Semantički REST servisi

Vujasinović, Tin

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Tin Vujasinović

Semantički REST servisi

DIPLOMSKI RAD

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Tin Vujasinović

Matični broj: 43503/14–R

Studij: Informacijsko i programsko inženjerstvo

Semantički REST servisi

DIPLOMSKI RAD

Mentor/Mentorica:

Izv. prof. dr. sc. Darko Andročec

Varaždin, rujan 2024.

Tin Vujasinović

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Ovaj rad istražuje primjenu semantičkih web tehnologija u REST servisima kako bi se unaprijedila interpretacija i obrada podataka. Kroz analizu trenutnog stanja web tehnologija, posebno REST i SOAP servisa, rad se fokusira na prednosti uvođenja semantičke komponente. Glavni cilj je prikazati kako semantičke tehnologije, kao što su RDF i OWL, omogućavaju bolje razumijevanje podataka i njihovo međusobno povezivanje. Implementacija semantičkog REST servisa uključuje korištenje standarda kao što su SA-REST i Hydra, koji omogućavaju dodavanje semantičkih metapodataka i olakšavaju interoperabilnost podataka. Predstavljena je i implementacija sustava koji integrira podatke iz različitih izvora, semantički ih obogaćuje i pohranjuje u RDF spremište, što omogućava napredno pretraživanje putem SPARQL-a. Rad zaključuje da je korištenje semantičkih tehnologija u REST servisima ključno za budući razvoj weba, jer omogućava automatiziranu obradu podataka, bolje razumijevanje konteksta i efikasnije upravljanje informacijama.

Ključne riječi: semantika, web servisi, RDF, REST servisi, semantički web, SPARQL

Sadržaj

1. Uvod

Ubrzani razvoj interneta i povećani broj međusobno povezanih uređaja doveo je do velikih količina podataka koje se generiraju svakoga dana. Takve količine podataka otežavaju korisnicima pronalaženje i pristup relevantnim informacijama. Informacije su dostupne putem raznih tražilica, internetskih enciklopedija, blogova, servisa i drugih načina za prikaza tih informacija. U ovome radu baviti ćemo se servisima koji korisnicima vraćaju podatke o određenoj temi. Većina takvih servisa u današnje je vrijeme implementirana kroz reprezentacijski prijenos stanja (engl. *Representational State Transfer*), što je arhitektura koja je naširoko prihvaćena kao standard za izradu web servisa. Ova arhitektura omogućava jednostavnu i efikasnu komunikaciju između uređaja, no jedan od nedostataka je što servisima koji vraćaju podatke nedostaje semantička komponenta kako bi se podaci koje vraćaju mogli lakše interpretirati.

Za rješavanje tog problema uvedene su semantičke web tehnologije kao što su RDF (engl. *Resource Definition Framework*) i OWL (engl. *Web Ontology Language*) koje omogućavaju obogaćivanje podataka semantičkim oznakama kako bi bili lakše dostupni, razumljivi te kako bi se mogli ponovo iskoristiti. Cilj ovog rada biti će istražiti kako se takve tehnologije mogu koristiti za kreiranje semantičkih REST servisa.

Ovaj rad istražiti će mogućnosti tehnologija za obogaćivanje semantičkim oznakama te njihovu kombinaciju s web servisima koji su implementirani pomoću REST arhitekture. Glavno pitanje je kako se ove tehnologije mogu kombinirati kako bi podaci bili, kako smo ranije naveli, lakše dostupni, razumljivi i kako bi se mogli ponovo koristiti. Fokus će biti na implementaciji semantičkog REST servisa koji će biti ograničen na podatke iz jedne domene, to će biti podaci o lokacijama, vremenskoj prognozi na toj lokaciji te prirodnim pojavama u blizini. Rad će se sastojati od sljedećih dijelova:

- istraživanje trenutno dostupnih tehnologija poput RDF, SPARQL
- odabir prikladne tehnologije za kombiniranje sa REST servisom
- opis rješenja i servisa
- implementacija servisa

Prije samog istraživanja i implementacije objasniti ćemo ukratko što je semantika, kako se uklapa u svijeta internetskih stranica, kako se takve stranice grade i od čega se sastoje te kasnije napraviti uvod u web servise, ukratko objasniti kako se implementiraju te prijeći na implementaciju semantičkog web servisa.

2. Semantika

Ovaj pojam većini ljudi koji se ne bave nekim proučavanjem semantike i ne razmišljaju o njoj donekle zvuči apstraktno. Semantika je svuda oko nas i svakodnevno se svatko od nas susreće s nekim oblikom semantike, iako toga nismo ni svjesni jer je kao takvu ne percipiramo već stvari koje imaju semantička obilježja uzimamo „zdravo za gotovo“. Možemo uzeti semafor na raskrižju kao primjer. Svakome od nas jasno je da kada je na semaforu crveno svjetlo, moramo stati. U ovom primjeru sama boja daje semantičku vrijednost običnom svjetlu koje stoji na prometnicama naših gradova. Na isti način, znak „Stop“ ima crvenu pozadinu te nam podsvjesno daje do znanja da se trebamo zaustaviti. S druge strane, zelena boja ima upravo suprotan efekt, daje nam do znanja da slobodno nastavimo. Na ovaj način, z pomoć boja, možemo dati dodatnu informaciju nečemu. Tako će rezultati ispita ispisani u crvenoj boji jasno dati studentu do znanja da nije prošao ispit, takoreći, ne može nastaviti, dok će zelena boja predstavljati prolaz i nastavak na kolegiju/studiju.

Semantika kao pojam pojavila se kraj 19. stoljeća u časopisu „*Reflected meanings: a point in the semantics“* [1]. Razvila se ponajprije kada su jezikoslovci proučavali riječi i njihovo značenje. Definicija semantike je sljedeća: „Semantika je termin koji se odnosi na proučavanje značenja.“ [1]. Kasnije se krenulo općenito proučavati značenje pa je tako semantika odmakla samo od značenja pojedinih riječi. Ovaj rad fokusirati će se na opis i značaj servisa koji se koriste za funkcioniranje današnjeg interneta.

2.1. Semantika u web tehnologijama

U web tehnologijama, kada se govori o semantici, ponajprije se misli na značenje i interpretaciju web stranica i njihovog sadržaja. Semantičke web tehnologije koriste meta podatke za definiranje veza između web resursa i pružanje dodatnih informacija o njima. Te informacije pomažu računalima da razumiju sadržaj i namjenu web stranica te tako interpretiraju njihovo značenje, što je posebno korisno tražilicama koje lakše klasificiraju web stranice čime se njihova točnost poboljšava.

Novije web stranice razvijene su pomoću moderne HTML 5 tehnologije koja se sastoji od semantičkih elemenata kao što su *header*, *footer*, *section*, *nav*, *aside* i ostalih. Pravilnim korištenjem ovih elemenata daje se dodatan kontekst sadržaju web stranice i dijelovi web stranice na smislen se način odvajaju u cjeline. Korištenje semantičkih web tehnologija omogućava tim uređajima da razumiju i komuniciraju s web stranicama što dovodi do boljeg korisničkog iskustva.

3. Web

Ono što se danas kolokvijalno naziva web ili internet, globalni je sustav međusobno umreženih dokumenata i resursa čija se lokacija unutar mreže definira URL-om, tj. (engl. *Unified Resource Locator*) adresom. Njegov je začetnik Tim Berners-Lee koji je u 90-im godinama prošlog stoljeća postavio temelje razvojem tehnologija HTML (engl. *Hypertext Markup Language*), HTTP (engl. *HyperText Transfer Protocol*) i URL [2].

Web stranice su glavne komponente weba. Web je zbirka web stranica koje sadrže tekst, slike, video zapise i druge vrste sadržaja. Ovim stranicama pristupa se putem web preglednika poput Google Chrome, Mozilla Firefox ili Safari. Web omogućuje različite aktivnosti, uključujući komunikaciju (e-pošta, društvene mreže, poruke), dohvaćanje informacija (tražilice), online kupovinu, zabavu (usluge streaminga, igranje igara) i još mnogo toga. Transformirao je način na koji pristupamo i dijelimo informacije, obavljamo posao i povezujemo se s drugima diljem svijeta. Web se temelji na modelu klijent-poslužitelj, gdje klijenti (web preglednici) zahtijevaju i primaju resurse od poslužitelja. Za funkcioniranje weba potrebno je nekoliko ključnih komponenti i tehnologija [2]:

- HTML: HTML je standardni jezik koji se koristi za stvaranje web stranica. Strukturira sadržaj i definira izgled web stranice pomoću oznaka. Oznake označavaju elemente poput naslova, odlomaka, slika, veza, obrazaca i još mnogo toga.
- HTTP: HTTP je protokol koji se koristi za komunikaciju između klijenata i poslužitelja na webu. Kada korisnik unese URL u preglednik ili klikne na poveznicu, preglednik šalje HTTP zahtjev poslužitelju, koji odgovara HTTP odgovorom koji sadrži traženi resurs (npr. web stranicu). Također postoji HTTPS koji enkripcijom omogućava sigurnu komunikaciju.
- URL: URL-ovi se koriste za identifikaciju resursa na webu. Sastoje se od prefiksa protokola (npr. "http://" ili "https://"), imena domene (npr. "primjer.com") i dodatnih putanja i parametara upita koji određuju lokaciju resursa
- Web preglednici: Web preglednici su programske aplikacije koje omogućuju korisnicima pristupanje i navigaciju webom. Oni interpretiraju HTML, izvršavaju skripte i prikazuju web stranice, omogućavajući korisnicima interakciju s web stranicama. Popularni web preglednici uključuju Google Chrome, Mozilla Firefox, Microsoft Edge i Safari
- Web poslužitelji: Web poslužitelji su računala ili softver koji poslužuju web stranice i odgovaraju na zahtjeve klijenata. Kada preglednik pošalje HTTP zahtjev, web

poslužitelj obrađuje zahtjev, dohvaća traženi resurs i šalje ga natrag kao HTTP odgovor

- Hiperlinkovi: Hiperlinkovi su elementi unutar web stranica koji korisnicima omogućuju navigaciju između različitih resursa na webu. Obično su prikazani kao poveznice ili slike koje, kada se kliknu, korisnika usmjeravaju na drugu web stranicu ili na drugi dio iste stranice
- CSS (engl. *Cascading Style Sheets*): CSS je stilski jezik koji se koristi za opisivanje izgleda i oblikovanja HTML dokumenata. On definira stilove za elemente poput boja, fontova, izgleda i pozicioniranja. CSS pomaže odvojiti sadržaj (HTML) od njegove vizualne reprezentacije [3]
- JavaScript: JavaScript je programski jezik koji omogućuje dinamičko i interaktivno ponašanje na web stranicama. Omogućuje razvojnim programerima stvaranje efekata, provjeru formi, upravljanje korisničkim interakcijama i modificiranje sadržaja web stranica u stvarnom vremenu [3]

Zajedno, ove tehnologije čine temelje weba, omogućavaju stvaranje, dostavu i konzumaciju informacija i usluga. Promijenile su razne aspekte naših života, uključujući komunikaciju, trgovinu, obrazovanje, zabavu i još mnogo toga.

4. Web servisi

Pod pojmom web servisi obuhvaćen je software, aplikacije ili druge tehnologije (oblak) koje korisnicima pružaju mogućnosti komunikacije, interoperabilnosti i razmjene podataka kroz standardizirane web protokole (HTTPS ili HTTP) putem interneta. Komunikacija i razmjena podataka najčešće se odvija korištenjem XML (engl. *Extensible Markup Language*) formata, koji se koristi za označavanje podataka, a definiran je pravilima kako ga ispravno koristiti. Glavna značajka web servisa je što aplikacije koje su programirane u različitim programskim jezicima mogu međusobno komunicirati i razmjenjivati podatke, kao klijenti koji komuniciraju s poslužiteljima. Klijent će poslati zahtjev u XML formatu, a poslužitelj će mu poslati odgovor, također u XML formatu.

Web servisi omogućavaju različitim aplikacijama da komuniciraju međusobno koristeći niz standarda otvorenog koda, kao što su SOAP, XML, HTML, WSDL i drugi. Kako postoji niz različitih standarda za komunikaciju, postoje i razni protokoli i standardi za web servise od kojih svaki ima drugačiju ulogu. Neki od važnijih protokola web servisa su opisani u nastavku.

XML-RPC (engl. *XML Remote Procedure Call*): jedan od najosnovnijih XML protokola koji služi za pozivanje procedura između raznih uređaja putem interneta. Putem jednostavnih HTTP zahtjeva brzo i lako prenosi komunikaciju između klijenta i poslužitelja. Dizajniran je da bude jednostavan, ali ima mogućnosti prenositi kompleksne strukture podataka.

UDDI (engl. *Universal Description, Discovery, and Integration*): standard baziran na XML protokolu za opisivanje, objavljivanje i otkrivanje web servisa. Samo ime govori da se radi o univerzalnom načinu označavanja koji korisnicima olakšava pronalazak web servisa. Služi kao registar za poduzeća s ciljem da olakša digitalno poslovanje s drugim poduzećima.

SOAP (engl. *Simple Object Access Protocol*): standard za web servise koji je također baziran na XML-u, omogućava razmjenu podataka i dokumenata putem HTTP, SMTP (engl. *Simple Mail Transfer Protocol*) i drugih protokola. Omogućava nezavisnim procesima na sustavima koji mogu biti potpuno različiti da međusobno komuniciraju.

REST (engl. *Representational State Transfer*): arhitektura za web servise koja otvara sučelje aplikacije prema ostatku interneta uz dokumentaciju i upute kako se s tom aplikacijom komunicira. Većina web servisa implementirana na ovaj način koristi HTTP za prijenos zahtjeva od klijenta do aplikacije

4.1. SOAP i REST

Ova dva standarda najčešće se javljaju u modernim web servisima, pa se često postavlja pitanje koji je od njih bolji, no jednostavan odgovor je da svaki od njih ima neke prednosti te zavisi od slučaja do slučaja gdje će se koristiti koji standard. Ova dva servisa nude različite mogućnosti pa je, na primjer, u slučaju kad su vrijeme izvršavanja i performanse važnije, bolje koristiti REST, dok će se SOAP koristiti u slučaju gdje postoje specifični zahtjevi, kao da se za prijenos koristi neki drugi protokol umjesto HTTP.

SOAP pruža sloj protokola za razmjenu poruka u sklopu protokola web usluga. To je protokol baziran na XML-u čije su glavne karakteristike [4]:

- Interoperabilnost i standardizacija: za komunikaciju različitih aplikacija nije potreban dodatan kod jer SOAP definira pravila za razmjenu podataka
- Proširivost**:** protokol je dizajniran na način da se nove funkcionalnosti lako dodaju, bez utjecaja na postojeće
- Sigurnost**:** postoji nekoliko mehanizama koji se koriste kako bi se osiguralo da samo autorizirani korisnici mogu pristupiti podacima što omogućava slanje osjetljivih podataka

SOAP ima primjenu u raznim tipovima aplikacija, kao što su integracije između poduzeća, poslovni subjekti mogu ga koristiti kao sučelje za primanje uplata od kupaca i drugim slučajevima gdje je potrebno otvoriti standardizirani pristup nekom sustavu. Uobičajena razmjena između sustava i SOAP servisa odvija se tako da klijent pripremi poruku sa SOAP zahtjevom koji tada šalje na poslužitelj putem nekog od internetskih protokola, npr. HTTP. Poruka koja se šalje naziva se omotnica, koja se sastoji od zaglavlja i tijela. U zaglavlju se mogu nalaziti detalji o autentifikaciji, enkripciji i slično. Tijelo omotnice sadrži zahtjev i podatke koji se razmjenjuju između klijenta i poslužitelja. Poslužitelj tijelo poruke provjerava zbog ispravnosti i na temelju njega odredi koju akciju treba odraditi, izvršiti neku poslovnu logiku, dohvatiti podatke iz baze podataka ili nešto treće. Nakon toga će klijentu vratiti odgovor, ponovo u obliku omotnice s rezultatom operacija koje je klijent zahtijevao.

4.2. REST

REST, također poznat kao RESTful API (engl. *Application programming interface*), je jednostavno sučelje koje omogućuje pristup podacima, medijima i drugim digitalnim resursima putem URL-a. REST API je najčešće korišten način implementacije web servisa u današnje vrijeme. Prije uvođenja REST-a, većina programera koristila je SOAP za integraciju i

implementaciju web servisa, što je bilo složeno i zahtjevno. Roy Fielding, zajedno s grupom programera, definirao je pravila za REST API-je, čime je pojednostavio razvoj i implementaciju servisa. Od prvog predstavljanja 2000. godine, REST API postao je široko prihvaćen [5]. REST API standardi definiraju jedinstveni jezik za digitalni svijet, osiguravajući dosljednost u razvoju i konzumaciji servisa. Kako bi servis bio RESTful, mora zadovoljiti šest osnovnih uvjeta:

1. Korištenje uniformnog sučelja: niz ograničenja definira ponašanje komponenti servisa, dok se svaki resurs jedinstveno definira kroz njegov URL
2. Klijent-poslužitelj arhitektura: klijenti od poslužitelja traže podatke koje potom prikazuju, dok se poslužitelj brine o sigurnosti te obradi i spremanju podataka. Odvajanje omogućava odvojeni razvoj klijenta i poslužitelja.
3. Bez stanja (engl. *stateless*): zahtjevi klijenta moraju sadržavati sve podatke potrebne za obradu, poslužitelj ne sprema nikakve informacije o klijentu
4. Predmemoriranje RESTful resursa: postoji mogućnost predmemoriranja čestih zahtjeva klijenata kako bi se odgovor poslužitelja ubrzao i smanjili troškovi
5. Slojevita arhitektura: REST se implementira u slojevima koji međusobno komuniciraju kroz određena sučelja, te osim toga, nemaju pristup drugim dijelovima ostalih slojeva
6. Kod na zahtjev: osim podatak u obliku JSON-a (*JavaScript Object Notation*) ili XML-a, poslužitelj klijentu može vratiti i kod za izvršavanje

REST API koristi resurse, koji mogu biti bilo koja informacija koja se može imenovati, poput dokumenata, slika ili zbirki drugih resursa. Ključne metode koje se koriste su: GET (za dohvaćanje podataka), PUT (za ažuriranje podataka), POST (za kreiranje novih podataka) i DELETE (za brisanje podataka). REST API koristi se u raznim područjima, uključujući aplikacije i usluge u oblaku, web i mobilne aplikacije, IoT uređaje i mnoge druge sustave [5]. Zbog svoje fleksibilnosti i jednostavnosti, REST API postao je ključna komponenta u razvoju suvremenih aplikacija. Neke od prednosti uključuju skalabilnost, fleksibilnost, neovisnost i lakoću upotrebe. REST API je lagan i brz, što ih čini idealnim za mobilne aplikacije i IoT uređaje.

Korištenje REST API-ja može donijeti izazove poput dosljednosti u oblikovanju URL-ova, verzioniranja API-ja, autentifikacije i sigurnosti API-ja. Važno je implementirati odgovarajuću autentifikaciju, autorizaciju i zaštitu podataka kako bi se izbjegle sigurnosne prijetnje. Najbolje prakse uključuju pravilno korištenje HTTP status kodova, pružanje informativnih poruka o pogreškama, osiguranje i verzioniranje API-ja, dokumentiranje API-ja i omogućavanje filtriranja, sortiranja i straničenja podataka.

4.2.1. HTTP status kodovi

Za vraćanje dodatnih informacija o rezultatu REST zahtjeva, osim traženih podataka, poslužitelj klijentu vraća statusni kod koji ovisi o uspješnosti samog zahtjeva. Kodovi su podijeljeni u pet grupa, svaka grupa ima svoje značenje [6]:

- Kodovi 1xx: zahtjev je zaprimljen i nastavlja se obrada
- Kodovi 2xx: zahtjev je uspješno obrađen
- Kodovi 3xx: potrebno je preusmjeravanje kako bi se zahtjev izvršio
- Kodovi 4xx: greška u poslanom zahtjevu, poslužitelj ga ne može obraditi
- Kodovi 5xx: greška poslužitelja, zahtjev je ispravan ali ga poslužitelj ne može obraditi

Neki od najčešćih kodova s kojima se korisnici susreću kod uspješno izvršenih zahtjeva su 200 OK, kada je zahtjev uspješno izvršen, još neki kodovi u toj grupi su 201 Created (kreirano) koji korisniku govori da je zahtjev uspješno izvršen te da je traženi resurs kreiran, 204 No Content (nema sadržaja) kada je zahtjev uspješno izvršen, ali nema novog sadržaja pa nije potrebno osvježiti klijentski prikaz.

Kodovi koji se najčešće javljaju kod greški su 404 Not Found (nije pronađeno), kada traženi resurs ne postoji na poslužitelju, 401 Not Authorised (ne autorizirano) kada se korisnik nije autentificirao prije slanja zahtjeva.

Kodovi koji se najčešće vraćaju zbog greške poslužitelja su 500 Internal Server Error (interna greška poslužitelja) koji je općeniti kod za pogrešku na poslužitelju i 503 Service Unavailable (servis nedostupan) kada je poslužitelj preopterećen te je zahtjev klijenta odbačen.

5. Semantički web servisi

Kako se u današnje vrijeme, iz dana u dan generiranju goleme količine podataka javlja se potreba za interpretacijom tih podataka. Ti podaci mogu biti različitog oblika te iz različitih izvora, što otežava pristup i razumijevanje. Rješenje za ovaj problem pružaju semantički web servisi. Semantički web servisi omogućavaju značajniju interakciju između web servisa i podataka koji pružaju, korištenjem poznatih arhitektura, kao REST, i semantičkih tehnologija, pružaju mogućnosti uređajima da osim pristupa podacima, razumiju podatke i veze između njih.

Semantički web servisi koriste standardizirane web protokole, kao što je HTTP, za uspostavljanje komunikacije između servisa. Za razliku od tradicionalnih web servisa, semantički web servisi podacima koje vraćaju pridodaju semantičke metapodatke te time povećavaju mogućnosti pristupa i korištenja podataka. Metapodaci koji se pridodaju definiraju se razvojnim okvirima kao RDF i OWL, koji definiraju veze i kategorije između podataka. Ovakav pristup omogućava uređajima da zadatke poput integracije i pronalaska podataka izvršavaju efikasnije zbog toga što bolje razumiju podatke i značenje istih [7]. Za razumijevanje semantičkih web servisa potrebno je poznavati ključne dijelove na kojima počivaju njihovi temelji. To su ontologije, RDF i povezani podaci (engl. *Linked data*).

5.1. Ontologije

Ontologija je filozofska disciplina koja se bavi proučavanjem onog što jest i čija su temeljna pitanja: „Što uopće znači biti ili postojati? Što postoji? Koja su osnovna svojstva onoga što postoji? Koji su osnovni odnosi među stvarima koje postoje?“ [8]. U kontekstu weba, ontologije predstavljaju formalni opis znanja kao skup koncepata unutar određene domene te veze između tih koncepata. Ontologije su slične klasnoj hijerarhiji koja se javlja kod objektno orijentiranog programiranja ali postoji razlika, u tome što klasne hijerarhije predstavljaju strukture koje se rijetko mijenjaju, dok se ontologije gotovo konstantno razvijaju. Za opis ontologije formalno se specificiraju komponente kao što su klase i objekti tih klasa, atributi i veze te ograničenja, pravila i aksiomi [9] – aksiomi su temeljne istine koje se ne dokazuju te se na njima gradi teorija. Ontologije ne samo da omogućavaju dijeljenje i ponovno korištenje prikaza znanja, već mogu stvarati dodatno znanje unutar neke domene.

Ontološki model podataka može se koristiti za stvaranje grafa znanja iz skupa pojedinačnih informacija. Ovaj graf se sastoji od entiteta, s čvorovima koji predstavljaju vrste i rubovima koji predstavljaju odnose među njima. Definiranjem strukture znanja u određenom

području, ontologija omogućuje grafu znanja da obuhvati i organizira te podatke. Osim ontologija, još postoje druge metode koje formalne specifikacije za prikaz znanja, kao što su logički modeli, rječnici, taksonomije i tezaurusi. Za razliku od ovih prikaza, ontologije prikazuju veze i omogućavaju korisnicima da spajaju koncepte na razne načine i dolaze do novih spoznaja.

Kao jedan od temeljnih elemenata semantičke tehnologije, ontologije su dio skupa standarda W3C [10] za semantički web. One pružaju korisnicima potrebnu strukturu za povezivanje informacija na webu u povezane podatke. Budući da se koriste za specificiranje zajedničkih modela predstavljanja podataka iz distribuiranih i heterogenih sustava i baza podataka, ontologije omogućuju interoperabilnost baza podataka, pretraživanje između baza podataka i učinkovito upravljanje znanjem.

U posljednjih nekoliko godina, sve se više ontologija opisuje formaliziranim jezicima, poput OWL (engl. *Web Ontology Language*), za izražavanje ontologija. OWL je jezik zasnovan na računalnoj logici za semantički web, osmišljen za predstavljanje bogatog i složenog znanja o stvarima i njihovim odnosima. Također, pruža detaljne, dosljedne i smislene razlike između klasa, svojstava i odnosa. Definiranjem kako klasa objekata, tako i svojstava odnosa te njihovog hijerarhijskog poretka, OWL obogaćuje modeliranje ontologije u semantičkim bazama podataka, poznatim kao RDF spremišta [11]. U takvim spremištima, OWL omogućuje provjere dosljednosti (kako bi se pronašle logičke nedosljednosti) i provjere zadovoljivosti (kako bi se utvrdilo postoje li klase koje ne mogu imati instance). OWL je opremljen sredstvima za definiranje ekvivalencije i razlika između instanci, klasa i svojstava. Ovi odnosi pomažu korisnicima u povezivanju koncepata, čak i ako različiti izvori podataka opisuju te koncepte na različite načine. Također osiguravaju razjašnjavanje razlika između različitih instanci koje dijele ista imena ili opise.

5.1.1. Benefiti i ograničenja ontologija

Jedna od glavnih značajki ontologija je da, s ugrađenim ključnim odnosima između koncepata, omogućuju automatizirano rasuđivanje o podacima. Takvo je rasuđivanje lako implementirati u semantičkim graf bazama podataka koje koriste ontologije kao svoje semantičke sheme. Ontologije funkcioniraju poput „mozga“. One „rade i rasuđuju“ s konceptima i odnosima na način koji je blizak načinu na koji ljudi percipiraju međusobno povezane koncepte. Osim značajke rasuđivanja, ontologije pružaju koherentniju i lakšu navigaciju dok korisnici prelaze s jednog koncepta na drugi u strukturi ontologije.

Još jedna vrijedna značajka je da se ontologije lako proširuju jer je lako dodati nove odnose i usklađivanje koncepata u postojeće ontologije. Kao rezultat toga, ovaj model se

razvija s rastom podataka bez utjecaja na ovisne procese i sustave ako nešto pođe po zlu ili treba biti promijenjeno. Ontologije također pružaju sredstva za predstavljanje bilo kojeg formata podataka, uključujući nestrukturirane, polustrukturirane ili strukturirane podatke, omogućujući jednostavniju integraciju podataka, lakše rudarenje koncepata i teksta, te analitiku vođenu podacima [9].

Dok ontologije pružaju bogat skup alata za modeliranje podataka, njihova upotrebljivost dolazi s određenim ograničenjima. Jedno takvo ograničenje su dostupni konstruktori svojstava. Na primjer, dok najnovija verzija OWL - OWL2 pruža moćne klase, ona ima donekle ograničen skup konstruktora svojstava [9]. Ovaj problem je riješen s RDF-Star (RDF*), koji omogućuje kreiranje izjava o drugim izjavama i na taj način dodavanje metapodataka na rubove u grafu. Drugo ograničenje dolazi iz načina na koji OWL primjenjuje ograničenja. Ona služe za specificiranje kako podaci trebaju biti strukturirani i sprječavaju dodavanje podataka koji nisu u skladu s tim ograničenjima. Međutim, to nije uvijek korisno. Često bi podaci uvezeni iz novog izvora u RDF spremištu bili strukturno nekompatibilni s ograničenjima postavljenim korištenjem OWL-a. Kao posljedica toga, ti novi podaci morali bi biti modificirani prije nego što budu integrirani s onim što je već učitano u spremište.

Novija alternativa korištenju ontologija za modeliranje podataka je korištenje jezika SHACL (engl. *Shapes Constraint Language*) za validaciju RDF grafova u skladu s nizom ograničenja. Oblik specificira metapodatke o vrsti resursa – kako se koristi, kako bi se trebao koristiti i kako se mora koristiti. Kao takav, slično OWL-u, SHACL se može primijeniti za validaciju podataka. Međutim, za razliku od OWL-a, SHACL se može primijeniti za validaciju podataka koji su već dostupni u spremištu [9]. Budući da ontologije definiraju pojmove koji se koriste za opisivanje i predstavljanje područja znanja, one se koriste u mnogim aplikacijama za hvatanje odnosa i poboljšanje upravljanja znanjem. Njihova sposobnost opisivanja odnosa i visoka međusobna povezanost čine ih osnovom za modeliranje visokokvalitetnih, povezanih i koherentnih podataka.

5.2. Resource Description Framework (RDF)

RDF je standard za razmjenu podataka koji se koristi za predstavljanje međusobno povezanih podataka na visokoj razini. Svaka RDF izjava je trodijelna struktura koja se sastoji od resursa, pri čemu je svaki resurs identificiran URI-em. Predstavljanje podataka u RDF-u omogućuje lako prepoznavanje, razjašnjavanje i međusobno povezivanje informacija putem interneta. Razvijen i standardiziran od strane W3C (engl. *World Wide Web Consortium*). Iako postoje mnogi konvencionalni alati za rad s podacima i posebno za rad s odnosima između podataka, RDF je do sada najlakši za korištenje, najmoćniji i najizražajniji standard.

5.2.1. RDF spremište (RDF trojke)

Način na koji RDF povezuje dijelove podataka je putem RDF trojki (tročlanih izjava). Jednostavnim jezikom, RDF izjava iznosi činjenice, odnose i podatke povezujući resurse različitih vrsta. Uz pomoć RDF trojke, gotovo sve se može izraziti jedinstvenom strukturom koja se sastoji od tri povezana dijela podataka.

Slika 1. RDF trojka

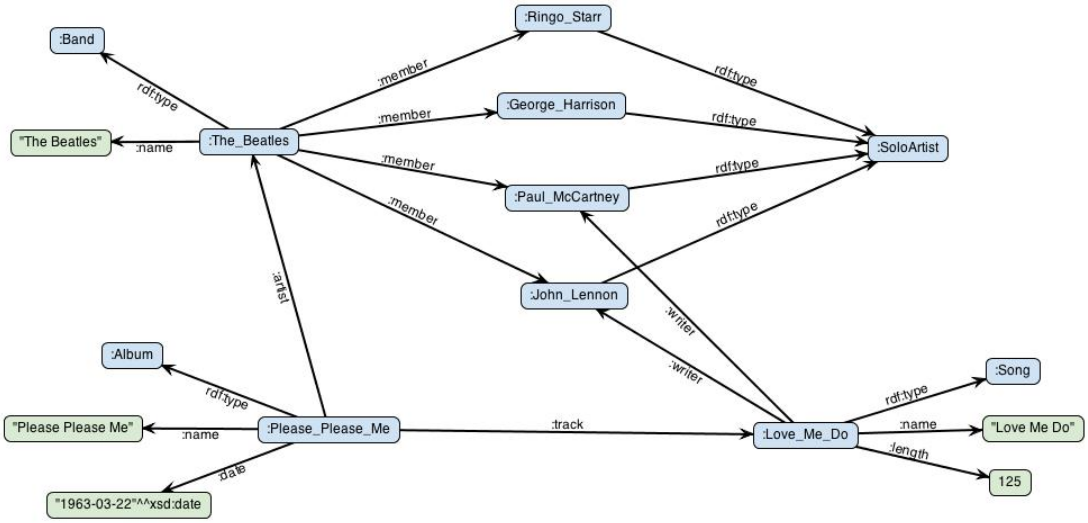
Razmotrimo jednostavniji primjer: Izjava: “Mačka je kućni ljubimac.” Ova izjava se može prikazati kao informacija uz sljedeći izjave, mačka je kućni ljubimac, mačka je životinja. To znači da je subjekt „mačka“ je identificiran kao pripadnik kategorije „kućni ljubimac“, ali je isti subjekt "mačka" identificiran kao tip „životinje“. U RDF-u, ovo bi bilo predstavljeno kao:

- <mačka> <je> <kućni ljubimac>
- <mačka> <tip> <životinja>

Ovdje je "mačka" subjekt u obje izjave, "je" i "tip" su predikati (ili odnosi), a "kućni ljubimac" i "životinja" su objekti. Ova struktura nam omogućuje jasno iznošenje odnosa između "mačke," "kućnog ljubimca" i "životinje."

RDF model utrostručuje snagu bilo kojeg dijela podataka dajući mu sredstva za ulazak u beskonačne odnose s drugim dijelovima podataka i postajući građevni blok većih, fleksibilnijih i bogato međusobno povezanih struktura podataka. Važno je znati da se svi podaci, bez obzira na njihov format, mogu pretvoriti u RDF oblik podataka. RDF se često koristi za predstavljanje podataka u formatu grafa, gdje su resursi (poput web stranica, ljudi ili događaja) čvorovi u grafu, a predikati (poput "je član" ili "je stvorio") su rubovi koji povezuju čvorove. Ovo omogućuje jednostavno predstavljanje složenih odnosa između resursa i pretraživanje podataka koristeći alate kao što je SPARQL (engl. *SPARQL Protocol and RDF Query Language*) [12]. Imenovani grafovi ili konteksti mogu se koristiti za upravljanje

komponentama u grafu (npr. prema porijeklu podataka). Svaki rub u grafu predstavlja činjenicu i može se vidjeti kao četvorka: subjekt, predikat, objekt i kontekst. Klase, predikati i imenovani grafovi svi su definirani kao URI-ji. Na ovaj način oni mogu biti prikazani kao čvorovi u grafu, dobiti svoje opise, tj. podaci o instancama i shemi mogu biti upravljani i pristupani u jedinstvenom modelu.



Slika 2. Primjer RDF grafa [13]

Na ovom primjer možemo vidjeti RDF graf o bendu Beatlesi, ako pogledamo objekt „John Lennon“, možemo vidjeti da je on tipa umjetnik („SoloArtist“), dok je njegova veza s bendom ta da je on član benda, a s pjesmom „Love me do“ da je on autor te pjesme.

5.3. Povezani podaci (Linked data)

U računarstvu, povezani podaci (engl. *Linked Data*) opisuju metodu objavljivanja i povezivanja podataka koji dolaze iz heterogenih izvora podataka koji se mogu međusobno povezivati i dijeliti. Povezani podaci temelje se na standardnim web tehnologijama kao što su HTTP i URI-jevi, ali umjesto da ih koriste za posluživanje web stranica za ljudske čitatelje, proširuju ih na način da dijele informacije koje računala mogu automatski čitati. Ovo omogućuje povezivanje i upitivanje podataka iz različitih izvora. Kao što je već ranije spomenuto, upiti nad povezanim podacima izvršavaju se korištenjem semantičkog upitnog jezika zvanog SPARQL, koji omogućuje dohvaćanje i manipulaciju podacima pohranjenim u RDF formatu. Postoje četiri jednostavna principa koja treba slijediti pri objavljivanju podataka na webu:

- Potrebno je koristiti URI-jeve kao nazive za stvari (to je jedinstveni identifikator koji smo ranije spomenuli)

- Korištenje HTTP URI-jeve kako bi korisnici mogli pogledati entitete (to znači da ID svakog entiteta treba biti dostupan putem HTTP URI-ja)
- Otvaranjem URI-ja prikazati korisne informacije, koristeći standarde (iza tih URI-jeva bi podaci trebali biti prikazani koristeći RDF)
- Potrebno je uključiti poveznice na druge URI-jeve kako bi korisnici mogli otkriti više podataka

Tim Berners-Lee predložio je sustav za objavljivanje otvorenih podataka podijeljen na 5 razina, odnosno 5 zvjezdica [15]. U nastavku su primjeri za svaki stupanj kako bi razumjeli zašto je objavljivanje 5-zvjezdanih povezanih podataka važno:

- 1 zvjezdica: sadržaj i podaci su dostupni na internetu uz otvorenu licencu
- 2 zvjezdice: sadržaj i podaci su dostupni u obliku koji je čitljiv računalima – na primjer, tablica s podacima umjesto slike tablice iz Excela
- 3 zvjezdice: podaci su dostupni kroz standardizirani format poput CSV (comma-separated values) umjesto kroz specifični format, kao Excel tablica
- 4 zvjezdice: svakom entitetu u podacima je dodijeljen jedinstveni i trajni URL kako bi se entiteti identificirali i učinili podatke lako dostupnima
- 5 zvjezdica: podaci i sadržaj su povezani s drugim podacima te im je pridružen kontekst

Jednostavno rečeno, 5-zvjezdani otvoreni skup podataka je način objavljivanja metapodataka (koristeći standard povezivanja podataka) koji ga čini čitljivim i dostupnim računalima. Zbog toga, povezivanjem teksta koji je napisao čovjek s otvorenim skupom podataka koji je stvorilo računalo, naš sadržaj postaje potpuno dostupan i prilagođen računalima. Neki od glavnih skupova podataka koji implementiraju 5-zvjezdani sustav povezanih podataka su temeljni za algoritme strojnog učenja koji stoje iza semantičkih pretraživača poput Googlea i Binga, kao i digitalnih osobnih asistenata poput Alexe, Cortane i Google Assistanta.

5.4. SPARQL

SPARQL omogućuje korisnicima pretraživanje informacija iz baza podataka ili bilo kojeg izvora podataka koji se može mapirati na RDF. SPARQL standard je dizajniran i podržan od strane W3C, pomaže korisnicima i programerima da se usmjere na ono što žele saznati, umjesto na to kako je baza podataka organizirana. Baš kao što SQL (engl. *Structured Query Language*) omogućuje korisnicima dohvaćanje i izmjenu podataka u relacijskoj bazi podataka,

SPARQL pruža istu funkcionalnost za graf baze podataka. Osim toga, SPARQL upit se može izvršiti na bilo kojoj bazi podataka koja se putem posredničkog softvera može prikazati kao RDF. Na primjer, relacijska baza podataka može se pretraživati koristeći SPARQL uz uporabu softvera za mapiranje relacijskih baza podataka u RDF.

Za razliku od SQL-a, SPARQL upiti nisu ograničeni na rad unutar jedne baze podataka: upiti mogu pristupati višestrukim pohranama podataka. Ovo je tehnički moguće jer SPARQL nije samo upitni jezik. On je također HTTP-bazirani transportni protokol, gdje se bilo kojoj SPARQL pristupnoj točci može pristupiti putem standardiziranog transportnog sloja. RDF rezultati mogu se vratiti u nekoliko formata za razmjenu podataka, a RDF entiteti identificiraju se univerzalnim identifikatorima resursa (URI). Korištenje URI-jeva za označavanje podataka omogućuje nedvosmisleno referenciranje podataka u različitim aplikacijama i zaobilazi ograničenja pretrage u pojedinačnim bazama podataka. Posljedično, mogu se razviti dodatni API-ji specifični za aplikacije koji se odnose na te podatke.

Dizajniran je na način koji omogućava kreiranje upita nad distribuiranim izvorima na neujednačenim podacima i da omogući Povezane podatke (Linked Data) za implementaciju semantičkog weba. Njegov cilj je obogatiti podatke povezivanjem s drugim globalnim semantičkim resursima, čime se omogućuje dijeljenje, spajanje i ponovno korištenje podataka na smisleniji način. Kao rezultat toga, snaga SPARQL-a uz fleksibilnost RDF-a može smanjiti troškove razvoja olakšavajući spajanje rezultata iz više izvora podataka.

SPARQL upit se sastoji od skupa trojki u kojima svaki element (subjekt, predikat i objekt) može biti varijabla. Rješenja za varijable pronalaze se uspoređivanjem obrazaca u upitu s trojkama u skupu podataka.

Postoje četiri vrste upita. Može se koristiti za:

- ASK: provjeru postoji li barem jedno podudaranje uzorka upita u RDF grafu podataka
- SELECT: dohvaćanje svih ili nekih od tih podudaranja u tabličnom obliku (uključujući agregaciju i straničenje)
- CONSTRUCT: izgradnju RDF grafa zamjenom varijabli u tim podudaranjima u skupu trojki
- DESCRIBE: opisivanje pronađenih podudaranja konstruiranjem relevantnog RDF grafa

Vodeće semantičke graf baze podataka koje podržavaju SPARQL imaju intuitivne SPARQL uređivače s funkcijama kao što su automatsko dovršavanje i mnoge druge značajke koje olakšavaju izgradnju moćnih SPARQL upita.

Najveća snaga SPARQL-a je u navigaciji odnosa u RDF grafu podataka putem podudaranja upita u grafu. U ovom procesu, jednostavni upiti mogu se kombinirati u složenije, koji istražuju detaljnije odnose u podacima. Odnosi se mogu istraživati korištenjem osnovnih upita, spajanjem upita, unijama, dodavanjem neobaveznih upita koji mogu proširiti informacije o pronađenim rješenjima. Osim toga, staze svojstava omogućuju sekvencijalnu kompoziciju, paralelnu kompoziciju, iteracije, inverziju. Kao što je već spomenuto, osnovni upit sastoji se od trojki u kojima svaki element (subjekt, predikat i objekt) može biti varijabla. Upit „Ivan“ (subjekt) -> „ima sina“ (predikat) -> X (objekt) kao rješenje će imati trojke u RDF grafu koje se podudaraju sa subjektom i predikatom, i objekt koji im pripada.

Slika 3. Primjer upita „Ivan – ima sina – X“

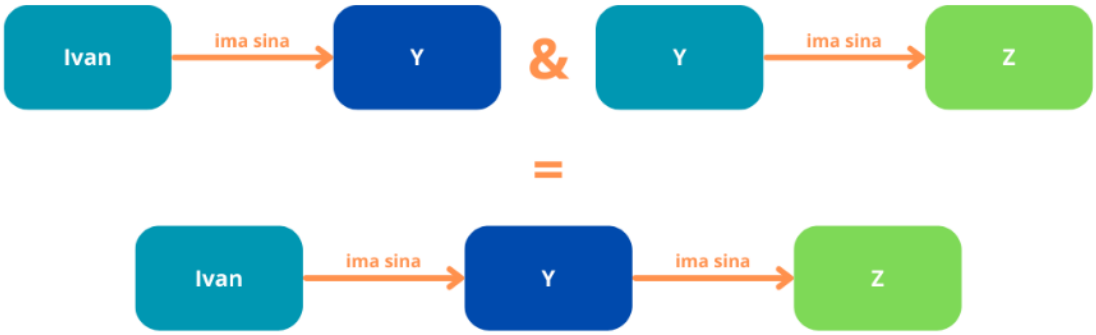
Dakle, ako Ivan ima dva sina – Josipa i Petra, trojke „Ivan“ -> „ima sina“ -> „Josip“ i „Ivan“ -> „ima sina“ -> „Petar“ će biti rezultati SPARQL upita.

SPARQL upit također može izraziti uniju upita. Svako rješenje barem jednog upita predstavlja rješenje unije. Na primjer, unija upita „Ivan“ -> „ima sina“ -> X i „Ivan“ -> „ima kćer“ -> X kao rješenja će imati sve Ivanove sinove i sve Ivanove kćeri. Takva unija prikazana je na sljedećoj slici.

Slika 4. Unija upita

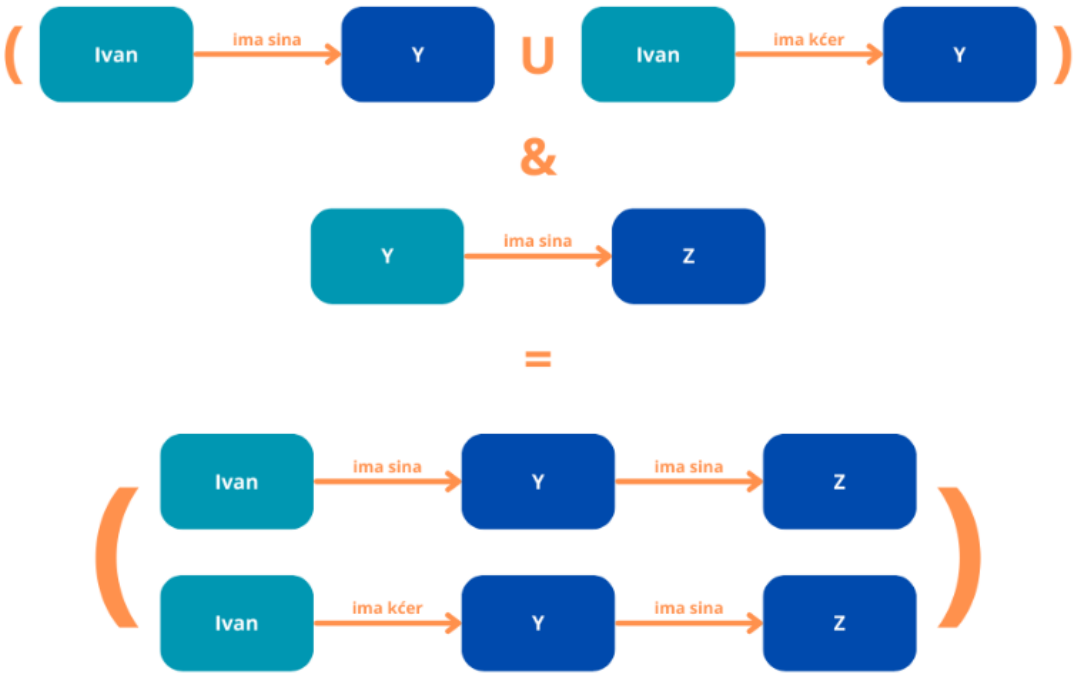
Osim unije, upiti se mogu spajati i u grupe. Grupni upit na grafu predstavlja spajanje dvaju (ili više) osnovnih upita. Za razliku od unije, grupni upit zahtijeva da se svi uvjeti zadovoljavaju. Dakle, spajanjem upita „Ivan“ -> „ima sina“ -> Y i Y -> „ima sina“ -> Z, kao

podudarajuća rješenja imat ćemo sinove Ivanovih sinova. Sinovi Ivanovih kćeri neće se vratiti kao rezultat upita jer taj upit nije sadržan u grupi upita kao što je sadržan u prethodnom primjeru gdje je unija upita. Takav grupni upit prikazan je na sljedećoj slici.



Slika 5. Grupa upita koja vraća sinove Ivanovih sinova

Grupa upita koja je prikazana iznad neće vratiti sinove Ivanovih kćeri kao rezultat, ali jednostavno se grupa i unija upita mogu kombinirati, te na taj način možemo dobiti svu Ivanovu unučad kao rezultat. Na sljedećoj slici prikazan je takav upit.



Slika 6. Kombinacija unije i grupe upita koja vraća svu Ivanovu unučad

SPARQL nije samo upitni jezik, već sveobuhvatan skup specifikacija. SPARQL UPDATE uključuje upite za brisanje podataka, umetanje podataka i manipulaciju grafovima. Općenito, SPARQL protokol definira kako pristupiti SPARQL pristupnim točkama i formatima rezultata te se može dodatno proširiti kako bi iskoristio jedinstvenost različitih tipova podataka. Postoji širok raspon rezultata koji se mogu pretraživati putem SPARQL upita, što odražava raznolikost podataka za čije je pretraživanje SPARQL dizajniran. Kao rezultat toga, SPARQL može učinkovito izdvojiti informacije skrivene u neujednačenim podacima pohranjenim u različitim formatima i izvorima.

6. Standardi za semantičke REST servise

Standardi za semantičke REST API-je imaju za cilj unaprijediti tradicionalnu REST arhitekturu uključivanjem tehnologija semantičkog weba, omogućujući bogatije i smislenije interakcije između klijenata i usluga. Ovi standardi olakšavaju dodavanje semantičkih metapodataka RESTful API-jima, podaci tako postaju interoperabilni, razumljivi računalima te ih je lakše pronaći kroz različite platforme. Integracijom koncepata poput povezanih podataka i RDF-a, ovi standardi omogućuju API-jima ne samo pružanje podataka već i prenošenje njihovog značenja, odnosa i konteksta, čime se otvara put za inteligentnije i automatizirane sustave. U nastavku su neki od ključnih standarda koji potiču implementaciju semantičkih REST API-ja.

6.1. SA-REST

Semantičke anotacije za REST (SA-REST) definiraju tri osnovna svojstva koja se koriste za označavanje HTML dokumenata semantičkim metapodacima. Ta svojstva, koja su dio formata *poshformat*, omogućuju procesoru da izvuče dodatne informacije iz sadržaja dokumenta. Poshformati, iako su povezani s mikroformatima, možda nisu prošli kroz jednako rigorozan proces validacije od strane zajednice. SA-REST svojstva: *domain-rel*, *sem-rel* i *sem-class* specificiraju se pomoću atributa *class* i *title* definiranih HTML specifikacijom, pri čemu se opseg anotacije određuje prema HTML elementu koji nosi tu oznaku. SA-REST ima tipa svojstava. Ovi tipovi služe samo za razlikovanje sposobnosti svojstva da ugnijezdi druga svojstva. Sva svojstva u SA-REST standardu mogu imati više vrijednosti. Prvi od ta dva tipa je blok koje može sadržavati druga (SA-REST) svojstva unutar sadržaja koji se označava dok je drugi tip svojstvo koje se primjenjuje samo na jedan element i ne bi trebalo sadržavati druga svojstva. SA-REST ima tri osnovna svojstva [16]:

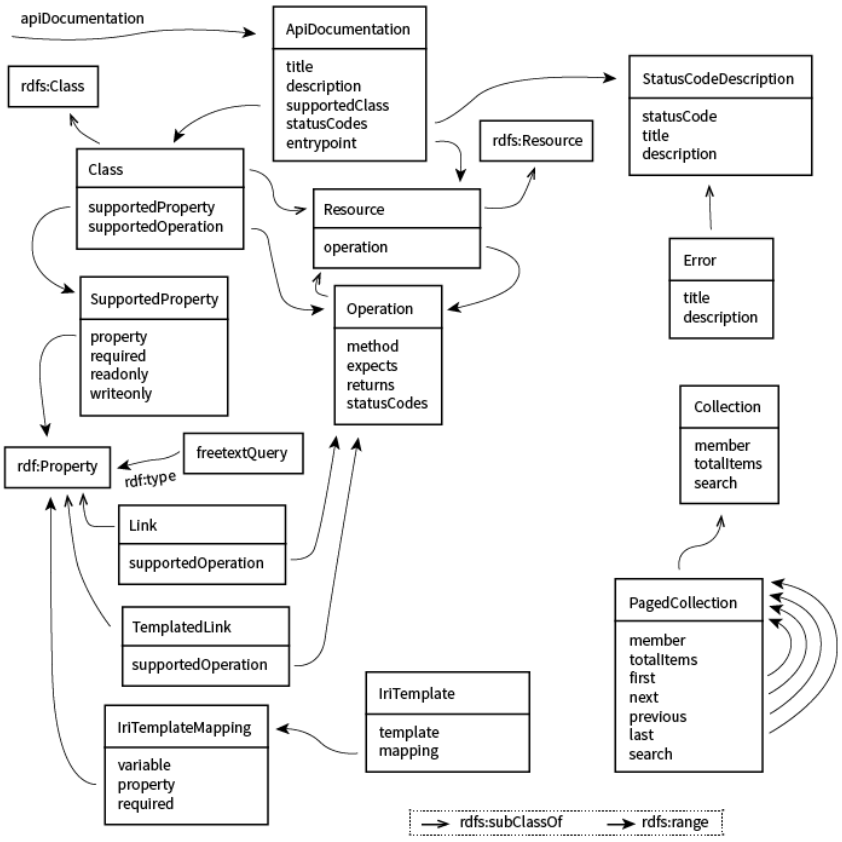
1. domain-rel: svojstvo bloka *domain-rel* omogućuje opis informacija o domeni za određeni resurs. Ako resurs (poput bloga) ima sadržaj koji obuhvaća više domena, preporučuje se dodavanje više unosa svojstva *domain-rel*, svaki za odgovarajući dio resursa. Ako takva podjela nije moguća, resurs se može povezati s enumeracijom vrijednosti kao vrijednošću svojstva *domain-rel*
2. sem-rel: svojstvo elementa *sem-rel* bilježi semantiku veze i evoluira iz popularne *rel* oznake. Svojstvo *sem-rel* omogućuje dodavanje vanjskih anotacija na dokumente trećih strana. Svojstvo *sem-rel* se može koristiti samo s elementom *anchor* (<a>)

3. sem-class: svojstvo elementa *sem-class* može se koristiti za označavanje pojedinačnog entiteta unutar resursa. Taj entitet može biti tekstualni fragment ili ugrađeni objekti kao što je video

Neke od primjena ovog standarda su aspektna pretraživanja, SA-REST poboljšava pretraživanje API-ja dodavanjem metapodataka, čime omogućuje precizniju i učinkovitiju pretragu, posebno u specifičnim domenama, kompozicije servisa poznate kao i *mashup***,** koje omogućavaju korisniku da bira pružatelja usluga za određeni servis ili poluautomatsku anotaciju teksta, zbog velikih količina teksta u znanstvenim radovima, nije moguće ručno anotirati takve količine teksta pa se SA-REST može iskoristiti za rješavanje ovog problema.

6.2. Hydra

Hydra je rječnik dizajniran za poboljšanje povezanih podataka pružanjem mehanizama za definiranje API-ja vođenih hipermedijima. Za razliku od tradicionalnog RDF-a, koji koristi IRI-je kao identifikatore bez podrške za hipermediju, Hydra omogućuje poslužiteljima da oglašavaju valjane prijelaze stanja klijentima. To omogućuje klijentima da dinamički konstruiraju HTTP zahtjeve, čineći ih prilagodljivima promjenama i odvojenima od poslužitelja.



Slika 7. Jezgra Hydra rječnika [17]

Jezgra Hydrinog riječnika je klasa *ApiDocumentation*, koja dokumentira glavnu ulaznu točku API-ja, podržane klase i svojstva, te operacije koje API omogućuje. Hydra uvodi klasu *Resource* kako bi naznačila IRI-je koji se mogu dereferencirati, razlikujući ih od IRI-ja koji se koriste isključivo kao identifikatori. Također uključuje klase *Link* i *TemplatedLink*, pri čemu *TemplatedLink* koristi IRI predloške za rukovanje vezama koje zahtijevaju parametre tijekom izvođenja. Hydrina klasa *Operation* pruža okvir za opisivanje valjanih HTTP zahtjeva za manipulaciju stanjem poslužitelja, uključujući HTTP metodu, očekivani unos i moguće statusne kodove. Operacije su povezane s klasama, vezama ili specifičnim resursima, pri čemu se ciljni IRI određuje tijekom izvođenja. Kako bi informirala klijente o podacima koji se očekuju za operacije, Hydra koristi *SupportedProperty* za definiranje svojstava povezanih s klasama, uključujući atribute poput *required*, *readonly* ili *writeonly*. Hydra također uključuje unaprijed definirane operacije za uobičajene zadatke kao što su kreiranje, brisanje i zamjena resursa, kao i rukovanje kolekcijama putem klasa *Collection* i *PagedCollection* [17]. Ove značajke čine Hydru moćnim alatom za razvoj fleksibilnih, generičkih klijenata web API-ja.

6.3. OpenAPI sa semantičkim anotacijama

Koncept OpenAPI sa semantičkim anotacijama unapređuje OpenAPI specifikaciju uvođenjem semantičkih oznaka koje pružaju računalima razumljive semantike, koje trenutno nedostaju. OpenAPI, najčešće korišten standard za opis API-ja, koristi ove semantičke anotacije kako bi omogućio bolje razumijevanje onoga što svaki resurs predstavlja. Za implementaciju ovoga, semantičke oznake se dodaju putem postojećeg opcionalnog *tags* polja u OpenAPI-ju. Svaka semantička oznaka sastoji se od prefiksa i sufiksa, odvojenih dvotočkom. Prefiks je skraćenica za identifikator (IRI) otvorenog rječnika, dok sufiks predstavlja specifičan termin u tom rječniku, usklađen s ontologijom. Ovaj format omogućuje da oznake funkcioniraju kao ekvivalent IRI-ju tog termina, što čini semantiku resursa eksplicitnom.

Takve semantičke oznake dodaju se na popis *tags* za opis resursa. IRI-ji koji odgovaraju ovim oznakama zatim se specificiraju u *url* polju objekta *externalDocs* svake oznake. Ova metoda je kompatibilna s OpenAPI specifikacijom i poboljšava ju bez kršenja standarda. U korijensku putanju dodaje se *OPTIONS* metoda kako bi se klijentima dostavila anotirana uslužna specifikacija. Ova metoda, koja se inače koristi za opisivanje opcija komunikacije za ciljani resurs u HTTP-u, sada omogućuje klijentima da preuzmu cijelu specifikaciju servisa, koja uključuje sve povezane resurse. Kada su ove anotacije dodane, semantički opisi mogu se pretvoriti u RDF. Ovaj proces generira graf resursa, omogućujući klijentima da otkriju i prepoznaju potrebne resurse unutar servisa putem semantičkog

zaključivanja. Ovaj pristup poboljšava interoperabilnost, posebno kada različiti servisi i klijenti koriste različite rječnike za opis sličnih entiteta.

6.4. JSON-LD

JSON-LD je format temeljen na JSON-u, dizajniran za serijalizaciju povezanih podataka (Linked Data), omogućujući mu jednostavnu integraciju u postojeće sustave koji već koriste JSON. Omogućuje lakši prijelaz s JSON-a na JSON-LD, čineći ga učinkovitim načinom korištenja povezanih podataka u web okruženjima, izgradnji interoperabilnih web usluga i pohrani povezanih podataka u sustavima za pohranu temeljenim na JSON-u. JSON-LD je potpuno kompatibilan s JSON-om, tako da se postojeći JSON parseri i knjižnice mogu ponovno koristiti bez izmjena. JSON-LD uvodi nekoliko značajki koje proširuju mogućnosti JSON-a, kao što su:

- Univerzalni mehanizam identifikacije za JSON objekte pomoću IRI-ja
- Način prepoznavanja ključeva u različitim JSON dokumentima mapiranjem na IRI-je pomoću konteksta
- Sposobnost referenciranja resursa na različitim web stranicama
- Mogućnost dodavanja oznaka jezika
- Način povezivanja tipova podataka s vrijednostima poput datuma i vremena
- Sposobnost izražavanja jednog ili više usmjerenih grafova, poput društvene mreže, unutar jednog dokumenta

Sintaksa je dizajnirana za korištenje kao JSON bez potrebe za poznavanjem RDF-a, iako se također može koristiti kao RDF u kombinaciji s drugim tehnologijama povezanih podataka poput SPARQL-a. Model podataka JSON-LD-a je označeni, usmjereni graf gdje čvorovi predstavljaju resurse ili podatkovne vrijednosti, povezane usmjerenim lukovima [19]. Ovi resursi mogu uključivati vrijednosti poput nizova i brojeva ili složenije podatkovne strukture. Fleksibilnost JSON-LD-a čini ga moćnim za modeliranje raznih vrsta podataka, dok njegova kompatibilnost s JSON-om osigurava jednostavno usvajanje u postojećim sustavima.

7. Semantički REST servis – implementacija

U ovom poglavlju opisat ćemo implementaciju semantičkog REST servisa koji komunicira s vanjskim API-jima za dohvat podataka o vremenskoj prognozi u određenom gradu i prirodnim događajima u njegovoj okolici, u radijusu od 1500 kilometara ili prema unosu korisnika. Servis u trenutku upita pomoću vanjskih API-ja dohvaća podatke, semantički ih obogaćuje i pohranjuje u RDF spremište, kao RDF trojke. Pohranjene podatke moguće je pretraživati i filtrirati pomoću SPARQL upitnog jezika pomoću posebne pristupne točke. Prednost ovakvih servisa je što omogućavaju povezivanje podataka iz više izvora te pretraživanje podataka pomoću posebne pristupne točke, što korisniku omogućava jednostavniji pristup podacima.

7.1. Arhitektura servisa

Servis se sastoji od tri glavne komponente, od kojih svaka igra svoju ulogu u pravilnom radu servisa, a to su semantički REST servis, vanjski API-ji i RDF spremište. Implementiran je kao Node.js servis [20] koji pokreće poslužitelj pomoću paketa *express* [21]. Za slanje HTTP zahtjeva na vanjske servise koristi se paket *axios* [22] dok se za spremanje RDF trojki koristi *rdflib.js* [23]. Paket *rdflib.js* omogućava kreiranje RDF spremišta u koji se spremaju RDF trojke kreirane iz podataka koje vraćaju vanjski servisi te postavljanje SPARQL upita nad tim spremištem.

Semantički REST servis služi za dohvaćanje podataka pomoću vanjskih servisa i odgovaranje na upite korisnika te tako predstavlja sučelje između korisnika i podataka. Vanjski API-ji koji međusobno nisu povezani vraćaju podatke o lokaciji grada, vremenskoj prognozi te prirodnim događajima u blizini na zahtjev korisnika, kasnije se ti podaci transformiraju u RDF trojke i spremaju u RDF spremište. Na posljetku, RDF spremište služi za spremanje podataka koji su dohvaćeni kroz vanjske API-ije te transformirani u RDF trojke te omogućava izvršavanje SPARQL upita nad njima za pronalazak odgovora na upite korisnika. Arhitektura osigurava slabu povezanost između komponenti i fleksibilnost u prikupljanju, integraciji i pretraživanju podataka.

Kako bi se postigle semantičke funkcionalnosti servisa, korištene su RDF i SPARQL semantičke tehnologije. RDF se koristiti kako bi se podaci dobiveni od vanjskih API-ija pretvorili u RDF trojke (subjekt, predikat, objekt) te spremili u RDF spremište za kasnije korištenje, dok

se SPARQL upitni jezik koristi na posebnoj pristupnoj točci koja prima upita za pretraživanje RDF spremišta.

7.2. Dohvat podataka – vanjski API-ji

Kao što je ranije spomenuto, semantički REST servis dohvaća podatke od tri vanjska API-ja. Vremena koja se vraćaju u podacima su u UTC (engl. *Coordinated Universal Time*) [24] vremenskoj zoni. Podaci o geolokaciji grada dohvaćaju se putem servisa OpenWeatherMap, Geocoding koji je dostupan na sljedećoj URL adresi: https://openweathermap.org/api/geocoding-api. Geokodiranje (eng. *geocoding*) je postupak pretvaranja naziva lokacije u geo koordinate, dok je obrnuti postupak, obrnuto geokodiranje (eng. *reverse geocoding*) postupak pretvaranja geo koordinata u naziv lokacije koja se nalazi na tom mjestu. [25]


```
"geolocation":{
   "name":"Buenos Aires",
   "lat":-34.6075682,
   "lon":-58.4370894,
   "country":"AR"
}
```


Slika 8. Primjer odgovora servisa za geokodiranje

Nakon dohvata podataka o lokaciji, za istu lokaciju se dohvaća vremenska prognoza za sljedećih 8 dana, putem servisa https://openweathermap.org/api/one-call-3. [26] Vremenska prognoza sastoji se od sljedećih podataka za svaki od narednih 8 dana, vrijeme izlaska i zalaska sunca, minimalna i maksimalana temperatura tog dana, tlak, relativna vlažnost zraka, brzina i smjer vjetra te sažetak o vremenu tog dana. Vremenska prognoza može se povezati s lokacijom pomoću geo koordinata. Rezultat odgovora servisa je lista podataka o vremenskoj prognozi u kojoj se nalazi prognoza za svaki od sljedećih 8 dana.

```
"forecast":{
   "lat":-34.6075682,
   "lon":-58.4370894,
   "dailys":[
      {
         "date":"2024-09-01T15:00:00.000Z",
         "sunrise":"2024-09-01T10:12:11.000Z",
         "sunset":"2024-09-01T21:35:29.000Z",
         "minTemp":8.67,
         "maxTemp":14.67,
         "summary":"Expect a day of partly cloudy with rain",
         "humidity":79,
         "pressure":1021,
         "windSpeed":8.21,
         "windDirection":"South"
      },
      "..."
   ]
}
```


Slika 9. Primjer odgovora servisa za vremensku prognozu

Uz podatke o vremenskoj prognozi, koristi se dodatan servis za dohvat prirodnih pojava u blizini te lokacije. Za te podatke koristi se NASA-in [27] servis za praćenje prirodnih pojava, EONET (engl. *Earth Observatory Natural Event Tracker*) [28] koji je dostupan na URL adresi https://eonet.gsfc.nasa.gov/docs/v3. Dohvaćaju se podaci o prirodnim pojavama koje su se dogodile u posljednjih 7 dana, svaka pojava ima svoj jedinstveni identifikator, naziv, kategoriju pojave, npr. požar ili poplava, koordinate na kojima se događa te datum. S obzirom da se prirodne pojave ne događaju točno na geo koordinatama neke lokacije ili grada, zadana vrijednost u kojem se dohvaćaju prirodne pojave za neku lokaciju je radijus od 1500 kilometara. Korisnik može povećati ili smanjiti taj radijus kod dohvata podataka. Odgovor servisa je lista prirodnih pojava koje se događaju u zadanom radijusu oko tražene lokacije.

```
"naturalEvents":[
   {
      "id":"EONET_10721",
      "title":"Wildfire in Brazil 1021889",
      "category":"wildfires",
      "lat":-11.836656934658494,
      "lon":-50.8927867944357,
      "date":"2024-08-30T19:00:00Z"
   },
   "..."
]
```

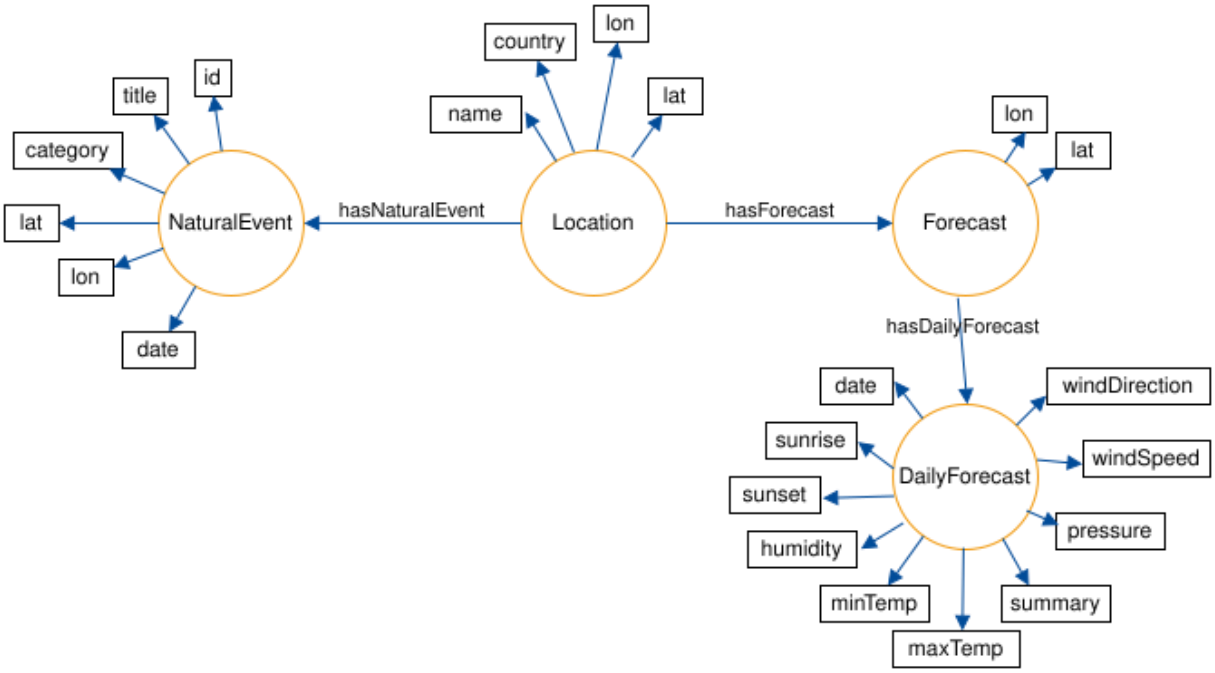

Slika 10. Primjer odgovora servisa za prirodne pojave

7.3. Semantičko obogaćivanje i pohrana podataka

Podaci prikupljeni od tri API-ja semantički su modelirani i pohranjeni kao RDF trojke. RDF omogućuje predstavljanje odnosa između točaka podataka (npr. "Lokacija ima vremensku prognozu", "Lokacija ima prirodnu pojavu u blizini"). Sustav slijedi dobro definirane ontologije kako bi se osigurala interoperabilnost i standardizirano predstavljanje podataka. Za svaki objekt definira se jedinstveni identifikator, IRI, za lokaciju je to npr. *location:N* gdje je N redni broj objekta prema redoslijedu dohvaćanja. Kod dohvata podataka o lokaciji, dohvaćaju se podaci o vremenskoj prognozi i prirodnim pojavama, to su objekti koji također imaju svoju jedinstvene identifikatore. Ti objekti povezuju se s lokacijom putem veza *hasForecast* i *hasNaturalEvent*, te se stvaraju izjave gdje je subjekt lokacija, predikat jedna od ove dvije veze, a objekt je identifikator vremenske prognoze ili prirodne pojave. U nastavku je prikazan primjer kako se spremaju podaci za neku lokaciju, npr. korisnik je tražio podatke o Mexico Cityju. Primjer izjava koje vrijede za Mexico City:

```
<location:1> <name> <Mexico City>
<location:1> <lat> <19.4326296>
<location:1> <lon> <-99.1331785>
<location:1> <country> <MX>
<location:1> <hasForecast> <dailyForecast:1>
<location:1> <hasNaturalEvent> <naturalEvent:EONET_10215>
```

U primjeru su prikazani podaci o Mexico Cityu, te kako se podaci o vremenskoj prognozi i prirodnim pojavama povezuju s lokacijom. Podaci o vremenskoj prognozi i prirodnim pojavama na sličan se način spremaju u RDF spremište. Na sljedećoj slici prikazana je ontologija u obliku grafa.



Slika 11. Ontologija servisa

Slika prikazuje ontologiju koja se sastoji od tri klase, to su Lokacija (engl. *Location*), Prognoza (engl. *Forecast*) i Prirodna pojava (engl. *Natural Event*). Svaka klasa ima set podataka koji ju opisuju, kao što su ime, zemlja za klasu Lokacija, te veze s drugim klasama u grafu, kao što je "ima prirodnu pojavu" (engl. *hasNaturalEvent*), koja je veza između klasa Lokacija i Prirodna pojava. Pomoću veza se nad podacima mogu postavljati upiti te se korisniku omogućava jednostavniji pristup znanju putem SPARQL upita.

7.4. Scenarij korištenja servisa

U ovom dijelu biti će opisano kako servis dohvaća podatke, sprema ih te kako je moguće postavljati upite koristeći SPARQL upitni jezik. Najprije je potrebno pokrenuti servis, s obzirom da se radi od Node.js servisu, pokreće se naredbom *npm start.* Nakon što je servis uspješno pokrenut, ispisati će se poruka "*Server listening http://localhost:3000*" što nam govori na kojoj je URL adresi poslužitelj dostupan. Kako bi servis dohvatio podatke o Mexico Cityju,

vremenskoj prognozi i prirodnim pojavama u radijusu od 1500 kilometara od grada, potrebno je pozvati URL adresu: "*http://localhost:3000/data?location=Mexico City*". Ukoliko želi, korisnik može dodati dodatan parametar *radius* u poziv kako bi definirao radijus u kojem želi dohvatiti prirodne pojave oko lokacije. Servis će podatke vratiti u obliku JSON-a te ih spremiti u RDF spremište u obliku RDF trojki.

```
const locationIRI = DEFAULT(`location:${locationCounter}`);

store.add(locationIRI, DEFAULT("name"), rdf.literal(result.name));
store.add(locationIRI, DEFAULT("lat"), rdf.literal(result.lat));
store.add(locationIRI, DEFAULT("lon"), rdf.literal(result.lon));
store.add(locationIRI, DEFAULT("country"), rdf.literal(result.country));
```

Slika 12. Generiranje IRI-a lokacije i spremanje podataka u RDF spremište

Isti će se postupak spremanja provesti za vremensku prognozu i prirodne pojave, samo što će se spremiti podaci koji se nalaze u odgovoru vanjskih servisa za te dvije klase. Servis će kao odgovor korisniku vratiti sve podatke koje je primio od vanjskih servisa, grupirane u podatke o lokaciji, prirodnim pojavama i vremenskoj prognozi.

```
{
  "geolocation": {
    "name": "Mexico City",
    "lat": 19.4326296,
    "lon": -99.1331785,
    "country": "MX"
  },
  "naturalEvents": [
    {
      "id": "EONET_10215",
      "title": "Hurricane Gilma",
      "category": "severeStorms",
      "lat": 14.6,
      "lon": -111.8,
      "date": "2024-08-18T18:00:00Z"
    }
  ],
  "forecast": {
    "lat": 19.4326296,
    "lon": -99.1331785,
    "dailys": [
      {
        "date": "2024-09-02T18:00:00.000Z",
        "sunrise": "2024-09-02T12:21:49.000Z",
        "sunset": "2024-09-03T00:50:41.000Z",
        "minTemp": 15.96,
        "maxTemp": 22.75,
        "summary": "Expect a day of partly cloudy with rain",
        "humidity": 68,
        "pressure": 1015,
        "windSpeed": 2.82,
        "windDirection": "East"
      },
```


Slika 13. Primjer odgovora servisa

Nakon što su podaci dohvaćeni, biti će spremljeni u RDF spremište i korisnik će ih moći pretraživati putem SPARQL upita. Upit koji može postaviti kako bi dohvatio sve RDF trojke koje su spremljene u spremište i provjerio da su se podaci ispravno spremili je sljedeći:

```
SELECT ?subject ?predicate ?object WHERE { ?subject ?predicate ?object . }
```

```
{
  "subject": "http://example.com/diplomski/v1/location:1",
  "predicate": "http://example.com/diplomski/v1/name",
  "object": "Mexico City"
},
{
  "subject": "http://example.com/diplomski/v1/location:1",
  "predicate": "http://www.opengis.net/ont/geosparql/lat",
  "object": "19.4326296"
},
{
  "subject": "http://example.com/diplomski/v1/location:1",
  "predicate": "http://www.opengis.net/ont/geosparql/lon",
  "object": "-99.1331785"
},
{
  "subject": "http://example.com/diplomski/v1/location:1",
  "predicate": "http://example.com/diplomski/v1/country",
  "object": "MX"
},
{
  "subject": "http://example.com/diplomski/v1/location:1",
  "predicate": "http://example.com/diplomski/v1/hasForecast",
  "object": "http://example.com/diplomski/v1/locationForecast:1"
},
```


Slika 14. Odgovor servisa na SPARQL upit 1

Prethodni upit vratiti će sve RDF trojke iz spremišta i ispisati ih korisniku. Možemo vidjeti kako je lokacija Mexico City spremljena pod identifikatorom *location:1* te da ima vezu *hasForecast* na vremensku prognozu pod identifikatorom *locationForecast:1*. Korisnik može postavljati i specifične upite, na primjer, kako bi dobio sve prirodne pojave koje su spremljene u RDF spremište, može izvršiti sljedeći upit:

```
PREFIX ex: <http://example.com/diplomski/v1/>
SELECT ?naturalEvent WHERE { ?location ex:hasNaturalEvent ?naturalEvent . }
```


```
[
  {
    "?naturalEvent": "http://example.com/diplomski/v1/naturalEvent:EONET_10215"
  },
  {
    "?location": "http://example.com/diplomski/v1/location:1"
  }
],
[
  {
    "?naturalEvent": "http://example.com/diplomski/v1/naturalEvent:EONET_10702"
  },
  {
    "?location": "http://example.com/diplomski/v1/location:4"
  }
]
```


Slika 15. Odgovor servisa na SPARQL upit 2

Prefiks u ovom slučaju definira domenu u kojoj su definirane veze kao što je "*hasNaturalEvent*" i olakšava pisanje upita tako da se izbjegava redundancija pisanja cijelog URL-a. Korisnik upit putem SPARQL upitnog jezika može postaviti na URL adresi "*http://localhost:3000/sparql?query=SPARQL upit*". Kao odgovor, servis će vratiti sve lokacije u cijem je radijusu neka prirodna pojava.

Odgovor servisa sadrži dvije prirodne pojave, prva je povezana s lokacijom *location:1*, dok je druga povezana s lokacijom *location:4*. Možemo zaključiti da je korisnik dohvatio dvije dodatne lokacije, 2 i 3 u čijoj blizini nije bilo nikakvih prirodnih pojava. Kako bi saznao o kojim lokacijama se radi, može pozvati SPARQL upit koji će vratiti naziv lokacije *location:1*. Upit koji će vratiti taj podatak je sljedeći:

```
SELECT ?location WHERE { <http://example.com/diplomski/v1/location:1>
<http://example.com/diplomski/v1/name> ?location . }
```

Slika 16. Odgovor servisa na SPARQL upit 3

Kako bi korisniku servis vratio podatke o vremenskoj prognozi, specifično minimalne i maksimalne vrijednosti temperature za sljedećih 8 dana, može pozvati sljedeći upit:

```
SELECT ?date ?minTemp ?maxTemp WHERE {
<http://example.com/diplomski/v1/location:1>
<http://example.com/diplomski/v1/hasForecast> ?locationForecast .
?locationForecast <http://example.com/diplomski/v1/hasDailyForecast>
?dailyForecast . ?dailyForecast <http://example.com/diplomski/v1/date>
?date ; <http://example.com/diplomski/v1/minTemp> ?minTemp ;
<http://example.com/diplomski/v1/maxTemp> ?maxTemp . } ORDER BY ?date
```

```
[
  {
    "?date": "2024-09-02T18:00:00.000Z"
  },
  {
    "?maxTemp": "23.82"
  },
  {
    "?minTemp": "15.96"
  },
  {
    "?dailyForecast": "http://example.com/diplomski/v1/dailyForecast:1"
  },
  {
    "?locationForecast": "http://example.com/diplomski/v1/locationForecast:1"
  }
],
```

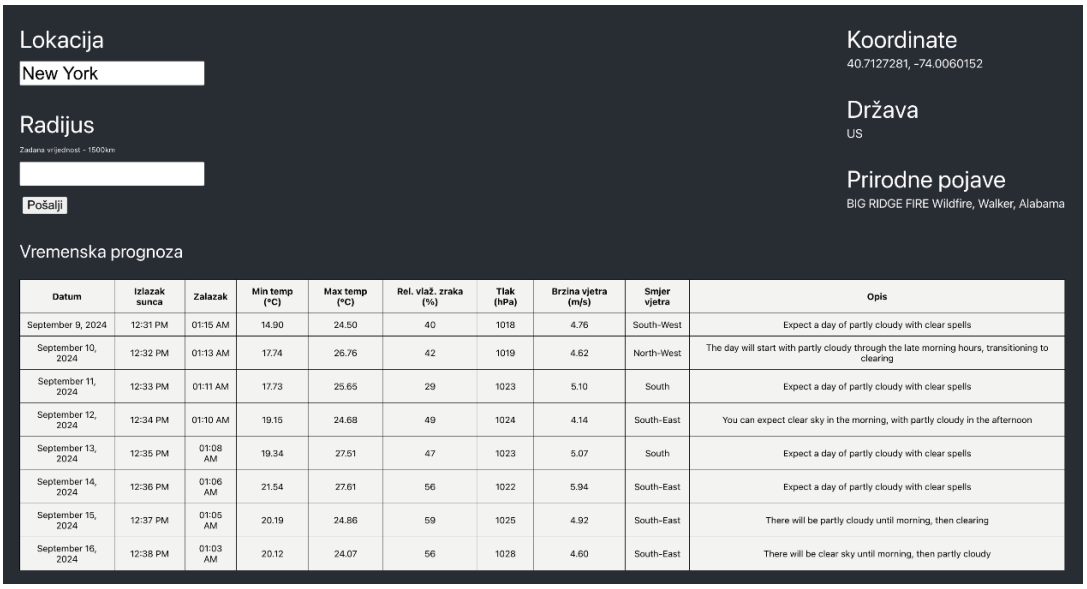

Slika 17. Odgovor servisa na SPARQL upit 4

Na ovaj način, korisniku je omogućeno da sam odredi koje podatke će i na koji način tražiti od servisa.

7.5. Web klijent

Za lakšu interakciju sa servisom, implementiran je jednostavan web klijent putem kojeg korisnik može upisati lokaciju i radijus za koji traži podatke, te nakon što su podaci dohvaćeni, može postavljati SPARQL upite nad njima. Aplikacija je implementirana pomoću *react.js* [29] paketa koji služi za razvoj korisničkih sučelja na webu. Za slanje HTTP zahtjev koristi se isti paket kao i kod semantičkog REST servisa, *axios*.

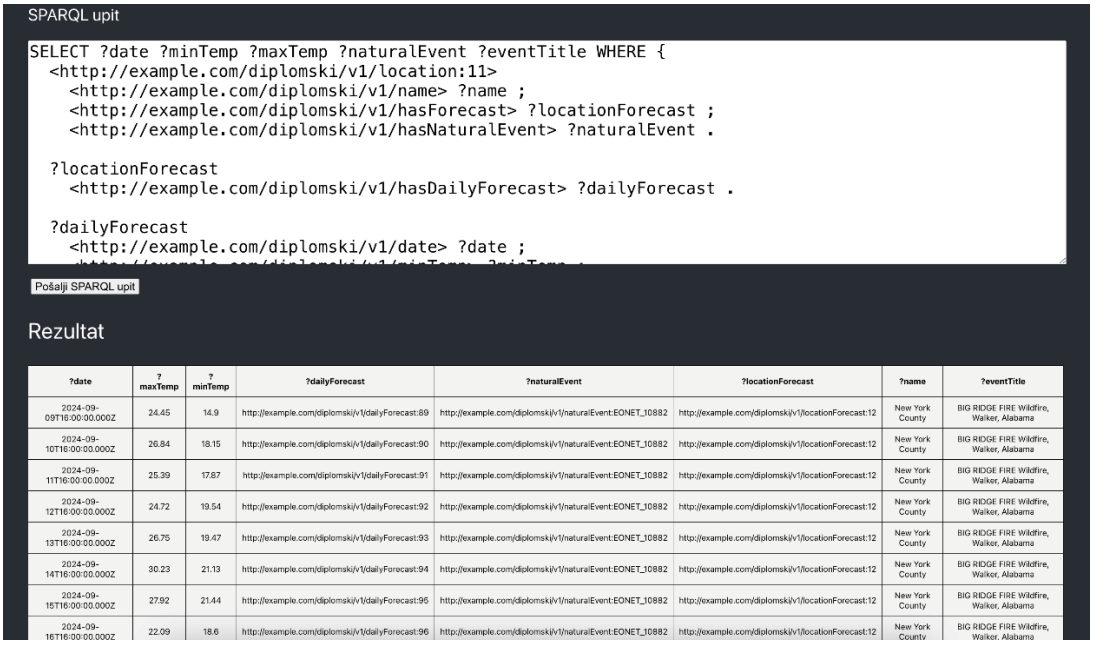
Web klijent sastoji se od polja za upis lokacije i radijusa, te gumba kojim se unos potvrđuje. Nakon potvrde, šalje se zahtjev na semantički REST servis koji dohvaća podatke od vanjskih API-ja te ih sprema u RDF spremište. Na drugom dijelu web klijenta nalazi se polje za unos SPARQL upita gdje korisnik može upisati svoj upit te poslati zahtjev na obradu semantičkom REST servisu. Nakon obrade, korisniku se dobiveni rezultati prikazuju u obliku tablice.



Datum	Izlazak sunca	Zalazak	Min temp (°C)	Max temp (°C)	Rel. vlaž. zraka (%)	Tlak (hPa)	Brzina vjetra (m/s)	Smjer vjetra	Opis
September 9, 2024	12:31 PM	01:15 AM	14.90	24.50	40	1018	4.76	South-West	Expect a day of partly cloudy with clear spells
September 10, 2024	12:32 PM	01:13 AM	17.74	26.76	42	1019	4.62	North-West	The day will start with partly cloudy through the late morning hours, transitioning to clearing
September 11, 2024	12:33 PM	01:11 AM	17.73	25.65	29	1023	5.10	South	Expect a day of partly cloudy with clear spells
September 12, 2024	12:34 PM	01:10 AM	19.15	24.68	49	1024	4.14	South-East	You can expect clear sky in the morning, with partly cloudy in the afternoon
September 13, 2024	12:35 PM	01:08 AM	19.34	27.51	47	1023	5.07	South	Expect a day of partly cloudy with clear spells
September 14, 2024	12:36 PM	01:06 AM	21.54	27.61	56	1022	5.94	South-East	Expect a day of partly cloudy with clear spells
September 15, 2024	12:37 PM	01:05 AM	20.19	24.86	59	1025	4.92	South-East	There will be partly cloudy until morning, then clearing
September 16, 2024	12:38 PM	01:03 AM	20.12	24.07	56	1028	4.60	South-East	There will be clear sky until morning, then partly cloudy

Slika 18. Sučelje web klijenta

Na slici je vidljiv prikaz rezultata za lokaciju „New York“, radijus nije unesen pa se u zahtjevu koristi zadana vrijednost od 1500 kilometara. Servis vraća koordinate tražene lokacije, kraticu države u kojoj se nalazi, prirodne pojave u traženom radijusu te vremensku prognozu za sljedećih 8 dana u obliku tablice. Možemo vidjeti da postoji jedna prirodna pojava u radijusu od 1500 kilometara od New Yorka, požar u saveznoj državi Alabami.



?date	? maxTemp	? minTemp	?dailyForecast	?naturalEvent	?locationForecast	?name	?eventTitle
2024-09-09T16:00:00.000Z	24.45	14.9	http://example.com/diplomski/v1/dailyForecast:89	http://example.com/diplomski/v1/naturalEvent:EONET_10882	http://example.com/diplomski/v1/locationForecast:12	New York County	BIG RIDGE FIRE Wildfire, Walker, Alabama
2024-09-10T16:00:00.000Z	26.84	18.15	http://example.com/diplomski/v1/dailyForecast:90	http://example.com/diplomski/v1/naturalEvent:EONET_10882	http://example.com/diplomski/v1/locationForecast:12	New York County	BIG RIDGE FIRE Wildfire, Walker, Alabama
2024-09-11T16:00:00.000Z	25.39	17.87	http://example.com/diplomski/v1/dailyForecast:91	http://example.com/diplomski/v1/naturalEvent:EONET_10882	http://example.com/diplomski/v1/locationForecast:12	New York County	BIG RIDGE FIRE Wildfire, Walker, Alabama
2024-09-12T16:00:00.000Z	24.72	19.54	http://example.com/diplomski/v1/dailyForecast:92	http://example.com/diplomski/v1/naturalEvent:EONET_10882	http://example.com/diplomski/v1/locationForecast:12	New York County	BIG RIDGE FIRE Wildfire, Walker, Alabama
2024-09-13T16:00:00.000Z	26.75	19.47	http://example.com/diplomski/v1/dailyForecast:93	http://example.com/diplomski/v1/naturalEvent:EONET_10882	http://example.com/diplomski/v1/locationForecast:12	New York County	BIG RIDGE FIRE Wildfire, Walker, Alabama
2024-09-14T16:00:00.000Z	30.23	21.13	http://example.com/diplomski/v1/dailyForecast:94	http://example.com/diplomski/v1/naturalEvent:EONET_10882	http://example.com/diplomski/v1/locationForecast:12	New York County	BIG RIDGE FIRE Wildfire, Walker, Alabama
2024-09-15T16:00:00.000Z	27.92	21.44	http://example.com/diplomski/v1/dailyForecast:95	http://example.com/diplomski/v1/naturalEvent:EONET_10882	http://example.com/diplomski/v1/locationForecast:12	New York County	BIG RIDGE FIRE Wildfire, Walker, Alabama
2024-09-16T16:00:00.000Z	22.09	18.6	http://example.com/diplomski/v1/dailyForecast:96	http://example.com/diplomski/v1/naturalEvent:EONET_10882	http://example.com/diplomski/v1/locationForecast:12	New York County	BIG RIDGE FIRE Wildfire, Walker, Alabama

Slika 19. Postavljanje SPARQL upita i rezultati

U drugom dijelu, korisnik može unijeti svoj SPARQL upit u polje za unos teksta te će se ispod tog dijela prikazati tablica sa rezultatima nakon što se zahtjev izvrši. U tablici se dinamički ispisuju pojmovi koje je korisnik tražio te pripadne vrijednosti iz RDF spremišta.

8. Zaključak

U ovom radu detaljno je prikazana važnost integracije semantičkih tehnologija u REST servise, što donosi značajno unapređenje u načinu na koji se podaci prezentiraju, koriste i međusobno povezuju na webu. Kroz analizu i implementaciju, rad je pokazao kako semantičke tehnologije poput RDF-a, OWL-a i povezanih standarda omogućavaju stvaranje bogatijih web servisa. Ovi servisi ne samo da pružaju podatke korisnicima, već ih također stavljaju u kontekst i povezuju, čime se poboljšava razumijevanje i mogućnost automatizirane obrade informacija.

Poseban naglasak stavljen je na implementaciju semantičkog REST servisa koji koristi više API-ja za dohvat podataka o lokaciji, vremenskoj prognozi i prirodnim pojavama. Podaci o geolokaciji dohvaćaju se pomoću OpenWeatherMap Geocoding API-ja, koji pretvara naziv lokacije u geografske koordinate. Nakon toga, za istu lokaciju, koristi se API za vremensku prognozu koji pruža informacije o vremenskim uvjetima za narednih osam dana, uključujući podatke o temperaturi, vjetru i atmosferskim prilikama. Na posljetku, integriran je NASA-in EONET API za dohvat podataka o prirodnim događajima, kao što su poplave i požari, unutar zadanog radijusa oko odabrane lokacije.

Svi prikupljeni podaci transformirani su u RDF trojke, koje omogućuju strukturiranje informacija na način da svaki podatak dobije semantički kontekst. RDF trojke su pohranjene u RDF spremište, što omogućava kasnije pretraživanje i obradu podataka. Korisnici mogu pristupiti ovim podacima i postavljati složene upite koristeći SPARQL upitni jezik. SPARQL omogućava pretraživanje po specifičnim kriterijima, kao što su vremenski uvjeti ili prirodni događaji u blizini određene lokacije, čime se korisnicima omogućava efikasno pronalaženje relevantnih informacija.

Zaključak ovog rada je da integracija semantičkih tehnologija u REST servise predstavlja ključni korak prema naprednijim web servisima. Semantičke tehnologije omogućavaju automatiziranu obradu podataka, bolje razumijevanje konteksta i efikasnije upravljanje informacijama, što je neophodno za rješavanje izazova modernog informatičkog društva. Ovaj rad demonstrira praktične koristi takvog pristupa, pokazujući kako se velike količine podataka mogu brzo i efikasno povezati, analizirati i koristiti na način koji prije nije bio moguć.

9. Popis literature

[1] L. Kazuya, “Palmer f.r. semantics a new outline,” *Semantics*, sij. 2017, Pristupljeno: 14. lipanj 2024. [Na internetu]. Dostupno na: https://www.academia.edu/42620758/Palmer_f_r_semantics_a_new_outline

[2] „History of the Web - World Wide Web Foundation“, World Wide Web Foundation - Founded by Tim Berners-Lee, inventor of the Web, the World Wide Web Foundation empowers people to bring about positive change. Pristupljeno: 15. lipanj 2024. [Na internetu]. Dostupno na: https://webfoundation.org/about/vision/history-of-the-web/

[3] C. Little, „An Introduction to Web Development Technologies | Tiller Digital“, Tiller. Pristupljeno: 18. lipanj 2024. [Na internetu]. Dostupno na: https://tillerdigital.com/blog/an-introduction-to-web-development-technologies/

[4] „SOAP in Web Services | Ramotion Branding Agency“, Web Design, UI/UX, Branding, and App Development Blog. Pristupljeno: 20. lipanj 2024. [Na internetu]. Dostupno na: https://www.ramotion.com/blog/soap-in-web-services/

[5] T. P. Team, „What Is a REST API? Examples, Uses & Challenges“, Postman Blog. Pristupljeno: 21. lipanj 2024. [Na internetu]. Dostupno na: https://blog.postman.com/rest-api-examples/

[6] zelja@carnet.hr, „HTTP: kodovi grešaka“. Pristupljeno: 21. lipanj 2024. [Na internetu]. Dostupno na: https://sysportal.carnet.hr/node/118

[7] D. D. Gessler *i ostali*, „SSWAP: A Simple Semantic Web Architecture and Protocol for semantic web services“, *BMC Bioinformatics*, sv. 30, str. 309, ruj. 2009, doi: 10.1186/1471-2105-10-309

[8] T. Berners-Lee, J. Hendler, i O. Lassila, „The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities“, ScientificAmerican.com, svi. 2001.

[9] „What are Ontologies?“, Ontotext. Pristupljeno: 15. srpanj 2024. [Na internetu]. Dostupno na: https://www.ontotext.com/knowledgehub/fundamentals/what-are-ontologies/

[10] „W3C“, W3C. Pristupljeno: 15. srpanj 2024. [Na internetu]. Dostupno na: https://www.w3.org/

[11] Peurača, D. "Alati za izradu ontologija," Diplomski rad, Filozofski fakultet, Sveučilište Josipa Jurja Strossmayera u Osijeku, Osijek, 2021. Dostupno na: https://urn.nsk.hr/urn:nbn:hr:142:022318.

[12] „What is RDF?“, Ontotext. Pristupljeno: 19. srpanj 2024. [Na internetu]. Dostupno na: https://www.ontotext.com/knowledgehub/fundamentals/what-is-rdf/

[13] „What is SPARQL?“, Ontotext. Pristupljeno: 23. srpanj 2024. [Na internetu]. Dostupno na: https://www.ontotext.com/knowledgehub/fundamentals/what-is-sparql/

[14] „RDF Graph Data Model“, Stardog Documentation. Pristupljeno: 28. srpanj 2024. [Na internetu]. Dostupno na: https://docs.stardog.com/tutorials/rdf-graph-data-model

[15] „What is Linked Data and Why is it Important for your Website?“, WordLift Blog. Pristupljeno: 29. srpanj 2024. [Na internetu]. Dostupno na: https://wordlift.io/blog/en/entity/linked-data/

[16] „SA-REST: Semantic Annotation of Web Resources“. Pristupljeno: 01. kolovoz 2024. [Na internetu]. Dostupno na: https://www.w3.org/submissions/SA-REST/

[17] „Hydra Core Vocabulary“. Pristupljeno: 02. kolovoz 2024. [Na internetu]. Dostupno na: https://www.hydra-cg.com/spec/latest/core/

[18] C. Peng i G. Bai, „Using Tag based Semantic Annotation to Empower Client and REST Service Interaction“, predstavljeno na 3rd International Conference on Complexity, Future Information Systems and Risk, kol. 2024, str. 64–71. Pristupljeno: 12. kolovoz 2024. [Na internetu]. Dostupno na: https://doi.org/10.5220/0006682500640071

[19] „JSON-LD 1.1“. Pristupljeno: 12. kolovoz 2024. [Na internetu]. Dostupno na: https://www.w3.org/TR/json-ld/

[20] „Node.js — Run JavaScript Everywhere“. Pristupljeno: 31. kolovoz 2024. [Na internetu]. Dostupno na: https://nodejs.org/en

[21] „Express - Node.js web application framework“. Pristupljeno: 31. kolovoz 2024. [Na internetu]. Dostupno na: https://expressjs.com

[22] „Getting Started | Axios Docs“. Pristupljeno: 31. kolovoz 2024. [Na internetu]. Dostupno na: https://axios-http.com/docs/intro

[23] *linkeddata/rdflib.js*. (01. rujan 2024.). HTML. Read-Write Linked Data. Pristupljeno: 31. kolovoz 2024. [Na internetu]. Dostupno na: https://github.com/linkeddata/rdflib.js

[24] „Coordinated Universal Time“, *Wikipedia*. 15. kolovoz 2024. Pristupljeno: 02. rujan 2024. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=Coordinated_Universal_Time&oldid=1240385039

[25] „Geocoding API - OpenWeatherMap“. Pristupljeno: 02. rujan 2024. [Na internetu]. Dostupno na: https://openweathermap.org/api/geocoding-api

[26] „One Call API 3.0 - OpenWeatherMap“. Pristupljeno: 02. rujan 2024. [Na internetu]. Dostupno na: https://openweathermap.org/api/one-call-3

[27] „NASA“. Pristupljeno: 02. rujan 2024. [Na internetu]. Dostupno na: https://www.nasa.gov/

[28] „Earth Observatory Natural Event Tracker (EONET) | Welcome“. Pristupljeno: 02. rujan 2024. [Na internetu]. Dostupno na: https://eonet.gsfc.nasa.gov/

[29] „React“. Pristupljeno: 09. rujan 2024. [Na internetu]. Dostupno na: https://react.dev/

10. Popis slika