudjelovao“, „Radovi“ i „Detalji“ rada. Na kartici „Osoba“ nalazi se obrazac za pretraživanje autora i postavke grafa. Kartica „Sudjelovao“ prikazana je na slici 6.3. Prilikom označavanja autora na kartici „Sudjelovao“, čvor označenog autora mijenja boju prikaza.

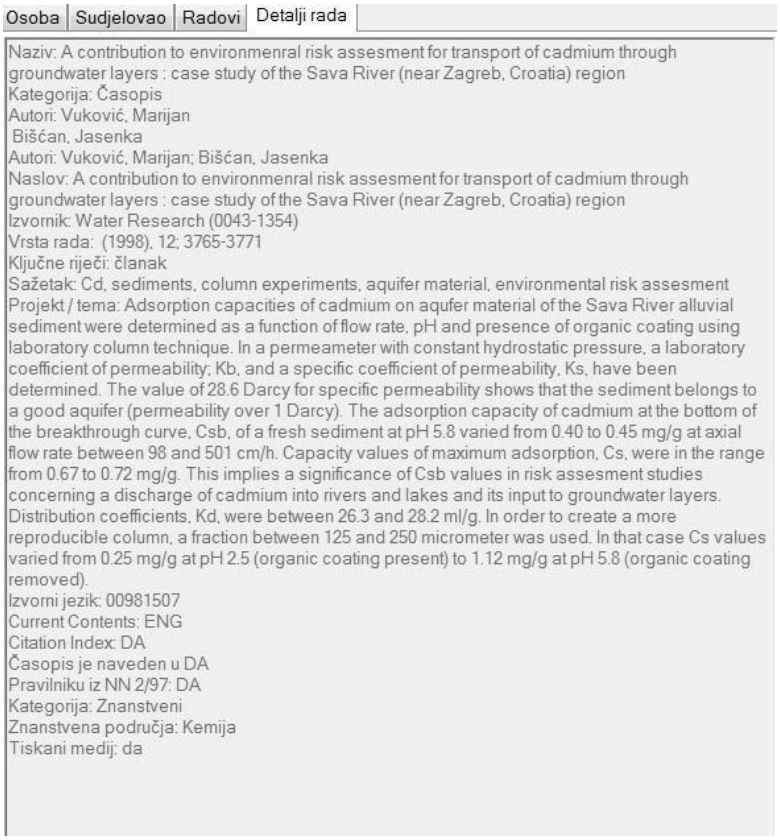


Slika 6.3 Kartica "Sudjelovao"

Kartica „Radovi“, koja je prikazana na slici 6.4, sadrži radove u čijoj je izradi autor sudjelovao. Ukoliko je dretva za raščlanjivanje XML dokumenta s detaljima radova izvršila svoj posao, tada je moguće odabirom određenog rada prikazati detalje. Navedena situacija prikazana je na slici 6.5.



Slika 6.4 Kartica "Radovi"

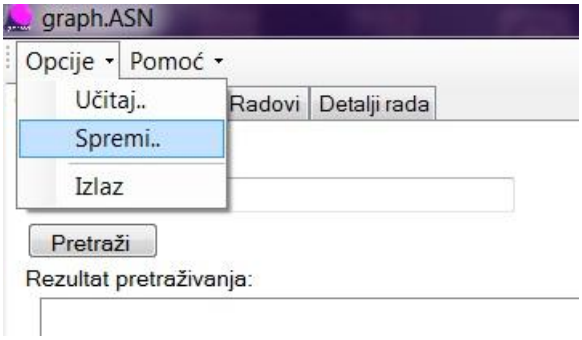


Slika 6.5 Kartica „Detalji rada“

6.3 Spremanje i učitavanje projekta

Ukoliko korisnik želi spremiti trenutni projekt, tada može koristiti opciju „Spremi projekt“ koja se nalazi na padajućem izborniku „Opcije“. U aplikaciji su implementirane dvije metode koje

služe za spremanje i učitavanje projekta. Prva metoda služi za spremanje trenutnih postavki aplikacije, listi čvorova i veze te detalja svih radova autora. Na slici 6.6 su prikazane opcije „Spremi“ i „Učitaj“.



Slika 6.6 Opcije „Spremi“ i „Učitaj“

Za spremanje projekta u datoteku, korištena je klasa „XmlTextWriter“ koja služi za pohranjivanje podataka u XML datoteku, kao i klasa „SaveFileDialog“ koja služi za odabir putanje i naziva dokumenta. U nastavku je opisano korištenje klase „SaveFileDialog“ uz pomoć programskog koda:

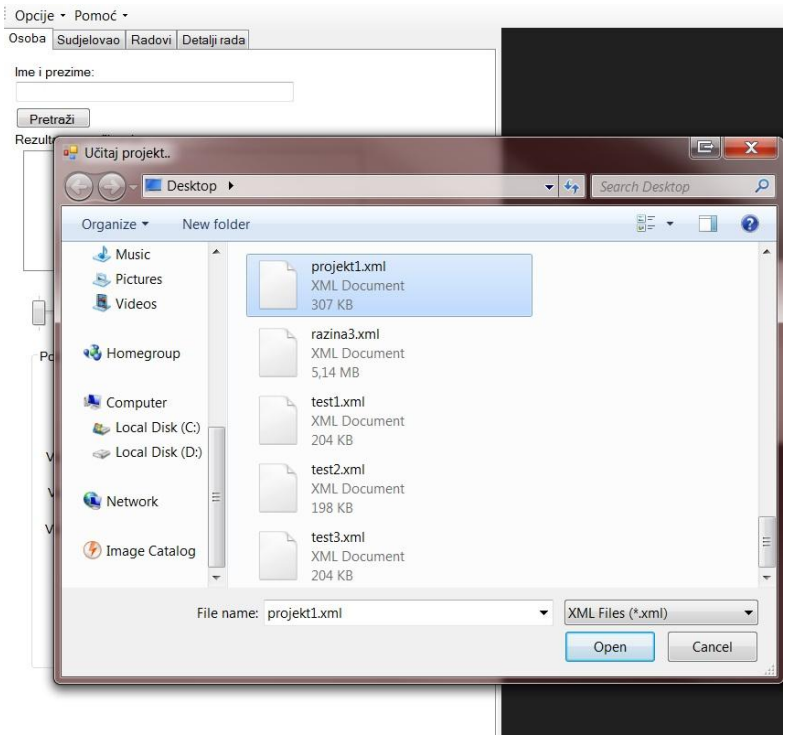
```
public static XmlTextWriter GetWriterForFile()
        {   //instanciramo objekt klase SaveFileDialog
            SaveFileDialog dlg = new SaveFileDialog();
            dlg.Filter = "XML Files (*.xml)|*.xml";//Ograničavamo tip datoteke za pohranu
            dlg.Title = "Spremi projekt..";//određujemo naslov prozora
            dlg.CheckPathExists = true;//uključujemo provjeru postajanja putanje
            dlg.DefaultExt = "xml";//uključujemo podrazumijevanu ekstenziju
            //ukoliko nije odabrana putanja metoda završava s radom
            if (dlg.ShowDialog() != DialogResult.OK)  return null;

            Encoding encoding = Encoding.UTF8;
            //instanciramo objekt klase XmlTextWriter
            XmlTextWriter writer = new XmlTextWriter(dlg.FileName, encoding);
            writer.Formatting = Formatting.Indented;//uključujemo formatiranje
            return writer;//vraćamo instancirani objekt
        }
```

Nakon korisničkog odabira, podaci se spremaju u XML datoteku koja ima sljedeću strukturu:

```
<Graph.ASN_Project>
       <Info>Osnovne informacije o aplikaciji</Info>
```

```
        <Postavke>Sve postavke grafa</Postavke>
        <Objetki>
                <trenutniAutor>..</trenutniAutor>
                <radoviID>..</radoviID>
                <nodeList>..</nodeList>
                <linkList>..</linkList>
                <radoviXML>..</radoviXML>
        </Objekti>
</Graph.ASN_Project>
```

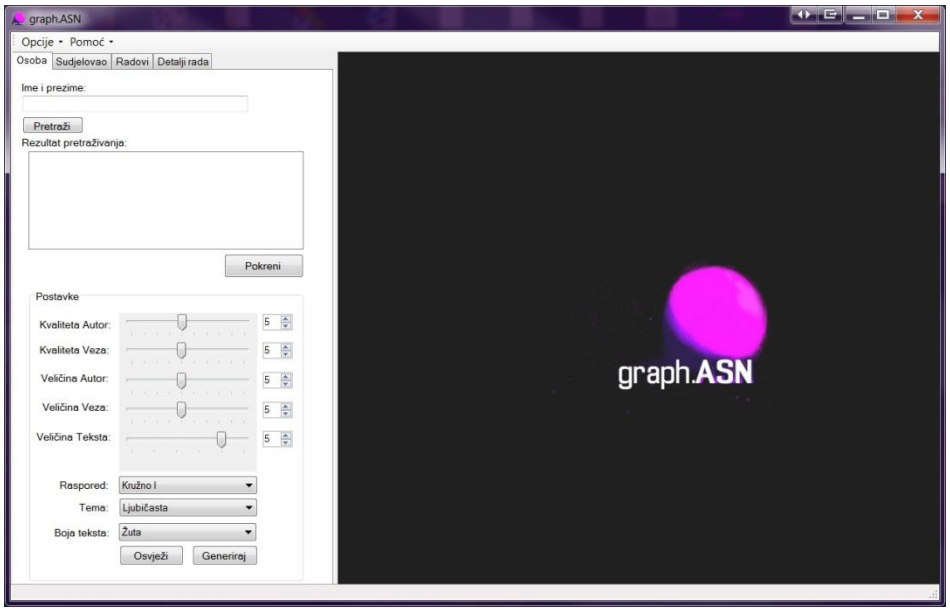
Spremljeni projekt moguće je učitati uz pomoć opcije „Učitaj“ koja otvara prozor za odabir projektne datoteke. Prethodno navedeni prozor prikazan je na slici 6.7.



Slika 6.7 Učitavanje projekta

7. Izrada grafičkog modula

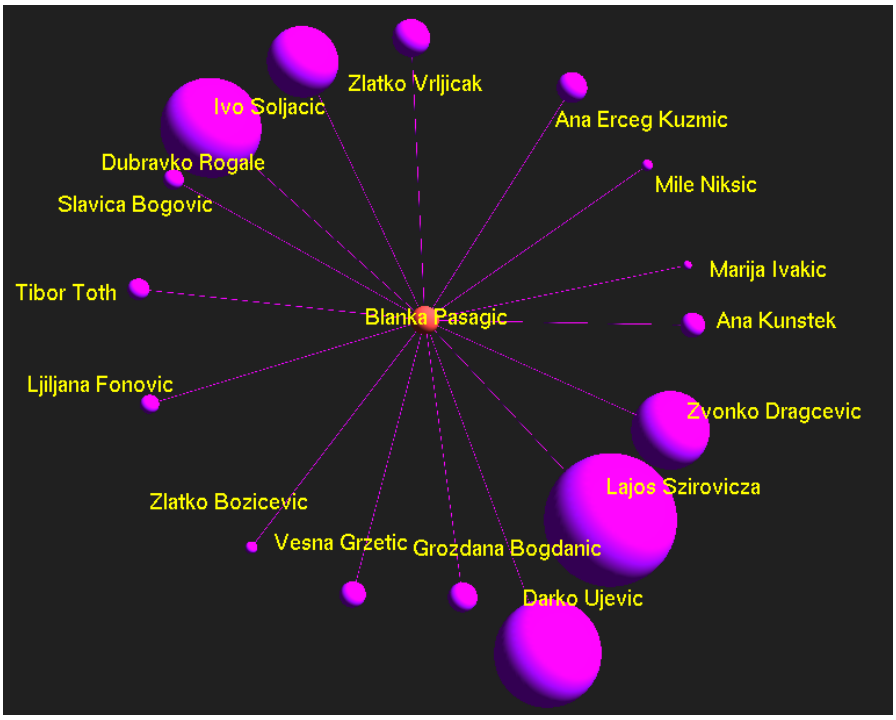
U ovom poglavlju opisana je posljednja faza implementacije aplikacije. Završetak druge faze omogućava implementaciju vizualnog prikaza grafa akademskih društvenih mreža uz pomoć kreiranog grafa sudjelovanja. Navedeni graf ilustriran je u 3D prostoru, a sastoji se od čvorova koji predstavljaju autore i veze koje predstavljaju koautorstva. U nastavku poglavlja opisana je implementacija grafa, a prije toga potrebno je opisati izradu osnovnog grafičkog konteksta. Na slici 7.1 prikazan je osnovni grafički kontekst koji je ugrađen u prozor Windows Forms aplikacije.



Slika 7.1 Grafički kontekst i prikaz grafa

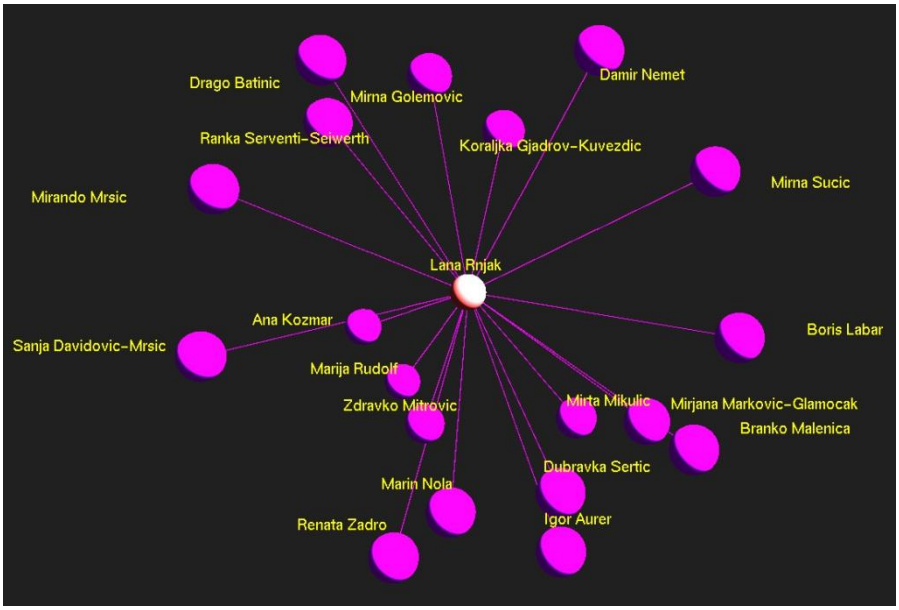
7.1 Implementacija pozicioniranja grafa

Za prikazivanje čitljivog grafa, čvorove je potrebno pravilno pozicionirati u prostoru. Algoritme pozicioniranja potrebno je prilagoditi različitoj veličini grafa. Kako bi graf bio pregledan, potrebno je smanjiti broj križanja veza i čvorove pozicionirati dovoljno udaljene jedne od drugih. Prilikom razvoja aplikacije, implementirano je nekoliko različitih algoritama za pozicioniranje. Prvi algoritam pozicioniranja pod nazivom „Kružno I“ stavlja naglasak na traženog autora, kojeg postavlja na središte grafa. Oko traženog autora pozicioniraju se njegovi koautori u obliku kružnice. Opisani algoritam prilagođen je za male i velike grafove, a primjer je prikazan na slici 7.2.



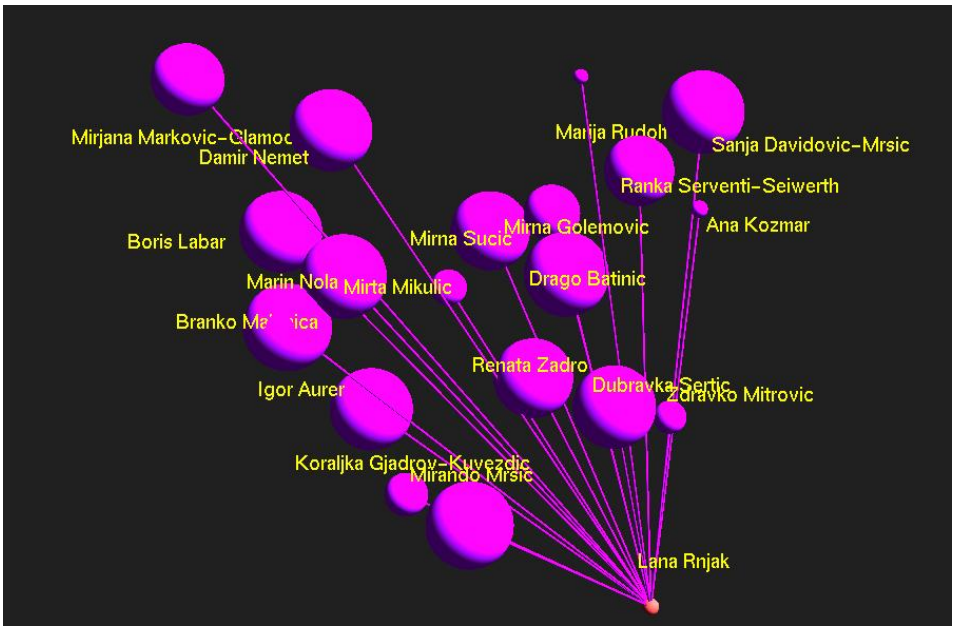
Slika 7.2 Pozicioniranje "Kružno I"

Sljedeći algoritam pod nazivom „Kružno II“ sličan je prethodno opisanom algoritmu. Traženi autor nalazi se u središtu grafa, dok su njegovi koautori udaljeni od središta ovisno o broju koautorstava. Ukoliko autor ima veći broj koautorstava, tada je on udaljeniji od središta. Navedeni algoritam bolji je za pozicioniranje manjih grafova zbog slabijih performansi i preglednosti kod većih grafova. Prethodno navedeni algoritam pozicioniranja prikazan je na slici 7.3.



Slika 7.3 Pozicioniranje "Kružno II"

Sljedeći implementirani algoritmi pozicioniranja su „Pseudoslučajno I“ i „Pseudoslučajno II“. Navedeni algoritmi pozicioniraju autore u 3D prostoru pseudoslučajnim izborom koordinata čvora. Pozicioniranje „Pseudoslučajno I“ prikazano je na slici 7.4, a služi za pozicioniranje manjih grafova, dok pozicioniranje „Pseudoslučajno II“, zbog varijabilne udaljenosti između čvorova koja ovisi o ukupnom broju čvorova, služi za pozicioniranje većih grafova. Posljednji algoritam pozicioniranja je „Matrično“, za kojeg je karakteristično da se čvorovi raspoređuju u trodimenzionalnu matricu.



Slika 7.4 Pozicioniranje "Pseudoslučajno I"

7.2 Kreiranje grafičkog konteksta

Za uspješnu nadogradnju glavne forme s OpenGL kontrolom potrebno je referencirati .dll datoteke[8] Tao razvojnog okruženja. Izrada grafičkog konteksta počinje ugradnjom „SimpleOpenGLControl“ kontrole. Prilikom učitavanja glavne forme, potrebno je pozvati naredbe za inicijalizaciju konteksta koje su opisane u programskom kodu koji slijedi u nastavku. Naredba „InitializeContexts“ inicijalizira grafički kontekst, a naredba „SwapBuffers“ zamjenjuje sadržaj međuspremnika.

```
//događaj koji se pokreće prilikom učitavanja glavne forme
private void GlavnaForma_Load(object sender, EventArgs e)
{
   //inicijaliziramo kontekst
   openglSpremnik.InitializeContexts();
   openglSpremnik.SwapBuffers();

   //postavljamo boju pozadine
   Gl.glClearColor(0.129f, 0.129f, 0.129f, 0.9f);

   //određujemo širinu i visinu spremnika
   height = openglSpremnik.Height;
   width = openglSpremnik.Width;

   //dodajemo rukovatelj događaja za „openglSpremnik“ kontrolu
   openglSpremnik.Load += new EventHandler(simpleOpenGlControl1_Load_1);
   openglSpremnik.MouseMove += new System.Windows.Forms.MouseEventHandler(glOnMouseMove);
   openglSpremnik.SizeChanged += new EventHandler(simpleOpenGlControl1_SizeChanged);

   //pozivamo metodu za promjenu veličine kontrole zbog resetiranja kontrole
   simpleOpenGlControl1_SizeChanged(sender, e);

   LastTransformation.SetIdentity(); // Resetiramo rotaciju
   ThisTransformation.SetIdentity(); // Resetiramo rotaciju
   ThisTransformation.get_Renamed(matrix);
…
```

Prilikom promjene veličine kontrole poziva se metoda za rukovanje događajem koja postavlja grafičku scenu i osvjetljenje. Navedena metoda ujedno služi i za resetiranje grafičke scene, a programski kod navedene metode opisan je u nastavku.

[8] Tao.OpenGl.dll, Tao.Platform.Windows.dll, tao.FreeGlut.dll

```
//Postavljamo veličinu kontrole
Size s = Size;
height = s.Height;
width = s.Width;

Gl.glShadeModel(Gl.GL_SMOOTH);//aktiviramo uglađene sjene

Gl.glClearColor(0.129f, 0.129f, 0.129f, 0.9f);//postavljamo boju pozadine
Gl.glClearDepth(1.0f);// postavljamo buffer za dubinu objekata
Gl.glEnable(Gl.GL_DEPTH_TEST); //aktiviramo provjeru dubine objekata
//aktiviramo funkciju za provjeru međuspremnika za uspoređivanje dubine objekata
Gl.glDepthFunc(Gl.GL_LEQUAL);
//uključujemo korekciju perspektive
Gl.glHint(Gl.GL_PERSPECTIVE_CORRECTION_HINT, Gl.GL_NICEST);

Gl.glViewport(0, 0, width, height);//definiramo prikazno polje

Gl.glPushMatrix();//stavljamo matricu na stog
Gl.glMatrixMode(Gl.GL_PROJECTION);//uključujemo matricu projekcije
Gl.glLoadIdentity();//učitava matricu identiteta
//postavlja matricu projekcije za perspektivni prikaz
Glu.gluPerspective(25.0, (double)width / (double)height, 1.0, 15.0);
Gl.glTranslatef(0.0f, 0.0f, -4.0f);//izvršavamo tranlaciju matrice
Gl.glMatrixMode(Gl.GL_MODELVIEW);//uključujemo modelview matricu
Gl.glPopMatrix();//sklanjamo matricu sa stoga

arcBall.setBounds((float)width, (float)height);//ažuriramo okvir interakcije sa mišem
PlotGL();//Metoda za ocrtavanje grafa
…
```

7.3 Prikaz sudjelovanja autora

Pod pojmom sudjelovanja smatra se direktno koautorstvo između pojedinih autora. Za prikaz sudjelovanja potrebno je prvo implementirati uvjete za prikaz grafičke scene. Postavljanje osnovnog grafičkog konteksta i osvjetljenja omogućava daljnju izradu prikaza sudjelovanja autora. Graf se sastoji od autora koji su prikazani u obliku sfere, a veze između njih su cilindri usmjereni od čvora *A* do čvora *B*. Zbog poboljšanja performansi, za kreiranje svih čvorova u obliku sfera korištena je lista prikaza koja se generira samo po potrebi, a ocrtavanje sfera izvršava se uz pomoć poziva navedene liste. Metoda koja slijedi u nastavku generira listu čvorova koji se naknadno prikazuju.

```
private void Sfera()
{
     // generiramo listu prikaza
     plot_glListAutori = Gl.glGenLists(1);
     //počinjemo dodavati objekte u listu prikaza
     Gl.glNewList(plot_glListAutori, Gl.GL_COMPILE);
     foreach(Node node in nodeList)
     {
         //Određujemo boju sfere
         if(tema==1)  Gl.glColor3f(node.R, node.G, node.B);
         else if (tema == 2) Gl.glColor3f(0.28f, 0.81f, 0.8f);
         else if (tema == 3) Gl.glColor3f(0.28f, 0.81f, 0.8f);
         else if (tema == 4) Gl.glColor3f(0.1f, 0.4f, 0.1f);
         else if (tema == 5) Gl.glColor3f((float)random.NextDouble()/2.4f,
                                          (float)random.NextDouble(), 0);
         else if (tema == 6) Gl.glColor3f((float)random.NextDouble() / 3,
                                          (float)random.NextDouble() / 2.8f,
                                          (float)random.NextDouble());
       if (n++ == 0)
       {
             Gl.glColor3f(0.5546f, 0.1367f, 0.1367f);
             if (tema == 2 || tema == 3) Gl.glColor3f(1f, 0.6445f, 0.3085f);
       }
       if (node.IdAutora == OdabraniCvor)
       {
             Gl.glColor3f(0.43f, 0.855f, 0.574f);
             if (tema == 2 || tema == 3) Gl.glColor3f(0.937f, 0.898f, 0.5468f);
       }
       //stavljamo matricu na stog i izvršavamo translaciju
         Gl.glPushMatrix();
         Gl.glTranslatef(node.X,node.Y, node.Z);

         //inicijaliziramo novu kvadriku
         quadratic = Glu.gluNewQuadric();
         //ocrtavamo sferu
         Glu.gluSphere(quadratic,radiusSfera+node.Velicina, slicesSfera,stacksSfera);
         //uklanjamo matricu sa stoga
         Gl.glPopMatrix();

           }
           //zatvaramo listu prikaza
           Gl.glEndList();
       }
```

Nakon izrade liste prikaza sfera slijedi izrada liste prikaza veza između njih u obliku cilindara. Crtanje cilindra zahtjevnije je nego sfere, zbog toga što je potrebno usmjeriti cilindar iz središta sfere *A* u središte sfere *B*. Za rješavanje problema prikaza cilindra između dvije točke, čiji programski kod slijedi u nastavku, korišten je dio programskog koda kojeg je napisao Kelleher (2008).

```
// Metoda ocrtava cilindar između točke A i B
void OcrtajCilindar (float x1, float y1, float z1, float x2, float y2, float z2)
{
  //računamo koordinate pomoćnog vektora
  float vx = x2 - x1;
  float vy = y2 - y1;
  float vz = z2 - z1;
  double v = Math.Sqrt(vx * vx + vy * vy + vz * vz);//udaljenost dvije točke
  double ax; //kut rotacije

  if (Math.Abs(vz) < 1.0e-3)//ako je z koordinata 0
  {
      ax = 180/Math.PI * Math.Acos(vx / v); // kut rotacije u x-y ravnini
      if (vy <= 0.0)  ax = -ax;
  }
  else
  {
      ax = 180/Math.PI * Math.Acos(vz / v); // kut rotacije
      if (vz <= 0.0) ax = -ax;
  }

  float rx = -vy * vz;
  float ry = vx * vz;

  Gl.glPushMatrix();//stavljamo matricu na stog
  Gl.glTranslatef(x1, y1, z1); //postavljamo početne koordinate na točku A
  if (Math.Abs(vz) < 1.0e-3)//ako je z koordinata 0
  {
      Gl.glRotated(90.0, 0, 1, 0.0); //rotiramo i poravnavamo sa x osi
      Gl.glRotated(ax, -1.0, 0.0, 0.0); // rotiramo prema točki B u x-y ravnini
  }
  else
  {
      Gl.glRotated(ax, rx, ry, 0.0); // rotiramo oko pomoćnog vektora
  }
  Glu.GLUquadric q = Glu.gluNewQuadric();//inicijaliziramo novu kvadriku
```

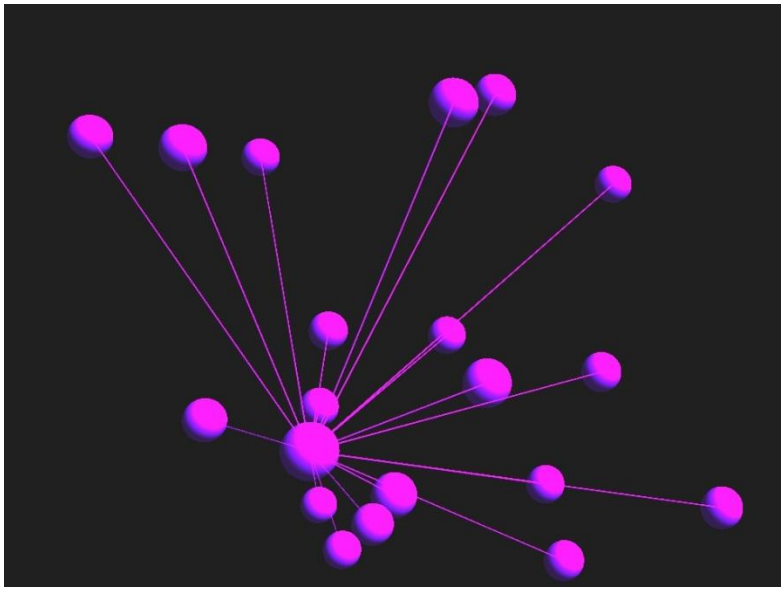
```
  Glu.gluQuadricOrientation(q, Glu.GLU_OUTSIDE);//orijentacija kvadrike

  //crtamo cilindar
  Glu.gluCylinder(q, radiusCilindar, radiusCilindar, v, slicesCilindar, 1);
  //uklanjamo matricu sa stoga
  Gl.glPopMatrix();
}
```

Metoda za kreiranje liste prikaza svih veza između autora, čiji programski kod slijedi u nastavku, koristi prethodno opisanu metodu. Prolaskom kroz listu veza poziva se metoda „OcrtajCilindar“ te se prosljeđuju koordinate čvorova *A* i *B*. Prilikom završetka ocrtavanja svih cilindara, poziva se naredba „Gl.glEndList();“ koja služi za zatvaranje liste prikaza.

```
//Metoda generira listu prikaza veza između autora
public void Cilindar()
{
     // generiramo listu prikaza
     plot_glListVeze = Gl.glGenLists(1);
     //počinjemo dodavati objekte u listu prikaza
     Gl.glNewList(plot_glListVeze, Gl.GL_COMPILE);

     //određujemo boju cilindara
            if (tema == 1)      Gl.glColor3f(0.5f, 0.0f, 0.8f);
            else if (tema == 2) Gl.glColor3f(0.93f, 0.25f, 0.00f);
            else if (tema == 3) Gl.glColor3f(0.28f, 0.46f, 1f);
            else if (tema == 4) Gl.glColor3f(0.55f, 0.24f, 0.18f);
            else if (tema == 5) Gl.glColor3f((float)random.NextDouble(),0 ,
                                             (float)random.NextDouble());
            else if (tema == 6) Gl.glColor3f((float)random.NextDouble() / 3,
                                             (float)random.NextDouble() / 2.8f,
                                             (float)random.NextDouble());

            //za svaku vezu iz liste veza  crtamo pripadajući cilindar
            foreach (Link link in linkList)
            {
                OcrtajCilindar(link.aX,link.aY,link.aZ, link.bX, link.bY, link.bZ);
            }
            //zatvaramo listu prikaza
            Gl.glEndList();
}
```

Prethodno opisane liste prikaza koriste se za ocrtavanje kompletnog grafa sudjelovanja. Naredbom „Gl.glClearColor“ određujemo boju pozadine, a prije samog ocrtavanja liste prikaza izvršavamo dinamičku transformaciju uz pomoć naredbi „Gl.glPushMatrix“, „Gl.glMultMatrixf“ i „Gl.glTranslatef“. Ocrtan graf s čvorovima i vezama između njih prikazan je na slici 7.5. Ocrtavanje liste prikaza omogućuje metoda „Gl.glCallList“, a potpuni programski kod metode za ocrtavanje slijedi u nastavku.

```
//Metoda koja crta graf
public void PlotGL()
{
   …
   Gl.glClear(Gl.GL_COLOR_BUFFER_BIT | Gl.GL_DEPTH_BUFFER_BIT);//čistimo međuspremnike
   Gl.glClearColor(0.129f, 0.129f, 0.129f, 0.9f);//određujemo boju pozadine
   Gl.glLoadIdentity();//učitavamo matricu identiteta

   Gl.glPushMatrix();//stavljamo matricu na stog(priprema dinamičku transformaciju)
   Gl.glMultMatrixf(matrix);//izvršavamo dinamičku transformaciju
   Gl.glTranslatef(X , Y, Z );//translatiramo koordinate prikaza

   if (renderiraj)//ako je vrijeme za renderiranje grafa
   {
      Gl.glPolygonMode(Gl.GL_FRONT, Gl.GL_FILL);//način prikaza poligona
      Gl.glCallList(plot_glListAutori); //crtamo  listu prikaza autora
      Gl.glCallList(plot_glListVeze); //crtamo  listu prikaza sudjelovanja
      Gl.glCallList(plot_glListTekst); //crtamo listu prikaza teksta
   }
   Gl.glPopMatrix(); //uklanjamo matricu sa stoga
   Gl.glFlush();     // brišemo opengl „pipeline“

   this.openglSpremnik.Invalidate();//šaljemo signal za osvježavanje opengl kontrole
 }
 …
}
```



Slika 7.5 Prikaz autora i sudjelovanja

Kao što možemo vidjeti, grafu nedostaju ime i prezime autora te korisničke kontrole manipulacije grafom u obliku pomicanja grafa u 3D prostoru uz pomoć miša. Renderiranje teksta čini se kao trivijalna funkcionalnost, ali u OpenGL-u to je zahtjevnije nego što izgleda. Dodatni problem predstavlja korištenje OpenGL omotača za programski jezik C#, a ujedno s tim dolazi i kompletno oslanjanje na implementirane funkcionalnosti omotača. Postoje tri mogućnosti implementacije prikaza teksta u OpenGL jeziku. Navedene mogućnosti opisane su u sljedećim odlomcima.

Prva opcija je renderiranje „bitmap“[9] fonta, prilikom čega se koristi naredba Gdi.CreateFont za generiranje fonta iz instaliranih fontova na korisničkom operacijskom sustavu i ocrtavanje uz pomoć naredbe „Wgl.wglUseFontBitmapsA“. U Tao okviru naredba „Wgl.wglUseFontBitmaps“ izaziva rušenje programa bez ikakvih obavijesti o pogrješki te navedenu pogrješku nikako nije moguće spriječiti.

Druga opcija je ocrtavanje „outline“[10] fonta, koja je slična prethodnoj opciji, ali se vektorski fontovi ocrtavaju u 3D prostoru i mogu imati različitu veličinu i debljinu. Za generiranje vektorskog fonta se također koristi naredba Gdi.CreateFont koja služi za generiranje fonta, a za ocrtavanje fonta koristi se naredba „WglUseFontOutlinesA“, koja za razliku od naredbe „Wgl.wglUseFontBitmapsA“, u Tao razvojnom okruženju ne izaziva rušenje. Vektorski font

[9] Bitmap ili rasterski font koji se sastoji od serije točaka koje prikazuju sliku svakoga znaka

[10] Outline, obrisni ili vektorski font koristi Bezierove krivulje za ocrtavanje svakog znaka

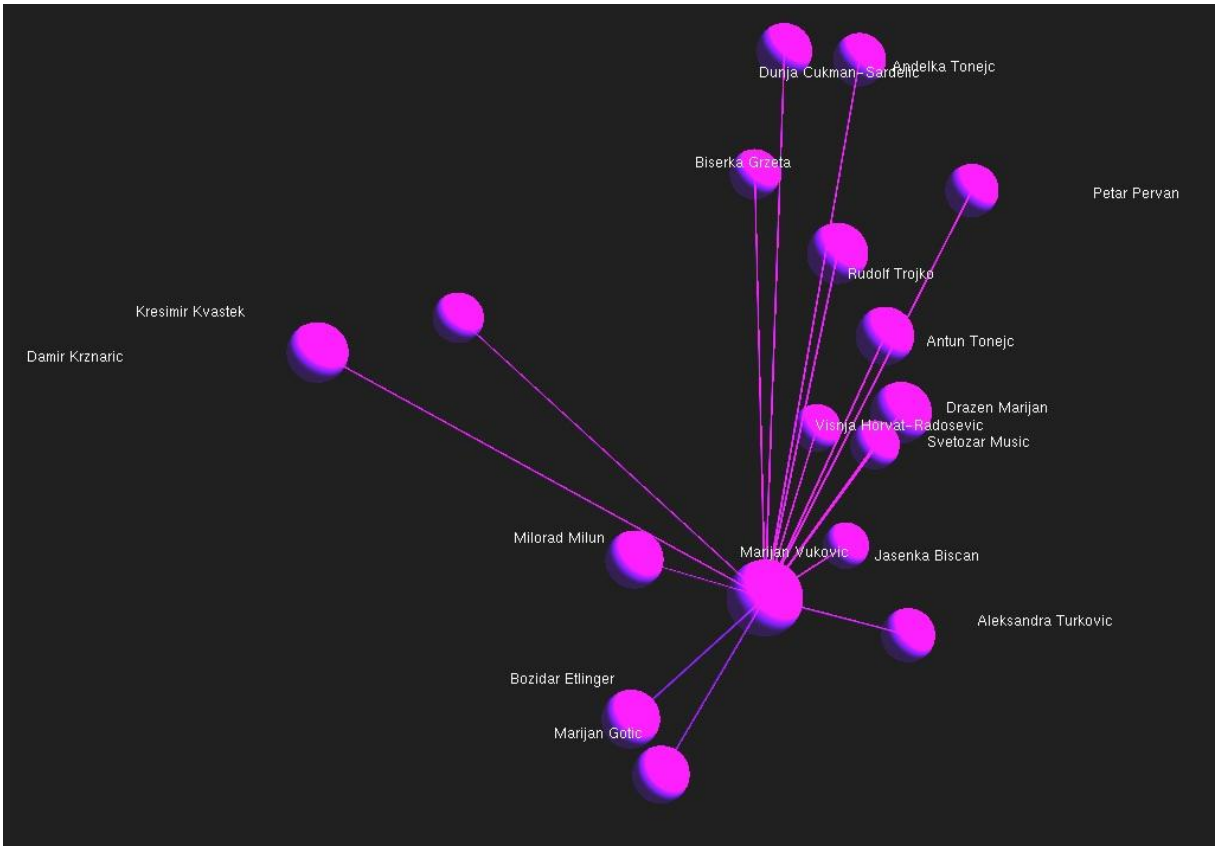
nalazi se u 3D prostoru i prilikom svakog pomicanja grafa u prostoru potrebno je osvježiti poziciju i rotaciju teksta. Koristeći naredbu „wglUseFontOutlinesA“, prikaz hrvatskih dijakritičkih znakova omogućen je parametrom Gdi.UNICODE_CHARSET opcije Gdi.CreateFont. Klasa „Tao.Platforms.Windows.Gdi“ u Tao razvojnom okruženju ima implementiranu samo opciju Gdi.ANSI_CHARSET, gdje druge opcije nije moguće koristiti. Prilikom pokušaja ispisa teksta koji sadrži dijakritičke znakove prekida se izvršavanje OpenGL programskog koda.

Treći način ocrtavanja fonta je naredba „Glut.glutBitmapString“, koja se nalazi u „FreeGlut“ klasi. Naredba ne omogućava nikakve naprednije postavke fonta te omogućava korištenje malog broja ugrađenih fontova i veličina. Kao ni kod prethodno opisane opcije, naredba „Glut.glutBitmapString“ ne podržava hrvatske dijakritičke znakove. Rješenje navedenog problema implementirano je uz pomoć zamjene znakova (č, ć, đ, š, ž) znakovima (c, c, d, s, z).
Uz pomoć „switch“ selekcije izvršava se naredba za ocrtavanje različitog fonta i veličine, a kompletni programski kod slijedi u nastavku.

```
//metoda za ispis teksta  u OpenGL kontroli
static void text(string tekst)
{
string[] pocetniZnak = { "Š", "š", "Đ", "đ", "Č", "č", "Ć", "ć", "Ž", "ž"};
string[] noviZnak = { "S", "s", "D", "d", "C", "c", "C", "c", "Z", "z"};
string noviString = tekst;
//prolazimo kroz ime i prezime autora te mijenjamo hrvatske dijakritičke znakove
for (int i = 0; i < 10; i++)
{
     noviString = noviString.Replace(pocetniZnak[i], noviZnak[i]);
}
//ovisno o korisničkom izboru ocrtavamo tekst
switch (velicinaTeksta)
{
    case 1: break;
    case 2: Glut.glutBitmapString(Glut.GLUT_BITMAP_HELVETICA_10, noviString); break;
    case 3: Glut.glutBitmapString(Glut.GLUT_BITMAP_HELVETICA_12, noviString); break;
    case 4: Glut.glutBitmapString(Glut.GLUT_BITMAP_HELVETICA_18, noviString); break;
    case 5: Glut.glutBitmapString(Glut.GLUT_BITMAP_8_BY_13, noviString); break;
    case 6: Glut.glutBitmapString(Glut.GLUT_BITMAP_TIMES_ROMAN_24, noviString); break;
}
}
```

Za svaki čvor koji prikazuje autora poziva se metoda za ocrtavanje teksta te se tako kreira lista prikaza imena i prezimena. Konačan izgled grafa prikazan je na slici 7.6. Programski kod koji slijedi u nastavku prikazuje implementaciju ispisa teksta.

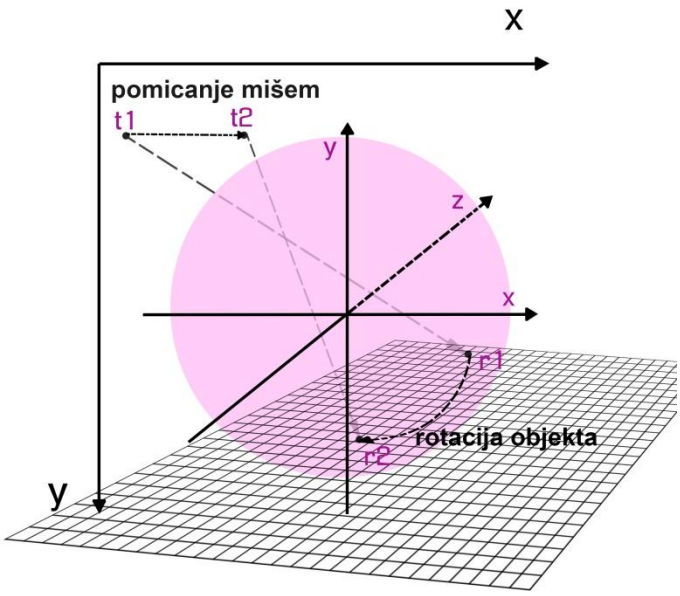
```
// Metoda za generiranje liste prikaza imena autora
public void IspisiTekst()
{
   // generiramo listu prikaza
   plot_glListTekst = Gl.glGenLists(1);
   //počinjemo dodavati objekte u listu prikaza
   Gl.glNewList(plot_glListTekst, Gl.GL_COMPILE);

   //matricu prebacujemo u  način PROJECTION
   Gl.glMatrixMode(Gl.GL_PROJECTION);
   //stavljamo matricu na stog
   Gl.glPushMatrix();
   Gl.glDisable(Gl.GL_TEXTURE_2D);//isključujemo teksture
   Gl.glDisable(Gl.GL_LIGHTING);//isključujemo osvjetljenje
   Gl.glLoadIdentity();//učitavamo matricu identiteta
   Gl.glColor3f(1f, 1f, 1f);//mijenjamo boju teksta u bijelu

   //za svaki čvor pozicioniramo tekst u prostoru i ispisujemo ime i prezime autora
   foreach (Node node in nodeList)
   {
      Gl.glRasterPos3d(node.X, node.Y, node.Z);
      text(node.ImePrezime);
   }
  Gl.glEnable(Gl.GL_TEXTURE_2D); //uključujemo ponovno teksture
  Gl.glEnable(Gl.GL_LIGHTING);//uključujemo osvjetljenje
  Gl.glPopMatrix();//uklanjamo matricu sa stoga
  Gl.glMatrixMode(Gl.GL_MODELVIEW);//prebacujemo matricu u način MODELVIEW

  Gl.glEndList();//zatvaramo listu prikaza
}
```



Slika 7.6 Ispis autora sa imenom i prezimenom

7.4 Manipulacija grafom

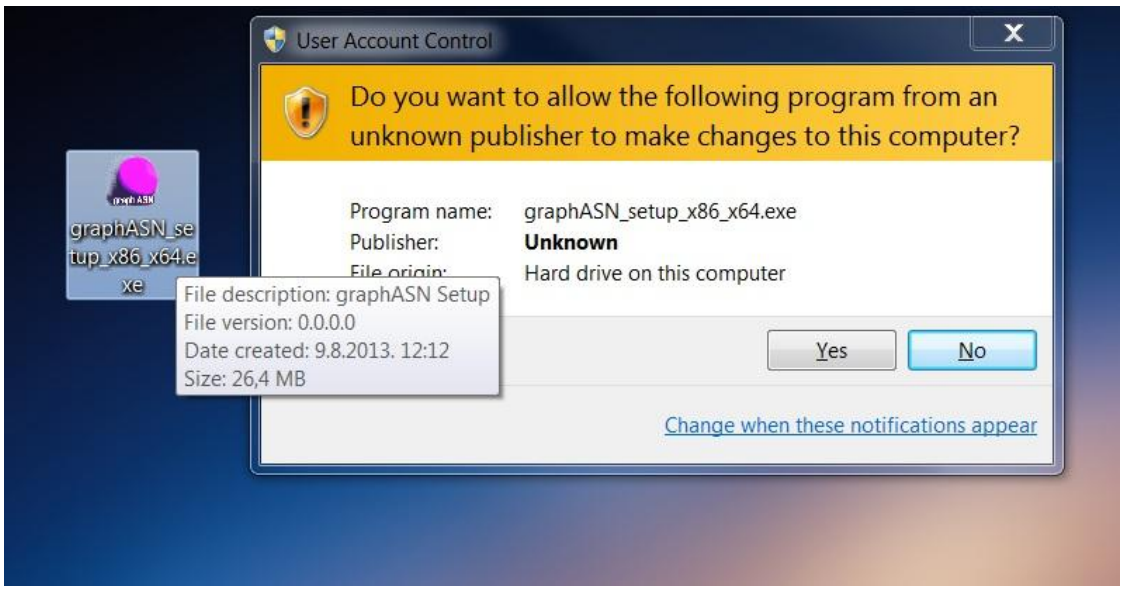
Za manipulaciju vrstom objekata poput 3D grafova, u pravilu se koristi takozvana Arcball rotacija koja omogućava korisniku rotaciju objekta poput zamišljene sfere oko svojeg središta. U projektu je korištena implementirana klasa (Kam, 2008) za rotaciju objekta uz pomoć miša. Na slici 7.7 prikazan je model Arcball rotacije koji je preuzet sa (ZZD, 2009) Uz pomoć lijevog klika miša graf se pomiče oko zamišljenog središta sfere. S druge strane, desnim klikom miša možemo pomicati graf gore, dolje, lijevo i desno, a uz pomoć srednje tipke miša možemo približavati i udaljavati graf prema kameri.



Slika 7.7 Model Arcball rotacije objekta (ZZD, 2009)

8. Upotreba aplikacije

Aplikacija je razvijena za Windows 7 32-bitnu i 64-bitnu platformu te je za pokretanje aplikacije potrebno imati instaliran Windows .NET 4.5 Framework. Instalacija na starijim Windows operacijskim sustavima nije moguća. Testiranje je izvršeno na Windows 7 32-bitnom i 64-bitnom operacijskom sustavu. Za pokretanje projekta[11] u VS 2012 potrebno je instalirati taoframework-2.1.0-setup.exe i sqlite-netFx45-setup-bundle-x86-2012-1.0.88.0.exe[12]. U nastavku poglavlja slijedi opis instalacije, ažuriranje te upotreba aplikacije.

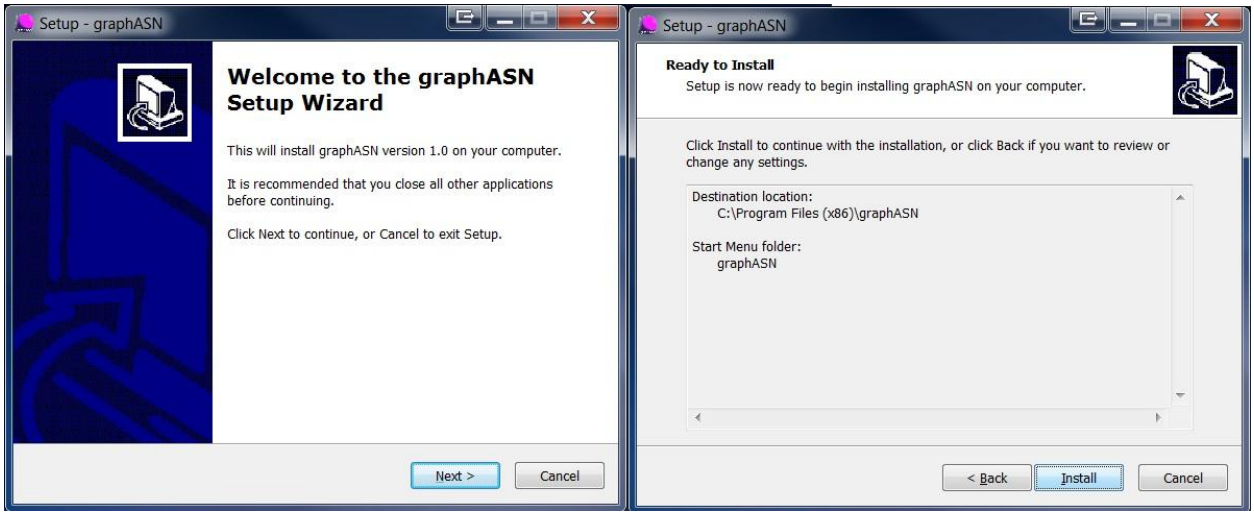


Slika 8.1 Instalacija

Na slici 8.1 prikazan je početak instalacije aplikacije. Dvostrukim klikom miša na izvršnu instalacijsku datoteku potrebno je dodijeliti administratorske ovlasti aplikaciji zbog instalacije potrebnih .dll datoteka u system32 ili syswow64 direktorij. Slika 8.2 prikazuje čarobnjaka za instalaciju aplikacije koji je implementiran uz pomoć programa InnoSetup. Prilikom klika mišem na tipku „Install“, aplikacija se instalira u odabrani direktorij te je spremna za korištenje.

[11] Svi potrebni okviri za instalaciju nalaze se u direktoriju projekta pod nazivom „Potrebni okviri“

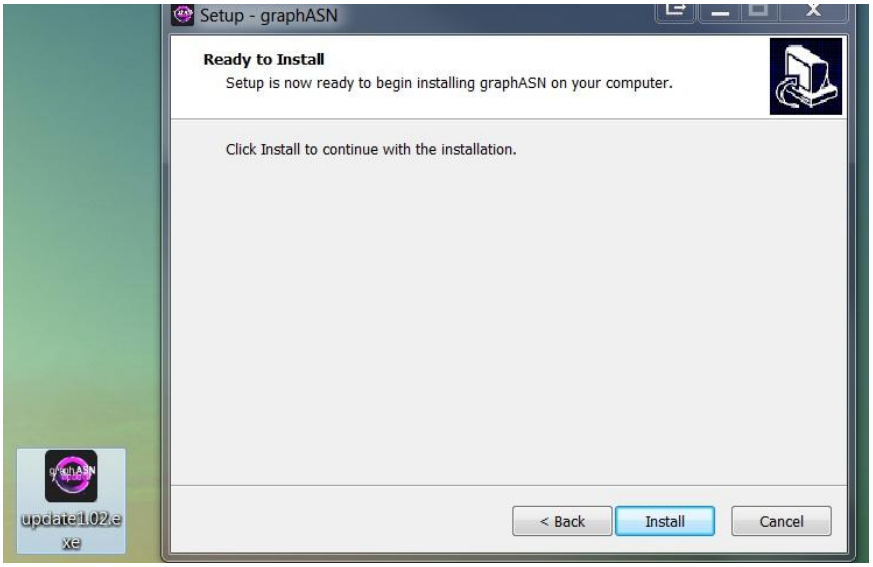
[12] bundle se nalazi na adresi : http://system.data.sqlite.org/index.html/doc/trunk/www/downloads.wiki , jedini način za pokretanje sqlite baze podataka u VS2012 je instalacija paketa sqlite-netFx45-setup-bundle-x86-2012-1.0.88.0.exe



Slika 8.2 Čarobnjak za instalaciju

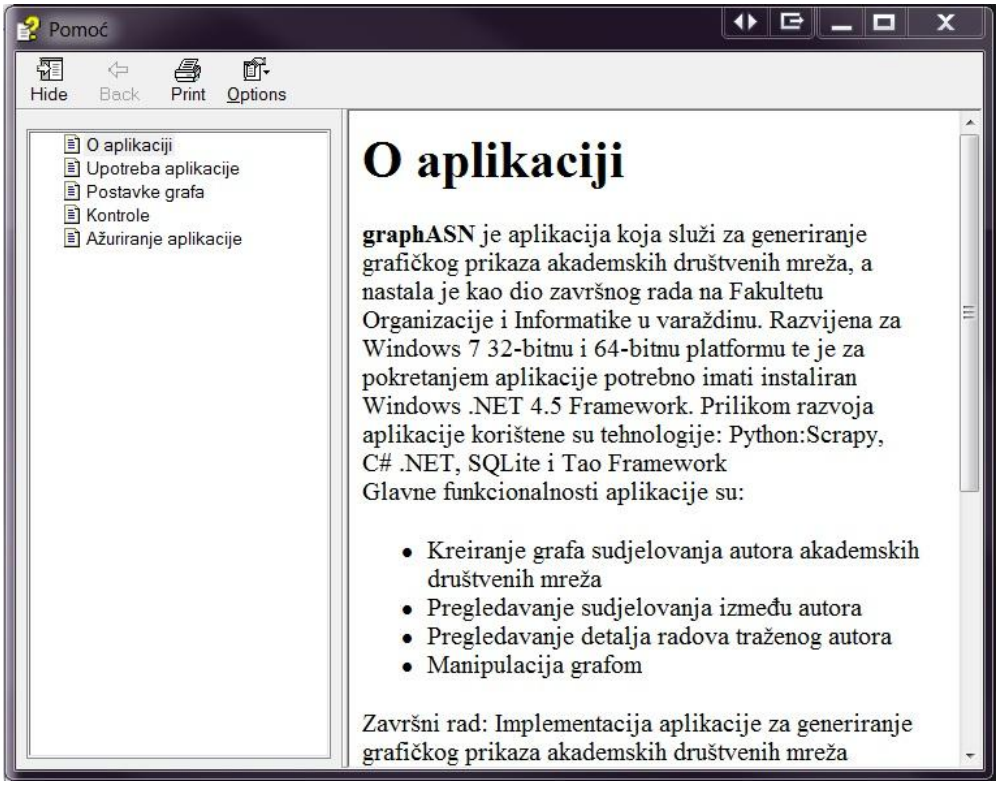
Zbog lokalne kopije web mjesta bib.irb.hr, aplikaciju je potrebno ažurirati kako bi kreirani graf prikazivao aktualno stanje. Ažuriranje je omogućeno uz pomoć zamjene stare verzije baze podataka i datoteke „bib_radovi“. Aplikacija se ne ograničava samo na projekt *Hrvatska Znanstvena Bibliografija*, nego je moguće uz pomoć naknadnog prikupljanja podataka s različitih web mjesta, na isti način kao što je to prethodno opisano u potpoglavljima 5.1-5.4, proširiti popis autora i znanstvenih radova. Postupak ažuriranja aplikacije prikazan je na slici 8.4, dok slika 8.3 prikazuje prozor s ispisanom trenutnom verzijom baze podataka.

Slika 8.3 Verzija baze podataka



Slika 8.4 Postupak ažuriranja aplikacije

Za pomoć pri korištenju aplikacijom, uz pomoć HTML Help Workshop-a implementiran je prozor s osnovnim informacijama o aplikaciji te načinu korištenja. Prozor za pomoć prikazan je na slici 8.5.



Slika 8.5 Prozor pomoći

8.1 Kreiranje grafa

Aplikaciju je moguće koristiti na dva načina:

a) Učitati već postojeći projekt uz pomoć izbornika „Opcije“ na kojem se nalazi opcija „Učitaj“. Odabirom projektne datoteke učitava se prethodno spremljen projekt.
b) Izraditi vlastiti graf uz pomoć upisivanja imena i prezimena autora. Ukoliko upisano ime nije postojeće, u kontrolu će se ispisati svi autori sa sličnim imenom i prezimenom. Nakon završene pretrage potrebno je klikom miša odabrati željenog autora iz liste ponuđenih autora. Kada se odabere autor, klikom na tipku „Pokreni“ pokreće se generiranje grafa sudjelovanja. Po završetku generiranja grafa, u statusnoj traci biti će prikazana poruka o uspješnom generiranju. U međuvremenu, korisnik ima mogućnost pregledavanja detalja o sudjelovanjima i radovima autora. Nakon prikaza grafa, moguće je istim upravljati uz pomoć kontrola opisanih u nastavku poglavlja.

Postavke grafa su sljedeće:

a) Kvaliteta Autor: označava kvalitetu sfera koje prikazuju autore[13] (1- 10 kvaliteta)
b) Kvaliteta Veza: označava kvalitetu cilindara koji prikazuju sudjelovanja (1 - 10 kvaliteta)
c) Veličina Autora: Označava veličinu sfera (1-10 veličina)
d) Veličina Veza: Označava veličinu cilindara (1-10 veličina)
e) Veličina Teksta: Označava veličinu teksta (1-6 veličina)
f) Raspored: označava vrstu pozicioniranja grafa u 3D prostoru
g) Tema: označava bojanje grafa po određenoj temi
h) Boja teksta: označava boju teksta na grafu
i) Osvježi: ponovno ocrtava graf i osvježava grafički kontekst.
j) Generiraj: ponovno generira pozicioniranje grafa i liste prikaza objekata

8.2 Kontrole grafa

Grafom je moguće upravljati u realnom vremenu. Kontrole su:

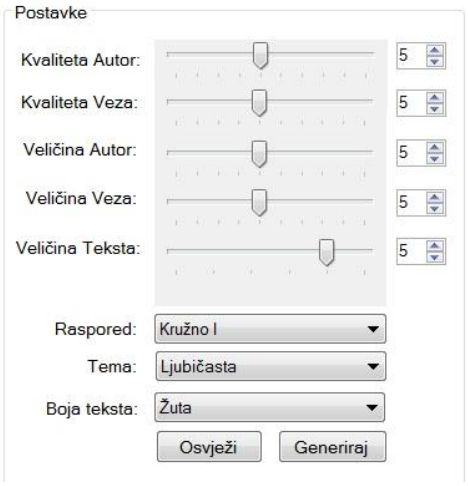
a) Desni klik miša + pomicanje dolje: graf se pomiče prema gore
b) Desni klik miša + pomicanje gore: graf se pomiče prema dolje

[13] O kvaliteti sfera ovise i performanse aplikacije prilikom generiranja grafa s većim brojem autora. Što je veći graf, poželjnije je koristiti manju kvalitetu objekata.

c) Desni klik miša + pomicanje lijevo: graf se pomiče prema lijevo
d) Desni klik miša + pomicanje desno: graf se pomiče prema desno
e) Lijevi klik miša + pomicanje: rotacija grafa uz pomoć Arcball rotacije
f) Klik sa kotačićem miša + pomicanje: detaljno zumiranje
g) Okretanje kotačića miša: zumiranje

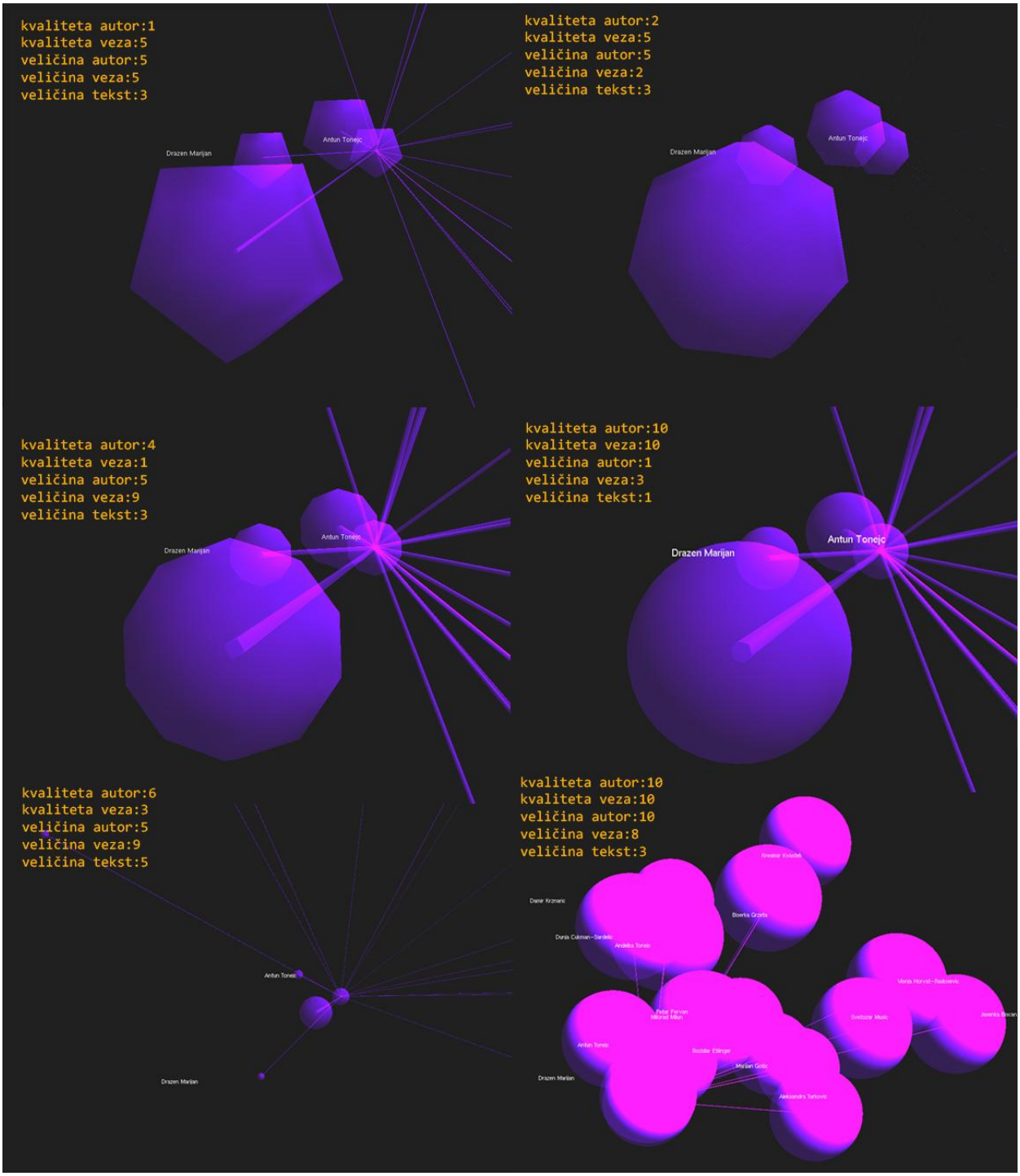
8.3 Izmjena postavki

U ovom poglavlju opisana je implementacija izmjena postavki grafa u realnom vremenu prilikom korištenja aplikacije. Korisnik je u mogućnosti mijenjati kvalitetu prikaza autora i veza te veličinu autora, veza i teksta uz pomoć kontrole „Slide“. Na slici 8.6 prikazane su kontrole grafa kojima korisnik može upravljati. Sve kontrole kreću se od 1 do 10, osim veličine teksta koja ide od 1 do 6. Prilikom interakcije korisnika s kontrolom, automatski se osvježava scena i ponovno ocrtava graf. Tipka „Osvježi“ služi za ručno osvježavanje grafa, a tipka „Generiraj“ za ponovno kreiranje grafa i resetiranje kamere.



Slika 8.6 Kontrole grafa

Na slici 8.7 prikazane su različite postavke grafa kojima je moguće kontrolirati omjer između kvalitete prikaza grafa i performansi aplikacije, ovisno o veličini grafa. Navedene postavke omogućuju korisniku prilagođavanje korištenja aplikacije prema vlastitim potrebama. Povećanjem broja vrhova, plohe sfere i cilindra izgledaju uglađenije, a time postaju vizualno ugodnije. Postavke veličine sfere i cilindra korisne su kod velikih grafova.

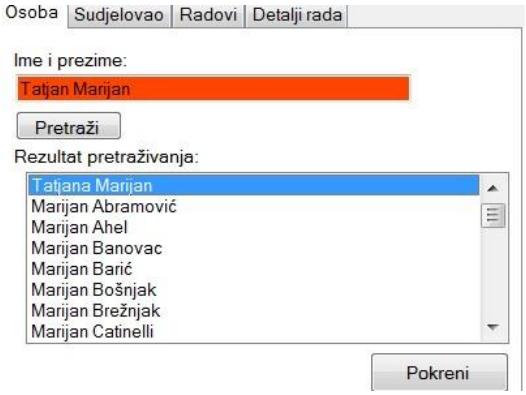


Slika 8.7 Postavke grafa

8.4 Opis slučaja korištenja

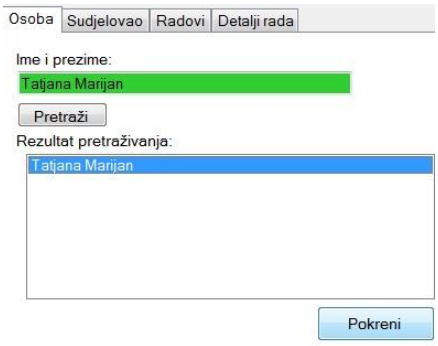
U sljedećem potpoglavlju opisan je jedan slučaj korištenja aplikacije. Pretraga prema autoru započinje upisom imena i prezimena u polje za tekstualni unos podataka. Za primjer pretrage odabrana je autorica Tatjana Marijan. Nakon pokretanja aplikacije potrebno je upisati ime i

prezime traženog autora. Ako korisnik pogriješi prilikom unosa autora, odnosno ukoliko traženi autor ne postoji u bazi podataka, polje za unos promijenit će boju u crveno. Kada u bazi podataka ne postoji traženi autor u polje rezultata pretraživanja ispisuju se autori sa sličnim imenom ili prezimenom. Opisana situacija prikazana je na slici 8.8.



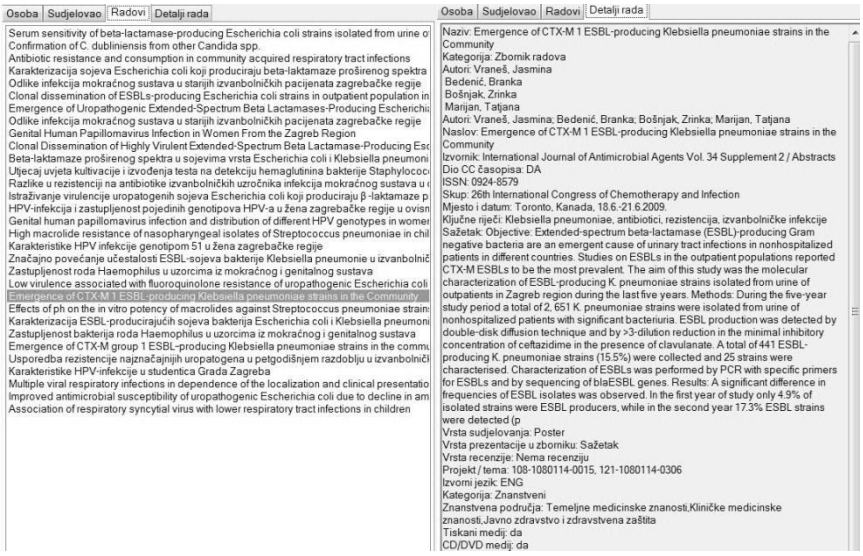
Slika 8.8 Pogrješka kod pretraživanja autora

Ukoliko se traženi autor nalazi u listi, moguće je nastaviti pretraživanje odabirom traženog autora iz liste i pritiskom na tipku „Pokreni“. Ako postoji traženi autor u bazi podatka, tada polje za unos poprima zelenu boju. Opisana situacija prikazana je na slici 8.9.



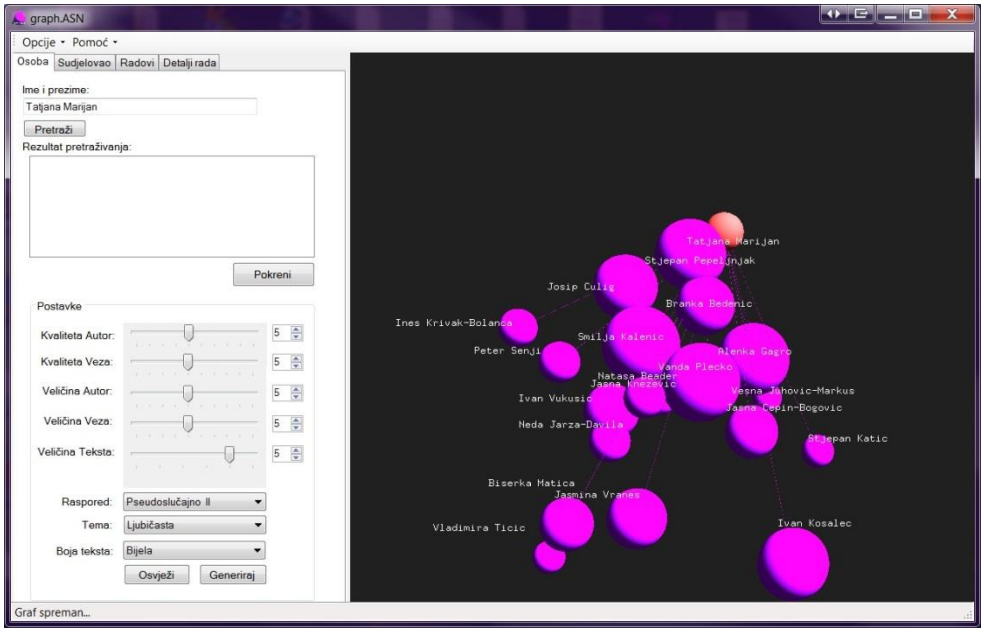
Slika 8.9 Uspješno pretraživanje

Nakon pokretanja grafa, u kartice „Sudjelovao“ i „Radovi“ ispisuju se relevantni podaci o autoru. Prilikom odabira rada iz kartice „Radovi“ u karticu „Detalji rada“ upisuju se podaci označenog rada. Korisnik je u mogućnosti pregledavati sve detalje radova traženog autora. Na slici 8.10 prikazane su kartice „Radovi“ i „Detalji rada“ za odabrani rad „Emergence of CTX-M 1 ESBL-producing Klebsiella pneumoniae strains in the Community“.

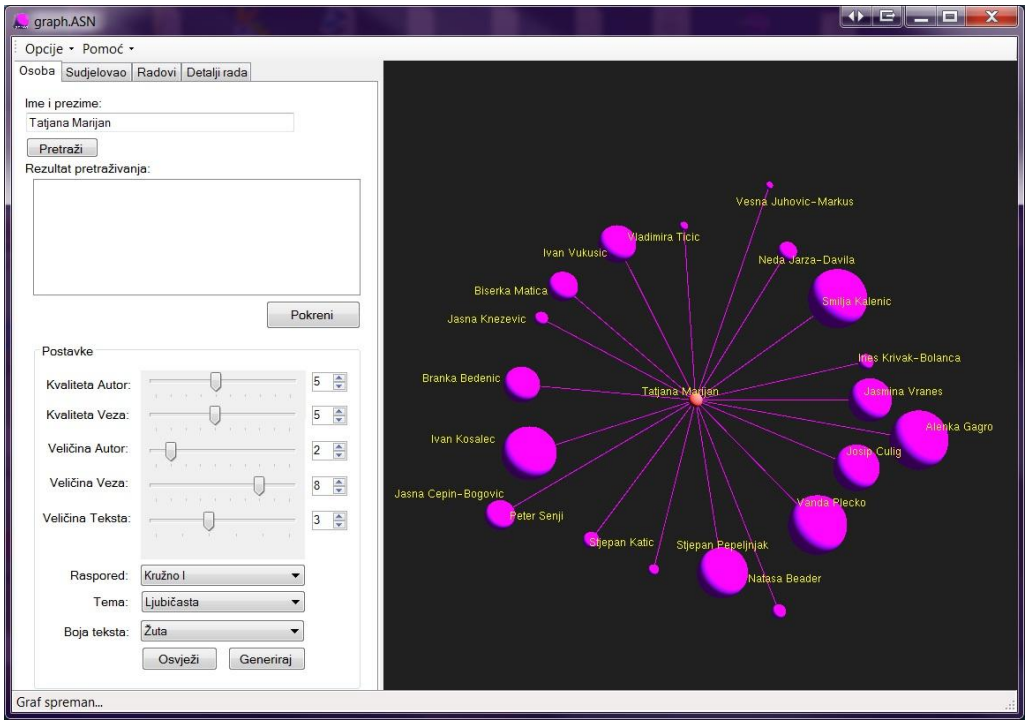


Slika 8.10 Kartice "Radovi" i "Detalji rada"

Početni generirani graf prikazan na slici 8.11 nije dovoljno čitljiv pa je potrebno izmijeniti postavke te tako prilagoditi prikaz ovisno o slučaju korištenja. „Kvalitetu Autora“ i „Kvalitetu Veza“ nije potrebno mijenjati, zato što se na grafu ne nalazi puno autora te su performanse zadovoljavajuće. Odabirom rasporeda pozicioniranja „Kružno I“ generira se čitljiviji graf, a njegov odabir u ovome slučaju zahtijeva promjenu veličine autora na vrijednost 2. Za povećanje veza, vrijednost opcije „Veličina Veza“ povećana je na 8, a za povećanje čitljivosti teksta promijenjena je njegova veličina na vrijednost 3. Ukoliko prikazani tekst nije dovoljno uočljiv zbog boje autora, tada je potrebno promijeniti opciju „Boja teksta“ u željenu boju. U ovome primjeru boja je promijenjena u opciju „Žuta“, a konačni izgled grafa prikazan je na slici 8.12.



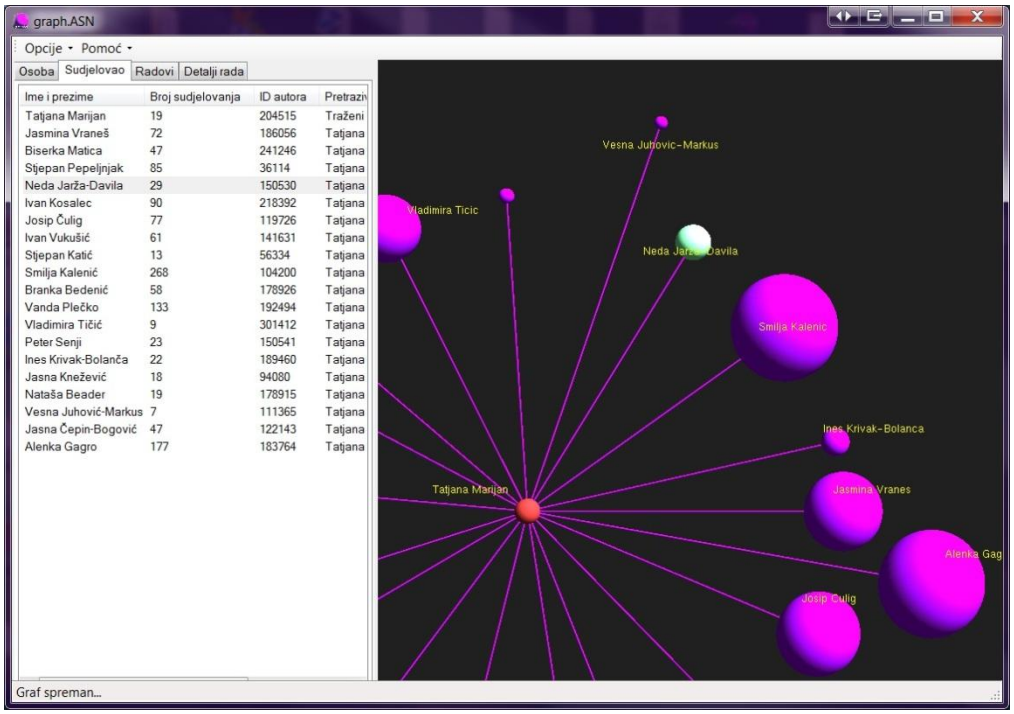
Slika 8.11 Početne postavke grafa



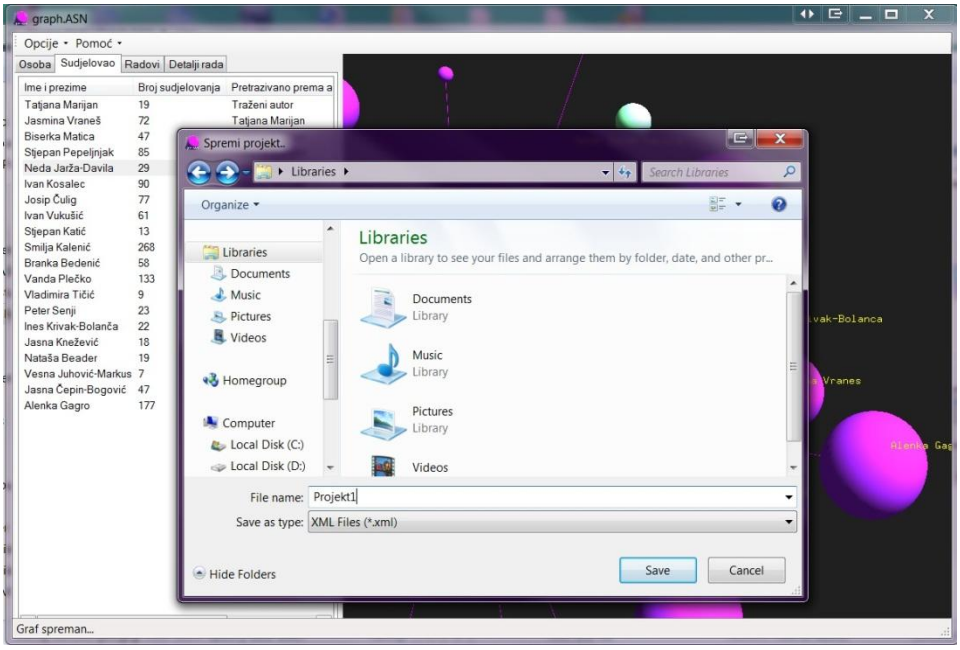
Slika 8.12 Konačne postavke grafa

Kada izgled grafa udovoljava željama korisnika, isti je u mogućnosti nastaviti s pregledavanjem podataka o traženom autoru. U kartici „Sudjelovao“ nalaze se podaci o svim autorima, a sukladno odabiru željenog autora izvršava se promjena boje čvora. Slika 8.13 prikazuje navedenu situaciju odabira autora te približavanje grafa uz pomoć kretanja mišem. U konačnici, projekt je moguće spremiti te tako sačuvati namještene postavke. Primjer spremanja projekta

prikazan je na slici 8.14. Nakon uspješnog spremanja projekta, u „statusBar“ kontrolu se upisuje poruka „Projekt uspješno pohranjen“.



Slika 8.13 Odabir autora



Slika 8.14 Spremanje projekta

9. Zaključak

U ovome radu opisana je implementacija aplikacije za generiranje grafičkog prikaza akademskih društvenih mreža. Pored toga, opisane su i akademske društvene mreže i servisi koji služe za prikaz povezanosti između autora radova. Uz pomoć različitih dijagrama prikazane su funkcionalnosti i način rada aplikacije. Za izradu praktičnog dijela rada korištene su sljedeće tehnologije: Python-Scrapy, C# .NET, SQLite, OpenGL i TAO Framework. Korištene tehnologije pogodne su za razvoj prototipa i manjih nekomercijalnih aplikacija, dok su zbog slabih performansi i neslužbene podrške OpenGL omotača za objektno orijentirani razvoj u C# programskom jeziku korištene tehnologije neprikladne za komercijalni razvoj. Praktični dio rada podijeljen je na tri faze. Prva faza je prikupljanje i obrada podataka s web mjesta projekta *„Hrvatska znanstvena bibliografija“*. Druga faza je izrada podatkovnog modula koja se sastoji od implementacije pretraživanja baze podataka, kreiranja grafa sudjelovanja i spremanja projekta. Treća faza je izrada grafičkog modula koja se sastoji od implementacije pozicioniranja grafa, kreiranja grafičkog konteksta, prikaza sudjelovanja i manipulacije grafom. Naposljetku, opisana je upotreba aplikacije i jedan slučaj korištenja aplikacijom. Razvijena aplikacija sadrži slične funkcionalnosti kao servis ArnetMiner, a to su: pretraživanje autora, pregledavanje podataka o traženom autoru i kreiranje grafa sudjelovanja. Za razliku od ArnetMiner servisa, razvijena aplikacija omogućava rad bez potrebe za povezivanjem na Internet, upravljanje postavkama grafa te spremanje i učitavanje projekta. Poboljšanju kvalitete aplikacije doprinijele bi funkcionalnosti poput direktnog preuzimanja podataka s više različitih servisa za pretraživanje akademskih društvenih mreža, pomicanje pojedinog autora na grafu te poboljšanje performansi kod prikaza većih grafova i razvoj većeg broja algoritama pozicioniranja grafa.

10. Literatura

Aggarwal, C. C. (2011). An Introduction to Social Network Data Analytics. *Social Network Data Analytics*. Hawthorne, NY, USA: Springer US. pristupano 23. srpnja 2013. na http://link.springer.com/chapter/10.1007/978-1-4419-8462-3_1#page-1

Analysis. (2010). Social Network Analysis: Theory and Applications. *http://analysis3.com/*. pristupano 23. srpnja 2013 na http://analysis3.com/Social-Network-Analysis-Theory-and-Applications-pdf-e553.pdf%20str.%202

Bastian, M., Heymann, S., & Jacomy, M. (2009). Gephi : An Open Source Software for Exploring and Manipulating Networks. pristupano 23. srpnja 2013 na https://gephi.org/publications/gephi-bastian-feb09.pdf

Boldi, P., Codenotti, B., Santini, M., & Vigna, S. (2003). UbiCrawler: A Scalable Fully Distributed Web Crawler. Milano. pristupano 24. srpnja 2013 na http://vigna.dsi.unimi.it/ftp/papers/UbiCrawler.pdf

Chang, C.-H., Kayed, M., Girgis, M. R., & Shaalan, K. (2003). A Survey of Web Information Extraction Systems. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*. pristupano 23. srpnja 2013 na http://student.bus.olemiss.edu/files/conlon/Others/Others/InformationExtraction/A%20Survey%20of%20Web%20Information%20Extraction%20Systems.pdf

Cothey, V. (2004). Web-Crawling Reliability. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE AND TECHNOLOGY, 55(14):1228–1238*. Wolverhampton, United Kingdom. pristupano 24. srpnja 2013 na http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.117.185&rep=rep1&type=pdf

Fu, T. Z., Song, Q., & Chiu, D. M. (19. Jun 2013). The Academic Social Network. *arXiv:1306.4623 [cs.SI]*. Hong Kong, Shatin, N.T. Hong Kong. pristupano 23. srpnja 2013 na http://arxiv.org/pdf/1306.4623v1.pdf

Hipp, D. R. (2013). *sqlite.org/*. Preuzeto 23. srpnja 2013 na sqlite.org/: http://www.sqlite.org/about.html

Hogan, B., Carrasco, J. A., & Wellman, B. (2007). Visualizing Personal Networks: Working with Participant-Aided Sociograms. *International Sunbelt Social Network Conference*. Vancouver. pristupano 23. srpnja 2013 na http://groups.chass.utoronto.ca/netlab/wp-content/uploads/2012/05/Visualizing-Personal-Networks-Working-with-Participant-Aided-Sociograms.pdf

Huisman, M., & van Duijn, M. A. (2003). Software for Social Network Analysis. University of Groningen. pristupano 23. srpnja 2013 na http://wiki.commres.org/pds/Project_7eFrontPage/10.1.1.102.6602.pdf

Jackson, M. O., & Watts, A. (2002). The Evolution of Social and Economic Networks. *Journal of Economic Theory 106, 106*. Caltech, Pasadena, California: Elsevier Science (USA). Preuzeto 23. srpnja 2013 na http://jmvidal.cse.sc.edu/library/jackson02a.pdf

Kam. (2008). *Arcball Module in C# - Tao.OpenGL*. pristupano 23. srpnja 2013 iz http://www.codeproject.com/: http://www.codeproject.com/Articles/22484/Arcball-Module-in-C-Tao-OpenGL

Kelleher, C. (2008). *Rendering a Cylinder Between Two Points in OpenGL*. pristupano 23. srpnja 2013 na: http://lifeofaprogrammergeek.blogspot.com/2008/07/rendering-cylinder-between-two-points.html

Microsoft. (2013). Introduction to the C# Language and the .NET Framework. *msdn.microsoft.com*. pristupano 23. srpnja 2013 na http://msdn.microsoft.com/en-us/library/vstudio/z1zx9t92.aspx

Moody, K., & Palomino, M. (2003). SharpSpider: Spidering the Web through Web Services. Cambridge: Proceedings of the First Latin American Web Congress (LA-WEB 2003). pristupano 23. srpnja 2013 na http://pdf.aminer.org/000/480/678/sharpspider_spidering_the_web_through_web_services.pdf

Newman, M., Barabási, A.-L., & Watts, D. J. (2006). The Structure and Dynamics of Networks. 1-4. Princeton, New Jersey: Princeton University Press. pristupano 23. srpnja 2013 na http://press.princeton.edu/chapters/s8114.pdf

Norman L. Biggs, E. Keith Lloyd, Robin J. Wilson. (1976). Oxford University Press, pristupano 06. rujna 2013 na http://books.google.hr/books/about/Graph_Theory_1736_1936.html?id=XqYTk0sXmpoC&redir_esc=y

Scrapy, *scrapy.org*. (2013). pristupano 24. srpnja 2013 iz http://scrapy.org/

Serrat, O. (2009). Social Network Analysis. Ithaca, NY, USA. pristupano 23. srpnja 2013 na http://digitalcommons.ilr.cornell.edu/cgi/viewcontent.cgi?article=1192&context=intl&sei-redir=1&referer=http%3A%2F%2Fscholar.google.hr%2Fscholar%3Fq%3DSocial%2BNetwork%2BAnalysis%26hl%3Dhr%26as_sdt%3D0%26as_vis%3D1%26oi%3Dscholart%26sa%3DX%26ei%3D9EECUsKb

Tang, J., Zhang, J., Yao, L., & Li, J. (2008). Extraction and Mining of an Academic Social Network. Beijing, China. pristupano 23. srpnja 2013 na http://wwwconference.org/www2008/papers/pdf/p1193-tang.pdf

The Khronos Group. (2013). About. pristupano 23. srpnja 2013 na www.opengl.org: http://www.opengl.org/about/#1

ZZD. (2009). Screen to Vector. *http://www.cnblogs.com/graphics/*. pristupano 23. srpnja 2012 na http://www.cnblogs.com/graphics/archive/2009/11/28/1612832.html