Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Gregor Šobar

Vizualizacija semantično obogatenih podatkov s pomočjo HTML5

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Dejan Lavbič

Ljubljana, 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Ideja semantičnega spleta je pripraviti podatke obstoječega spleta v strukturirani obliki, primerni za samodejno obdelavo s strani računalniških algoritmov. Na voljo je že veliko takšnih virov, ki so zapisani v standardiziranem jeziku RDF in omogočajo semantično integracijo različnih podatkovnih virov. Pri tako veliki množici, med seboj povezanih podatkov, je pregledovanje zelo oteženo. V okviru diplomske naloge se tako osredotočite na vizualizacijo in možnost sprehajanja po veliki količini podatkov, ki ne zahteva veliko tehničnega znanja in je primerno tudi za poslovne uporabnike. V ta namen izdelajte prototip pregledovalnika omrežja podatkov zapisanega v jeziku RDF. Pregledovalnik naj na vhodu prejme poljubno ontologijo, ki jo vizualizirajte in omogočite sprehajanje po prikazanem grafu, filtriranje prikazanih entitet in povezav ter izvajanje SPARQL poizvedb.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Gregor Šobar, z vpisno številko **63080035**, sem avtor diplomskega dela z naslovom:

Vizualizacija semantično obogatenih podatkov s pomočjo HTML5

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Dejana Lavbiča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 22. september 2014 Podpis avtorja:

Iskrena hvala mentorju doc. dr. Dejanu Lavbiču za pomoč in ves vložen čas ter kolektivu Laboratorija za podatkovne strukture za dostop do obširne podatkovne baze. Hvala vsem bližnjim za spodbudo in potrpežljivost. Ne nazadnje hvala tudi vsem sošolcem in prijateljem, s katerimi smo se skupaj učili, delali in zabavali.

Kazalo

Seznam uporabljenih kratic

AJPES – Agencija Republike Slovenije za javnopravne evidence in storitve.
API – angl. *Application Programming Interface*; programski vmesnik aplikacij ali storitev, ki opisuje in določa način medsebojnega sodelovanja različnih programskih komponent.
ARQ – stroj za izvajanje poizvedb, skladen z najnovejšimi specifikacijami SPARQL.
CSS – angl. *Cascading Style Sheets*; slogovni jezik za opis videza spletnih dokumentov, napisanih v označevalnem jeziku.
DOM – angl. *Document Object Model*; model za upravljanje z objekti znotraj dokumenta (spletne strani).
HTML – angl. *Hypertext Markup Language*; označevalni jezik za izdelavo spletnih dokumentov.
HTML5 – peta različica označevalnega jezika HTML.
HTTP – angl. *Hypertext Transfer Protocol*; osnovni aplikacijski protokol za prenos podatkov in informacij v svetovnem spletu.
JSON – angl. *JavaScript Object Notation*; notacija za zapis JavaScript objektov.
OWL – angl. *Web Ontology Language*; jezik za opredelitev podrobnejših ontologij.
RDF – angl. *Resource Description Framework*; osnovni standard za predstavitev podatkov na semantičnem spletu; jezik za izražanje podatkovnih modelov, ki se nanašajo na objekte in njihove odnose.
RDFS – angl. *Resource Description Framework Schema*; standardni način

za opisovanje pomena stvari.

REST – angl. *Representational State Transfer*; arhitekturni slog načrtovanja spletnih storitev.

SOH – angl. *SPARQL over HTTP*; SPARQL prek protokola HTTP.

SPARQL – angl. *SPARQL Protocol and RDF Query Language*; protokol in jezik za povpraševanje po semantičnih podatkih.

SQL – angl. *Structured Query Language*; strukturirani povpraševalni jezik za delo s podatkovnimi bazami.

SVG – angl. *Scalable Vector Graphics*; splošni standard za opis dvorazsežne vektorsko-rastrske grafike v formatu XML.

URI – angl. *Uniform Resource Identifier*; enotni identifikator vira.

W3C – angl. *World Wide Web Consortium*; mednarodna organizacija za razvoj in potrjevanje standardov za svetovni splet.

XHTML – angl. *Extensible Hypertext Markup Language*; razširjena verzija HTML, ki predpisuje strogo strukturo dokumentov XML.

Povzetek

Zadnji trendi, ki javnosti omogočajo dostop do velikih količin digitaliziranih gradiv različnih držav, raziskovalnih ustanov in nevladnih organizacij, so nas postavili pred izziv, kako vse te podatke preučiti in jih uporabiti v novih produktih ali za dodajanje vrednosti obstoječim. Tovrstne baze podatkov imenujemo tudi baze znanja, saj poleg podatkov običajno vsebujejo tudi metapodatke. Ponujajo nam možnost združevanja in primerjave informacij iz več podatkovnih skladišč, s tem pa tudi možnost pridobivanja novih vzorcev in znanj. V diplomskem delu je bil raziskan koncept semantičnega spleta, predstavljene so bile njegove prednosti, tehnologije ter prihodnji trendi razvoja. Izdelan je bil prototip aplikacije – pregledovalnik omrežja podatkov. Pregledovalnik preko poizvedb SPARQL sprejme ontologijo, zapisano v standardnem jeziku RDF, in pridobljene podatke vizualizira, omogoča pa tudi iskanje in filtriranje po specifičnih entitetah oziroma njihovih lastnostih. Vizualizacija je interaktivna, prikazani graf je mogoče podrobneje raziskovati. Implementacija aplikacije temelji na označevalnem jeziku HTML5 in različnih JavaScript knjižnicah, ki olajšajo delo pri vizualizaciji grafov in drugih funkcionalnostih. Podatki za demonstracijo prototipa so bili pridobljeni iz baze, nastale na podlagi luščenja podatkov iz Poslovnega registra Slovenije (AJPES), vsebinsko pa so bili obogateni še s podatki iz DBpedie.

Ključne besede: semantični splet, vizualizacija semantičnih podatkov in metapodatkov, vizualizacija grafa, raziskovanje po grafu, SPARQL, RDF, HTML5, JavaScript.

Abstract

Recent trends that allow public access to large volumes of digitized materials of various countries, research institutes and non-government organizations have set us the challenge of how to examine and use them in new products or add value to existing ones. Such databases are also called knowledge bases because in addition to data they usually contain metadata. They enable us the combining and comparing of information from several databases and thus the opportunity to acquire new patterns and skills. In the thesis the concept of semantic web has been studied and its advantages, technology and future development trends presented. A prototype of the application – a data network browser – was made. The browser accepts the ontology, written in the standard RDF language, via SPARQL queries and visualizes the information obtained. It also allows searching and filtering for specific entities and their properties. The visualization is interactive and the displayed graph can be explored further. The implementation of the application is based on the HTML5 markup language and various JavaScript libraries which help with the visualization of graphs and other functionalities. Data for the demonstration of the prototype was obtained from a database generated by the extracting of data from the Business Register of Slovenia (APLRS), the content being enriched further by information from DBpedia.

Keywords: semantic web, visualization of semantic data and metadata, graph visualization, graph exploration, SPARQL, RDF, HTML5, JavaScript.

Poglavje 1

Uvod

Internet trenutno uporablja že 2,9 milijarde ljudi, kar je 40,8 % vsega svetovnega prebivalstva, ki je trenutno ocenjeno na 7,1 milijarde posameznikov. Še leta 1995 je bilo vseh uporabnikov le 16 milijonov, torej le 0,4 % vsega svetovnega prebivalstva [19].

Širše zanimanje za internet je spodbudila njegova komercializacija na začetku 90-ih let 20. stoletja, k izrednemu povečanju zanimanja zanj pa je v veliki meri prispeval prav prihod t. i. svetovnega spleta (angl. *World Wide Web*). V splošni javnosti je svetovni splet postal celo sinonim za internet. Uporaba interneta se je iz predvsem vojaških ter znanstvenih krogov začela širiti tudi v civilno družbo. Ljudje so ga hitro vzljubili, saj je omogočal izjemno hitro širjenje in izmenjavo informacij, predvsem pa komunikacijo ne glede na fizično oddaljenost med ljudmi ali napravami, prek državnih meja in ne oziraje se na jezik.

Od javnega predloga svetovnega spleta je minilo le okoli 25 let [12]. Širše zgodovinsko gledano je to obdobje izjemno kratko, a izjemno pomembno za človeštvo. Svetovni splet je namreč zaslužen za številne posredne in neposredne globalne spremembe: od družbenih, ekonomskih do političnih in drugih.

Splet, kot ga poznamo danes, je sicer nastal iz preproste ideje o avtomatiziranem deljenju informacij med znanstveniki na univerzah in inštitutih po vsem svetu. Osnovni namen je bila vzpostavitev decentraliziranih medijev,

prek katerih bi bilo mogoče deliti informacije, ki bi bile v vsakem trenutku dosegljive vsem povezanim v isto omrežje. Danes to omrežje imenujemo medmrežje ali internet.

Sprva so se po spletu izmenjevali hipertekstovni dokumenti (danes jih imenujemo spletne strani), ki so bili na pogled precej preprosti. S hitrim širjenjem uporabe spleta se je širil tudi nabor tehnologij. Dokumenti so čez čas postajali vedno bolj interaktivni in vizualno bogati, kar pa je nekaterim ljudem predstavljalo težavo. Taki dokumenti so namreč vsebinsko razumljivi človeku, računalniku pa neposredno ne.

Zato je majhna skupina ljudi skupaj s samim idejnim očetom svetovnega spleta Timom B. Leejem leta 2001 objavila predlog o nadgradnji svetovnega spleta. Predlagali so, kot so ga sami poimenovali, semantični splet (angl. *Semantic Web*). Osnovna ideja semantičnega spleta je izboljšati trenutni splet tako, da bi računalniki lahko obdelovali podatke na spletu, jih interpretirali, povezali in ljudem še bolj olajšali iskanje zahtevanega znanja [5]. Za to je treba podatke v okviru različnih spletnih strani in dokumentov zapisati na enoličen in strukturiran način. Namen takega pristopa je računalnikom omogočiti iskanje po vseh straneh spleta kot po eni, izredno veliki bazi podatkov. Koncept v praksi omogoča, da na podlagi podatkov z ene strani vsebinsko obogatimo druge – iz različnih virov, a s pomensko povezano vsebino.

Ker so pred prihodom omenjenega koncepta baze podatkov sprva temeljile na zapisu v obliki tabel, je očitno, da tabele v ta koncept ne spadajo. Zaradi drugačnega pristopa so se začele razvijati drugačne vrste baz, predvsem take, ki omogočajo zapis podatkov v obliki grafov – posledično seveda to vključuje tudi iskanje, urejanje in brisanje. Nastale so različne tehnologije, ki so danes znane kot RDF, OWL, SPARQL ipd.

Semantični splet je bil dolgo časa zapostavljen, odmaknjen v izobraževalno-raziskovalno sfero. Zdi se, da sta šele odpiranje in digitalizacija javnih podatkov različnih vlad ter vladnih in nevladnih organizacij ponovno obudila širše zanimanje zanj. Interes zanj so pokazala tudi komercialna tehnološka

podjetja, saj semantični splet predstavlja novo področje, ki omogoča visoko dodano vrednost. Zgolj objavljanje podatkov pa seveda ni dovolj. Pomembna dejavnost je luščenje podatkov in metapodatkov iz obstoječih dokumentov (fizičnih dokumentov, spletnih strani itd.) ter njihovo zapisovanje v t. i. baze znanj. Ne nazadnje pa so potrebne tudi aplikacije in orodja, ki bi omogočala uporabo oziroma raziskovanje po teh bazah.

Potencial, ki ga zgoraj omenjeni koncepti in sama tehnologija prinašajo, na prvi pogled mogoče ni viden. Nekaj predlogov uporabe je mogoče prebrati v članku "The Semantic Web in Action"[6], nekaj pa jih bomo spoznali tudi v tem dokumentu. Namen diplomskega dela je, da bi bilo znanje, ki ga baze s takimi zapisi vsebujejo in ponujajo v uporabo, lažje dostopno tudi širši javnosti. Zato smo v okviru diplomskega dela implementirali prototipno aplikacijo, ki s pomočjo posebnega grafičnega vmesnika omogoča intuitivno, vizualno enostavno in uporabniku prijazno raziskovanje in iskanje po bazi znanja. Podatke za demonstracijo prototipa smo pridobivali iz baze, ki je nastala na podlagi luščenja podatkov iz Poslovnega registra Slovenije (AJPES), vsebinsko pa so bili obogateni še s podatki iz DBpedie.

V nadaljevanju (v poglavju 2) bomo sprva spoznali nekaj obstoječih orodij in aplikacij. Po predstavitvi našega predloga rešitve (v poglavju 3) bomo poskušali podrobneje obrazložiti njegovo teoretično ozadje (v poglavju 4). Sledili bodo praktični del s predstavitvijo prototipa (v poglavju 5) in sklepne ugotovitve (poglavje 6).

Poglavje 2

Pregled sorodnega dela

Razvoj tehnologij za semantični splet je napredoval zelo počasi, temu primerno skromen je bil tudi prvi odziv razvijalcev orodij in aplikacij. S časom pa se je tudi na tem področju stanje izboljšalo. V zadnjih letih izredno hitro napreduje tako razvoj raziskovalnih kot razvoj komercialnih produktov [8].

Poznamo številne načine prikaza semantičnih podatkov, vendar jih je večina namenjenih določeni uporabi in določenim podatkovnim množicam. Opažene načine prikaza podatkov v aplikacijah in orodjih lahko okvirno razvrstimo v dve kategoriji.

V prvi kategoriji so aplikacije in orodja, ki prikazujejo podatke v besedilnem načinu, tj. v obliki tabel, seznamov in podobno. Pri teh je običajno prikazan en sam vir, poleg njega pa seznam vseh njegovih lastnosti. Pri izbiri enega od njih se običajno odpre okno z novim seznamom. Slabost takega pristopa je tudi, da ni mogoče prikazati prave strukture grafa povezav, ki sicer obstajajo med elementi, zapisanimi v RDF.

V drugo kategorijo uvrščamo aplikacije in orodja, ki prikazujejo podatke v grafičnem načinu. Nekatere omogočajo prikaz v grafikonih, večina pa v obliki grafa. Pri grafičnem prikazu je pogosto na sredini zaslona le en opazovani vir s svojimi lastnostmi. Nekatere aplikacije omogočajo tudi prikaz bližnje soseščine drugih virov. Dodatne nastavitve prikaza podatkov na zaslonu pogosto niso mogoče. Tudi navigacija po grafičnih elementih pogosto

ni najbolj intuitivna.

Pomemben parameter, ki razlikuje aplikacije na semantičnem spletu, je nivo, na katerem aplikacije upravljajo podatke. Kot izpostavlja [7], lahko pristope k vizualizaciji zbirke virov razdelimo v naslednje skupine:

- nivo zbirke (ang. *collection level*);
- nivo vira (ang. *resource level*);
- nivo znotraj vira (ang. *intra-resource level*).

Nivo zbirke vizualizira lastnosti zbirke. Te vizualizacije so običajno namenjene zagotavljanju splošnega pregleda vsebine zbirke, nivo se uporablja tudi za napovedovanja. Nivo vira prikazuje lastnosti posameznih virov in vizualizira povezave med njimi, torej prikaže več podrobnosti o določenem viru. Nivo znotraj virov je usmerjen k vizualizaciji notranje zgradbe vira tj. k porazdelitvi tem in lastnosti v viru. Uporablja se predvsem za podrobnejšo analizo. V diplomskem delu bomo obravnavali predvsem prikaz na nivoju vira.

V tem poglavju se bomo osredotočili na splošna orodja za brskanje po zapisih RDF s poudarkom na njihovi vizualizaciji. Eden največjih seznamov takih orodij je dosegljiv na uradni strani organizacije za razvoj in standardizacijo semantičnih tehnologij W3C [20]. Tu ne bomo obravnavali prav vseh rešitev. Predstavili bomo le nekaj tistih, ki vsebujejo nadvse širok spekter različnih pristopov.

Anzo Suite

Družba Cambridge Semantics ponuja programsko opremo – odprto platformo, imenovano Anzo. Anzo je prilagojen za posebne poslovne namene. Osredotočen je na končnega uporabnika, ki ima potrebo po zbiranju (urejenih ali neurejenih) podatkov iz različnih virov in njihovi hitri ter učinkoviti obdelavi. Prikazi, ki so na voljo v osnovni rešitvi, so predvsem tabele, gra-

fikoni in podobno. V okviru tega so na voljo trije izdelki: Anzo Data Collaboration Server (strežnik za hrambo in obdelavo podatkov), Anzo on the Web (uporabniški vmesnik odjemalca) ter Anzo for Excel (vtičnik za dodatni dostop, vnos in analizo podatkov s strežnika). Platforma je razvita na osnovi programskih jezikov JavaScript, Java in .Net. Koda ni prosto dostopna. Fizičnim osebam ponujajo v brezplačno uporabo osnovno različico, poslovni uporabniki pa morajo plačati licenčno. Razvijalci s to platformo merijo predvsem na večja podjetja, še posebej finančno industrijo [21]. Po našem mnenju gre za sicer precej intuitiven grafični vmesnik, vendar zaradi obsežnosti in moči analitičnih ter ostalih komponent potrebuje precej prilagoditev in nastavitev. Uporaba je zato vse prej kot enostavna.

Cytoscape z vtičnikom RDFScape

Cytoscape je programska platforma, razvita na ameriškem Inštitutu za biološke raziskave (*Institute for Systems Biology*). Sprva je bila namenjena predvsem vizualizaciji in analizi interakcij molekularnih mrež ter integraciji teh interakcij z genskimi profili in drugimi podatki. Zaradi precejšnjega zanimanja za uporabo orodja tudi v drugih vejah znanosti razvijalci njegov razvoj vedno bolj usmerjajo k splošni odprti vizualizacijski platformi. Platforma je razvita v programskem jeziku Java in je odprtokodna, licenca LGPL (*GNU Lesser General Public License*) [22]. Gre za platformo, katere glavna prednost je vizualizacija kompleksnih omrežij podatkov, predvsem v obliki grafa. Osnovne funkcionalnosti je mogoče razširiti z dodatnimi vtičniki. RDFScape je eden izmed vtičnikov, ki omogoča vključitev funkcionalnosti semantičnega spleta v to okolje. Omogoča poizvedovanje, vizualizacijo in razlaganje ontologij, zastopanih z OWL in RDF. Vizualizacija je v obliki grafa. Vtičnik je odvisen tudi od semantičnega ogrodja Jena [23].

VisualDataWeb

VisualDataWeb predstavlja skupino orodij za vizualizacijo. Obsega kar nekaj naprednih orodij, kot so RelFinder (iskalnik relacij med objekti), SemLens (analizator trendov in korelacij med podatki v RDF), gFacet (generator poizvedb po podatkih na podlagi izbranih pogojev), gFacet (brskalnik po hierarhiji podatkov RDF). Pridobljene podatke prikazuje v obliki tabel, grafov in razsevnih grafikonov. Med rezultati omogoča tudi interakcijo in poglobljeno raziskovanje. Namestitev in uporaba sta namenjeni naprednejšim uporabnikom. Izdelovalci orodje oglašujejo kot raziskovalni prototip, zato zanj tudi ne ponujajo posebne podpore. Paket orodij je treba kot aplikacijo namestiti na računalnik (oziroma na strežnik), sicer pa odjemalčev vmesnik deluje v brskalniku. Toda ker temelji na tehnologiji Adobe Flex, za uporabo v brskalniku potrebuje dodaten vtičnik, predvajalnik Adobe Flash, zato ne deluje na mobilnih napravah. Zadnje uradne posodobitve so iz leta 2011. Koda je dostopna pod licenco GNU GPL [24].

RDF Gravity

RDF Gravity je aplikacija za vizualizacijo ontologij RDF/OWL. Glavne oglaševane funkcionalnosti so vizualizacija grafa, globalni in lokalni filtri, iskanje z nizi besed, ustvarjanje pogledov iz poizvedb RDQL (poizvedovalni jezik za RDF, podoben SPARQL) ter vizualizacija. Aplikacija sprejema le vhodne podatke iz datotek, ne pa tudi iz zunanjih dostopnih točk. Implementacija temelji na programskem jeziku Java, JUNG Graph API ter semantičnem ogrodju Jena. Za uporabo je treba aplikacijo namestiti na računalnik, izvaja se v okolju Java WebStart. Aplikacija je dostopna za brezplačno uporabo, vendar ni odprtokodna [25].

Rhizomer

Rhizomer je orodje za izdelavo spletnih strani na podlagi tehnologij semantičnega spleta. Projekt upravlja skupina GRIHO, ki deluje pod okriljem Oddelka za računalništvo in industrijsko inženirstvo (*Computer Science and Industrial Engineering Department*) Univerze v Lleidi. Strežniški del upravlja koda, napisana v programskem jeziku Java, odjemalski del pa je napisan v programskem jeziku JavaScript in deluje kot spletna aplikacija. Razvijalci v splošnem oglašujejo, da omogoča objavljanje, poizvedovanje, brskanje, urejanje in interakcijo s semantičnimi podatki. Vendar sami tik pred prenosom kode opozarjajo, da gre za predhodno izdajo, ki ne zagotavlja vseh oglaševanih funkcionalnosti in stabilnega delovanja. Tudi med referencami navajajo zgolj nekaj prototipnih namestitev. Koda je bila zadnjič posodobljena julija 2013. Osnovne funkcionalnosti so razdelili na t. i. Overview, Zoom & Filter ter Details. Overview vključuje navigacijske menije in drevesne sezname. Zoom & Filter glede na podane pogoje izpisuje podatke v obliki seznama oziroma tabele, lahko pa podatke na zahtevo prikazuje tudi v obliki časovnice, na zemljevidu (v primeru dostopnosti podatkov z GPS koordinatami) ter v grafikonu. Koda je dostopna pod licenco GNU GPL v3 [26].

Sgvizler

Sgvizler je JavaScript knjižnica, ki omogoča različne načine vizualizacije semantičnih podatkov. V osnovi deluje tako, da uporabnik oziroma spletna aplikacija sama pripravi poizvedbo SPARQL. Poizvedba se iz spletne aplikacije pošlje v naprej določeno dostopno točko SPARQL. Dostopna točka mora rezultate vrniti v notaciji XML ali JSON. Sgvizler podatke razčleni in preoblikuje tako, kot zahteva Google Visualization API. Za izris vseh grafičnih elementov se namreč uporablja izključno spletno vizualizacijsko ogrodje Google Charts. Rešitev je zelo zanimiva zaradi enostavne implementacije. Ponuja kar nekaj možnosti vizualizacije podatkov (časovnica, tabela,

tortni grafikon, mehurčni grafikon, histogram, razpršeni grafikon, zemljevid in podobno), vendar je ta po drugi strani grafično omejena ravno zaradi ponudbe Google Charts knjižnice. Ne ponuja na primer prikaza v obliki grafa. Rešitev je odprtokodna in v stabilni različici. Avtorji še vedno skrbijo za občasne popravke [27].

Primerjava obravnavanih del

V predstavljenem seznamu orodij lahko opazimo velik razkorak med pristopi in nameni njihove uporabe. Orodja segajo vse od enostavnih aplikacij do zelo obsežnih platform. Nekatera so namenjena zgolj za enostavne vizualizacije, druga pa podpirajo celo analitične in druge zahtevnejše funkcionalnosti. Kljub temu se večino upravlja preko namiznih aplikacij. Velik delež slednjih je razvit v programskem jeziku Java. Opaziti je mogoče tudi, da je bila večina orodij razvitih v izobraževalno-raziskovalnih ustanovah. Zadnje čase so vedno pogostejši tudi komercialni izdelki, ki pa so na področju semantičnega spleta običajno bolj usmerjeni in specializirani za reševanje specifičnih problemov in namenjeni sorazmerno omejeni uporabniški skupini.

Vizualizacijske tehnike so predvsem pri orodjih, usmerjenih predvsem v analitično obdelavo podatkov, preproste. Opazimo predvsem tabelarične prikaze, razsevne grafikone oziroma grafikone na splošno. Pri raziskovalno usmerjenih ali preprostih pregledovalnikih pa prevladuje predvsem tabelarični prikaz in prikaz v obliki grafov.

Poglavje 3

Predlog lastne rešitve

Čeprav ideja o pregledovalnikih semantičnih podatkov ni več tako sveža, pa so bila prva orodja še nedavno zelo okorna. Področje se je začelo zares prebujati šele po skoraj desetletju relativnega mirovanja. V prejšnjem poglavju smo pregledali in predstavili širši spekter različnih orodij in njihovih namenov uporabe. Večina je namenjena zahtevnejšim uporabnikom ali pa so okorna in neintuitivna ter tako neprimerna za splošno uporabo. Težava zahtevnejših orodij je še posebej potreba po nameščanju aplikacij oziroma vtičnikov na računalnike. Uporaba na mobilnih napravah zaradi tega ni mogoča. To, po našem mnenju, za doseganje širšega kroga občinstva ni zaželeno, saj lahko zahteva po nameščanju dodatkov deluje kot odklonilni dejavnik.

Na podlagi teh informacij nam je uporaba spletne aplikacije predstavljala najboljši kompromis – posebej pa smo poudarili, da mora spletna aplikacija delovati brez dodatnih vtičnikov, torej zgolj z osnovnim spletnim brskalnikom.

Kot smo že v uvodu omenili, je bil cilj diplomskega dela narediti pregledovalnik omrežja podatkov, torej aplikacijo, ki bi tudi tehnično neusposobljenemu uporabniku omogočila preiskovanje izbrane strukturirane baze podatkov – tako sheme kot tudi samih instanc (entitet in njihovih lastnosti), zapisanih v RDF. S pomočjo posebnega grafičnega vmesnika bi omogočala iskanje in raziskovanje po podatkih RDF na vizualno prijazen način. Pregle-

dovalnik mora biti napisan za širšo uporabo.

Glavne zahteve

Na podlagi zgoraj navedenega smo si pred razvojem prototipne aplikacije zadali naslednja glavna merila.

- Spletna aplikacija na osnovi HTML5.
- Vizualizacija trojčkov RDF na podlagi avtomatično ustvarjenih poizvedb SPARQL.
- Vizualizacija tudi ročno napisanih poizvedb SPARQL – vnešenih preko aplikacije.
- Grafični vmesnik mora omogočati sprehajanje po prikazanem grafu.
- Možnost iskanja in filtriranja podatkov:
  - podatki so vedno sestavljeni iz sheme in dejanskih podatkov; uporabnik bi, glede na shemo, imel možnost prikazovati izbrane podatke (v ozadju se ustvari poizvedba SPARQL) – na primer izpis vseh javnih zavodov (Javni zavod je razred, ki je določen v shemi AJPES) na izbranem naslovu;
  - če uporabnik o izbrani točki na grafu želi izvedeti več, se v ozadju izvede nova poizvedba SPARQL in vrne podatke z izbrano točko v središču.
- Konfiguracija prikaza:
  - možnost izbire prikaza po lastnostih;
  - barvno razlikovanje prikaza različnih vozlišč grafa glede na tip oziroma vrsto podatkov.
- Grafični vmesnik mora biti odziven, čim bolj preprost in pregleden.
- Možnost menjave virov baz znanja (zamenjava dostopne točke SPARQL).

Dodatne podrobnosti

- Izhajati je treba iz sheme dostopne točke, ki se prikaže ob vstopu v aplikacijo.
- Z izbiro koncepta se prikažejo dodatne informacije (npr. podatkovne in objektne lastnosti).
- S pomočjo tega določimo filter iskanja in v ozadju ustvarimo poizvedbo SPARQL.
- Pri vizualizaciji je treba upoštevati splošne lastnosti beleženja podatkov (na primer anotacij rdfs:comment in rdfs:label).
- Možnost, da za neko obstoječe vozlišče v grafu prikažemo vse informacije v rangu n (en, dva, trije koraki naprej oziroma v globino).
- Dodatni podatki iz zunanjih virov (npr. Wikipedija – prek "SPARQL Service").
- Omejitev dolžine besedila vozlišč (celotno besedilo se prikaže ob kliku na vozlišče, ob izpustitvi tipke se besedilo ponovno skrči).

Poglavje 4

Predstavitev področja

V tem poglavju bomo spoznali koncepte, tehnologije in orodja, ki jih je treba poznati in razumeti za uspešno izdelavo rešitve na zastavljeni način. Sprva bomo predstavili širši pogled na koncepta "osnovnega" (svetovnega) in semantičnega spleta. Nadalje si bomo ogledali raznovrstne tehnologije semantičnega spleta, ki se obravnavajo kot razširitev svetovnega spleta, namenjene pa so računalnikom za enostavnejše razumevanje in dostop do podatkov. Poleg podatkov, predstavljenih s tehnologijami semantičnega spleta, in dostopa do njih bomo predstavili tudi orodja za njihovo hrambo in distribuiranje. V tem sklopu bodo predstavljene tudi vse uporabljene spletne tehnologije. Na njih namreč temelji odjemalčev del naše rešitve vključno z vizualizacijo prej omenjenih podatkov.

4.1 Zgodovinski mejniki svetovnega spleta

Za razumevanje koncepta t. i. svetovnega in semantičnega spleta – kaj sta in kako delujeta ter kakšne tehnologije uporabljata – se nam zdi pomembno poznavanje njunih širših okoliščin. Podrobnosti o nastanku spleta in vseh pripetljajev v njegovi zgodovini v tem delu ne bomo obravnavali, omenili bomo le zgodovinske mejnike.

V [28] N. Spivack mejnike razdeli na t. i. Splet 1.0, Splet 2.0 ter Splet

3.0. Splet 1.0 označuje kot "svetovni splet", Splet 2.0 kot "družbeni splet" ter Splet 3.0 kot "semantični splet". Za nekoliko bolj oddaljeno prihodnost, trenutno šele kot zametek ideje, načrtuje t. i. "inteligentni splet", ki naj bi ga označeval izraz Splet 4.0.

Naj opozorimo, da so te besedne skovanke nastale pozneje od idej in tehnologij, ki jih opredeljujejo. Prav tako ne predstavljajo pravih verzij v smislu zamenjave starejših z novejšimi. Vse tri obstoječe verzije namreč še vedno sobivajo v istem okolju. Poimenovanja opredeljujejo le večje tehnološke premike v evoluciji spleta. Tehnološki napredek pa je pri spletu opredeljen glede na nove funkcionalnosti.

Na sliki 4.1 so poimenovanja umeščena glede na funkcionalnosti in tehnologije, ki jih opredeljujejo. Pod časovno premico so predstavljene pomembnejše funkcionalnosti, nad njo pa nove tehnologije, ki te funkcionalnosti omogočajo. Slika hkrati prikazuje trenutne trende ter nakazuje, kaj se nam obeta. Pomemben je tudi odnos med osjo x ter y. Os y prikazuje razvoj informacijskih tehnologij v smislu rasti povezav (povezanosti) med informacijami, os x pa predstavlja rast družbene povezanosti med ljudmi prek novih tehnologij.

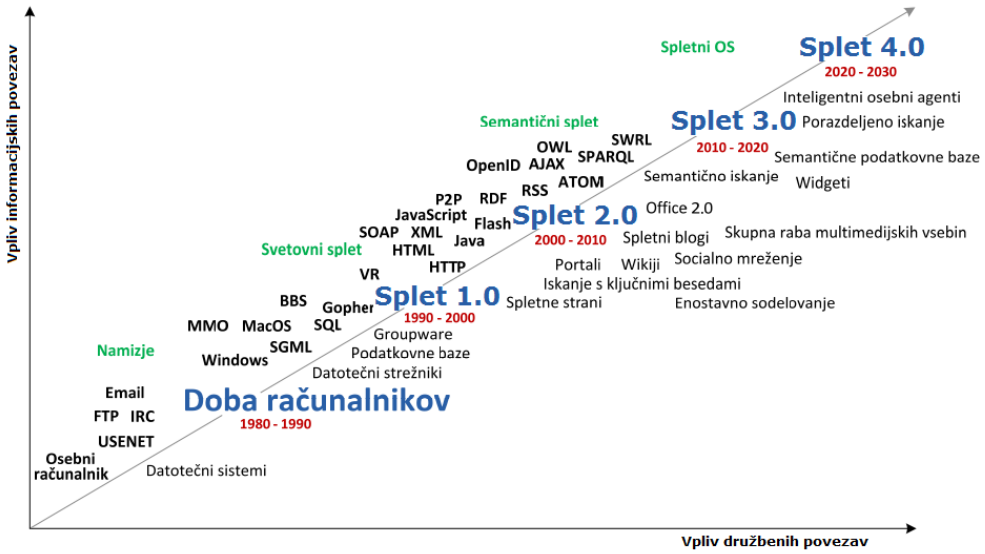
Mejniki glede na funkcionalnosti

Splet 1.0 – z današnjega stališča je prikaz informacij zelo suhoparen, spletni dokumenti so sestavljeni iz osnovnega besedila, tabel, povezav med različnimi spletnimi viri ter iz slik.

Splet 2.0 – zaznamuje obdobje bogatejših vsebin, interaktivnosti in udeležbe uporabnikov, ki so prišli iz položaja gledalcev v položaj soustvarjalcev vsebin. Sem spadajo predvsem spletni dnevniki, forumi, klepetalnice, takojšnje sporočanje, video in zvočni repozitoriji itd.

Splet 3.0 – semantično in storitveno usmerjen splet. Zmogljivosti spletnih aplikacij se približujejo zmogljivostim namiznih. Množice obstoječih podatkovnih baz se preslikuje v semantične. Številne storitve v oblaku uporabljajo semantično povezane podatkovne baze in ustvarjajo storitve z visoko

dodano vrednostjo.



Slika 4.1: Evolucija spleta. Povzeto po [28].

4.2 Tehnologije semantičnega spleta

4.2.1 Predstavitev semantičnega spleta

Trenutni splet vsebuje številne podatke, informacije in znanja, vendar je vloga računalnikov v tem trenutku zgolj dostavljanje in predstavitev vsebine dokumentov, ki opisujejo to znanje. Ljudje moramo sami povezati vse vire pomembnih informacij in jih tudi interpretirati. Osnovna ideja semantičnega spleta je izboljšati trenutni splet tako, da bodo računalniki lahko obdelovali podatke na spletu, jih interpretirali ter ne nazadnje tudi povezali. Ljudem bi na takšen način olajšali iskanje zahtevanega znanja in ga naredili učinkovitejšega.

Ker svetovni splet predstavlja porazdeljen sistem med seboj povezanih dokumentov, želi semantični splet vzpostaviti porazdeljen sistem med seboj povezanih podatkov v bazah znanja. To je obsežen projekt, saj gre za sode-

lovanje, ki se v splošnem nanaša na vse skrbnike spletnih strani in njihova podatkovna skladišča. Vodi ga organizacija W3C, ki skrbi tudi za vzpostavitev skupnega ogrodja, ki bi omogočalo skupno rabo podatkov iz različnih virov [10].

Semantični splet (angl. *Semantic Web*) je skovanka Tima Bernersa-Leeja, Jamesa Hendlerja in Ora Lassila, njegovih idejnih pobudnikov. V različnih člankih ga omenjajo tudi kot splet podatkov (angl. *Web of data*). Predlog o razvoju svetovnega spleta, imenovanega semantični splet, so leta 2001 objavili v članku "The Semantic Web" v reviji Scientific American [5].

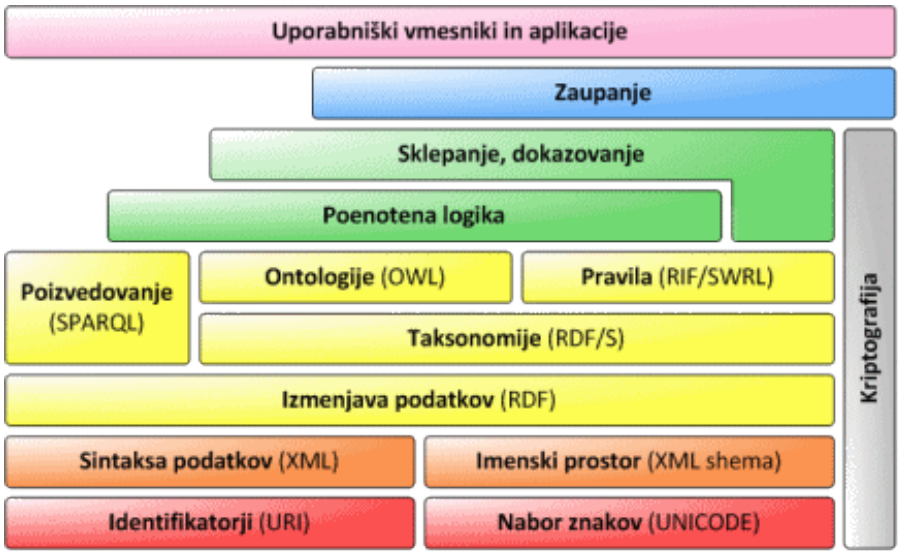
Na temo uporabnosti takega koncepta se je pojavilo kar precejšnje število člankov. V članku "The Semantic Web in Action" [6] so se avtorji razpisali o širši uporabnosti semantičnega spleta. Navedli so številne aplikativne rešitve iz industrije, biologije, znanosti o ljudeh pa tudi družbeni potencial semantičnega spleta v podjetjih, zdravstvenem sektorju in družbenih omrežjih.

Kljub temu se je zanimanje za uporabo semantičnega spleta in njegovih tehnologij v širši javnosti začelo prebujati šele v zadnjih letih. Predvsem po tem, ko so se zanj začela zanimati velika tehnološka podjetja, podprle pa so ga tudi vlade (na primer sofinanciranje nekaterih projektov s strani Evropskega razvojnega sklada). Nastalo je kar nekaj eksperimentalnih, vedno več pa je tudi komercialnih produktov [8]. Za splošno javnost so najbolj uporabne storitve v segmentu spletnih iskanj, med katerimi so najbolj aktivni in splošno uporabljeni Google Knowledge Graph, Facebook Graph Search, Yahoo SearchMonkey, Microsoft Bing Tiles .

Slika 4.2 prikazuje sklad tehnologij semantičnega spleta, sestavljen iz različnih standardov in orodij, ki so hierarhično razvrščeni. Na dnu vidimo najbolj osnovne ali tudi podporne standarde, kot so UNICODE za nabor znakov, URI za identifikacijo vira in XML za pravilno sintakso zapisa podatkov. Stopnjo višje so naprednejši standardi, med katerimi sta RDF za predstavitev in izmenjavo podatkov ter SPARQL za poizvedovanje po njih. Še stopnjo višje so logika, sklepanje in dokazovanje. Ta nivo je bolj razgiban, saj ne gre za poenoten standard, temveč za orodja, ki rešujejo te probleme. Orodij

je več vrst glede na namen. V okviru tega diplomskega dela smo uporabili semantično orodje Jena, s katerim se bomo v nadaljevanju še srečali. Na najvišjem nivoju so uporabniški vmesnik in aplikacije. Semantične aplikacije za delovanje ne potrebujejo prav vseh tehnologij, prikazanih na sliki, zadostuje že uporaba teh, ki smo jih navedli sami.

Za izvedbo naše prototipne aplikacije sta najpomembnejša standarda RDF in SPARQL, zato si ju bomo v nadaljevanju pogledali podrobneje.



Slika 4.2: Sklad tehnologij semantičnega spleta [10].

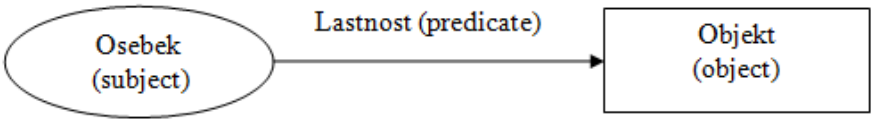
4.2.2 RDF

RDF (angl. *Resource Description Framework*) je osnovni standard za predstavitev podatkov na semantičnem spletu. Je jezik za izražanje podatkovnih modelov, ki se nanašajo na objekte in njihove odnose [10, 4]. Omogoča združevanje podatkov, tudi če se njihove osnovne sheme razlikujejo. Omogoča razvoj shem skozi čas, ne da bi bilo zaradi tega treba spreminjati obstoječe podatke. Prav tako RDF razširja povezovalno strukturo spleta, saj uporablja URI za poimenovanje razmerij med stvarmi kot povezave med dvema vozliščema. Taka struktura je pogosto poimenovana

tudi trojček (angl. *triple*). Uporaba tako preprostega modela omogoča, da se strukturirani podatki med seboj mešajo, izpostavljajo in delijo med različnimi aplikacijami. Tako povezana struktura tvori usmerjen, označen graf, kjer povezava predstavlja poimenovano vez med dvema viroma, ki sta predstavljena kot vozlišči grafa. Predstavitev z grafom je najpreprostejši miselni model predstavitve RDF in se pogosto uporablja.

Specifikacija RDF je sestavljena iz skupine priporočil W3C, ki so bila objavljena leta 2004 in so dosegljiva na naslovu `http://www.w3.org/standards/techs/rdf`.

RDF je osnovni format za zapis podatkov na semantičnem spletu. Predpisana sintaksa za serializacijo RDF je XML v obliki RDF/XML. Predstavlja podatkovni model v obliki grafa, sestavljen iz trojčka osebek, lastnost in objekt. Trojčke lahko predstavimo z usmerjenim grafom [9], kot je prikazano na sliki 4.3.



Slika 4.3: Zgradba trojčka RDF.

4.2.3 SPARQL

Za dostop do podatkov v zapisu RDF sta potrebna posebna zgradba in mehanizem dostopa, drugačna kot pri relacijskih podatkovnih bazah, ki se običajno uporabljajo pri standardnih spletnih aplikacijah. Obstaja več poizvedovalnih jezikov RDF (angl. *RDFQL – RDF query language*), vendar je SPARQL edini priporočen s strani organizacije W3C. Prav tako ga uporablja semantično ogrodje Jena, ki ga bomo podrobneje spoznali v poglavju 4.4.

SPARQL (angl. *SPARQL Protocol and RDF Query Language*) je protokol in jezik za povpraševanje po spletnih virih podatkov. Uporablja se za po-

izvedovanje tako po podatkih RDF kot po ontologijah RDFS in OWL[10, 4]. SPARQL je ena od ključnih tehnologij semantičnega spleta.

Jezik je standardiziran v okviru organizacije za standardizacijo W3C in je od marca 2013 v verziji SPARQL 1.1. Celotna specifikacija in dokumentacija sta dostopni na spletnem naslovu
`http://www.w3.org/TR/sparql11-overview/`.

Poizvedovalna točka SPARQL

Poizvedovalna točka SPARQL (angl. *SPARQL endpoint*) sprejema in vrača poizvedbe SPARQL prek izbranega protokola. Poenostavljeno je to URI, na katerega se posreduje zahteva SPARQL in s katerega dobimo odziv v obliki odgovora na to zahtevo. S prihodom standarda SPARQL 1.1 pojem poizvedovalne točke postane bolj splošen, saj je mogoče poleg poizvedbe poslati tudi posodobitve z uporabo SPARQL in HTTP [4].

Poizvedovalne točke delimo na dva tipa [9]:

- splošna (angl. *Generic*) poizvedovalna točka poizveduje po spletno dostopnih podatkih RDF; podatke sprejema iz poljubnega URI;
- specifična (angl. *Specific*) poizveduje po posebnih, običajno internih, podatkovnih množicah; uporaba je možna le iz vnaprej določenega nabora podatkov RDF.

Rezultat poizvedbe SPARQL lahko poizvedovalna točka vrne v različnih formatih, na primer v XML (osnovni način po specifikaciji RDF), JSON (objektna notacija JavaScript – posebej uporabno za spletne aplikacije), CSV/TSV (najbolj primerno za programe, ki delujejo z razpredelnicami), RDF (preko serializacije na RDF/XML, N-Triples, Turtle itd.) [16].

Zgradba poizvedbe SPARQL[9]:

- deklaracije predpon za določanje okrajšav URI;

- deklaracija podatkovnih množic navaja izvor grafov RDF, po katerih se bo poizvedovalo;
- tip povpraševanja določa, katere informacije se vračajo glede na poizvedovalni stavek, npr. SELECT, CONSTRUCT, ASK, DESCRIBE;
- vzorec povpraševanja določa vzorec, po katerem sistem išče po izbranih podatkovnih množicah (grafih);
- modifikatorji povpraševanja določajo različne omejitve za preurejanje rezultatov poizvedbe, npr. ORDER BY, OFFSET, LIMIT, DISTINCT, REDUCED.

Koda 4.1 ponazarja osnovno postavitev elementov poizvedbe SPARQL. Kot je mogoče opaziti, je SPARQL sintaktično podoben jeziku SQL (angl. *Structured Query Language*). Pravzaprav je hibrid med jezikom SQL in Prolog (logični programski jezik, povezan z umetno inteligenco in računalniškim jezikoslovjem). Zaradi specifičnega okolja, v katerem deluje, tj. v bazi podatkov s trojčki, ki predstavljajo graf, pa seveda obstajajo odstopanja. Primer uporabe si bomo ogledali pri predstavitvi prototipa v poglavju 5.

```
# deklaracija predpon
PREFIX foo: <http ://example.com/resources />
...
# deklaracija podatkovnih mnozic
FROM ...
# tip povprasevanja
SELECT ...
# vzorec povprasevanja
WHERE {
...
}
# modifikatorji povprasevanja
ORDER BY ...
```

Koda 4.1: Osnovna zgradba poizvedbe SPARQL.

4.3 Spletne tehnologije

Svetovni splet in njegova zgodovina sta na splošno že dovolj poznana, zato ju v tem poglavju ne bomo podrobneje opisovali. Si bomo pa ogledali novosti in sodobne tehnologije, ki smo jih uporabili pri razvoju spletne aplikacije.

4.3.1 HTML5

HTML (angl. *Hypertext Markup Language*) je osnovni standard za oblikovanje in predstavitev vsebin na svetovnem spletu. HTML5 predstavlja peto različico osnovnega standarda, zato ima enako primarno nalogo kot njegovi predhodniki.

Ker se zadnji standard, HTML 4.01, ni posodobil že od leta 2001, je leta 2003 organizacija Web HyperText Application Technology Working Group začela razvijati nov standard, takrat imenovan Web Application 1.0. Precej pozneje, predvsem zaradi interesov podpornikov različnih konceptov razvoja svetovnega spleta, se je skupini pridružila tudi krovna organizacija W3C, ki bdi nad svetovnim spletom. W3C je zato leta 2009 opustila razvoj svojega standarda XHTML 2.0 [1, 3].

Naj poudarimo, da HTML5 trenutno še vedno ni standard, saj je uradno še vedno v razvoju. Po načrtih W3C naj bi do dokončne standardizacije prišlo do konca leta 2014.

HTML5 je vpeljal veliko novih elementov in dodal nove lastnosti, ki spreminjajo lastnosti elementov. Novi elementi omogočajo lažjo, hitrejšo in učinkovitejšo gradnjo spletnih strani. S tehnologijami, ki stojijo za novimi elementi, lahko v veliki meri odpravimo tudi potrebo po uporabi zunanjih, predvsem večpredstavnostnih vtičnikov.

Najpomembnejše spremembe HTML5 v primerjavi s starejšo različico so[1]:

- novi strukturni elementi (npr. nav, section, header, footer, article, aside, figure),
- novi vrstni elementi (npr. mark, time, meter, progress),

- elementi za podporo dinamičnim stranem (npr. details, datagrid, menu, command),
- novi tipi obrazcev (npr. datalist, keygen, output),
- nove lastnosti za obrazce (npr. calendar, date, time, email, url, search),
- zvočni in vizualni elementi (npr. video, audio, canvas, svg),
- odstranjeni elementi (npr. applet, center, font, strike),
- dodatne lastnosti pri elementih,
- boljša podpora lokalnemu shranjevanju podatkov.

Elementi HTML5 za vizualizacijo

Pri standardu HTML5 nas še posebej zanima podpora grafičnim elementom. Ker gre za bistveni del naše aplikacije, na katerega se opira vizualizacija grafa, si jih bomo ogledali nekoliko podrobneje.

Canvas (sl. platno) je element, ki je del standarda HTML5 (imenovan HTML5 Canvas). Z njim je mogoče prikazati poljubne grafične elemente, kot bi jih v resničnem življenju risali na platno. Osnovni gradniki so črte, loki, krogi, različni vnaprej pripravljeni vzorci, besedila itd. Lahko pa nanj pripnemo tudi slike in video. Trenutno ga podpirajo že vsi večji in popularnejši brskalniki, npr. Google Chrome 3.0+, Mozilla Firefox 3.0+, Opera 10.0+, Microsoft Internet Explorer 9.0+ ter celo mobilna brskalnika Android 1.0+ in iOS (Mobile Safari) 1.0+. V osnovi omogoča izris tako dvorazsežne (t. i. Canvas 2D) kot tudi trirazsežne grafike. Vendar podporo trirazsežni grafiki načrtno prepušča WebGL API, ki je še v razvoju in ga ne podpira veliko brskalnikov. Za prikazovanje in manipulacijo dvorazsežne grafike potrebujemo Canvas JavaScript API ter znanje skriptnega programskega jezika JavaScript[1].

SVG (angl. *Scalable Vector Graphics*) je splošni standard za opis dvorazsežne vektorsko-rastrske grafike v formatu XML. Prišel je iz sveta namiznih aplikacij, vendar trenutno obstaja predlog, da bi ga W3C posvojila kot del svojih standardov. V HTML5 je dosegljiv prek istoimenskega elementa

SVG. To je seveda prva razlika v primerjavi z elementom canvas, ki velja le za spletni standard.

Z razliko od elementa canvas pa je slika, ustvarjena v tehnologiji SVG, dostopna tudi prek objektnega modela (angl. *DOM – Document Object Model*), kar različnim tehnologijam omogoča dostop do objektov znotraj slike. Opredeljuje tri tipe grafičnih objektov, to so vektorski elementi, rastrske slike ter besedila. Nad temi objekti pa je mogoče izvajati tudi različne slikovne transformacije. Na splošno je z njim mogoče na zaslonu prikazati enake podobe kot z elementom canvas. V kombinaciji s skriptnim jezikom JavaScript pa je mogoče narediti grafiko SVG tudi interaktivno [1].

Kljub mnogim prednostim pa ima tehnologija SVG tudi svoje omejitve in slabosti. Potrebuje namreč precej časa, da si ustvari scenski graf, na katerem poteka prikazovanje sestavljenih elementov. Uporabna je predvsem za prikaz velikih statičnih površin, medtem ko je element canvas bolje uporabiti pri izdelavi animacij in iger, saj omogoča hitrejši izris več grafičnih objektov hkrati. Prav tako je koda, potrebna za izris grafičnih elementov, nekoliko prijaznejša kot pri tehnologiji SVG, kjer pri zapletenejših oblikah hitro postane precej kompleksna.

4.3.2 JavaScript

JavaScript je objektno orientirani programski jezik, razvit v podjetju Netscape. Sam po sebi ni uporaben, ker ga ni mogoče uporabljati kot samostojen jezik. Navadno se uporablja za omogočanje dostopa do objektov znotraj okolja gostitelja. Najpogosteje se uporablja na strani odjemalca, kot del spletnega brskalnika. Uporablja pa se tudi v aplikacijah, ki niso del spleta. Aplikacije, znotraj katerih se izvaja, imajo svoje objektne modele za delo z objekti dokumentov, ki zagotavljajo dostop do gostiteljevega okolja [2]. Programerjem prek rokovanja z objekti omogoča izdelavo izboljšanih uporabniških vmesnikov in dinamični učinek sicer statičnih dokumentov. Samo jedro jezika JavaScript pa je v vseh aplikacijah večinoma enako.

4.3.3 jQuery

jQuery je ogrodje za lažje in hitrejše programiranje novih funkcionalnosti v programskem jeziku JavaScript. Dosegljivo je v obliki JavaScript knjižnice. Osnovne funkcionalnosti programskega jezika JavaScript naredi naprednejše, vendar hkrati olajša njihovo uporabo. Omogoča na primer naprednejše izbiranje, spreminjanje in upravljanje z elementi HTML, ustvarjanje animacij, boljše rokovanje z dogodki in poenotene funkcije za delo z objekti, vezanimi na brskalnik (funkcije se med različnimi brskalniki lahko razlikujejo), ter še mnoge druge izboljšave[2]. Ogrodje je razvila spletna skupnost razvijalcev za pomoč spletnim oblikovalcem in razvijalcem pri pisanju kode v JavaScriptu. Njihov moto je "Napiši manj (kode), naredi več (funkcionalnosti)"(angl. *Write Less, Do More*).

4.3.4 CSS

CSS (angl. *Cascading Style Sheets*) so prekrivni slogi, predstavljeni skozi slogovni jezik za opis oziroma slogovno oblikovanje označb strukturiranih dokumentov, običajno za HTML, XHTML, SVG, XML itd. Primarno je bil zasnovan z namenom ločitve vsebine dokumenta od njegove grafične predstavitve [3]. Takšna ločitev pripomore k boljšemu dostopu do vsebine, večji prilagodljivosti in nadzoru pri vizualni predstavitvi besedil oziroma na splošno elementov v dokumentu. Posledica takega pristopa je, da si lahko več dokumentov deli isto oziroma podobno oblikovanje, kar zmanjša kompleksnost in v veliki meri odpravlja podvajanja vsebine. Sami smo ga uporabili za oblikovanje dokumentov HTML5.

CSS3 je zadnja verzija specifikacije CSS. Ni še standardizirana, ker je standard še v razvoju. Kljub temu precej njenih funkcionalnosti mnogi brskalniki že podpirajo. CSS3 vključuje vse funkcionalnosti iz prejšnjih verzij in dodaja nove, ki razvijalcem olajšajo številne probleme, npr. podporo dodatnim izbirnikom (angl. *selectors*), senčenju, obrobljanju, animacijam, prosojnosti ter še mnogo več [3].

4.3.5 Ajax

Ajax je okrajšava za asinhroni JavaScript in XML (angl. *Asynchronous JavaScript and XML*). Besedo je skoval Jesse James Garrett [15], vendar se je tehnologija, na kateri temelji, razvijala in uporabljala že veliko prej. Pri tem gre pravzaprav za več medsebojno povezanih spletnih razvojnih tehnologij, uporabljenih za ustvarjanje interaktivnih spletnih aplikacij na strani odjemalca. Z uporabo pristopa Ajax lahko spletna aplikacija pošlje ali pa pridobi podatke s strežnika preko asinhronskih zahtev HTTP, ne da bi blokirala obnašanje obstoječe strani. Ni sicer obvezno, da je zahteva asinhronska, lahko je tudi sinhronska, vendar je obnašanje aplikacije temu primerno drugačno. Podatke je mogoče pridobiti z uporabo objekta XMLHttpRequest ter skriptnega jezika JavaScript [2]. Za izgradnjo aplikacije Ajax ni potrebna uporaba vseh omenjenih tehnologij. S to tehniko izdelovanja spletnih strani se med odjemalcem in strežnikom izmenja veliko manj podatkov. Zmanjša se tudi obremenitev spletnega strežnika in, najpomembnejše, uporabniški vmesnik se veliko hitreje odziva na vnose uporabnika.

Pristop Ajax se v osnovi nanaša na tehnologije [15]:

- HTML (oziroma XHTML) za opis strukture tekstovnih informacij v dokumentih,
- CSS za opis videza strani,
- DOM za dinamični prikaz in interakcijo s podatki,
- objekt XMLHttpRequest za asinhrono izmenjavo podatkov,
- XML (lahko tudi JSON ali golo besedilo) za prenos podatkov in
- skriptni jezik JavaScript za povezavo vseh prej naštetih tehnologij.

4.3.6 Arbor.js

Arbor.js je knjižnica za vizualizacijo in manipulacijo grafa znotraj grafičnih elementov dokumenta HTML5. Napisana je v jeziku JavaScript – na osnovi

JavaScript niti (t. i. *Web Workers*) in knjižnice jQuery. Knjižnica zagotavlja le abstrakcijo organizacije grafa in algoritem za simulacijo n-teles v grafu [13]. Z drugimi besedami, poskrbi za to, da so povezave in vozlišča čim bolj enakomerno razporejena po risalni plošči (slika 4.4b), pri čemer uporablja celo vrsto metod in pristopov iz fizike. Grafična podoba je prepuščena željam in potrebam programerja.

Glavne prednosti:

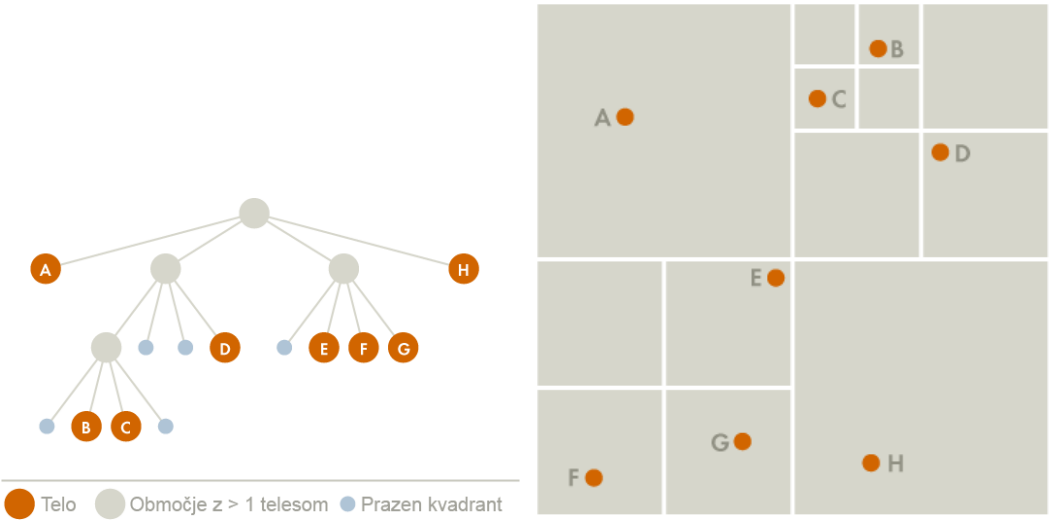
- omogoča uporabo treh različnih vrst gradnikov za prikaz (prek elementa SVG, elementa canvas, pozicijskih elementov HTML),
- risanje grafičnih elementov in lovljenje dogodkov sta povsem poljubna,
- relativno hitro delovanje tudi večjih grafov – z veliko vozlišči in povezavami,
- kratka in jasna dokumentacija.

Glavne slabosti:

- programer mora sam napisati precej kode za prikaz in detekcijo vseh grafičnih elementov,
- omejenost (oz. poudarek) knjižnice s prikazom grafičnih elementov le v obliki grafa.

Simulacija n-teles v grafu

Razporeditev teles grafa po risalni plošči temelji na Barnes-Hutovem algoritmu [13]. Vendar je končni položaj elementov lahko povsem drugačen od začetnega, saj na simulacijo vplivajo različni parametri. Parametri, ki lahko vplivajo na položaj, so "teža" vozlišč, "napetosti" povezav (učinek elastike), odbojnost vozlišč, učinek "gravitacije" (privlačnost vozlišč z nižjih nivojev tistim na višjem – npr. otrok starševskemu vozlišču) ter ne nazadnje uporabnikova interakcija z elementi (glede na uporabnikove želje po spremembi položajev elementov).



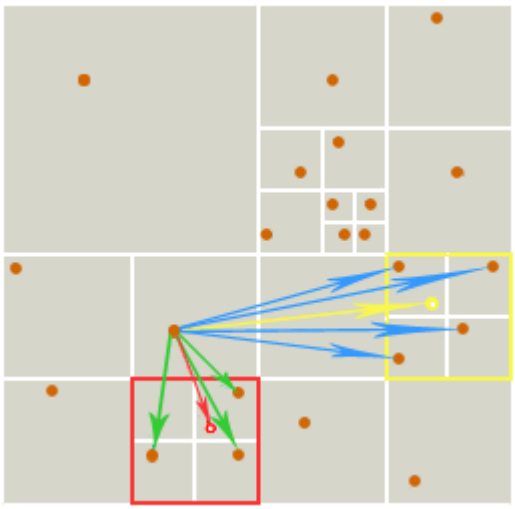
(a) vozlišča v navideznem drevesu (b) vozlišča na platnu

Slika 4.4: Začetna postavitev vozlišč z Barnes-Hutovim algoritmom v Arbor.js [13].

Barnes-Hutov algoritem za izračun sil uporablja hierarhično strukturo podatkov, imenovano štiriško drevo (angl. *quadtree*). Ta približno izračuna sile med n-tim številom teles v sistemu, kjer telesa vplivajo druga na drugo. Barnes-Hutov algoritem hirarhično razdeli prostornino okoli n-teles v zaporedje manjših celic [14]. V našem primeru jim lahko rečemo tudi kvadranti, saj simulacija deluje le v dveh razsežnostih. Vsak kvadrant tvori notranje vozlišče štiriškega drevesa in povzema informacije o telesih, ki jih vsebuje; zlasti njihovo skupno maso in središče težnosti. T. i. listi drevesa so posamezna telesa.

Osnovno postavitev vozlišč osmih teles pri uporabi algoritma prikazuje slika 4.4. Na levi strani so prikazana telesa, razporejena v drevo. Ta del velja le kot prispodoba. Prava upodobitev, vidna na zaslonu brskalnika, je na desni strani – na platnu. Vozlišča so le začasno transformirana v drevo. Tam jih imenujemo telesa. Po končani simulaciji in tranformaciji na platno jih ponovno imenujemo vozlišča.

Pri simulaciji n-tega števila teles moramo v splošnem upoštevati O(n^2) interakcij, natančen izračun je torej kvadrat števila teles. Hierarhija, ki jo uvaja algoritem, zmanjša čas za izračun sil med n-tim številom teles na O(n*log n). To velja, ker za dovolj oddaljena telesa zadošča le izračun ene posplošene sile s kvadrantom namesto izračuna sil z vsakim telesom znotraj kvadranta [14].



Slika 4.5: Vpliv odaljenosti med telesi na izračun sil pri Barnes-Hutovem algoritmu.

To lahko prenesemo na primer na sliki 4.5. Opazovano telo je vozlišče, iz katerega izvirajo puščice. S puščicami so označene sile, ki delujejo med telesi. Predpostavimo, da so telesa v rumenem kvadrantu dovolj oddaljena od opazovanega telesa. V tem primeru velja, da namesto izračuna vseh modro označenih sil z opazovanim telesom zadošča le izračun rumene sile, ki predstavlja silo posplošenega telesa ali tudi center težnosti tega kvadranta (rumeno vozlišče). Ob ponovni predpostavki, da telesa znotraj rdečega kvadranta niso dovolj oddaljena od opazovanega telesa, pa velja, da je treba izračunati vse sile, označene z zelenimi puščicami.

4.4 Semantično ogrodje Jena

Jena je ogrodje za razvoj aplikacij semantičnega spleta na osnovi programskega jezika Java. Razvijalcem zagotavlja obširne Java knjižnice za pomoč pri razvoju kode, ki manipulira s tehnologijami RDF, RDFS, OWL, SPARQL in drugimi, in sicer glede na priporočila organizacije W3C [16]. Uporaba jezika Java omogoča razvoj, neodvisen od platforme. Ponuja tudi različne pristope shranjevanja podatkov RDF.

Ogrodje so razvili v podjetju Hewlett Packard, vendar so ga kot odprto kodo prepustili fundaciji Apache (Apache Software Foundation) in je sedaj zaščiteno pod licenco Apache License 2.0, ki dovoljuje distribucijo in spreminjanje programske kode. Podrobna dokumentacija in koda sta dostopni na uradni spletni strani projekta, na naslovu `http://jena.apache.org` [16].

Ogrodje Jena na splošno vključuje [16]:

- API za branje, obdelavo in pisanje podatkov RDF v formatih XML, N-triples in Turtle;
- API za delo z ontologijami OWL in RDFS;
- mehanizem sklepanja, ki odloča na osnovi ontologij OWL in RDFS;
- podatkovne shrambe na osnovi trojčkov RDF, ki omogočajo učinkovito hrambo velikega števila podatkov;
- stroj za izvajanje poizvedb ARQ, ki je skladen z najnovejšimi specifikacijami SPARQL;
- strežnik, ki aplikacijam omogoča dostop do podatkov RDF prek različnih protokolov in formatov.

4.4.1 "Triplestore" podatkovna baza – Jena TDB

Shrambe podatkov v strukturi RDF lahko glede na arhitekturo razdelimo na izvirne shrambe RDF, hibridne shrambe RDF (s podporo relacijskim podatkovnim bazam) in ovojnice RDF [11]. TDB je komponenta semantičnega ogrodja Jena, namenjena shranjevanju in poizvedovanju po podatkih RDF.

Uvršča se med t. i. izvirne shrambe podatkov. Podatki so v njej lahko shranjeni v glavnem pomnilniku ali na disku. Vsaka množica podatkov je na disku shranjena v svoji datoteki [16].

Za izvirne shrambe RDF v splošnem velja, da so optimizirane za obdelavo podatkov RDF in delujejo neodvisno od drugih sistemov za upravljanje s podatkovnimi bazami. Nastale so iz potrebe po hrambi in strežbi velikih količin podatkov veliki množici uporabnikov. Bolj kot konsistentnost podatkov sta v njih pomembni skalabilnost in visoka razpoložljivost. Izvirne shrambe RDF omogočajo prilagodljivo podatkovno shemo, zato so primerne za aplikacije s pogostimi spremembami v podatkovnem modelu. V takih aplikacijah bi nas toga shema relacijskih podatkovnih baz ovirala. Izvirne shrambe RDF prinašajo številne prednosti, vendar niso univerzalna rešitev, saj je na račun omenjenih pridobitev treba sprejeti možno občasno nekonsistentnost [11].

Podatke RDF predstavljamo v obliki trojčkov. V splošni terminologiji pa so trojčki bolj znani po imenu *triples*, zato takim podatkovnim bazam rečemo tudi "triplestore" podatkovne baze. Uvrščamo jih med t. i. nerelacijske podatkovne baze.

Objavljeni so podatki o delujoči bazi, ki vsebuje 1,7 milijarde trojčkov pri zmogljivosti strežbe 12.000 trojčkov/s [17]. Uradnih omejitev pri velikosti shranjevanja v bazo Jena TDB ni. Zaradi tega se TDB lahko uporablja kot visoko zmogljiva shramba podatkov RDF.

Do baze Jena TDB je mogoče dostopati in jo upravljati s predvidenimi skriptami ukazne vrstice ali prek vmesnika Jena API, omogoča pa tudi uporabo transakcij, pri katerih zna nabor podatkov obvarovati pred popačenjem v primeru nepričakovanih prekinitev procesa ali sesutja sistema [16]. Uradna dokumentacija je na voljo na naslovu `http://jena.apache.org/documentation/tdb/`.

4.4.2 Spletni strežnik – Jena Fuseki

Še ena pomembna komponenta ogrodja Jena je strežnik SPARQL, ki se imenuje Fuseki. Fuseki streže podatke v zapisu RDF prek protokola HTTP. Nudi v slogu REST (angl. *Representational State Transfer*) podoben SPARQL HTTP Update, SPARQL Query in SPARQL Update, ki uporabljajo SPARQL prek protokola HTTP (angl. *SOH – SPARQL over HTTP*) [16].

SOH je nabor skript ukazne vrstice za delo s SPARQL. Skripte so neodvisne od strežnika in delujejo s katerim koli sistemom, ki je združljiv s standardom SPARQL 1.1 ter omogoča dostop prek protokola HTTP. SOH je napisan v programskem jeziku Ruby [16].

Kljub tem dodatkom pa se Fuseki v osnovi obnaša kot običajen spletni stežnik. Podrobna dokumentacija je dostopna na naslovu `http://jena.apache.org/documentation/serving_data/`.

Poglavje 5

Implementacija prototipa VisualRDF

5.1 Načrtovanje in organizacija

Funkcionalne zahteve smo predstavili že v poglavju 3. Za tem pa je bilo seveda treba določiti okolje za razvoj, katere tehnologije se bodo pri tem uporabile ter kako bodo organizirane v okviru končne rešitve.

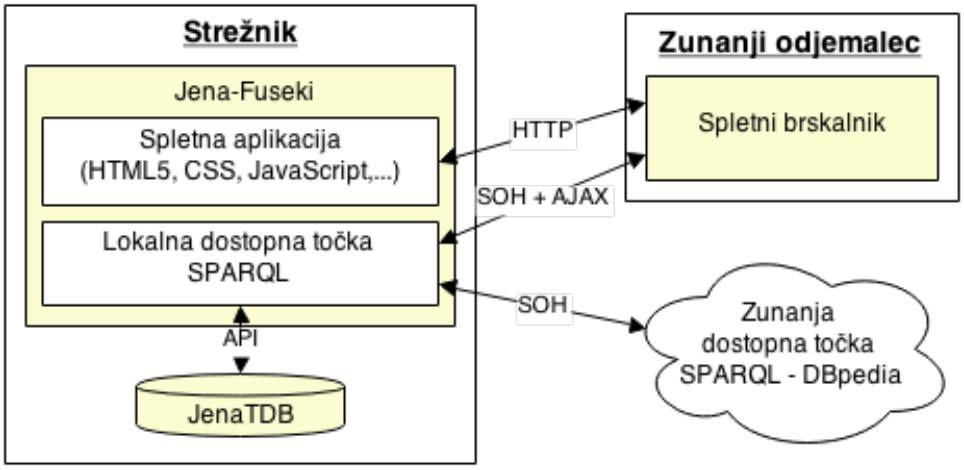
Prototip aplikacije VisualRDF, razvit v okviru diplomskega dela, uporablja različne tehnologije. Ključni sta tehnologiji HTML5 ter skriptni programski jezik JavaScript. Za hitrejši razvoj in boljšo združljivost JavaScript kode z različnimi brskalniki smo uporabili knjižnico jQuery. Poleg te smo uporabili še eno zunanjo knjižnico, in sicer Arbor.js. Z njo smo si pomagali pri prikazu grafov znotraj spletne aplikacije. Vse naštete tehnologije smo nekoliko podrobneje obravnavali že v poglavju 4.3.

Aplikacija je pretežno zasnovana tako, da deluje neodvisno od vrste in izvora baze, od vrste podatkov ali njihove sheme. Deluje lahko v odprtem okolju kot javna aplikacija z javno dostopno bazo ali v zaprtem okolju z javno ali zasebno dostopno bazo. Za delovanje same spletne aplikacije je potreben le osnovni spletni strežnik in navedba poti do podatkovnega strežnika s poizvedovalno dostopno točko SPARQL. Dostopna točka na drugi strani

mora biti sposobna izvajati standardne poizvedbe SPARQL ter serializacijo končnih rezultatov iz osnovnega zapisa RDF v zapis JSON.

Za spletni strežnik smo uporabili kar strežniško komponento semantičnega ogrodja Jena, Fuseki. Za vir podatkov smo za namene diplomskega dela uporabili tako lokalno kot zunanjo bazo podatkov. Lokalna baza podatkov je shranjena v podatkovni bazi TDB, ki je prav tako del semantičnega ogrodja Jena. Zunanja baza pa je DBpedia in je dostopna prek zunanje poizvedovalne točke SPARQL. Podrobnosti o namenu in delovanju semantičnega ogrodja Jena smo opisali v poglavju 4.4.

Kako so vse naštete komponente in tehnologije med seboj povezane, prikazuje slika 5.1. Pri tem poudarimo, da je zunanjih odjemalcev lahko več – na sliki je zaradi poenostavljenosti primera prikazan le en.



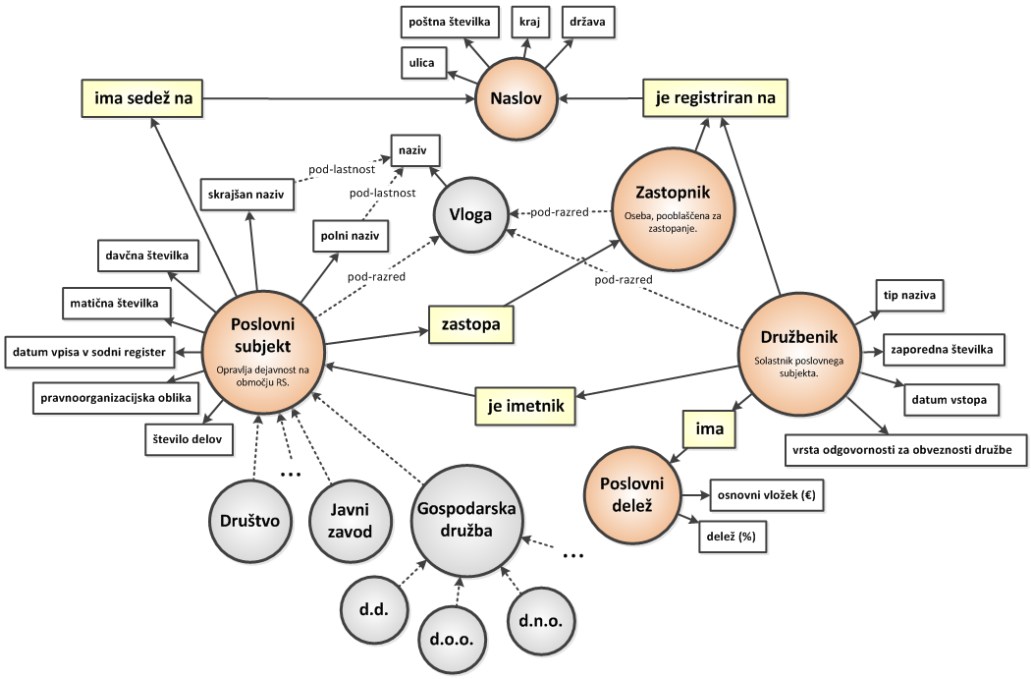
Slika 5.1: Organizacija komponent in tehnologij.

5.2 Izvor in pregled podatkov

Kot je bilo omenjeno že v uvodu, smo podatke črpali iz baze, ki je nastala na podlagi luščenja podatkov iz Poslovnega registra Slovenije, ki ga vzdržuje AJPES (Agencija Republike Slovenije za javnopravne evidence in storitve). Te podatke smo za razvojno-testne namene pridobili v Laboratoriju za po-

datkovne tehnologije na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Podatki so shranjeni v bazi Jena TDB v obliki zapisov RDF. Skupaj je zapisov RDF 4.697.606 (pridobljeno s poizvedbo SPARQL: *SELECT COUNT(*) WHERE {?s ?p ?o}*), kar predstavlja okoli 1 GB prostora na strežniku.

Baza vsebuje veliko količino podatkov o poslovnih subjektih, njihovih naslovih, letnih poročilih, zastopnikih, družbenikih, poslovnih deležih in o drugem. Razen osnovnega povzetka se v podrobne podatke v tem delu ne bomo spuščali. Predvsem zato, ker to ni potrebno, saj je aplikacija narejena tako, da deluje s poljubnimi podatki ne glede na njihovo shemo ali izvor. Primer dela sheme ontologije teh podatkov je razviden na sliki 5.2.



Slika 5.2: Primer dela sheme ontologije podatkov Poslovnega registra Slovenije (AJPES).

Poleg osnovne baze smo za obogatitev obstoječih podatkov uporabili še zunanjo bazo podatkov. Ker osnovna baza AJPES vsebuje podatke o lju-

deh, podjetjih, ustanovah pa tudi krajih, se nam je zdel najbolj primeren način obogatitve teh podatkov s podatki iz največje prostodostopne spletne enciklopedije Wikipedije.

Wikipedija je brezplačna spletna enciklopedija, ki jo podpira neprofitna organizacija Wikimedia Foundation. Dostopna je v kar 287 jezikih in obsega 30 milijonov člankov, od katerih jih je 4,3 milijona na njeni angleški različici. Nastaja v sodelovanju s prostovoljci z vsega sveta, ki nenehno dodajajo in urejajo vsebino s skoraj vseh področij. Po zadnjih podatkih naj bi imela 365 milijonov bralcev z vsega sveta, kar jo uvršča na prvo mesto med največjimi in najpopularnejšimi priročniki na internetu [18].

Zaradi precej odprtega pristopa k urejanju vsebin je pri uporabi Wikipedije seveda treba biti previden zaradi morebitnih zlorab. Ne glede na to je treba priznati njeno odprtost in dostopnost obsežne količine informacij in znanj. Kljub temu pa je njen način podajanja informacij za nas precej pomanjkljiv. Podatki so namreč zapisani nestrukturirano, zato so berljivi le ljudem, ne pa tudi računalnikom. Ta problem uspešno rešuje projekt DBpedia.

DBpedia je projekt, katerega cilj je izluščiti vsebino Wikipedije, jo zapisati v strukturirani obliki in jo ponovno deliti na svetovnem spletu. Z uporabo tehnologij semantičnega spleta omogoča napredno poizvedovanje po vsebini Wikipedije. Poleg tega omogoča povezovanje drugih podatkovnih skladišč na spletu.

Angleška različica baze znanja DBpedie trenutno opisuje okoli 4,3 milijona stvari, od katerih je 3,2 milijona razvrščenih znotraj trenutne ontologije. Vključuje 832.000 ljudi, 639.000 krajev (od tega 427.000 naseljenih krajev), 209.000 organizacij (vključno z 49.000 podjetji in 45.000 izobraževalnimi ustanovami) ter številne podatke z drugih področij. Podatki in uporaba so prosto dostopni pod pogoji licence Creative Commons Attribution – ShareAlike 3,0 in GNU Free Documentation License [18].

5.3 Dostop do podatkov

V prejšnjem poglavju 5.2 smo se seznanili z dejstvom, da aplikacija uporablja dva izvora podatkov. Prvi so zapisi AJPES v lokalni bazi podatkov, drugi pa zapisi z Wikipedije, shranjeni v zunanji bazi DBpedia. Zaradi tega ju bomo obravnavali ločeno.

Dostop do lokalnih podatkov

V poglavju 4.4 smo spoznali semantično ogrodje Jena, ki med drugim vsebuje tudi strežnik Fuseki in podatkovno bazo TDB za hrambo podatkov. Pred uporabo moramo seveda zagnati strežnik ter ga povezati s podatkovno bazo, ki hrani vsebino v datotekah. To storimo s konzolnim ukazom, prikazanim v kodi 5.1.

```
java -classpath "fuseki-server.jar" -Xmx1200m
org.apache.jena.fuseki.FusekiCmd
 --loc= \Data\AJPES /dataset
```

Koda 5.1: Zagon strežnika Jena Fuseki in nastavitev poti do lokalne baze podatkov.

Po zagonu strežnika imamo na voljo dve možnosti: ali poženemo poizvedbo SPARQL prek konzolnega načina ali prek SOH. Za dostop do lokalnih podatkov s strani odjemalca (natančneje iz spletne aplikacije v brskalniku) je treba uporabiti dostop prek protokola SOH. Pri uporabi strežnika Fuseki deluje SOH v okviru servisnih dostopnih točk (angl. *Service endpoints*). Ker znotraj aplikacije ne dodajamo ali spreminjamo nobenih podatkov, temveč po njih le povprašujemo, zadostuje le uporaba dostopne točke za poizvedovanje. Ta je v okviru strežnika dosegljiva prek naslova *http://localhost:3030/dataset/query*. Domena "localhost" je rezerviran IP (v4) naslov 127.0.0.1 in v resnici predstavlja le lokalni dostop na isti napravi. Če bi želeli podatke deliti s svetom, bi morali dostopno točko postaviti v okviru javnega IP naslova. Koda 5.2 ponazarja primer osnovne poizvedbe SPARQL, ki jo lahko posredujemo na omenjeno lokalno dostopno točko.

```
SELECT * {
<http://www.lavbic.net/owl/
AJPES.owl#SI54162513_UNIVERZA_V_LJUBLJANI>  ?prop ?s
}
```

Koda 5.2: Primer poizvedbe SPARQL za dostop do lokalnih podatkov v okviru lokalne poizvedovalne točke SPARQL.

Dostop do zunanjih podatkov

Za dostop do drugih baz podatkov na semantičnem spletu moramo v splošnem le spremeniti URI naslov poizvedovalne točke. Do podatkov iz DBpedie tako lahko dostopamo prek poizvedovalne točke na naslovu *http://dbpedia.org/sparql*. Do nje lahko dostopamo na dva načina: neposredno prek njene dostopne točke ali posredno prek druge dostopne točke. Prvi način smo spoznali ravno prej (v okviru dostopa do lokalne dostopne točke), drugi pa deluje prek posebnega ukaza SERVICE, ki je del standarda SPARQL 1.1. Ukaz SERVICE omogoča, da prek poljubne dostopne točke poizvedujemo po podatkih iz drugih poljubnih dostopnih točk. Ta način ima še eno prednost, znotraj iste poizvedbe namreč omogoča dostop do več virov podatkov (v našem primeru lokalnih in zunanjih) – kot da bi iskali znotraj ene skupne baze podatkov.

V naši spletni aplikaciji za dostop do zunanjih podatkov iz DBpedie uporabljamo posredni dostop prek lokalne dostopne točke, ki gostuje na lokalnem strežniku Jena Fuseki. Primer dostopa do podatkov iz DBpedie prek lokalne dostopne točke je prikazan v kodi 5.3.

```
SELECT ?prop ?s
WHERE {
  SERVICE <http://dbpedia.org/sparql> {
 <http://dbpedia.org/resource/University_of_Ljubljana> ?prop
    ?s
  FILTER( lang(?s) = "en")
  }
}
```

Koda 5.3: Primer poizvedbe SPARQL za dostop do podatkov iz zunanje dostopne točke DBpedia prek lokalne dostopne točke.

5.4 Komunikacija odjemalca s strežnikom

Naša rešitev deluje po arhitekturnem slogu strežnik–odjemalec. Komponenta Fuseki semantičnega ogrodja Jena predstavlja strežnik. Za odjemalca pa v tem primeru uporabljamo spletni brskalnik. Aplikacija in podatki gostujejo na strežniku, zato jih je na zahtevo treba prenesti do brskalnika. Aplikacijo je treba prenesti le enkrat, in to le v smeri proti brskalniku, podatki pa se prenašajo v obe smeri, in to s poljubnimi ponovitvami (odvisno od uporabnikove uporabe aplikacije). Zato se bomo v tem poglavju osredotočili le na prenose podatkov.

```
var qText = [ ' SELECT *   ',
 '  { <http://www.lavbic.net/owl/
 AJPES.owl#SI54162513_UNIVERZA_V_LJUBLJANI> ?prop ?s ',
'    } '].join('\n');

//send request
$.ajax({
        url : endpointTab[selectedEndpointTab].url,
        data : "query=" + encodeMyURI(
        urlQueryGeneralPrefix + qText) + urlQueryType,
        dataType : "json",
        type : "GET",
        success : function(data) {
                        ....
        }, error : function(XMLHttpRequest, textStatus,
        errorThrown) {
                        ...
    }
});
```

Koda 5.4: Primer zahtevka Ajax, ki poizvedbe pošilja na dostopno točko SPARQL, pridobi pa podatke v zapisu JSON.

Od aplikacije na brskalniku je odvisen potek celotne komunikacije. Aplikacija je namreč tista, ki zahteva določene podatke, strežnik pa le sledi njenim zahtevam in se nanje odzove v obliki serije podatkov. S strani odjemalca za komunikacijo s poizvedovalno dostopno točko, gostovano na strežniku, uporabljamo pristop Ajax. V brskalniku za oddajanje zahtevkov strežniku in sprejemanje povratnih odgovorov skrbi ukaz *ajax* knjižnice jQuery. Preprost primer delovanja zahtevka ter poizvedbe SPARQL po primerku razreda Uni-

verze v Ljubljani je prikazan v kodi 5.4.

Vrsta ter format podatkov, ki se izmenjujejo med strežnikom in odjemalcem, sta odvisna od smeri komunikacije. Za boljšo predstavo ju bomo predstavili na primerih.

Podatki v smeri od odjemalca proti strežniku

Spletna aplikacija v smeri proti strežniku prenaša poizvedbe SPARQL prek protokola SOH (SPARQL preko HTTP). Za prenos se uporablja HTTP metoda GET. Na podlagi zahtevka v kodi 5.4 se proti strežniku pošlje HTTP, razviden v kodi 5.5.

```
GET /ds/query?query=SELECT%20%2A%0A%7B%3Chttp%3A%2F%2F
www%2Elavbic%2Enet%2Fowl%2FAJPES%2Eowl%23SI54162513%5F
UNIVERZA%5FV%5FLJUBLJANI%3E%20%3Fprop%20%3Fs%0A%7D%0A
&output=json HTTP/1.1
Host: localhost:3030
Connection: keep-alive
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Iron/30.0.1650.0
Chrome/30.0.1650.0 Safari/537.36
DNT: 1
Referer: http://localhost:3030/visualrdf/indexAjpes.html
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Cookie: JSESSIONID=1n7x24h6w6ljg13irljrj53jur
```

Koda 5.5: Primer glave poslanega zahtevka HTTP.

Podatki v smeri od strežnika proti odjemalcu

Strežnik pošlje proti odjemalcu rezultate zahtevane poizvedbe. Poslani podatki so strukturirani v zapisu JSON. Pretvorbo oziroma serializacijo rezultatov poizvedbe SPARQL iz zapisa RDF v zapis JSON opravi že sama dostopna točka SPARQL.

Primer glave uspešno prejetega zahtevka HTTP je predstavljen v kodi 5.6. Podatke istega prejetega zahtevka pa dobimo prek spremenljivke *data*

metode *success* prej navedenega zahtevka Ajax (iz kode 5.4). Omenjeno metodo aktivira brskalnik ob uspešnem prenosu. Znotraj nje pridobimo prejete podatke, ki so razvidni iz kode 5.7.

```
HTTP/1.1 200 OK
Fuseki-Request-ID: 277
Access-Control-Allow-Origin: *
Server: Fuseki (0.2.7-SNAPSHOT)
Vary: Accept,Accept-Encoding,Accept-Charset
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/sparql-results+json; charset=utf-8
Content-Encoding: gzip
Transfer-Encoding: chunked
```

Koda 5.6: Odgovor HTTP – "Response Header".

```
{
  "head": {
    "vars": [ "d" , "f" ]
  } ,
  "results": {
    "bindings": [
      {
        "d": { "type": "uri" ,
        "value": "http://www.w3.org/1999/02/
        22-rdf-syntax-ns#type" } ,
        "f": { "type": "uri" ,
        "value": "http://www.lavbic.net/owl/
        AJPES.owl#JavniZavod"}
      } ,
      {
        "d": { "type": "uri" , "value":
        "http://www.lavbic.net/owl/
        AJPES.owl#skrajsanNaziv"},
        "f": { "datatype": "http://www.w3.org/2001/
        XMLSchema#string" , "type": "typed-literal" ,
        "value": "UL" }
      } ,
      {
        "d": { "type": "uri" ,
        "value": "http://www.w3.org/2000/01/
        rdf-schema#seeAlso" } ,
        "f": { "type": "literal" , "xml:lang": "sl" ,
        "value": "http://www.ajpes.si/prs/
        podjetjeSRG.asp?s=1&e=125439" }
      } ,
      {
        "d": { "type": "uri" ,
        "value": "http://www.lavbic.net/owl/
```

```
    AJPES.owl#pravnoorganizacijskaOblika" } ,
    "f": { "datatype": "http://www.w3.org/2001/
    XMLSchema#string" , "type": "typed-literal" ,
    "value": "Javni zavod" }
  } ,
  {
    "d": { "type": "uri" ,
    "value": "http://www.lavbic.net/owl/
    AJPES.owl#poslovniSubjektImaImetnikaVDruzbeniku"},
    "f": { "type": "uri" ,
    "value": "http://www.lavbic.net/owl/
    AJPES.owl#D_572725_REPUBLIKA_SLOVENIJA" }
  } ,
  {
    "d": { "type": "uri" ,
    "value": "http://www.lavbic.net/owl/
    AJPES.owl#polniNaziv" } ,
    "f": { "datatype": "http://www.w3.org/2001/
    XMLSchema#string" , "type": "typed-literal" ,
    "value":   "UNIVERZA V LJUBLJANI" }
  } ,
  {
    "d": { "type": "uri" ,
    "value": "http://www.lavbic.net/owl/
    AJPES.owl#davcnaStevilka" } ,
    "f": { "datatype": "http://www.w3.org/2001/
    XMLSchema#string" , "type": "typed-literal" ,
    "value": "SI54162513" }
  } ,
  {
    "d": { "type": "uri" ,
    "value": "http://www.lavbic.net/owl/
    AJPES.owl#maticnaStevilka" } ,
    "f": { "datatype": "http://www.w3.org/2001/
    XMLSchema#string" , "type": "typed-literal" ,
    "value": "5085063000" }
  } ,
  {
    "d": { "type": "uri" ,
    "value": "http://www.lavbic.net/owl/
    AJPES.owl#datumVpisaVSodniRegister"},
    "f": { "datatype": "http://www.w3.org/2001/
    XMLSchema#dateTime" , "type": "typed-literal" ,
    "value": "1979-12-18T23:04:00Z" }
  } ,
  {
    "d": { "type": "uri" ,
    "value": "http://www.lavbic.net/owl/
    AJPES.owl#poslovniSubjektImaSedezNaNaslovu" },
    "f": { "type": "uri" ,
    "value": "http://www.lavbic.net/owl/
    AJPES.owl#1000_Ljubljana_Kongresni_trg_12" }
  } ,
```

```
      {
        "d": { "type": "uri" ,
        "value": "http://www.lavbic.net/owl/
        AJPES.owl#dbpediaInfo" } ,
        "f": { "type": "literal" ,
        "value": "http://dbpedia.org/resource/
        University_of_Ljubljana" }
      }
    ]
  }
}
```

Koda 5.7: Podatki iz prejetega zahtevka HTTP v zapisu JSON.

5.5 Grafična predstavitev podatkov

Podatki so v bazi v zapisu RDF. Zaradi lažje obravnave trojčke v aplikacijo dobimo v zapisu JSON. Kot smo že navedli, za serializacijo poskrbi dostopna točka SPARQL. Predstavitev podatkov smo razdelili na predstavljene v obliki grafa ter zapisane v obliki stolpcev oziroma alinej. V tem poglavju se bomo osredotočili na predstavitev podatkov v obliki grafa.

Za predstavitev podatkov v obliki grafa smo uporabili element HTML5 Canvas, ki služi kot platno, po katerem lahko rišemo grafične elemente. Za samo risanje po platnu pa je uporabljen skriptni jezik JavaScript ter metode iz Canvas API. Ker je podatkov v obliki grafa lahko veliko, smo za enakomerno razporeditev vozlišč in povezav (te predstavljajo trojčke RDF) po zaslonu uporabili JavaScript knjižnico Arbor.js.

Kot smo navedli v poglavju 4.3.6, Arbor.js zagotavlja le abstrakcijo organizacije grafa, sami pa moramo poskrbeti za grafično predstavitev. Iz podatkov, pridobljenih s strežnika, in z metodami knjižnice Arbor.js smo zgradili navidezni graf. Navidezni zato, ker smo prek drugih metod knjižnice Arbor.js ter metod Canvas API morali navidezni graf še izrisati na zaslon.

Omenjeni postopek smo želeli prikazati z zelo poenostavljeno predlogo uporabe knjižnice Arbor.js, ki je prikazana v kodi 5.8.

```
//  poenostavljena predloga za uporabo knjiznice arbor.js

(function($){

  var Renderer = function(canvas){
    var canvas = $(canvas).get(0)
    var ctx = canvas.getContext("2d");
    var particleSystem

    var that = {
      init:function(system){
        // arbor.js poklice funkcijo ob inicializaciji
           sistema
        // sem spadajo zacetne nastavitve parametrov sistema
         ...
        // inicializacija akcij nad elementi (vozlisca,
           povezave)
        that.initMouseHandling()
      },

      redraw:function(){
        //  arbor.js klice funkcijo neprestano oz. po potrebi
            (na to vpliva vec parametrov v sistemu)
        //  programer mora tu vstaviti kodo za izris
           elementov grafa, ki bodo vidni na zaslonu

        particleSystem.eachEdge(function(edge, pt1, pt2){
         // koda za izris povezav
         ...
        })

        particleSystem.eachNode(function(node, pt){
          // koda za izris vozlisc
          ...
        })
       ...
      },

      initMouseHandling:function(){
         // inicializacija akcij (klik z misko, vlecenje
            vozlisc ...) in odziva sistema na njih
        var dragged = null;
                ...
        var handler = {
          clicked:function(e){
                ...
          },
          dragged:function(e){
                ...
          },

          dropped:function(e){
            ...
```

```
          }
        }

      },

    }
    return that
  }

  $(document).ready(function(){
    // sledi koda za nastavitev sistema Arbor.js ter
       polnjenje slednjega s podatki

    // ustvari sistem z razlicnimi parametri (elasticnost
       povezav/trenjem vozlisc ...)
    var sys = arbor.ParticleSystem(1000, 600, 0.5)
    // parameter za lepso razporeditev vozlisc
    sys.parameters({gravity:true})
    //inicializacija  elementa canvas, na katerem bo izrisan
       graf
    sys.renderer = Renderer("#mojCanvas")

    // vnasanje podatkov v Arbor.js - vozlisca, povezave ter
       njihove lastnosti
    //vnos dveh vozlisc
    sys.addNode('vozlisce1', { mass:.25, color:454545})
    sys.addNode('vozlisce2', { /* lastnosti elementa(ime,
       barva,...) ...*/ })
    //vnos povezave med tema dvema vozliscema
    sys.addEdge('vozlisce1','vozlisce2'){ /* lastnosti */ })
    ...

  })

})(this.jQuery)
```

Koda 5.8: Koda poenostavljene predloge za uporabo knjižnice Arbor.js.

Za risanje po elementu canvas v dveh razsežnostih vtičniki niso potrebni. Treba je le vstaviti nov element v dokument HTML ter uporabiti metode Canvas API prek jezika JavaScript. Poenostavljen primer upodobitve trojčka RDF v obliki vozlišč in povezav (<vozlisce1, povezava, vozlisce2>) po elementu canvas prikazujeta slika 5.3 in njej pripadajoča koda 5.9.

```
<!DOCTYPE html>
<html>
<body>

<canvas id="mojCanvas" width="300" height="100" style="border
```

```
   :1px solid #d3d3d3;">
Your browser does not support the HTML5 canvas tag.</canvas>

<script>

// inicializacija in deklaracija splosnih spremenljivk
var pt1={},pt2={};
pt1.x=50; pt1.y=50;
pt2.x=250;pt2.y=50;

var extraPx =4;
var label1= "vozlisce1";
var label2= "vozlisce2";

// dostop do elementa v DOM
var c=document.getElementById("mojCanvas");

// inicializacija/deklaracija canvas 2D
var ctx=c.getContext("2d");
var w1 = ctx.measureText(label1).width + 20;
var w2 = ctx.measureText(label2).width + 20;


//////////////////
// povezava

// povezava med vozliscema
 ctx.beginPath()
  ctx.lineWidth =  1;
  ctx.strokeStyle = "#cccccc";
  ctx.fillStyle = null;

  ctx.moveTo(pt1.x, pt1.y)
  ctx.lineTo(pt2.x, pt2.y)
 ctx.stroke()

// pravokotnik na sredini povezave -- za napis na povezavi
ctx.beginPath()
 ctx.fillStyle = ctx.strokeStyle = "#cccccc";
 var wLink = ctx.measureText("povezava").width +4;
 ctx.rect((pt1.x+pt2.x)/2 -wLink /2 -extraPx , (pt1.y+pt2.y)
    /2 -extraPx -12/2,     wLink +2*extraPx,12+extraPx, 4, {
    fill:ctx.fillStyle});
 ctx.fill();

// napis znotraj pravokotnika
 ctx.font ='9px  Helvetica, Arial, sans-serif';
 ctx.textAlign = "center";
 ctx.fillStyle = "white"        ;
 ctx.fillText("povezava",(pt1.x+pt2.x)/2,(pt1.y+pt2.y)/2 );

ctx.stroke()
```

```
////////////////
// vozlisci

ctx.beginPath()

 // pravokotniki
 ctx.fillStyle = ctx.strokeStyle=  "#7F5BB2";
 ctx.rect(pt1.x - w1/2, pt1.y-10, w1,20, 15,{fill:ctx.
    fillStyle})
 ctx.rect(pt2.x - w2/2, pt2.y-10, w2,20, 15, {fill:ctx.
    fillStyle})
 ctx.fill();

 // napis znotraj vozlisc (labele)
 ctx.font = '12px  Helvetica, Arial, sans-serif';
 ctx.textAlign = "center"
 ctx.fillStyle = "white"

 ctx.fillText(label1, pt1.x, pt1.y+4)
 ctx.fillText(label2, pt2.x, pt2.y+4)

ctx.stroke()

</script>

</body>
</html>
```

Koda 5.9: Primer risanja vozlišč in povezav po elementu canvas v HTML5.

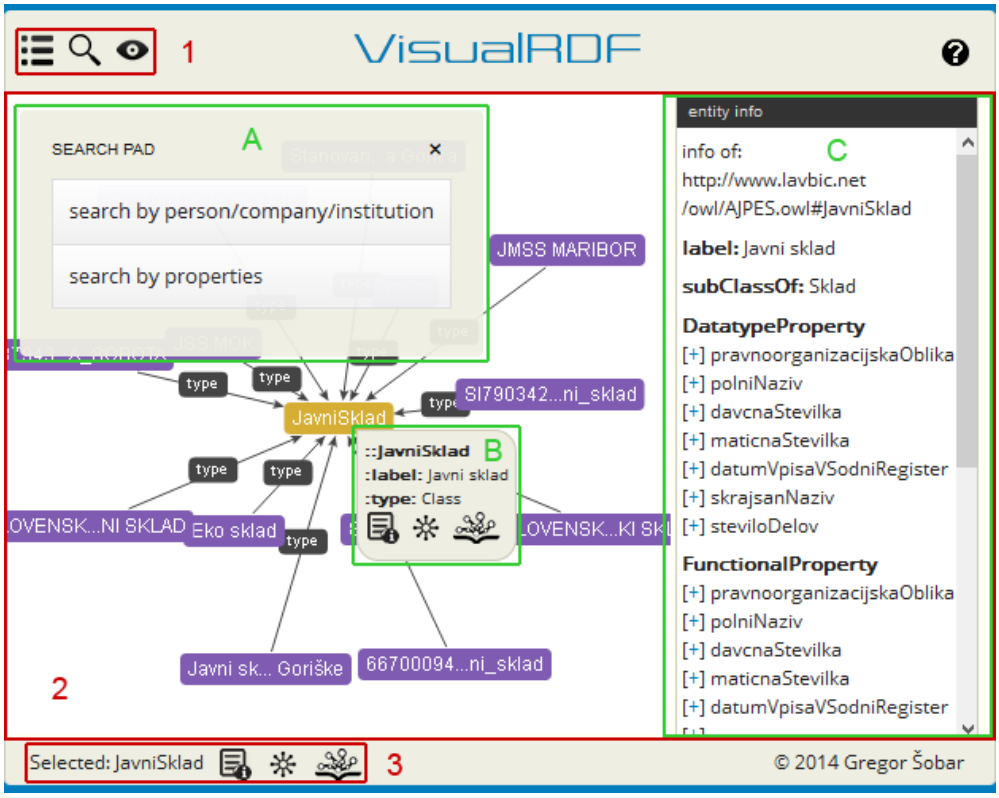
Slika 5.3: Rezultat primera poenostavljenega risanja vozlišč in povezav po elementu canvas v HTML5.

5.6 Grafični uporabniški vmesnik

Pomemben cilj pri gradnji uporabniškega vmesnika nam je predstavljala izdelava preglednega, enostavnega, a hkrati učinkovitega uporabniškega vme-

snika. Ker se zavedamo, da je uporabniški vmesnik pravzaprav uporabnikovo okno do naprednih funkcionalnosti, ki se izvajajo v ozadju, mora biti kolikor je mogoče preprost in intuitiven. Uporabili smo minimalistični pristop, kar pomeni, da so v nekem trenutku na zaslonu v ospredju le najpomembnejši podatki in grafični elementi.

Osnovni gradniki vmesnika

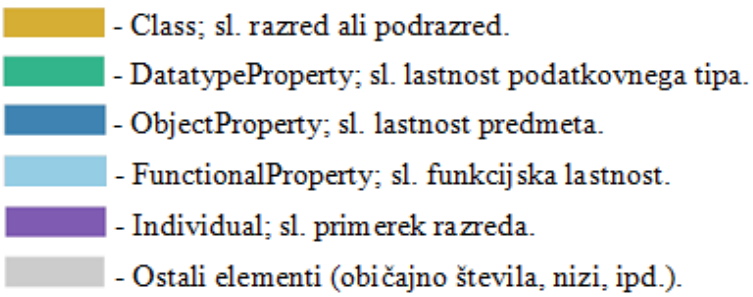


Slika 5.4: VisualRDF – strnjen pogled na osnovne gradnike uporabniškega vmesnika.

Grafični vmesnik je v osnovi razdeljen na tri glavne sekcije, označene z rdečo barvo, kot prikazuje slika 5.4. Okvir številka 1 označuje osnovni meni, številka 2 označuje prostor, namenjen prikazu grafa, številka 3 pa statusno vrstico.

Okvirji zelene barve na isti sliki prikazujejo opcijska okna. Ta se aktivirajo glede na uporabnikove trenutne akcije. S črko A je označeno pojavno okno, ki se aktivira na podlagi izbire gumbov v osnovnem meniju (meni je označen s številko 1). V nadaljevanju bomo ob primerih lahko videli, kaj vse uvrščamo v omenjeno skupino. Črka B označuje pojavno okno, ki se aktivira na podlagi uporabnikove izbire vozlišča. Okno ponudi na izbiro nekoliko več akcij, predstavi pa tudi nekaj osnovnih informacij o izbranem vozlišču. Pod črko C je posebno drsno okno, namenjeno prikazu podrobnejših informacij ali dodatnih funkcionalnosti.

Statusna vrstica (številka 3) pa tudi pojavno okno nad vozliščem (črka B) vsebujeta dodatne ikone, ki so namenjene nadaljnjim akcijam nad izbranim elementom. Element je lahko izbran prek iskalnega mehanizma ali pa je to kar vozlišče z grafa. Leva akcijska ikona (imenovana tudi "gumb za podrobnejše informacije") tako omogoča podroben izpis podatkov o izbranem elementu. Sredinska ikona (imenovana tudi "gumb za prikaz elementa v grafu") omogoča aktivacijo drsnega okna, v katerem so funkcionalnosti za grafični prikaz izbranega elementa in tudi njegove soseščine elementov v grafu. Desna ikona (imenovana tudi "gumb za izpis podrobnosti o elementu iz zunanjega vira") aktivira drsno okno, v katerem se prikažejo podrobnosti iz zunanjih virov, na primer iz DBpedie.

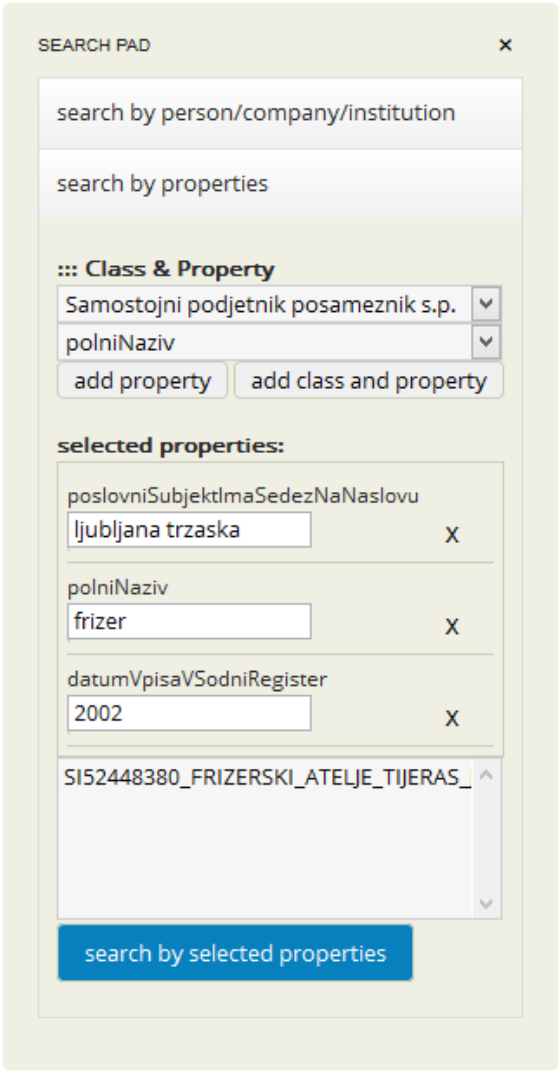


Slika 5.5: VisualRDF – legenda barv vozlišč glede na vrsto podatka.

Kot je mogoče opaziti na osrednji površini slike 5.4, so vozlišča različnih barv, in sicer glede na vrsto podatka, ki ga predstavljajo. Barvna lestvica

glede na vrsto podatka je predstavljena v legendi na sliki 5.5.

Menijska okna



Slika 5.6: VisualRDF – "SEARCH PAD" – menijsko okno za iskanje.

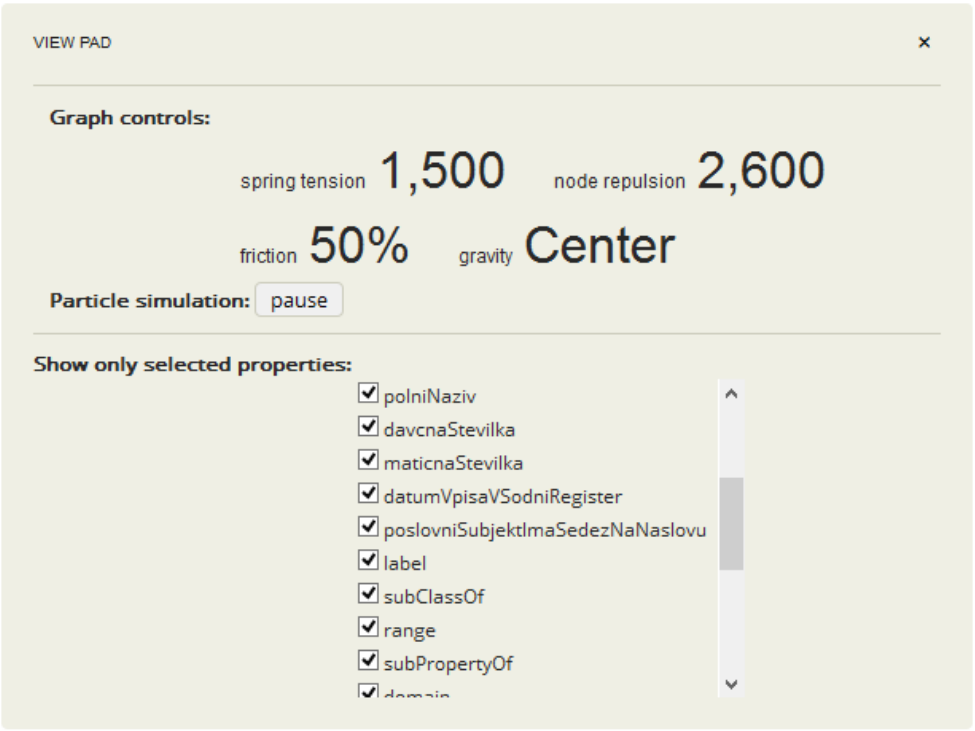
Menijska okna se aktivirajo s klikom na eno od ikon v osnovnem meniju. Osnovni meni je na sliki 5.4 označen s številko 1, položaji prikaza menijskih oken pa s črko A.

Menijsko okno "SEARCH PAD", prikazano na sliki 5.6, omogoča iskanje po imenih primerkov razredov ali pa iskanje primerkov glede na določen razred ali razredno lastnost. Ravno zato je okno razdeljeno na dve spustni podokni. Prvo smo poimenovali "search by person/company/institution", drugo pa "search by properties". V splošnem lahko iščemo glede na več različnih lastnosti, prav tako so lahko iskalni nizi, ki jih vnašamo, tudi le delni. Tako nam ni treba povsem opredeliti iskanega elementa, kar precej olajša morebitno ne povsem določeno iskanje. To lahko opazimo tudi na priloženi sliki. Na primer naslov poslovnega subjekta (lastnost *poslovniSubjetkImaSedezNaNaslovu*) vsebuje le ime kraja in nedokončano ime ulice. Tudi naziv (lastnost *polniNaziv*) samostojnega podjetnika je naveden le z dejavnostjo, ki jo opravlja. Delno zapisan pa je ne nazadnje tudi datum vpisa v sodni register (lastnost *datumVpisaVSodniRegister*), saj je navedeno le leto, manjka pa še navedba meseca in dneva. Ko oddamo iskalni zahtevek (pritisnemo gumb za iskanje), aplikacija v ozadju avtomatično ustvari poizvedbo SPARQL in jo posreduje na dostopno točko. Koda 5.10 prikazuje avtomatično ustvarjeno poizvedbo iz konkretnega primera.

```
PREFIX d: <http://www.lavbic.net/owl/AJPES.owl#>

SELECT ?s  ?l0  (COUNT(?s) AS ?count)
WHERE {
?someobj ?p ?s .

        ?s d:poslovniSubjektImaSedezNaNaslovu ?l0.
FILTER ( regex(str(?l0), 'ljubljana',"i")
        && regex(str(?l0), 'trzaska',"i") ).

        ?s d:polniNaziv ?l1.
FILTER ( regex(str(?l1), 'frizer',"i") ).

        ?s d:datumVpisaVSodniRegister ?l2.
FILTER ( regex(str(?l2), '2002',"i") ).

FILTER (!regex(str(?s), '^http://www.w3.org/2001/XMLSchema'))
   .
FILTER (!regex(str(?s), '^http://www.w3.org/1999/02/
        22-rdf-syntax-ns')).
FILTER (!regex(str(?s), '^http://www.w3.org/2000/01/
        rdf-schema')).
FILTER (!regex(str(?s), '^http://www.w3.org/2002/07/owl')).
```

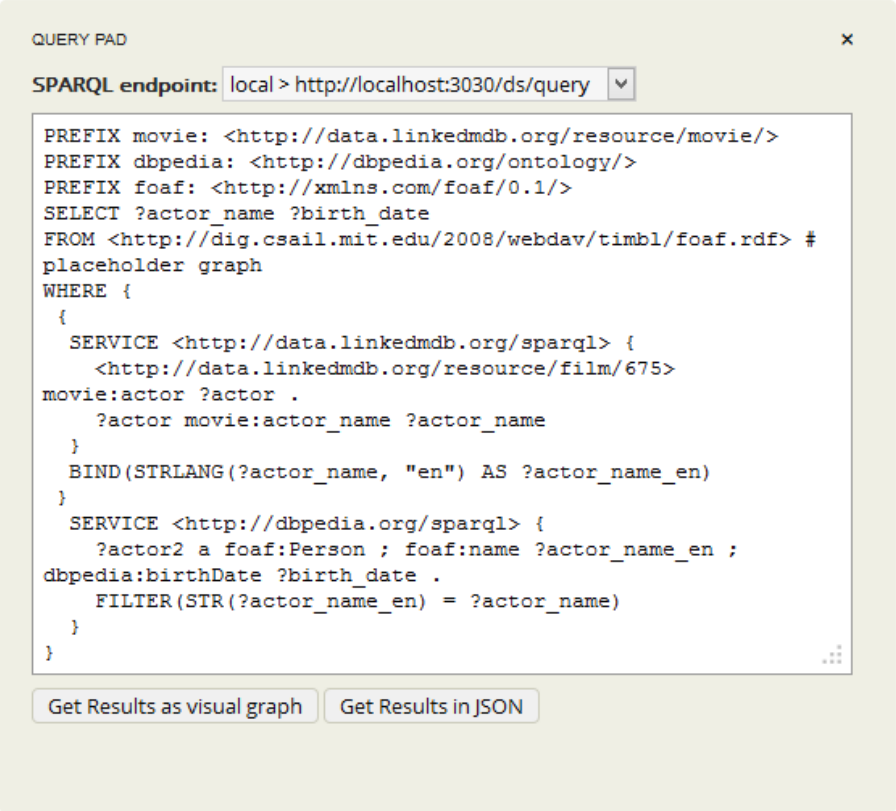
```
FILTER (!regex(str(?s), '^http://www.w3.org/2005/
        xpath-functions')).
FILTER (!isLiteral(?someobj)).
}
GROUP BY ?s ?l0
ORDER BY DESC(?count)
```

Koda 5.10: Avtomatično ustvarjena poizvedba SPARQL glede na iskani zahtevek s slike 5.6.



Slika 5.7: VisualRDF – "VIEW PAD" – menijsko okno za konfiguracijo grafičnih elementov na zaslonu.

Menijsko okno "VIEW PAD", prikazano na sliki 5.7, je namenjeno konfiguraciji grafičnih elementov na zaslonu. Za vsako poizvedbo (prek grafičnega vmesnika ali ročno prek lastnih poizvedb) je mogoče omejiti prikaz elementov le na izbrane lastnosti. Seznam se posodobi avtomatsko, glede na trenutne rezultate poizvedbe. Poleg tega je v tem oknu mogoče upravljati parametre obnašanja povezav in vozlišč pri njihovi grafični simulaciji.



Slika 5.8: VisualRDF – "QUERY PAD" – menijsko okno za ročno vnašanje poizvedb SPARQL.

Menijsko okno "QUERY PAD", prikazano na sliki 5.8, je namenjeno zahtevnejšim uporabnikom, ki podrobneje poznajo tehnologije semantičnega spleta. Okno ponuja grafični vmesnik za neposredni vnos ročno sestavljenih poizvedb SPARQL. Rezultat takih poizvedb je besedilo v obliki zapisa JSON ali grafični prikaz v obliki grafa na zaslonu. S tem načinom je mogoče prikazati grafe kompleksnejših poizvedb, ki jih sicer z osnovno funkcionalnostjo aplikacije ne bi mogli. Na sliki je vnešena poizvedba, ki poišče datume rojstev vseh igralcev iz filma *Star Trek: The Motion Picture*. Podatki se sicer pridobijo iz zunanjih dostopnih točk.

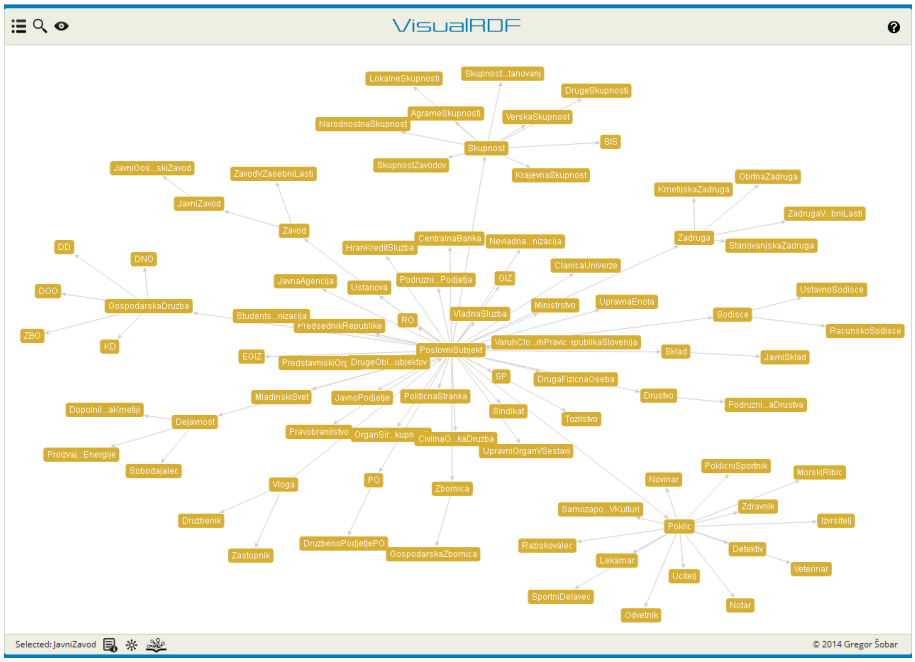
Vmesnik s primerom uporabe

Ob prvem obisku spletne aplikacije se v ozadju iz baze prenese shema podatkov, tj. opis, kako so podatki v bazi med seboj logično povezani. Preneseni podatki so torej trojčki, ki vsebujejo predikate *rdf:type*, *owl:Class* ter *rdfs:label*. Poleg tega se v ozadju prenese tudi struktura razredov in podrazredov, ki jo vmesnik grafično predstavi v obliki grafa na osrednjem delu zaslona.

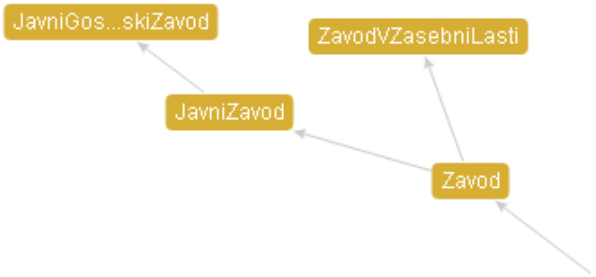
Uvodna stran je razvidna na sliki 5.9. Za pridobitev podatkov o strukturi razredov in podrazredov iz baze je potrebna poizvedba po predikatu *rdfs:subClassOf*. Za tako poizvedbo zadostuje preprost SPARQL stavek: *SELECT ?s ?p { ?s rdfs:subClassOf ?p }*. S takim stavkom pridobimo seznam podatkov z dvema stolpcema, ki ju predstavljata spremenljivki *?s* in *?p*. V spremenljivki *?p* dobimo nadrazred, v *?s* pa podrazred. Grafično oba razreda predstavimo kot vozlišči, podrejenost podrazreda v primerjavi z razredom pa ponazorimo z usmerjeno povezavo, ki je iz razreda usmerjena proti podrazredu. Če isti postopek ponovimo za vse prenešene podatke, dobimo graf, kot je na omenjeni sliki.

Od uvodne strani dalje imamo na izbiro več možnosti: lahko iščemo prek elementov, trenutno predstavljenih na grafu (torej prek vozlišč), ali pa neposredno po želenih lastnostih prek iskalnega okna. Na kratkem primeru bomo predstavili oba načina. S tem želimo pokazati širino aplikacije, ki uporabniku omogoča dostop do želenih vsebin na več načinov, glede na trenutne zahteve.

Vzemimo primer, da si želimo ogledati vse dostopne informacije o javnem zavodu Univerza v Ljubljani. Opis bo sledil rdečim puščicam na sliki 5.10. Ob kliku z desnim miškinim gumbom na vozlišče *JavniZavod* se je odprlo majhno pojavno okno, ki nam je ponudilo nekaj več osnovnih informacij o izbranem vozlišču ter nekaj dodatnih akcijskih ikon. Izbrali smo akcijsko ikono za prikaz podrobnejših informacij, ki je na sliki označena z rdečo številko 2. Na desnem delu zaslona, v drsnem oknu, se je pojavil podrobnejši opis izbranega elementa. Vmesnik je pri posameznih lastnostih iz seznama ponudil še nadaljnje mogoče akcije, ki omogočajo iskanje po izbranih lastnostih. Ker



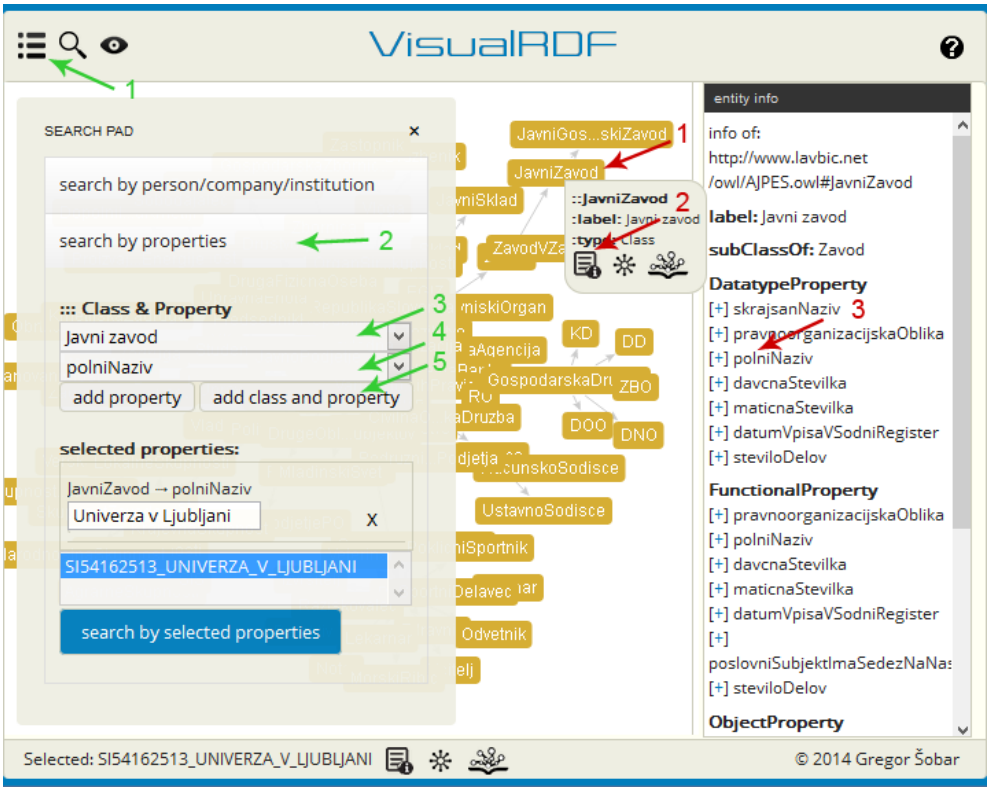
(a) celozaslonski posnetek



(b) izsek celozaslonskega posnetka

Slika 5.9: VisualRDF – grafični prikaz razredov in podrazredov.

želimo iskati po nazivu javnega zavoda, smo nadalje izbrali lastnost *polniNaziv* (natančneje znak + pred njim). V tistem trenutku se nam je odprlo iskalno okno (”SEARCH PAD”), vidno na levem delu slike. Izbrana lastnost se je avtomatično vstavila v modul za iskanje. Lastnost se je v modulu za iskanje zapisala kot *JavniZavod->polniNaziv*. Pod njim pa se je pojavilo tudi polje za vpis niza.



Slika 5.10: VisualRDF – prikaz dveh načinov dostopa do razreda in lastnosti.

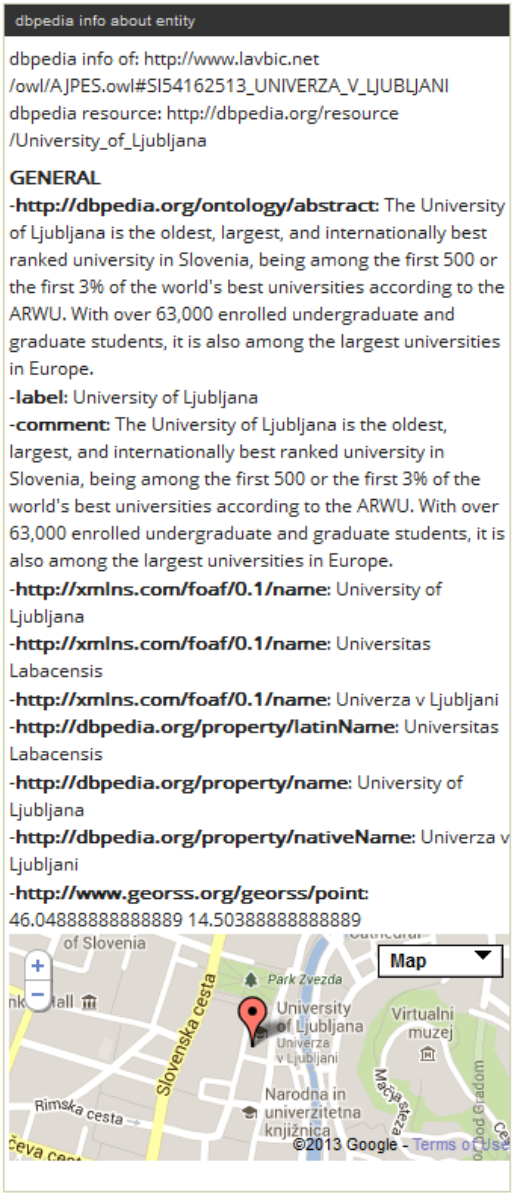
Do istega rezultata bi lahko prišli tudi po drugi poti. Drugi način prikazujejo zelene puščice na sliki. Če jim sledimo, bi najprej morali izbrati iskalno okno prek zgornjega menija, kar prikazuje puščica s številko 1. Nato bi morali izbrati spustno podokno "search by properties". Od tu pa bi morali na spustnem seznamu še ročno izbrati razred *Javni zavod* in lastnost *polniNaziv* (na sliki označena s puščicama 3 in 4). Po izbiri obeh parametrov bi morali pritisniti gumb "add class and property". S tem postopkom bi prišli do povsem istega rezultata kot v prejšnjem odstavku. Postopek je nekoliko daljši kot prvi, vendar bolj univerzalen in omogoča več funkcionalnosti.

V prejšnjih dveh odstavkih smo opisali dva načina, kako pridemo do polja, kamor lahko vneseno iskalni niz. Zato bomo nadaljevali od tu. V prazno polje lastnosti *polniNaziv* smo vpisali iskalni niz, torej kar ime zavoda "Univerza

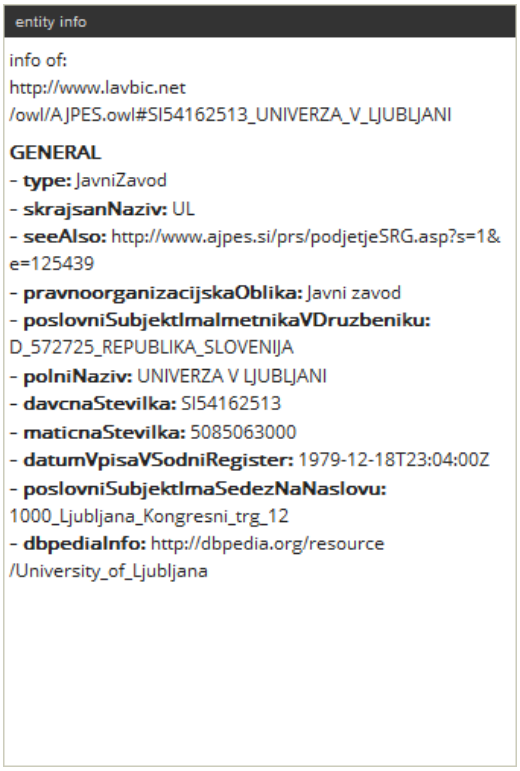
v Ljubljani". Aplikacija je seveda dovolj zmogljiva, da lahko išče tudi po le delnih nizih. V primeru nepopolnega niza pa bi najverjetneje vrnila več možnih rezultatov, izmed katerih bi uporabnik sam izbral najustreznejšega. V tokratnem primeru je bil vrnjen le en rezultat. Niz je bil dovolj enoličen, da je aplikacija natanko vedela, kaj iščemo. Po izbranem rezultatu se je spremenila tudi statusna vrstica. Poleg izpisa imena nam je ponudila tudi nekaj dodatnih akcijskih ikon.

Če bi izbrali akcijsko ikono za podrobnejši izpis informacij o najdenem rezultatu (levi gumb v statusni vrstici), bi se odprlo drsno okno, razvidno na sliki 5.11b. Okno vsebuje podrobnejše informacije o izbranem elementu, ki se nahajajo v lokalni bazi. V našem primeru so to informacije iz baze AJPES.

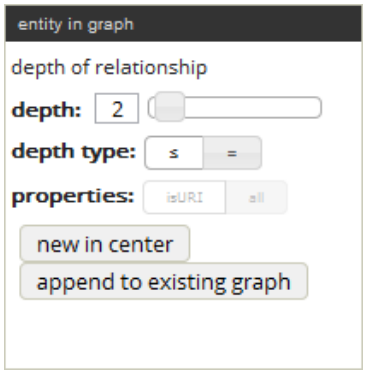
Poleg podrobnega izpisa o rezultatu imamo na voljo tudi prikaz elementa v grafu. Z izbiro srednje ikone v statusni vrstici se odpre drsno okno, kar prikazuje slika 5.11c. V njem imamo na izbiro, ali želimo element vstaviti v prazen graf (omogoča gumb z besedilom "new in center") ali pa ga želimo vstaviti kot dodatno vozlišče k obstoječim vozliščem v trenutnem grafu (to omogoča gumb z besedilom "append to existing graph"). Nastavimo lahko tudi globino. Globina predstavlja najmanjše število povezav, ki so potrebne, da od izbranega vozlišča v grafu pridemo do poljubnega drugega vozlišča. Globina 1 tako predstavlja vsa sosednja vozlišča izbranega vozlišča. V splošnem lahko grafično prikažemo vozlišča s poljubno globino, vendar se število vozlišč s povečevanjem oddaljenosti (globine) eksponentno povečuje. V splošnem bi lahko s pomočjo teh kontrol tudi grafično preverili, ali so različni iskani elementi med seboj povezani. To bi naredili tako, da bi prek iskalnika poiskali posamezne elemente ter jih z gumbom "append to existing graph" v tem oknu dodali v skupni graf. Z določitvijo globine pa bi uravnavali še dovoljeno oddaljenost med vozlišči.



(a) podatki o elementu iz DBpedie



(b) podatki o elementu iz AJPES



(c) nastavitev prikaza elementa v grafu

Slika 5.11: VisualRDF – drsno okno z različnimi funkcionalnostmi.

Kot zadnjo možnost akcij nad izbranim elementom nam desna ikona v statusni vrstici omogoča prikaz dodatnih informacij o elementu iz zunanjih virov. Rezultat te funkcionalnosti je razviden na sliki 5.11a. Slika prikazuje

podatke, ki so na voljo o izbranem elementu v bazi DBpedie. Poleg izpisa osnovnega seznama prenešenih podatkov se je na dnu drsnega okna prikazal tudi zemljevid. V aplikacijo smo namreč implementirali detekcijo geolokacijskih koordinat. Če so podatki na voljo (tako kot v tem primeru), se torej prikaže tudi zemljevid.

5.7 Testiranje in podpora

Testiranje spletne aplikacije je potekalo tako v razvojnem okolju Aptana Studio 3 kot tudi v končnih odjemalcih. Končni odjemalci so v našem primeru seveda spletni brskalniki. Razvojno okolje nam je pomagalo ugotavljati in odpravljati težave na področju implementacije funkcionalnosti in njihovega delovanja. Pri brskalnikih pa smo v okviru razvojnih vtičnikov posameznih brskalnikov preverjali predvsem odzivnost in podporo določenim tehnologijam ter morebitnim anomalijam pri njihovi implementaciji.

Podpora brskalnikom

Kljub precejšnji zavzetosti razvijalcev brskalnikov je podpora standardu HTML5 še vedno pomanjkljiva. Ker je naša aplikacija v začetni fazi razvoja, smo se odločili, da sprva upoštevamo najpomembnejše brskalnike – predvsem tiste, ki so dostopni na čim večjem številu naprav, ne glede na operacijski sistem. Aplikacijo smo testirali v brskalnikih Mozilla Firefox 27+, Google Chrome 32+ ter Opera 18+, ki je še posebej močno zastopana na mobilnih napravah.

Poglavje 6

Sklepne ugotovitve

V diplomskem delu smo raziskali koncept semantičnega spleta. Pregledali smo njegove prednosti in tehnologije ter trende razvoja, ki jih nakazuje. V okviru diplomskega dela smo izdelali prototip pregledovalnika omrežja podatkov, zapisanih v jeziku RDF. Pri implementaciji aplikacije smo se oprli na vedno bolj uveljavljen standard HTML5 in različne JavaScript knjižnice, ki so nam olajšale delo pri vizualizaciji grafov in drugih funkcionalnostih.

Med razvojem smo se srečali s številnimi izzivi: od splošnih, kot sta nameščanje in konfiguracija podatkovnega in spletnega testnega strežnika, do zapletenejših, povezanih z razvojem funkcionalnosti, kot je avtomatsko ustvarjanje poizvedb po poljubnem podatkovnem viru, shemi podatkov in podatkih samih. Precej časa smo namenili tudi razvrščanju različnih grafičnih elementov ter funkcionalnosti po zaslonu. Motivirala nas je težnja k minimalizmu in preglednosti vmesnika, vendar ne na račun zmogljivosti in uporabnosti aplikacije.

V aplikacijo smo implementirali vse funkcionalnosti, ki smo jih za osnovo prototipa načrtovali na začetku. V glavnem so to: vizualizacija trojčkov RDF na podlagi ročno napisanih ali avtomatično ustvarjenih poizvedb SPARQL; grafični vmesnik, ki omogoča sprehajanje po prikazanem grafu; možnost iskanja in filtriranja podatkov iz dostopne točke; dodatna konfiguracija prikaza; odziven, nadvse preprost in pregleden grafični vmesnik; možnost menjave

virov podatkov. Kljub temu pa ostaja še veliko možnosti za razširitev aplikacije. Zelo prav bi na primer prišlo avtomatizirano iskanje poti med dvema ali več izbranimi elementi, česar nismo vključili, saj bi morali poseči v kodo podatkovnega strežnika, s tem pa bi se oddaljili od področja diplomskega dela.

Koda prototipa aplikacije VisualRDF je prosto dostopna in odprtokodna [29].

Literatura

[1] A. Goldstein, E. Weyl, L. Lazaris and R. Weakley, *HTML5 & CSS3 for the real world*, 1st ed. Collingwood, Vic., Australia: SitePoint Pty. Ltd., 2011.

[2] D. McFarland and D. McFarland, *JavaScript & jQuery*, 1st ed. Sebastopol, Calif.: O'Reilly, 2011.

[3] J.C. Teague, *CSS3: Visual QuickStart Guide*, 1st ed. Berkeley, CA: Peachpit Press, 2011.

[4] B. DuCharme, *Learning SPARQL*, 1st ed. Sebastopol, O'Reilly, 2011.

[5] T. Berners-Lee, J. Hendler and O. Lassila, 'The Semantic Web', *Scientific American*, maj 2001, str. 29–37.

[6] L. Feigenbaum, I. Herman, T. Hongsermeier, E. Neumann and S. Stephens, 'The Semantic Web in Action', *Scientific American*, vol. 297, dec. 2007, str. 90–97.

[7] D. Herrmannova and P. Knoth, 'Visual Search for Supporting Content Exploration in Large Document Collections', *D-Lib Magazine*, vol. 18, no. 78, 2012. Dostopno na: `http://dx.doi.org/10.1045/july2012-herrmannova`.

[8] I. Herman, 'Semantic Web Adoption and Applications', *W3.org*, 2012. Dostopno na: `http://www.w3.org/People/Ivan/CorePresentations/Applications/Applications.pdf`.

[9] L. Feigenbaum and E. Prud'hommeaux, 'SPARQL by Example – Cambridge Semantics', *Cambridgesemantics.com*, 2013. Dostopno na: http://www.cambridgesemantics.com/sl/semantic-university/sparql-by-example.

[10] D. Lavbič, 'Dejan Lavbič – Delo in raziskovanje – Semantični splet', *Lavbic.net*, 2014. Dostopno na: http://www.lavbic.net/delo-in-raziskovanje/semanticni-splet/.

[11] S. Puntar, *Pregled in primerjava triplestore podatkovnih baz*, Diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2012.

[12] T. Berners-Lee, 'Information Management : A Proposal', *Cern.ch*, 1989. Dostopno na: http://cds.cern.ch/record/1405411/files/ARCH-WWW-4-010.pdf.

[13] C. Swinehart, 'arbor.js', *Arborjs.org*, 2014. Dostopno na: http://arborjs.org/.

[14] T. Ventimiglia and K. Wayne, 'COS 126 Programming Assignment: Barnes-Hut Galaxy Simulator', *Cs.princeton.edu*, 2003. Dostopno na: http://www.cs.princeton.edu/courses/archive/fall03/cs126/assignments/barnes-hut.html.

[15] J. Garrett, 'Ajax: A New Approach to Web Applications', *Adaptivepath.com*, 2005. Dostopno na: http://web.archive.org/web/20110102130434/http://www.adaptivepath.com/ideas/essays/archives/000385.php.

[16] The Apache Software Foundation, 'Apache Jena – A free and open source Java framework for building Semantic Web and Linked Data applications', *Jena.apache.org*, 2014. Dostopno na: http://jena.apache.org.

[17] W3.org, 'LargeTripleStores – W3C Wiki', 2014. Dostopno na: `http://www.w3.org/wiki/LargeTripleStores`.

[18] Dbpedia.org, 'wiki.dbpedia.org : About', 2007. Dostopno na: `http://www.dbpedia.org/About`.

[19] Internetworldstats.com, 'Internet Growth Statistics – the Global Village Online', 2014. Dostopno na: `http://www.internetworldstats.com/emarketing.htm`.

[20] W3.org, 'Category:Visualizer – Semantic Web Standards', 2014. Dostopno na: `http://www.w3.org/2001/sw/wiki/Category:Visualizer`.

[21] Cambridgesemantics.com, 'Anzo Solutions Overview – Cambridge Semantics', 2014. Dostopno na: `http://www.cambridgesemantics.com/home`.

[22] K. Ono, 'Cytoscape: An Open Source Platform for Complex Network Analysis and Visualization', *Cytoscape.org*, 2014. Dostopno na: `http://cytoscape.org/`.

[23] Bioinformatics.org, 'RDFScape : Home Page', 2014. Dostopno na: `http://www.bioinformatics.org/rdfscape/wiki/`.

[24] P. Heim and S. Lohmann, 'Tools - Visual Data Web', *Visualdataweb.org*, 2014. Dostopno na: `http://www.visualdataweb.org/tools.php`.

[25] Semweb.salzburgresearch.at, 'RDF-Gravity', 2014. Dostopno na: `http://semweb.salzburgresearch.at/apps/rdf-gravity/index.html`.

[26] Rhizomik.net, 'Rhizomik', 2014. Dostopno na: `http://rhizomik.net/html/rhizomer/`.

[27] Dev.data2000.no, 'Sgvizler development site', 2014. Dostopno na: `http://dev.data2000.no/sgvizler/`.

[28] N. Spivack, 'Semantic Web Talk – Making Sense of the Semantic Web', *Slideshare.net*, 2007. Dostopno na: `http://www.slideshare.net/syawal/nova-spivack-semantic-web-talk`.

[29] G. Šobar, 'VisualRDF', *Bitbucket.org*, 2014. Dostopno na: `https://bitbucket.org/SGrega/visualrdf/`.