SVEUČILIŠTE U ZAGREBU, GEODETSKI FAKULTET
Zavod za kartografiju i fotogrametriju, Katedra za geoinformacije
Institute of Cartography and Photogrammetry, Chair of Geoinfo
Kačićeva 26, 10 000 Zagreb
tel.: + 385 (1) 4639 222, faks: + 385 (1) 4828 081

DIPLOMSKI RAD

Geoinformacijski servis utemeljen na AdriaTOPO podacima

Izradio:
Ivica Širjan

mentor: doc. dr. sc. Dražen Tutić

Zagreb, rujan 2012

I. Autor	
Ime i prezime:	Ivica Širjan

II. Diplomski rad	
Naslov:	Geoinformacijski servis utemeljen na AdriaTOPO podacima
Mentor:	doc. dr. sc. Dražen Tutić

III. Ocjena i obrana	
Datum zadavanja zadatka:	27. 01. 2012.
Datum obrane:	14. 09. 2012.
Sastav povjerenstva pred kojim je branjen diplomski rad:	doc. dr. sc. Dražen Tutić dr. sc. Ivka Kljajić dr. sc. Nada Vučetić

Geoinformacijski servis utemeljen na AdriaTOPO podacima

Sažetak: Ovaj rad opisuje postupak izrade geoinformacijskog web servisa pomoću testnog skupa podataka dobivenog iz *AdriaTOPO* karte. Za izradu kartografskog prikaza je korišten *GeoServer*, a za izradu korisničkog sučelja su korištene web tehnologije (*HTML* i *JavaScript*). Rezultat ovog rada je web servis koji omogućava pregledavanje i pretraživanje podataka sadržanih u karti.

Ključne riječi: web karta, web GIS, AdriaTOPO, GeoServer

Geoinformational service based on AdriaTOPO data

Abstract: This paper describes the process of creating geo-information Web service using the test data set obtained from *AdriaTOPO* map. To create a cartographic representation was used *GeoServer*, and for making user interfaces are used web technologies (*HTML* and *JavaScript*). Result of this work is a web service that allows viewing and retrieval of data contained in the map.

Keywords: web map, web GIS, AdriaTOPO, GeoServer

Sadržaj

1. Uvod

S obzirom na današnju razvijenost informatičke infrastrukture i na sveprisutan Internet pomalo je neobična činjenica da jedan vrlo popularan proizvod poput *AdriaTOPO* karte još nije zaživio i na ovom mediju. Iako spomenuta činjenica s jedne strane ima temelje u komercijalnoj isplativosti, s druge strane bi se možda baš trebala početi razvijati u tom smjeru i iskoristiti vrlo vrijednu bazu podataka za izradu specifičnih, korisnicima prilagođenih proizvoda, te se nametnuti kao vodeća u tom području.

Ovim radom nastojat će se postaviti temelje razvoja spomenutog proizvoda u pravcu Interneta. Sama izrada će se bazirati na ustupljenom testnom setu podataka *AdriaTOPO* karte, a demonstrirati će se mogućnosti njihove manipulacije korištenjem relativno širokog spektra besplatnih alata otvorenog koda.

Počevši od samih podataka i njihove pripreme, bilo u *GIS*, bilo u grafičkim alatima, doći će se do *Geoserver*-a, koji na neki način čini bazu samog rada, jer omogućuje posluživanje i stiliziranje sirovih podataka. Zatim će se demonstrirati web tehnologije, gdje su korišteni *HTML* i *CSS*. Obuhvatiti će se primjena protokola *WMS* i *WFS*, koji se koriste za razmjenu prostornih podataka. I na kraju će se izvršiti povezivanje svih tih tehnologija i funkcionalnosti pomoću *JavaScript* programskog jezika i alata proizvedenih pomoću njega (*OpenLayers* i *jQuery*).

2. *AdriaTOPO* karta

AdriaTOPO je topografska nerutabilna karta za *Garmin®* GPS uređaje i osobna računala. Proizvod je tvrtke *Navigo Sistem d.o.o .* i izrađuje se u tri inačice od čega najnaprednija verzija nosi naziv *AdriaTOPO XL*. U trenutku pisanja ovog rada objavljena je inačica *1.0* navedene karte i na njoj je detaljno prikazano područje osam država: Hrvatska, Slovenija, Bosna i Hercegovina, Crna Gora, Srbija, Kosovo, Makedonija i Albanija (URL 1).

Karta je zasnovana na izvornicima (digitaliziranim topografskim kartama) izrađenim u mjerilu 1:25000, te je konačni vizualni izgled i kartogarafika prilagođena toj razini mjerila. Sadržaj karte se konstantno nadopunjuje novim podacima većinom prikupljenih pomoću GPS uređaja.

Slika 1. Prikaz AdriaTOPO karte na računalu pomoću Garmin BaseCamp® programa

AdriaTOPO XL 1.0 sadrži visinski prikaz terena pomoću kota i slojnica sa ekvidistancom od 25m na čitavom području država, a na pojedinim područjima kako što su parkovi prirode, nacionalni parkovi, otoci i slična područja je ekvidistancija 10m i gušća. Također je uključen i digitalni model terena (*DTM*) koji omogućava 3D dojam prikaza karte, te izradu i pregled poprečnih (vertikalnih) profila trase. Dubine Jadranskog mora prikazane su izobatama koje označavaju dubine 10, 20 i 50m, te točkama koje označavaju dubine mora za neke karakteristične lokacije kao što su pličine, grebeni i sl.

Karta sadrži detaljnu cestovnu i željezničku mrežu navedenih država (približno 360 000 km prometnica), planinarske i biciklističke staze, hidrografske podatke, unutarnje vodene putove (riječni), nacionalne parkove i parkove prirode, šumska i urbana područja i dr. Označeno je približno 300 000 POI (Point Of Interest – točka od interesa) točaka koje se mogu jednostavno pretraživati, bilo da se radi o izvoru, planinskom vrhu, naselju ili nečem drugom. Pretraga POI točaka i ostalih sadržaja

može se napraviti prema imenu (ili nekoj od riječi iz imena), kategoriji točke ili udaljenosti sadržaja od trenutne lokacije.

Sadržaj karte se može pregledavati na brojnim *Garmin®* GPS navigacijskim uređajima i na osobnim računalima pomoću besplatnih programa *Garmin BaseCamp®* i *Garmin MapSource®,* a ovim radom se realizira i mogućnost pregledavanja putem Interneta, za sve uređaje koji posjeduju web preglednik i vezu na Internet.

2.1. AdriaTOPO podaci

Izvorne podatke korištene za izradu ovog rada ustupila je firma Navigo Sistem d.o.o. Podaci su spremljeni u više *ESRI shape* datoteka, a sam sadržaj je grupiran i spremljen u datoteke prema tematskim i geometrijskim obilježjima. Podjela prema geometrijskim karakteristikama je bila neizbježna zbog ograničenja samih *ESRI shape* datoteka koje dozvoljavaju isključivo spremanje podataka istog tipa geometrije (točke, linije i poligoni) u pojedinu datoteku. S obzirom da se radi o relativno velikom setu podataka koji se konstantno nadopunjuje, napravljena je njihova podjela i po tematskim cjelinama (prometnice, željeznice, POI točke, poligoni naselja…), čime su dobivene manje cjeline, odnosno datoteke, koje su praktičnije za daljnji rad zbog manjih zahtjeva prema resursima računala, a i zbog lakše raspodjele zadataka u slučajevima kada grupa ljudi radi obradu podataka.

U atributnoj tablici svih datoteka definiran je atribut *GRMN_TYPE* pomoću kojeg se definira tip svakog objekta, a samim time i njegov način prikaza na karti. Osim spomenutog, objektima su pridodani i atributi kojima se oplemenjuje sadržaj same karte, pa su tako npr. POI točkama koje predstavljaju škole pridodane informacije o nazivu škole, te adresa i telefonski broj. U *tablici 1* je prikazan primjer definiranja vrste i naziva objekata u atributnoj tablici, te veza između atributa *GRMN_TYPE* i vrste kartografskog znaka kojim je objekt prikazan na karti. Tablica sa popisom svih tipova korištenih objekata i oznaka na karti nalazi se u prilogu na priloženom optičkom mediju.

ESRIshape datoteka	**GRMN_TYPE**	**NAME_HR**	**Kartografski znak**
poi_hrvatska	SCHOOL	Škola	
	MARINA	Marina Poreč	
	PARKING	Parkiralište	
	…	…	…

Tablica 1. Struktura AdriaTOPO podataka

2.2. Garmin® kartografski znakovi

Budući da je glavna ideja ovog rada izraditi kartu što sličniju originalnoj *AdriaTOPO* karti, kako bi izbjegli mogućnost zbunjivanja postojećih korisnika karte, neophodna je bila upotreba originalnih kartografskih znakova. Navedeni znakovi su zaštićeni licencom, no vlasnik licence (*Garmin®*) je, nakon upita za dozvolu korištenja, odobrio njihovu upotrebu za potrebe ovog rada.

Ti kartografski znakovi su zapravo malene sličice spremljene u *PNG*[1] formatu, pomoću kojih se označavaju svi točkasti objekti na karti, a to su uglavnom POI točke. Samim pregledom tih znakova možemo uočiti nekoliko nedostataka. Prvo što se na njima može primijetiti je da nemaju definirane standardne dimenzije, pa se tako njihova veličina kreće u rasponu od 13x13 do 24x24 piksela. To je na neki način nepovoljno iz razloga što prilikom stiliziranja karte za svaki znak moramo provjeravati njegove originalne dimenzije i točno ih definirati u stilu, kako izbjegli njihovo zamućivanje na karti. Drugi je njihov nedostatak u samome *PNG* formatu i njihovim fiksnim veličinama koje nas na neki način ograničavaju, jer na karti ne možemo koristiti znakove niti manje niti veće od originalnih dimenzija. Iako su znakovi izrađeni za veliki broj objekata koje najčešće pregledavamo na karti, poput banaka, trgovina, restorana, benzinskih postaja, autobusnih i željezničkih kolodvora i sl., primijećeno je da nedostaju zasebni znakovi za kategorije poput vatrogasaca, policijskih postaja i pojedinih znamenitosti, koje su prikazane istim znakom (plavi kvadrat sa popratnim tekstualnim opisom točke) što je nepovoljno, jer se na taj način smanjuje preglednost karte i kvalitetno interpretiranje sadržaja, te bi bilo poželjno za navedene kategorije napraviti nove znakove.

Iz ovih nedostataka se može zaključiti da bi bolja preporuka za kartografske znakove bila korištenje znakova u vektorskom obliku, npr. *SVG*[2] koji omogućuje promjenu veličine znakova bez gubitka kvalitete.

3. Korištene tehnologije

Prilikom izrade geoinformacijskog web servisa utemeljenog na *AdriaTOPO* podacima korištene su različite tehnologije otvorenog koda. Primjerice, sučelje servisa je izrađeno pomoću *HTML* jezika, karta je stilizirana pomoću *XML* jezika, a interakcija korisnika sa kartom je izrađena pomoću programskog jezika JavaScript. Osim navedenih korištene su i druge tehnologije, koje su opisane u narednim poglavljima.

[1] PNG (Portable Network Graphics) - rasterski oblik zapisa slike koji podržava sažimanje bez gubitka kvalitete podataka
[2] SVG (Scalable Vector Graphics) - oblik za prikazivanje dvodimenzionalne vektorske grafike, bilo statične ili animirane.

3.1. HTML

HTML je kratica za *HyperText Markup Language*, što predstavlja jezik za strukturiranje i prezentiranje sadržaja na Internetu. Prikaz hipertekst dokumenta omogućuje web preglednik. Dakle, pomoću *HTML* jezika možemo definirati logičku i fizičku strukturu elemenata unutar *HTML* dokumenta. Osim toga možemo stilski, odnosno grafički urediti te elemente, čime internetskom pregledniku dajemo do znanja kako će *HTML* dokument izgledati prilikom prikaza. Uz tekst, *HTML* omogućava uključivanje poveznica i multimedijalnih sadržaja (slikovni, zvučni, video i sl.) u dokumente, čime se postiže njihova interaktivnost.

U ovom radu *HTML* je korišten za izradu korisničkog sučelja, što obuhvaća definiranje načina prikaza i rasporeda elemenata na stranici.

3.2. XML i SLD

XML je kratica za *EXtensible Markup Language,* što predstavlja proširivi jezik za označavanje podataka. Standardizirani je jezik i za njegovu se standardizaciju brine organizacija *World Wide Web Consortium* (poznatija kao *W3C*). Stvoren je s idejom da bude jezik koji će biti jednostavno čitljiv i ljudima i računalnim programima. Princip izrade XML dokumenata je jednostavan jer se odgovarajući sadržaj treba samo uokviriti odgovarajućim oznakama (*tag*-ovima) koje ga opisuju i imaju poznato, ili lako shvatljivo značenje. Sam format tih oznaka je vrlo sličan formatu oznaka u npr. *HTML* jeziku. Iako je po principu pisanja sličan *HTML*-u, razlikuje se u samoj funkcionalnosti, tj. tome što se ne koristi za prezentaciju podataka nego radi upravo suprotno, odvaja podatke od prezentacije, što je pogodno za razmjenu podataka, pohranu podataka, povećavanje dostupnosti podataka i izradu novih specijaliziranih jezika za označavanje (URL 2). Jedan od takvih specijaliziranih jezika za označavanje je i *SLD*, koji je korišten definiranje stilova prikaza prostornih podataka pomoću *Geoservera*.

SLD (*Styled Layer Descriptor*) je opisni jezik za definiranje parametara vizualizacije. To je zapravo standardizirana *XML* shema, specificirana od strane *OGC*-a (*Open Geospatial Consortium*), kojom se definira način prikazivanja slojeva na krati (URL 3). U mogućnosti je prikazivati vektorske i rasterske podatke, a njegova uobičajena primjena je da upućuje *WMS* na koji način da prikazuje pojedine slojeve. Rad sa *SLD*-om podržavaju brojne desktop (*ArcGis, Quantum GIS* …) i serverske (*Geoserver, Mapserver* …) aplikacije koje rade sa prostornim podacima.

Same mogućnosti *SLD*-a i način definiranja stilova biti će objašnjene na konkretnim primjerima u poglavlju 5.3.

3.3. JavaScript

JavaScript je skriptni programski jezik, koji se izvršava u web pregledniku na strani korisnika (eng. *client-side scripting programming language*) i u prvom redu je namijenjen razvoju interaktivnih HTML stranica. Jezgra JavaScripta je integrirana u većini današnjih internetskih preglednika, a sam JavaScript kod se obično ugrađuje direktno u HTML stranice čime se omogućuje izvršavanje određenih radnji u inače statičnim HTML dokumentima, npr. interakcija s korisnikom, promjena svojstava prozora web preglednika, dinamičko stvaranje HTML sadržaja i dr. (URL 4).

U ovom radu je korišten za izradu funkcija koje se nalaze unutar korisničkog sučelja karte, a omogućavaju pretraživanje i filtriranje sadržaja karte i sl.

3.4. Web Map Service

WMS (eng. *Web Map Service* - mrežni kartografski servis) je protokol izrađen od strane OGC-a koji služi za slanje i primanje georeferenciranih slika preko Interneta. Navedene slike se generiraju na serveru pomoću podataka (vektorskih i rasterskih) dohvaćenih iz prostornih baza podataka ili drugih web servisa. Upotreba WMS protokola je veoma raširena, te je implementiran u brojne programske pakete, a jedan od njih je i GeoServer koji je korišten za izradu ovog rada (URL 5).

Konkretna svrha WMS protokola u ovom radu je prikazivanje karte u rasterskom obliku u internetskom pregledniku.

3.5. Web Feature Service

WFS (eng. *Web Feature Service*) je još jedan protokol izrađen od strane OGC-a koji omogućuje dohvat prostornih podataka preko Interneta. Za razliku od WMS protokola koji kao rezultat vraća podatke u rasterskom obliku, WFS nam rezultate prikazuje u vektorskom obliku. Ti rezultati su zapravo sirovi podaci karte, tj. geometrija sa pripadajućim atributima, što nam omogućava izvođenje puno bogatijih prostornih analiza i upita na samoj karti (URL 6).

Primjerice u ovom radu je taj protokol iskorišten za filtriranje i isticanje sadržaja na karti.

3.6. ESRI shapefile

ESRI shapefile ili skraćeno samo *shape* je popularni podatkovni format za spremanje prostornih podataka u vektorskom obliku, čija svrha je upotreba u geoinformacijskim sustavima. Razvila ga je tvrtka ESRI kao otvoreni podatkovni format koji bi omogućavao interoperabilnost između ESRI-jevih i

drugih programskih paketa koji rukuju prostornim podacima. Svaka *shape* datoteka se zapravo sastoji od više datoteka, od čega su tri neophodne:

.*shp* – spremanje geometrijskih podataka

.*shx* – sprema položajne pokazivače, odnosno indekse geometrije, kako bi se omogućilo brže i lakše pretraživanje atributa

.*dbf* – datoteka koja sadrži atributne podatke

Osim navedenih obaveznih datoteka po potrebi još možemo uključiti i neke dodatne, kao što je najčešće .*prj* datoteka, u kojoj su sadržani podaci o referentnom koordinatnom sustavu i projekciji spremljenih podataka. Geometrijski oblici se opisuju pomoću točaka, linija i poligona, a svaki komad geometrije se može opisati pomoću atributa, koji se spremaju u bazu podataka, odnosno u .*dbf* datoteku (URL 7).

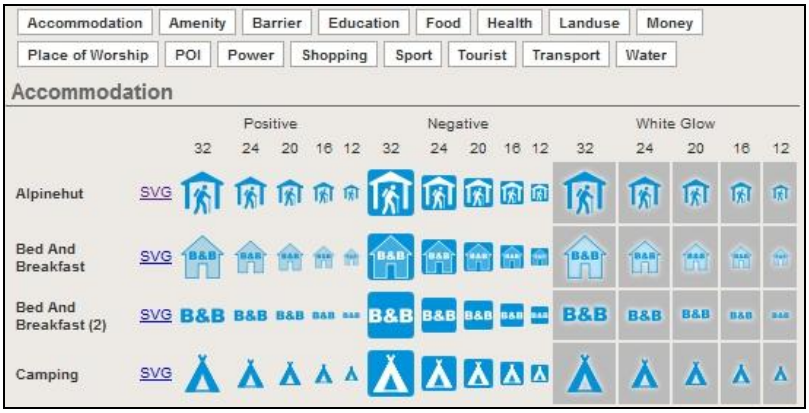
3.7. Zbirke kartografskih znakova

S obzirom na nastalu neizvjesnost oko dobivanja dopuštenja za slobodno korištenje *Garmin®* kartografskih znakova, razmatrana je mogućnost korištenja neke od alternativnih zbirka kartografskih znakova. Nakon opsežnog pretraživanja Interneta zapažene su dvije kolekcije kartografskih znakova koje bi bile pogodne za izradu ovog, a i većine drugih radova vezanih uz kartografiju, te će biti ukratko opisane u narednim poglavljima.

3.7.1. SJJB SVG Map Icons

Ova galerija se nalazi na lokaciji *http://www.sjjb.co.uk/mapicons* i sadrži veliki skup vektorskih ikonica koje su pogodne za kartografsku upotrebu. Naime cijeli set je izrađen s namjerom, a danas se i koristi, za označavanje objekata na *Open Street Map* karti. Upotreba znakova je besplatna i nije zaštićena autorskim pravima, no jedino u slučaju izrade većih projekata autori ipak traže da se barem navede njihova zasluga za navedene znakove te napravi poveznica na izvornu stranicu kartografskih znakova.

Sama galerija je koncipirana tako da su kartografski znakovi raspoređeni u 16 velikih tematskih grupa (npr. smještaj, prehrana, zdravstvo, sport, turizam, transport i dr.), a svaka grupa sadrži po desetak različitih vrsta objekata za koje su izrađene oznake. Svaka oznaka je napravljena u SVG obliku, te u rasterskom PNG obliku i to u 5 različitih veličina (32, 24, 20, 16 i 12 piksela) (*slika 2*).

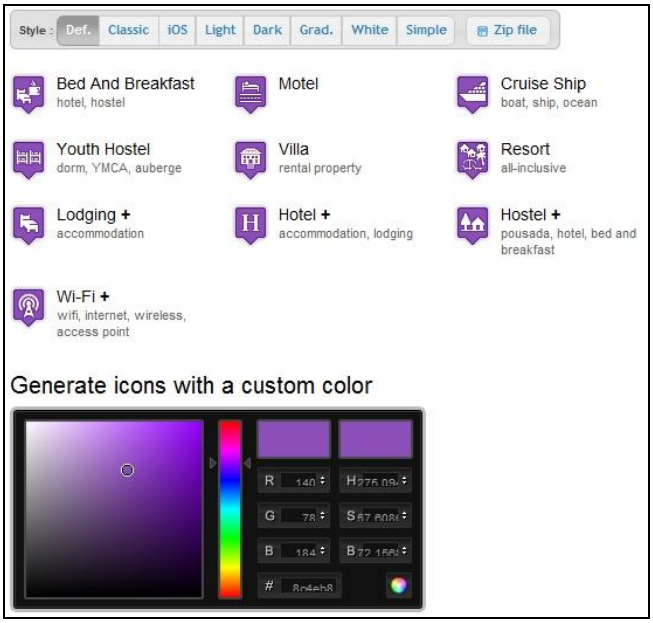


Slika 2. Primjer znakova iz galerije SJJB

Osim u različitim veličinama oznake u PNG obliku su izrađene i u tri različita stila i to kao pozitiv i negativ, te kao pozitiv sa bijelim sjenčanim obrubom. Navedeni znakovi se mogu preuzimati pojedinačno ili skupno u jednoj arhivskoj datoteci.

3.7.2. Map Icons Collection

Galerija sadrži skup više od 700 besplatnih kartografskih znakova koji su napravljeni s namjerom da se pomoću njih označavaju lokacije na *Google Maps* kartama. Njihov autor je Nicolas Mollet i dozvoljava nam upotrebu za sve svrhe, ali uz uvjet navođenja njegovih zasluga za izradu znakova i stavljanja poveznice na izvornu stranicu galerije (*http://mapicons.nicolasmollet.com*).



Slika 3. Map Icons Collection - primjer znakova

Svi znakovi su grupirani u 14 glavnih tematskih kategorija, od kojih neke imaju još i dodatne potkategorije. U njima se nalaze kartografski znakovi koji su napravljeni u 8 različitih stilova, a uz njih postoji i mogućnost izbora vlastitih boja za prikaz znakova. Nakon definiranja i odabira željenog stila

znakovi iz pojedine kategorije se mogu preuzeti skupno u jednoj arhivskoj datoteci ili svaki pojedinačno u PNG obliku.

4. Korišteni alati

Prilikom izrade ovog geoinformacijskog servisa, korišten je niz alata koji su ubrzali i omogućili izradu samog rada. Geoserver je alat koji čini njegovu osnovu, budući da služi za posluživanje i prikazivanje prostornih podataka preko Interneta, što je i bio glavni cilj. Samo korisničko sučelje je izrađeno pomoću HTML-a i JavaScript-a, pri čemu su korištene gotove JavaScript funkcije iz OpenLayers-a i jQuery-a, kako bi se sama izrada ubrzala i pojednostavila, budući da nam trebaju standardne funkcije koje se koriste na većini WEB karata. Navedeni alati su opisani u narednim poglavljima.

4.1. Geoserver

Geoserver je poslužiteljska aplikacija otvorenog koda, napisana u *Java* programskom jeziku, koja omogućava korisnicima posluživanje i uređivanje prostornih podataka (URL 8). Razvijen je kao projekt koji se distribuira pod GPL licencom (eng. *General Public License*), a razvija ga, testira, nadograđuje i dokumentira zajednica, odnosno individualni korisnici i organizacije diljem svijeta. Geoserver je funkcionalno implementacija OGC standardiziranih protokola kako što su Web Map Service (WMS), Web Feature Service (WFS) i Web Coverage Service (WCS), pomoću kojih vrši posluživanje podataka. Kompatibilan je sa brojnim tipovima podataka, pa tako ulazni podaci mogu biti u vektorskom (ESRI shape, PostGIS baze, vanjski WFS slojevi, Java Properties datoteke i dr.) i rasterskom (ArcGrid, GeoTIFF, Gtopo30, ImageMosaic, WorldImage i dr.) obliku, a moguć je i uvoz podataka iz vanjskih WMS slojeva. Izlazni podaci se mogu generirati u slikovne (*KML, JPEG, GIF, SVG, TIFF, PNG, OpenLayers, PDF i dr.*) i tekstualne formate (*AtomPub, GeoRss, GeoJSON* i *CSV*), a moguć je i izvoz podataka u GML2, GML3 i ESRI shape datoteke.

4.2. OpenLayers

OpenLayers je klijentska JavaScript biblioteka, odnosno skup gotovih programskih funkcija, otvorenog koda, namijenjena za stvaranje interaktivnih web karata i njihovo prikazivanje u web preglednicima. Navedene biblioteke omogućavaju izradu kompletnih kartografskih aplikacija, s mogućnošću prilagodbe svake komponente karte (slojeva, kontrola, događaja i dr.) prema vlastitim potrebama. Pružaju podršku za rad sa brojnim formatima poput *GeoRSS*, *KML*, *GML*, *GeoJSON* i kartografskim

podacima iz bilo kojeg poslužitelja koji koriste *OGC* standarde kao što su *WMS* i *WFS*. Jedan od takvih poslužitelja je i *Geoserver* pa se njihova simbioza nameće kao idealna u izradi ovog rada (URL 9).

4.3. jQuery

jQuery je jedna od najpopularnijih JavaScript biblioteka otvorenog koda čija je namjena pojednostavljivanje pisanja klijentskih skripti u HTM dokumentima. Sintaksa *jQuery*-a je napravljena tako da programerima olakšava navigaciju kroz dokumente, selektiranje pojedinih elemenata HTML dokumenta (tzv. DOM elementa), stvaranje animacija, upravljanje događajima, te izradu *Ajax* aplikacija koje omogućavaju komunikaciju sa serverom. Zbog svog modularnog pristupa *jQuery* biblioteke u konačnici omogućavaju stvaranje vrlo moćnih dinamičkih web stranica i aplikacija (URL 10). Sama funkcionalnost navedenih biblioteka se može proširivati brojnim dodacima koje korisnik osobno izrađuje ili implementira već postojeće, razvijene od strane drugih korisnika. Još jedna velika prednost *jQuery*-a je to što su njegove funkcije testirane i prilagođene za rad u najpopularnijim web preglednicima današnjice, što je veliko olakšanje, budući da se na taj način zaobilazi mukotrpan proces prilagođavanja aplikacija svim web preglednicima koji se danas koriste.

5. Koraci u izradi geoinformacijskog servisa

Izrada samog servisa je podijeljena u nekoliko koraka, koji će biti objašnjeni u narednim poglavljima. Početni korak je bio postavljanje *Geoserver*-a, te učitavanje dobivenog seta podataka i izrada stilova vizualizacije, čime se kreira osnovni kartografski prikaz. Nakon toga slijedi implementacija karte sa kontrolama za pregledavanje u sučelje web servisa, a posljednji korak je izrada prilagođene funkcije za pretraživanje sadržaja same karte.

5.1. Postavljanje Geoservera

Sam proces postavljanja Geoservera na računalo je relativno jednostavan. Sastoji se od preuzimanja instalacijskih datoteka sa službene stranice programa (http://geoserver.org/display/GEOS/Download) i instaliranja na računalo. Budući da je Geoserver aplikacija napravljena u *Java* programskom jeziku, prije same instalacije je potrebno instalirati i *Java Runtime Environment* (*JRE*) kako bi aplikacija mogla ispravno raditi. Ovisno od sustava na kojem radimo imamo mogućnosti odabrati koje instalacijske pakete želimo preuzeti, pa tako za slučaj *MS Windows* operativnog sustava možemo preuzeti takozvani samoinstalacijski paket, koji je nakon preuzimanja potrebno samo pokrenuti, a tokom instalacije je potrebno definirati direktorij u koji će se instalirati Geoserver, a zatim je potrebno

navesti putanju do direktorija u kojem se nalazi *Java Runtime Environment*, te definirati direktorij u kojem će se nalaziti naši prostorni podaci. Nakon toga moramo odabrati da li želimo da se Geoserver instalira na računalo kao servis koji će biti stalno aktivan ili da se instalira kao samostalna aplikacija koju ćemo pokretati ručno kada nam zatreba. Na kraju je još potrebno postaviti ime i lozinku administratora i port preko kojeg će Geoserver komunicirati. Ukoliko ostavimo pretpostavljene (eng. *default*) postavke korisničko ime administratora će biti *admin* , lozinka *geoserver* i standardni port *8080*. Kako bi se zaštitili od neovlaštenih pristupa našim podacima poželjno je postaviti neke druge vrijednosti za te postavke ili navedene promijeniti prilikom prvog pokretanja aplikacije.

Tim korakom smo završili instalaciju Geoservera i samu aplikaciju možemo pokrenuti tako da najprije, u direktoriju u kojem smo instalirali Geoserver, otvorimo poddirektorij *bin* (npr. C:\Program Files (x86)\GeoServer 2.1.3\bin) i pokrenemo izvršnu datoteku *startup.bat*. Nakon toga u našem web pregledniku upisujemo adresu: http://localhost:8080/geoserver, čime pristupamo sučelju za mrežno upravljanje Geoserverom (eng. *Web Administration Interface*). Ukoliko je sve ispravno instalirano otvara nam se sučelje za mrežnu administraciju *Geoserver*-a i kako bi mogli koristiti administratorske opcije, potrebno je prijaviti se kao administrator na *Geoserver*, na način da u rubrike *Username* i *Password* upišemo postavljenje vrijednosti. Ukoliko smo se na sustav prijavili sa pretpostavljenim vrijednostima, iste možemo izmijeniti u rubrici *Security* u izborniku *Users*. Osim izmjena podataka postojećeg korisnika, u tom izborniku možemo i dodavati nove korisnike, kao i mijenjati razinu ovlaštenja postojećim korisnicima.

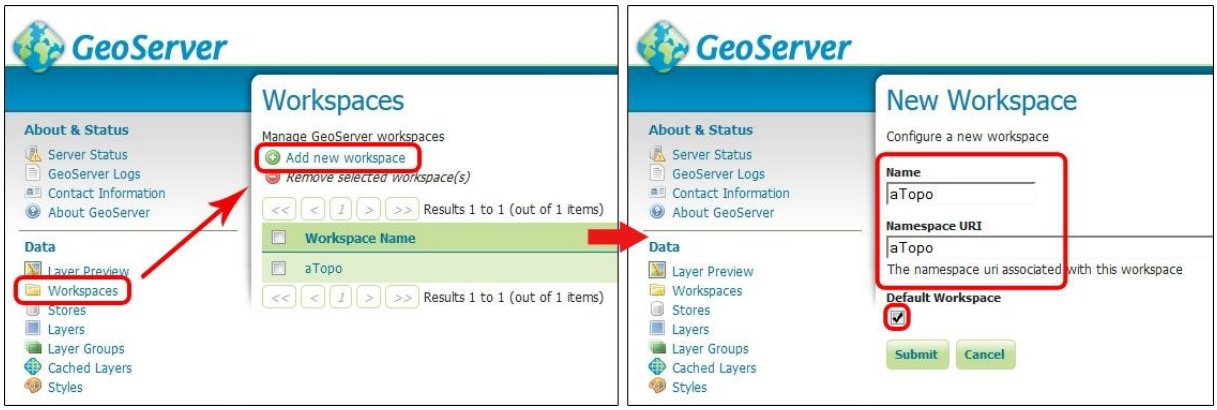
Za napomenuti je još da postoje i druge mogućnosti instalacije *Geoservera*, od kojih bih istaknuo instalaciju pomoću univerzalnog skupa izvršnih datoteka (eng. *OS Independent Binary*), koja je neovisna o vrsti operacijskog sustava kojeg koristimo, no ona neće biti opisana, budući da nije korištena prilikom izrade ovog rada.

5.2. Unos podataka u Geoserver

Izvorni podaci koji se koriste za izradu ovog Geoinformacijskog servisa su dobiveni u *ESRI Shape* datotekama. Konkretno se radi o 16 *ESRI Shape* datoteka u koje su raspoređeni podaci ovisno o vrsti geometrije (točke, linije i poligoni) i ovisno o tematskom sadržaju (prometnice, POI točke, poligoni naselja …) koji prikazuju. Referentni sustav u kojem su podaci prikupljeni i spremljeni je *WGS84*. Način interpretacije znakova (eng. *encoding*) atributnih zapisa je promijenjen u *UTF-8*, na način da su datoteke otvorene pomoću programa *Quantum GIS (1.7.3)* i ponovno spremljene pomoću funkcije *Save As*, pri čemu je prethodno postavljeni način interpretacije zamijenjen sa navedenim *UTF-8*. Taj je korak proveden s namjerom da se omogući ispravno prikazivanje hrvatskih dijakritičkih znakova.

Sam proces unošenja podataka u *Geoserver* je proveden na način da su sve datoteke kopirane u poddirektorij nazvan *istra* (jer podaci obuhvaćaju područje Istarskog poluotoka), koji je smješten na lokaciju GeoServer 2.1.3\data_dir\data\, koju je i predviđena za spremanje svih podataka koje želimo vizualizirati na karti.

Slijedeći korak je stvaranje radnog prostora (eng. *Workspace*), koji funkcionira kao svojevrsno skladište za spremanje podataka. To se provodi na način da se prijavimo u sučelje za mrežno upravljanje *Geoserver*-om, odabiremo poveznicu *Workspace*s, čime se otvara nova stranica u kojoj odabiremo opciju *Add new workspace*, gdje upisujemo naziv i *Namespace URI*[3] našeg novog radnog prostora. U konkretnom slučaju je za obadvije vrijednosti dodan naziv *aTopo*, a pomoću opcije *Submit* su navedene vrijednosti spremljene i kreiran je radni prostor (*slika 2*).



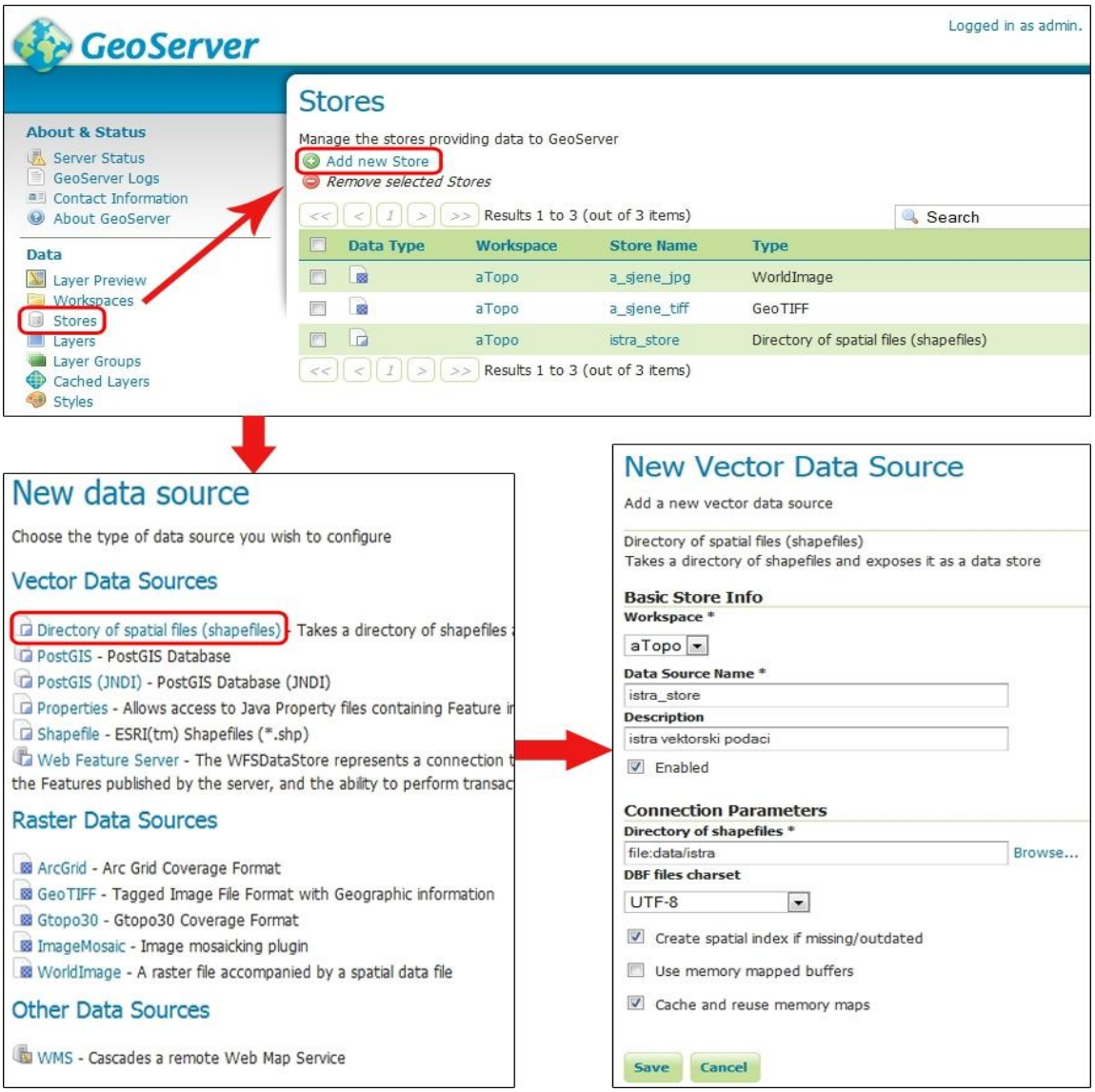
Slika 4. Stvaranje novog radnog prostora

Slijedeći korak u izradi je uvoz samih podataka u *Geoserver*, na način da se u rubrici *Data* odabere poveznica *Stores* i unutar nje *Add new Store*. Time se dolazi do izbornika u kojem je omogućen izbor tipa podataka koji se želi uvesti. S obzirom da je potrebno uvesti više *shape* datoteka najpraktičnije je odabrati opciju *Directory of spatial files (shapefiles)*, čime nam se otvara stranica sa postavkama za uvoz svih *shape* datoteka koje se nalaze u direktoriju *istra*.

Na toj stranici je potrebno odabrati radni prostor na koji će se podaci odnositi (u ovom slučaju je odabran *Workspace aTopo*), zatim se navodi naziv spremišta (*istra_store*) i eventualno njegov opis. Nakon toga se definira relativna putanja (u odnosu na lokaciju GeoServer 2.1.3\data_dir\) do mape u kojoj su spremljeni svi vektorski podaci, konkretno do mape naziva *istra* (data_dir\data\ istra\). U rubrici *DBF files charset* još je preostalo za definirati na koji će se način tumačiti znakovi atributnih zapisa iz *.dbf* datoteka. Budući da su sve datoteke spremljene sa UTF-8 *encoding*-om, isti je potrebno postaviti i u ovoj rubrici kako bi se provela ispravna interpretacija znakova. Ostale postavke ostavljamo sa pretpostavljenim vrijednostima i spremamo ih funkcijom *Save* (*slika 3*). Ukoliko je sve

[3] URI (Uniform Resource Identifier) - niz znakova za identifikaciju skupa podataka na Internetu

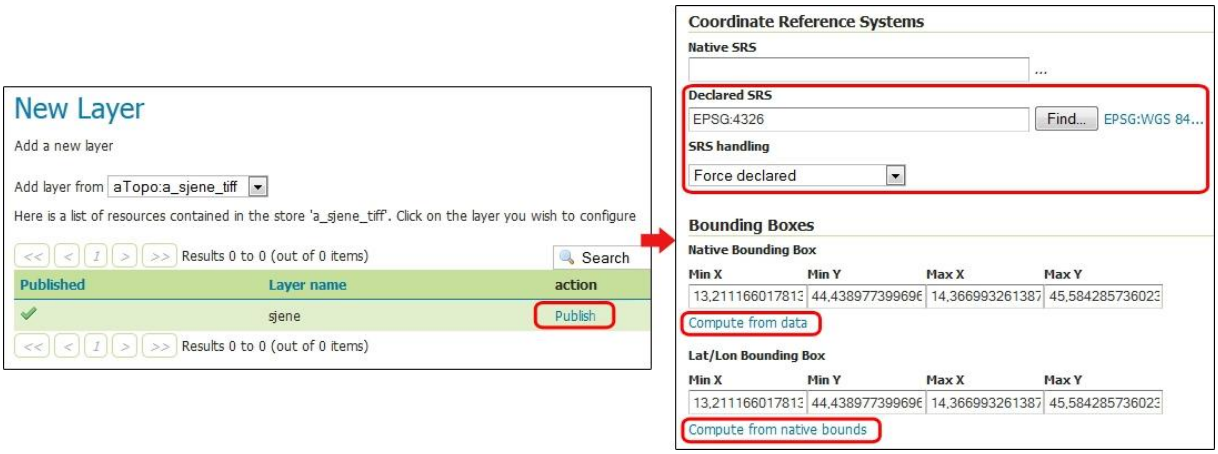
ispravno napravljeno automatski se otvara stranica sa ispisom svih vektorskih slojeva koji su iz mape *istra* uvezeni u spremište *istra_store*. Uz svaki od navedenih slojeva se tada pojavljuje opcija *Publish*, koja služi za objavljivanje slojeva.



Slika 5. Stvaranje novog spremišta podataka

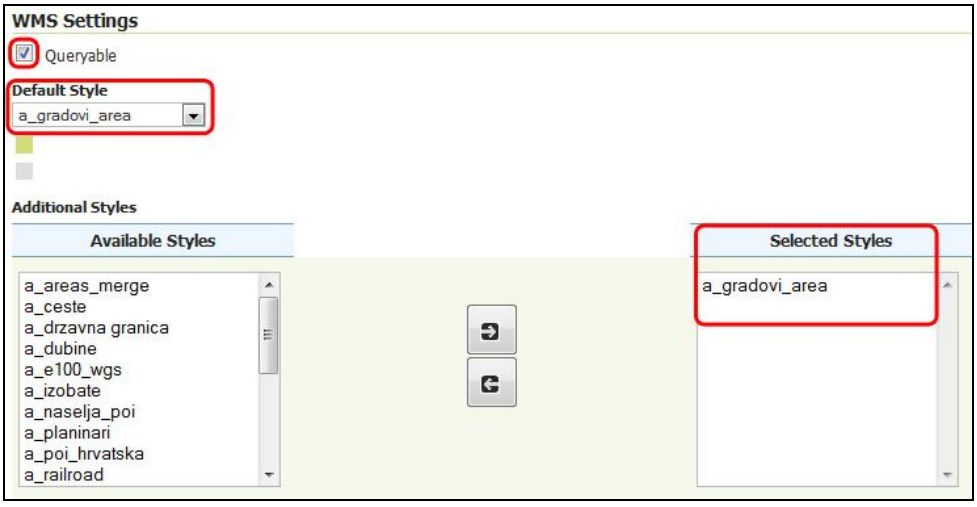
Postupak objavljivanja slojeva se provodi tako da nakon odabira poveznice *Publish* dolazimo na stranicu za definiranje postavki slojeva. U kartici *Data* se upisuje naziv svakog sloja (konkretno su zadržani nazivi s*hape* datoteka, kako ne bi došlo do zabuna tokom rada) i eventualno kratki opis, te ključne riječi i funkcije sloja. Zatim se definira referentni sustav na koji se podaci odnose. Budući da su podaci prikupljeni u *WGS84* sustavu isti je potrebno pronaći na listi, te unijeti njegov *EPSG* kod (*4326*). Nakon tog koraka potrebno je još definirati granični okvir (eng. *Bounding Box*) koji obuhvaća sve podatke sloja. To se može provesti ručnim unošenjem granica zadanog područja, ili jednostavnije odabirom opcije *Compute from data* za rubriku *Native Bounding Box*, kojom *Geoserver* automatski

određuje granice sloja. U slučaju da koristimo neku vlastitu projekciju isti postupak moramo provesti i za *Lat/Lon Bounding Box* pomoću opcije *Compute from native bounds*, kao što je prikazano na slici 4.



Slika 6. Objavljivanje novog sloja i definiranje graničnog okvira sloja podataka

Budući da ćemo podatke prikazivati pomoću *WMS* servisa potrebno je otvoriti i karticu *Publishing*, te u rubrici *WMS Settings*, pod stavkom *Default Style* sloju pridružiti odgovarajući stil koji će definirati njegov izgled na karti, kao što je prikazano na slici 5. Navedene je stilove potrebno prethodno pripremiti u *SLD* obliku, što će biti objašnjeno u slijedećem poglavlju. Nakon toga spremamo postavke i kako bi se uvjerili da su podaci slojeva ispravno uvezeni možemo izvršiti njihov prikaz pomoću opcije *Layer Preview*, koja se nalazi u *Data* izborniku. Ukoliko imamo namjeru atributne podatke iz slojeva prikazivati ih na karti potrebno je uključiti i opciju *Queryable*, kako bi se mogli vršiti upiti prema pojedinom sloju.



Slika 7. Pridruživanje stilova slojevima

5.3. Definiranje vizualnog izgleda karte

Definiranje izgleda pojedinih objekata na karti provedeno je unutar *Geoserver*-a u rubrici za definiranje stilova (*Style Editor*). Stilovi se izrađuju pomoću *SLD*-a, pri čemu je glavni cilj izraditi prikaz karte, koji će vizualno biti što sličniji prikazu karte na *Garmin®* GPS uređajima i prikazu pomoću *Garmin BaseCamp®* programa, kako ne bi došlo do zbunjivanja korisnika koji se sa zapravo istim proizvodom susreću na novom mediju. Budući da nije raspoloživa službena specifikacija prema kojoj bi se određivali prikazi pojedinih objekata na karti, definiranje stilova je provedeno tzv. *metodom pokušaja*, tj. promatran je prikaz karte u *Garmin BaseCamp®* programu i nastojana je što vjernija reprodukcija tog prikaza pomoću *SLD*-a u *Geoserver*-u. To znači da je, osim što vjernije reprodukcije prikaza boja i stilova prikaza za pojedine objekte, trebalo voditi računa i o stupnju generalizacije kako karta ne bi postala prenatrpana sadržajem, a samim time nepregledna i u krajnjem slučaju neupotrebljiva. Uzorkovanje boja je provedeno besplatnim programom *Just Color Picker*, koji nam omogućava ispisivanje heksadecimalnog koda boje odabrane sa ekrana, kojeg zatim unosimo u *SLD* stil u Geoserveru. Generalizacija prikaza je provedena na način da se na određenim razinama mjerila karte pojedini sadržaji prikazuju ili isključuju iz prikaza, odnosno da se mijenjaju dimenzije prikaza objekata na karti, kao što je to u slučaju prometnica koje su pri sitnijem mjerilu karte prikazane tankim linijama, a pri krupnijem mjerilu debljim linijama. Navedeni primjeri definiranja stilova će biti detaljnije objašnjeni u narednim poglavljima, zasebno sa svaki tip vektorskih podataka i jedan primjer za rasterski.

5.3.1. Točkasti znakovi

Točkastim tipom znakova su na ovoj karti prikazana naselja, točke od interesa (bolnice, policijske postaje, škole, poštanski uredi, parkirališta, restorani, hoteli i dr.), te lokacije specifičnih dubina mora. U nastavku je naveden primjer kako je pomoću *SLD*-a, odnosno *XML* opisnog jezika, uređen prikaz znakovi za poštanske urede na karti.

Svaki *SLD* dokument sadrži standardne elemente koji su navedeni u prvih 10 redaka primjera. Prvi red čini zaglavlje i u njemu je navedeno koja verzija *XML*-a i koji način interpretacije znakova se koristi u dokumentu. U drugom retku započinje definiranje elementa *StyledLayerDescriptor*, kojem se također navodi koja verzija se koristi, te se do sedmog retka navode lokacije na kojima se nalaze specifikacije vezane za *SLD* dokument. Osmi red sadrži standardni element *NamedLayer*, a u devetom retku unutar elementa *Name* možemo navesti naziv stila. Sa desetim retkom otvaramo element *UserStyle* nakon kojeg počinjemo definirati naš stil.

```
1  <?xml version=1.0 encoding=ISO-8859-1?>
2  <StyledLayerDescriptor version=1.0.0
```

```
3     xsi:schemaLocation=http://www.opengis.net/sld StyledLayerDescriptor.xsd
4     xmlns=http://www.opengis.net/sld
5     xmlns:ogc=http://www.opengis.net/ogc
6     xmlns:xlink=http://www.w3.org/1999/xlink
7     xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance>
8      <NamedLayer>
9        <Name>POI oznake</Name>
10        <UserStyle>
```

Elementom *FeatureTypeStyle* se navode definicije koje će odrediti izgled objekata. Taj element je potrebno definirati za sve vrste objekata (npr. škole, pošte, bolnice …) koji su sadržani u *Shape* datoteci čije podatke želimo prikazati nekim određenim znakom. U primjeru je definiran stil za poštanske urede, što je i navedeno u liniji 13, koja predstavlja naziv grupe objekata koji prikazujemo. Taj element nije neophodan, zbog činjenice da nema nikakav konkretni utjecaj na sami prikaz, nego više služi za samu orijentaciju u kodu. Element *ogc:Filter* služi za izdvajanje određene grupe objekata iz naše Shape datoteke, te konkretno definira da se navedeni stil odnosi samo na objekte koji su u atributnom stupcu *GRMN_TYPE* označeni atributom *POST_OFFICE*, što znači da će se definiranim stilom prikazivati samo poštanski uredi.

```
11         <FeatureTypeStyle>
12          <Rule>
13            <Name>POST_OFFICE</Name>
14               <ogc:Filter>
15                 <ogc:PropertyIsEqualTo>
16                    <ogc:PropertyName>GRMN_TYPE</ogc:PropertyName>
17                    <ogc:Literal>POST_OFFICE</ogc:Literal>
18                 </ogc:PropertyIsEqualTo>
19                </ogc:Filter>
```

Kako bi odredili na kojoj razini približenja (zooma), dnosno pri kojem mjerilu će nam se prikazivati pojedini objekti koristimo *MinScaleDenominator* i *MaxScaleDenominator* elemente. Unutar njih upisujemo vrijednost nazivnika mjerila, koji će određivati kada će se znakovi početi, odnosno prestati prikazivati. *MinScaleDenominator* određuje najkrupnije mjerilo do kojeg će se znakovi prikazivati, a *MaxScaleDenominator* određuje najsitnije mjerilo pri kojem će se znakovi početi prikazivati. Prema tome možemo zaključiti da će se u konkretnom slučaju znakovi početi prikazivati tek kad mjerilo prikaza bude krupnije od 1:15000.

```
20                <MaxScaleDenominator>15000</MaxScaleDenominator>
```

Prethodni elementi imaju bitnu ulogu u cijelom procesu izrade karte, jer se pomoću njih provodi proces generalizacije podataka. Stoga je potrebno odrediti prioritete, ovisno o namjeni karte, i prema njima najprije na najsitnijim mjerilima prikazivati samo objekte najveće važnosti, a nakon njih s promjenom mjerila postupno uključivati i ostale objekte, kako ne bi došlo do njihovog preklapanja, a samim time i do narušavanja funkcionalnosti i vizualnog izgleda karte.

Element *PointSymbolizer* definira da će se za prikaz objekta na karti koristi točkasti znak. U ovom slučaju je navedeno da će to biti slikovni znak, odnosno vanjski element (*ExternalGraphic*), koji je

spremljen na lokaciji garmin_ikone\post.png, unutar ishodišne mape za stilove GeoServer 2.1.3\data_dir\styles\. Budući da je navedeni znak zapravo sličica u *.png* obliku, potrebno je u elementu *Format* upisati *image/png*, kako bi je *Geoserver* mogao ispravno interpretirati. Uz to je još potrebno navesti i dimenzije slike u elementu *Size*, i to samo visinu u pikselima, a širina će ostati nepromijenjena u odnosu na originalnu sliku. Ukoliko upišemo iznos koji ne odgovara originalnoj visini, slika će biti prisilno razvučena na dimenzije koje smo zadali, što nije povoljno, jer će tada doći do njenog zamućenja.

```
21              <PointSymbolizer>
22                  <Graphic>
23                      <ExternalGraphic>
24                          <OnlineResource
25                              xlink:type=simple
26                              xlink:href=garmin_ikone\post.png/>
27                          <Format>image/png</Format>
28                      </ExternalGraphic>
29                      <Size>24</Size>
30                  </Graphic>
31              </PointSymbolizer>
```

Osim slikovnih znakova možemo koristiti i obične točkaste znakove, kojima možemo definirati oblik (krug, trokut, pravokutnik), boju, boju i debljinu obruba, rotaciju i druge elemente, što je detaljnije objašnjeno u uputama za rad sa *Geoserver*-om (URL 11).

Uz svaku točkasti znak se može prikazati i dodatni tekstualni opis, na način da se kreira element *TextSymbolizer,* u kojem definiramo da uz svaki znak želimo prikazati naziv objekta koji predstavlja, odnosno vrijednost koja je spremljena u atributnoj tablici na polju *NAME_HR*. Način prikaza teksta se definira elementom *Font*, u kojem se određuje da će tip slova biti *Arial*, veličine *11* piksela i stila *normal*.

```
32              <TextSymbolizer>
33                  <Label>
34                      <ogc:PropertyName>NAME_HR</ogc:PropertyName>
35                  </Label>
36                  <Font>
37                      <CssParameter name=font-family>Arial</CssParameter>
38                      <CssParameter name=font-size>11</CssParameter>
39                      <CssParameter name=font-style>normal</CssParameter>
40                  </Font>
```

Zatim pomoću *LabelPlacement* određujemo poziciju teksta u odnosu na ishodišne koordinate znaka (*AnchorPoint*), tako da definiramo vrijednost odmaka (*Displacement*) u pikselima, od te točke po X i Y osi.

```
41                  <LabelPlacement>
42                      <PointPlacement>
43                          <AnchorPoint>
44                              <AnchorPointX>0.0</AnchorPointX>
45                              <AnchorPointY>0.0</AnchorPointY>
46                          </AnchorPoint>
47                          <Displacement>
48                              <DisplacementX>12</DisplacementX>
49                              <DisplacementY>0</DisplacementY>
```

```
50                        </Displacement>
51                      </PointPlacement>
52                    </LabelPlacement>
```

Kako bi postigli bolju čitljivost teksta korisno je dodati *Halo* efekt, odnosno pozadinu tekstu u nekoj kontrastnoj boji. Budući da se tekst prikazuje plavom bojom, pozadinu je najpogodnije postaviti kao bijelu (heksadecimalna vrijednost *#FFFFFF*), jer pruža najugodniji kontrast.

```
53                    <Halo>
54                        <Radius>1</Radius>
55                        <Fill>
56                          <CssParameter name=fill>#FFFFFF</CssParameter>
57                        </Fill>
58                    </Halo>
```

Osim za *Halo* efekt postavlja se i boja ispune teksta i to na vrijednost *#0000F1*, što predstavlja plavu boju, a nakon toga zatvaramo element *TextSymbolizer.*

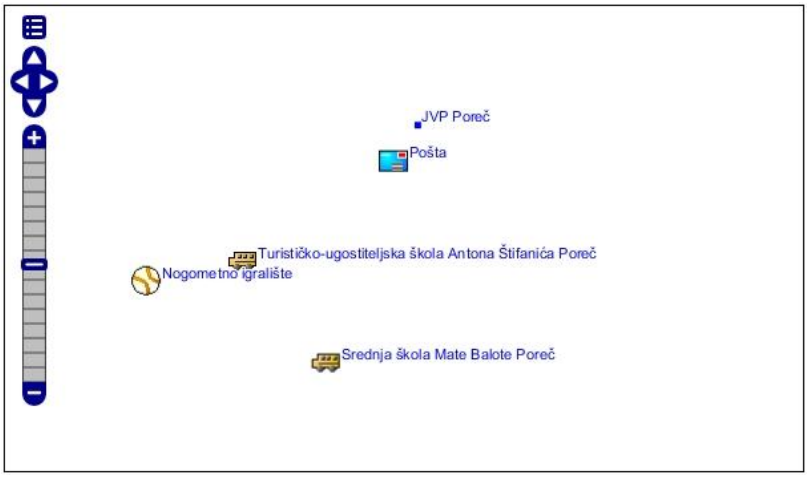
```
59                    <Fill>
60                      <CssParameter name=fill>#0000F1</CssParameter>
61                    </Fill>
62                  </TextSymbolizer>
```

Budući da smo završili sa definiranjem svih elemenata koji su nam potrebni za određivanje prikaza elemenata na karti, potrebno je provesti zatvaranje svih korištenih elemenata, kako bi se definirani stil ispravno interpretirao.

```
63            </Rule>
64          </FeatureTypeStyle>
65        </UserStyle>
66      </NamedLayer>
67  </StyledLayerDescriptor>
```

Prije konačnog spremanja napravljenog stila, korisno je provesti proces provjere koda (funkcija *Validate*). Ukoliko su se tokom izrade dogodile neke greške ili propusti, iste je potrebno korigirati, jer se u suprotnom stil neće moći prikazati. Nakon tog koraka funkcijom *Submit* spremamo stil, te isti pridružujemo odgovarajućem sloju podataka, kao što je objašnjeno na kraju prethodnog poglavlja (*Unos podataka u Geoserver*). Ukoliko je proces ispravno izvršen, svi poštanski uredi iz baze podataka bi na karti trebali biti prikazanim slijedećim simbolom: 

. Isti postupak se provodi i za ostale točkaste objekte, čime se dobiva njihov prikaz kao na slijedećoj slici.



Slika 8. Prikaz točkastih objekata na karti

5.3.2. Linijski znakovi

Linijskim znakovima su prikazane prometnice, željeznice, planinarske i biciklističke staze, rijeke, te izohipse i izobate. U ovom primjeru će biti objašnjen princip prikazivanja rijeka na karti. Budući da je prvih deset redaka identično za sve *SLD* dokumente, objašnjavanje istih će biti izostavljeno. Kao i u prethodnom primjeru i ovdje se pomoću filtera izdvaja željeni tip objekata iz atributne tablice, u ovom slučaju su to rijeke (*GRMN_TYPE = RIVER*), te pomoću elementa *MaxScaleDenominator* definiramo da će se takvi objekti prikazivati na karti tek kod mjerila krupnijih od 1:2000000.

```
11        <FeatureTypeStyle>
12          <Rule>
13            <ogc:Filter>
14              <ogc:PropertyIsEqualTo>
15                <ogc:PropertyName>GRMN_TYPE</ogc:PropertyName>
16                <ogc:Literal>RIVER</ogc:Literal>
17              </ogc:PropertyIsEqualTo>
18            </ogc:Filter>
19            <MaxScaleDenominator>2000000</MaxScaleDenominator>
```

I u ovom slučaju treba voditi računa o generalizaciji, te na najsitnijim mjerilima prikazivati samo objekte najveće važnosti, a zatim s povećanjem približenja karte (*zoom*-a) postupno uključivati i ostale objekte manje važnosti.

Elementom *LineSymbolizer* se definira da će navedeni objekt biti prikazan linijskim simbolom, a unutar tog elementa se određuje kako će taj prikaz izgledati na karti. Konkretno definiraju boja i debljina linije, no ukoliko je potrebno može se namještati i druge elemente, npr. može se definirati da linija bude isprekidana i sl. U slučaju da želimo napraviti rubove linija u nekoj drugoj boji, to nije moguće postići dodavanjem parametara unutar ovog elementa, nego je potrebno kreirati novi stil (*FeatureTypeStyle*) za iste objekate i njemu pridodati neku drugu boju i debljinu linije kako bi dobili željeni obrub linija.

```
20            <LineSymbolizer>
```

```
21            <Stroke>
22               <CssParameter name=stroke>#5864F8</CssParameter>
23               <CssParameter name=stroke-width>2</CssParameter>
24            </Stroke>
25         </LineSymbolizer>
```

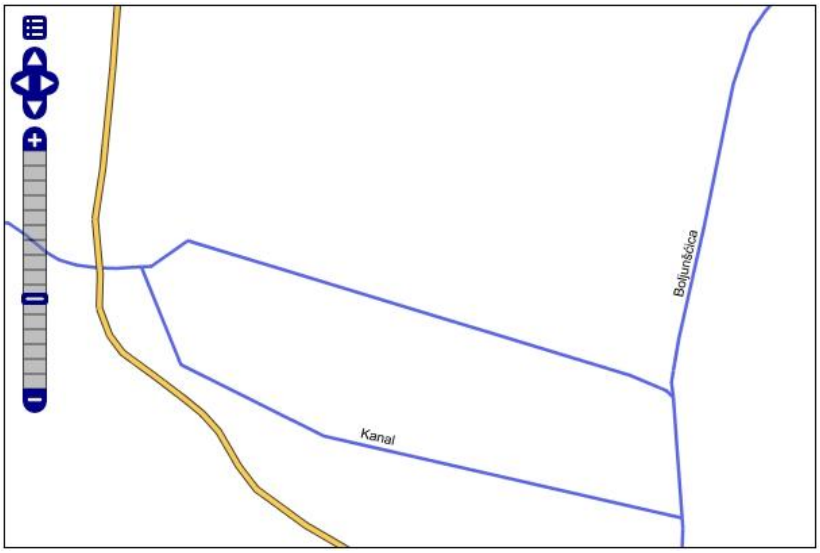
Također je moguće definirati i da se uz liniju prikazuje dodatna tekstualna oznaka objekta. To se ostvaruje elementom *TextSymbolizer* u kojem određujemo koji atributi će biti prikazani (*NAME_HR*) i na koji način će taj tekst biti prikazan, što je već opisano u prethodnom primjeru. Elementom *LabelPlacement* definiramo gdje će taj tekst biti smješten u odnosu na liniju, pa je konkretno definirano kako će isti biti odmaknut od linije za 8 piksela.

```
26         <TextSymbolizer>
27            <Label>
28               <ogc:PropertyName>NAME_HR</ogc:PropertyName>
29            </Label>
30            <Font>
31               <CssParameter name=font-family>Arial</CssParameter>
32               <CssParameter name=font-size>10</CssParameter>
33               <CssParameter name=font-style>normal</CssParameter>
34            </Font>
35            <LabelPlacement>
36                  <LinePlacement>
37                    <PerpendicularOffset>8</PerpendicularOffset>
38                  </LinePlacement>
39            </LabelPlacement>
40            <Halo>
41               <Radius>1</Radius>
42               <Fill>
43                  <CssParameter name=fill>#FFFFFF</CssParameter>
44               </Fill>
45            </Halo>
46            <Fill>
47               <CssParameter name=fill>#000000</CssParameter>
48            </Fill>
```

Osim pozicije teksta, potrebno je definirati i njegov izgled na karti, tj. pozadinu i boju teksta, a uz to je još pomoću elementa *VendorOption* potrebno definirati kako će se taj tekst ponašati na karti, pa konkretno određujemo tekstu da prati liniju čije atribute prikazuje. Nakon toga zatvaramo sve otvorene elemente, te provodimo provjeru i spremanje koda.

```
49            <VendorOption name=followLine>true</VendorOption>
50         </TextSymbolizer>
51      </Rule>
52   </FeatureTypeStyle>
```

Ukoliko je sve provedeno ispravno trebali bi na karti dobiti konačan prikaz rijeka kako što je vidljivo na slijedećoj slici.



Slika 9. Prikaz rijeka i prometnica na karti

5.3.3. Površinski znakovi

Površinskim znakovima, odnosno poligonima je prikazano more, jezera, naseljena područja, parkovi, parkirališta i dr. Postupak definiranja stilova za poligone je sličan prethodno objašnjenim postupcima, pa je tako i u ovom slučaju provedeno filtriranje sadržaja, te definiranje razine mjerila pri kojoj će sadržaj biti prikazan. U konkretnom slučaju se definira stil za prikazivanje većih naselja na karti, odnosno svih objekata koji u atributnoj tablici pod atribut *GRMN_TYPE* imaju upisanu vrijednost *LARGE_CITY*.

```
11      <FeatureTypeStyle>
12        <Rule>
13          <Name>LARGE_CITY</Name>
14            <ogc:Filter>
15               <ogc:PropertyIsEqualTo>
16                  <ogc:PropertyName>GRMN_TYPE</ogc:PropertyName>
17                  <ogc:Literal>LARGE_CITY</ogc:Literal>
18               </ogc:PropertyIsEqualTo>
19            </ogc:Filter>
20            <MaxScaleDenominator>2000000</MaxScaleDenominator>
```

Kao što je i logično, da bi prikazali objekte pomoću poligona, potrebno je koristiti element *PolygonSymbolizer*, kojem zatim definiramo način prikaza na karti. Budući da se ovim poligonom samo naznačava rasprostiranje nekog naselja na karti nema potrebe za kreiranjem složenog stila, stoga se u konkretnom slučaju samo definira ispuna poligona u sivoj boji (*#DFDFDF*).

```
21            <PolygonSymbolizer>
22               <Fill>
23                  <CssParameter name=fill>#DFDFDF</CssParameter>
24               </Fill>
25            </PolygonSymbolizer>
26        </Rule>
27      </FeatureTypeStyle>
```

U ovom slučaju je također potrebno provoditi generalizaciju sadržaja i to na način da se pri najsitnijem mjerilu prikazuju samo poligoni najveće važnosti, poput mora, otoka i velikih gradova, a zatim da se sa približavanjem prema najkrupnijim mjerilima postupno dodaju manje uočljivi poligoni poput manjih naselja i šumskih područja. Konačni izgled poligonskog elementa je prikazan na slici.

Slika 10. Prikaz poligona na karti

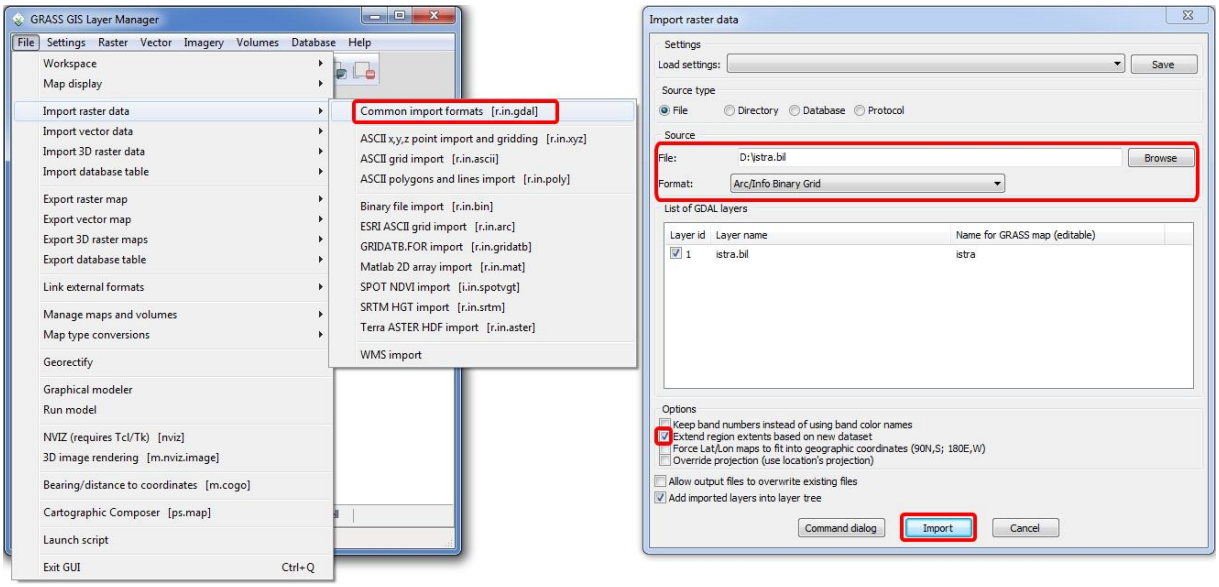
5.3.4. Reljef

Budući da je *AdriaTOPO* topografska karta koju većinom koriste osobe koje često borave u prirodi (planinari, izletnici, biciklisti …) bitno je na njoj prikazati što više informacija o samoj konfiguraciji terena, kako bi im se olakšalo snalaženje u prostoru. Uz izohipse i kote to se može ostvariti i dodavanjem sjenčanog prikaza reljefa u rasterskom obliku. U tom slučaju se dobiva potpunija slika terena jer nam sjenčani reljef detaljno prikazuje njegovu konfiguraciju, a izohipse i kote pridružuju konkretne brojčane vrijednosti visina. Kako bi dodali sjenčani reljef na kartu potrebno je napraviti određene predradnje, koje se sastoje od generiranja samih sjena iz sirovih podataka o visinama terena, zatim međusobno preklapanje više takvih sjenčanih prikaza u jedan konačni, definiranja stila prikaza na karti, te samog dodavanja sloja na kartu, što će biti detaljnije opisano u narednim koracima.

5.3.4.1. Izrada sjenčanog prikaza reljefa

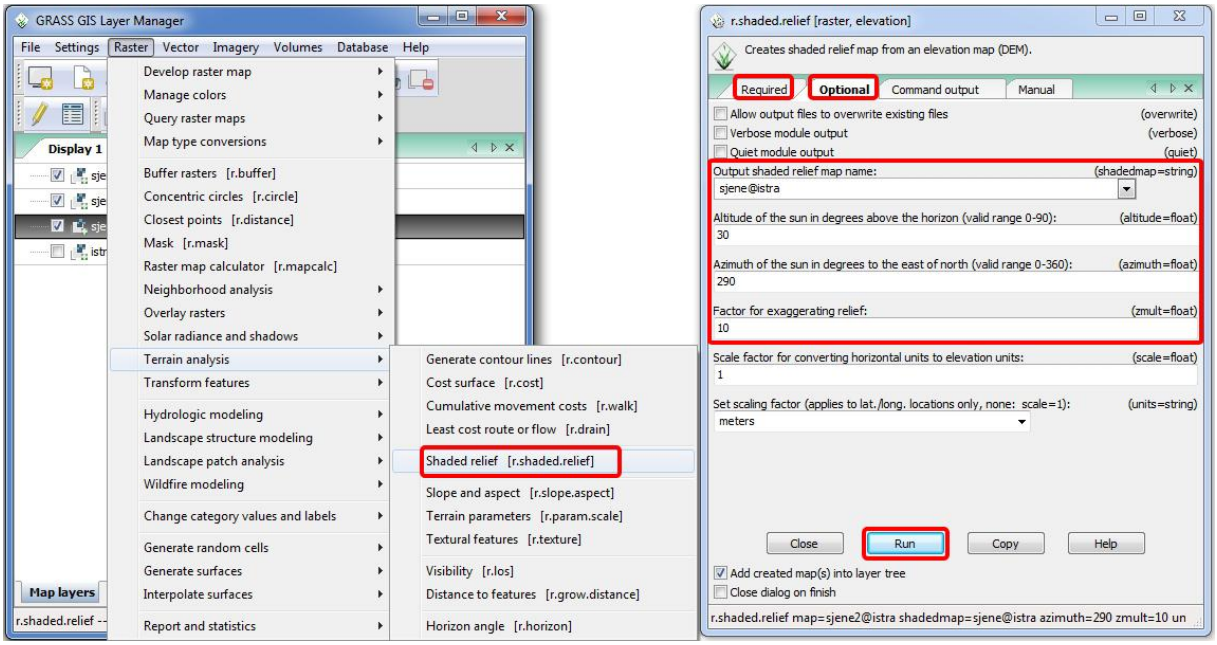
Sam proces izrade sjenčanog prikaza reljefa započinje se učitavanjem podataka o visinama terena u program *GRASS GIS*. Vrijednosti visina su spremljene u jednostavnoj binarnoj rasterskoj datoteci *istra.bil*, a njezino učitavanje se izvršava pomoću opcije *File > Import Raster Data > Common Import Formats* u *GRASS GIS*-u. Time se otvara prozor sa opcijama za učitavanje u kojem je kao format ulaznih podataka potrebno izabrati *Arc/Info Binary Grid*, a zatim definirati koju datoteku želimo

uvesti. Poželjno je uključiti i opciju *Extend region extents based on new dataset*, kojom će se automatski postaviti granice regije radnog prostora *GRASS GIS*-a na granice učitanih podataka. Nakon toga je potrebno još samo funkcijom *Import* pokrenuti učitavanje navedene datoteke (*slika 9*).



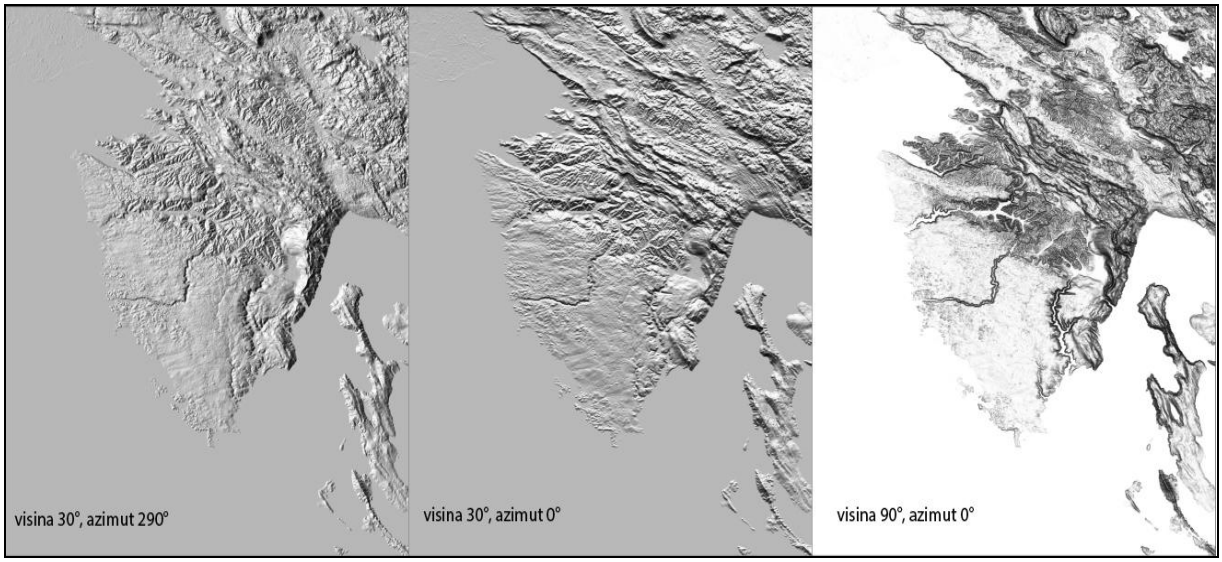
Slika 11. Učitavanje podataka u GRASS GIS

Slijedeći korak nakon učitavanja datoteke je izrada sjenčanog reljefa izborom funkcije *Raster > Terrain Analysis > Shaded relief*. Njenim pokretanjem se otvara prozor sa postavkama gdje u kartici *Requred* definiramo izvornu datoteku (*istra.bil*) iz koje će se generirati sjene, a zatim u kartici *Optional* namještamo postavke za sami prikaz sjena (*slika 10*).



Slika 12. Izrada sjenčanog reljefa

Tu je potrebno najprije definirati naziv izlaznog sloja, zatim postaviti visinu Sunca iznad horizonta na vrijednost 30°, azimut Sunca na 290°, te faktor množenja visina na vrijednost 10, čime se postiže izraženiji dojam visina. Ostale elemente ostavljamo na pretpostavljenim vrijednostima, te funkcijom *Run* pokrećemo generiranje sjenčanog reljefa. Isti postupak se provodi još i za situaciju kada se Sunce nalazi na visini 30° i pod azimutom od 0°, čime se dobivaju sjene terena suprotne onima iz prethodnog slučaja. Kako bi postigli poseban efekt kojim naglašavamo strma područja na terenu, provodimo još jedno generiranje sjena i to sa vrijednostima visine Sunca 90° i azimutom 0°. Tim postupcima smo dobili tri različita prikaza terena, kao što je vidljivo na *slici 11.*

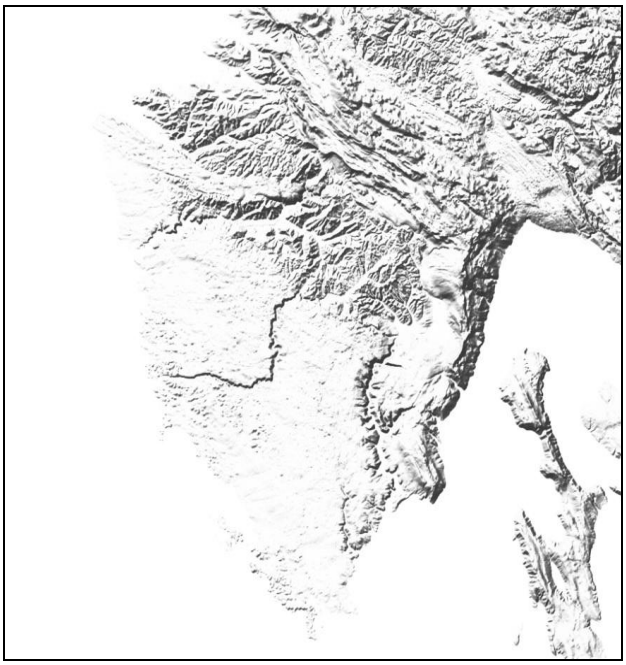


Slika 13. Usporedni prikaz sjenčanog reljefa

Budući da niti jedan od tih prikaza ne dočarava teren na zadovoljavajući način, potrebno je izvršiti daljnje manipulacije nad rezultatima pomoću nekog od programa za obradu fotografija (*Photoshop, GIMP* …). Kako bi to mogli provesti, rezultate je potrebno eksportirati iz GRASS GIS-a u slikovne datoteke. Jedan od tipova takvih datoteka je *TIFF* (eng. *Tagged Image File Format*). Izvoz u taj format se provodi funkcijom *File > Export raster map > TIFF export.* U prozoru koji se zatim otvara potrebno je u kartici *Requred* definirati nazive ulazne i izlazne datoteke, a u kartici *Optional* treba uključiti opciju *Output TIFF world file*, te pokrenuti naredbu *Run* kojom započinje izvoz podataka. Opcija *Output TIFF world file* je bitna zato jer se na taj način uz datoteku zapisuju i njezini prostorni podaci, koji nose informacije o veličini piksela na *x* osi, rotacijama oko *x* i *y* osi, veličini piksela na *y* osi, te koordinatama *x* i *y* centra gornjeg lijevog piksela (ishodišnog piksela za slikovni koordinatni sustav), a pomoću njih je kasnije moguće pravilno prikazivanje slika u prostoru.

Nakon izvoza, datoteke učitavamo u program za obradu fotografija, konkretno *Photoshop.* Zatim datoteke preklopimo na način da se svaka nalazi u posebnom sloju. Definiranjem svojstava pojedinih slojeva, prvenstveno prozirnosti i aritmetičke operacije spajanja jednog sloja prema drugima moguće

je postići različite efekte kojima se postiže sjenčani prikaz reljefa sličniji onima kakvi se izrađuju prema kartografskim pravilima. Nakon tih postupaka dobiva se konačni izgled sjenčanog reljefa, kao što je prikazan na *slici 12*, koji spremamo u datoteku *sjene.jpg*.



Slika 14. Konačni izgled sjenčanog reljefa

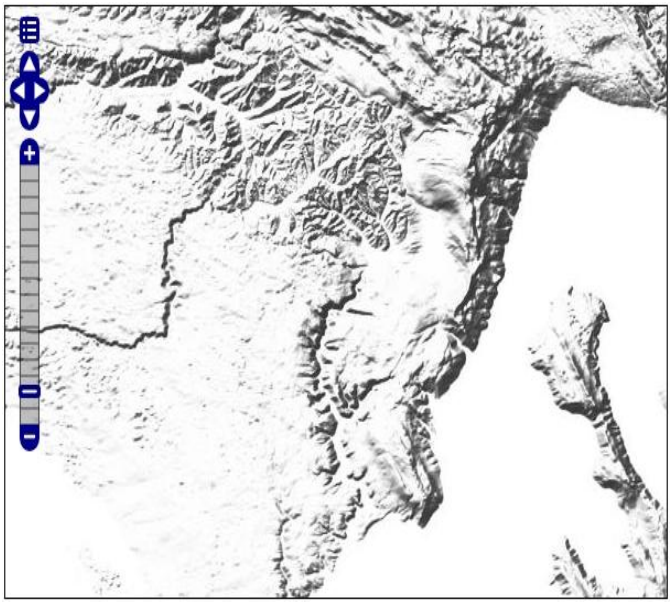
5.3.4.1. Izrada stila i učitavanje u Geoserver

Kako bi sjenčani prikaz reljefa mogli prikazati u *Geoserver*-u, potrebno je izraditi njegov stil prikaza, na sličan način kao i sa prethodnim slojevima. Dakle pomoću *SLD*-a definiramo da će se prikazivati rasterski sloj (*RasterSymbolizer*), kojemu ćemo samo promijeniti vrijednost prozirnosti (*Opacity*) na iznos *0.3*, što znači da će vidljivost rastera biti *30%*. To je potrebno postaviti iz razloga što će se raster sa sjenčanim terenom biti postavljen iznad poligonskih slojeva, što bi u slučaju neprozirnosti imalo posljedicu da se slojevi koji su ispod njega ne bi vidjeti na karti.

```
10    <UserStyle>
11      <Title>Sjencani prikaz terena</Title>
12      <FeatureTypeStyle>
13        <Rule>
14          <RasterSymbolizer>
15            <Opacity>0.3</Opacity>
16          </RasterSymbolizer>
17        </Rule>
18      </FeatureTypeStyle>
19    </UserStyle>
```

Učitavanje rastera u *Geoserver* se provodi tako da unutar postojećeg radnog prostora stvorimo novo spremište. Kako tip podataka koji želimo uvesti odabiremo poveznicu *WorldImage*, kojom se otvara nova stranica u kojoj upisujemo naziv za navedeni sloj i kratki opis, a nakon toga navodimo putanju do željene datoteke (data/istra/istra_dem/sjene.jpg). Bitno je napomenuti da uz navedenu datoteku

treba je priložiti i jednu od .*wld* datoteka koje su generirane prilikom izvoza sjenčanog reljefa iz *GRASS GIS*-a u slikovne datoteke. Nju je potrebno preimenovati u *sjene.jgw,* kako bi *Geoserver* mogao razumjeti sa kojom datotekom je povezana. Slijedeći korak, nakon stvaranja spremišta, je objavljivanje učitanog sloja, na način koji je opisan u prethodnim poglavljima. Dakle, sloju pridružimo koordinatni sustav i definiramo granice, a nakon što mu pridružimo stil prikaza i potvrdimo njegovo objavljivanje dobivamo njegov prikaz na karti (*slika 13*).



Slika 15. Prikaz sjena na karti

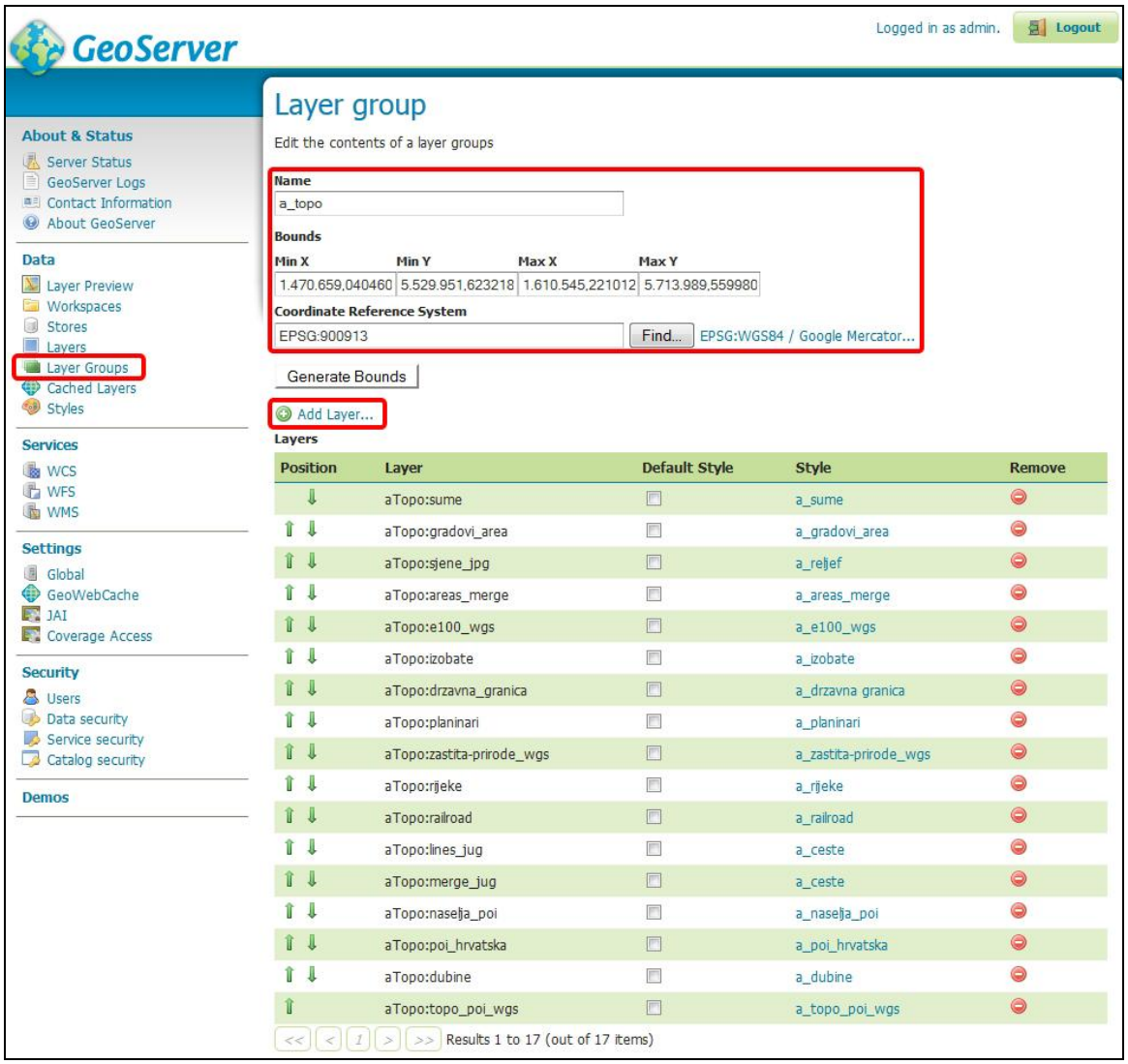
5.4. Grupiranje slojeva

Kako bi od svih slojeva koje smo zasebno učitali u *Geoserver* napravili konačni proizvod, tj. topografsku kartu, potrebno je provesti njihovo grupiranje, odnosno smisleno preklapanje. Pri tome se mora voditi računa o tome da podaci koje jedan sloj prikazuje ne zaklanjaju neki drugi sloj. Isto tako treba voditi računa o logičnoj dosljednosti same karte, što znači da slojeve treba postavljati na isti način kako stoje i objekti, koji su sadržani u slojevima, u prirodi, npr. ceste prelaze preko rijeke, a ne obrnuto.

Sam postupak grupiranja slojeva se provodi tako da u *Geoserver*-u preko poveznice *Layer Groups* otvaramo novu stranicu u kojoj pomoću funkcije *Add new layer group* dolazimo na stranicu za stvaranje nove grupe slojeva (*slika 14*). Tu se najprije upisuje naziv same grupe (*a_topo*), a zatim se definira granični okvir i projekcija u kojoj će grupirani slojevi biti prikazani. Za prikaz grupe je odabrana *Google Mercator* projekcija. Njezin *EPSG* kod je *900913* i potrebno ga je upisati u polje *Coordinate Reference System*. Navedena projekcija je bazirana na Mercatorovoj projekciji, a predznak *Google* je dobila iz razloga što je istoimena kompanija koristi za prikazivanje satelitskih snimaka i

kartografskih prikaza na *Google Maps* kartografskom servisu. Osim što je pogodna za ispravno prikazivanje područja Istarskog poluotoka, navedena projekcija ima i prednost iz razloga što nudi mogućnost preklapanja sa slojevima koje nudi navedeni servis, ukoliko se ukaže potreba. Koordinate graničnog okvira u kojem će se podaci prikazivati možemo upisati ručno ili generirati automatski iz granica učitanih slojeva.

Sam postupak dodavanja slojeva u grupu provodimo pomoću poveznice *Add Layer...* koja otvara skočni prozor sa popisom svih slojeva koje smo učitali i objavili u *Geoserver*-u. Slojeve koje želimo prikazati na karti odabiremo, čime se oni automatski dodaju na listu, kao što je prikazano na *slici 14*.



Slika 16. Grupa učitanih slojeva

Na tablici u prethodnoj slici prikazan je popis svih slojeva koji se prikazuju na konačnoj karti. Bitna činjenica je da njihov redoslijed određuje i redoslijed prikazivanja na karti, što znači da će sloj koji je prvi po redu u tablici (*aTopo:sume*) prvi biti i prikazan na karti, a ostali će se slijedom preklapati preko njega. Promatrajući sami raspored, primjećujemo kako su poligoni šuma i gradova preklopljeni

rasterskim slojem (*sjene_jpg*). To je napravljeno iz razloga što će zbog transparentnosti rasterski sloj samo zasjeniti spomenute poligone i time će se postići realniji prikaz terena. Postavljanje poligona *areas_merge* (sadrži poligone mora i jezera) iznad rastera je napravljeno iz razloga što rasterski sloj ne prati dosljedno samu granicu između mora i kopna, pa je na taj način napravljeno njihovo strogo razgraničavanje. Zatim su dodavani linijski slojevi koji su imaginarnog karaktera poput izobata, izohipsa, granica države i zaštićenih prirodnih područja, a preko njih su redom preklapane rijeke, željeznice i ceste. Nakon njih su dodani slojevi sa točkastim znakovima, iz razloga da se prikazuju iznad svih prethodnih slojeva. Konačan izgled karte, sa svim preklopljenim slojevima je prikazan na slici 15.

Slika 17. Konačni izgled karte

5.5. Izrada korisničkog sučelja

Izrada korisničkog sučelja je podijeljena u dvije cjeline. Jedna je izrada samog vizualnog sučelja u kojem će karta biti prezentirana korisnicima i preko kojeg će biti omogućena njihova interakcija, a druga je izrada i implementacija funkcionalnosti koje će omogućavati spomenutu interaktivnost. Te činjenice su utjecale i na izbor korištenih alata, pa su za prvi slučaj korišteni prezentacijski alati *HTML* i *CSS*, a u drugom *JavaScript* (*jQuery* i *OpenLayers*). Sam postupak izrade je opisan u narednim poglavljima.

5.5.1. Izrada vizualnog sučelja

Osnovna ideja prilikom izrade vizualnog sučelja ovog geionformacijskog web servisa je da se pomoću *HTML* jezika izradi kostur sučelja u koji će se zatim smjestiti sve njegove funkcionalne komponente.

Taj postupak je proveden tako da je najprije kreiran osnovni *HTML* dokument, na način kao što je prikazano u slijedećem isječku:

```
1  <html>
2     <head>
3     </head>
4     <body>
5     </body>
6  </html>
```

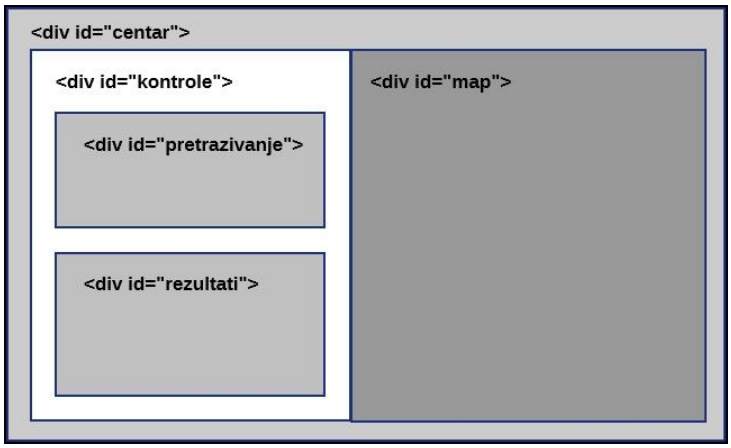
U njemu se nalaze osnovni elementi koji čine svaki *HTML* dokument, a to su početna *<html>* i završna *</html>* oznaka (eng. *tag*), koja govori da se radi o tipu datoteke napisane *HTML* jezikom. Unutar navedenih oznaka se definiraju zaglavlje (eng. *head*) i tijelo (eng. *body*) stranice. Zaglavlje je element koji služi za definiranje opisnih podataka vezanih uz dokument i za navođenje referenci na vanjske dokumente koji se uključuju u stranicu, a u tijelo dokumenta se unose svi elementi koji se žele prikazati na samoj stranici u pregledniku. U konkretnom slučaju u zaglavlje je ugrađen slijedeći sadržaj:

```
<head>
    <!-- Način interpretacije znakova u web pregledniku -->
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

    <!-- Kratki naslov stranice koji se prikazuje u naslovnoj traci preglednika -->
    <title>AdriaTOPO web GIS</title>

    <!-- Referenca na datoteku u kojoj je definiran stil za prikaz sadržaja stranice -->
    <link rel="stylesheet" type="text/css" href="css/sucelje_stil.css"/>

    <!-- Referenca za uključivanje datoteke sa OpenLayers skriptama -->
    <script src="js/OpenLayers/OpenLayers.js" type="text/javascript" ></script>
    <!-- Referenca za uključivanje datoteke sa jQuery skriptama -->
    <script src="js/jquery.js" type="text/javascript" ></script>
    <!-- Referenca za uključivanje datoteke sa korisničkim funkcijama -->
    <script src="js/korisnicke_funkcije.js" type="text/javascript" ></script>
</head>
```

Prvi po redu korišteni element nosi oznaku *meta*, čija funkcija je da definira metapodatke pomoću kojih će preglednik razumjeti na koji način treba interpretirati znakove. Za konkretni je slučaj navedeno da će se koristiti *UTF-8* enkodiranje. Slijedeći je po redu element *title* gdje je naveden kratki naslov stranice, koji će se prikazivati u naslovnoj traci preglednika. Nakon toga se definira element sa oznakom *link*, čija je funkcija povezivanje *HTML* dokumenta sa vanjskom datotekom u kojoj je definiran *CSS*[4] stil prikaza pojedinih elemenata na stranici. Budući da se u radu koriste *OpenLayers* i *jQuery* alati, potrebno je izvršiti uključivanje njihovih funkcionalnosti elementom *script*, kako bi se moglo početi sa razvijanjem prilagođenih kontrola za interakciju korisnika sa kartom. Te će se kontrole također nalaziti u vanjskoj datoteci (*korisnicke_funkcije.js*), pa je neophodno i referiranje na nju. Bitno je navesti kako se sve putanje do vanjskih datoteka navode relativno u odnosu na lokaciju gdje se nalazi *HTML* datoteka, koja će služiti za prikazivanje sadržaja u pregledniku.

[4] CSS (eng. *Cascading Style Sheets*) - je stilski jezik koji se koristi za opis prezentacije dokumenta napisanog pomoću *HTML* jezika.

Nakon definiranja zaglavlja dokumenta potrebno je u tijelu dokumenta definirati kako će se izvršiti prikaz svih korisničkih komponenti ovog servisa na stranici. Kako bi si jednostavnije predočili na koji način da rasporedimo sve elemente pogodno je izraditi shemu po kojoj bi elementi trebali biti raspoređeni (*slika 16*).



Slika 18. Shema rasporeda elemenata

Iz shematskog prikaza je lako vidljiv hijerarhijski odnos elemenata (roditelj - dijete), što nam olakšava izradu samog *HTML* koda koji će te odnose određivati na samoj stranici. Počevši od *body* elementa, koji obuhvaća sve elemente sučelja, prvi slijedeći element u hijerarhiji je element sa identifikatorom *centar*. On obuhvaća dva manja elementa, *kontrole* i *map*. U element *map* će biti postavljena sama karta, a element *kontrole* će sadržavati dva podelementa, *pretrazivanje* i *rezultati*, a iz njihovih imena je lako zaključiti da će sadržavati funkcije za pretraživanje i prikazivanje rezultata. Navedenu hijerarhiju je tada moguće jednostavno pretvoriti u *HTML* kod, koji je prikazan u nastavku:

```
<body>
    <div id="centar">
        <div id="kontrole">
            <div id="pretrazivanje"></div>
            <div id="rezultati"></div>
        </div>
        <div id="map"></div>
    </div>
</body>
```

Kako bi stvorili konačnu konstrukciju sučelja, svim elementima je potrebno pridružiti *CSS* stilove prikaza u vanjskoj datoteci *sucelje_stil.css*. Budući da za element *body* želimo da obuhvaća cijeli prostor stranice, izostavit ćemo ga prilikom definiranja stila i samim time njegove postavke ostaviti na pretpostavljenim vrijednostima. Element sa identifikatorom *centar* će u sebi sadržavati sve ostale elemente sučelja, stoga ćemo mu ograničiti dimenzije (širinu na *900*, a visinu na *600* piksela), kako ne bi došlo do neželjenog razmještanja elemenata prilikom pregledavanja stranice na monitorima različitih rezolucija. Kako bi se element automatski smještao u sredinu stranice, definiramo mu marginu koja će sa lijeve i desne strane elementa automatski postavljati jednaki iznos praznog

prostora, a kako bi ga vizualno razgraničili sa *body* elementom dodajemo mu crni obrub debljine *1* piksel. Sam *CSS* kod je vidljiv u nastavku:

```
#centar {
    width: 900px;
    height: 600px;
    margin: auto;
    border: 1px solid #000000;
}
```

Element *kontrole* će podijeliti element *centar,* na dva dijela i pri tome zauzeti 200 piksela njegove širine. Biti će mu definirana i boja pozadine, kako bi se istaknuo od okoline, kao što je prikazano u kodu:

```
#kontrole {
    float: left;
    width: 200px;
    height: 600px;
    background: #ece9e9;
}
```

Elementi *pretrazivanje* i *rezultati* će se nalaziti unutar elementa *kontrole* i biti će smješteni jedan ispod drugog. Prvi će zauzimati 200 piksela njegove visine i prostirat će se na cijeloj širini, dok će drugi (*rezultati*) nalaziti ispod njega i zauzeti će preostali slobodni prostor. Također će mu biti dodana funkcija *overflow* kojom se, u slučaju da se element ispiše sadržaj koji je veći od njegovih dimenzija, automatski dodaje klizač kojim se možemo pomicati kroz sve rezultate.

```
#pretrazivanje {
    width: 200px;
    height: 200px;
    padding: 10px;
}
#rezultati {
    width: 200px;
    height: 400px;
    overflow: auto;
}
```

Sama karta (element *map*) se postavlja do elementa *kontrole* i prikazivati će se na cijelom preostalom području elementa *centar*.

```
#map {
    float: left;
    width: 700px;
    height: auto;
}
```

Nakon definiranja same konstrukcije sučelja potrebno je dodati još i elemente pomoću kojih će korisnik postavljati upite na karti. Predviđeno mjesto za njihov smještaj je element *pretrazivanje*. Krenuvši redom po kojem se pojavljuju na sučelju, prvi takav element je padajući izbornik u kojem će biti omogućen izbor kategorija podataka koje se želi pretraživati. Sam izbornik je izrađen pomoću oznaka *select* (prikazano u kodu ispod), no kako bi se kasnije mogao povezivati sa funkcijama i ostalim elementima stranice, dodani su mu atributi naziv (eng. *name*) i identifikator (*id*). Unutar njega

se dalje ugrađuju opcije (eng. *option*) sa kategorijama po kojima će se vršiti pretraživanje podataka. Navedene kategorije su definirane prema setu podataka na koji se odnose, pa su tako kao atributi u svakoj opciji navedeni *value*, koji zapravo nosi naziv sloja iz kojeg će se podaci pretraživati, a u atribut *title* je upisan naziv filtera koji će biti primijenjen prilikom pretraživanja.

```
<select name="izbornik_kategorija"
        id="izbornik_kategorija">
    <option value="naselja_poi" title="filterNaselja" >Naselja</option>
    <option value="poi_hrvatska" title="filterPOI" >POI točke</option>
</select>
```

Slijedeći element koji se dodaje je polje za upisivanje pojma koji će se tražiti u podacima. To je izvedeno pomoću oznake *input*, tipa *text*. Također mu se definira identifikator, a u atribut *value* se stavlja početna vrijednost teksta koji će biti prikazivan ukoliko korisnik nije upisivao neki pojam za pretraživanje. Pomoću atributa *onclick* je postignuta funkcionalnost da se navedeni pretpostavljeni tekst izbriše kada korisnik pokazivačem klikne unutar polja. Suprotan efekt se postiže atributom *onblur*, kojim se postiže da se neki pretpostavljeni tekst automatski upiše u polje kad korisnik ostavi prazno polje i klikne izvan polja za upis. Navedene su funkcionalnosti ugrađene s razlogom da se onemogući pretraživanje bez zadanih kriterija, što bi kao posljedicu moglo imati vračanje prevelikog skupa podataka, a to bi moglo izazvati probleme u radu samog servisa. Sam kod opisanog elementa je naveden u nastavku:

```
<input type="text"
       id="traziTekst"
       value="upiši naziv..."
       onclick="if(this.value=='upiši naziv...'){this.value=''}"
       onblur="if(this.value==''){this.value='upiši naziv...'}" />
```

Posljednji element koji će se nalaziti u rubrici za pretraživanje je gumb kojim će se pokretati sam proces pretraživanja. Kao što je vidljivo u kodu ispod, element se također izrađuje pomoću oznake *input*, no ovaj put je tipa *button*. I njemu je definiran jedinstveni identifikator i vrijednost (*value*), koja je zapravo njegov naziv koji će biti prikazan u sučelju. Posljednji atribut u ovom elementu je *onclick*, a pomoću njega se postiže pozivanje funkcije *pretrazi* (implementirana je u korisničkim funkcijama) kad se navedeni gumb pritisne.

```
<input type="button"
       id="searchGumb"
       value="Traži"
       onclick="pretrazi()" />
```

Nakon što je završen postupak definiranja svih elemenata koji će se koristiti za pretraživanje podataka, potrebno je stvoriti prostor koji će se koristiti i za prikazivanje samih rezultata. To je izvedeno tako da je unutar elementa *rezultati* stvorena prazna brojčana lista (oznaka *ol*) sa identifikatorom *ispis* (prikazano u kodu ispod). Na taj smo način, osim prostora u koji će se podaci ispisivati, definirali i samu strukturu izlaznih podataka.

```
<div id="rezultati">
    <ol id="ispis"></ol>
</div>
```

Glavnu ulogu u ovom elementu ima identifikator, preko kojeg će se izvršiti povezivanje sa funkcijama koje će prilikom pokretanja funkcije za pretraživanje automatski početi puniti tablicu rezultatima, što će detaljnije biti objašnjeno u slijedećem poglavlju.

5.5.2. Implementacija funkcionalnosti

Proces implementacije funkcionalnosti u korisničko sučelje ovog servisa sastojao se iz dva dijela. Prvi dio je izrada funkcije koja će omogućiti učitavanje i pregledavanje karte u sučelju, što se uglavnom sastojalo od implementacije *OpenLayers* funkcija za upravljanje kartom, a u drugom koraku su izrađene prilagođene funkcije koje obavljaju specifične dužnosti, poput filtriranja podataka iz pojedinih slojeva i sl. Više detalja o svakom koraku biti će rečeno u narednim poglavljima.

5.5.2.1. Implementacija bazne karte

Nakon same izrade karte u *Geoserveru* dobivamo mogućnost njenog pregledavanja sa pretpostavljenim postavkama i funkcionalnostima, što često ne zadovoljava potrebe pojedinih aplikacija, te je iste potrebno u većoj ili manjoj mjeri prilagoditi zahtijevanoj situaciji. Taj proces se uglavnom sastoji od izrade funkcije, koja će se automatski pozivati prilikom učitavanja same web stranice servisa, i u koju će biti ugrađene sve postavke vezane uz prikaz karte. U slučaju ovog servisa, u vanjskoj datoteci sa funkcijskim skriptama (*korisnicke_funkcije.js*) je izrađena funkcija *init()*, koja obavlja takvu ulogu.

Prvi korak prilikom izrade *init()* funkcije je stvaranje objekta *OpenLayers* karte (*map*) pomoću konstruktora *OpenLayers.Map*. Spomenutom objektu je pridružen i argument *options,* koji omogućava definiranje svojstava pomoću kojih utječemo na sami izgled karte, pa smo tako u njemu redom definirali da ćemo koristiti *OpenLayers* kontrole (*controls*), te stvaramo prazno polje u koje će kasnije biti učitane kontrole koje su nam potrebne. Zatim smo definirali granični okvir karte (*maxExtent*) koji će preuzeti vrijednosti rubnih (lijevo, dolje, desno, gore) koordinata definiranih u varijabli *granice*. Element *maxResolution* će sadržaj karte automatski prilagođavati okviru u kojem se prikazuje. Pomoću *numZoomLevels* se definira broj razina promjene mjerila, odnosno razina približenja. Elementom *projection* definiramo da će se sadržaj prikazivati u *Google Mercator* projekciji (*EPSG:900913*), a kako bi se koordinate pokazivača na karti, umjesto u metrima (koje propisuje *Google Mercatorova* projekcija), prikazivale u korisnicima razumljivijim koordinatama u stupnjevima prema *WGS 84* sustavu, definiramo i element *displayProjection* sa *EPSG* kodom tog sustava (*EPSG:4326*). Navedene definicije se zapisuju na slijedeći način:

```
var granice = new OpenLayers.Bounds(
        1470659.0404607197, 5529951.623218162,
        1610545.2210123378, 5713989.559980625
);

var options = {
        controls: [],
        maxExtent: granice,
        maxResolution: "auto",
        numZoomLevels: 10,
        projection: "EPSG:900913",
        displayProjection: "EPSG:4326",
};

var map = new OpenLayers.Map('map', options);
```

Nakon definiranja elemenata karte, potrebno je dodati sloj koji želimo prikazivati na karti. To provodimo pomoću slijedećeg koda, koji upisujemo u nastavku prethodne definicije karte:

```
var wms_sloj = new OpenLayers.Layer.WMS(
        "WMS sloj",
        "http://localhost:81/geoserver/wms",
        {
          layers: 'a_topo',
          format: 'image/png',
          tiled: true
        },
        {
          displayOutsideMaxExtent: true,
          isBaseLayer: true,
          transitionEffect: 'resize',
        {
);
map.addLayer(wms_sloj);
```

Dakle, ponovo koristimo konstruktor kako bi stvorili objekt *wms_sloj*, no ovaj put je konstruktor tipa *OpenLayers.Layer.WMS*, zato jer želimo učitati *WMS* sloj podataka. Kao prvi argument navodimo naziv koji ćemo dodijeliti sloju (*"WMS sloj"*), a zatim navodimo lokaciju poslužitelja na kojem se nalazi navedeni sloj, te navodimo točan naziv sloja koji želimo prikazivati. Njemu zatim definiramo da će se u pregledniku prikazivati u više manjih sličica (tzv. "pločica") u *.png* formatu. Taj princip prikazivanja karte putem Interneta je puno praktičniji zato jer se, umjesto jedne velike slike koja obuhvaća cijeli prikaz, kartu ovim putem šalje u manjim segmentima se koji puno brže prenose do korisnika. Pomoću *displayOutsideMaxExtent* postižemo da se prikazuje i sadržaj karte koji se nalazi izvan definiranog graničnog okvira, što je korisno kako bi izbjegli prazan prostor koji se može pojaviti na rubovima prikaza kada kartu pregledavamo na najsitnijim mjerilima. Navedeni sloj će biti postavljen kao bazni sloj. Kako bi se postigao efekt veće brzine rada, prilikom promjene razine mjerila, uključena je funkcija koja "prisilno" uvećava sličice i prikazuje ih na karti sve dok ih ne zamijeni originalna sličica koja odgovara zadanoj razini mjerila.

Kako bi uključili kontrole, pomoću kojih pregledavamo sadržaj kartografskog prikaza, potrebno je unijeti slijedeći sadržaj:

```
map.addControl(new OpenLayers.Control.PanZoomBar());
```

```
map.addControl(new OpenLayers.Control.Navigation());
map.addControl(new OpenLayers.Control.LayerSwitcher());
map.addControl(new OpenLayers.Control.ScaleLine());
map.addControl(new OpenLayers.Control.MousePosition());
map.zoomToExtent(granice);
```

U ovom slučaju smo u kartu uključili kontrolu *PanZoomBar()* (prikazuje skalu promjene mjerila i virtualne strelice za kretanje po karti), *Navigation()* (omogućuje pomicanje karte pomoću pokazivača miša), *LayerSwitcher()* (dodaje izbornik koji omogućava uključivanje i isključivanje prikaza pojedinih slojeva), *ScaleLine()* (prikaz grafičkog mjerila na karti), te MousePosition() koja prikazuje trenutne koordinate pokazivača miša. Uz navedene kontrole smo dodali i funkciju *zoomToExtent,* koja omogućava prikazivanje cijelog sadržaja karte unutar samog okvira prikaza.

Osim navedenih kontrola integrirana jedna prilagođena kontrola *info,* koja će se isto pozivati iz *init()* funkcije, a služit će za dohvaćanje informacija o slojevima i njihovo prikazivanje u tzv. skočnim prozorima (eng. *pop-up*). Njezin kod je ispisan u nastavku:

```
var info = new OpenLayers.Control.WMSGetFeatureInfo({
    url: 'http://localhost:81/geoserver/wms',
    layer: wms_sloj,
    queryVisible: true,
    eventListeners:
            {
        getfeatureinfo: function(event)
                    {
                        map.addPopup(new OpenLayers.Popup.FramedCloud(
                            "chicken",
                            map.getLonLatFromPixel(event.xy),
                            null,
                            event.text,
                            null,
                            true
                        ),
                            exclusive = true
                        );
                    {
            {
});
map.addControl(info);
info.activate();
```

Navedena kontrola se kreira slično kao i prethodne, dakle pomoću adekvatnog konstruktora (*WMSGetFeatureInfo*) stvaramo objekt koji će služiti za dohvaćanje i prikazivanje atributnih podataka. U argumente mu navodimo lokaciju poslužitelja i naziv sloja iz kojeg će izvlačiti informacije. Zatim postavljamo uvjet da upite može vršiti samo prema vidljivim slojevima. Pomoću funkcija koje prate događaje (eng. *eventListeners*) se postiže da prilikom klika pokazivačem na neko mjesto na karti dobivamo povratnu informaciju sa atributima koje taj sloj sadrži. Kako bi izbjegli situaciju da nam se prostor karte zatrpa skočnim prozorima uključena je opcija *exclusive,* koja prilikom otvaranja novog prozora zatvara prethodno otvoreni. Kako bi navedena kontrola ispravno funkcionirala potrebno ju je dodati u listu kontrola karte i aktivirati ju, na način da pozovemo funkciju *info.activate()*.

Time je završen postupak izrade funkcije *init(),* te ju možemo integrirati u samu web stranicu, na način da je pridružimo body elementu pomoću koda: "*<body onload="init()"*>". Navedeni će dio koda djelovati tako da će web preglednik najprije učitati samu stranica, a kad se učitavanje završi pozvat će se događaj *onload*, koji pokreće funkciju *init(),* koja zatim vrši prikazivanje karte u sučelju.

5.5.2.2. Implementacija funkcija za pretraživanje

Osnovicu funkcija za pretraživanje čini *WFS* protokol i njegove funkcionalnosti, pogotovo u smislu filtriranja sadržaja prema zadanim kriterijima. Stoga je glavna ideja bila izraditi funkciju (*pretrazi()*) koja će se pozivati pritiskom na gumb *Traži*. Nakon pokretanja filter će preuzeti kriterije pretraživanja, obraditi podatke, te iscrtati rezultate pomoću *WFS* protokola na karti i istovremeno ih ispisati u rubrici za rezultate. Cijeli taj postupak je proveden slijedećim redoslijedom.

Nakon otvaranja samog tijela funkcije *pretrazi()* implementiran je slijedeći kod:

```
wfs_sloj = map.getLayersByName('WFS sloj')[0];

if (wfs_sloj){
    map.removeLayer(wfs_sloj);
{

map.zoomToExtent(granice);
```

Funkcija naredbe iz prvog reda je da provjeri da li već postoji sloj pod nazivom "*WFS sloj*", te ukoliko postoji da njegovu vrijednost postavi u varijablu *wfs_sloj*. Pomoću uvjetnog iskaza *if* se zatim provjerava da li postoji varijabla *wfs_sloj*, te ukoliko je uvjet istinit pokreće se naredba koja tu varijablu, odnosno sloj, briše sa karte. Taj postupak se provodi iz razloga da ne bi došlo do nagomilavanja prevelike količine podataka na kartu, zbog činjenice da se prilikom pokretanja pretraživanja svaki put nanovo generira *WFS* sloj sa rezultatima pretraživanja. Posljednja linija iz ovog navoda služi za promjenu mjerila karte, na način da se u okviru preglednika prikaže cijeli sadržaj karte. Taj je korak potrebno provoditi iz razloga što *WFS* provodi upite samo na području koje je prikazano na karti, stoga bi u situaciji kada smo pri krupnijim razinama mjerila postavljeni iznad nekog manjeg područja došlo do dobivanja samo djelomičnih ili nikakvih rezultata pretraživanja.

Korak koji zatim slijedi je dodavanje *WFS* sloja koji će se koristiti za prikazivanje filtriranog sadržaja, a njegov kod je slijedeći:

```
wfs_sloj = new OpenLayers.Layer.Vector("WFS sloj", {
    strategies: [new OpenLayers.Strategy.BBOX()],
    protocol: new OpenLayers.Protocol.WFS({
        version: "1.1.0",
        srsName: "EPSG:900913",
        url: "http://localhost:81/geoserver/wfs",
        featureNS: "aTopo",
        featureType: $('#izbornik_kategorija :selected').attr('value')
    }),
    styleMap: new OpenLayers.StyleMap({
        "default": new OpenLayers.Style({
```

```
                pointRadius: 5,
                fillColor: "#ff9933",
                strokeWidth: 2,
                strokeColor: "#ff0000",
                graphicZIndex: 1
            })
        }),
        filter: new OpenLayers.Filter.Logical({
            type: OpenLayers.Filter.Logical.OR,
            filters: [
                new OpenLayers.Filter.Comparison({
                    type: OpenLayers.Filter.Comparison.LIKE,
                    matchCase: false,
                    property: "NAME",
                    value: "*" + document.getElementById('traziTekst').value + "*"
                }),
                new OpenLayers.Filter.Comparison({
                    type: OpenLayers.Filter.Comparison.LIKE,
                    matchCase: false,
                    property: "NAME_HR",
                    value: "*" + document.getElementById('traziTekst').value + "*"
                })
            ]
        })
    });
```

Dakle, kod u prvoj liniji govori da se stvara novi vektorski sloj sa nazivom "WFS sloj", a zatim mu se definiraju slijedeći parametri: *strategies*, *protocol*, *styleMap* i *filter*.

Strategies je element koji definira na koji način sloj dohvaća podatke, stoga parametar *OpenLayers.Strategy.BBOX()* navodi da se podaci sa poslužitelja dohvaćaju samo za područje koje se nalazi u vidnom polju, tj. koje je u trenutku pokretanja ove funkcije prikazano u okviru karte.

Element *protocol* definira parametre upita za dohvaćanje prostornih podataka sa poslužitelja. U ovom slučaju je definirano da se koristi *WFS* protokol, verzije *1.1.0* , a podaci se dohvaćaju u *Google Mercator* projekciji (*EPSG:900913*). Zatim se navodi adresa poslužitelja, te naziv prostora (*Workspace*-a) koji je korišten za skladištenje podataka. Posljednji element koji treba definirati je naziv sloja iz kojeg će se podaci dohvaćati. U ovom slučaju se ta vrijednost ne navodi doslovno, nego se pomoću postavljene funkcije ona automatski preuzima iz elementa sa identifikatorom *izbornik_kategorija*, tj. iz padajućeg izbornika kategorija koji se nalazi u rubrici za pretraživanje u sučelju samog servisa.

Pomoću elementa *styleMap* se definira stil koji određuje na koji način će se *WFS sloj* prikazivati na karti. U konkretnom slučaju je definirano da će se traženi objekti na karti označavati kružnom oznakom radijusa 5 piksela sa obrubom debljine 2 piksela i odgovarajućom bojom.

Posljednji element je *filter* koji je zapravo glavni alat za pretraživanje sadržaja karte. U konkretnom slučaju je, zbog specifičnosti podataka, korištena kombinacija više vrsta filtera. Prvi po redu je logički *ili* filter (*OpenLayers.Filter.Logical.OR*), koji ima funkciju da proslijeđuje podatke filtriranja, koji zadovoljavaju uvjet jednog od unutarnja dva filtera. Spomenuti unutarnji filteri su tzv. filteri

usporedbe (*OpenLayers.Filter.Comparison*), koji provjeravaju da li je pojam, koji je element pretraživanja sličan (*OpenLayers.Filter.Comparison.LIKE*) nekom od zapisa koji se nalazi spremljen u unaprijed zadanom sloju podataka, i to u atributnim stupcima NAME, odnosno NAME_HR. Pojam koji se pretražuje se automatski povlači iz polja za upis, koji se nalazi u rubrici za pretraživanje u sučelju samog servisa. Kako bi sama funkcija za pretraživanje bila što jednostavnija, isključena je opcija koja razlikuje unos velikih i malih slova u pretraživanju. Isto tako je za pretraživanje dovoljno upisati samo dio pojma koji želimo pronaći, jer će filter vratiti sve pojmove koji su mu slični, pa iz samih ispisanih rezultata možemo odabrati nama odgovarajući rezultat.

Nakon što smo definirali način na koji će se podaci pretraživati i prikazivati na karti, provedena je još implementacija funkcionalnost koja dobivene rezultate zapisuje u za to predviđeno polje, pomoću koda u nastavku:

```
wfs_sloj.events.register('loadend', wfs_sloj, function (evt) {

    $("#ispis").empty();

    rezultati = wfs_sloj.features;

    $.each(rezultati, function(index, value) {
        $("#ispis").append('<li><a href="#' + value.fid + '" class="feature_link">'
            + value.attributes['NAME_HR'] + '</a></li>');
    });

    $('.feature_link').on('click', function (evt) {
        evt.preventDefault();
        var myFid = evt.currentTarget.href.split('#')[1];
        $.each(rezultati, function (index, value) {
            if (value.fid == myFid) {
                var koordinate = new OpenLayers.LonLat(
                    wfs_sloj.getFeatureByFid(myFid).geometry.x,
                    wfs_sloj.getFeatureByFid(myFid).geometry.y
                );
                map.setCenter(koordinate, 4);
            }
        });
    });
});

map.addLayer(wfs_sloj);
```

Prethodna implementacija radi na principu praćenja događaja, pa iz definicije u prvom retku vidimo kako ona počinje s radom tek kad se završi (*loadend*) filtriranje i generiranje *WFS sloja*. Prva funkcija koju zatim izvršava je brisanje sveg sadržaja iz liste sa identifikatorom *ispis*, koja se nalazi u polju za ispis rezultata. Nakon toga se stvara varijabla *rezultati*, koju će činiti sva obilježja novostvorenog WFS sloja. Iz nje se zatim automatski generiraju poveznice svakog obilježja i zapisuju se u listu *ispis*. Svakom poveznici se dodjeljuje naziv koji zapisan pod *NAME_HR* atributom. Nakon toga je još izrađena funkcija koja stvara relaciju između poveznica u tablici i lokacije svakog obilježja na karti. Njena konačna funkcija je da se prilikom odabira pojedine poveznice, njena lokacija automatski postavi u centar karte. Na kraju je još samo potrebno dodati na kartu generirani *WFS sloj*.

Budući da je funkcija *pretrazi()* već prethodno implementirana, kao događaj koji se pokreće prilikom pritiska na gumb *Traži*, nije potrebna dodatna intervencija u sami kod korisničkog sučelja, te je moguće napraviti spremanje svih datoteka i provesti testiranje rada samog servisa u web pregledniku.

6. Zaključak

Ovim radom je prikazan način izrade relativno jednostavnog geoinformacijskog web servisa primjenom besplatnih alata otvorenog koda (*Geoserver*, *OpenLayers* …). Korištenjem standardnih funkcija za manipulaciju prostornim podacima postignuto je veliko rasterećenje samih programerskih zahtjeva kako u količini samog ispisanog koda, tako i u uštedi na vremenu. Krajnjim postignućem samog rada ostvaren je glavni cilj i napravljen je, u kartografskom smislu, proizvod koji se može uspoređivati sa već postojećim komercijalnim i daleko popularnijim proizvodom, *AdriaTOPO* kartom. Iako još nema implementirane sve funkcionalnosti koje nudi spomenuta karta, rad zapravo nudi čvrste temelje za proširivanje ponude i na još jednom mediju, Internetu.

Samim radom je do sada ostvarena mogućost pregledavanja karte pomoću integriranih kontrola, te mogućnost osnovnog pretraživanja sadržaja i isticanja lokacija dobivenih rezultata na karti. Neke od planiranih funkcija, koje nisu realizirane u samom radu, su naprednija mogućnost pretraživanja sadržaja, npr. na način da se prilikom upisivanja pojma, u tekstualno polje za pretraživanje, korisniku automatski predlažu pojmovi iz baze podataka koji sadrže upisani niz znakova (tzv. *autocomplete*). Daljnjim razvojem bi se također mogli integrirati izbornici koji bi omogućavali pretraživanje, odnosno prikazivanje sadržaja samo pojedinih tipova objekata (npr. hotela, bolnica, banaka i dr.).

Neke od prednosti ovog rada su to što je izrađen za korištenje putem Interneta, koji predstavlja veliki potencijal jer širokom krugu korisnika omogućava jednostavan pristup proizvodu, bilo u prezentacijske svrhe u kojima se korisnici samo upoznaju sa njime prije kupnje, bilo u informativne gdje samo pregledavaju sadržaj. Drugi veliki potencijal ovog rada je i njegova fleksibilnost, te mogućnost korištenja vrlo vrijednih podataka u potpuno nove svrhe, pri čemu je jedan od mogućih pravaca razvoja izrada karata prilagođenih individualnim potrebama korisnika (npr. turisti, izletnici i sl…).

Literatura

Hazzard, E. (2011): OpenLayers 2.10 Beginner's Guide

URL 1. Topografska kartografija, http://www.garmin.com.hr/Products/Details.aspx?productID=711&kultura=hr , 05.09.2012.

URL 2. XML, http://hr.wikipedia.org/wiki/XML, 01.09.2012.

URL 3. Styled Layer Descriptor, http://en.wikipedia.org/wiki/Styled_Layer_Descriptor, 01.09.2012.

URL 4. JavaScript, http://hr.wikipedia.org/wiki/JavaScript, 24.05.2012.

URL 5. WMS reference, http://docs.geoserver.org/2.1.3/user/services/wms/reference.html, 25.05.2012.

URL 6. WFS basics, http://docs.geoserver.org/2.1.3/user/services/wfs/basics.html, 29.08.2012.

URL 7. Shapefile, http://en.wikipedia.org/wiki/Shapefile, 25.05.2012.

URL 8. GeoServer User Manual, http://docs.geoserver.org/stable/en/user/index.html, 25.05.2012.

URL 9. OpenLayers, http://en.wikipedia.org/wiki/OpenLayers, 30.08.2012.

URL 10. jQuery, http://en.wikipedia.org/wiki/JQuery, 30.08.2012.

URL 11. Points, http://docs.geoserver.org/2.1.3/user/styling/sld-cookbook/points.html, 01.09.2012.

Prilozi

Sadržaj priloženog medija

Br.	Ime datoteke	Opis
1.	Diplomski_rad.pdf	Tekst diplomskog rada
2.	Geoserver	Mapa koja sadrži Geoserver aplikaciju sa izvornim podacima korištenim za izradu rada
3.	Geoserver\data_dir\data\istra	Mapa koja sadrži izvorne podatke za izradu rada
4.	Geoserver\data_dir\styles	Mapa koja sadrži stilove za prikaz podataka na karti
5.	Izvorni_kod	Mapa koja sadrži izvorni kod web sučelja i implementiranih funkcija
6.	Struktura_podataka.pdf	Struktura podataka i kartografski znakovi