Sveučilište u Zagrebu – Geodetski fakultet
University of Zagreb – Faculty of Geodesy
Katedra za geoinformatiku
Chair of Geoinformation Science
Kačićeva 26, 10000 Zagreb, Croatia
Web: http://geoinfo.geof.hr; Tel.: +385 (1) 46 39 227; Fax: +385 (1) 48 26 953

Diplomski rad

Implementacija WFS-T standarda na primjeru webGIS-a Rovinjskog sela

Izradio: Damir Kontrec

Mentor: prof. dr. sc. Damir Medak

Zagreb, rujan 2010.

Implementacija WFS-T standarda na primjeru webGIS-a Rovinjskog sela

Sažetak: *Ovaj rad opisuje postupak izrade webGIS aplikacije generalnog urbanističkog plana Rovinjskog sela. Za njezino se funkcioniranje koristi WFS-T standard za rukovanje prostornim podacima na internetu. U izradi aplikacije su korišteni skupovi gotovih alata OpenLayers i GeoExt. Njihovim korištenjem je pojednostavljeno i ubrzano razvijanje prije spomenute aplikacije, napisane korištenjem programskog jezika JavaScript. Kao rezultat, razvijena je aplikacija kojom se mogu pregledavati i izmjenjivati geometrija i atributi zona generalnog urbanističkog plana. Izmjene navedenih prostornih podataka se potom šalju na udaljeni poslužitelj korištenjem prethodno spomenutog WFS-T standarda.*

Ključne riječi: *WFS-T, webGIS, otvoreni kod, OpenLayers, GeoExt*

Implementing WFS-T standard on example of Rovinjsko selo webGIS

Abstract: *This paper describes the process of writing a webGIS application for manipulating master urbanistic plan of town of Rovinjsko selo. This application implements WFS-T standard for handling geospatial data. OpenLayers and GeoExt JavaScript frameworks are used for accomplishing faster and simpler development of such an application. As a result, a webGIS application was produced, which can then be used to view and edit geometry and attributes of master plan's zones. Changes made to geospatial data by using this application are then sent to a remote sever using before mentioned WFS-T standard.*

Keywords: *WFS-T, webGIS, open source, OpenLayers, GeoExt*

Sadržaj

1. Uvod

Razvojem interneta i informatike počinju se primjećivati promijene u svakoj struci zastupljenoj u današnjem društvu. Primjerice, u geodeziji i geoinformatici je sve češća upotreba geoinformacijskih sustava (skraćeno *GIS*), tj. aplikacija koje omogućuju pregledavanje i izmjenu prostornih podataka. Ovakve aplikacije su uglavnom komercijalne prirode i često veoma skupe, te ih krajnji korisnik pribavlja samo ako će od njih na kraju imati veliku financijsku korist. Također, pojam *webGIS*-a, tj. internet aplikacije koja omogućuje rad s prostornim podacima (koji se također preuzimaju s interneta) nije dovoljno zastupljen, iako nudi brojne prednosti pred klasičnim *GIS* aplikacijama. Primjerice, za korištenje *webGIS*-a je u većini slučajeva potrebno posjedovati samo uređaj koji sadrži internet preglednik, te mu je omogućeno spajanje na internet.

Zato, kao prvi korak prema razvoju geo-osposobljenog društva, svakog potencijalnog korisnika bi trebalo upoznati s prednostima (*web*)*GIS*-a. Kako bi ih se motiviralo za korištenje ovakve vrste aplikacija, nužno je za početak sniziti njihovu cijenu. U postizanju tog cilja, potrebno je smanjiti financijske izdatke, utrošeno vrijeme i ostale resurse korištene u razvoju ovakve aplikacije. Ovdje na red stupaju tehnologije otvorenog koda (eng. *Open source technologies*), koje omogućuju puno brži (a samim time i puno jeftiniji) razvoj, između ostalih, i *webGIS* aplikacija. Također, veliku ulogu u cijelom procesu imaju i standardi, koji omogućuju jasno definirani način razmjene, primjerice, prostornih podataka.

Ovaj rad prati primjer razvoja jednostavnije *webGIS* aplikacije, pisane isključivo korištenjem tehnologija otvorenog koda. Prije spomenute vrste tehnologija korištene u izradi ove aplikacije uključuju *JavaScript* programski jezik, te *OpenLayers* i *GeoExt* skupove gotovih programskih alata (eng. *frameworks*). One svojim gotovim funkcijama uvelike smanjuju broj redaka koda potrebnog za implementaciju funkcionalnosti aplikacija pisanih pomoću njih. Sve prostorne podatke kojima ova aplikacija barata poslužuje *Geoserver* (također aplikacija otvorenog koda), a njima se pristupa, te ih se modificira, korištenjem *WFS-T* standarda za razmjenu prostornih podataka.

2. Korištene tehnologije

Prilikom izrade *webGIS* aplikacije generalnog urbanističkog plana Rovinjskog sela, korištene su različite tehnologije otvorenog koda. Primjerice, aplikacija je pisana u programskom jeziku *JavaScript*, a za razmjenu informacija između aplikacije i poslužitelja prostornih podataka korišten je *WFS-T* standard, jedan od mnogih standarda koje je propisala neprofitna organizacija za standardizaciju *Open Geospatial Consortium*. Korištene tehnologije opisane su u nekim od sljedećih potpoglavlja.

2.1. HyperText Markup Language

HTML (*HyperText Markup Language*) je vrlo raširen opisni jezik korišten za izradu strukture dokumenata (uglavnom internet stranica). Korištenjem oznaka (eng. *tags*), omogućuje se pridjeljivanje semantičkog značenja naslovima, odlomcima teksta, poveznicama, popisima i ostalim elementima tekstualnih datoteka. Osim svega prije navedenog, *HTML* omogućava uključivanje slika, te video, zvukovnih i sličnih zapisa u same tekstualne dokumente, kako bi se povećala njihova interaktivnost (URL 1).

Prve ideje o izradi *HTML*-u sličnog jezika javile su se krajem 1980-tih godina. Britanski inženjer fizike, Tim Berners-Lee, tada je predložio stvaranje sustava koji će znanstvenicima omogućiti korištenje i razmjenu dokumenata međusobno referenciranih poveznicama (eng. *hypertext*) putem interneta. Koncept *HTML*-a je vrlo sličan *SGML* (eng. *Standard Generalized Markup Language*) općenitom opisnom jeziku, koji je bio propisan kao standard *ISO* (eng. *International Standards Organization*) organizacije, još od sredine 1980-tih godina (URL 1).

HTML jezik od 1996. godine održava i razvija *W3C* (*eng. World Wide Web Consortium*) organizacija za internet standardizaciju. 2000., *HTML* također postaje međunarodni *ISO* standard, na temelju zadnje revizije iz 1999., koja nosi oznaku *HTML* 4.0.1 (URL 1). Prije opisani jezik će se u ovome radu koristiti za izradu osnovnog rasporeda elemenata internet stranice koja će sadržavati *webGIS* aplikaciju.

2.2. JavaScript programski jezik

JavaScript (podskup *ECMAScript* standarda) (URL 2) *j*e skriptni programski jezik, nastao sa svrhom omogućavanja izrade dinamičkih internet aplikacija koje se izvršavaju na korisnikovom računalu (eng. *client-side scripting programming language*). Pojavom *World Wide* Web servisa početkom 1990-tih, sadržaj (pisan uglavnom u *HTML*-u) postavljen na njega nije pružao nikakvu mogućnost interakcije s korisnikom. Kako bi se omogućilo dinamičko izmjenjivanje sadržaja i izgleda internet stranica, bilo je potrebno uvesti skriptni programski jezik, koji bi upravljao internet stranicom ovisno o korisnikovim željama.

U počecima razvoja *World Wide Web*-a, dva pretraživača su držala dominaciju na tržištu – *Netscape Navigator* i *Microsoft Internet Explorer*. Tvrtka *Netscape* je 1995. prva predstavila programski jezik koji je omogućio interaktivno izmjenjivanje internet stranica i prozvala ga *LiveScript*. Svrha mu je bila omogućiti jednostavnije skriptiranje dinamičkih internet stranica (u usporedbi s programskim jezikom *Java*, koji se pojavio u istom vremenskom razdoblju, a također je mogao poslužiti istoj svrsi). Isto tako, omogućavao je izvršavanje programskog koda bez kompiliranja (eng. *compiling*). Zbog ove je činjenice *LiveScript* skriptni programski jezik, a prevoditelj (eng. *interpreter*) za njega je integriran u sam pretraživač (URL 2).

Zbog tadašnje popularnosti programskog jezika *Java* i činjenice da je *LiveScript* nastao s namjenom da isti zamjeni na internetu, tvrtka *Netscape* mijenja ime svojeg programskog jezika u *JavaScript*, u dogovoru s tvrtkom *Sun* (danas dio *Oracle*-a), kreatorom *Java*-e (URL 3). Zbog svojeg imena, *JavaScript* danas uzrokuje često miješanje s programskim jezikom *Java*, iako se ta dva jezika uvelike razlikuju. Jedina im je sličnost sintaksa, zato što su oba jezika bila inspirirana sintaksom programskog jezika *C* (URL 4).

Microsoft je potaknut razvojem *Netscape*-ovog *JavaScript*-a ubrzo u svoj internet pretraživač implementirao podršku za dva programska jezika za izradu interaktivnih internet stranica. Jedan se zvao *VBScript*, te se temeljio na *BASIC* programskom jeziku, ali do danas nije značajnije zaživio. Drugi je jezik, nazvan *Jscript*, bio je veoma sličan *JavaScript*-u. Ova sličnost proizlazi iz činjenice da je u to vrijeme *Netscape Navigator* bio češće korišteni internet preglednik od *Internet*

Explorer-a, te je *JScript* svakom novom verzijom bio sve sličniji *JavaScript*-u, kako bi omogućio pokretanje istog koda u oba internet preglednika.

Kako je *Netscape* vidio sve veću prijetnju *Microsoft*-ovim razvojem *JScript*-a, svoju verziju programskog jezika je predao 1996. godine Međunarodnom tijelu za standardizaciju – *ECMA* (eng. *European Computer Manufacturers Association*). Ova organizacija je tada postala odgovorna za daljnji razvoj jezika. Kao rezultat ove činjenice, jezik je službeno preimenovan u *ECMAScript* (ili *ECMA-262*), ali je u upotrebi ostao korišten stari naziv *JavaScript* (URL 2).

Zbog činjenice da se izvodi na korisnikovom računalu, kod pisan u *JavaScript*-u omogućuje vrlo brzo izvršavanje korisničkog sučelja i same aplikacije, što korisniku može odavati dojam da se radi o klasičnoj stolnoj aplikaciji (eng. *desktop application*), umjesto o dinamičkoj internet stranici. Kao rezultat ove činjenice, razvijeni su mnogi skupovi gotovih funkcija (eng. *frameworks*), koji uvelike ubrzavaju i olakšavaju izradu internet aplikacija, poput *OpenLayers*-a ili *GeoExt*-a, koji su obrađeni u ostatku ovog rada.

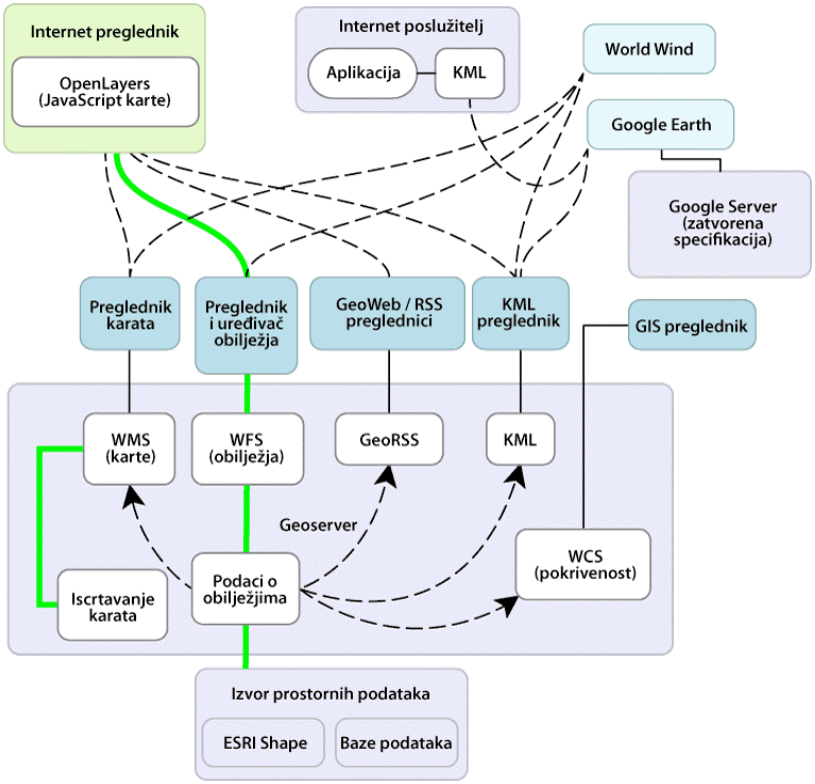
JavaScript posjeduje i nekoliko sigurnosnih odlika, kao što je, primjerice ograničeno izvođenje aplikacija (eng. *sandboxing*). Ovo svojstvo onemogućuje izvršavanje radnji koje manipuliraju podacima na korisnikovom računalu, kao što je npr. stvaranje ili brisanje datoteka, te pristup osjetljivim podacima poput korisničkih imena, lozinki ili *cookies*-a (tekstualne datoteke koje sadržavaju informacije o pristupu pojedinim internet stranicama). Ograničeno izvršavanje aplikacija se može isključiti s korisnikovim dopuštenjem, te je tada moguće modificirati datoteke na njegovom računalu.

JavaScript programski jezik je u ranom periodu svojeg razvoja bio poznat po mnogim sigurnosnim manama i nedostatku podržavajuće programske okoline (eng. *IDE – Integrated Development Enviroment*), te ostalih programskih alata. Kao takav je bio smatran presličnim ostalim programskim jezicima, te nedovoljno moćnim za razvoj ozbiljnijih aplikacija (URL 3). Bez obzira na prije spomenute činjenice, *JavaScript* je danas najčešće korišten skriptni programski jezik koji služi kao podloga za razvoj brojnih modernih dinamičkih internet aplikacija koje se izvršavaju na korisnikovom računalu.

2.3. Važniji OGC standardi

OGC ili punim imenom *Open Geospatial Consortium* (Udruženje za otvorenost prostornih podataka) je neprofitna standardizacijska organizacija nastala 1994. godine. Okuplja preko 370 komercijalnih, vladinih, neprofitnih i istraživačkih organizacija sa svih strana svijeta. Svrha ovog udruženja je poticanje razvoja i implementiranja standarda za rukovanje prostornim podacima i njihovu razmjenu, servisima za rad s njima i slično. Ovo je udruženje nastalo na temeljima *OGF*-a (eng. *Open Grass Foundation*), osnovanog 1992, a koji je u razdoblju od 1994. do 2004. godine nosio ime *Open GIS Consortium* (URL 5).

Većina standarda propisanih od strane *OGC*-a ovisi o grubo definiranom ustroju opisanom u skupu dokumenata zajedničkim imenom zvanih *Abstract Specification* (hrv. Opći tehnički opis). Ovi dokumenti opisuju osnovni podatkovni model za predstavljanje geografskih obilježja, te se na njima temelji niz specifikacija koje služe interoperabilnosti (omogućavanju međusobnog rada) različitih tehnologija koje se bave prostornim podacima i lociranjem.



Slika 1. Povezanost i korištenje nekih standarda nastalih prema specifikaciji OGC-a (URL 5)

Neke od specifikacija propisanih od strane *OGC*-a su *WMS*, *WMS-C*, *TMS* i *WFS* protokoli, *GML* i *KML* opisni jezici, i sl. Prije navedene specifikacije su opisane u ovom ili detaljnije objašnjene u ostalim poglavljima ovog rada. Međusobna povezanost i način korištenja ovih tehnologija može se vidjeti na dijagramu prikazanom na slici (Slika 1).

2.3.1. Web Map Service

WMS (eng. *Web Map Service* – Mrežni kartografski servis) je protokol koji služi posluživanju georeferenciranih kartografskih slika preko interneta. Ove slike su rasterskog oblika, generirane od strane poslužitelja na temelju podataka dohvaćenih iz prostornih baza podataka (originalni prostorni podaci su u vektorskom obliku) ili nekog drugog oblika prostornih podataka (URL 6). Mogu se koristiti, primjerice, kao pozadinske slike za preklapanje s drugim slojevima koji sadrže prostorne podatke prilikom, primjerice, izrade *GIS* aplikacije.

Prva probna verzija ovog protokola je razvijena i izdana od strane *OGC*-a krajem 1999. godine, a trenutna verzija (1.3.0) izdana je početkom 2004. godine (URL 6). Ovaj protokol propisuje parametre upita koji se koriste prilikom dohvaćanja rasterskih slika i komunikacije s poslužiteljem:

- ***GetCapabilities*** – vraća mogućnosti poslužitelja (postavke *WMS*-a), i popis dostupnih slojeva (eng. *layers*).
- ***GetMap*** – vraća rastersku sliku karte ovisno o definiranim parametrima u upitu (zatraženi slojevi, koordinate vrhova graničnog okvira, i slično).

Osim gore navedenih parametara upita, ovisno o implementaciji, *WMS* poslužitelj može prihvaćati i sljedeće upite: *GetFeatureInfo* (dohvaćanje atributnih podataka o pojedinim obilježjima), *DescribeLayer* (dohvaćanje dodatnih podataka o sloju) i *GetLegendGraphic* (dohvaćanje legende kartografskog prikaza) (URL 7).

Također, prilikom dohvaćanja podataka korištenjem *WMS* upita potrebno je definirati još neke parametre: *Layers* (odabir željenih slojeva u prikazu), *Styles* (kombiniranje stilova za vizualizaciju prostornih podataka), *SRS* (definiranje željenog koordinatnog sustava), *Height* i *Width* (visina i širina izlazne slike), *Format* (podatkovni oblik dostavljene slike), i drugo (URL 7).

WMS protokol je implementiran u veliki broj aplikacija, kako komercijalnih, tako i onih otvorenog koda. Dvije najčešće korištene implementacije *WMS*-a otvorenog koda su *Geoserver* (opisan u jednom od sljedećih poglavlja) i *Mapserver*. Od komercijalnih, poznatije su *ArcGIS* i *GeoMedia* aplikacije za izradu *GIS*-a.

Primjer jednostavnog *WMS* upita (razdvojenog u retke radi preglednosti):

```
1.  http://localhost:8080/geoserver/wms?
2.  service=wms&
3.  version=1.3.0&
4.  request=GetMap&
5.  layers=rovinjsko-selo:gup&
6.  bbox=13.706,45.1,13.728,45.114&
7.  width=800&
8.  height=700&
9.  format=image/png
```

Kao rezultat ovog upita (upisanog u adresnu traku internet pretraživača), prikazati će se rasterska slika zatraženog područja (pod pretpostavkom da je postavljen poslužitelj prostornih posataka).

Prvi redak upita sadrži adresu poslužitelja s kojeg se dohvaćaju prostorni podaci. Od ostatka upita prvi je redak (za razliku od ostalih) odvojen znakom upitnika "?". Drugi redak sadrži ime korištenog protokola, a treći navodi verziju korištenog protokola. Oba su retka (kao i svi ostali retci poslje njih) od ostatka upita odvojeni znakom "&".

Četvrti i peti redak definiraju tip upita (*GetMap* – dohvaćanje rasterske slike karte) i sloj koji se želi dohvatiti (u ovom slučaju sloj koji sadrži Generalni urbanistički plan Rovinjskog sela). Šesti redak navodi granični okvir (eng. *bounding box*), u vidu koordinata donjeg lijevog i gornjeg desnog vrha.

Sedmi i osmi redak preciziraju širinu i visinu rasterske slike, tj. njezinu prostornu rezoluciju. Potrebno je napomenuti da ako odnos širine i visine (eng. *aspect ratio*) dohvaćane rasterske slike u pikselima ne odgovara odnosu širine i visine graničnog područja, kao rezultat se dobiva rasterska slika krivih proporcija. Zadnji

redak poslužitelju definira podatkovni oblik zatražene rasterske slike. U ovom slučaju se koristi *PNG*[1] podatkovni oblik.

2.3.2. WMS Tile Caching

WMS-C (punim imenom *WMS Tile Caching*) standard je nadogradnja na *WMS* protokol, a njegova je svrha ubrzano dostavljanje rasterskih slika kartografskih prikaza određenog područja (URL 8). Ovo se postiže korištenjem već spremnih slika (eng. *prerendered cached images*), koje obično pokrivaju dosta manje područje nego korisnik zahtjeva. Kombiniranjem većeg broja slika u pravokutnu mrežu postiže se pokrivenost većeg područja (eng. *tiling* ili *tessellation* - popločenje).

Iako ova metoda uvelike ubrzava dostavljanje slika prostornih podataka krajnjem korisniku, ona ima i određena ograničenja koja se javljaju zbog njezinog načina rada. Zato što su slike spremne, korisnik prilikom upita ne može mijenjati izgled slika definiranjem dodatnih argumenata u upitu. Primjerice, onemogućeno je vizualiziranje prostornih podataka drukčije od već definiranog, izlazni podatkovni oblik je fiksan, ne mogu se skrivati ili prikazivati slojevi prema želji korisnika, širina i visina slika su također predefinirani, koristi se isključivo *WGS 84* datum, i slično. Također, slike su spremane samo na određenim uvećanjima (eng. *zoom level*), te je zato moguće koristiti samo prethodno definirane promjene mjerila.

U slučaju da *WMS-C* posrednički poslužitelj (eng. *proxy server*) primi puni *WMS* zahtjev (sa definiranim svim argumentima u upitu, poput *Layers*, *Styles*, *SRS*, *Format* parametara i slično), isti vraća poruku o grešci ili preusmjerava upit na klasični *WMS* poslužitelj (URL 8). U tom slučaju se gubi brzina dostavljanja kartografskih slika.

WMS-C protokol koristi primjerice *NASA*-in *World Wind* servis za prikaz satelitskih snimaka zemljine površine. Također, servisi poput *Google Maps*-a, rade na način

---

[1] ***PNG*** (*Portable Network Graphics*) – rasterski oblik zapisa slike koji podržava sažimanje bez gubitka kvalitete podataka.

sličan *WMS-C* protokolu, te tako omogućuju brzo i efikasno dostavljanje slika prostornih podataka velikom broju korisnika istovremeno.

2.3.3. Tile Map Service

TMS (eng. *Tile Map Service*) je standard propisan od strane *OGC*-a i omogućava pristup unaprijed pripremljenim (eng. *cached*) slikama georeferenciranih podataka. Ovaj standard definira način na koji korisnik zahtjeva slikovne jedinice (eng. *tiles*), te se njihovim slaganjem u pravokutnu mrežu postiže kartografski prikaz pojedinog područja (URL 9).

Korištenjem ovog protokola, slično kao i korištenjem *WMS-C* protokola, korisnik dohvaća slikovne jedinice koje su već unaprijed pripremljene na poslužitelju. Slikovne jedinice su dostupne samo u prethodno definiranim mjerilima, sadržavaju unaprijed određene slojeve, pripremljene su u unaprijed definiranoj projekciji i slično. Primjerice, promjenom mjerila, korisnik preuzima drugi niz slikovnih jedinica. To znači da poslužitelj ne priprema prikaz prostornih podataka nakon korisnikovog zahtjeva, nego poslužuje već pripremljen skup slika.

Prednost ovakvog načina posluživanja prostornih podataka je brzina dostavljanja slike nekog područja (što se lako uočava ako uslugu koja se temelji na ovom protokolu istovremeno koristi veliki broj korisnika). No, nedostatak ovog standarda je njegova nefleksibilnost, tj. nemogućnost prilagođavanja prikaza prostornih podataka ovisno o korisnikovim željama.

2.3.4. Keyhole Markup language

KML (eng. *Keyhole Markup Language*) je opisni jezik temeljen na *XML*[2] opisnom jeziku, te omogućuje modeliranje i pohranu prostornih podataka u obliku točaka, (poli)linija, poligona, rasterskih slika (snimaka) i trodimenzionalnih modela.

---

[2] ***XML*** (*eXtensible Markup Language*) – nadogradivi opisni jezik. Nadograđivanje jezika (ovisno o primjeni) se vrši korištenjem tzv. *schema* datoteka koje definiraju pravila za oblikovanje podataka.

Također, korištenjem *KML*-a omogućeno je spremanje i mjesnih oznaka (eng. *landmarks*), njihovog opisa u *HTML* jeziku, opisnih stilova za vizualizaciju geometrije i slično (URL 10). Za referentni sustav, *KML* opisni jezik koristi *WGS84* datum, a kojim se definira geografska širina i dužina, te nadmorska visina (pri definiranju se nužno navode tim redoslijedom). Ako se prilikom navođenja ovih parametara ne navede nadmorska visina, ona se postavlja na vrijednost nula (približno razina mora).

KML je u početku bio razvijan sa svrhom upotrebe u *Google Earth* geopregledniku (koji se u ranijim verzijama izdanim prije 2004. godine zvao *Keyhole Earth Viewer*). *OGC* je sredinom 2008. *KML* opisni jezik prihvatio kao službeni standard, te se tada pokrenuo razvoj i ostalih preglednika *KML* datoteka, poput *Marble* geopreglednika (eng. *geobrowser*) otvorenog koda (URL 11).

2.4. Web Feature Service - Transactional

Web Feature Service (skraćeno *WFS*) je protokol koji omogućuje dohvat i izmjenu prostornih podataka preko interneta koristeći standardizirane upite (URL 12). Održava ga i razvija, kao i ostale prethodno navedene standarde, *OGC* standardizacijska organizacija. Postoje mnoge komercijalne *WFS* implementacije, kao i implementacije otvorenog koda, među koje spada i *Geoserver* poslužiteljska aplikacija, korištena u izradi ovog rada.

Za razliku od *WMS* protokola, koji kao rezultat svakog upita poslužuje rastersku sliku traženog područja (slično kao i mnogi popularni servisi za pružanje kartografskih prikaza na internetu), *WFS* protokol praktički omogućava dohvaćanje "izvornog koda" karte. Dohvaćeni su podaci najčešće u obliku vektorskih podataka, tj. geometrije s pripadajućim atributima. Ovakav oblik podataka omogućava izvođenje puno bogatijih analiza u daljnjem radu (izmjena stila vizualizacije prostornih podataka, kombiniranje s drugim prostornim podacima, njihovo preuzimanje na računalo, i slično), umjesto samo pregledavanja slike karte, što je slučaj kod *WMS* protokola (URL 12).

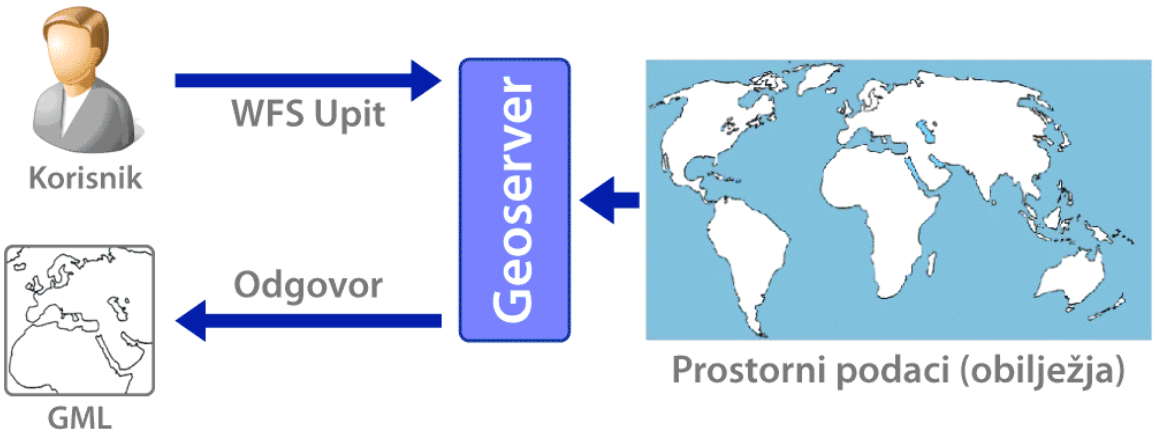
Prostorni podaci preuzeti *WFS*-om se korisniku najčešće dostavljaju u *GML* (eng. *Geography Markup Language*) podatkovnom obliku, koji je izveden iz *XML*-a. Ovaj

zapis omogućuje pohranu svih potrebnih podataka u vidu geometrije (točaka, linija i poligona) s pripadajućim tekstualnim atributnim podacima. *GML* podatkovni oblik je detaljnije obrađen u sljedećem potpoglavlju. Osim *GML*-a, pri posluživanju prostornih podataka *WFS* protokolom mogu se koristiti i *ESRI Shape*, *JSON* (eng. *JavaScript Object Notation*) i ostali podatkovni oblici, no oni se u praktičnoj upotrebi (u usporedbi s *GML*-om) skoro uopće ne koriste.

Osim dohvaćanja prostornih podataka, proširena specifikacija *WFS* protokola omogućuje i njihovu izmjenu u vidu stvaranja novih te brisanja ili izmjene postojećih. Također, moguće je i zaključavanje prostornih podataka, koje se koristi u situacijama kada više korisnika istovremeno pristupa jednom skupu istih.

Ova proširena verzija *WFS* protokola, koja omogućuje gore navedene transakcije, naziva se *WFS-T* (eng. *Web Feature Service - Transactional*). Važno je napomenuti da je kod rada s *Geoserver*-om svaka transakcija nedjeljiva (eng. *atomic*), što znači da ako se samo jedna radnja u cijeloj transakciji ne uspije izvršiti, podaci na poslužitelju ostaju neizmijenjeni.

WFS upit se preko *HTTP* protokola sa korisnikovog računala šalje poslužitelju, koji potom bira koje će informacije posluživati, ovisno o parametrima navedenim u upitu. Poslužitelj potom korisniku šalje odgovor u obliku geometrije s atributima zapisane u, primjerice, *GML* obliku (Slika 2).



Slika 2. Shematski prikaz WFS GetFeature zahtjeva (URL 14).

Trenutna verzija *WFS* protokola je 1.1.0., iako u ovom radu korišteni *Geoserver* omogućava i rad sa starijom verzijom protokola 1.0.0. Između ove dvije verzije

postoje određene razlike – primjerice starija verzija prostorne podatke poslužuje u starijem *GML* 2 obliku podataka, dok novija koristi *GML* 3 oblik, koji na nešto drugačiji način opisuje geometriju. Također, starija verzija *WFS* protokola za razliku od novije verzije ne omogućava promjenu projekcije "u letu", tj. nije moguće prostorne podatke korisniku pružiti u projekciji drukčijoj od one u kojoj su zapisani na poslužitelju. Osim svih navedenih razlika, starija verzija također ima izmijenjen redosljed navođenja geografske širine i dužine prilikom definiranja upita (URL 13). Iako se *WFS* verzije 1.0.0. još i danas koristi, puno je zastupljenija novija verzija protokola.

Primjer i objašnjenje jednostavnog *WFS* upita i pripadajućih parametara u obliku internet adrese (adresa je rastavljena u više redaka radi bolje čitljivosti):

```
1.  http://localhost:8080/geoserver/wfs?
2.  service=wfs&
3.  version=1.1.0&
4.  request=GetFeature&
5.  typeName=rovinjsko-selo:gup
```

Prvi redak navodi internet adresu poslužitelja koji sadrži tražene prostorne podatke. Od ostatka upita ovaj je dio odvojen znakom upitnika "?".

Drugi redak je obavezan i definira protokol koji će se koristiti za dohvaćanje željenih prostornih podataka. U ovome slučaju je riječ o *WFS* protokolu opisanom u ovom poglavlju.

Treći redak je također obavezan ako koristimo *WFS* protokol i definira verziju korištenog protokola, u ovom slučaju radi se o novijoj verziji – 1.1.0.

Četvrti redak definira radnju koju želimo da poslužitelj odradi. U ovom slučaju dohvaćamo prostorne podatke u vidu geometrije i njezinih atributa, te se zato koristi operacija *GetFeature*. Shematski prikaz *GetFeature* zahtjeva prikazan je na slici (Slika 2). Osim ovog, moguće je koristiti i neke od ostalih zahtjeva definiranih *WFS* protokolom (URL 13):

- ***GetCapabilities*** - Dohvaća popis svih podataka s poslužitelja.

- ***DescribeFeatureType*** – Opisuje značajke definiranog obilježja[3].
- ***LockFeature*** - Zaključava obilježje, kako bi se spriječilo njegovo modificiranje. Najčešće se koristi u višekorisničkom radu.
- ***Transaction*** - Izmjenjuje ili briše postojeće podatke ili stvara nove.

Peti redak definira značajku koju želimo dohvatiti. Radnja *typeName* sadrži dva argumenta odvojena znakom dvotočke ":". Prvi argument je ime imenskog prostora[4] u kojem se nalaze traženi prostorni podaci (u ovom slučaju rovinjsko-selo, ime skupa obilježja naselja čiji se generalni urbanistički plan obrađuje). Drugi argument definira ime traženog obilježja (eng. *feature type*, u ovom slučaju skup zona generalnog urbanističkog plana Rovinjskog sela). Imenski prostor i samo ime obilježja se definiraju prilikom uvoza izvornih prostornih u poslužitelj, primjerice *Geoserver*.

Prilikom korištenja *WFS* protokola moguće je koristiti i sljedeće parametre (njihov redosljed navođenja je proizvoljan nakon četvrtog retka, tj. nakon što se u upitu upiše adresa poslužitelja, definira protokol i navede njegova verzija, te odabere željena radnja) (URL 13):

- ***outputFormat*** – Definira oblik posluženih prostornih podataka (*GML 2*, *GML 3*, *ESRI Shape*, *JSON*, itd.).
- ***maxFeatures*** – Ograničavanje broja posluženih obilježja.
- ***sortBy*** – Sortiranje obilježja ovisno o njihovim atributima, kojih se može navesti i više, ali ih se tada odvaja zarezom (dostupno tek od verzije 1.1.0. *WFS* protokola).

---

[3] **Obilježje** (eng. *feature*) – najmanja jedinica prostornih podataka; skup geometrije (točke, linije ili poligona) i pripadajućih atributa.

[4] **Imenski prostor** (eng. *namespace*) - logička cjelina unutar koje su sadržana obilježja . U *Geoserver*-u imenski prostor određuje imenovanje skupa obilježja koji se dohvaćaju WFS upitom.

- ***bbox*** – Definiranje granice obuhvata, koja se definira s dva para koordinata međusobno odvojenih zarezom. Prethodno spomenuta dva para koordinata obilježavaju dijagonalno suprotne vrhove graničnog pravokutnika.

Ovo su samo neki od parametara dostupnih u specifikaciji *WFS* protokola, što svjedoči njegovoj složenosti, dokazuje da zaista služi za rad s prostornim podacima u "otvorenom" obliku (dostupnim svima na korištenje, izmjenu i analizu). Isto tako, prije navedene osobine pokazuju zašto je *WFS* standard osnova za manipulaciju prostornim podacima na internetu.

2.5. Geography Markup Language

Geography Markup Language (skraćeno *GML)* je opisni jezik korišten za prijenos, opisivanje i manipulaciju prostornim podacima. Podskup je *XML* jezika za opis podataka, a izdan je i održavan od strane *Open Geospatial Consortium*-a (URL 15). Zbog jednostavnosti, početne specifikacije *GML* opisnog jezika su ograničavale njegovu upotrebu na opis samo dvodimenzionalne geometrije, ali su se ubrzo pojavile nadogradnje (eng. *extensions*), koje omogućavaju korištenje dvo-i-pol dimenzionalne, te na kraju i trodimenzionalne geometrije.

GML opisuje prostor u obliku geografskih obilježja (eng. *geographical features*). Svako obilježje sadrži popis atributa i geometrije. Atributi podrazumijevaju opisne informacije poput imena, tipa ili primjerice površine određenog obilježja. Geometrija opisuje prostornu komponentu obilježja korištenjem osnovnih geometrijskih elemenata poput točaka, nizova linija (eng. *line strings*) i poligona (URL 16).

Sama definicija *GML*-a omogućuje izradu vrlo kompleksnih obilježja grupiranjem više manjih i jednostavnijih u jedno veće i složenije. Primjerice, složeno obilježje koje opisuje morsku luku se može sastojati od više jednostavnih obilježja koja opisuju cestovni prilaz luci, dokove, zgradu lučke kapetanije i slično.

Kao i kod svakog jezika izvedenog iz *XML*-a, postoje dva dijela *GML* jezika - shema koja opisuje *GML* dokument (eng. *schema file*) i sami *GML* dokument koji sadrži podatke. Ova činjenica omogućuje da se čak i sami korisnici udružuju kako bi nadogradili *GML* jezik i tako ga prilagodili svojim potrebama. U tom slučaju,

korisnici stvaraju primjenske sheme (eng. *application schemas*), upotrebom kojih se prostorni podaci mogu opisati korištenjem pojmova poput stabala, prometnica i naselja, umjesto točaka, linija i poligona.

Neki primjeri primjenskih shema *GML*-a su *CityGML* (namijenjen trodimenzionalnom opisivanju gradskih sredina), *GPML* (*GPlates Markup Language* – služi opisivanju ponašanja tektonskih ploča), *CSML* (*Climate Science Modelling Language* – prilagođen opisivanju klime određenog područja), te brojne druge (URL 17).

Jedna od ključnih osobina *GML*-a je njegova mogućnost referenciranja geografskih obilježja naspram zemljine površine. Trenutna verzija *GML*-a sadrži nadogradivi prostorni referentni sustav (eng. *Spatial Reference System* - *SRS*), koji sadrži sve češće korištene projekcije i referentne okvire u upotrebi danas . To znači da *GML* omogućuje korištenje svih referentnih sustava dostupnih na internet stranici organizacije *EPSG* (eng. *European Petroleum Standards Group*) (URL 15). Također, shema *GML* jezika omogućuje korisniku definiranje vlastitih mjernih jedinica i parametara referentnih sustava.

Zbog prethodno navedenih mogućnosti, *GML* je prilagođeniji radu s prostornim podacima u usporedbi s drugim nadogradnjama *XML* jezika, poput vrlo često korištenog *SVG*-a (*Scalable Vector Graphics*, služi opisu računalne grafike) ili *X3D*-a (podskup *XML*-a za opis trodimenzionalne geometrije). Također, zato što je *GML* jezik izveden iz *XML*-a, datoteke nastale njegovim korištenjem su tekstualne i čitljivog oblika. Kao rezultat ove činjenice, lako je izmijeniti samu *GML* datoteku uz korištenje najjednostavnije aplikacije za manipulaciju tekstom (URL 15).

2.6. ESRI Shape

ESRI Shape je popularni podatkovni oblik koji sadrži (najčešće dvodimenzionalne) prostorne podatke u vektorskom obliku, sa svrhom upotrebe u geoinformacijskim sustavima. Razvijen je od strane tvrtke *ESRI* kao otvoreni podatkovni oblik, kako bi omogućio međusobnu komunikaciju *ESRI*-jevog softvera s ostalim aplikacijama koje rukuju prostornim podacima. Ovaj podatkovni oblik je uveden početkom 1990-tih godina, zajedno s izdavanjem aplikacije *ArcView* 2

(URL 18). Danas ga je moguće pregledavati i uređivati korištenjem i ostalih aplikacija, uglavnom komercijalnih, ali i onih otvorenog koda (primjerice *Quantum GIS*).

Shape datoteka, donekle slično kao i (ali dosta ograničenije od) datoteka u *GML* obliku, opisuje geometriju koristeći točke, polilinije i poligone. Svaki "komad" geometrije također može sadržavati i atributne podatke, ali je tada nužno uz *.shp* datoteku (koja sadržava geometriju) distribuirati i *.shx* (sadrži položajne pokazivače – eng. *indexes* geometrije, kako bi se ubrzalo pretraživanje atributa) te *.dbf* (sadržava atributne podatke) datoteke. Također, uz sve navedene datoteke (koje su nužne za pravilno funkcioniranje ovog podatkovnog oblika), prema potrebi se (najčešće) prilaže još i *.prj* datoteka, koja u tekstualnom obliku sadrži podatke o koordinatnom sustavu, projekciji, datumu, korištenim jedinicama i slično (URL 19). Naravno, ne treba ni napominjati da pritom sve datoteke trebaju biti istog imena. Također, važno je spomenuti da su u svakoj od priloženih datoteka zapisi o obilježjima poredani u redosljedu prema kojem se, primjerice, geometrija iz *.shp* datoteke povezuje s atributima iz *.dbf* datoteke.

No, ovaj podatkovni oblik posjeduje i određena ograničenja. *Shape* datoteka primjerice koristi samo polilinije, odnosno ne podržava krivulje, te se zato na krupnom mjerilu ponekad može uočiti izlomljenost prikaza određenih obilježja. Ovaj problem se može izbjeći povećanjem gustoće točaka u poliliniji, ali se tada znatno može povećati veličina same datoteke. Isto tako, najveća veličina *.shp* datoteke je 2 GB, što znači da se u najboljem slučaju u pojedinačnu datoteku može spremiti najviše 70 milijuna točaka. Također, u jednu *.shp* datoteku se može spremiti samo jedna vrsta obilježja (ili točke ili polilinije ili poligoni, ali ne i kombinirano) (URL 18).

Osim svega navedenog, *.dbf* datoteka, koja skladišti atributne podatke, ne omogućava spremanje praznih polja (umjesto toga prazno polje označava s brojem nula, što može značajno naštetiti primjerice, naknadnim statističkim analizama), maksimalna veličina polja s imenima iznosi samo 10 znakova, najveći broj polja u datoteci je 255, i slično.

3. Korišteni alati

Prilikom izrade *webGIS*-a generalnog urbanističkog plana Rovinjskog sela, upotrijebljen je niz alata koji su ubrzali proces izrade aplikacije. Za posluživanje prostornih podataka korištena je aplikacija otvorenog koda *Geoserver*, s kojom se komuniciralo korištenjem prethodno opisanih *OGC* standarda. Sama *webGIS* aplikacija je izrađena u programskom jeziku *JavaScript*, a pisanje koda su ubrzali skupovi gotovih programskih alata *OpenLayers* i *GeoExt*. Svaki od prije spomenutih alata je pobliže opisan u zasebnom potpoglavlju.

3.1. Geoserver

Geoserver je poslužiteljska aplikacija otvorenog koda napisana u programskom jeziku *Java*, čija je svrha posluživanje i uređivanje prostornih podataka (URL 20). Zbog činjenice da je pisana u *Java*-i, omogućeno je njezino lagano postavljanje i izvršavanje na bilo kojem široko korištenom operativnom sustavu današnjice (*Windows*, različite distribucije *Linux*-a, *Mac OS X*, te slični *unix*-oidni operativni sustavi), što uvelike pridonosi njezinoj popularnosti. Zato što je aplikacija otvorenog koda, izdana pod *GPL* (eng. *General Public License*) licencom, *Geoserver* dokumentira, ispituje, usavršava i nadograđuje široka zajednica ljudi i organizacija diljem svijeta. Za krajnjeg je korisnika u potpunosti besplatna, a kada se koristi u komercijalne svrhe, tada drastično umanjuje cijenu konačnog proizvoda.

Ova poslužiteljska aplikacija je izrađena s kompatibilnošću i iskoristivošću kao glavnim ciljevima. Zato prostorne podatke poslužuje u popularnim i standardiziranim rasterskim i vektorskim slikovnim oblicima kao što su, primjerice, *SVG*, *PNG*, prethodno opisani *GML* i slični. Posluživanje se vrši preko danas vrlo zastupljenih protokola za rad s prostornim podacima, poput *WMS-a*, *WFS-a*, *WCS*-a (*Web Coverage Service*) i ostalih. *WFS-T* protokol, koji je korišten u izradi *WebGIS*-a, je detaljnije obrađen u jednom od prethodnih poglavlja.

Od rasterskih oblika slikovnih zapisa, čije posluživanje preko *WMS* protokola *Geoserver* podržava, trebalo bi spomenuti i *PNG*. Ovaj oblik slikovnog zapisa je razvijen od strane zajednice koja se bavi razvojem aplikacija otvorenog koda, sa

svrhom zamjene zastarjelog *GIF* (*Graphics Interchange Format*) oblika slikovnog zapisa zatvorenog koda. Za razliku od njega, između ostalog nudi i sažimanje podataka bez gubitka kvalitete.

Među vektorske oblike podataka koje, između ostalih, *Geoserver* poslužuje spadaju *GML* i *SVG* zapisi, koji su oboje izvedeni iz *XML*-a. Ovi podatkovni oblici nude veći stupanj interakcije od prije spomenutih rasterskih oblika, zbog činjenice da mogu sadržavati i atributne podatke, te se lagano modificiraju (korištenjem osvnovne aplikacije za uređivanje teksta). *GML*, oblik zapisa korišten u ovome radu, je također detaljnije opisan u jednom od prošlih poglavlja.

Isto tako, *Geoserver* podržava i korištenje ulaznih skupova prostornih podataka u raznim široko rasprostranjenim oblicima. Tako se primjerice koristi *PostgreSQL* prostorna baza podataka otvorenog koda, ili vrlo popularni *ESRI Shape* oblik, koji je uz sve prije nabrojane podatkovne oblike, također bio korišten u izradi *webGIS* aplikacije.

3.2. OpenLayers

OpenLayers je skup gotovih programskih funkcija (eng. *framework*), namijenjenih ubrzavanju pisanja *webGIS* aplikacija u *JavaScript* programskom jeziku. Korištenjem *OpenLayers*-a, prostorni podaci se na relativno lagan i brz način prikazuju u internet pregledniku (URL 21). Korištenje ovog skupa gotovih funkcija omogućuje izradu aplikacija koje svojim radom dodatno ne opterećuju poslužitelj, već se za iscrtavanje dobivenih prostornih podataka koriste resursi korisničkog računala. To znači da su resursi poslužitelja posvećeni isključivo obradi pristiglih zahtjeva i (na temelju njih) posluživanju prostornih podataka.

Drugim riječima, *OpenLayers* ima sličnu svrhu kao i *Google Maps* i *MSN Virtual Earth* sučelja za razvoj aplikacija (eng. *Application Programming Interface – API*), ali za razliku od njih je izdan u obliku otvorenog koda. Zbog toga je za krajnjeg korisnika besplatan i vrlo je široko korišten, te ga podržava i razvija veliki broj individualaca i organizacija diljem svijeta.

Zato što je pisan u *JavaScript* programskom jeziku, omogućeno je izvršavanje aplikacija pisanih njegovim korištenjem u velikoj većini modernih internet preglednika. Samim time, isti kod se može izvršavati i na skoro svim danas široko korištenim operativnim sustavima.

Svrha *OpenLayers*-a je razdvajanje kartografskih alata i prostornih podataka, kako bi svi alati mogli raditi u kombinaciji sa svim podacima. Ova podjela omogućuje neovisnost od komercijalnih proizvoda, koji su u prošlosti često štetili razvoju (i ponekad svojim prestankom izdavanja u potpunosti prekinuli razvoj) različitih *GIS* aplikacija (URL 21).

Ovaj skup gotovih funkcija sadržava podršku za rad s brojnim vektorskim i rasterskim oblicima podataka (*GML*, *PNG*, itd.) te protokolima (*WMS*, *WFS*, i sl.). Ova činjenica ga čini idealnim alatom za korištenje u kombinaciji s *Geoserver*-om, kako bi se razvijale internet aplikacije koje prikazuju, modificiraju i razmjenjuju prostorne podatke.

Razvoj *OpenLayers*-a je započet od strane tvrtke *MetaCarta* (osnovane 1999. godine u Cambridgeu, SAD) (URL 22), te je prva verzija izdana za nešto manje od godine dana razvoja, u obliku otvorenog koda. *OpenLayers* je od studenog 2007. postao projekt *OSGeo*-a (*Open Source Geospatial Foundation*), neprofitne organizacije sa ciljem promoviranja i razvoja tehnologija otvorenog koda za rukovanje prostornim podacima (URL 21).

3.3. GeoExt

GeoExt je skup gotovih funkcija prilagođen izradi sučelja internet aplikacija koje prikazuju prostorne podatke uz pomoć *OpenLayers*-a. Temeljen je na *ExtJS*-u, skupu gotovih funkcija koje drastično ubrzavaju izradu internet aplikacija sa sučeljima sličnim aplikacijama koje se izvršavaju direktno na računalu (eng. *desktop class applications*).

Prednost ovakvog pristupa izradi aplikacije je što se tada sučelje internet aplikacije uvelike (po funkcionalnosti i izgledu) približava sučeljima klasičnih aplikacija. Tako je moguće kompleksnost većeg broja funkcija prikriti njihovim logičnim

rasporedom, na koji su korisnici već navikli rabeći standardne računalne aplikacije s grafičkim korisničkim sučeljima u prošlosti. Znači, internet aplikacije u ovom slučaju posjeduju većini korisnika poznate elemente poput, primjerice, gumba, izbornika, stablastih prikaza organizacije podataka, klizača i slično.

Kao i *OpenLayers*, *GeoExt* i *ExtJS* su pisani u *JavaScript* programskom jeziku. Ova činjenica omogućuje izvođenje aplikacija pisanih korištenjem ovih skupova gotovih programskih alata u skoro bilo kojem internet pregledniku i na skoro bilo kojem operativnom sustavu. Na taj se način uvelike povećava broj potencijalnih korisnika aplikacije, a isto tako se i drastično smanjuje vrijeme njezinog razvoja, koje bi inače bilo utrošeno na njeno prilagođavanje različitim operativnim sustavima.

Kao osnova *GeoExt*-a, *ExtJS* verzije 3.0 je dostupan od srpnja 2009., a 2010. se spojio s *JQTouch* i *Raphaël* skupovima gotovih funkcija za izradu sučelja, te je poprimio novo ime - *Sencha* (URL 23). Spajanjem ovih triju skupova gotovih funkcija omogućena je jednostavna izrada sučelja i za mobilne uređaje (dlanovnike i mobilne telefone), te je se samim time broj uređaja za koje je moguća izrada internet aplikacija poput, primjerice, *webGIS*-a, značajno porastao. Također, ovakav tip aplikacija postaje mnogo korisniji kada se koristi na prijenosnim uređajima i u pokretu.

Sam *GeoExt* skup gotovih programskih alata je izdan pod *BSD* licencom, što mu teoretski omogućuje slobodnu distribuciju koda (u izvršnom i izvornom obliku), ali uz navođenje autorskih prava. No, zbog činjenice da je temeljen na *ExtJS*-u, potrebno je otkupiti komercijalnu licencu dotičnog ukoliko se planira distribucija aplikacije bez otkrivanja izvornog koda (URL 23).

4. Koraci u izradi webGIS aplikacije

Postupak izrade webGIS aplikacije generalnog urbanističkog plana Rovinjskog sela je podijeljen na nekoliko koraka, opisanih u sljedećim potpoglavljima. Prvo je bilo potrebno postaviti *Geoserver* poslužiteljsku aplikaciju, a zatim je u sljedećem koraku u nju izvršen unos prostornih podataka. Nakon toga je bilo potrebno definirati koordinatni sustav koji će se koristiti prilikom posluživanja prethodno unesenih podataka, a na kraju je opisan cijeli postupak pisanja koda *webGIS* aplikacije.

4.1. Postavljanje Geoservera

Proces postavljanja *Geoserver*-a je relativno jednostavan. Zbog činjenice da je pisan u programskom jeziku *Java*, autorima *Geoserver*-a je pisanjem jedinstvenog izvornog koda omogućeno pokretanje njihove aplikacije na svim zastupljenijim računalnim operativnim sustavima današnjice. Kao rezultat ove činjenice, korisnik samo treba preuzeti univerzalni skup izvršnih datoteka, koje se pokreću na većini operativnih sustava (eng. *OS Independent Binary*) sa, primjerice, sljedeće internet adrese:

http://downloads.sourceforge.net/geoserver/geoserver-2.0.2-bin.zip

Nakon preuzimanja, arhivu u .zip obliku je potrebno otpakirati na korisniku poznatu lokaciju – primjerice na *Windows* operativnom sustavu, ta lokacija bi bila "`C:\Program Files\GeoServer`", a na *Unix*-oidnim operativnim sustavima "`/usr/local/geoserver`" (URL 24).

Potom se potrebno uvjeriti da je na računalu instaliran paket potreban za izvršavanje koda pisanog u programskom jeziku *Java*, pod nazivom *Java JRE* (eng. *Java Runtime Enviroment* – okruženje za pokretanje *Java* koda). Nakon preuzimanja i instalacije, isti je potrebno dodatno namjestiti kako bi radio s *Geoserver*-om.

U *Windows* operativnom sustavu, ta se radnja se izvršava posjetom sljedećem mjestu: *Control Panel* > *System* > *Advanced* > *Environment Variables*. Pod

stavkom "*System Variables*" odabire se gumb *New*, te se potom u polje "*Variable Name*" se unosi tekst (bez navodnika) "*JAVA_HOME*", a u polje "*Variable Value*" se unosi putanja direktorija u kojem je instaliran *Java JRE*.

Proces postavljanja *Geoserver*-a na *Unix*-oidnim operativnim sustavima je nešto kraći (slijedi primjer konfiguracije na *Mac OS X*-u). U prozoru s naredbenim retkom (eng. *Terminal*) izvršavaju se sljedeće dvije naredbe (navedene u dva retka i prilikom izvršavanja odvojene tipkom *Enter*):

```
echo "export GEOSERVER_HOME=/usr/local/geoserver" >> ~/.profile
. ~/.profile
```

Nakon ove radnje, još je potrebno trenutnog korisnika proglasiti vlasnikom *Geoserver*-ovog direktorija, kako bi se sama aplikacija mogla pokrenuti:

```
sudo chown -R USER_NAME /usr/local/geoserver/
```

Nakon izvršenja gore navedenog postupka *Geoserver* je spreman za pokretanje. Na *Windows* operativnom sustavu, korisnik pokreće *Geoserver* posjetom direktorija u kojem je postavljen *Geoserver*, te potom odabire poddirektorij "*bin*" ("`C:\Program Files\GeoServer\bin`") i pokreće izvršnu datoteku *startup.bat*. Korisnik zaustavlja izvođenje *Geoserver*-a pokretanjem datoteke *shutdown.bat*, koja se nalazi u istom direktoriju (URL 24).

Procedura pokretanja na *Unix*-oidnim operativnim sustavima je vrlo slična – korisnik pokreće *startup.sh* skriptu iz poddirektorija "*bin*" koji se nalazi u direktoriju u kojem je postavljen *Geoserver* ("`/usr/local/geoserver/bin`"). Kao i u slučaju rada na *Windows* operativnom sustavu, izvršavanje *Geoserver*-a se na *Unix*-oidnim operativnim sutavima prekida pokretanjem *shutdown.sh* skripte.

Kako bi se odradio zadnji korak pokretanja *Geoserver*-a, korisnik u svoj internet pretraživač upisuje sljedeću adresu: http://localhost:8080/geoserver, preko koje pristupa sučelju za mrežno upravljanje *Geoserver*-om (eng. *Web Administration Interface*). Ako je *Geoserver* ispravno postavljen i uspješno se pokrene, korisnika će dočekati sučelje za mrežnu administraciju *Geoserver*-a.

Pri prvom pokretanju *Geoserver*-a, i nakon prijavljivanja koristeći pretpostavljeno korisničko ime ("*administrator*") i lozinku ("*geoserver*"), poželjno je da korisnik promijeni podatke za prijavu, kako bi *Geoserver* zaštitio od neovlaštenog pristupa i neželjene promjene podataka. Ova se radnja izvršava odabiranjem poveznice "*Users*", pod "*Security*" skupinom stavki ponuđenih u izborniku s lijeve strane sučelja za mrežno upravljanje *Geoserver*-om. Nakon toga je potrebno odabrati postojećeg korisnika nazvanog "*admin*", klikom na njegovo korisničko ime označeno plavom bojom. Tada pristupamo sučelju za promjenu informacija o tome korisniku, poput korisničkog imena (pretpostavljenom korisniku "*admin*" nije moguće promijeniti korisničko ime), lozinke (poželjno ju je promijeniti s pretpostavljene vrijednosti "*geoserver*"). Potom je potrebno tipkom "*Save*" spremiti promjene. Potrebna radnja za promjenu korisničkog imena i lozinke je prikazana na slici (Slika 3).

Slika 3. Postupak promjene korisničkog imena i lozinke

U istom dijelu sučelja za mrežno upravljanje *Geoserver*-om omogućena su dodavanja novih korisnika i brisanja postojećih, promjena korisničkih prava i propisivanje novih, i slično. Ako postoji samo jedan korisnik, u ovom slučaju onaj s korisničkim imenom "*admin*", njega nije moguće ukloniti (URL 24).

4.2. Unos podataka u Geoserver

Izvorni podaci za izradu *webGIS*-a Generalnog urbanističkog plana Rovinjskog sela su dobiveni u *.dwg* obliku. Isti podaci su obrađeni u aplikaciji *AutoCAD Map 2010*, a obrada je uključivala izoliranje željenih slojeva koji će se koristiti u izradi *webGIS*-a, te na kraju, čišćenje topologije, kako bi se željena geometrija mogla izvoziti (eng. *export*) u *ESRI Shape* oblik podataka.

Nakon izvoza, za svaki sloj su dobivene po tri datoteke i to: *.shp* (koja sadrži geometriju), *.shx* (sadrži položajne pokazivače) i *.dbf* datoteka (sadrži atributne podatke). Sve skupa, iz *AutoCAD Map*-a je u *ESRI Shape* datoteke izvedeno 10 slojeva (9 slojeva zona generalnog urbanističkog plana i jedan sloj prometnica), tj. dobiveno je ukupno 30 datoteka. Korištenjem aplikacije *Quantum GIS* dobivenih 10 slojeva je spojeno u jedan, te su dodatno uređeni atributni podaci.

Sljedeći korak je obuhvaćao kopiranje dobivenih datoteka u poddirektorij nazvan "rovinjsko-selo", koji je postavljen na lokaciji "`geoserver/data_dir/data`" . Postavljanje podataka na ovu lokaciju je nužno za njihov unos u *Geoserver*.

Potom je prijavljivanjem na sučelje za mrežno upravljanje *Geoserver*-om stvoreno novo tzv. radno mjesto (eng. *workspace*), koje će poslužiti kao svojevrsno skladište za spremanje prostornih podataka. Ova radnja je obavljena odabirom poveznice "*Add new workspace*", koja se nalazi na "*Workspaces*" stranici. Pritom je potrebno upisati naziv tog radnog mjesta (u ovom slučaju radno mjesto je nazvano "rovinjsko-selo"), te *Namespace URI*[5], kojem je postavljen isti naziv. Potom je, u ovom slučaju, ovo radno mjesto odabrano kao pretpostavljeno (eng. *default*), te su postavke spremljene klikom na tipku "*Submit*" (postupak prikazan na slici - Slika 4).

---

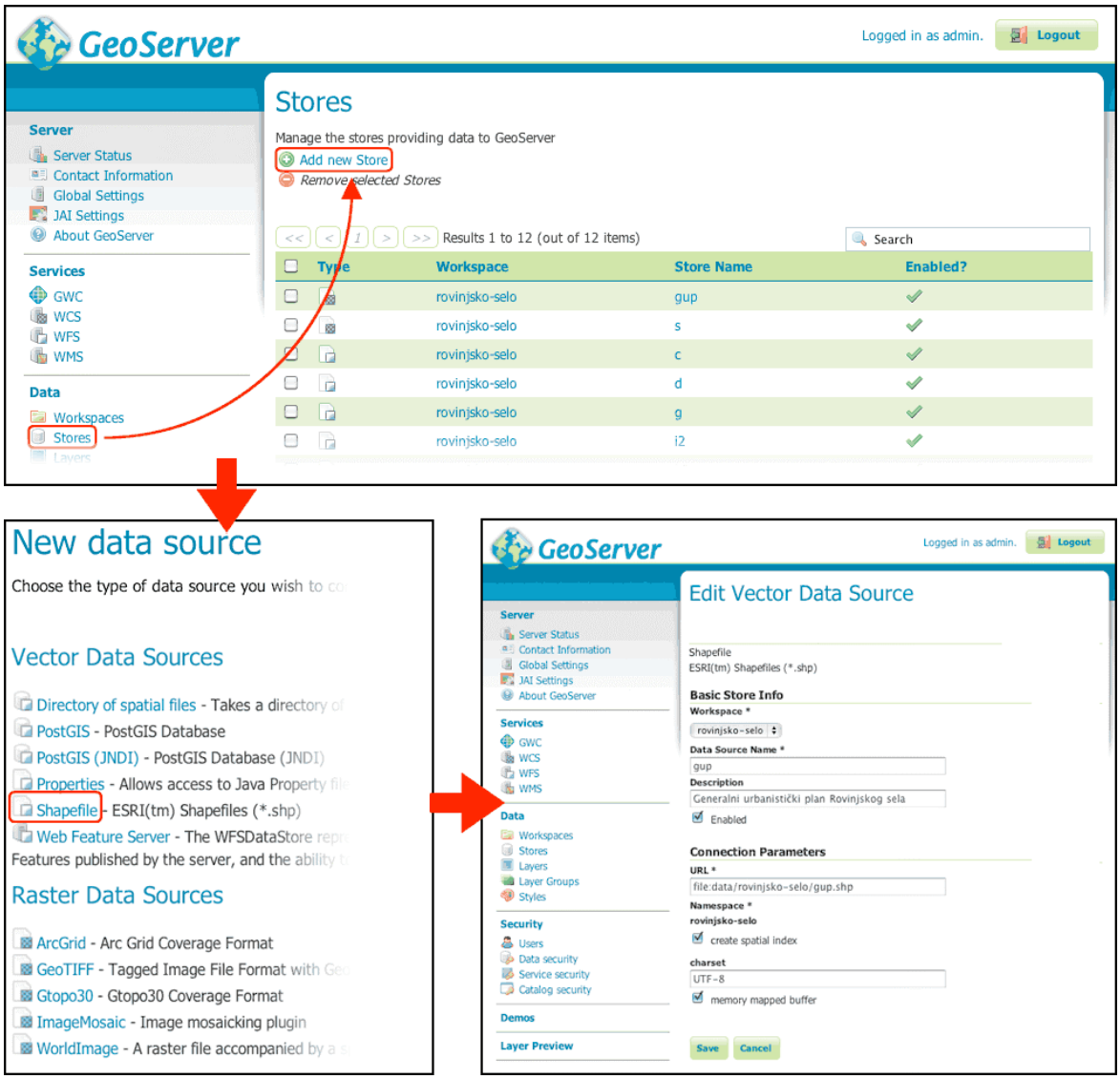
[5] ***Namespace URI*** (*Uniform Resource Identifier*) - niz znakova za identifikaciju skupa podataka na internetu).

Slika 4. Postupak stvaranja novog radnog mjesta (eng. Workspace)

Nakon stvaranja radnog mjesta slijedi postupak uvoza podataka u *Geoserver*. Podaci su uvoze odabirom poveznice "*Stores*" (hrv. skupovi obilježja), koja se nalazi u "*Data*" podskupini izbornika. Odabirom poveznice "*Add new Store*" dolazimo do izbora vrste podataka koje želimo uvesti u *Geoserver*. Sa popisa mogućih oblika podataka odabire se stavka "*Shapefile*", koja korisnika dovodi na stranicu s postavkama za unošenje podataka.

Na stranici s postavkama za unošenje podataka potrebno je odabrati radno mjesto (u ovom slučaju odabrano je "rovinjsko-selo"), unijeti ime izvora podataka (u ovom slučaju uneseno je ime "gup", koje označava prostorne podatke generalnog urbanističkog plana Rovinjskog sela) i eventualno njegov opis. Također je potrebno omogućiti upotrebu ovog skupa obilježja odabiranjem stavke "*Enabled*", ako ista stavka već nije označena. Potom je u polje *URL*[6] potrebno unijeti relativnu putanju do *Shape* datoteke (relativnu gledanu od lokacije "`geoserver/data_dir/`").

[6] ***URL*** (*Uniform Resource Locator*) – za razliku od *URI*-ja prije adrese navodi i mehanizam za dohvaćanje resursa, u ovom slučaju "*file:*".



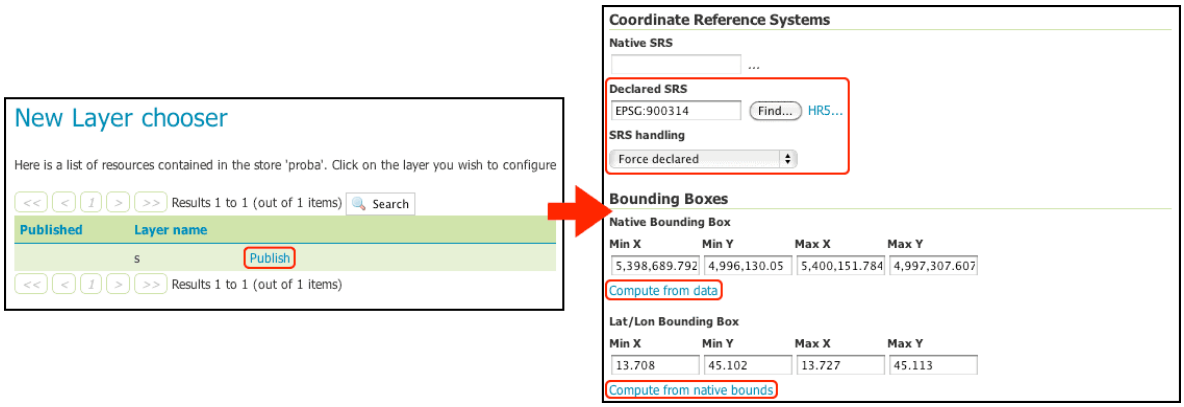
Slika 5. Postupak dodavanja prostornih podataka u Geoserver

Ostale postavke su ostavljene na svojim pretpostavljenim vrijednostima, osim polja "*charset*" koji definira podršku za prikaz dijakritičkih znakova atributnih podataka. Ovo polje se postavlja na vrijednost "*UTF-8*". Potrebni koraci za izvršavanje prethodno opisanog postupka su prikazani na slici (Slika 5).

Nakon klika na gumb "*Save*", dolazimo do koraka za objavljivanje upravo dodanog skupa obilježja. Klikom na poveznicu "*Publish*", učitava se stranica za definiranje postavki sloja, na kojoj se, između ostalog, navodi ime sloja, prema potrebi kratki opis njegove namjene, ključne riječi koje ga opisuju i slično.

Potom je potrebno definirati korišteni koordinatni sustav. Ponekad je potrebno definirati vlastiti koordinatni sustav (u ovom slučaju onaj označen kodom

EPSG:900314), a postupak definiranja istog je objašnjen u sljedećem poglavlju. Tada se izračunavaju granice opsega uvezenog skupa obilježja. Postavljanje ovih stavki se izvodi pod "*Bounding Boxes*" skupinom parametara, gdje *Geoserver* automatski izračunava vršne koordinate graničnog okvira (eng. *Bounding box*), nakon klika na poveznicu "*Compute from data*" pod stavkom "*Native Bounding Box*". Kada se koristi vlastita projekcija, potrebno je također odabrati poveznicu "*Compute from native bounds*" pod stavkom "*Lon/Lat Bounding Box*", kako bi se koordinate vršnih točaka graničnog okvira preračunale u stupnjeve korištenjem zadanog koordinatnog sustava. Potom je potrebno kliknuti gumb "*Save*", kako bi se odabrane postavke spremile (dio postupka je prikazan na slici - Slika 6).



Slika 6. Dio postupka definiranja parametara sloja

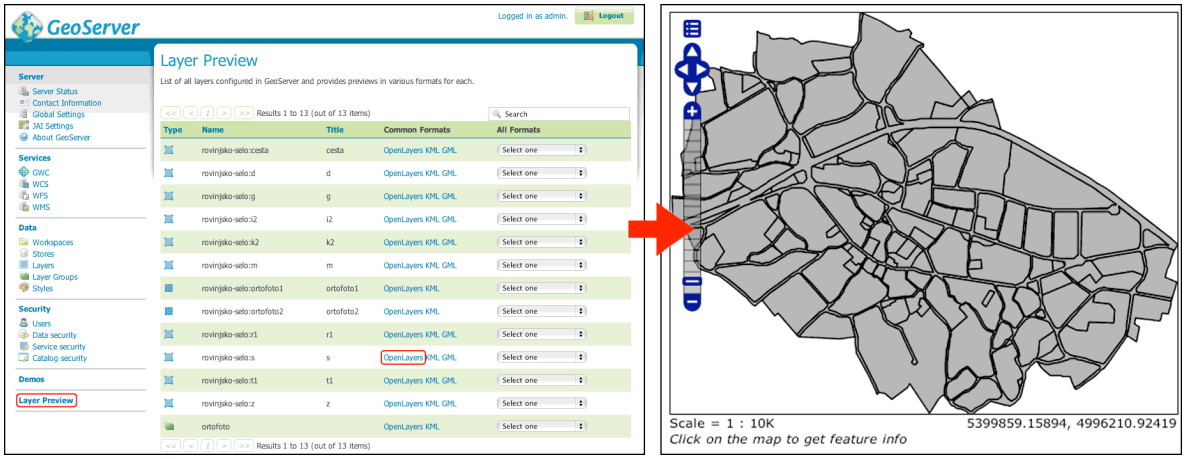
Ako se podaci na *Geoserver*-u poslužuju preko *WMS* protokola, potrebno je prije spremanja postavki još definirati stil koji će se koristiti za prikaz geometrije. Ova radnja se obavlja odabirom kartice "*Publishing*", te se tada pod stavkom "*Default Style"* odabire željeni stil (koji je prethodno definiran u *SLD*[7] obliku). Zato što se u ovom radu koristi dohvaćanje prostornih podataka u vektorskom obliku putem *WFS* protokola, ovaj postupak nije potreban (URL 24).

Za kraj bi bilo dobro uvjeriti se da je *Geoserver* prepoznao uvezene podatke. Ova se provjera može izvršiti odabirom poveznice "*Layer Preview*" u izborniku s desne strane sučelja, te se potom odabire vrsta vizualizacije podataka (na slici je

---

[7] ***SLD*** (*Styled Layer Descriptor*) – opisni jezik za definiranje parametara vizualizacije, temeljen na *XML*-u.

prikazana vizualizacija sloja GUP-a Rovinjskog sela). Sama vizualizacija je izvedena korištenjem *OpenLayers* skupa gotovih alata (Slika 7).



Slika 7. Provjera uspješnosti unošenja podataka u Geoserver

4.3. Koordinatni sustavi i projekcije

Prilikom izrade aplikacije, korišteni su prostorni podaci koji opisuju generalni urbanistički plan mjesta Rovinjsko selo, koje se nalazi u Istri, tj. u 5. zoni Hrvatskog državnog koordinatnog sustava (HDKS). Prilikom izvoza (eng. *exporting*) prostornih podataka iz *AutoCAD Map 2010* aplikacije u *ESRI Shape* oblik, isti nisu uključivali *.prj* datoteku, koja navodi koordinatni sustav u kojoj su prostorni podaci definirani. Zbog ove je činjenice prilikom unošenja podataka u *Goeserver* nužno definirati vlastite parametre koordinatnog sustava.

Upravo spomenuti parametri nisu uključeni u *Geoserver* prilikom instalacije, te ih je potrebno naknadno definirati. Niže navedeni parametri (razdijeljeni u više redaka radi čitljivosti) se dodaju na kraj datoteke "*epsg.properties*", koja se nalazi u "`data_dir/user_projections`" poddirektoriju *Geoserver* instalacije (URL 24):

```
PROJCS["HR5",
  GEOGCS["HR5",
    DATUM["HR1901",
      SPHEROID["Bessel 1841", 6377397.155, 299.1528128],
      TOWGS84[514.0188, 155.448, 507.0461, 5.6136, 3.676, -11.4667, 2.091]],
    PRIMEM["Greenwich", 0.0],
    UNIT["degree", 0.017453292519943295],
    AXIS["Geodetic longitude", EAST],
```

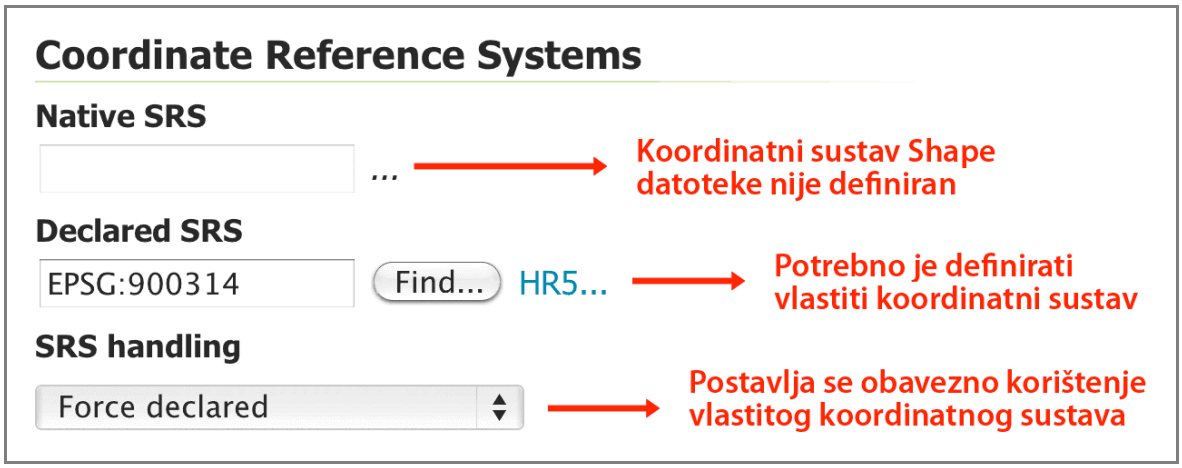
```
    AXIS["Geodetic latitude", NORTH]],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["central_meridian", 15.0],
  PARAMETER["latitude_of_origin", 0.0],
  PARAMETER["scale_factor", 0.9999],
  PARAMETER["false_easting", 5500000.0],
  PARAMETER["false_northing", 0.0],
  UNIT["m", 1.0],
  AXIS["Easting", EAST],
  AXIS["Northing", NORTH],
  AUTHORITY["EPSG","900314"]]
```

Nakon dodavanja ovih parametara na kraj gore navedene datoteke, potrebno je ponovno pokrenuti *Geoserver*, kako bi isti prepoznao promijene. Ovaj postupak se provodi pokretanjem izvršnih datoteka (nužno navedenim redoslijedom) "*Shutdown.bat*" i "*Startup.bat*" pod *Windows* operativnim sustavom, odnosno "*shutdown.sh*" i "*startup.sh*" skripti u *Unix*-oidnim operativnim sustavima. Ove datoteke se nalaze u "*bin*" poddirektoriju instalacije *Geoserver*-a.

Za kraj je u prethodno opisanom postupku definiranja postavki sloja potrebno definirati korištenje vlastite *SRS*[8] definicije (URL 26), koja se odabire u "*Declared SRS*" polju (pri dnu stranice s postavkama sloja). Ovaj korak omogućuje definiranje koordinatnog sustava korištenih prostornih podataka, iako isti nije bio definiran u početku.

U ovom je slučaju korišten koordinatni sustav s oznakom *EPSG:900314*, koji sadrži parametre potrebne za definiranje 5. zone HDKS-a. Odabir željene SRS definicije prilikom definiranja parametara sloja u *Geoserver*-u, prikazan je na slici (Slika 8).

[8] ***SRS*** (*Spatial Referencing System*) – lokalni, regionalni ili globalni koordinatni sustav korišten za pozicioniranje geografskih obilježja.



Slika 8. Definiranje korištenja vlastitog koordinatnog sustava

Nakon određivanja ovih postavki, *Geoserver* koristi prethodno definirani koordinatni sustav za posluživanje unesenih prostornih podataka. Prilikom zaprimanja prostornog upita, u kojem je kao jedan od parametara naveden željeni koordinatni sustav za prikaz podataka, *Geoserver* će izvršiti transformaciju iz postavljenog koordinatnog sustava u željeni. Nakon toga će se korisniku poslati odgovor, koji sadrži prostorne podatke u zatraženom koordinatnom sustavu.

webGIS aplikacija, čija je izrada objašnjena u nekoliko sljedećih potpoglavlja, za prikaz prostornih podataka koristi *Google Mercator* projekciju (u kojoj od *Geoserver*-a zatražuje dostavljanje prostornih podataka). Ova je projekcija bazirana na Merkatorovoj projekciji, a koristi se za prikazivanje satelitskih snimaka i kartografskih prikaza na *Google Maps* kartografskom servisu. Projekcija nosi *EPSG* kod 900913, a u *webGIS* aplikaciji se koristi kako bi se u njoj prikazani prostorni podaci mogli, prema potrebi, preklopiti s pozadinama preuzetima s prije spomenutog servisa.

4.4. Izrada web aplikacije

Sam proces kodiranja *webGIS* aplikacije je podijeljen u nekoliko koraka. U prvom se koraku izrađuje kostur internet stranice korištenjem *HTML* opisnog jezika. Sljedeći korak je izrada sučelja putem kojeg će korisnik upravljati aplikacijom. Nakon toga slijedi korak implementacije prikaza obilježja (geometrije i atributa), koji će se moći izmjenjivati nakon što se implementira funkcionalnost aplikacije u zadnjem koraku. Ovaj postupak je opisan u sljedećim potpoglavljima.

4.4.1. Pisanje HTML koda

Izrada *webGIS* aplikacije započinje pisanjem osnovnog dijela koda zaduženog za prikazivanje internet stranice. Ovaj dio aplikacije se izrađuje pisanjem koda u *HTML* opisnom jeziku.

Prvo je potrebno definirati osnovne elemente stranice koje nalaže *HTML* jezik, a to su zaglavlje (eng. *head*) i tijelo (eng. *body*). Kod potreban za stvaranje prije navedenih elemenata prikazan je u sljedećem isječku:

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

Oznake (eng. *tags*) označavaju o kojem se dijelu *HTML* koda radi. Tako, primjerice, oznaka `<head>` govori da se radi o početku zaglavlja, a oznaka `</head>` označava kraj zaglavlja. Krajnja se oznaka od početne razlikuje po kosoj crti ispred naziva elementa.

Drugi korak obuhvaća dodavanje elemenata nužnih za ispravno funkcioniranje stranice u njeno zaglavlje (omeđeno prethodno opisanim oznakama). Za početak će se dodati naslov stranice i podrška za dijakritičke znakove (dodaje se samo kod ispisan masnim slovima):

```
<head>
   <title>GUP Rovinjskog sela</title>
   <meta http-equiv="content-type" content="text/html; charset=utf-8" />
</head>
```

Kao što je već objašnjeno u prethodnom dijelu ovog poglavlja, elementi stranice se definiraju unutar parova oznaka. Tako se, primjerice, naslov stranice definira unutar `<title>` i `</title>` oznaka. Izuzetak su elementi koji ne obuhvaćaju nikakav tekstualni sadržaj, nego samo navode pravila (primjer podrške za dijakritičke znakove). Oni se navode u jednoj oznaci, koja prije zatvarajućeg znaka zagrade ">" sadrži znak kose crte "/", koji govori da je riječ o kraju elementa.

Zbog činjenice da je aplikacija izrađena korištenjem *JavaScript* programskog jezika, sav njezin kod biti će sadržan u zaglavlju *HTML* koda. Iako će kod tijela stranice ostati prazan, *GeoExt* skup gotovih alata će se pobrinuti za prikazivanje elemenata aplikacije unutar tijela stranice.

No, prije nego li se počne pisati *JavaScript* kod, potrebno je uključiti vanjske datoteke s kodom gotovih skupova programskih alata, koji će se koristiti prilikom izrade aplikacije. To se može učiniti dodavanjem sljedećih linija koda unutar zaglavlja stranice, odmah nakon definiranja podrške za dijakritičke znakove i prije `</head>` završne oznake zaglavlja:

```
<!--  Uključivanje vanjske datoteke OpenLayers gotovog skupa alata -->
<script src="OpenLayers/OpenLayers.js" type="text/javascript"></script>

<!--  Uključivanje vanjske datoteke ExtJS gotovog skupa alata -->
<script src="ExtJS/adapter/ext/ext-base.js" type="text/javascript"></script>
<script src="ExtJS/ext-all.js"  type="text/javascript"></script>

<!--  Uključivanje vanjske datoteke GeoExt gotovog skupa alata -->
<script src="GeoExt/script/GeoExt.js" type="text/javascript"></script>
```

Putanje do vanjskih datoteka (navedene unutar `src=""` elementa oznake) moraju odgovarati stvarnoj lokaciji datoteka na disku. U aplikaciju su uz *OpenLayers* uključena i dva skupa gotovih programskih alata (*ExtJS* i *GeoExt*), zato što funkcioniranje *GeoExt*-a ovisi o već postojećim funkcijama implementiranim u *ExtJS*.

Nakon koraka dodavanja skupova alata, potrebno je još uključiti i vanjske datoteke sa stilovima zaduženim za vizualizaciju elemenata pisanih korištenjem gore navedenih skupova gotovih alata:

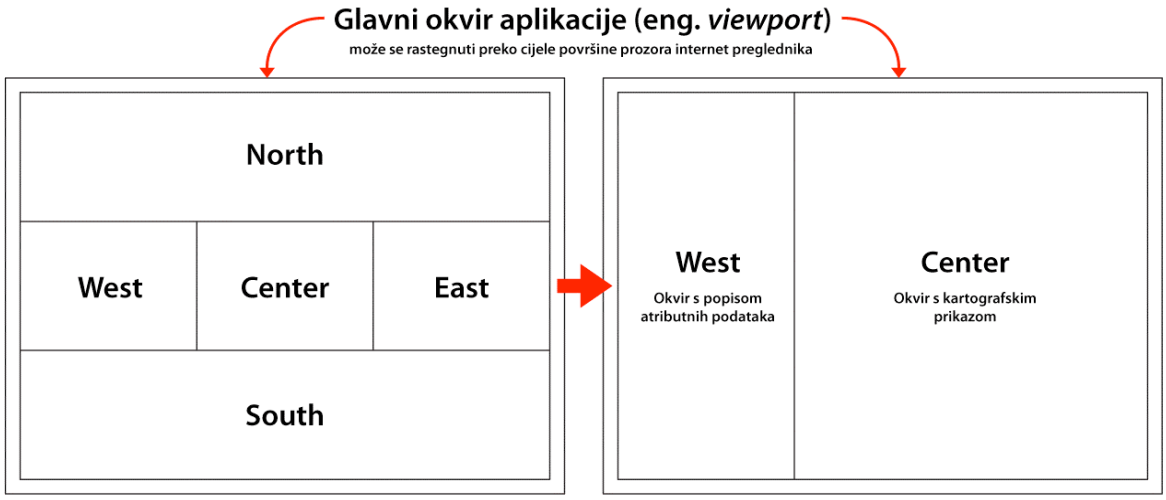
```
<!--  Uključivanje vanjske datoteke sa stilom za ExtJS elemente-->
<link rel="stylesheet" type="text/css" href="ExtJS/resources/css/ext-all.css" />

<!--  Uključivanje vanjske datoteke sa stilom za GeoExt elemente-->
<link rel="stylesheet" type="text/css" href="GeoExt/resources/css/geoext-all.css"
/>
```

Ovim se korakom definiraju svi potrebni parametri potrebni za početak izrade aplikacije i zaključuje postupak pisanja *HTML* opisnog koda. Sljedeći korak je izrada sučelja korištenjem *ExtJS* i *GeoExt* skupova gotovih programskih alata.

4.4.2. Izrada sučelja aplikacije

Kako bi se osigurao jednaki izgled i ponašanje internet aplikacije u različitim internet preglednicima, *ExtJS* definira različite elemente sučelja i način njihovog razmještaja u prozoru preglednika. Isto tako, korištenjem *ExtJS*-a, različiti se okviri mogu sakrivati (eng. *minimizing*) iz prikaza aplikacije, ili biti ponovno prikazani odabirom gumba. Na ovaj se način omogućuje jednostavan način upotrebe aplikacije, kojom korisnik može brzo ovladati.

Izrada sučelja započinje definiranjem glavnog okvira aplikacije (eng. *viewport*), koji će se rastezati preko cijelog prozora internet preglednika. Glavni će okvir sadržavati ostale okvire (eng. *panels*), koji se, primjerice, koriste za smještanje tabličnog prikaza atributa ili kartografskog prikaza geometrije nekog obilježja (eng. *feature*). Okviri će u aplikaciji biti smješteni u jednoj od pet strana prozora (eng. *regions*) internet preglednika, koju definira korisnik (eng. *center*, *north*, *south*, *east* i *west*). Svih pet strana ne mora biti definirano, ali se uvijek mora definirati središnji okvir, jer inače dolazi do nepravilnosti prilikom prikazivanja sučelja (URL 27). Način definiranja razmještaja okvira prikazan je na slici (Slika 9).



Slika 9. Primjer razmještaja okvira aplikacije pisane ExtJS-om

Željeni raspored okvira u može se postići uključenjem sljedećeg isječka koda u zaglavlje *HTML* datoteke, odmah ispod definicije stilova.

```
<script type="text/javascript">
Ext.onReady(function() {

	var mapPanel = new Ext.Panel({
		title: "Kartografski prikaz",
		region: "center",
		html: "<p>Okvir kartografskog prikaza</p>",
	});

	var gridPanel = new Ext.Panel({
		title: "Popis atributnih podataka",
		region: "west",
		width: 200,
		collapsible: true,
		html: "<p>Okvir popisa atributnih podataka</p>",
	});

	var viewport = new Ext.Viewport({
		layout: "fit",
		items: {
			layout: "border",
			items: [mapPanel, gridPanel]
		},
	});
})
</script>
```

U ovom isječku stvorena su tri okvira:

- ***mapPanel*** (okvir kartografskog prikaza) – Sadrži naslov, definiciju strane prozora preglednika u kojoj će se iscrtavati (*center*, središnji dio) i opisni tekst (u *HTML* obliku), koji ga popunjava dok se u njega ne postavi kartografski prikaz.
- ***gridPanel*** (okvir popisa atributnih podataka) – Također sadrži naslov i mjesto iscrtavanja u prozoru (west, lijevi dio), širinu (definiranu u pikselima), mogućnost sakrivanja (parametar *collapsible*) i opisni tekst koji ga popunjava dok se u njega ne postavi popis atributnih podataka.

- ***viewport*** (glavni okvir aplikacije) – Definirana je veličina glavnog okvira koji se rasteže preko cijelog prozora preglednika (`layout: "fit"`), način odjeljivanja okvira (`layout: "border"` – odjeljivanje graničnikom), te je naveden popis okvira koji ga popunjavaju (prethodno definirani *mapPanel* i *gridPanel*).

Nakon definiranja okvira aplikacije potrebno je još napraviti i alatnu traku s gumbima za manipulaciju s prostornim podacima. Alatna traka će se dodati na vrh okvira kartografskog prikaza i sadržavati će gumbe za dodavanje obilježja, njihovo brisanje i spremanje izmjena, te za uvećavanje obuhvata (eng. *zooming to extents*). Alatna traka postavlja se u aplikaciju dodavanjem koda za stvaranje alatne trake s *ExtJS* gumbima unutar *mapPanel* objekta, odmah ispod retka s opisnim tekstom. Cijeli *mapPanel* objekt tada izgleda kao u sljedećem isječku:

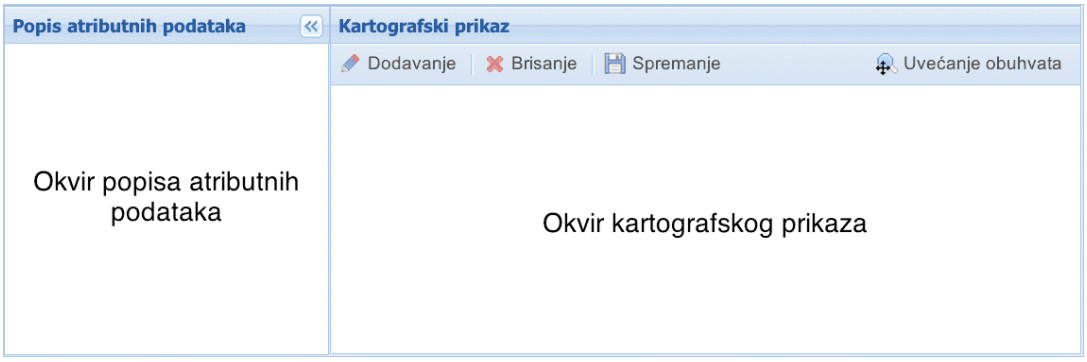
```
var mapPanel = new Ext.Panel({
       title: "Kartografski prikaz",
       region: "center",
       html: "<p>Okvir kartografskog prikaza</p>",
       tbar: [
              new GeoExt.Action({
                     text: "Dodavanje",
                     iconCls: "dodavanje",
                     enableToggle: true
              }),
              "-",
              new GeoExt.Action({
                     text: "Brisanje",
                     iconCls: "brisanje",
              }),
              "-",
              new GeoExt.Action({
                     text: "Spremanje",
                     iconCls: "spremanje",
              }),
              "->",
              new GeoExt.Action({
                     text: "Uvećanje obuhvata",
                     iconCls: "obuhvat",
              }),]});
```

Na već postojeći kod za kreiranje *mapPanel*-a dodan je element s alatnom trakom (eng. *toolbar*). Označen je parametrom tbar (eng. *top bar* – gornja traka) i sadržava popis prije spomenutih gumbiju. Gumbi, koji se stvaraju naredbom "`new GeoExt.Action`", trenutno nemaju dodijeljenu funkciju. Zasad je svakom definiran samo natpis (parametar *text*) i ikona (parametar *iconCls*), a gumbu za dodavanje geometrije je parametar *enableToggle* postavljen na vrijednost true. Prethodno navedeni parametar se dodaje kako bi ovaj gumb ostao aktivan nakon odabira (kako bi omogućio crtanje geometrije), a isključuje se tek ponovnim odabirom. Parametar "-" dodaje crtu koja vizualno odjeljuje gumbe, a parametar "->" pomiče gumbe na desni dio alatne trake (URL 28).

Također, kako bi se prikazale ikone na gumbima, potrebno je posebnim stilom definirati ponašanje *iconCls* parametara, tj. svakom parametru su treba pridijeliti slika koja će se pokazivati kao ikona. Ovo se postiže dodavanjem sljedećeg isječka koda odmah iznad retka u kojem počinje *JavaScript* kod aplikacije (redak koji sadrži tekst `<script type="text/javascript">`):

```
<!-- Definiranje stila za prikaz GoeExt kontrola -->
<style>
  .obuhvat    {background-image: url(Elementi/obuhvat.png)   !important;}
  .spremanje  {background-image: url(Elementi/spremanje.png) !important;}
  .brisanje   {background-image: url(Elementi/brisanje.png)  !important;}
  .dodavanje  {background-image: url(Elementi/dodavanje.png) !important;}
</style>
```

Spremanjem koda opisanog u ovom i prethodnom potpoglavlju u datoteku s *.html* nastavkom, i njezinim pokretanjem u internet pregledniku, može se vidjeti upravo stvoreno sučelje aplikacije, kao što je prikazano na slici (Slika 10).



Slika 10. Izgled sučelja aplikacije

4.4.2.1 Implementacija prikaza obilježja

Nakon definiranja okvira, potrebno je iste popuniti odgovarajućim prikazima obilježja. Prije samog popunjavanja okvira kartografskog prikaza, potrebno je definirati *OpenLayers* objekt karte (sa svim pripadajućim parametrima), te sloj (ili više njih) koji će ta karta sadržavati. Kartografski prikaz se dodaje sljedećim isječkom koda, koji se postavlja prije retka s definicijom *mapPanel* okvira:

```
var map = new OpenLayers.Map("map", {
	projection: new OpenLayers.Projection("EPSG:900913"),
	displayProjection: new OpenLayers.Projection("EPSG:4326"),
	units: "m",
	numZoomLevels: 19,
	maxResolution: 156543.0339,
	maxExtent: new OpenLayers.Bounds(
		1525769.07602, 5637546.13799,
		1528114.73732, 5639399.73592
	),
	controls: [
		new OpenLayers.Control.Navigation(),
		new OpenLayers.Control.PanZoomBar(),
		new OpenLayers.Control.MousePosition(),
		new OpenLayers.Control.KeyboardDefaults(),
		new OpenLayers.Control.Scale(),
		new OpenLayers.Control.ScaleLine()
	]
});
```

Kartografski prikaz se stvara korištenjem *OpenLayers.Map* klase (URL 29), te se novostvorenom objektu definira naziv (u ovom slučaju prvi parametar "*map*") i niz ostalih parametara. U ovom slučaju radi se o sljedećim parametrima:

- ***projection*** – Pomoću *EPSG* koda se definira projekcija koja će se koristiti prilikom prikazivanja ove kartografskog prikaza (*Google Mercator* projekcija, *EPSG:900913*).
- ***displayProjection*** – Pomoću *EPSG* koda se definira projekcija koja će se koristiti za prikaz tekstualnih podataka o kartografskom prikazu. Primjerice, umjesto pokazivanja koordinata pokazivača miša u metrima (kako propisuje *Google Mercator* projekcija), te iste koordinate će se prikazivati u stupnjevima (prema *WGS84* datumu – *EPSG:4326*).

- ***units*** – Jedinice korištene za, primjerice, prikaz linije mjerila (eng. *scale line*).
- ***numZoomLevels*** – Broj razina promjene mjerila.
- ***maxExtent*** – Definiranje koordinata vršnih točaka okvira obuhvata (eng. *extent bounds*), koji će ograničavati područje koje će se prikazati prilikom pokretanja aplikacije. Dijagonalno suprotne vršne točke okvira su definirane parom X,Y koordinata, koje su navedene kao udaljenost (u metrima) od početnog meridijana za X koordinatu i udaljenost od ekvatora za Y koordinatu.
- ***controls*** – Popis kontrola koje su dodijeljene kartografskom prikazu. U ovom slučaju koristi se *Navigation()* (omogućuje kretanje po kartografskom prikazu korištenjem pokazivača miša – eng. *panning*), *PanZoomBar()* (prikazuje skalu promjene mjerila i virtualne strelice za kretanje po kartografskom prikazu), *MousePosition()* (prikazuje trenutne koordinate pokazivača miša), *KeyboadDefaults()* (omogućuje korištenje tipkovničkih strelica i tipki "+" i "-" za kretanje po kartografskom prikazu i promjenu mjerila), *Scale()* (prikaz mjerila) i *ScaleLine()* (prikaz linije mjerila).

Kao sljedeći korak, potrebno je definirati slojeve koji će se koristiti u kartografskom prikazu. U ostatku ovog poglavlja će se opisati dodavanje vektorskog sloja, koji će preuzimati prostorne podatke o generalnom urbanističkom planu Rovinjskog sela s *Geoserver* poslužitelja (koji iste skladišti u *ESRI Shape* podatkovnom obliku).

Zahtjev za podacima (definiran parametrima sloja) se u ovom slučaju poslužitelju šalje u obliku *WFS* upita, a poslužitelj vraća podatke u *GML* zapisu. Dobiveni podaci se oblikuju prema definiranim stilovima (primjerice ispuna i obrub poligona), te potom prosljeđuju kartografskom prikazu koji ih iscrtava na ekran.

Prije definiranja sloja, potrebno je definirati stil, odnosno pravila vizualizacije za geometriju tog sloja. Pravila za vizualizaciju zona generalnog urbanističkog plana Rovinjskog sela se definiraju sljedećim (nepotpunim) isječkom koda, koji se, primjerice, postavlja iznad retka kreiranja karte (redak s "`var map = new OpenLayers.Map`" naredbom):

```
var styleMap = new OpenLayers.StyleMap();
var styleRules = [
       new OpenLayers.Rule({
```

```
            filter: new OpenLayers.Filter.Comparison({
                  type: OpenLayers.Filter.Comparison.LIKE,
                  property: "tip",
                  value: "M"
            }),
            symbolizer: {
                  strokeColor: "6fa082",
                  fillColor: "6fa082"
            },
            elseFilter: true
      })
];
styleMap.styles["default"].addRules(styleRules);
```

Ovaj isječak koda je nepotpun zato što sadržava samo jedno pravilo (ono za vizualizaciju GUP zona M1 i M2). Pravila za vizualizaciju ostalih GUP zona se mogu naći u izvornom kodu aplikacije koji je priložen uz ovaj rad (Prilog 1).

Prethodno prikazanim isječkom prvo se definira skup stilova (*OpenLayers.StyleMap()*), kojem se svi parametri, zato što nisu navedeni, postavljaju na pretpostavljene vrijednosti (URL 29). Pravila koja se navode u nizu iza skupa stilova prilagođavati će pretpostavljene parametre stila ovisno o obilježju na koje se primjenjuju.

Unutar niza pravila se definira novo pravilo (*OpenLayers.Rule()*), koje sadrži parametre filtriranja, tj. u ovom se slučaju traže poligoni kojima je kao atribut "tip" (misli se na tip GUP zone) dodijeljena vrijednost "M". Zato što se u gornjem isječku koda koristi operator za uspoređivanje (eng. *comparator*) *OpenLayers.Filter.Comparison.LIKE*, ovo pravilo se pridjeljuje svim zonama čiji tip počinje s "M" (u ovom slučaju radi se o zonama M1 i M2). Nakon što filter izolira željenu geometriju, na istu se primjenjuje vizualizacija propisana parametrom *symbolizer*, kojim se u ovom slučaju mijenja boja obruba i ispune.

Nakon navođenja svih pravila za vizualizaciju, potrebno je ta ista pravila pridijeliti prethodno definiranom skupu stilova. Na ovaj se način, u ovom slučaju, mijenja samo prije definirana boja ispune i obruba, dok svi ostali parametri skupa stilova ostaju na pretpostavljenim vrijednostima.

Nakon što se navedu svi parametri vizualizacije prostornih podataka, potrebno je dodati sloj koji će ih sadržavati i primjenjivati tu vizualizaciju. Ova se radnja postiže sljedećim isječkom koda, koji se postavlja ispod definiranja kartografskog prikaza:

```
var vectorLayer = new OpenLayers.Layer.Vector("GUP", {
       strategies: [
              new OpenLayers.Strategy.Fixed()
       ],
       protocol: new OpenLayers.Protocol.WFS({
              version: "1.1.0",
              srsName: "EPSG:900913",
              url: "http://localhost:8080/geoserver/wfs",
              featureNS: "rovinjsko-selo",
              featureType: "gup"
       }),
       isBaseLayer: true,
       projection: new OpenLayers.Projection("EPSG:900913"),
       styleMap: styleMap
});
map.addLayers([vectorLayer]);
```

Prvom linijom isječka se stvara novi vektorski sloj, koji nosi naziv "GUP", te definira sljedeće parametre:

- ***strategies*** – Sadrži definiranje načina na koji sloj dohvaća podatke. Parametar *OpenLayers.Strategy.Fixed()* navodi da se podaci s poslužitelja dohvaćaju samo jednom iako se, primjerice, vrši promjena mjerila.
- ***protocol*** - Definira parametre upita za dohvaćanje prostornih podataka s poslužitelja. U ovom slučaju se koristi *WFS* protokol, verzije 1.1.0, a podaci se dohvaćaju u *Google Mercator* projekciji (*EPSG:900913*). Također se definira adresa poslužitelja, imenski prostor korišten za skladištenje obilježja ("rovinjsko-selo") i obilježja koja se dohvaćaju ("gup").
- ***isBaseLayer*** – Parametar je potrebno postaviti na vrijednost true, ukoliko je ovo jedini sloj koji se dodaje na kartografski prikaz.
- ***projection*** – Kao i kod definiranja kartografskog prikaza, pomoću *EPSG* koda se definira projekcija u kojoj će se sloj prikazivati.
- ***styleMap*** – sloju se dodjeljuje skup stilova za njegovu vizualizaciju. Koristi se skup stilova definiran u prethodnom koraku.

Nakon definiranja sloja, potrebno je isti pridijeliti kartografskom prikazu, što se postiže korištenjem naredbe u zadnjem retku isječka. Također, kao zadnji korak implementacije kartografskog prikaza, potrebno je klasu objekta *mapPanel* (koji predstavlja okvir kartografskog prikaza) iz *Ext.Panel* promijeniti u *GeoExt.MapPanel*, kako bi se omogućilo prikazivanje geometrije (URL 29).

Za kraj je još potrebno iz *mapPanel* objekta izbaciti redak s parametrom *html*, i umjesto njega postaviti sljedeći redak koda (bez navodnika): "`map: map,`", kako bi se okvir povezao s kartografskim prikazom i pritom izbrisao opisni tekst. Prvih četiri retka *mapPanel* objekta (do parametra *tbar*) bi tada trebala izgledati ovako:

```
var mapPanel = new GeoExt.MapPanel({
       title: "Kartografski prikaz",
       region: "center",
       map: map,
   ...
```

U zadnjoj fazi implementacije prikaza slijedi dodavanje tabličnog prikaza atributa obilježja, točnije tipova zona i njihove namjene, u lijevi okvir aplikacije. Ova funkcionalnost se postiže u nekoliko koraka.

Prvo je potrebno definirati skup obilježja (eng. *feature store*), koji atribute obilježja dohvaća iz prethodno definiranog vektorskog sloja. Skup se definira korištenjem koda iz sljedećeg isječka, koji se postavlja iznad postojećeg *gridPanel* objekta:

```
var featureStore = new GeoExt.data.FeatureStore({
       fields: [
              {name: "tip", type: "string"},
              {name: "namjena", type: "string"}
       ],
       layer: vectorLayer,
});
```

Definirani skup obilježja (predstavljen objektom *featureStore*) sadrži parametre *fields* i *layer*. Parametar *fields* definira popis naziva atributa dohvaćenih obilježja i njihov tip, u ovom slučaju *string* (eng. *string* – niz znakova, tekst). Drugi parametar, *layer*, definira sloj iz kojeg se atributi dohvaćaju. U ovom slučaju radi se o vektorskom sloju *vectorLayer*, koji je prethodno je definiran u ovom poglavlju.

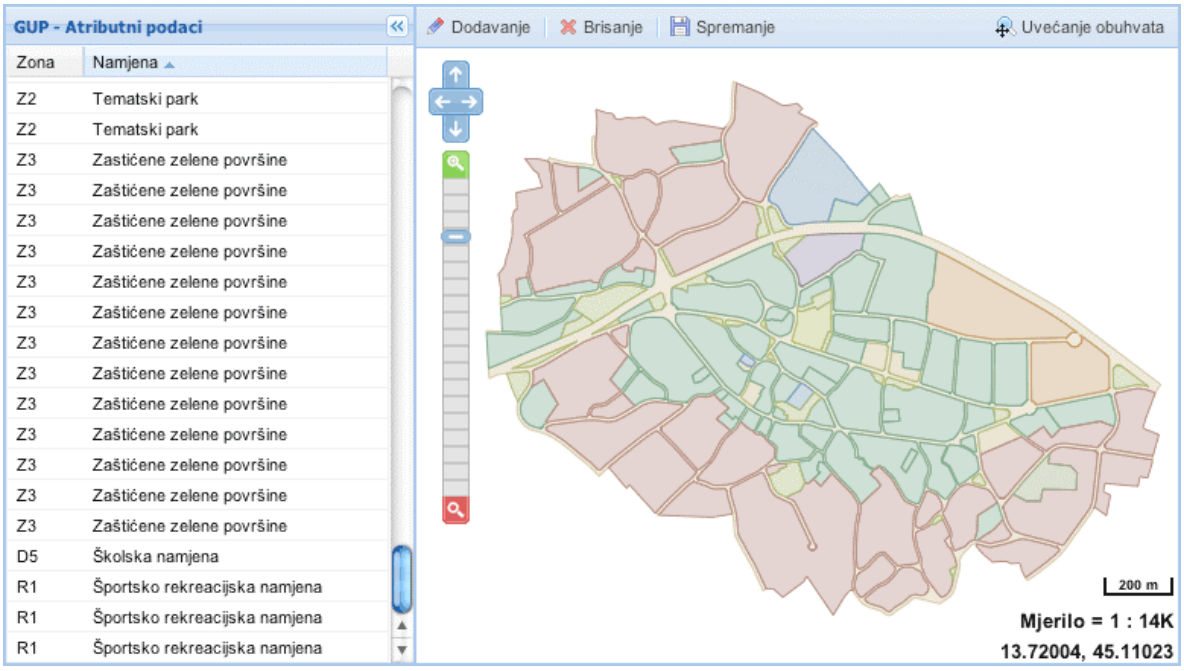
Nakon definiranja skupa atributa, skup je potrebno dodati u okvir s tabličnim prikazom atributa (*gridPanel*) i definirati njegove parametre (prikaz popisa, omogućavanje sortiranja po abecedi i slično). Isto tako, potrebno je i promijeniti klasu okvira iz *Ext.Panel* u *Ext.grid.ColumnModel*, kako bi se omogućio prikaz i kasnija izmjena atributnih podataka (URL 29). Ovo se može postići korištenjem sljedećeg isječka koda (ovaj isječak u potpunosti zamjenjuje prethodno definirani objekt *gridPanel*):

```
var gridPanel = new Ext.grid.EditorGridPanel({
	title: "GUP - Atributni podaci",
	region: "west",
	width: 270,
	collapsible: true,

	store: featureStore,

	cm: new Ext.grid.ColumnModel({
		defaults: {
			sortable: true,
			editor: {xtype: "textfield"}
		},
		columns: [
			{header: "Zona", dataIndex: "tip", width: 50},
			{header: "Namjena", dataIndex: "namjena", width: 200},
		]
	})
});
```

Neki parametri klase *Ext.grid.ColumnModel* su već otprije poznati (*title*, *region*, *width* i *collapsible*), a novo dodani je *store* (koji definira korišteni skup atributnih podataka). Ovaj parametar dohvaća prethodno definirani skup podataka nazvan *featureStore*.

Unutar samog okvira sada je potrebno definirati i parametar za popisivanje atributa u tabličnom prikazu, a to se postiže korištenjem klase *Ext.grid.ColumnModel* (URL 29). Tada je moguće definirati parametre poput *sortable* (omogućavanje sortiranja atributa po abecednom redu) i *editor* (omogućavanje izmjene tekstualnih polja). Također, uređivanjem parametra columns definiraju se imena zaglavlja stupaca tabličnog prikaza, te njihov sadržaj i širina u pikselima.

Isto tako, unutar *mapPanel* objekta dodan je novi redak "`header: false`", koji sakriva naslov kartografskog prikaza, a umjesto njega se postavlja prethodno definirana alatna traka s gumbima. Spremanjem koda i ponovnim pokretanjem aplikacije u internet pretraživaču, javlja se sljedeći prikaz (Slika 11).



Slika 11. Izgled aplikacije s implementiranim kartografskim prikazom i popisom atributa

4.4.3. Implementacija funkcionalnosti

Izmjena geometrije ili atributa obilježja trenutno neće izvoditi promjene na samim podacima na poslužitelju, zato što još nije implementiran mehanizam za promjenu stanja obilježja (eng. *feature state*) i slanje izvedenih promjena *WFS* transakcijom na poslužitelj. U ovom će se poglavlju, osim implementacije gore navedene funkcionalnosti, objasniti i dodavanje mehanizma za crtanje nove geometrije (i samim time dodavanje novih obilježja), te brisanje postojećih obilježja.

Za početak, potrebno je dodati sljedeći isječak koda nakon objekta *mapPanel*. Ovaj isječak omogućuje odabiranje i istovremenu izmjenu geometrije obilježja:

```
var modifyControl = new OpenLayers.Control.ModifyFeature(vectorLayer);
mapPanel.map.addControl(modifyControl);
modifyControl.activate();
```

Definirana je nova kontrola zvana *modifyControl*, koja se dodaje kartografskom prikazu i potom aktivira. U prvom se retku isječka definira jedan parametar, a to je ime sloja na kojem će kontrola moći izvršavati promjene (URL 28).

Nakon ove radnje potrebno je uskladiti odabir geometrije obilježja s odabirom njegovih atributa u okviru s tabličnim prikazom. Ovo se radi kako bi korisnik, primjerice, imao uvid u atribute poligona čiju je geometriju odabrao ili izmijenio (i obrnuto) (URL 28). Ova funkcionalnost se postiže dodavanjem sljedećeg isječka koda unutar *gridPanel* objekta, odmah ispod retka koji definira korišteni skup obilježja (redak "`store: featureStore,`"):

```
sm: new GeoExt.grid.FeatureSelectionModel({
       selectControl: modifyControl.selectControl,
       singleSelect: true
}),
```

Treći redak isječka omogućuje istovremeno označavanje samo jednog obilježja, zato što se istovremeno ne može mijenjati tekstualni sadržaj atributa većeg broja obilježja.

Spremanjem koda i ponovnim pokretanjem aplikacije omogućeno je odabiranje geometrije obilježja, koja prilikom odabira prikazuje vršne točke koje omogućuju promjenu oblika poligona. Također, u tabličnom se prikazu istovremeno označavaju (eng. *highlighting*) atributi obilježja čiji je poligon odabran.

Potom je potrebno definirati funkcionalnost dodavanja novih obilježja crtanjem poligona. To se postiže korištenjem sljedećeg isječka, koji se postavlja iznad *mapPanel* objekta:

```
var drawControl = new OpenLayers.Control.DrawFeature(
       vectorLayer,
       OpenLayers.Handler.Polygon,
       {handlerOptions: {multi: true}}
);
```

Ova funkcionalnost se postavlja iznad *mapPanel* objekta zato što mora biti definirana prije nego ju pozove gumb za dodavanje obilježja (koji se nalazi unutar *mapPanel* objekta). Dakle, potrebno je još dodati radnju pozivanja prethodno

definirane funkcionalnosti u gumb. To se čini dodavanjem retka "`control: drawControl`" na kraj definicije gumba, koja nakon dodavanja izgleda ovako:

```
new GeoExt.Action({
       text: "Dodavanje",
       iconCls: "dodavanje",
       enableToggle: true,

       control: drawControl,
}),
```

Također, potrebno je dodati i sljedeći redak nakon mapPanel objekta, kako bi se kartografskom prikazu dodala prije definirana funkcionalnost:

```
mapPanel.map.addControl(drawControl);
```

Vrijedi napomenuti da zbog načina rada mehanizma koji usklađuje odabir atributa i poligona nekog obilježja, svaki poligon koji se dodaje zajedno sa sobom stvara i prazan skup atributa koji ga opisuju (u ovom slučaju "Zona" i "Namjena"). Prazni atributi se potom popunjavaju željenim vrijednostima.

Prije objašnjenja implementacije funkcije izmjene i brisanja obilježja, potrebno je navesti promjene stanja koje obilježja poprimaju nakon što se izmjene njihovi atributi ili geometrija. Stanja opisuju radnju koja će izvršiti nad obilježjem prilikom sljedeće *WFS* transakcije (URL 28). Korištena stanja obilježja su:

- ***OpenLayers.State.INSERT*** – koristi se za označavanje tek stvorenih obilježja, koja se unose na poslužitelj prilikom sljedeće transakcije. Automatski se dodjeljuje obilježju koje je dodano funkcijom crtanja.
- ***OpenLayers.State.UPDATE*** – koristi se za označavanje obilježja čiji su atributi ili geometrija promijenjeni, te je njih potrebno osvježiti prilikom sljedeće transakcije.
- ***OpenLayers.State.DELETE*** – označava obilježje odabrano za brisanje, koje se uklanja iz zapisa na poslužitelju prilikom sljedeće transakcije.

Korištenjem ovih stanja moguće je implementirati funkciju izmjene atributa obilježja. U *gridPanel* objekt (predstavlja tablični prikaz atributa) se dodaje sljedeći

isječak koda, odmah nakon retka koji definira korišteni skup obilježja ("`store: featureStore,`"):

```
listeners: {
	afteredit: function(e) {
		var feature = e.record.get("feature");
		if(feature.state !== OpenLayers.State.INSERT) {
			feature.state = OpenLayers.State.UPDATE;
		}
	}
},
```

Ovaj dio koda prati sve promjene unesene u tekstualna polja tabličnog prikaza, te ukoliko promijenjeni atributi nisu tek dodani (tj. nemaju stanje *OpenLayers.State.INSERT*), obilježava ih se stanjem za ažuriranje (eng. *update*) korištenjem stanja *OpenLayers.State.UPDATE*. Prilikom sljedeće *WFS* transakcije, atributi ovako označenih obilježja se ažuriraju na poslužitelju. Potrebno je napomenuti da mehanizam izmjene geometrije obilježja (definiran *OpenLayers*-om) automatski mijenja stanje obilježja kojem taj poligon pripada, te zato ovu funkcionalnost nije potrebno zasebno implementirati.

Potom je potrebno implementirati funkcionalnost brisanja, što se izvodi u dva koraka. Za početak, potrebno je spriječiti prikazivanje atributa i geometrije svih obilježja sa stanjem za brisanje. To se postiže korištenjem filtra koji se dodaje u skup obilježja. Prije spomenuti filtar se definira u sljedećem isječku koda, koji se postavlja u objekt *featureStore*, na sam njegov kraj (nakon retka "`layer: vectorLayer,`"):

```
addFeatureFilter: function(feature) {
	return feature.state !== OpenLayers.State.DELETE;
}
```

U drugom je koraku potrebno definirati rad gumba za brisanje obilježja, a to se postiže dodavanjem sljedećeg isječka koda na dno definicije prethodno spomenutog gumba:

```
handler: function() {
	gridPanel.getSelectionModel().each(function(rec) {
```

```
            var feature = rec.get("feature");
            modifyControl.unselectFeature(feature);
            featureStore.remove(rec);
            if (feature.state !== OpenLayers.State.INSERT) {
                  feature.state = OpenLayers.State.DELETE;
                  vectorLayer.addFeatures([feature]);
            }
      });
}
```

Ovim dijelom koda se miče odabir s obilježja koje se briše, te mu se potom postavlja stanje za brisanje (koje će utjecati na njegovo filtriranje iz skupa obilježja, definirano u prethodnom isječku). Prilikom sljedeće *WFS* transakcije ovakvo obilježje će biti izbrisano s poslužitelja.

Spremanjem promjena koda i ponovnim pokretanjem aplikacije u internet pregledniku, moguće je isprobati rad funkcije brisanja.

Nakon implementiranja funkcije brisanja, potrebno je definirati način rada funkcije spremanja, zato što se trenutno promjene obilježja koje korisnik izvršava ne šalju na poslužitelj. Kao prvi korak implementacije, dodaje se sljedeći redak iznad bloka koda koji definira vektorski sloj. Ovaj redak će se pozivati kako bi pokrenuo *WFS* transakciju kojom će se operacije dodavanja, izmjene i brisanja obilježja izvesti na samom poslužitelju:

```
var saveStrategy = new OpenLayers.Strategy.Save();
```

Potom se u *strategies* popisu parametara vektorskog sloja dodaje redak *saveStrategy*, koji definira ponašanje sloja prilikom spremanja pozivajući prethodno definiranu funkcionalnost. Početak definicije vektorskog sloja tada izgleda kao što je prikazano u sljedećem isječku:

```
var vectorLayer = new OpenLayers.Layer.Vector("GUP", {
      strategies: [
            new OpenLayers.Strategy.Fixed(),
            saveStrategy
      ],

      ...
```

Za mogućnost korištenja funkcije spremanja, potrebno ju je pozvati pritiskom gumba za spremanje. Ovo se obavlja dodavanjem koda u definiciju gumba, koji nakon izmjene izgleda ovako:

```
new GeoExt.Action({
	text: "Spremanje",
	iconCls: "spremanje",

	handler: function() {
		featureStore.commitChanges();
		saveStrategy.save();
	}
}),
```

Pritiskom gumba spremanja, pokreće se izvršavanje promjena u skupu obilježja, a potom se sljedećim retkom koda poziva prethodno definirano izvršavanje *WFS* transakcije, koja poslužitelju šalje promjene.

Kao zadnji korak implementiranja funkcionalnosti aplikacije ostaje definiranje rada gumba za uvećanje obuhvata. Nadopunjeni kod gumba za uvećanje obuhvata izgleda kao što je prikazano u isječku:

```
new GeoExt.Action({
	text: "Uvećanje obuhvata",
	iconCls: "obuhvat",

	control: new OpenLayers.Control.ZoomToMaxExtent(),
	map: map
}),
```

Ovim se dijelom koda poziva gotova funkcionalnost preuzeta iz OpenLayers skupa gotovih alata (*OpenLayers.Control.ZoomToMaxExtent()*), a sljedećim retkom se definira kartografski prikaz kojem se mijenja mjerilo korištenjem ovog gumba. Funkcioniranje koda se sada može provjeriti njegovim spremanjem i ponovnim pokretanjem aplikacije u internet pregledniku.

5. Upute za rad s aplikacijom

Korisnika će prilikom pokretanja aplikacije dočekati prozor internet pretraživača popunjen sučeljem *webGIS* aplikacije. U središnjem dijelu sučelja nalazi se kartografski prikaz Generalnog urbanističkog plana Rovinjskog sela, a s lijeve strane je postavljen tablični prikaz atributnih podataka.

U gornjem lijevom kutu kartografskog prikaza postavljeni su alati koji omogućuju kretanje po kartografskom prikazu i promjenu njegovog mjerila. Kretanje po kartografskom prikazu se vrši odabirom jedne od plavih strelica, ovisno o smjeru kojim se korisnik želi kretati. Također, moguće je koristiti i tipkovničke strelice za postizanje iste funkcionalnosti. Osim toga, kretati prikazom se može i pomicanjem miša, ako se pritisne i drži lijevi gumb miša dok se pokazivač nalazi na bijeloj pozadini.

Promjena mjerila se izvršava korištenjem klizača mjerila, koji se vuče prema gore kako bi korisnik dobio prikaz generalnog urbanističkog plana u krupnijem mjerilu i pomicanjem klizača prema dolje, kako bi se postigao prikaz u sitnijem mjerilu. Osim ovog načina, promjenu mjerila je moguće kontrolirati kotačićem miša, korištenjem tipki "+" i "-" na tipkovnici ili odabirom zelenog gumba iznad skale s klizačem (za postizanje krupnijeg mjerila), odnosno crvenog ispod skale (za sitnije mjerilo). Odabir mjerila koje pokriva cijelo područje generalnog urbanističkog plana Rovinjskog sela vrši se klikom na gumb "Uvećanje obuhvata", koji se nalazi u gornjem desnom kutu sučelja aplikacije.

U donjem desnom kutu kartografskog prikaza nalaze se različite informacije o trenutnom prikazu. Informacije uključuju (navedene po redu, od najviše prema najnižoj) liniju mjerila, trenutno mjerilo i trenutne koordinate položaja pokazivača miša (izražene u stupnjevima).

Odabir zone generalnog urbanističkog plana moguć je klikom na poligon koji predstavlja tu zonu. Nakon odabira, poligon zone će poprimiti plavu boju i biti će obrubljen točkama, čijim se povlačenjem može izmijeniti njegova geometrija. Također, prilikom odabira poligona zone označiti će se i njezini atributni podaci u

tabličnom prikazu s lijeve strane aplikacije. Odabir poligona se može ukloniti ponovnim klikom pokazivača miša na poligon, ili klikom na prazno bijelo područje.

Klikom na jedan od atributa u tabličnom prikazu s lijeve strane aplikacije odabrati će se taj par atributa. Također, prilikom odabira atributa, odabrati će se i poligon zone u kartografskom prikazu.

Izmjena atributnih podataka je moguća dvostrukim klikom na jedno od dostupnih tekstualnih polja s atributima, nakon čega se može izmijeniti njihov sadržaj. Izmjena teksta se prihvaća pritiskom tipke *Enter* ili *Tab*, a okvir za unos teksta se pomiče na sljedeći atribut. Važno je napomenuti da će prije spremanja izmjena modificirani atributi biti označeni crvenim trokutom u gornjem lijevom kutu pripadajućeg tekstualnog polja.

Atributni podaci se mogu i razvrstavati po abecedi (uzlazno i silazno). Redoslijed razvrstavanja se mijenja klikom na zaglavlje stupca tabličnog prikaza. Isto tako, moguće je isključiti pojedine stupce odabirom strelice sa lijeve strane zaglavlja stupca, nakon čega se u novootvorenom izborniku odabire podizbornik "Odabir stupca" i miče kvačica pored imena stupca s atributima, kako bi se uklonio prikaz istog. Vraćanje prikaza stupca sa željenim atributima radi se ponovnim klikom na kvačicu.

Dodavanje novih zona generalnog urbanističkog plana se vrši aktiviranjem gumba "Dodavanje". Klikom na bilo kojie područje kartografskog prikaza započinje se s crtanjem geometrije. Zadnja točka poligona dodaje s dvostrukim klikom. Nakon toga je moguće na isti način dodati i još nekoliko poligona ili prekinuti funkciju dodavanja zona ponovnim odabirom gumba "Dodavanje". Nakon što se nacrta geometrija, u tabličnom prikazu atributnih podataka pojavljuje se prazno polje s mjestima za upis atributa upravo dodane zone. Geometrija koja nema popunjene atributne podatke neće se prikazivati prilikom sljedećeg pokretanja aplikacije.

Važno je napomenuti da je funkcija dodavanja uključena sve dok je aktiviran gumb "Dodavanje". To znači da se tada nije moguće pomicati po kartografskom prikazu povlačenjem miša, zato što će ova radnja pokrenuti dodavanje novog poligona.

Brisanje zone generalnog urbanističkog plana se vrši odabiranjem njezine geometrije (u kartografskom prikazu) ili atributa (u tabličnom prikazu), te odabirom tipke "Brisanje".

Za kraj je potrebno napomenuti da je nakon izvršenja željenih promjena iste potrebno i spremiti. Odabirom gumba "Spremanje", izmjene se šalju na poslužitelj. Spremanjem će izvedene promjene biti zapamćene i vidljive prilikom sljedećeg pokretanja aplikacije. Također, crveni trokuti u gornjem lijevom rubu tekstualnog polja izmijenjenih ili novo dodanih atributnih podataka će prilikom spremanja nestati, kako bi se korisniku dalo do znanja da su promjene ažurirane.

6. Zaključak

Ovim je radom prikazan način izrade jednostavnije *webGIS* aplikacije na primjeru generalnog urbanističkog plana mjesta Rovinjsko selo. Korištenjem već definiranih standarda za manipulaciju prostornim podacima i skupova gotovih programskih alata, poput *OpenLayers*-a i *GeoExt*-a, moguće je s relativno malom količinom programskog koda implementirati funkcionalnost i sučelje ovakve aplikacije.

Također, prednost *webGIS* aplikacija (u usporedbi s klasičnim *GIS* aplikacijama, koje se pokreću na računalu korisnika) je mogućnost njihovog izvođenja neovisno o operativnom sustavu koji korisnik posjeduje na svojem računalu. Jedini uvjet za korištenje ovakve vrste aplikacije je mogućnost korištenja internet preglednika i pristupa internetu. No, uvjet pristupa internetu je postavljen i klasičnim *GIS* aplikacijama u slučaju da se prostorni podaci preuzimaju s udaljenog poslužitelja. Isto tako, činjenicom da se ovakva aplikacija izvršava na računalu korisnika (eng. *client side web application*), smanjuje se opterećenje poslužitelja i omogućuje istodoban rad većeg broja korisnika na istom skupu prostornih podataka.

Za kraj, bitno je napomenuti da je vrlo malenim izmjenama koda (primjerice, izmjenom samo adrese poslužitelja prostornih podataka) ovakvu vrstu aplikacije moguće prilagoditi radu s podacima koji se skladište na nekim drugim poslužiteljima. Također, moguće je aplikaciju brzo prilagoditi i za rad s drukčijim skupom prostornih podataka, tj. ista nije ograničena na rad samo s generalnim urbanističkim planovima.

7. Literatura

Longley, P. A., Goodchild, M. F., Maguire, D. J., Rhind D.W., (2005): *Geographic Information Systems and Science*, John Wiley & Sons, United Kingdom.

Davis S. (2007): *GIS for Web Developers: Adding 'Where' to Your Web Applications*, Pragmatic Bookshelf, United Kingdom.

Mitchell T. (2005): *Web Mapping Illustrated: Using Open Source GIS Toolkits*, O'Reilly Media, United Kingdom.

Popis *URL*-ova:

URL 1. HTML, http://en.wikipedia.org/wiki/HTML, 2.8.2010.

URL 2. A brief history of JavaScript, http://javascript.about.com/od/reference/a/history.htm, 2.8.2010.

URL 3. JavaScript, http://en.wikipedia.org/wiki/Javascript, 2.8.2010.

URL 4. Why is Javascript called Javascript, if it has nothing to do with Java?, http://stackoverflow.com/questions/2475505/why-is-javascript-called-javascript-if-it-has-nothing-to-do-with-java, 2.8.2010

URL 5. Open Geospatial Consortium, http://en.wikipedia.org/wiki/Open_Geospatial_Consortium, 11.8.2010.

URL 6. Web Map Service, http://www.opengeospatial.org/standards/wms, 11.8.2010.

URL 7. Web Map Service, http://docs.geoserver.org/stable/en/user/services/wms/index.html, 11.8.2010.

URL 8. WMS Tile Caching, http://wiki.osgeo.org/wiki/WMS_Tile_Caching, 13.8.2010.

URL 9. Tile Map Service Specification, http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification, 13.8.2010.

URL 10. KML Tutorial, http://code.google.com/apis/kml/documentation/kml_tut.html, 13.8.2010.

URL 11. Keyhole Markup Language, http://en.wikipedia.org/wiki/Keyhole_Markup_Language_2.2, 13.8.2010.

URL 12. Web Feature Service, http://en.wikipedia.org/wiki/Web_Feature_Service, 5.8.2010.

URL 13. Web Feature Service,
http://docs.geoserver.org/stable/en/user/services/wfs/index.html, 5.8.2010.

URL 14. How a Get Feature Works,
http://geoserver.org/display/GEOSDOC/How+a+GetFeature+request+works, 9.8.2010.

URL 15. Introduction to GML,
http://www.w3.org/Mobile/posdep/GMLIntroduction.html, 10.8.2010.

URL 16. Geography Markup Language,
http://en.wikipedia.org/wiki/Geography_Markup_Language, 10.8.2010.

URL 17. GML Application Schemas,
http://en.wikipedia.org/wiki/GML_Application_Schemas, 10.8.2010.

URL 18. Shapefile,
http://en.wikipedia.org/wiki/ESRI_shape, 15.8.2010.

URL 19. ESRI Shapefile Technical Description,
http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf, 15.8.2010.

URL 20. Geoserver user manual,
http://docs.geoserver.org/stable/en/user/, 15.8.2010.

URL 21. What is OpenLayers?,
http://docs.openlayers.org/, 16.8.2010.

URL 22. MetaCarta,
http://en.wikipedia.org/wiki/MetaCarta, 16.8.2010.

URL 23. ExtJS,
http://en.wikipedia.org/wiki/Extjs, 17.8.2010.

URL 24. Geoserver user manual,
http://docs.geoserver.org/2.0.1/user/, 1.7.2010.

URL 25. MGI / Balkans Coordinate Systems,
http://spatial-analyst.net/wiki/index.php?title=MGI_/_Balkans_coordinate_systems, 18.8.2010.

URL 26. Spatial Referencing System,
http://en.wikipedia.org/wiki/Spatial_referencing_system, 18.8.2010.

URL 27. Layouts in ExtJS,
http://www.packtpub.com/article/layouts-in-ext-js, 18.8.2010.

URL 28. http://svn.opengeo.org/workshops/modules/geoext/, 18.8.2010.

URL 29. http://svn.opengeo.org/workshops/modules/openlayers/, 19.8.2010.

8. Popis slika

9. Prilozi

9.1. Prilog 1 – Izvorni kod aplikacije

```
<html>

<head>

<!--  Naslov stranice koji se pokazuje na vrhu internet preglednika -->
<title>GUP Rovinjskog sela</title>

<!--  Postavljanje podrške za dijakritičke znakove  -->
<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<!--  Uključivanje vanjske datoteke OpenLayers-a -->
<script src="OpenLayers/OpenLayers.js" type="text/javascript"></script>

<!--  Uključivanje vanjskih datoteka ExtJS-a i stila za prikaz njegovih elemenata -->
<script src="ExtJS/adapter/ext/ext-base.js" type="text/javascript"></script>
<script src="ExtJS/ext-all.js"  type="text/javascript"></script>
<link rel="stylesheet" type="text/css" href="ExtJS/resources/css/ext-all.css" />

<!--  Uključivanje vanjskih datoteka GeoExt-a i stila za prikaz njegovih elemenata-->
<script src="GeoExt/script/GeoExt.js" type="text/javascript"></script>
<link rel="stylesheet" type="text/css" href="GeoExt/resources/css/geoext-all.css" />

<!-- Definiranje stila za prikaz GoeExt kontrola -->
<style>
        .obuhvat        {background-image: url(Elementi/obuhvat.png)         !important;}
        .spremanje      {background-image: url(Elementi/spremanje.png)       !important;}
        .brisanje       {background-image: url(Elementi/brisanje.png)        !important;}
        .dodavanje      {background-image: url(Elementi/dodavanje.png)       !important;}
</style>

<script type="text/javascript">

// Pokretanje koda pisanog GoeExt skupom gotovih alata.
Ext.onReady(function() {

// Postavljanje novog stila za vizualizaciju geometrije s pretpostavljenim postavkama.
var styleMap = new OpenLayers.StyleMap();

// Definiranje pravila po kojima se stil dodjeljuje određenim poligonima.
var styleRules = [

        // Postavljanje novog pravila
        new OpenLayers.Rule({

                //Korištenje gotove OpenLayers funkcije za filtriranje željenih poligona.
                filter: new OpenLayers.Filter.Comparison({

                        /*
                        Definiranje tipa usporedbe. Koristi se
                        "OpenLayers.Filter.Comparison.LIKE" uvjet, zato što određeni
                        poligoni imaju kao tip zone postavljeno više vrijednosti
                        koje započinju istim slovom (primjerice GUP zone M1 i M2,
                        ili Z1, Z2 i Z3, itd.). Korištenjem LIKE uvjeta, omogućuje
                        se definiranje samo prvog slova tipa GUP zone, a njeni
                        podtipovi poprimaju isti stil.

                        Pravila za vizualiziranje ostalih GUP zona su definirana
                        na identičan način.
                        */
                        type: OpenLayers.Filter.Comparison.LIKE,
                        // Vrsta atributa koji se uspoređuje.
                        property: "tip",
                        // Vrijednost atributa za usporedbu - Zona C.
                        value: "C"
                }),

                /*
                Definiranje stila koji će se primjeniti na iznad definiranu
```

```
                GUP zonu. Osim promjene boje obruba i ispune poligona,
                moguće je mijenjati i njihovu prozirnost, debljinu obruba,
                stil prikazivanja točaka (za točkaste podatke) i slično.

                Za parametre koji nisu navedeni postavljaju se
                pretpostavljene (eng. default) vrijednosti.
                */
                symbolizer: {
                        strokeColor: "d1bf94", // Boja obruba, u heksadecimalnom zapisu.
                        fillColor: "d1bf94"    // Boja ispune, u heksadecimalnom zapisu.
                },

                /*
                Korištenjem uvjeta "elseFilter: true" omogućuje se prikazivanje
                točaka (eng. handles) za promjenu oblika poligona. Bez ovog retka
                pravila bi filtrirala (tj. sakrivala) svaki oblik geometrije koji
                nije poligon, tj. točke i (poli)linije se ne bi prikazivale.
                */
                elseFilter: true
        }),

        new OpenLayers.Rule({
                filter: new OpenLayers.Filter.Comparison({
                        type: OpenLayers.Filter.Comparison.LIKE,
                        property: "tip",
                        value: "G"
                }),
                symbolizer: {
                        strokeColor: "abb858",
                        fillColor: "abb858"
                },
                elseFilter: true
        }),

        new OpenLayers.Rule({
                filter: new OpenLayers.Filter.Comparison({
                        type: OpenLayers.Filter.Comparison.LIKE,
                        property: "tip",
                        value: "Z"
                }),
                symbolizer: {
                        strokeColor: "a1b569",
                        fillColor: "a1b569"
                },
                elseFilter: true
        }),

        new OpenLayers.Rule({
                filter: new OpenLayers.Filter.Comparison({
                        type: OpenLayers.Filter.Comparison.LIKE,
                        property: "tip",
                        value: "T"
                }),
                symbolizer: {
                        strokeColor: "8fa37a",
                        fillColor: "8fa37a"
                },
                elseFilter: true
        }),

        new OpenLayers.Rule({
                filter: new OpenLayers.Filter.Comparison({
                        type: OpenLayers.Filter.Comparison.LIKE,
                        property: "tip",
                        value: "M"
                }),
                symbolizer: {
                        strokeColor: "6fa082",
                        fillColor: "6fa082"
                },
                elseFilter: true
        }),

        new OpenLayers.Rule({
                filter: new OpenLayers.Filter.Comparison({
                        type: OpenLayers.Filter.Comparison.LIKE,
                        property: "tip",
                        value: "R"
                }),
```

```
                symbolizer: {
                        strokeColor: "698faa",
                        fillColor: "698faa"
                },
                elseFilter: true
        }),

        new OpenLayers.Rule({
                filter: new OpenLayers.Filter.Comparison({
                        type: OpenLayers.Filter.Comparison.LIKE,
                        property: "tip",
                        value: "S"
                }),
                symbolizer: {
                        strokeColor: "aa7d76",
                        fillColor: "aa7d76"
                },
                elseFilter: true
        }),

        new OpenLayers.Rule({
                filter: new OpenLayers.Filter.Comparison({
                        type: OpenLayers.Filter.Comparison.LIKE,
                        property: "tip",
                        value: "K"
                }),
                symbolizer: {
                        strokeColor: "8478a0",
                        fillColor: "8478a0"
                },
                elseFilter: true
        }),

        new OpenLayers.Rule({
                filter: new OpenLayers.Filter.Comparison({
                        type: OpenLayers.Filter.Comparison.LIKE,
                        property: "tip",
                        value: "I"
                }),
                symbolizer: {
                        strokeColor: "be8f5b",
                        fillColor: "be8f5b"
                },
                elseFilter: true
        }),

        new OpenLayers.Rule({
                filter: new OpenLayers.Filter.Comparison({
                        type: OpenLayers.Filter.Comparison.LIKE,
                        property: "tip",
                        value: "D"
                }),
                symbolizer: {
                        strokeColor:"b8b172",
                        fillColor: "b8b172"
                },
                elseFilter: true
        })
];      // Završetak definiranja pravila za vizualizaciju GUP zona.

// Definirana pravila za vizualiziranje se dodaju u pretpostavljeni stil.
styleMap.styles["default"].addRules(styleRules);

/*
Definiranje obuhvata (eng. bounds). Koordinate su
navedene u X,Y parovima i definiraju dvije točke
suprotnih vrhova graničnog okvira (eng. Bounding Box).
*/
var rovinjskoSelo = new OpenLayers.Bounds(
        1525769.07602, 5637546.13799,
        1528114.73732, 5639399.73592
);

/*
Postavljanje naziva kartografskog prikaza
koji će koristiti ostali elementi ove aplikacije.
*/
var map = new OpenLayers.Map("map", {
```

```
        /*EPSG kod projekcije. Koristi se ista projekcija
        kao što je definirano u sloju (Google Mercator).*/
        projection: new OpenLayers.Projection("EPSG:900913"),

        /*Prikazivanje koordinata kursora u stupnjevima
        (prema WGS84 datumu - EPSG kod 3426), umjesto u metrima.*/
        displayProjection: new OpenLayers.Projection("EPSG:4326"),

        // Jedinice korištene za prikaz linije mjerila, u ovom slučaju metri.
        units: "m",
        // Najveći broj razina promjene mjerila (eng. zoom).
        numZoomLevels: 19,
        // Postavljanje najveće razine promjene mjerila koja se može koristiti.
        maxResolution: 156543.0339,
        /* Definiranje obuhvata (eng. bounds). Pozivaju se već
        spremljene koordinate u istoimenom objektu (rovinjskoSelo).*/
        maxExtent: rovinjskoSelo,

        // Gotovi OpenLayers alati za rad s kartografskim prikazom.
        controls: [
                // Omogućavanje pomicanja kartografskog prikaza (eng. panning).
                new OpenLayers.Control.Navigation(),
                /*Strelice za pomicanje kartograskog prikaza
                i skala promjene mjerila (eng. Pan and Zoom bars).*/
                new OpenLayers.Control.PanZoomBar(),
                // Ispis koordinata položaja srelice miša.
                new OpenLayers.Control.MousePosition(),
                /*Korištenje tipkovničkih kratica za navigaciju:
                strelice za pomicanje; "+" i "-" za promjenu mjerila.*/
                new OpenLayers.Control.KeyboardDefaults(),
                // Ispis trenutnog mjerila.
                new OpenLayers.Control.Scale(),
                // Linija mjerila (eng. scale line).
                new OpenLayers.Control.ScaleLine()
        ]
});

// Pokretanje WFS transakcije - spremanje promjena.
var saveStrategy = new OpenLayers.Strategy.Save();

/*
Definiranje vektorskog sloja, koji sadrži zone Generalnog
urbanističkog plana. Postavljanje vrste sloja (vektorski)
i definiranje imena.
*/
var vectorLayer = new OpenLayers.Layer.Vector("GUP", {

        strategies: [
                /*Podaci se dohvaćaju samo jednom prije osvježavanja
                (eng. refresh) sadržaja stranice.*/
                new OpenLayers.Strategy.Fixed(),

                /*
                Zbog toga što je radnja spremanja promjena prilikom WFS transakcije
                po definiciji atomarna (eng. atomic - ako jedna od radnji u skupu
                promjena propadne, tada se cijela radnja spremanja prekida),
                potrebno je prvo definirati parametre radnje spremanja.
                */

                /*Definiranje načina rada spremanja pozivanjem prethodno
                definirane radnje spremanja.*/
                saveStrategy
        ],

        // Korišteni protokol (u ovom slučaju OGC Web Feature Service).
        protocol: new OpenLayers.Protocol.WFS({
                // Verzija korištenog protokola - aktualna verzija 1.1.0
                version: "1.1.0",
                /*EPSG kod projekcije u kojoj se prostorni podaci dohvaćaju
                s poslužitelja (u ovom slučaju Google Mercator).*/
                srsName: "EPSG:900913",
                // Adresa poslužitelja.
                url: "http://localhost:8080/geoserver/wfs",
                /*Naziv imenskog prostora u kojem su spremljena željena
                obilježja (eng. features). Koristi se isti naziv kao
                što je definiran u Geoserver-u*/
                featureNS: "rovinjsko-selo",
                // Naziv obilježja za dohvat.
                featureType: "gup"
```

```
        }),

        /*Ovaj je parametar potrebno namjestiti ako je
        ovaj sloj jedini na kartografskim prikazu.*/
        isBaseLayer: true,
        // EPSG kod korištene projekcije (u ovom slučaju Google Mercator).
        projection: new OpenLayers.Projection("EPSG:900913"),
        // Definiranje pravila koja će se koristiti za vizualizaciju ovog sloja.
        styleMap: styleMap
});

// Dodavanje prethodno definiranog sloja u kartografski prikaz.
map.addLayers([vectorLayer]);

// Definiranje radnje za crtanje nove geometrije.
var drawControl = new OpenLayers.Control.DrawFeature(
        // Odabir sloja na kojem će se crtati novi poligoni.
        vectorLayer,
        OpenLayers.Handler.Polygon,
        {handlerOptions: {multi: true}}
);

/*
Okvir (eng. panel) za iscrtavanje kartografskog prikaza,
koji će se postaviti u centar stranice, uz pomoć ExtJS-a.
*/
var mapPanel = new GeoExt.MapPanel({
        /*Naziv okvira koji će se prikazivati na
        naslovnoj traci, ako ista nije skrivena.*/
        title: "Kartografski prikaz",
        // Dio prozora internet preglednika u koji se okvir postavlja.
        region: "center",
        // Micanje prikaza naslova s vrha okvira.
        header: false,
        /*Definiranje objekta kartografskog prikaza
        koji će se iscrtavati u okviru.*/
        map: map,

        // Definiranje gornje alatna trake sa gumbina (eng. toolbar).
        tbar: [

        //Gumb "Dodavanje".
        new GeoExt.Action({
                // Natpis na gumbu.
                text: "Dodavanje",
                // Ikona definirana stilom uključenim u zaglavlje ove datoteke.
                iconCls: "dodavanje",
                /*Odabirom gumba, funkcija ostaje upaljena,
                sve dok se isti ponovno ne odabere.*/
                enableToggle: true,
                /*Definiranje funkcije gumba - poziva se prethodno
                definirana radnja za crtanje nove geometrije.*/
                control: drawControl,
        }),

        "-", // Postavljanje crte koja odjeljuje dva gumba.

        //Gumb "Brisanje".
        new GeoExt.Action({
                text: "Brisanje",
                iconCls: "brisanje",

                /*Handler definira funkcionalnost gumba. Koristi
                se umjesto control parametra (prikazanog na primjeru
                gornjeg gumba), ako se funkcionalnost piše izravno
                u gumb, umjesto da se poziva već gotova funkcija.*/
                handler: function() {
                        /*
                        Prvi dio koda (sljedeća četiri retka) miče geometriju
                        prostornih podataka s kartografskog prikaza i atributne
                        podatke iz skupa atributa.
                        */
                        gridPanel.getSelectionModel().each(function(rec) {
                                // Dohvaćanje obilježja.
                                var feature = rec.get("feature");
                                // Micanje odabira sa odabranog obilježja.
                                modifyControl.unselectFeature(feature);
                                // Micanje odabranog obilježja iz skupa atributa.
                                featureStore.remove(rec);
```

```
                                /*
                                Drugi dio koda postavlja stanje obilježja (koje
                                nije tek dodano, odnosno nema stanje OpenLayers.State.INSERT)
                                na OpenLayers.State.DELETE, kako bi se isto obrisalo sa
                                poslužitelja prilikom izvršavanja WFS transakcije.
                                */
                                // Provjera da li je obilježje upravo dodano.
                                if (feature.state !== OpenLayers.State.INSERT) {
                                        // Ako nije, stanje mu se postavlja na brisanje.
                                        feature.state = OpenLayers.State.DELETE;
                                        /*Modificirano obilježje se vraća u skup obilježja sa
                                        novim stanjem, koje sprječava prikazivanje njegove
                                        geometrije (za atribute se brine drugi kod).*/
                                        vectorLayer.addFeatures([feature]);
                                }
                        });
                }
        }),

        "-",

        //Gumb "Spremanje".
        new GeoExt.Action({
                text: "Spremanje",
                iconCls: "spremanje",

                /*
                Pokreće se radnja spremanja promjena prostornih podataka,
                prema prije definiranim parametrima (saveStrategy objekt
                u prijašnjem dijelu koda).
                */
                handler: function() {
                        /*Izvršavanje promjena u skupu atributa.
                        Potrebno zbog atomarnog principa rada WFS transakcija.*/
                        featureStore.commitChanges();
                        // Pozivanje prije definirane radnje spremanja promjena.
                        saveStrategy.save();
                }
        }),

        "->",   // Potiskivanje svih gumba nakon ovog retka u desni kraj alatne trake.

        //Gumb "Uvećanje obuhvata".
        new GeoExt.Action({
                text: "Uvećanje obuhvata",
                iconCls: "obuhvat",

                /*Pozivanje gotove OpenLayers alata za uvećanje
                obuhvata (eng. zoom to extents).*/
                control: new OpenLayers.Control.ZoomToMaxExtent(),
                // Definiranje karte s kojom ovaj alat radi.
                map: map
        }),
        ]
});

// Dodavanje alata koja omogućava crtanje nove geometrije.
mapPanel.map.addControl(drawControl);

// Kontrola koja omogućuje izmjene sloja.
var modifyControl = new OpenLayers.Control.ModifyFeature(vectorLayer);
// Dodavanje kontrole na postojeći kartografski prikaz.
mapPanel.map.addControl(modifyControl);
// Aktiviranje kontrole.
modifyControl.activate();

/*Skup obilježja (eng. feature store).
Podaci se dohvaćaju izvektorskog sloja.*/
var featureStore = new GeoExt.data.FeatureStore({
        // Nazivi i tipovi dohvaćanih atributa.
        fields: [
                {name: "tip", type: "string"},
                {name: "namjena", type: "string"}
        ],

        // Definiranje sloja s kojeg se preuzimaju atributni podaci.
        layer: vectorLayer,
```

```
        /*
        Sljedeće linije koda spriječavaju da se
        atributni podaci obilježja označenih za brisanje
        (obilježja sa stanjem OpenLayers.State.DELETE)
        pojavljuju na popisu.
        */
        addFeatureFilter: function(feature) {
                return feature.state !== OpenLayers.State.DELETE;
        }
});

// Okvir s popisom atributnih podataka.
var gridPanel = new Ext.grid.EditorGridPanel({
        title: "GUP - Atributni podaci",
        region: "west",
        // Širina okvira u pikselima.
        width: 270,
        // Omogućuje se sakrivanje (eng. minimizing) okvira.
        collapsible: true,

        // Koristi se prije definirani skup atributnih podataka "store".
        store: featureStore,

        // Praćenje promjene na temelju kojih se pokreću naredbe u ovom bloku.
        listeners: {
                afteredit: function(e) {
                        // Dohvaćanje promijenjenog obilježja.
                        var feature = e.record.get("feature");
                        // Provjera da li je odabrano obilježje tek dodano.
                        if(feature.state !== OpenLayers.State.INSERT) {
                                // Ako nije, onda se njegov status postavlja na ažuriranje.
                                feature.state = OpenLayers.State.UPDATE;
                        }
                }
        },

        /*Usklađivanje odabira atributnih podataka i
        označavanja odgovarajućih poligona.*/
        sm: new GeoExt.grid.FeatureSelectionModel({
                // Omogućavanje izmjene poligona uz njegov odabir
                selectControl: modifyControl.selectControl,
                /*Istovremeno se može odabrati samo jedan poligon
                ili jedan par atributnih podataka (par Zona - Namjena).*/
                singleSelect: true
        }),

        // Definiranje prikaza atributnih podataka (u dva stupca).
        cm: new Ext.grid.ColumnModel({
                defaults: {
                        // Omogućavanje sortiranja atributa po abecedi, uzlazno i silazno.
                        sortable: true,
                        // Omogućavanje izmjene teksta.
                        editor: {xtype: "textfield"}
                },
                /*Naslovi stupaca, tip podataka koji sadržavaju
                (u ovom slučaju tip GUP zone i njezina namjena)
                i širina stupca u pikselima.*/
                columns: [
                        {header: "Zona", dataIndex: "tip", width: 50},
                        {header: "Namjena", dataIndex: "namjena", width: 200},
                ]
        })
});

/*Definiranje glavnog okvira, koji sadržava
kartografski prikaz i popis atributnih podataka.*/
var viewport = new Ext.Viewport({
        /*Rastezanje okvira preko cijele površine
        prozora internet preglednika.*/
        layout: "fit",
        items: {
                /*Okviri kartografskog prikaza ipopisa
                atributa se razdvajaju graničnikom.*/
                layout: "border",
                // Izlistavanje okvira koji su sadržani u ovom prikazu (eng. layout).
                items: [mapPanel, gridPanel]
        }
});
})
```

```
</script>
</head> <!-- Zatvaranje zaglavlja HTML koda stranice -->

<body>
	<!-- Tijelo stranice koje zauzima glavni okvir s
	kartografskim prikazom i popisom atributnih podataka -->
</body>

</html>
```

9.2. Sadržaj priloženog optičkog medija

Br.	Putanja do datoteke	Opis sadržaja
1.	DiplomskiRad.doc	Tekst diplomskog rada
2.	Aplikacija/index.html	*webGIS* aplikacija
3.	Podaci/GUP.shp	Shape datoteka sloja Generalnog urbanističkog plana Rovinjskog sela