SVEUČILIŠTE U ZAGREBU

GEODETSKI FAKULTET

Adam Vinković

WebGIS groblja Velika Trapinska

Diplomski rad

Zagreb, srpanj 2012.

Zahvala:

Prije svega se zahvaljujem svome mentoru prof. dr. sc. Damiru Medaku i voditelju Mariju Mileru na iznimnoj potpori, pristupačnosti i strpljenju prilikom izrade ovog diplomskog rada.

Također se zahvaljujem općini Suhopolje na ustupljenim podacima, te tvrtki Geosistem d.o.o. i njenim radnicima na pomoći prilikom izmjere i obrade podataka.

Posebno se zahvaljujem roditeljima i djevojci na ljubavi, razumijevanju i podršci za vrijeme studija, te svim prijateljima i kolegama uz koje je to vrijeme nažalost prebrzo završilo.

I. Autor

Ime i prezime: Adam Vinković
Datum i mjesto rođenja: 06.12.1987., Vinkovci, Hrvatska

II. Diplomski rad

Predmet: Programiranje u geoinformacijskim sustavima
Naslov: WebGIS groblja Velika Trapinska
Mentor: prof. dr. sc. Damir Medak
Voditelj: M. Miler

III. Ocjena i obrana

Datum zadavanja zadatka:
Datum obrane: 14.09.2012
Sastav povjerenstva pred kojim je obranjen rad: prof. dr. sc. Damir Medak
prof. dr. sc. Drago Špoljarić
dr. sc. Ivan Medved

WebGIS groblja Velika Trapinska

Sažetak

Ovim radom prikazuje se postupak razvijanja webGIS aplikacije na primjeru groblja Velika Trapinska. Prvi dio rada opisuje tehnologije i alate korištene prilikom izrade aplikacije. Naglasak se stavlja na upotrebu isključivo otvorenih tehnologija rukovanja prostornim podacima i alata otvorenog koda. U drugom dijelu rada opisan je cjelokupni postupak izrade webGIS aplikacije. Upotrebom prostorne baze podataka PostGIS omogućeno je skladištenje podataka. Za posluživanje kartografskog prikaza korišten je aplikacijski poslužitelj GeoServer. Izrada korisničkog sučelja i implementacija funkcionalnosti webGIS aplikacije omogućena je korištenjem OpenLayers, ExtJS i GeoExt skupova gotovih funkcija pisanih programskim jezikom JavaScript.

Ključne riječi: *webGIS, otvoreni kod, PostGIS, GeoServer, OpenLayers, ExtJS, GeoExt*

WebGIS of Velika Trapinska cemetery

Abstract

This paper shows the process of developing a WebGIS application on the example of Velika Trapinska cemetery. The first part describes the technology and tools used when creating the application. Emphasisis is placed on the use of open source tools and open technologies for handling geospatial data. The second section describes the complete process of the WebGIS application development.The spatial database PostGIS is used to store data. Map content is served using the application server GeoServer. Development of the user interface and implementation of the WebGIS functionality is enabled by using OpenLayers, ExtJS and GeoExt frameworks written in JavaScript.

Keywords: *webGIS, open source, PostGIS, GeoServer, OpenLayers, ExtJS, GeoExt*

Sadržaj

1. Uvod

U protekla dva desetljeća svjedoci smo naglog napretka informacijskih tehnologija i ubrzanog razvoja informatike. Sa sigurnošću možemo reći da živimo u informacijskom dobu u kojem je osnovna pokretačka snaga *Internet*, globalni sustav povezanih računalnih mreža koji omogućuje nove oblike komunikacije, poslovanja, učenja i razmjene informacija. Razvojem interneta dolazi i do razvoja velikog broja web aplikacija. Web aplikacije su računalni programi kojima se pristupa putem interneta, a pisani su programskim jezikom koji je razumljiv internet preglednicima.

S obzirom na svakodnevni rast količine podataka u svijetu, već duže vrijeme postoji potreba za prikupljanjem, skladištenjem, obradom, analizom, prikazivanjem i dijeljenjem tih podataka korištenjem informacijskih tehnologija i sustava. U slučaju prostornih podataka, odnosno podataka koji sadrže prostornu komponentu, koriste se posebni informacijski sustavi koji se nazivaju *geografski informacijski sustavi* (*GIS*). Imajući u vidu brojne prednosti koje nudi internet kao medij, sve je češći razvoj web aplikacija koje nude mogućnosti jednog GIS-a, tzv. *webGIS* aplikacija. Njihovim razvojem olakšava se prikazivanje i dijeljenje prostornih podataka bez potrebe postavljanja specijaliziranih alata. Osim toga smanjuje se redundantnost podataka s obzirom da nije potrebno višestruko preuzimanje tih podataka, već se oni nalaze na web-u (*World Wide Web*). No ključna prednost webGIS-a jest činjenica da je svakoj osobi koja ima mogućnost spajanja na internet omogućen pristup prostornim podacima.

Svrha ovog rada je uspostaviti webGIS kojim će se lokalnoj samoupravi olakšati donošenje odluka i financijska održivost u poslovanju, te omogućiti efikasnija upotreba vlastitih resursa digitalizacijom podataka groblja kojima raspolaže. Cilj je na primjeru groblja prikazati razvoj jedne webGIS aplikacije koristeći isključivo tehnologije otvorenog koda.

2. Korištene tehnologije

U ovom poglavlju opisane su informacijske tehnologije koje su upotrijebljene u izradi zadatka diplomskog rada. Prilikom odabira tehnologije naglasak je stavljen na upotrebu tehnologije otvorenog koda (eng. *open source*). Značaj te tehnologije leži u činjenici da je izvorni kod programa i servisa dostupan („otvoren“) svakome te ga je moguće mijenjati, kopirati i dijeliti bez ikakve naknade.

2.1. HyperText Markup Language

Izrada bilo koje web aplikacije, pa tako i jednog webGIS-a, ne bi bila moguća bez upotrebe *HTML*-a (*Hypertext Markup Language*). Iako *HTML* nije programski jezik, već običan znakovni (eng. *markup*), opisni i prezentacijski jezik, on zajedno s *CSS*-om (*Cascade Style Sheets*) čini jezgru razvoja koda jedne internet stranice. Prilikom razvijanja internet stranica *CSS* se koristi za opisivanje grafičkih osobina stranice kao što su boje, fontovi, format i sl., dok *HTML* služi za opisivanje strukture internet stranica korištenjem *HTML* elemenata, odnosno oznaka (eng. *tag)*. Pomoću oznaka se označava sadržaj internet stranice kao što su tablice, tekst, liste, poveznice na druge stranice, fotografije, itd. *HTML* dokument je obični tekstualni dokument (**.html* ili **.htm*) koji je moguće stvoriti i urediti u bilo kojem tekstualnom editoru. Web (skraćenica od eng. *World Wide Web*) uglavnom čine *HTML* dokumenti koji se od web poslužitelja (eng. *webserver*) do internet pretraživača (eng. *browser*) prenose koristeći *Hypertext Transfer Protocol (HTTP)*[1].

Glavna svjetska organizacija za standardizaciju *HTML*-a i *CSS*-a jest *W3C* (eng. *World Wide Web Consortium*). Osnovao ju je i trenutno ju vodi *Sir Tim Bernars-Lee*, koji je također prvi specificirao *HTML* te utemeljio informacijski sustav za razmjenu znanstvenih istraživanja i radova, preteču današnjeg interneta. Otkada je u ranim 90-tim predstavljen u svijetu interneta, *HTML* je kontinuirano razvijan. Zadnja službena verzija *HTML* standarda predstavljena je 1999. pod nazivom *HTML 4.01*. Iako je već 2004. *W3C* u suradnji s *Web Hypertext Application Technology Working Group* započeo razvoj nove verzije standarda, *HTML 5* je još uvijek u konceptualnoj fazi (URL 1). Unatoč tome što je još uvijek u razvojnoj fazi, *HTML5* je već prisutan u

[1] *Hypertext Transfer Protocol(HTTP)* –osnovna i najčešća metoda komunikacije i prijenosa informacija na web-u kojom se omogućuje objavljivanje i prezentacija *HTML* dokumenata (URL 11)

brojnim internet stranicama i web aplikacijama koje iskorištavaju njegove prednosti kao što su: novi *HTML* elementi (*video*, *audio*, *canvas*, itd.), čišći i jednostavniji kod, manja ovisnost o vanjskim dodacima (eng. *plugin*), te mogućnost razvijanja mobilnih aplikacija i stranica. Jedna od takvih internet stranica koje već koriste *HTML5* je http://www.giscloud.com/ hrvatske tvrtke *Omnisdata*.

2.2. JavaScript

U počecima web-a svaka stranica korisniku je pružena u obliku statičnog dokumenta. *JavaScript* programski jezik je omogućio uvođenje interaktivnih sadržaja u prikazu internet stranica iz razloga što se izvršava unutar internetskog pretraživača te ima direktan pristup svim elementima dokumenta.

JavaScript je objektno-orijentiran skriptni jezik koji se isključivo izvršava na korisnikovom računalu (eng. *client-side*). Najčešća upotreba programiranja *JavaScript* koda jest prilikom pisanja funkcija ugniježđenih (eng. *embedded*) unutar *HTML* stranica. Upravo zbog činjenice da se kôd izvršava na strani korisnika, a ne na nekom udaljenom poslužitelju (eng. *remote server*), moguća je brža reakcija pretraživača na korisničke radnje čime se postiže dinamičnost same web aplikacije koja nije omogućena isključivom upotrebom *HTML*-a. Danas svaki internet pretraživač ima u sebi integriran *JavaScript engine,* odnosno prevoditelj (eng. *interpreter*) koji interpretira JavaScript izvorni kôd i izvršava skriptu.

Prvi put se ovaj skriptni jezik pojavljuje 1995. u *Netscape Navigator* pretraživaču, istovremeno s integracijom tada popularne *Java* tehnologije u istoimenom pretraživaču. Netočna početna pretpostavka da *JavaScript* potječe od programskog jezika Java dovela je do zbrke o povezanosti tih dviju tehnologija. No sami naziv novog skriptnog jezika bio je tek marketinški trik kojim se nastojalo profitirati od popularnosti *Java*-e (URL 2).

Danas je JavaScript najčešće korišteni skriptni programski jezik u svijetu. Od samog početka vrlo je brzo postigao veliku popularnost među web programerima u izradi internet stranica. Upravo zato je i konkurentni *Microsoft* već 1996. u svojem Internet pregledniku *Internet Explorer 3.0* uveo podršku za JavaScript. Iako su okorjeli programeri dugo izbjegavali ovaj skriptni jezik, pojavom *AJAX*-a (*Asynchronous*

JavaScript and XML) dolazi do razvoja velikog broja skupova gotovih funkcija (eng. *frameworks*) i biblioteka (eng. *libraries*) kojima je olakšan i ubrzan razvoj internet stranica i web aplikacija.

Pod pojmom *AJAX* se podrazumijeva skup metoda integriranih unutar *JavaScripta* za pozadinski prijenos podataka između internet preglednika i poslužitelja. Na taj način je web aplikacijama omogućeno primanje i slanje podataka sa poslužitelja u pozadini (asinkrono), bez ometanja sadržaja i ponašanja same stranice. Odličan primjer implementacije *AJAX* tehnologije su *Google Maps* karte kod kojih se svaka nova sekcija karte skida sa poslužitelja onda kada je korisniku potrebna, bez da se stranica ponovno učitava.

2.3. PHP

PHP je u početku bila skraćenica za *Personal Home Page Tools* ili *Professional Home Page*, dok danas označava pojam *Hypertext Preprocessor.* Početno je *PHP* kreiran kao skup skripti u programskom jeziku *Perl*, a napisao ga je *Rasmus Lerdorf* kako bi održavao vlastitu internet stranicu. Kasnije je preradio taj skup skripti u *CGI* (*Comon Gateway Interface*) binarne datoteke u programskom jeziku *C* kako bi omogućio komunikaciju s bazama podataka, manipulaciju formi te ugradnju u *HTML* prilikom stvaranja dinamičnih web aplikacija. Dva izraelska programera, *Zeev Suraski* i *Andi Gutmans*, 1997. godine prerađuju raščlanjivač (eng. *Parser* [2]) i stvaraju *PHP 3*. Kasnije slijede daljnje izmijene, a zadnja službena inačica programa je *PHP 5.4.3* (URL 3).

Iako je *PHP* prvobitno bio namijenjen za stvaranje dinamičnih stranica s interaktivnim sadržajem, danas se koristi prvenstveno za izvršavanje skripti na poslužiteljskoj strani (eng. *server-side*) za razliku od *JavaScript*-a koji se izvršava na strani klijenta. *PHP* može biti ugniježđen u *HTML* kôd, postavljen na web poslužitelju ili izvršen u komandnoj liniji. Iako ima podužu listu nedostataka, kao što su npr. slaba podrška objektno-orijentiranom programiranju, veliki broj globalnih funkcija i varijabli zbog nepostojanja prostora naziva (eng. *namespace*) ili slabe performanse u odnosu na konkurentne jezike, činjenica da nudi podršku za sve operativne sustave, najčešće

[2] Parser - dio programa koji analizira primljeni sadržaj i čini ga razumljivim programu

korištene web poslužitelje i većinu baza podataka čini ga jednim od najpopularnijih skriptnih jezika koji se izvršavaju na poslužitelju.

2.4. JavaScript Object Notation

JSON (*JavaScript Object Notation*) je tekstualni format za razmjenu strukturiranih podataka. Uglavnom služi za razmjenu podataka putem mrežnih veza (eng. *network connection*) između poslužitelja i web aplikacija. *JSON* format čovjeku je lako čitljiv, a računalu jednostavan za prepoznavanje i raščlanjivanje. Osim toga je potpuno je neovisan o drugim programskim jezicima.

Po strukturi *JSON* može biti u obliku kolekcije parova "*naziv:vrijednost*", što se u mnogim programskim jezicima tumači kao objekt ili zapis (eng. *record*), te u obliku popisa vrijednosti, što se u većini programski jezika tumači kao red (eng. *array*). Osnovni tipovi *JSON* podatka su: broj, niz znakova (eng. *string*), niz, boolean, objekt i prazan podatak (eng. *null*). S obzirom na ulogu *JSON* formata u razmjeni podataka često se uspoređuje s *Extensible Markup Language* (*XML*). U pravilu se koristi kao alternativa *XML*-u. Glavna prednost *JSON* formata jest njegova jednostavnost zbog čega je stekao veliku popularnost u *AJAX* tehnologiji prilikom komunikacije poslužitelja i klijenta. Sljedeća dva isječka koda prikazuju isti podatak zapisan u *JSON* formatu i *XML*-u (URL 5):

JSON:

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

XML:

```
<menu id="file" value="File">
```

```
<popup>
<menuitem value="New" onclick="CreateNewDoc()" />
<menuitem value="Open" onclick="OpenDoc()" />
<menuitem value="Close" onclick="CloseDoc()" />
</popup>
</menu>
```

2.5. OGC standardi

Open Geospatial Consortium (*OGC*) je glavna međunarodna i industrijska organizacija za standardizaciju u domeni prostornih podataka. *OGC* je utemeljen 1994. pod nazivom *Open GIS Consortium,* a danas okuplja više od 400 komercijalnih tvrtki, vladinih i neprofitnih organizacija te istraživačkih i akademskih ustanova. Članovi konsenzusom sudjeluju u razvijanju i implementaciji otvorenih standarda kojima se nastoji omogućiti interoperabilnost i integracija prostornih podataka i servisa na web-u. Na taj način stvaraju se okviri za razvoj geoinformacijskih sustava, a korisnicima se omogućuje jednostavniji pristup i analiza velikog broja prostornih informacija, koje dolaze s raznovrsnih izvora iz područja otvorenih informacijskih tehnologija.

OGC web servisi (eng. *OGC Web Services*, skraćeno *OWS*) spadaju u skupinu najznačajnijih *OGC* standarda. Moglo bi se reći da je upravo njihovim razvojem standardizacija na području geoinformacija dobila na velikom značaju. Osnovna načela arhitekture *OGC* web servisa su (Shekhar & Xiong, 2008):

- Sastavni dijelovi servisa organizirani su na više razina (eng. *tier*)
- Suradnjom više servisa stvara se rezultat prilagođen korisniku
- Komunikacija servisa koristi otvorene internet standarde:
  - *World Wide Web* (*WWW*) protokoli (*HTTP, GET, HTTP POST* i *SOAP*) za komunikaciju između servisa
  - *Uniform Resource Locator* (*URL)* za adresiranje operacija između poslužitelja
  - *Multipurpose Internet Mail Extensions* (*MIME*) tipovi za prepoznavanje formata prijenosa podataka
  - *Extensible Markup Language* (*XML*) za kodiranje prenesenih podataka
- Sučelja servisa koriste otvorene standarde i relativno su jednostavna

U sljedeća dva potpoglavlja opisana su dva *OGC* web servisa koja su korištena u izradi zadatka diplomskog rada.

2.5.1. Web Map Service (WMS)

Web Map Service (*WMS*) jest *OGC* standard i istovremeno najpopularniji i najrašireniji *OGC* web servis kojim se specificira komunikacija između nepovezanih programa prilikom slanja ili pružanja unaprijed generiranih kartografskih prikaza (eng. *map imagery*) korisnicima koji su zatražili zahtjev (eng. *request*). Jednostavno rečeno, *WMS* je servis kojim klijent može poslati upit za primanjem karte s nekog poslužitelja. *WMS* je prvi standard koji je *OGC* donio na području web servisa (2000.). Trenutno aktualna verzija je *WMS 1.3.0*.

Ključna komunikacija kod *WMS*-a odvija se putem *XML*-a. Poslužitelj prima upit od *WMS* klijenta, raščlanjuje upiti kodira ga koristeći *XML*. Vrste upita koje su specificirane *WMS* standardom:

- **GetCapabilities** – pruža popis svih podataka, parametara i slojeva zatraženog WMS-a (obavezan – svaki WMS mora sadržavati tu vrstu upita)
- **GetMap** – pruža zatraženi kartografski prikaz (obavezan)
- **DescribeLayer** – pruža detaljne informacije o sloju zatraženog skupa podataka (neobavezan)
- **GetFeatureInfo** – pruža atributne informacije o kartografskom prikazu (neobavezan)
- **GetLegendGraphic** – pruža legendu kartografskog prikaza (neobavezan)

Prilikom slanja upita *WMS* poslužitelju potrebno je, osim adrese poslužitelja, vrste upita, naziva servisa i njegove verzije, definirati i dodatne parametre kao što su npr. sloj prikaza (eng. *layer*), koordinatni sustav prikaza (eng. *spatial reference system*, skraćeno *SRS*), granični okvir prikaza (eng. *bounding box*), visinu i širinu prikaza (eng. *height* i *width*), stil prikaza (eng. *styles*) te format. Primjer jednog *WMS* upita korištenog u izradi zadatka diplomskog rada:

```
http://localhost:8080/geoserver/groblje/wms?service=WMS&version=1.1.0&request=GetMa
p&layers=groblje:DOF&styles=&bbox=6459500.0,5064000.0,6461750.0,5067000.0&width=384
&height=512&srs=EPSG:900901&format=image/jpeg
```

Pod uvjetom da je *WMS* poslužitelj postavljen te da su traženi podaci smješteni na njemu, upisivanjem gore navedenog *URL*-a u adresnu traku internet pretraživača trebao bi se prikazati digitalni ortofoto s zatraženim parametrima.

Slika 1. Rezultat WMS upita

2.5.2. Web Feature Service (WFS)

Web Feature Service (*WFS*) je kao i *WMS* jedan od standarda *OGC* web servisa kojim se specificira razmjena prostornih podataka putem web-a te definiraju pravila za primanje i pružanje geografskih informacija koristeći *Hyper Text Transmission Protocol* (*HTTP*). Osim razmjene geografskih podataka, *WFS* opisuje procese manipulacije prostornim objektima, a također i podržava vektorski model podataka. Za razmjenu informacija koristi se na *XML*-u bazirani *Geographic Markup Language* (*GML*). Trenutno aktualna verzija je *WFS 1.1.0*.

Za razliku od *WMS* upita, kod kojeg klijent dobiva gotove predefinirane karte bez mogućnosti spremanja dohvaćenih podataka, prilikom *WFS* upita klijent dobiva sirove

informacije o geografskim objektima (točka, linija, poligon) u *GML* formatu. S tim informacijama klijent može izabrati kako će ih iskoristiti. Iako klijent u ovom slučaju mora obaviti više 'posla' kako bi prikazao prostorne podatke, WFS nudi klijentu puno veću fleksibilnost. To znači da je, osim dohvaćanja prostornih podataka s *WFS* poslužitelja, omogućena i manipulacija podacima – brisanje, izmjena, stvaranje, spremanje i zaključavanje (u slučaju da više korisnika pristupa podacima odjednom). Takav *WFS* naziva se transakcijskim, odnosno *transactional Web Feature Service* (*WFS-T*).

WFS standardom specificirani su sljedeći obavezni *WFS* upiti:

- **GetCapabilities** – *WFS* poslužitelj vraća *XML* dokument s popisom tipova podataka koje može pružiti i operacijama koje je nad njima moguće izvesti
- **DescribeFeatureType** – opis strukture svih tipova podataka koje *WFS* pruža
- **GetFeature** – *WFS* poslužitelj vraća *GML* podatke zatražene upitom

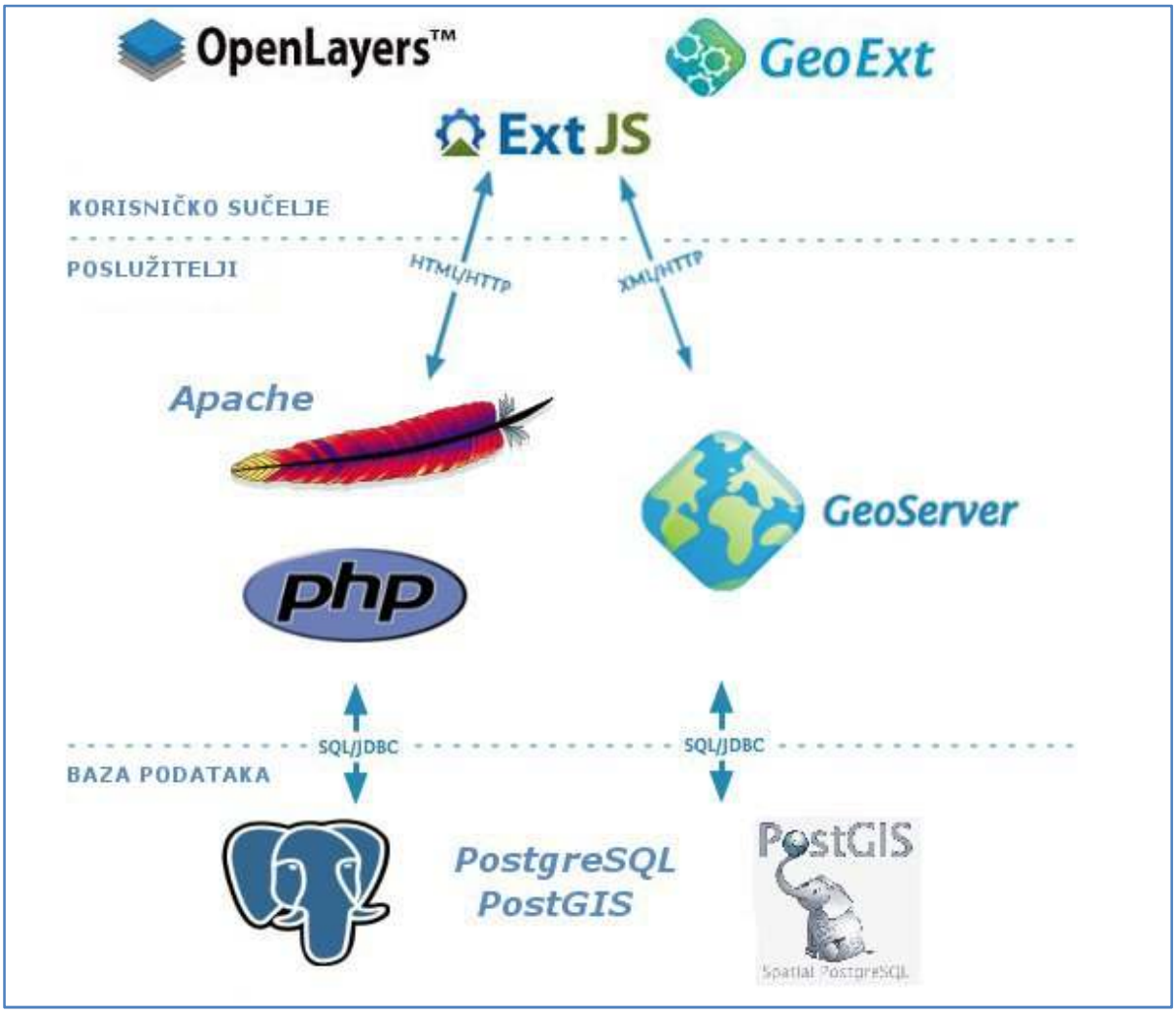
Osim obaveznih *WFS* upita postoje i neobavezni, ali ne manje važni *WFS* upiti kao što je npr. *LockFeature* (zaključavanje stanja nekog objekta kako bi se spriječilo njegova promjena) ili *Transaction* (izvršavanje operacija nad objektima - izmjena,brisanje ili stvaranje novih prostornih objekata).

Kao i kod *WMS*-s potrebno je prilikom slanja *WFS* upita navesti adresu poslužitelja, vrstu upita, naziv servisa i njegovu verziju. Moguće je navesti i dodatne parametre kao što su npr. format (eng. *output Format* - *GML2*, *GML3*, *ESRI Shape*, *JSON*, itd.), maksimalan broj objekata (eng. *maxFeatures*), sortiranje dohvaćenih objekata ovisno o njihovim atributima (eng. *sortBy*), granični okvir prikaza (eng. *bounding box*) i dr. Primjer jednog *WFS* upita korištenog u izradi zadatka diplomskog rada:

```
http://localhost:8080/geoserver/groblje/ows?service=WFS&version=1.0.0&request=GetFe
ature&typeName=groblje:grobovi&maxFeatures=50&outputFormat=json
```

3. Korišteni alati

Za potrebe izrade zadatka ovog rada korištene su razne tehnologije opisane u prijašnjem poglavlju. U ovom poglavlju opisani su alati, odnosno softveri, kojima je izrađena webGIS aplikacija groblja Velika Trapinska. S obzirom da se radi o geoinformacijskom sustavu kojem se pristupa putem interneta, arhitektura webGIS-a podijeljena je u tri razine (slika 2.) kao što je to slučaj i u većini aplikacija kojima se pristupa putem interneta.



Slika 2. Arhitektura webGIS aplikacije groblja Velika Trapinska

Prvu razinu čini baza podataka (eng. *database*). Unaprijed pripremljeni podaci pohranjuju se unutar objektno-relacijske baze podataka *PostgreSQL*. S obzirom da se u slučaju webGIS-a koriste i prostorni podaci potrebno je proširiti *PostgreSQL* bazu podataka s dodatkom *PostGIS*, koji služi kao podrška za prostorne objekte u objektno-relacijskoj bazi.

Drugu razinu webGIS-a čine poslužitelji (eng. *server*). Na lokalnom računalu uspostavljen je web poslužitelj pod nazivom *Apache HTTP Server* (u daljnjem tekstu: *Apache*) na kojem se nalazi web aplikacija, dok je za komunikaciju i povezivanje baze podataka i webGIS aplikacije korišten je skriptni jezik *PHP*. Za posluživanje prostornih podataka upotrijebljen je aplikacijski poslužitelj (eng. *application server*) *Geoserver.*

Korisničko sučelje aplikacije čini treću razinu cjelokupne arhitekture webGIS-a groblja Velika Trapinska. WebGIS aplikacija izrađena je u programskom jeziku *JavaScript* i to uz pomoć biblioteke otvorenog koda *ExtJS*. Osim toga su korištene i druge *JavaScript* biblioteke otvorenog koda, *GeoEXT* i *OpenLayers,* kojima je znatno olakšana izrada webGIS aplikacije. Uz navedene alate, korišteni su i pomoćni alati kao što je npr. tekstualni uređivač *Sublime Text 2* za pisanje izvornog koda webGIS aplikacije, internet pretraživač *Mozilla Firefox 12.0* za prikaz rezultata i testiranje izvornog koda, te alat za pomoć pri razvoju aplikacija i pronalaženju grešaka izvornog koda *Firebug*.

3.1. Apache HTTP Server

Apache je najrasprostranjeniji web poslužitelj razvijen od otvorene zajednice razvojnih programera pod okriljem *Apache Software Foundation.* Otvorenog je koda te je dostupan za cijeli niz operativnih sustava, uključujući *Unix*, *Linux*, *MacOS X*, *Microsoft Windows* i dr. *Apache* je igrao značajnu ulogu u razvoju World Wide Web-a od samog početka kada je 1995. predstavljen. 2009. je postao prvi web poslužitelj s više od 100 milijuna internet stranica koje poslužuje. Prema podacima iz ožujka 2012. *Apache* poslužuje oko 57% svih internet stranica, odnosno oko 400 milijuna internet stranica (URL 10).

Osnovna zadaća svakog web poslužitelja (eng. *web server*) jest na zahtjev klijenta pružiti internet stranicu (eng. *web page*) putem *HTTP* protokola. Drugim riječima, web poslužitelj klijentu dostavlja *HTML* dokument zajedno sa svim dodatnim sadržajem, kao što su slike, listovi proračunskih tablica (eng. *style sheets*) ili skripte. Klijent putem internet pretraživača inicira komunikaciju u obliku zahtjeva za određenim resursom koristeći *HTTP*, a poslužitelj odgovara sadržajem zatraženog resursa ili porukom o pogrešci u slučaju da ne može poslužiti zahtijevani sadržaj.

Iako je primarna funkcija web poslužitelja dostavljati određeni sadržaj, web poslužitelj je sposoban i za primanje podataka od strane klijenta putem web obrazaca (eng. *web form*). Većina web poslužitelja također podržava izvršavanje skripti na strani poslužitelja (eng. *server-side scripting*) pomoću skriptnih jezika *PHP*, *Active Server Pages (ASP*), *Perl*, *Python* i sl. U pravilu se ta funkcionalnost web poslužitelja koristi u slučaju stvaranja dinamičkih *HTML* dokumenata prilikom komunikacije s bazom podataka.

3.2. PostgreSQL i PostGIS

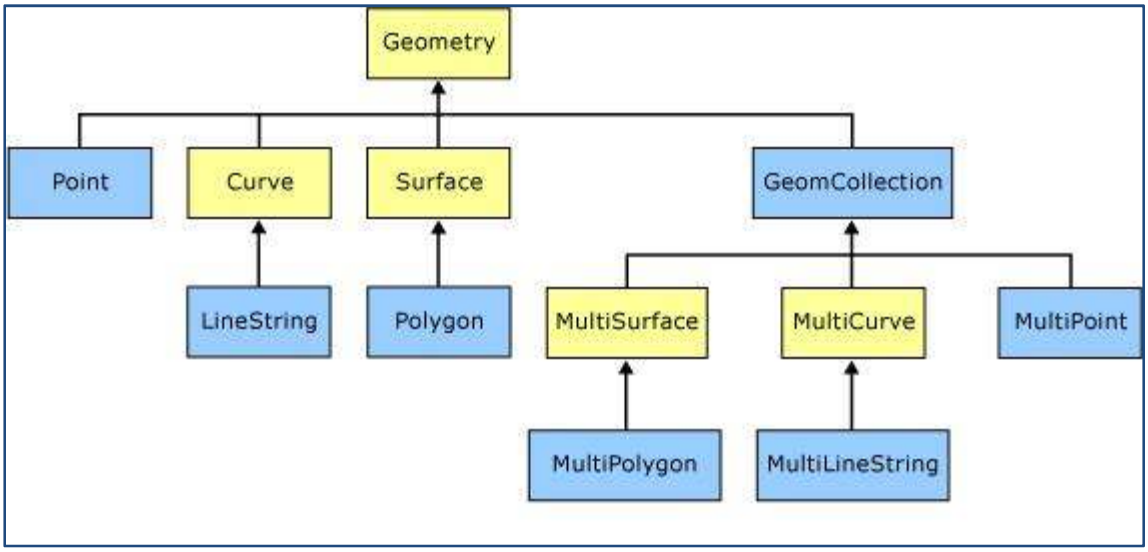
PostgreSQL (poznat i kao *Postgres*) je objektno-relacijski sustav za upravljanje bazom podataka (eng. *object-relational database management system,* skraćeno *ORDBMS)* otvorenog koda, dostupan velikom broju operativnih sustava. *PostgreSQL* je razvijen na sveučilištu Berkeley u Kaliforniji kao nastavak *Ingres* projekta s ciljem rješavanja problematike tadašnjih bazi podataka. Značajan razvoj *PostgreSQL* je doživio zahvaljujući činjenici da je izdan pod MIT licencom, čime je programerima dozvoljeno razvijanje i integracija sustava unutar drugih alata otvorenog koda. Tako 1994. Andrew Yu i Jolly Chen, apsolventi na sveučilištu Berkeley, stvaraju *Postgres95* zamijenivši tumač upitnog jezika (eng. *query language interpreter*) *QUEL*, preuzet iz Ingres projekta, s tumačem za *SQL* (*Structured Query Language*) upitni jezik. Kasnije (1996.) projekt mijenja ime u *PostgreSQL* da bi se naglasila upotreba *SQL* upitnog jezika (URL 14). Od tada se *PostgreSQL* aktivno razvija od grupe dobrovoljaca i programera sustava za upravljanje bazama podataka te je do danas izdano nekoliko novih inačica sustava.

Sustav za upravljanje bazom podataka (eng. *database management system,* skraćeno *DBMS*) je alat namijenjeni za učinkovito pohranjivanje i pristup podacima unutar baza podataka. Kako bi se podaci mogli smisleno pohraniti u bazu podataka, sustav mora imati definiranu strukturu, odnosno model podataka. Sustav za upravljanje bazom podataka može imati *hijerarhijski*, *mrežni*, *relacijski* ili *objektni* model podataka, a prema načinu na koji se pohranjuju podaci razlikujemo tri glavna tipa baza podataka: *relacijske*, *objektne* i *objektno-relacijske*. U geoinformatici je upravo posljednja skupina najpopularnija zahvaljujući činjenici da preuzima prednosti relacijskih baza podataka (primjena relacijske algebre, standardiziran upitni jezik,

jednostavna pravila za spremanje podataka bez redundancije, indeksiranje, transakcije), a da pri tome usvaja napredne osobine objektno-orijentiranog modela podataka podržavajući objekte i operacije nad objektima, definiranje tipova podataka, klase i nasljeđivanje (eng. *inheritance*).

PostGIS je dodatak *PostgreSQL* bazi podataka za pohranjivanje, upravljanje i analizu prostornih objekata. Usuglašen je s *OGC* standardom "*Simple Features for SQL*" kojim se specificira opći model pohranjivanja geografskih podataka koristeći *Well-known text* (*WKT*) i *Well-known binary* (*WKB*) formate zapisa. Moglo bi se reći da *PostGIS* nadograđuje *PostgreSQL* bazu podataka geometrijskim tipovima podataka i prostornim funkcijama. Podržani geometrijski tipovi (slika 3.) podataka su:

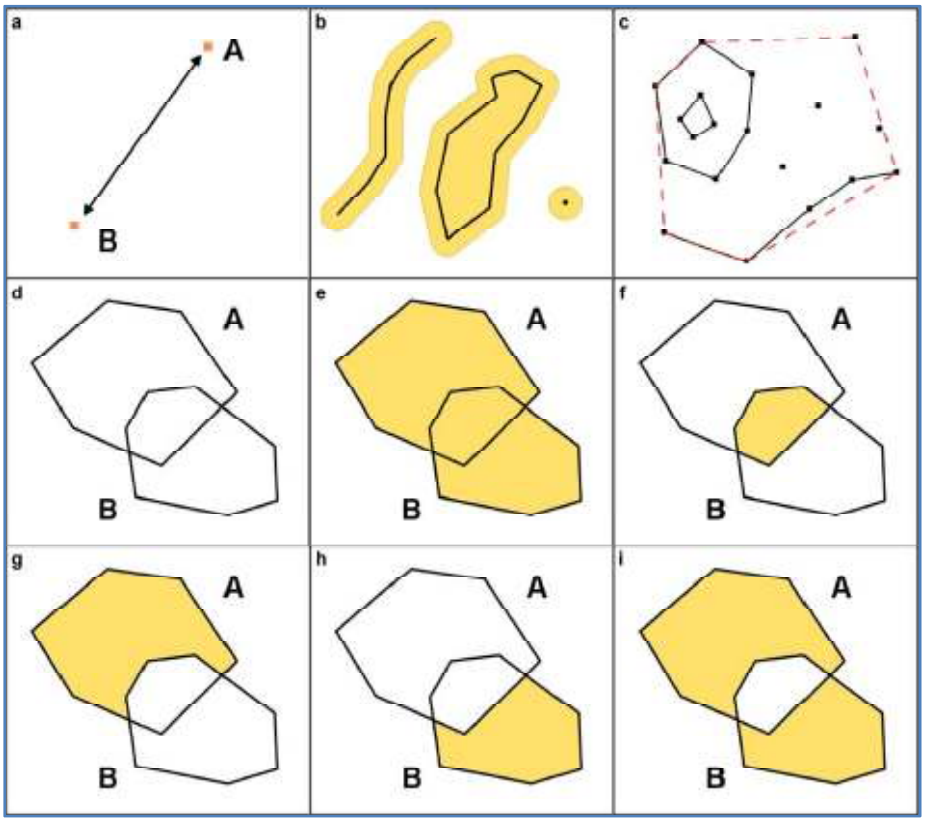
- **Points** – točke
- **LineStrings** – linije
- **Polygons** – poligoni
- **MultiPoints** – skup točaka
- **MultiLineStrings** – skup linija
- **MultiPolygons** – skup poligona
- **GeometryCollections** – kolekcija geometrije



Slika 3. Geometrijski tipovi podatak prema "Simple Features for SQL" Specifikaciji (plavi su oni koje PostGIS podržava)

Prostorne funkcije (slika 4.) omogućuju obradu i analizu geografskih objekata, a primjer su funkcije za izmjeru kao što su površina (eng. *area*), udaljenost (eng.

distance), duljina (eng. *length*) i opseg (eng. *perimeter*), te prostorni operatori kao što su unija (eng. *union*), razlika (eng. *difference*), simetrična razlika (eng. *symmetric difference*) i tampon (eng. *buffer*). Osim toga *PostGIS* podržava *SQL* funkcije za testiranje prostornih (topoloških) odnosa - jednakost (eng. *equals*), presijecanje (eng. *intersects*), križanje (eng. *crosses*), obuhvaćanje (eng. *within*), graničnost (eng. *touches*), sadržajnost (eng. *contains*), preklapanje (eng. *overlaps*) i približnost (eng. *relate*).



Slika 4. Prostorne funkcije – a) udaljenost, b) tampon, c) koveksna ovojnica, e) unija,f) presjek, g) razlika A - B, h) razlika B - A, i) simetrična razlika, d) poligoni korišteni za prostornu operaciju e – i

3.3. GeoServer

GeoServer je besplatan alat otvorenog koda napravljen u programskom jeziku *Java* kojim se korisnicima omogućuje posluživanje prostornih podataka. Njegova osnovna uloga jest služiti kao povezna točka u otvorenoj infrastrukturi prostornih podataka (eng. *Spatial Data Infrastructure*, skraćeno *SDI*). Moglo bi se reći da *GeoServer* ima istu ulogu kao i *Apache* u području web-a, samo za prostorne podatke. Koristeći otvorene standarde *OGC*-a, *GeoServer* omogućuje značajne funkcionalnosti u prikazu i dijeljenju geografskih objekata i karata te interoperabilnost s velikim brojem komercijalnih i besplatnih geoinformacijskih sustava i kartografskih aplikacija.

Zbog svojih značajki *GeoServer* je, uz *UMN MapServer*, najpopularnija i najrašireniji aplikacija za prikazivanje interaktivnih karata putem web-a. Osim što je kompatibilan s najvažnijim specifikacijama *OGC*-a kao što su *WMS*, *WFS* (i *WFS-T*) i *WCS* (*Web Coverage Service*), omogućuje "on the fly" reprojekciju rastera i vektora, sadrži veliku bazu *EPSG*[3] koordinatnih sustava i projekcija te omogućuje unošenje vlastitih.

Učitavanje i obrada prostornih podataka omogućena je putem web sučelja, dok se za prikazivanje učitanih podataka koristi integrirani skup gotovih kartografskih funkcija *OpenLayers*. *GeoServer* podržava velik broj formata podataka – *PostGIS*, *ESRIShapefile*, *ArcSDE*, *DB2*, *Oracle Spatial*, *MySQL*, *MapInfo*, *GeoTIFF*, *GTOPO30*, *ArcGrid*, *World Images*, *ImageMosiacs*, *JPEG2000*, *Erdas Imagine* i dr., te nudi mogućnost ispisa karata u raznim formatima – *JPEG*, *GIF*, *PNG*, *PDF*, *SVG*, *KML.* Osim navedenih formata moguće je sirove vektorske podatke pomoću *WFS*-a ispisati u obliku *GML*-a, *GeoJSON*-a ili kompresiranog *Shapefile*-a.

Jedna od ključnih tehnologija na kojima je baziran *GeoServer* je *GeoWebCache* tehnologija kojom se omogućuje iznimno brzo kartiranje unaprijed spremljenih dijelova karte u međuspremnik (eng. *cache*). Također jedna od bitnih karakteristika je i podrška za *SLD*[4]. Upravo zbog svih navedenih karakteristika *GeoServer* je izabran kao poslužiteljska aplikacija za posluživanje prostornih podataka u izradi zadatka ovog rada.

[3] *EPSG* (*European Petroleum Survey Group*) – organizacija geodetskih stručnjaka u naftnoj industriji zaslužna za stvaranje baze podataka s parametrima projekcijskih i geografskih koordinatnih sustava te elipsoida korištenih u geodeziji

[4] *SLD* (*Styled Layer Descriptor*) - otvoreni standard za definiranje stila prikaza karte

3.4. OpenLayers

OpenLayers je besplatna *JavaScript* biblioteka otvorenog koda namijenjena za prikaz karata u internet preglednicima. Razvijen je kao projekt *Open Source Geospatial Foundation* (*OSGeo*)[5] sa svrhom da posluži kao osnovno web sučelje za prikaz kartografskog sadržaja u izgradnji nekomercijalnih geoinformacijskih web aplikacija. *OpenLayers* ima sličnu funkciju kao i *Google Maps* ili *Bing Maps* sučelja u izradi aplikacijskih sučelja (eng. *Application Programming Interface*, skraćeno *API*), no za razliku od njih je otvorenog koda što omogućuje njegovo razvijanje i korištenje od strane čitave *Open Source*[6] zajednice.

S obzirom da je *OpenLayers* skup gotovih funkcija u potpunosti pisan *JavaScript* programskim jezikom, on se izvršava isključivo na korisnikovom računalu, odnosno u internet pregledniku klijenta. Na taj način se omogućuje izrada webGIS aplikacija koje nisu ovisne o resursima poslužitelja.

Pristup podacima odvija se putem otvorenih standarda čime se omogućuje preklapanje kartografskih slojeva iz više izvora unutar jedne webGIS aplikacije. Time je omogućena ne samo interoperabilnost s velikim brojem rasterskih i vektorskih formata, već i neovisnost od komercijalnih proizvoda i njihovih zatvorenih formata zapisa prostornih podataka. *OpenLayers* nudi mogućnost prikazivanja rastera iz različitih izvora kao što su *WMS, TMS (Tile Map Service), WMS-C (WMS Tile Caching), Google Maps, Bing Maps, Yahoo Maps, OpenStreetMap, ArcGIS Server, ArcIMS* i dr. Podržani su *KML (Keyhole Markup Language), GML, GeoJSON, GeoRSS i WFS* za prikazivanje i stiliziranje vektora, te *WFS-T* za web-baziranu manipulaciju nad vektorima.

Jedna od bitnih značajki jest mogućnost umetanja *OpenLayers* skupa gotovih funkcija u bilo koju *JavaScript* biblioteku gotovih funkcija kao što su *jQuery*, *Dojo*, *MooTools* ili *Ext.* Zahvaljujući tome moguća je implementacija karata u razna grafička sučelja internet stranica što *OpenLayers* čini vrlo popularnim u izradi webGIS aplikacija.

[5] *OSGeo* – neprofitna nevladina udruga koja podržava i promovira razvoj otvorenih tehnologija i podataka u domeni geoinformacija
[6] *Open Source* – inicijativa pokrenuta od zajednice programera i korisnika koji koriste alate otvorenog koda

3.5. ExtJS i GeoExt

ExtJS je *JavaScript* biblioteka, odnosno skup gotovih funkcija napisan u *JavaScript* programskom jeziku za izradu interaktivnih web aplikacija. Koristeći prednosti naprednih tehnologija *JavaScript* programskog jezika, poput *AJAX*-a, *DHTML*-a (*DynamicHTML*) ili *DOM* (*Document Object Model*) skriptiranja, *ExtJS* omogućuje bržu i efikasniju izradu vizualno dojmljivih, sadržajem bogatih i dinamičnih internet stranica.

Prednost *ExtJS*-a leži svakako u činjenici da je pisan *JavaScript* skriptnim jezikom čime se izvršavanje koda aplikacije direktno odvija na korisnikovom računalu. Korisnik pri tome ima slobodu odabira internet preglednika jer je *ExtJS* kompatibilan s gotovo svima internet preglednicima kao što su *Internet Explorer* (od inačice 6), *Mozilla Firefox* (od inačice 3.6), *Safari* (od inačice 4), *Google Chrome* (od inačice 10) i *Opera* (od inačice 11).

Neki web programeri izbjegavaju *ExtJS* zbog njegove veličine s obzirom da čitava biblioteka funkcija sadrži oko 1 MB (eng. *megabyte*) dok drugi skupovi gotovih funkcija pisani *JavaScript* programskim jezikom, poput *jQuery*-a, sadrže oko 50 KB (eng. *kilobyte*) (URL 20). No upravo u tome leži prednost *ExtJS*-a prilikom izrade naprednih i "bogatih" web aplikacija, koje izgledom grafičkog sučelja (eng. *Graphical User Interface*, skraćeno *GUI*), funkcionalnošću i kompleksnošću nalikuju klasičnim *desktop* aplikacijama, odnosno aplikacijama koje se direktno izvršavaju na računalu korisnika. Tako je prilikom izrade sučelja jedne web aplikacije moguće koristiti gotove kontrole (u informatičkom žargonu poznate pod nazivom "*widgets*") poput:

- Kontrola za upravljanje podacima tablica (uređivanje ili isključivo samo čitanje podataka (eng. *read-only*) tablice, sortiranje podataka, zaključavanje, povlačenje (eng. *drag*) i sl.)
- Kontrola za unos tekstualnih, numeričkih i datumskih polja
- Kontrola za stvaranje gumbova (eng. *button*), klizača (eng. *slider*), alatnih traka (eng. *toolbar*), izbornika (eng. *menu*), vektorskih grafova (eng. *vector graphics chart*) i sl.
- Kontrola za stvaranje panela stablaste organizacije podataka (eng. *Tree Panel*) i tab panela

- Kontrola za razmještanje pojedinih elemenata aplikacije, odnosno oblikovanje njenog izgleda (eng. *application layout*)

Značajno jest da većina tih kontrola posjeduje mogućnost komunikacije s udaljenim (eng. *remote*) web poslužiteljima putem *AJAX* tehnologije. Posljednja veće izdanje *ExtJS* skupa gotovih funkcija objavljeno je u travnju 2011. u inačici *ExtJS 4.0*. U izradi zadatka ovog rada korištena je inačica *ExtJS 3.4*. s obzirom da posljednje izdanje još uvijek nije kompatibilno s posljednjom inačicom *GeoExt* skupa gotovih funkcija (*GeoExt 1.1*).

GeoExt je, kao i *OpenLayers* i *ExtJS*, skup gotovih funkcija napisan u *JavaScript* programskom jeziku, a namijenjen kao pomoć pri izradi web aplikacija s kartografskim sadržajem. Temelji se na *ExtJS* biblioteci gotovih funkcija te u kombinaciji s *OpenLayers*-om pruža čitav niz prilagodljivih kontrola kojima se znatno olakšava i ubrzava prikazivanje, obrada i stiliziranje prostornih podataka, a time i izrada webGIS aplikacija. Posljednja službena inačica je *GeoExt 1.1*, a u pripremi je objavljivanje inačice *GeoExt 2.0* kojom će se omogućiti kompatibilnost s najnovijim izdanjem *ExtJS*-a.

4. Izrada webGIS aplikacije

Prije izrade sučelja webGIS aplikacije potrebno je pravilno instalirati i prilagoditi alate navedene u prethodnom poglavlju kako bi se mogao izvršiti unos podataka u njih. S obzirom na odabranu arhitekturu webGIS aplikacije, postavljanje alata korištenih prilikom izrade same aplikacije podijeljeno je u nekoliko faza. Prvo je potrebno postaviti bazu podataka *PostgreSQL* s pripadajućim dodatkom za podršku prostornim podacima *PostGIS*. Potom se unaprijed pripremljeni podaci unose u bazu podataka. Nakon toga je potrebno postaviti aplikacijski poslužitelj *GeoServer* kojim se omogućuje posluživanje prostornih podataka pohranjenih u bazi podataka, a u zadnjoj fazi izvršava se postavljanje web poslužitelja *Apache* i s njim povezanog skupa skriptnih funkcija *PHP* pomoću kojeg se omogućuje posluživanje atributnih podataka također pohranjenih u bazi podataka. Treba napomenuti kako je prilikom izrade webGIS aplikacije korišteno vlastito računalo s *Windows XP (32-bit)* operativnim sustavom.

No prije samog unosa podataka, potrebno je pripremiti izvorne podatke groblja Velika Trapinska. Izvorni geometrijski podaci korišteni u izradi webGIS aplikacije groblja Velika Trapinska dobiveni su terestričkom izmjerom pojedinih grobova. Podaci mjerenja uvezeni (eng. *import*) su u računalni alat *AutoCAD Map 5* te su potom obrađeni. Obrada mjerenih podataka uključuje smisleno povezivanje mjerenih točaka sukladno skici izmjere, brisanje dvostrukih čvorova i međusobno preklopljenih linija, te stvaranje topologije kako bi se željeni geometrijski podaci mogli izvesti (eng. *export*) u *ESRI Shape* format podataka. Tako dobiveni geometrijski podaci spremni su za unos u *PostgreSQL* bazu podataka.

Atributni podaci korišteni u izradi webGIS aplikacije groblja Velika Trapinska također su dobiveni terenskom izmjerom. Paralelno s terestričkom izmjerom grobova izvršeno je ručno popisivanje informacija zapisanih na grobovima. Na taj način bilo je moguće za većinu grobova prikupiti ime i prezime, spol te godinu rođenja i godinu smrti ukopane osobe. Za određeni dio grobova nije bilo moguće prikupiti navedene informacije s obzirom da su ti grobovi ili znatno oštećeni ili na njima ne postoji nikakav natpis. Uz prikupljene informacije o ukopanim osobama, od općine Suhopolje dobiveni su podaci o korisnicima grobova za groblje Velika Trapinska u tekstualnom formatu zapisa (**.txt*) koji su sadržavali ime i prezime, JMBG i adresu stanovanja

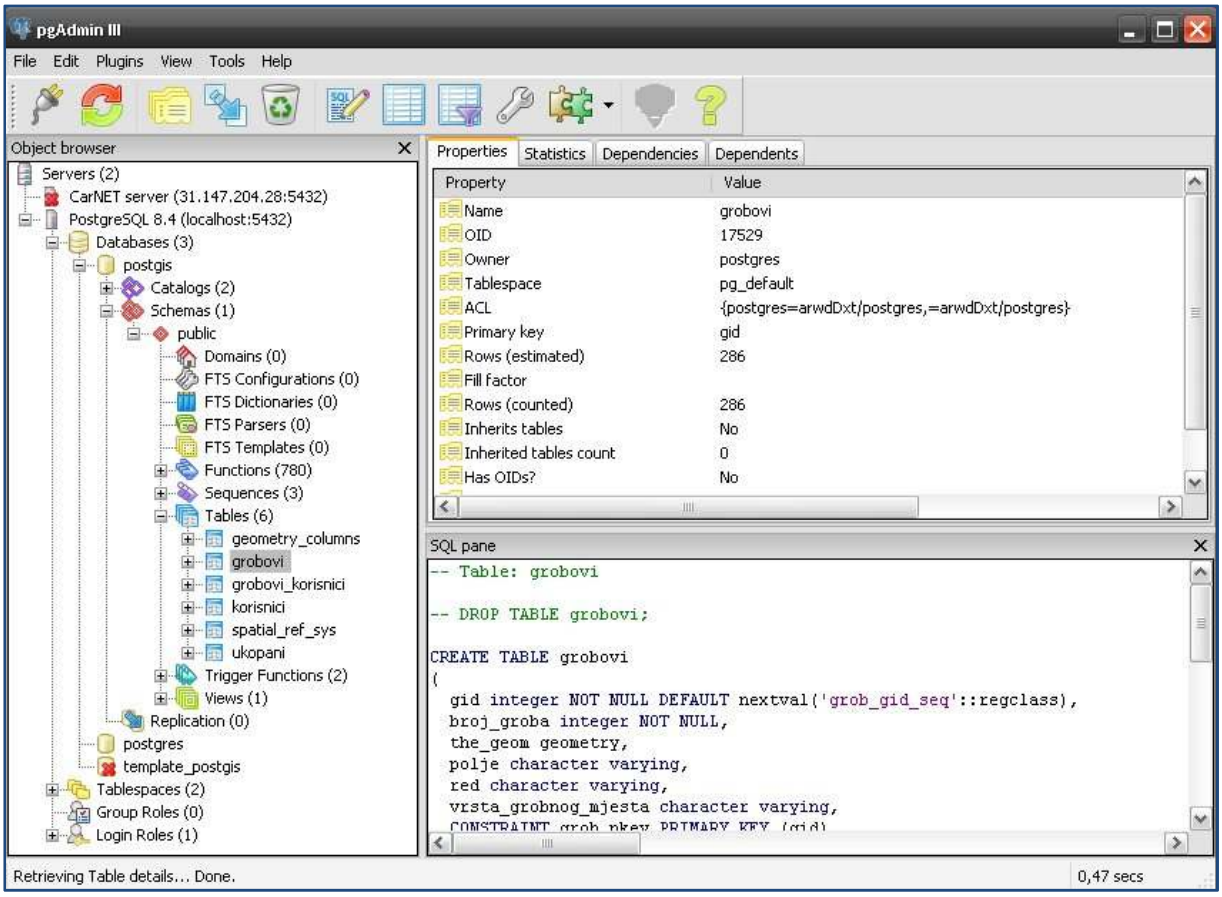
korisnika pojedinog groba. Nažalost zbog lošeg stanja u registru groblja općine Suhopolje nema podataka o korisnicima svih grobova groblja Velika Trapinska. Atributni podaci digitalizirani su i obrađeni korištenjem programskog paketa *Microsoft Office* i to uz pomoć alata za tablično obrađivanje podatak *Excel.* Tako pripremljeni atributni podaci spremni su za unos u *PostgreSQL* bazu podataka.

4.1. Postavljanje PostgreSQL baze podataka

Proces postavljanja *PostgreSQL* baze podataka sličan je i jednostavan kao i postavljanje bilo kojeg drugog softvera na računalu. Korisnik isključivo treba preuzeti skup izvršnih datoteka s službene *PostgreSQL* internet stranice koje su namijenjene za instalaciju na Windows XP operativnom sustavu, spremiti ih na korisniku poznatu lokaciju na računalu te potom pokrenuti. U izradi zadatka ovog rada izabrana je inačica *PostgreSQL* baze podataka pod brojem *8.4* s obzirom da je ta verzija poznata po stabilnosti i pouzdanosti.

Nakon pokretanja skupa izvršnih datoteka izvršava se instalacija *PostgreSQL* baze podataka, a nakon njene instalacije na ekranu računala prikazuje se pomoćni instalacijski program pod nazivom *StackBuilder* kojim se omogućuje postavljanje dodatnih programa *PostgreSQL* bazi podataka kao što su dodatni upravljački programi (eng. *drivers*), dodaci za prijavljivanje na bazu podataka, dodaci za replikaciju baze podataka, dodaci za razvijanje web aplikacija, te razne prostorne nadogradnje (eng. *spatial extensions*) među kojima se nalazi i *PostGIS*. Na taj način omogućeno je jednostavno i brzo postavljanje *PostGIS* dodatka za podršku prostornim podacima postavljenoj bazi podataka.

Gotovo najvažniji dodatni program prilikom korištenja neke baze podataka jest program za administraciju baze podataka. U slučaju *PostgreSQL* baze podataka nudi se mogućnost postavljanja programa za administraciju baze podataka direktno na korisnikovom računalu pod nazivom *pgAdmin,* te programa za administraciju baze podataka s udaljenog računala putem web sučelja (eng. *web administration interface*) pod nazivom *phpPgAdmin.* U ovom slučaju korišten je *pgAdmin* (slika 5.) s obzirom da se razvojnoj i radnoj bazi podataka pristupa s vlastitog računala.



Slika 5. Prikaz sučelja pgAdmin-a

Kako bi se pristupilo bazi podataka putem *pgAdmin*-a potrebno je uspostaviti vezu (eng. *connection*) s poslužiteljem na kojem se baza nalazi. Prilikom stvaranja nove veze potrebno je navesti naziv nove veze, adresu poslužitelja na kojem se nalazi baza podataka (eng. *host*), port (eng. *port*), te korisničko ime i lozinku. Za potrebe izrade webGIS aplikacije korištene su dvije identične baze podataka. Jedna je korištena u fazi razvoja webGIS aplikacije, a smještena je na vlastitom računalu (naziv: 'Postgre', host: 'localhost', port: '5432'). Druga se koristi kao radna baza podataka, a smještena je na *CarNET*-ovom udaljenom poslužitelju. U daljnjem postupku izrade webGIS aplikacije korištena je razvojna baza podataka.

4.1.1. Unos podataka u bazu

Prije unosa geometrijskih i prostornih podataka u bazu podataka potrebno je koncipirati model podataka kojim će se podaci smisleno bez redundancije pohraniti u bazu podataka. U slučaju izrade jedne webGIS aplikacije nekog groblja potrebno je uzeti u obzir da postoje tri kategorije podataka. Jedno su geometrijski podaci koje

čine sami *grobovi*. Ti grobovi osim svoje prostorne komponente (koordinata) sadrže i neke dodatne atribute kao što su broj groba, red, polje i vrsta grobnog mjesta. Drugu kategoriju čine *ukopane osobe* s pripadajućim atributima kao što su ID (identifikacijski broj), broj groba, redni broj ukopane osobe, ime i prezime, spol, nadnevak smrti i nadnevak rođenja. Treću kategoriju čine *korisnici grobova*, odnosno vlasnici grobnih mjesta. Uz njih se povezuju sljedeći atributi: ID, ime i prezime, JMBG (ili OIB), adresa stanovanja, osnova stjecanja grobnog mjesta, plaćanje i iznos uplate.

S obzirom na činjenicu da u jednom grobnom mjestu može biti ukopana jedna ili više osoba (odnos *1-N*; *one-to-many*) te da jedan korisnik može biti vlasnik više grobnih mjesta, a isto tako na jednom grobnom mjestu vlasnik može biti više korisnika istodobno (odnos *M-N*; *many-to-many*), potrebno je za svaku kategoriju uspostaviti vlastitu tablicu s atributima navedenim u prethodnom odlomku. Stoga su uspostavljene tablice "*grobovi*", "*korisnici*" i "*ukopani*". Osim toga potrebno je uspostaviti pomoćnu tablicu (nazvana "*grobovi_korisnici*") kojom će se omogućiti povezivanje tablica "*grobovi*" i "*korisnici*" bez nepotrebnog ponavljanja pojedinih entiteta. Na taj način je omogućeno da istovremeno jedan korisnik bude vlasnik više grobnih mjesta te da više različitih korisnika budu vlasnici jednog grobnog mjesta.

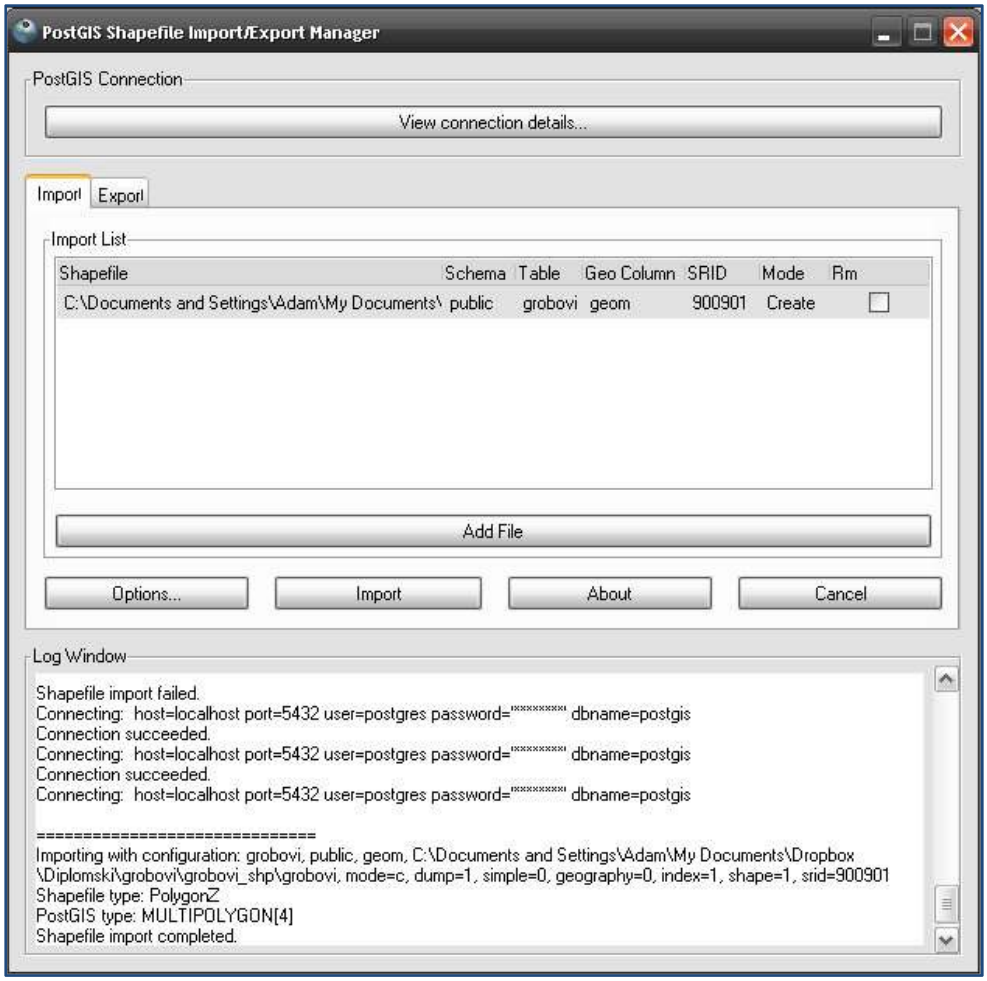
Uz navedene tablice koje su uspostavljene ručno u prostornoj bazi podataka *PostGIS* automatski se stvaraju dvije tablice "*geometry_columns*" i "*spatial_ref_sys*"*.* U tablici "*geometry_columns*" se prilikom unosa geometrijskih objekata stvara zapis o tome u kojoj tablici se ti objekti nalaze, koji je naziv stupca u tablici u kojoj se geometrijski objekti nalaze, koji je identifikacijski broj koordinatnog sustava unesene geometrije te kojeg su tipa (*point, line, polygon, multipoint, multiline, multipolygon)* geometrijski objekti*.* U tablici "*spatial_ref_sys*" nalaze se svi koordinatni sustavi poznati bazi podataka. Kako bi bilo moguće unijeti unaprijed pripremljene geometrijske podatke groblja Velika Trapinska, potrebno je definirati koordinatni sustav u kojem su podaci nalaze. U ovom slučaju radi se o *6.zoni Hrvatskog državnog koordinatnog sustava* (u nastavku teksta: *HDKS 6.zona*) pa je stoga izvršen sljedeći upit na bazu kojim je unesen željeni koordinatni sustav u tablicu "*spatial_ref_sys*"*:*

```
INSERT into spatial_ref_sys (srid, auth_name, auth_srid, proj4text, srtext) values
( 900901, 'avinkovic', 900901,

'+proj=tmerc +pm=greenwich +lat_0=0 +lon_0=18 +k=0.9999 +x_0=6500000 +y_0=0
+ellps=bessel +towgs84=514.0188,155.448,507.0461,5.6136,3.676,-11.4667,2.091
+units=m', 'PROJCS["HR6",GEOGCS["HR6",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", 18.0],PARAMETER["latitude_of_origin",
0.0],PARAMETER["scale_factor", 0.9999],PARAMETER["false_easting",
6500000.0],PARAMETER["false_northing", 0.0],UNIT["m", 1.0],AXIS["Easting",
EAST],AXIS["Northing", NORTH]]');
```

Tek sada moguć je unos unaprijed pripremljenih geometrijskih podataka, izvezenih u ESRI Shape format (**.shp*), u tablicu "*grobovi*". Taj postupak izvršava se pomoću dodatka *pgAdmin*-u pod nazivom *PostGIS Shapefile Import/Export Manager* (slika 6.). Samo je potrebno navesti parametre konekcije baze podataka, odrediti u koju tablicu se žele spremiti geometrijski objekti, dohvatiti na računalu **.shp* datoteku te kliknuti na gumb "*Import*" i unos podataka u bazu se izvršava automatski.



Slika 6. Prikaz unosa geometrijskih podataka u bazu putem PostGIS Shapefile Import/Export Manager-a

4.2. Postavljanje Geoserver-a

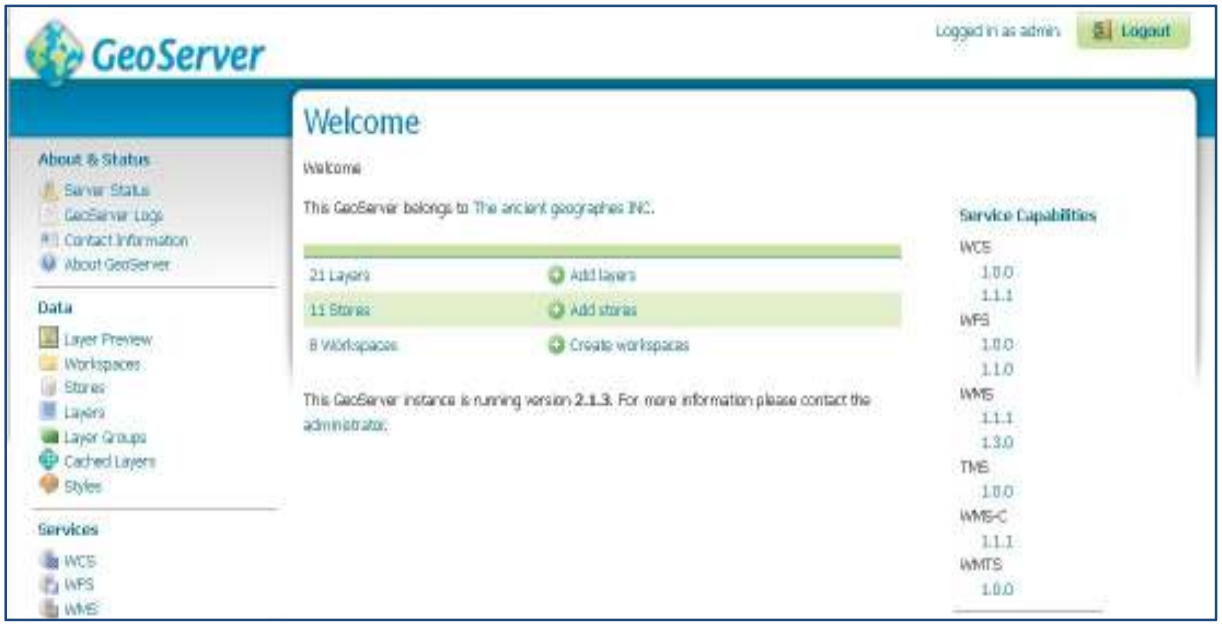
Slično kao i kod postavljanja *PostgreSQL* baze podataka, prilikom postavljanja aplikacijskog poslužitelja *GeoServer* potrebno je preuzeti univerzalni skup izvršnih datoteka. S obzirom da je *GeoServer* pisan u programskom jeziku *Java*, pokretanje skupa izvršnih datoteka neovisno je o postojećem operativnom sustavu korisnikovog računala. Nakon preuzimanja skupa izvršnih datoteka u obliku **.zip* arhive sa službene *GeoServer* internet stranice (http://geoserver.org/display/GEOS/Download) potrebno je arhivu otpakirati na korisniku poznatu lokaciju na računalu. Preporučena lokacija u slučaju *Windows* operativnog sustava je "`C:\Program Files\GeoServer`".

U slučaju da na korisnikovom računalu nije instaliran *Java JRE* (*Java Runtime Engine*), odnosno paket za izvršavanje koda pisanog u programskom jeziku *Java,* potrebno je s postaviti okruženje za pokretanje *Java* koda i dodatno ga prilagoditi za rad s *GeoServer*-om. Taj postupak izvršava se na računalu s *Windows* operativnim sustavom na sljedeći način:

- otvoriti *Control Panel >System >Advanced >Environment Variables*
- pod stavkom "System Variables" odabrati gumb New
- u polje "*Variable Name*" upisati "*JAVA_HOME*" (bez navodnika)
- u polje "*Variable Value*" unijeti putanju direktorija u kojem je instaliran Java JRE

Nakon navedenog postupka potrebno je pokrenuti aplikacijski poslužitelj na način da se u poddirektoriju GeoServer instalacije ("`C:\Program Files\GeoServer\bin`") pokrene izvršna datoteka *startup.bat.* U istom direktoriju nalazi se i izvršna datoteka *shutdown.bat* kojom se zaustavlja izvođenje aplikacijskog poslužitelja.

Zadnji korak u postavljanju *GeoServer*-a jest pokretanje web sučelja za administraciju. Taj postupak obavlja se otvaranjem internet preglednika i upisivanjem slijedeće adrese u adresnu traku: http://localhost:8080/geoserver, nakon čega se na ekranu računala pojavljuje sučelje za administraciju *GeoServer*-a (Slika 7.). Prilikom prvog pokretanja potrebno je prijaviti se koristeći korisničko ime “*admin*“ i lozinku “*geoserver*“. Zbog sigurnosnih razloga se preporuča promjena podataka za prijavu, no nije nužna.



Slika 7. Prikaz web sučelja za administraciju Geoserver-a

Bitno je naglasiti kako su, kao i kod *PostgreSQL* baze podataka, uspostavljena dva *GeoServer*-a u koje su uneseni izvorni podaci. Jedan je korišten u fazi razvoja webGIS aplikacije, a smještena je na vlastitom računalu te mu se pristupa putem internet preglednika upisivanjem internet adrese navedene u prethodnom odlomku. Drugi *GeoServer* koristi se kao radni aplikacijski poslužitelj u fazi primjene webGIS aplikacije, a smješten je na udaljenom poslužitelju koji se nalazi u CarNET-ovom računalnom centru.

4.2.1. Unos podataka u Geoserver

Prije unosa podataka u *GeoServer* potrebno je definirati parametre za vlastiti koordinatni sustav u kojem se nalaze prostorni podaci s obzirom da isti prilikom instalacije nije naveden u datoteci s popisom svih *GeoServer*-u poznatih koordinatnih sustava "*epsg.properties*". Navedena datoteka nalazi se u poddirektoriju instalacije na lokaciji "`data_dir/user_projections`". Za unos parametara HDKS 6.zone potrebno je na kraj navedene datoteke unijeti sljedeći tekst:

```
900901=
PROJCS["HR6",
GEOGCS["HR6",
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", 18.0],
   PARAMETER["latitude_of_origin", 0.0],
PARAMETER["scale_factor", 0.9999],
   PARAMETER["false_easting", 6500000.0],
PARAMETER["false_northing", 0.0],
UNIT["m", 1.0],
   AXIS["Easting", EAST],
AXIS["Northing", NORTH]]
```

Nakon unosa navedenih parametara u datoteku "*epsg.properties*" potrebno je ponovno pokrenuti *GeoServer* kako bi on mogao prepoznati promjene. Ovim postupkom omogućeno je prepoznavanje koordinatnog sustava prostornih podataka groblja Velika Trapinska prilikom unosa istih u *GeoServer*. U slučaju gore navedenih parametara vlastiti koordinatni sustav nosi oznaku *EPSG:900901.*

Postupak unosa podataka u *GeoServer* odvija se u nekoliko koraka. Prvi korak je stvaranje novog radnog mjesta (eng. *workspace*) koje služi kao skladište za spremanje prostornih podataka. U glavnom izborniku web sučelja za administraciju *GeoServer*-a na lijevoj strani ekrana odabire se u podizborniku "*Data*" poveznica "*Workspaces*". Nakon odabira navedene poveznice otvara se nova stranica na kojoj je potrebno odabrati poveznicu "*Add new workspace*", upisati naziv novog radnog mjesta (u ovom slučaju "*groblje*"), upisati *Namespace URI* (*Uniform Resource Identifier*) koji služi kao identifikacija podataka spremljenih u radnom mjestu na internetu (u ovom slučaju proizvoljno je odabrana internet adresa "http://www.geof.unizg.hr") te potom spremiti postavke klikom na gumb "*Submit*".

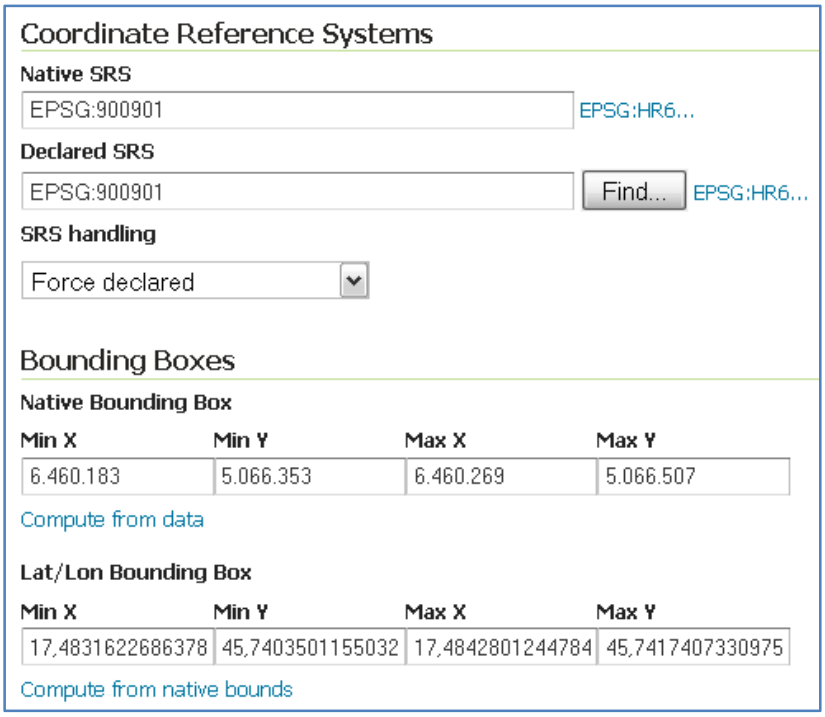
Sljedeći korak predstavlja uvoz podataka u *GeoServer*. U istom podizborniku "*Data*" potrebno je odabrati poveznicu "*Stores*" (hrv. skup obilježja), a potom u novo otvorenoj stranici izabrati poveznicu "*Add new store*" čime se otvara stranica s tipovima vektorskih i rasterskih podataka koje je moguće unijeti u *GeoServer*. U ovom koraku potrebno je odabrati tip prostornog podatka kojim se raspolaže. U slučaju webGIS-a groblja Velika Trapinska postoje dvije vrste prostornih podataka:

- poligoni (vektori) pojedinih grobova spremljeni u prostornu bazu podataka *PostGIS* (osnovni sloj kartografskog prikaza)
- fotogrametrijski snimak interesnog područja spremljen u *GeoTIFF* rasterskom formatu lokalno na računalu (pozadinski sloj kartografskog prikaza)

U prvom slučaju je za unos podataka iz baze potrebno na navedenoj stranici odabrati stavku "*PostGIS*". Potom se otvara nova stranica s postavkama gdje je nužno odabrati prethodno stvoreno radno mjesto te unijeti ime izvora podataka (npr. "*grobovi*") i parametre konekcije baze podataka kao što su adresa poslužitelja na kojem se nalazi baza, port, naziv baze podataka, naziv sheme (eng. *schema*), korisničko ime i lozinka.

Nakon što se navedeni izvor podataka spremi klikom na gumb "*Save*" otvara se, u slučaju *PostGIS* baze podataka, izbornik s popisom tablica sadržanih u upravo dodanoj bazi podataka. Potrebno je izabrati onu tablicu u kojoj se nalaze podaci koji sadrže geometriju (u ovom slučaju to je tablica "*grobovi*"). Potom se upravo dodani sloj objavljuje klikom na gumb "*Publish*" nakon čega se otvara nova stranica za definiranje postavki sloja i stila prikaza geometrije.

U postavkama sloja vrlo je važno dodijeliti mu vlastit koordinatni sustav prethodno definiran pod oznakom *EPSG:900901* (Slika 8.). Taj postupak obavlja se odabirom vlastitog koordinatnog sustavu u polju "*Declared SRS*" te odabirom stavke "*Force declared*" u polju "*SRS handling*". Osim toga potrebno je u podizborniku "*Bounding Boxes*" postaviti granice opsega (eng. *bounding box*) uvezenog sloja. *Geoserver* klikom na poveznicu "*Compute from data*" automatski izračunava koordinate rubova graničnog okvira uvezenih prostornih podataka u vlastitom koordinatnom sustavu, dok klikom na poveznicu "*Compute from native bounds*" preračunava koordinate rubova graničnog okvira iz vlastitog koordinatnog sustava u stupnjeve geografske širine i dužine (u decimalnom obliku).

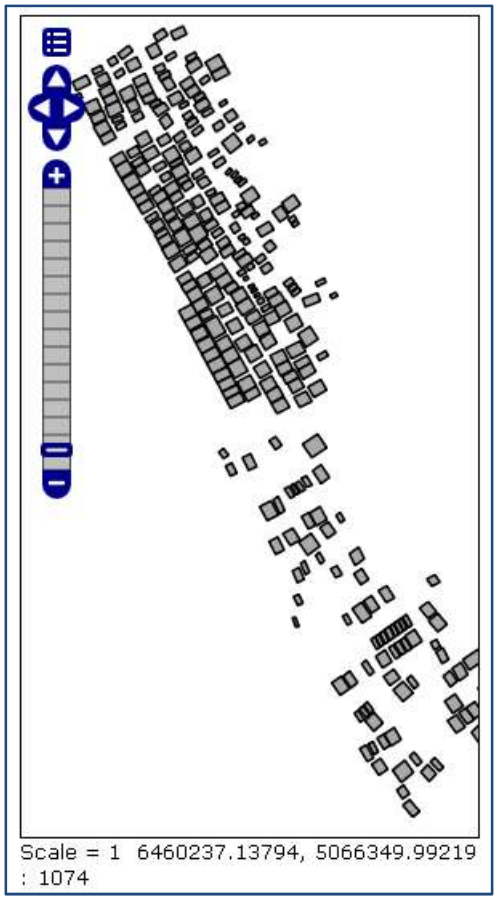


Slika 8. Prikaz dodjele vlastitog koordinatnog sustava uvezenom izvoru podataka

Prilikom unosa fotogrametrijskog snimka potrebno je, za razliku od unosa podataka iz baze podataka u *GeoServer*, odabrati stavku "*GeoTIFF*" prilikom odabira tipa rasterskog podatka. Jedina bitna razlika prilikom unosa jest ta da nije potrebno definirati parametre konekcije baze podataka, već samo unijeti apsolutnu putanju do *GeoTIFF* datoteke i pripadajuće **.tfw*[7] datoteke (*Tiff World File*) na računalu.

Kako bi se provjerila uspješnost unosa podataka u *GeoServer* potrebno je u glavnom izborniku web sučelja odabrati poveznicu "*LayerPreview*". Potom se izabere željeni sloj koji se želi prikazati te vrsta vizualizacije. Na slici broj 9. prikazana je vizualizacija uspješno uvezenog sloja "*grobovi*" korištenjem *OpenLayers* skupa gotovih funkcija.

[7] *.tfw* (*Tiff World File)* – format doteteke koja sadrži koordinate lijevog donjeg ruba okvira rastera u HDKS-u 6.zoni



Slika 9. Vizualizacija unesenog sloja "grobovi" korištenjem OpenLayers-a

4.3. Postavljanje Apache web poslužitelja i PHP-a

Prilikom postavljanja *Apache* web poslužitelja na *Windows XP* operativnom sustavu se skup izvršnih datoteka za postavljanje web poslužitelja ne preuzima sa službene internet stranice www.apache.org, već se na službenoj stranici dobiva poveznica na najbližu zrcalnu stranicu (eng. *mirror Site*) kako bi se osigurala protočnost i brzina samog preuzimanja instalacijskog paketa. U ovom slučaju adresa zrcalne internet stranice glasi:http://ftp.carnet.hr/misc/apache/httpd/binaries/win32/. Također je važno napomenuti kako je prilikom postavljanja *Apache*-ana *Windows XP* operativnom sustavu potrebno unaprijediti operativni sustav verzijom nadogradnje *Service Pack 3* ukoliko to nije prethodno učinjeno.

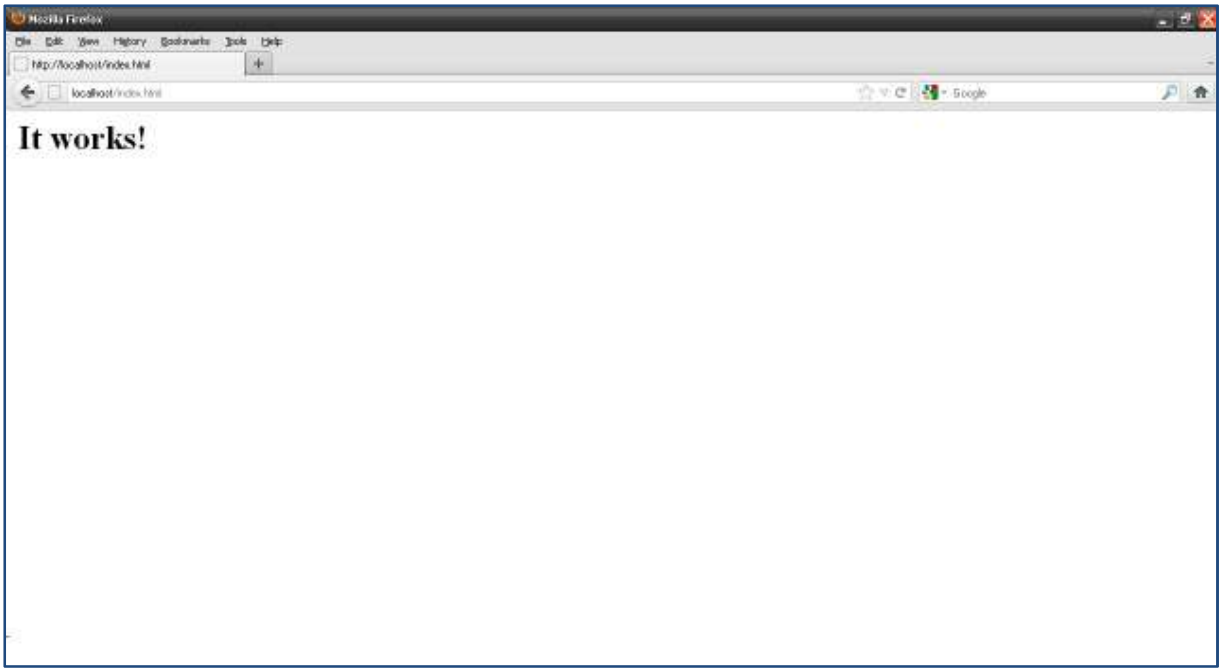
Izvršavanje instalacijskog paketa odvija se potpuno automatski. Jedino je potrebno prilikom upisivanja porta web poslužitelja pripaziti da se ne odabere isti kao i kod aplikacijskog poslužitelja *GeoServer* (8080). U ovom slučaju odabran je port pod brojem 80. Osim toga moguće je postaviti željenu lokaciju na kojoj će se nalaziti svi

dokumenti koje će Apache posluživati (eng. *document root*), no najbolje ju je postaviti na uobičajeno mjesto: `C:/Program Files/Apache Software Foundation/Apache2.2/htdocs`.

Nakon završetka instalacije *Apache* se pokreće klikom na ikonu koja se pojavi na traci sa zadacima (eng. *task bar*) ili odlaskom na lokaciju računala: *Start> Program Files > Apache HTTP server 2.2 > Control Apache Server > Start*. Kako bi se provjerilo uspješno postavljanje *Apache* web poslužitelja potrebno je u internet pregledniku učitati *HTML* datoteku pod nazivom *index.html*, koja je automatski stvorena u *Apache*-ovom direktoriju prilikom instalacije. Taj postupak moguće je obaviti upisivanjem jedne od slijedeći tri internet adrese u alatnu traku internet preglednika:

- http://127.0.0.1/index.html
- http://127.0.0.1:80/index.html
- http://localhost/index.html

Kao rezultat toga (Slika 10.) trebala bi se otvoriti internet stranica s velikim natpisom "*It Works!*" (hrv. "Radi!")



Slika 10. Provjera uspješnosti postavljanja Apache web servera

Slijedeći korak je postavljanje PHP-a. Na službenoj internet stranici http://php.net/downloads.php potrebno je odabrati sekciju "*Windows binaries*" te potom na novoj stranici odabrati poveznicu za skidanje *VC9 x86 Thread Safe* *.zip arhive najnovije stabilne verzije PHP-a. Preporuča se stvoriti novu mapu (eng. *folder*) na lokaciji `C:/php` te sadržaj prethodno skinute *.zip arhive otpakirati u tu mapu. Kako bi se PHP prilagodio potrebama projekta potrebno je u glavnom direktoriju PHP-a pronaći datoteku "*php.ini-production*" te ju preimenovati u "*php.ini*". Preimenovanu datoteku potrebno je otvoriti u nekom tekstualnom editoru, te potom pronaći i otkomentirati redak u kojem se nalazi nastavak (eng. *extension*) za pokretanje modula kojim se osigurava ispravna komunikacija *PHP*-a i *PostgreSQL*-a. Taj postupak izvršava se brisanjem oznake ";" ispred reda s slijedećim tekstom: `extension=php_pgsql.dll`.

Posljednji i najvažniji korak u postupku postavljanja *Apache*-a u kombinaciji s skriptnim jezikom *PHP* jest prilagoditi web poslužitelja kako bi mogao učitati *PHP* kao modul te ispravno raščlanjivati i izvršavati primljene *PHP* skripte. Također je nužno prilagoditi web poslužitelja za rad s *PostgreSQL* bazom podataka. To se postiže otvaranjem datoteke "*httpd.conf*", koja se nalazi u poddirektoriju `C:/Program Files/Apache Software Foundation/Apache2.2/htdocs/conf`, te upisivanjem slijedećeg tekstualnog sadržaja na samom kraju spomenute datoteke:

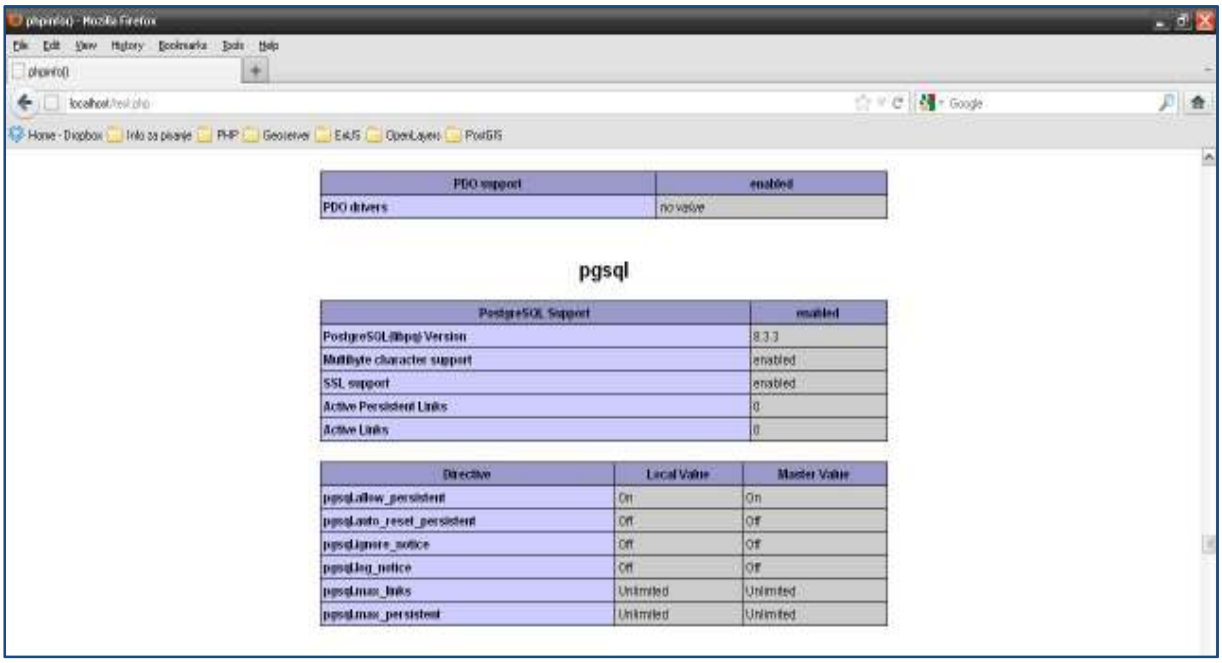
```
LoadFile "C:/Program Files/PostgreSQL/8.4/bin/libpq.dll"

LoadModule php5_module "C:/php/php5apache2_2.dll"
AddHandler application/x-httpd-php .php
PHPIniDir 'C:\php\php.ini'
```

Kako bi se zaokružio cjelokupni postupak nužno je na kraju provjeriti ispravnost učitavanja i izvršavanja *PHP* skripte na web poslužitelju te uspješnost učitavanja modula za komunikaciju *PHP*-a i *PostgreSQL*-a. Potrebno je u tekstualnom editoru otvoriti novi dokument pod nazivom "*test.php*", smjestiti ga u prethodno definirani *Apache* poddirektorij za dokumente (`C:/Program Files/Apache Software Foundation/Apache2.2/htdocs`) te upisati slijedeći *PHP* kod (URL 27):

```
<?php
phpinfo();
?>
```

Nakon toga potrebno je spremljeni dokument učitati u internet preglednik upisivanjem internet adrese http://localhost/test.php. Na ekranu računala trebala bi se pojaviti stranica s popisom modula koji su podržani od strane *PHP*-a, a nužno je da se učita onaj kojim se osigurava podrška za rad s *PostgreSQL* bazom podataka (Slika 11.).



Slika 11. Učitavanje PHP skripte test.php u Apache web poslužitelju

4.4. Pisanje koda webGIS aplikacije

Nakon što su postavljeni i prilagođeni alati kojima će se izraditi webGIS aplikacija groblja Velika Trapinska te nakon što je izvršena priprema i unos prostornih i atributnih podataka, stvoreni su svi preduvjeti za izradu webGIS aplikacije. U sljedećim potpoglavljima opisan je proces izrade webGIS aplikacije, koja ujedno predstavlja i krajnji produkt ovog diplomskog rada. Sami proces izrade programskog koda podijeljen je u nekoliko koraka. Prvi korak jest postavljanje temeljnog kostura internet stranice izradom *HTML* dokumenta. U drugom koraku izrađeno je grafičko sučelje internet stranice. Treći korak je implementacija kartografskog prikaza geometrijskih podataka, a odmah potom i tabličnog prikaza pripadajućih atributa geometrijskih podataka. Na taj način se uspostavlja kartografski prikaz svih grobnih mjesta na odgovarajućoj karti te tablični popis grobnih mjesta i njihovih atributa. Četvrti korak predstavlja izrada *PHP* skripte kojom se dohvaćaju atributni podaci o ukopanim osobama i korisnicima iz baze podataka. U petom koraku definira se

podatkovni prikaz unutar kojeg će se prikazivati atributi o ukopanim osobama i korisnicima grobova, a u zadnjem koraku povezan je kartografski prikaz i tablica s atributima geometrijskog prikaza s atributnim podacima, odnosno uspostavljena je veza između pojedinog groba i pripadajuće ukopane osobe te korisnika tog grobnog mjesta.

4.4.1. Početak izrade koda internet stranice

Kostur svake internet stranice, pa tako i webGIS aplikacije groblja Velika Trapinska, izrađuje se u znakovnom (opisnom) jeziku *HTML*. Struktura *HTML* jezika vrlo je jednostavna i lako se uči, a pisanje koda moguće je u bilo kojem tekstualnom editoru. Osnovni elementi svake internet stranice su zaglavlje (eng. *head*) i tijelo (eng. *body*). Korištenjem oznaka (eng. *tags*) definira se sekcija koda o kojoj se radi, a cjelokupni dokument omeđuje se oznakama `<html>` i `</html>`, kojima se definira početak i kraj dokumenta na sljedeći način:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title id='title'>WebGIS</title>
</head>
<body>
</body>
</html>
```

U prethodnom isječku koda prikazan je primjer dodavanja zaglavlja i tijela stranice u novi *HTML* dokument nazvan *index.html*. Uz to je u zaglavlje dodan naziv internet stranice "*WebGIS*" unutar oznaka `<title>` i `</title>`, te podrška za dijakritičke znakove hrvatskog jezika (*UTF-8*) unutar jednostrukih oznaka. Razlika između jednostrukih i dvostrukih oznaka je ta da se prilikom korištenja jednostrukih ne navodi nikakav tekstualni sadržaj, već se samo navode određena pravila.

Sljedeći korak je dodavanje elemenata potrebnih za ispravan rad i prikazivanje internet stranice u internet pregledniku. To uključuje *CSS* datoteke za uređivanje stila prikaza internet stranice te vanjske datoteke s kodom gotovih funkcija koje se koriste prilikom izrade aplikacije. Slijedeće linije koda dodane su u zaglavlje stranice direktno nakon koda koji određuje naziv stranice:

```
<!-- ** CSS ** -->
<link rel="stylesheet" type="text/css" href="ExtJS/resources/css/ext-all.css"/>
<link rel="stylesheet" type="text/css" href="GeoExt/resources/css/geoext-all.css"/>
<link rel="stylesheet" type="text/css" href="ExtJS/examples/layout-browser/layout-
browser.css"/>
<link rel="stylesheet" type="text/css" href="data_view.css"/>

<!-- ** OpenLayers ** -->
<script type="text/javascript" src="OpenLayers/OpenLayers.js"></script>

<!-- ** ExtJS ** -->
<script type="text/javascript" src="ExtJS/adapter/ext/ext-base.js"></script>
<script type="text/javascript" src="ExtJS/ext-all-debug.js"></script>

<!-- ** GeoExt ** -->
<script type="text/javascript" src="GeoExt/script/GeoExt.js"></script>
```

Prvo su uključene vanjske datoteke sa stilovima prikaza pojedinih elemenata pisanih kodom gotovih funkcija *ExtJS* i *GeoExt*, a zatim su dodane još dvije vanjske datoteke kojima se definira stil pojedinih dijelova web aplikacije. Prvu datoteku *layout-browser.css* moguće je pronaći u *ExtJS* poddirektoriju s različitim *Layout*[8] primjerima, a u izradi webGIS-a Velika Trapinska koristi se za definiranje izgleda slova i pozadine naslovne trake. Druga datoteka, pod nazivom *data_view.css*, preuzeta je iz(Garcia, 2011), a pomoću nje se definira stil i ponašanje podatkovnog prikaza (eng. *Data View*) atributnih podataka o ukopanim osobama i korisnicima grobova.

Nakon uključivanja *CSS* datoteka slijedi uključivanje *OpenLayers* skupa gotovih funkcija, a zatim i *ExtJS* i *GeoExt* skupova gotovih funkcija. Jezgru *ExtJS* skupa gotovih funkcija čini *ext-base.js* datoteka, a osim nje preporuča se prilikom razvijanja web aplikacije koristiti i razvojnu verziju *ExtJS*-a pod nazivom *ext-all-debug.js*. Vrlo bitno jest unutar elemenata oznake `src=″ ″` navesti točnu putanju do lokacije pojedinih datoteka na računalu.

Kako bi se provjerila uspješnost prikazivanja prethodno stvorenog *HTML* dokumenta u internet pregledniku potrebno je u zaglavlje stranice još dodatno postaviti slijedeći isječak *JavaScript* koda odmah ispod prethodno dodanih datoteka:

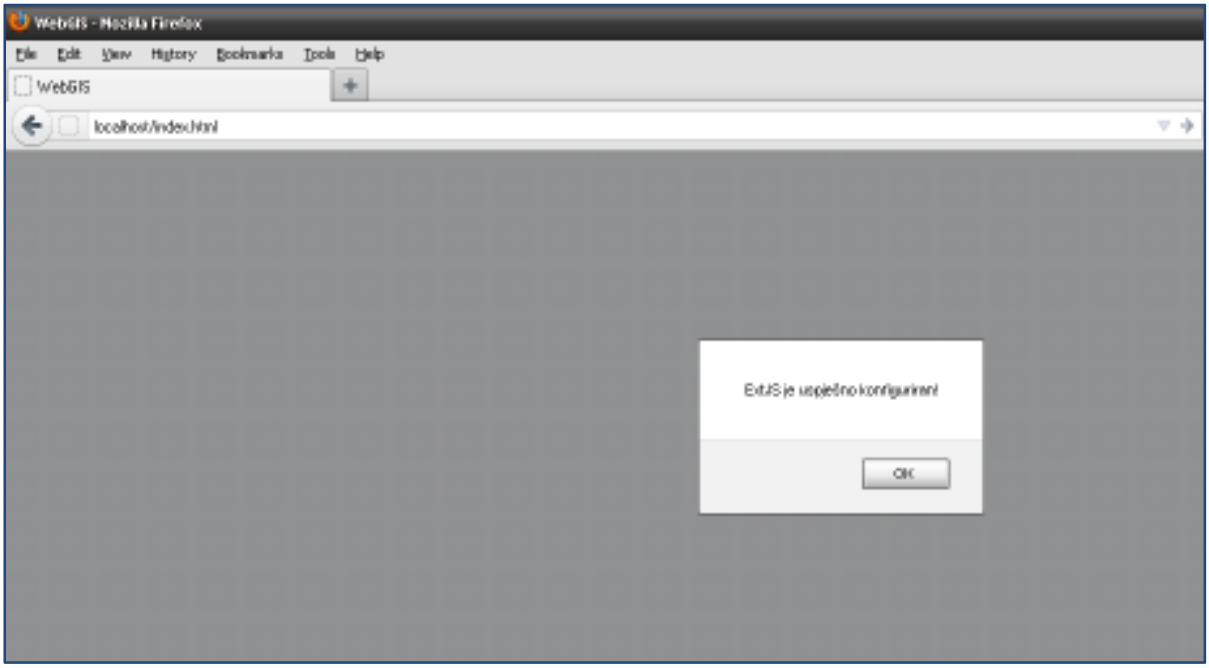
```
<script type="text/javascript">
```

[8] *Layout* (hrv. predložak, izgled, razmještaj) – način izgleda nekog spremnika (eng. *Container*) kojim se određuje vizualna organizacija, odnosno smještaj pojedinih elemenata spremnika

```
Ext.onReady(function(){
    alert("ExtJS je uspješno konfiguriran!");
});

</script>
```

Na taj način izbjegava se prikazivanje prazne internet stranice stvaranjem nove funkcije kojom se definira tekstualna obavijest "*ExtJS je uspješno konfiguriran!*" (Slika 12.). Važno je napomenuti kako se cjelokupni kod pisan korištenjem *ExtJS* skupa gotovih funkcija izvršava pozivanjem funkcije `Ext.onReady` kojom se postiže optimalno pokretanje izvornog koda prije samog učitavanja slika, a neposredno nakon učitavanja svih objekata internet stranice (Garcia, 2011).



Slika 12. Prikaz novog HTML dokumenta u internet pregledniku

U prethodno dodanom isječku koda vidljivo je kako se čitav *JavaScript* kod učitava unutar zaglavlja *HTML* dokumenta korištenjem oznake `<script type="text/javascript">` za početak pisanja i oznake `</script>` za kraj pisanja koda. To znači da je dio *HTML* dokumenta u kojem se nalazi tijelo stranice prazan. No, s obzirom na činjenicu da pojedine web aplikacije u praksi mogu dosegnuti veliku kompleksnost i dužinu *JavaScript* koda, u stručnoj literaturi ne preporuča se pisanje koda unutar *HTML* dokumenta, već se preporuča razlaganje koda aplikacije u segmente koji čine pojedine dijelove internet stranice. Tako su i u

slučaju izrade webGIS aplikacije Velika Trapinska pojedini dijelovi razloženi u samostalne *JavaScript* datoteke. Te zasebne datoteke su nakon njihovih izrada dodane u zaglavlje *HTML* dokumenta kako bi se mogle učitati u internet pregledniku.

4.4.2. Izrada sučelja webGIS aplikacije

U ovom dijelu pisanja koda započinje stvarno iskorištavanje mnogobrojnih mogućnosti koje nudi skup gotovih funkcija *ExtJS*. Jedna od njih jest i definiranje izgleda sučelja aplikacija te veličine i razmještaja pojedinih dijelova aplikacije unutar preglednika pomoću unaprijed priređenih predložaka (eng. *layout*). Za svaku klasu predložaka određen je skup svojstava (eng. *properties*) koje je moguće definirati za neki element sučelja. Hijerarhijski se klase predložaka dijele u dvije grupe: *predlošci komponenti* (eng. *component layout*) i *predlošci spremnika* (eng. *Container layout*). Prvi se koristi za kontrolu veličine i pozicije određenih komponenti sučelja, kao što su polja obrazaca (eng. *form field*) ili gumbovi, ali nema utjecaja na niti jednu komponentu kćer (eng. *child component*) unutar te komponente. Predlošci spremnika upravljaju s komponentama koje se nalaze unutar nekog spremnika, a koji ima za cilj držati na okupu određene dijelove sučelja (URL 28). *ExtJS* skup gotovih funkcija nudi slijedeće gotove predloške spremnika:

- **Absolute** – pozicionira komponente unutar spremnika koristeći relativne x i y koordinate
- **Anchor**– prilagođava veličinu komponenti unutar spremnika u odnosu na veličinu spremnika prilikom promjene njegove veličine
- **Auto** – uobičajeni predložak ukoliko nije drugi definiran
- **Border** – nudi mogućnost postavljanja višestrukih komponenti, automatsku traku za raščlanjivanje (eng. *splitter bar*) pojedinih komponenti te automatsko otkrivanje (eng. *expanding*) i zatvaranje (eng. *collapsing*) komponenti
- **Card** – upravlja višestrukim komponentama unutar jednog spremnika, ali da pri tome prikazuje samo jednu komponentu istovremeno
- **Column** – služi za stvaranje spremnika unutar kojeg se postavljaju komponente kao stupci s različitom širinom i dužinom ovisno o njihovom sadržaju

- **Fit** – upravlja samo jednom komponentom unutar spremnika koja popunjava cjelokupni prostor unutar spremnika
- **HBoxLayout** – organizira komponente spremnika horizontalno (s lijeva na desno)
- **VBoxLayout** - organizira komponente spremnika vertikalno (od gore prema dolje)
- **Table** – omogućuje učitavanje nekog sadržaja unutar tablice te nudi mogućnost stvaranja kompleksnih tabličnih izgleda

Za izradu korisničkih sučelja (eng. *User Interface*, skraćeno *UI*) web aplikacija koje se rastežu preko čitavog prozora internet preglednika, kao što je to i webGIS aplikacija groblja Velika Trapinska, najčešće se koristi *Border* (hrv. granični) predložak. Proces izrade sučelja započinje definiranjem glavnog okvira aplikacije (eng. *viewport*) koji će sadržavati sve ostale okvire (eng. *panel*) unutar kojih su smještene pojedine komponente webGIS aplikacije. U slučaju *Border* predloška moguće je smjestiti maksimalno pet regija (*north, east, west, south i center*) unutar glavnog okvira, a nužno je stvoriti minimalno jednu središnju regiju za ispravno prikazivanje sučelja aplikacije u prozoru preglednika.

Prikaz kartografskog sadržaja svakako je najvažniji dio jedne webGIS aplikacije stoga je on smješten unutar središnje regije sučelja. Zapadni dio sučelja odabran je za tablični prikaz atributa, odnosno za popis atributa pojedinih grobova. Istočni dio aplikacije podijeljen je vertikalno na dva dijela koja će služiti za prikaz atributa o ukopanim osobama i korisnicima pojedinih grobova. Kako bi se ta regija mogla vertikalno podijeliti primijenjen je *VBoxLayout* predložak. Sjeverni dio sučelja izabran je za prikaz naslovne trake.

Otvaranjem nove datoteke *app.js* u tekstualnom editoru i upisivanjem sljedećeg koda postiže se prethodno definirani razmještaj pojedinih okvira aplikacije:

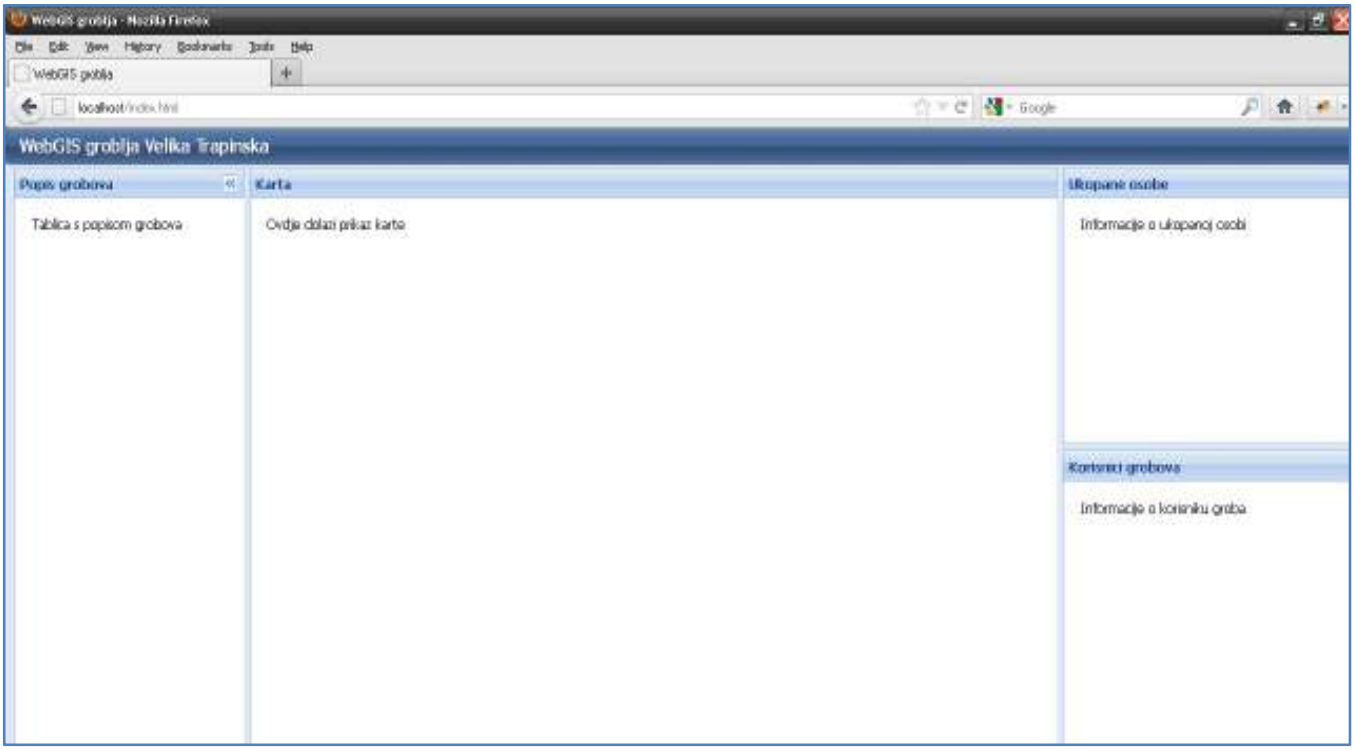
```
var viewport = new Ext.Viewport({
    layout: 'border',
    items: [{
        xtype: 'box',
        id: 'header',
        region: 'north',
        html: '<h1> WebGIS groblja Velika Trapinska</h1>',
```

```
        height: 30,
    },
{
        region: 'east',
        layout:'vbox',
        border: false,
        split: true,
        width: 250,
        margins: '2 5 5 0',
        layoutConfig : {
            align : 'stretch'
            },
        items: [{
            title: 'Ukopane osobe',
            layout: 'fit',
            flex: 1,
            html: '<p>Informacije o ukopanoj osobi</p>'
            },
            {
            title: 'Korisnici grobova',
            layout: 'fit',
            flex: 1,
            html: '<p>Informacije o korisniku groba</p>'
            }]
    },
{
        region: 'center',
        layout: 'fit',
        title: 'Karta',
        border: false,
        margins: '2 0 5 0',
        html: '<p>Ovdje dolazi prikaz karte</p>'
    },
{
        region: 'west',
        layout: 'fit',
        border: false,
        collapsible: true,
        split: true,
        title: 'Popis grobova',
        width: 200,
        margins: '2 0 5 5',
        cmargins: '2 5 5 5',
```

```
        html: '<p>Tablica s popisom grobova</p>'
    }]
});
```

Vidljivo je kako su veličine pojedinih regija, osim centralne regije koja se rasteže do rubova ostalih regija (parametar *fit*), određene upotrebom parametara *height* i *width*. Jedino je prilikom definiranja visine okvira unutar istočne regije primijenjen parametar *flex* kojim se definira relativni odnos pojedinih regija unutar *VBoxLayout* predloška (u ovom slučaju 1:1). Kako bi se pojedine regije vizualno odvojile jedna od druge definirane su margine pomoću parametra *margins*. Također je postavljena i traka za raščlanjivanje istočne i zapadne regije (parametar *split*), te mogućnost sakrivanja zapadne regije (parametar *collapsible*) kako bi se mogla proširiti središnja regija s kartografskim prikazom. Osim toga unesen je u pojedine regije opisni tekst dok se one ne popune odgovarajućim sadržajem.

Nakon spremanja prethodno pisanog koda unutar nove datoteke *app.js* potrebno je istu smjestit u poddirektorij web poslužitelja te ju ispravno navesti na dnu zaglavlja *HTML* dokumenta *index.html*. Pokretanjem *HTML* dokumenta u prozoru internet preglednika trebalo bi se učitati sučelje webGIS aplikacije (Slika 13.).



Slika 13. Prikaz sučelja webGIS aplikacije

4.4.3. Implementacija kartografskog prikaza

Nakon izrade praznog korisničkog sučelja webGIS aplikacije slijedi korak u kojem je potrebno pojedine regije sučelja popuniti sadržajem. *ExtJS* skup gotovih funkcija za to koristi okvire unutar kojih se postavlja određeni sadržaj. Za izradu okvira koji se popunjava kartografskim prikazom (eng. *map panel*) koristi se *GeoExt* skup gotovih funkcija. No prije samog stvaranja i popunjavanja okvira s kartografskim sadržajem, potrebno je definirati novi *OpenLayers* objekt karte, zadati mu odgovarajuće parametre te definirati slojeve (eng. *layer*) koji će biti prikazani na karti.

U tekstualnom editoru otvoren je novi dokument *map.js* u kojem će se biti smješten *JavaScript* kod vezan isključivo za kartografski prikaz webGIS aplikacije. Za početak je definiran novi *OpenLayers* objekt karte slijedećim isječkom koda:

```
var google_mercator = new OpenLayers.Projection("EPSG:900913");
var wgs84 = new OpenLayers.Projection("EPSG:4326");

var map = new OpenLayers.Map({
    projection: google_mercator,
    displayProjection: wgs84,
    units: "m",
    numZoomLevels: 8,
    maxResolution: 1.1,
    maxExtent: new OpenLayers.Bounds(
        1945458, 5738254,
        1947094, 5739699
    ),
    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(),
        new OpenLayers.Control.LayerSwitcher(),
    ]
});
```

Na samom početku definirani su, korištenjem *EPSG* koda, projekcije koje će se koristiti za prikaz slojeva karte. S obzirom da će kartografski prikaz sadržavati više pozadinskih slojeva, odnosno podloga (eng. *base layer*), kao najprikladnija projekcija

za prikaz prostornih podataka nameće se *Google Mercator* projekcija (*EPSG:900913*) koja je se primjenjuje za prikaz satelitskih snimaka i karata na *Google Maps* kartografskom servisu. Uz nju definirana je *WGS84* projekcija (*EPSG:4326*) koja će poslužiti za prikaz koordinata pokazivača na karti u stupnjevima geografske širine i dužine. Bitno je naglasiti kako se, prilikom slanja zahtjeva za prostornim podacima na aplikacijski poslužitelj *GeoServer*, transformacija iz vlastitog koordinatnog sustava (*EPSG:900901*), u kojem se nalaze uneseni prostorni podaci, u željeni koordinatni sustav odvija potpuno automatski.

Stvaranje novog kartografskog objekta "*map*" izvršava se pomoću klase "*OpenLayers.Map*". Prva dva parametra (*projection* i *displayProjection*) definiraju u prethodnom odlomku spomenutu projekciju za prikaz sloja karte i projekciju za prikaz koordinata pokazivača karte. Potom se pomoću parametra *units* definiraju jedinice za prikaz linije mjerila (u metrima) karte, pomoću parametra *numZoomLevels* se definira broj razina promjene mjerila karte te se pomoću parametra *maxExtent* definiraju koordinate rubnih točaka (lijeve donje i desne gornje) kartografskog prikaza u prozoru aplikacije (par X,Y koordinata izražen u *Google Mercator* projekciji). Na samom kraju se korištenjem parametra *controls* novom kartografskom objektu dodaju upravljačke kontrole kao što su navigacija po kartografskom prikazu (eng. *Navigation*), prikaz skale promjene mjerila karte (eng. *PanZoomBar*), prikaz trenutnih koordinata pokazivača miša (eng. *MousePosition*), prikaz mjerila karte (eng. *Scale*) i linije mjerila karte (eng. *ScaleLine*), te prikaz prekidača za promjenu trenutnog sloja karte (eng. *LayerSwitcher*).

Nakon što je definiran novi kartografski objekt *map* potrebno je definirati slojeve koji će se koristiti u njegovom prikazu. Prvo su u nastavku datoteke *map.js* dodani pozadinski slojevi koji služe kao podloga kartografskom prikazu na sljedeći način:

```
var openstreetmap = new OpenLayers.Layer.OSM();

var google_satellite = new OpenLayers.Layer.Google(
    "Google Satellite",
    {
     type: google.maps.MapTypeId.SATELLITE,
     numZoomLevels: 22,
     visibility: false
    }
```

```
);

var wmsLayer = new OpenLayers.Layer.WMS(
    "DOF groblja",
    "http://31.147.204.28:8080/geoserver/adam_diplomski/wms",
    {
        layers: 'adam_diplomski:6E29-13-DOF',
        isBaseLayer: true,
    }
);
```

Dohvaćanje prva dva sloja automatski je uređeno *OpenLayers* skupom gotovih funkcija korištenjem klase "*OpenLayers.Layer*" te navođenjem izvora podataka o kojem se radi. Za *OpenStreetMap*[9] kartu nije potrebno navesti nikakve parametre, dok je za sloj "*Google Satellite*" potrebno navesti određene parametre kao što su tip karte (eng. *type*), broj razina promjene mjerila karte te vidljivost (eng. *visibility*). Posljednji pozadinski sloj jest digitalni ortofoto područja na kojem se nalazi groblje Velika Trapinska koji se nalazi na *GeoServer*-u. Taj sloj, odnosno raster, dohvaća se *WMS*-om, a definira se korištenjem klase "*OpenLayers.Layer.WMS*". Osim što je novom sloju nužno zadati naziv ("*DOF groblja*") te navesti cjelokupnu internet adresu poslužitelja kojom se pristupa željenom *WMS* web servisu, potrebno je unutar parametra *layer* definirati imenski prostor ("adam_diplomski") korišten za skladištenje tog sloja te naziv sloja ("6E29-13-DOF") zabilježen u *GeoServeru.* Na kraju je pomoću parametra *isBaseLayer* određeno da upravo taj sloj bude prikazan kao početni pozadinski sloj prilikom učitavanja kartografsko prikaza u prozoru webGIS aplikacije.

Slijedeći korak u izradi kartografskog prikaza je dohvaćanje glavnog sloja karte, a to je vektorski sloj koji prikazuje grobove. Njegovo dohvaćanje sa *GeoServer*-a obavlja se putem *WFS*-a, no tu postoje određene prepreke. *WFS* za dohvaćanje podataka koristi *XMLHttpRequest*[10], no zbog sigurnosnih restrikcija u *JavaScript*-u nije moguće primanje informacija od udaljenih poslužitelja putem *XMLHttpRequest*-a(URL 24). Iz

[9] *OpenStreetMap (OSM)*- je projekt virtualne zajednice s ciljem stvaranja slobodne, svima dostupne karte koju svatko može sam i dorađivati (URL 26)

[10] *XMLHttpRequest (XHR)* – vrsta aplikacije koja se koristi u skriptnim jezicima kao što je JavaScript za slanje HTTP ili HTTPS zahtjeva na neki web poslužitelj i učitavanje primljenih podataka direktno nazad u skriptu (URL 25)

tog razloga nužno je postavljanje proxy[11] računala (eng. *proxy host*), odnosno postavljanje posredničkog računala između vlastitog web poslužitelja i udaljenog poslužitelja s kojeg se primaju podaci. *OpenLayers* u ovom slučaju nudi najbolje i najbrže rješenje u obliku gotove skripte koja 'glumi' proxy računalo. Navedenu skriptu pod nazivom *proxy.cgi* potrebno je skinuti sa sljedeće internet adrese: http://trac.osgeo.org/openlayers/browser/trunk/openlayers/examples/proxy.cgi te potom postaviti na lokaciju `C:\Program Files\Apache Software Foundation\Apache2.2\cgi-bin`. Nakon što je skripta postavljena na odgovarajuću lokaciju web poslužitelja potrebno je još dodati adresu udaljenog poslužitelja unutar varijable "*allowedHosts*" te potom na početak *JavaScript* dokumenta *map.js* unijeti sljedeću liniju koda:

```
OpenLayers.ProxyHost = "/cgi-bin/proxy.cgi?url=";
```

Tek sada je moguće stvaranje novog vektorskog sloja dohvaćanjem podataka s *GeoServer*-a korištenjem *WFS*-a. Sljedeće linije koda prikazuju taj postupak:

```
var wfsLayer = new OpenLayers.Layer.Vector("Grobovi", {
    strategies: [new OpenLayers.Strategy.BBOX()],
    reportError: true,
    visibility: true,
    protocol: new OpenLayers.Protocol.WFS({
        version: "1.1.0",
        url:  "http://31.147.204.28:8080/geoserver/adam_diplomski/wfs",
        featureType: "grobovi",
        featurePrefix: "adam_diplomski",
        featureNS: "http://www.geof.hr",
        geometryName: "the_geom",
        srsName: "EPSG:3857",
    }),
});

map.addLayers([wmsLayer, wfsLayer, openstreetmap, google_satellite]);
```

Za početak je definiran novi vektorski sloj "*Grobovi*" korištenjem klase "*OpenLayers.Layer.Vector*". Pomoću parametra *strategies* definira se način na koji se učitavaju podaci s poslužitelja. U ovom slučaju koristi se klasa "*OpenLayers.Strategy.BBOX()*" kojom se novi objekti učitavaju samo prilikom

[11] *Proxy* poslužitelj – računalo koje služi kao posrednik u komunikaciji između klijenta i glavnog poslužitelja

pomicanja kartografskog prikaza u glavnom okviru karte. Potom se definiraju parametri poput vidljivosti i izvještavanja o greškama (eng. *ReportError*). Na kraju definiranja vektorskog sloja se pomoću parametra *protocol* određuju parametri *WFS* upita za dohvaćanje prostornih podataka s *GeoServer*-a poput verzije, adrese poslužitelja, imenskog prostora korištenog za skladištenje sloja, naziva sloja, projekcije u kojoj se sloja dohvaća i sl. Zadnjom linijom koda je upravo stvoreni vektorski sloj pod nazivom "*wfsLayer*" dodan na kartografski prikaz zajedno s prethodno definiranim pozadinskim slojevima.

U sljedećem koraku definirana je i pridjeljenja kartografskom prikazu kontrola za selektiranje objekata vektorskog sloja. Na taj način je omogućeno označavanje i odznačavanje pojedinog groba klikom miša na kartografskom prikazu sljedećim linijama koda:

```
var selectControl = new OpenLayers.Control.SelectFeature(
    [wfsLayer],
    {
        clickout: true, toggle: false,
        multiple: false, hover: false,
    }
);

map.addControl(selectControl);
selectControl.activate();
```

Na samom kraju implementacije kartografskog prikaza u webGIS aplikaciji groblja Velika Trapinska potrebno je kreirati okvir koji će sadržavati cjelokupni kartografski prikaz:

```
var mapPanel = new GeoExt.MapPanel({
    title: 'Karta',
    map: map
});
```

Upravo stvoreni kartografski okvir postavlja se unutar središnje regije korisničkog sučelja na način da se u dokumentu *app.js* izbaci redak s parametrom *html* te umjesto njega postavi sljedeći red koda (bez navodnika): "`items: mapPanel`".

4.4.3.1. Izrada tabličnog prikaza atributa grobova

U ovom potpoglavlju prikazat će se postupak izrade tablice s popisom atributa vektorskog sloja "grobovi", koja će biti smještena u zapadnom dijelu korisničkog sučelja web aplikacije. Za početak se otvara novi dokument *grid.js* te se definira skup obilježja (eng. *Feature Store*) kojim se dohvaćaju atributi vektorskog sloja:

```
var wfsStore = new GeoExt.data.FeatureStore({
    autoload: true,
    layer: wfsLayer,
    fields: [
        {name: 'gid', type: 'int', mapping: 'gid'},
        {name: 'id', type: 'int', mapping: 'id'},
        {name: 'polje', type: 'string', mapping: 'polje'},
        {name: 'red', type: 'string', mapping: 'red'},
        {name: 'vrsta_grobnog_mjesta', type: 'string', mapping:
'vrsta_grobnog_mjesta'},
    ],
    sortInfo:{field: 'id', direction: 'ASC'}
});
```

Koristeći parametar *autoLoad* određuje se automatsko učitavanje dohvaćenih podataka. Parametrom *layer* definira se sloj, a parametrom *fields* definira se popis naziva atributa dohvaćenih obilježja i njihov tip. Na kraju se pomoću parametra *sortInfo* određuje uzlazno sortiranje popisa atributa koristeći atribut "*id*".

U slijedećem koraku potrebno je stvoriti model prikaza tabličnog popisa, odnosno definirati izgled, širinu i naziv stupaca tablice te stvoriti okvir s tabličnim prikazom atributa (*gridPanel*) i povezati ga s prethodno stvorenim skupom obilježja "*wfsStore*". To se postiže sljedećim kodom:

```
var groboviCM = new Ext.grid.ColumnModel(
    [{
      header: 'gid',
      readOnly: true,
      dataIndex: 'gid',
      width: 50,
      hidden: true
    },{
      header: 'Br groba',
      dataIndex: 'id',
      width: 50,
```

```
    },{
      header: 'Polje',
      dataIndex: 'polje',
      width: 50,
    },{
      header: 'Red',
      dataIndex: 'red',
      width: 50,
      hidden: true,
    },{
      header: "Vrsta grobnog mjesta",
      dataIndex: 'vrsta_grobnog_mjesta',
      width: 80,
    }]
);

var groboviGrid =  new Ext.grid.GridPanel({
    id: 'groboviGrid',
    store: wfsStore,
    cm: groboviCM,
    selModel: new GeoExt.grid.FeatureSelectionModel({
        autoPanMapOnSelection: true
    })
});
```

Koristeći klasu "*Ext.grid.ColumnModel*" stvara se model prikaza tablice, a ključni parametar je *dataIndex* kojim se pojedinom stupcu tablice dodjeljuje pripadajući atribut dohvaćen skupom obilježja "*wfsStore*". Klasa "*Ext.grid.GridPanel*" koristi se za definiranje okvira tabličnog popisa. Osim što je potrebno navesti skup obilježja koji se koristi za prikaz tablice (parametar *store*) i naziv modela prikaza tablice (parametar *cm*), moguće je i pomoću parametara *selModel* definirati kontrolu za selektiranje pojedinih objekata tablice. Koristeći klasu "*GeoExt.grid.FeatureSelectionModel*", omogućeno je selektiranje pojedinog groba na kartografskom prikazu klikom miša na taj grob u tabličnom prikazu i obrnuto.

Za kraj je potrebno okvir tabličnog popisa "*groboviGrid*" postaviti unutar zapadne regije korisničkog sučelja na način da se u dokumentu *app.js* izbaci redak s parametrom *html* te umjesto njega postavi sljedeći red koda (bez navodnika): "`items: groboviGrid`". Pokretanjem aplikacije u internet pregledniku dobiva se sljedeći prikaz (Slika 14.).

Slika 14. Sučelje webGIS aplikacije nakon implementacije kartografskog prikaza i izrade tabličnog prikaza atributa grobova

4.4.4. Izrada PHP skripte

U potpoglavlju 4.1.1 opisana je struktura baze podataka te razlozi zbog kojih je bilo potrebno atributne podatke o ukopanim osobama i korisnicima grobova smjestiti u zasebne tablice unutar baze podataka koje su neovisne o prostornim podacima. U ovom odlomku opisuje se postupak stvaranja *PHP* skripte kojom su atributni podaci o ukopanim osobama i korisnicima dohvaćeni iz baze podataka u obliku *JSON* zapisa.

Za početak je otvoren u tekstualnom editoru novi dokument pod nazivom *login.php* u kojem će se nalaziti informacije za prijavu na bazu podataka:

```
<?php
$conn_string = "host=localhost port=5432 user=postgres password=postgres
dbname=postgis";
?>
```

Nakon što su definirani parametri konekcije s bazom podataka započinje stvaranje glavne *PHP* skripte otvaranjem novog dokumenta pod nazivom *data.php*. Prvo je potrebno započeti pisanje *PHP* skriptnog koda korištenjem oznaka (bez navodnika)

"`<?php`", nakon toga uključiti prethodno definiranu skriptu naredbom "`require_once`", a potom spojiti se na bazu podataka.

```
<?php
require_once 'login.php';

$conn=pg_connect($conn_string);
if (!$conn) {
   die("Error in connection: " . pg_last_error());
}
```

Nakon što je uspostavljena veza s željenom bazom podataka započinje kodiranje ključnog djela *PHP* skripte *data.php* u kojem će se dohvatiti željeni podaci o ukopanim osobama i korisnicima grobova. Sljedeće linije koda potrebno je postaviti u nastavku skripte:

```
$broj_groba = '';

if ( isset($_GET['broj_groba'])){
  $broj_groba = $_GET['broj_groba'];
```

Prvo je definirana nova varijabla "`$broj_groba`" a potom je postavljen tzv. "*if*" uvjet kojim se definira da u slučaju zaprimanja zahtjeva (eng. *request*) metodom *GET*[12] pod nazivom "*broj_groba*" varijabla "`$broj_groba`" poprima njegovu vrijednost. Zadnji dio *PHP* koda najopsežniji je, a postavlja se unutar prethodno navedenog uvjeta:

```
if ( $broj_groba != "" ){
        $upit_ukopani = "SELECT * FROM ukopani WHERE broj_groba='$broj_groba'";
        $result_ukopani = pg_query($upit_ukopani);
        $nbrows_u = pg_num_rows($result_ukopani);
        if($nbrows_u>0){
                while($rec_u = pg_fetch_assoc($result_ukopani)) {
                        $arr_u[] = $rec_u;
                }
                $jsonresult_u = mb_json_encode($arr_u);
                echo
'{"ukopani":{"total_ukopani":'.$nbrows_u.',"results_ukopani":'.$jsonresult_u.'},';
        }
        else {
```

[12] *GET* – vrsta *HTTP* upita, odnosno metoda kojom se isključivo dobavljaju podaci sa zatraženog poslužitelja bez ikakvog drugog učinka (URL 11)

```
                echo '{"total":0, "results":""}';
        }

        $upit_korisnici = "SELECT id,ime,prezime,broj_groba,jmbg,
                           adresa,osnova_stjecanja,placanje,iznos_uplate
                           FROM korisnici,grobovi_korisnici
                           WHERE grobovi_korisnici.id_korisnika=korisnici.id
                           AND broj_groba='$broj_groba'";
        $result_korisnici = pg_query($upit_korisnici);
        $nbrows_k = pg_num_rows($result_korisnici);
        if($nbrows_k>0){
                while($rec_k = pg_fetch_assoc($result_korisnici)) {
                        $arr_k[] = $rec_k;
                }
                $jsonresult_k = mb_json_encode($arr_k);
                echo
'"korisnici":{"total_korisnici":'.$nbrows_k.',"results_korisnici":'.$jsonresult_k.'
}}';
        }
        else {
                echo '"korisnici":{"total_korisnici":0, "results_korisnici":""}}';
        }
```

Prvo se ponovo postavlja "*if*" uvjet kako bi se osiguralo da se kod unutar uvjeta ispunjava isključivo onda kada je uvjet zadovoljen. Uvjet je da varijabla "`$broj_groba`" poprimi neku vrijednost, a s obzirom na predzadnji isječak koda, ona će poprimiti vrijednost GET zahtjeva pod nazivom "*broj_groba*" ukoliko zahtjev bude zaprimljen. U tom slučaju izvršava se sljedeći postupak koji je gotovo identičan za podatke o ukopanim osobama i za podatke o korisnicima grobova.

Za početak se šalje *SQL* upit na bazu podataka kojim se zatražuje podatak iz tablice "*ukopani*" koji će odgovarati vrijednosti varijable "`$broj_groba`". Zatim se izvršava rezultat upita (naredba "`pg_query`") te određuje broj zaprimljenih redova, odnosno objekata zatražene tablice (naredba "`pg_num_rows`"). Potom se postavlja novi uvjet kojim se stvara željeni *JSON* objekt isključivo onda kada je broj zaprimljenih redova dobivenih rezultatom upita na bazu veći od 0. U slučaju da je uvjet ispunjen koristi se "*while*" petlja kako bi se čitav rezultat upita postavio u jedan niz (eng. *Array*). Nakon toga se taj niz kodira u *JSON* format koristeći gotovu funkciju "*mb_json_encode()*". Važno je naglasiti kako je na početku skripte potrebno tu funkciju navesti koristeći sljedeći kod: `include("mb_json_encode.php")`. Za kraj je potrebno rezultat

kodiran u *JSON* formatu ispisati koristeći naredbu "`echo`" poštujući sva pravila strukture *JSON* zapisa. U prvoj vitičastoj zagradi naveden je naziv prvog *JSON* objekta ("*ukopani*"), a potom je unutar tog objekta u vitičastoj zagradi ispisan ukupan broj primljenih objekata dobiveni upitom na bazu ("*total_ukopani*") i vrijednosti koje ti objekti poprimaju ("*results_ukopani*").

Postupak naveden u prethodnom odlomku primijenjen je također i za dohvaćanje podatka o korisnicima grobova s jednom bitnom razlikom. U slučaju podataka o korisnicima grobova potrebno je postaviti *SQL* upit na temelju kojeg će se zatražiti željeni podaci iz dvije tablice, glavne tablice "*korisnici*" i pomoćne tablice "*grobovi_korisnici*", a koji odgovaraju vrijednosti postavljene varijable "`$broj_groba`". Na samom kraju se dobiveni *JSON* objekt "*korisnici*" nadodaje, odnosno "lijepi", na prethodno ispisani *JSON* objekt "*ukopani*". Na taj način postiže se stvaranje jednog *JSON* zapisa koji istovremeno sadržava vrijednosti oba tražena objekta, odnosno podatke o ukopanim osobama i korisnicima grobova. Iz tog razloga nije potrebno slati dva GET zahtjeva na bazu podataka, već samo jedan, čime se znatno povećava efikasnost i brzina web aplikacije.

4.4.5. Izrada podatkovnog prikaza

Izrada podatkovnog prikaza (eng. *Data View*) znatno je lakša od izrade tabličnog prikaza s obzirom da je podatkovni prikaz vizualno puno siromašniji i ne posjeduje gotovo nikakvu funkcionalnost, osim šturog prikazivanja podataka. Podatkovni prikaz atributa o ukopanim osobama i korisnicima grobova smješten je u istočni dio korisničko sučelja webGIS aplikacije. Kako bi se prikaz atributa o ukopanim osobama vizualno odvojio od prikaza korisnika grobova, istočna regija podijeljena je vertikalno na pola o čemu je već rečeno u potpoglavlju 4.4.2.

Postupak izrade koda podatkovnog prikaza za ukopane osobe i podatkovnog prikaza za korisnike grobova istovjetan je pa će stoga u daljnjem djelu biti prikazana samo izrada koda za stvaranje podatkovnog prikaza o ukopanim osobama. Za početak je potrebno otvoriti novi dokument u tekstualnom editoru pod nazivom *data_view.js* i definirati skup obilježja:

```
var ukopaniDvStore = new Ext.data.Store({
        autoLoad: false,
```

```
        id: 'ukopaniDvStore',
        reader: new Ext.data.JsonReader({
            root: 'results_ukopani',
            totalProperty: 'total_ukopani'
        },
        [
        { name: 'id', mapping: 'id' },
        { name: 'broj_groba', mapping: 'broj_groba' },
        { name: 'redni_broj', mapping: 'redni_broj' },
        { name: 'ime', mapping: 'ime' },
        { name: 'prezime', mapping: 'prezime' },
        { name: 'spol', mapping: 'spol' },
        { name: 'nadnevak_rodjenja', mapping: 'nadnevak_rodjenja' },
        { name: 'nadnevak_smrti', mapping: 'nadnevak_smrti' }
        ])
});
```

Navedeni skup obilježja koristi čitač (eng. *reader*) za *JSON* zapis s obzirom da će prikazivati podatke koje će primiti od strane *PHP* skripte u tom formatu. Pomoću parametra *root* određuje se dio *JSON* objekta koji nosi informaciju o vrijednosti, a parametar *totalProperty* definira koji dio *JSON* objekta nosi informaciju o ukupnom broju primljenih vrijednosti. Potom je potrebno usuglasiti, odnosno mapirati nazive vrijednosti koje će biti primljene *JSON* objektom s nazivima pojedinih atributa u podatkovnom prikazu.

Posljednji korak ključan je za stvaranje predložaka (eng. *template*) kojim se definira izgled podatkovnog prikaza. Predložak se kreira koristeći jednostavne *HTML* oznake na sljedeći način (Garcia, 2011):

```
var ukopaniDVTpl = new Ext.XTemplate(
        '<tpl for=".">',
        '<div class="Wrap" id="ukopani_{id}">',
        '<div class="Name">{broj_groba} {prezime} {ime}</div>',
        '<div>',
        '<span class="title">Redni broj: </span>',
        '{redni_broj}',
        '</div>',
        '<div>',
        '<span class="title">Spol: </span>',
        '{spol}',
        '</div>',
```

```
        '<div>',
        '<span class="title">Nadnevak rođenja: </span>',
        '{nadnevak_rodjenja}',
        '</div>',
        '<div>',
        '<span class="title">Nadnevak smrti: </span>',
        '{nadnevak_smrti}',
        '</div>',
        '</div>',
        '</tpl>'
);
```

Za kraj je potrebno još postaviti podatkovni prikaz pomoću klase "*Ext.DataView*", odnosno definirati okvir koji će sadržavati prethodno stvoreni skup obilježja te ga prikazivati na temelju postavljenog predloška. Uz to je moguće zadati određene parametre kojima se određuje stil i boja prikaza u slučaju npr. odabira pojedinog prikaza, prelaska miša preko pojedinog prikaza i sl. Definicija stil i boja prikaza regulirani su *data_view.css* dokumentom (Garcia, 2011). Sljedeće linije koda prikazuju stvaranje podatkovnog prikaza atributnih podataka o ukopanim osobama:

```
var ukopaniDv = new Ext.DataView({
        tpl: ukopaniDVTpl,
        store: ukopaniDvStore,
        singleSelect: true,
        itemSelector: 'div.Wrap',
        selectedClass: 'Selected',
        overClass: 'Over',
        style: 'overflow:auto; background-color: #FFFFFF;'
});
```

Kako bi se podatkovni prikaz "*ukopaniDv*" ispravno prikazivao unutar istočne regije korisničkog sučelja aplikacije, potrebno je u dokumentu *app.js* izbaciti redak s parametrom *html* te umjesto njega postaviti sljedeći red koda (bez navodnika): "`items: ukopaniDv`".

4.4.6. Implementacija funkcionalnosti

U ovom potpoglavlju opisuje se postupak implementacije ključne funkcionalnosti webGIS aplikacije groblja Velika Trapinska, a to je mogućnost prikazivanja svih atributnih podataka vezanih za pojedino grobno mjesto prikazanog

groblja Velika Trapinska. To znači da je klikom miša na kartografski prikaz na neki od objekata vektorskog sloja “grobovi“ moguće dobiti ne samo informacije o tom grobu, kao što su npr. broj groba, polje ili vrsta grobnog mjesta, već je moguće dobiti i informacije o svim ukopanim osobama i korisnicima tog groba. Osim toga omogućena je ista funkcionalnost i za tablični prikaz u kojem se nalazi popis svih grobova.

Kako bi se omogućila spomenuta funkcionalnost u kartografskom prikazu potrebno je dodati sljedeće linije koda unutar dokumenta *map.js*:

```
wfsLayer.events.on ({
    "featureselected": function(e) {
        var record = e.feature.data.id;
        if (record) {
                Ext.Ajax.request({
                    url: 'data.php',
                    method: 'GET',
                    params: {broj_groba: record},
                    success: function(response, options) {
                            var json = Ext.decode(response.responseText);
                        ukopaniDvStore.loadData(json.ukopani);
                        korisniciDvStore.loadData(json.korisnici);
                         },
                    failure: function (){
                      alert("Pogreška!")
                    }
                });
        }
    }
});
```

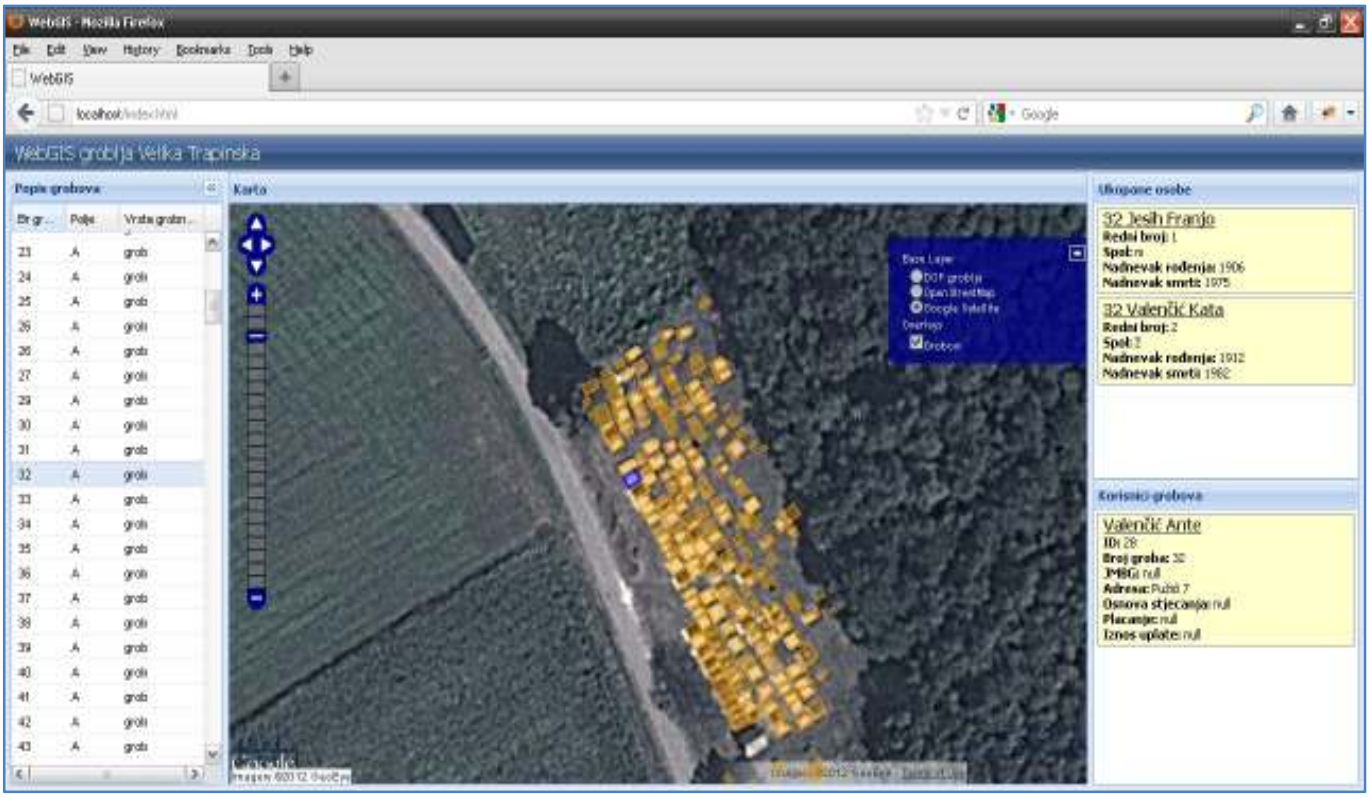
Na taj način stvoren je događaj (eng. *event*) koji se dodaje vektorskom sloju. Prvo se koristeći parametar "*featureselected*" definira nova funkcija za odabrani objekt karte. Na početku te funkcije određuje se koja je vrijednost broja groba odabranog objekta karte (u *GeoServer*-u atribut *id* sadrži vrijednost broja groba). Potom se kreira *Ajax* upit kojim se šalje zahtjev za upravo odabranim objektom na *PHP* skriptu *data.php* koristeći metodu GET. *PHP* skripta na temelju zaprimljenog parametra “*broj_groba*“ definira o kojem se grobu točno radi i za njega vraća odgovor u *JSON* zapisu. U slučaju uspješnog primanja odgovora (eng. *response*) isti se prvo dekodira, a potom i učitava unutar pojedinog skupa obilježja za ukopane osobe i korisnike grobova.

Ovime je postignuto učitavanje podatkovnog prikaza atributa o ukopanim osobama i korisnicima odabranog groba na kartografskom prikazu. Kako bi se ista ta funkcionalnost mogla primijeniti na tablični prikaz potrebno je unutar objekta "*groboviGrid*" kojim se definira okvir tabličnog prikaza dodati sljedeći isječak koda:

```
listeners: {
    rowclick: function(grid, rowIndex) {
                var record = grid.getStore().getAt(rowIndex);
                if (record) {
                  Ext.Ajax.request({
                      url: 'data.php',
                      method: 'GET',
                      params: {broj_groba: record.get('id')},
                      success: function(response, options) {
                              var json = Ext.decode(response.responseText)
                          ukopaniDvStore.loadData(json.ukopani);
                          korisniciDvStore.loadData(json.korisnici);
                           },
                      failure: function (){
                        alert("Pogreška!")
                      }
                  });
                }
              }
}
```

Ovaj isječak koda vrlo je sličan onom koji je primijenjen na kartografski prikaz. Tabličnom prikazu dodaje se parametar *listeners* (hrv. slušatelj). Jednostavnije rečeno, tablični prikaz osluškuje kada korisnik klikom miša odabere neki red u tablici (parametar *rowclick*). Potom se kao i u gornjem slučaju izvršava nova funkcija, a jedina razlika jest u načinu određivanja vrijednosti broja groba koji je odabran. U ovom slučaju određuje se broj groba na temelju indeksa reda u kojem se odabrani objekt nalazi unutar skupa obilježja.

Nakon što je cjelokupni kod aplikacije spremljen moguće je provjeriti završni izgled i funkcionalnost webGIS aplikacije pokretanjem dokumenta *index.html* u internet pregledniku. Promjenom pozadinskog sloja karte i klikom miša na neki grob na kartografskom ili tabličnom prikazu dobiva se sljedeći prikaz (Slika 15.)



Slika 15. Završni izgled webGIS aplikacije

5. Zaključak

U današnje vrijeme neophodna je primjena informacijskih sustava u poslovanju jedinica lokalne samouprave. Jedan od aspekata tog poslovanja jest vođenje digitalnog katastra pojedinih groblja na području jedinice lokalne samouprave. Zakonskom regulativom obvezuje se vođenje digitalnog registra umrlih i digitalnog grobnog očevidnika zajedno s položajnim planom grobnih mjesta. Iz tog razloga preporuča se upotreba geografskog informacijskog sustava prilikom digitalizacije prostornih i neprostornih podataka groblja.

U okviru ovog diplomskog rada izrađena je *webGIS* aplikacija na primjeru groblja Velika Trapinska kojom se predstavlja primjer jednog digitalnog katastra groblja. Prednost jednog takvog *webGIS*-a u odnosu na uobičajene geografske informacijske sustave su brojne. Već iz samog naziva moguće je primijetiti kako se *webGIS* aplikacije izvode u internet pregledniku na računalu korisnika. Njihovo korištenje neovisno je o operativnom sustavu računala korisnika, a jedini uvjet jest mogućnost pristupa internetu. Upravo zbog toga nije potrebno postavljanje dodatnih specijaliziranih alata, koji često znaju opterećivati računalo (i novčanik) korisnika.

S obzirom da su prilikom izrade *webGIS* aplikacije korišteni otvoreni standardi i tehnologije poput *WMS*-a i *WFS*-a, besplatni alati poput aplikacijskog poslužitelja *GeoServer*, te skupovi gotovih programskih funkcija otvorenog koda kao što su *ExtJS* i *GeoExt*, omogućena je potpuno besplatna izrada jedne ovakve webGIS aplikacije. Upravo ta činjenica, uz prethodno navedene prednosti, čini webGIS vrlo popularnim alatom ne samo za upravljanje grobljima, već i za mnoge druge svrhe.

Na samom kraju treba spomenuti kako razvoj *webGIS* aplikacije groblja Velika Trapinska ne mora završiti ovim radom. Daljnjim razvojem koda aplikacije, odnosno implementacijom dodatnih funkcionalnosti, kao što je to npr. mogućnost obrade i stvaranja novih vektorskih objekata na karti, mogućnost ispisa kartografskog prikaza ili mogućnost obrade i unosa novih atributnih podataka, postigla bi se zavidna razina funkcionalnost, a time i mogućnost šire praktične upotrebe.

Literatura

1. Davis S. (2007): *GIS for Web Developers: Adding 'Where' to Your Web Applications*, Pragmatic Bookshelf
2. Garcia J. (2011): *ExtJS in Action*, Manning
3. Medak D. (2007): Baze podataka, Slajdovi s predavanja, Geodetski fakultet u Zagrebu
4. Nixon R. (2009): *Learning PHP, MySQL, and JavaScript*, O'Reilly
5. Shekhar S., Xiong H. (2008): *Encyclopedia of GIS*, Springer
6. Williams H., Lane D. (2002): *Web Database Applications with PHP & MySQL*, O'Reilly

URL 1: HTML,http://hr.wikipedia.org/wiki/HTML, 13.06.2012

URL 2: JavaScript, http://en.wikipedia.org/wiki/JavaScript, 13.06.2012

URL 3: PHP, http://en.wikipedia.org/wiki/PHP, 14.06.2012

URL 4: PHP manual, http://www.php.net/manual/en/index.php, 14.06.2012

URL 5: JSON, http://www.json.org/, 14.06.2012

URL 6: Open Geospatial Consortium, http://www.opengeospatial.org/ogc, 15.06.2012

URL 7: OGC, http://en.wikipedia.org/wiki/Open_Geospatial_Consortium, 15.06.2012

URL 8: OpenGeo Architecture,

http://opengeo.org/publications/opengeo-architecture/, 16.06.2012

URL 9: Web Server, http://en.wikipedia.org/wiki/Web_server, 18.06.2012

URL 10: Apache HTTP Server,

http://en.wikipedia.org/wiki/Apache_HTTP_Server, 18.06.2012

URL 11: HTTP,http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol, 18.06.2012

URL 12: Object-relational database,

http://en.wikipedia.org/wiki/Object-relational_database, 19.06.2012

URL 13: PostgreSQL, http://www.postgresql.org/, 19.06.2012

URL 14: PostgreSQL, http://en.wikipedia.org/wiki/PostgreSQL, 19.06.2012

URL 15: PostGIS, http://trac.osgeo.org/postgis/wiki/UsersWikiMain, 19.06.2012

URL 16: GeoServer,

http://geoserver.org/display/GEOS/What+is+Geoserver, 20.06.2012

URL 17: OpenLayers, http://opengeo.org/technology/openlayers/, 21.06.2012

URL 18: OpenLayers documentation, http://docs.openlayers.org/, 21.06.2012

URL 19: Sencha ExtJS Framework,

http://www.sencha.com/products/extjs/. 22.06.2012

URL 20: What is ExtJS, http://whatisextjs.com/what-is-extjs, 22.06.2012

URL 21: ExtJS, http://en.wikipedia.org/wiki/ExtJS, 22.06.2012

URL 22: GeoExt, http://geoext.org/, 22.06.2012

URL 23: GeoExt, http://opengeo.org/technology/geoext/, 22.06.2012

URL 24: OpenLayers FAQ,

http://trac.osgeo.org/openlayers/wiki/FrequentlyAskedQuestions, 26.06.2012

URL 25: XMLHttpRequest, http://en.wikipedia.org/wiki/XMLHttpRequest, 26.06.2012

URL 26: OpenStreetMap, http://hr.wikipedia.org/wiki/OpenStreetMap, 26.06.2012

URL 27: How to Install and Configure PHP 5 to Run with Apache on Windows,

http://www.thesitewizard.com/php/install-php-5-apache-windows.shtml, 26.06.2012

URL 28: Ext JS 4 Layouts Guide,

http://rawberg.com/wp-content/uploads/ExtJS-Layouts.pdf, 27.06.2012

URL 29: ExtJS Layout Examples,

http://dev.sencha.com/deploy/ext-4.0.0/examples/layout-browser/layout-browser.html, 27.06.2012

Popis slika

Prilozi

Prilog 1. Izvorni kod aplikacije

- *index.html*

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title id='title'>WebGIS</title>

<!-- ** CSS ** -->
<link rel="stylesheet" type="text/css" href="ExtJS/resources/css/ext-all.css"/>
<link rel="stylesheet" type="text/css" href="GeoExt/resources/css/geoext-all.css"/>
<link rel="stylesheet" type="text/css" href="ExtJS/examples/layout-browser/layout-
  browser.css"/>
<link rel="stylesheet" type="text/css" href="data_view.css"/>

<!-- ** OpenLayers ** -->
<script type="text/javascript" src="OpenLayers/OpenLayers.js"></script>

<!-- ** Javascript ** -->
<script type="text/javascript" src="ExtJS/adapter/ext/ext-base.js"></script>
<script type="text/javascript" src="ExtJS/ext-all-debug.js"></script>

<!-- ** GeoExt ** -->
<script type="text/javascript" src="GeoExt/script/GeoExt.js"></script>

<!-- ** Google Maps layer ** -->
<script src="http://maps.google.com/maps/api/js?v=3.2&sensor=false"></script>

<!-- ** aplikacija ** -->
<script type="text/javascript" src="map.js"></script>
<script type="text/javascript" src="data_view.js"></script>
<script type="text/javascript" src="grid.js"></script>
<script type="text/javascript" src="app.js"></script>

</head>
<body>
</body>
</html>
```

- *app.js*

```
Ext.BLANK_IMAGE_URL = 'ExtJS/resources/images/default/s.gif';

Ext.onReady(function() {

    var viewport = new Ext.Viewport({
        layout: 'border',
        items: [{
            xtype: 'box',
            id: 'header',
            region: 'north',
            html: '<h1> WebGIS groblja Velika Trapinska</h1>',
            height: 30,
        },{
            region: 'east',
            layout:'vbox',
            border: false,
            split: true,
            width: 250,
            margins: '2 5 5 0',
            layoutConfig : {
                align : 'stretch'
                },
            items: [{
                title: 'Ukopane osobe',
                layout: 'fit',
                flex: 1,
                items: ukopaniDv
                },
                {
                title: 'Korisnici grobova',
                layout: 'fit',
                flex: 1,
                items: korisniciDv
                }]
        },{
            region: 'center',
            layout: 'fit',
            border: false,
            margins: '2 0 5 0',
            items: mapPanel
        },{
```

```
            region: 'west',
            layout: 'fit',
            border: false,
            collapsible: true,
            split: true,
            title: 'Popis grobova',
            width: 200,
            margins: '2 0 5 5',
            cmargins: '2 5 5 5',
            items: groboviGrid
        }]
    });

});
```

- *map.js*

```
var map;
var wmsLayer;
var wfsLayer;
var selectControl;
var mapPanel;
var wfsStore;

OpenLayers.ProxyHost = "/cgi-bin/proxy.cgi?url=";

    var hdks6 = new OpenLayers.Projection("EPSG:900901");

    var google_mercator = new OpenLayers.Projection("EPSG:900913");

    var wgs84 = new OpenLayers.Projection("EPSG:4326");

    map = new OpenLayers.Map({
        projection: google_mercator,
        displayProjection: wgs84,
        units: "m",
        numZoomLevels: 8,
        maxResolution: 1.1,
        maxExtent: new OpenLayers.Bounds(
            1945458, 5738254,
            1947094, 5739699
        ),
        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(),
            new OpenLayers.Control.LayerSwitcher(),
        ]
    });

    var openstreetmap = new OpenLayers.Layer.OSM();

    var google_satellite = new OpenLayers.Layer.Google(
        "Google Satellite",
        {type: google.maps.MapTypeId.SATELLITE, numZoomLevels: 22, visibility:
false}
    );

    wmsLayer = new OpenLayers.Layer.WMS(
        "DOF groblja",
        "http://31.147.204.28:8080/geoserver/adam_diplomski/wms",
        {
            layers: 'adam_diplomski:6E29-13-DOF',
            isBaseLayer: true,
        }
    );

    wfsLayer = new OpenLayers.Layer.Vector("Grobovi", {
        strategies: [new OpenLayers.Strategy.BBOX()],
        reportError: true,
        visibility: true,
        protocol: new OpenLayers.Protocol.WFS({
            version: "1.1.0",
            url:  "http://31.147.204.28:8080/geoserver/adam_diplomski/wfs",
            featureType: "grobovi",
            featurePrefix: "adam_diplomski",
            featureNS: "http://www.geof.hr",
            geometryName: "the_geom",
            srsName: "EPSG:3857",
        }),
        isBaseLayer: false
    });
```

```
map.addLayers([wmsLayer, wfsLayer, openstreetmap, google_satellite]);

selectControl = new OpenLayers.Control.SelectFeature(
    [wfsLayer],
    {
        clickout: true, toggle: false,
        multiple: false, hover: false,
    }
);

map.addControl(selectControl);
selectControl.activate();

wfsLayer.events.on ({
    "featureselected": function(e) {
        var record = e.feature.data.id;
        if (record) {
                Ext.Ajax.request({
                    url: 'data.php',
                    method: 'GET',
                    params: {broj_groba: record},
                    success: function(response, options) {
                            var json = Ext.decode(response.responseText);
                        ukopaniDvStore.loadData(json.ukopani);
                        korisniciDvStore.loadData(json.korisnici);
                         },
                    failure: function (){
                      alert("Pogreška!")
                    }
                });
        }
    }
});

mapPanel = new GeoExt.MapPanel({
    title: 'Karta',
    //region: 'center',
    map: map
});

wfsStore = new GeoExt.data.FeatureStore({
    autoload: true,
    layer: wfsLayer,
```

```
        fields: [
            {name: 'gid', type: 'int', mapping: 'gid'},
            {name: 'id', type: 'int', mapping: 'id'},
            {name: 'polje', type: 'string', mapping: 'polje'},
            {name: 'red', type: 'string', mapping: 'red'},
            {name: 'vrsta_grobnog_mjesta', type: 'string', mapping:
'vrsta_grobnog_mjesta'},
        ],
        sortInfo:{field: 'id', direction: 'ASC'}

    });
```

- *grid.js*

```
var groboviCM;
var groboviGrid;

groboviCM = new Ext.grid.ColumnModel(
    [{
      header: 'gid',
      readOnly: true,
      dataIndex: 'gid',
      width: 50,
      hidden: true
    },{
      header: 'Br groba',
      dataIndex: 'id',
      width: 50,
    },{
      header: 'Polje',
      dataIndex: 'polje',
      width: 50,
    },{
      header: 'Red',
      dataIndex: 'red',
      width: 50,
      hidden: true,
    },{
      header: "Vrsta grobnog mjesta",
      dataIndex: 'vrsta_grobnog_mjesta',
      width: 80,
    }]
);
```

```
groboviGrid =  new Ext.grid.GridPanel({
    id: 'groboviGrid',
    store: wfsStore,
    cm: groboviCM,
    enableColLock: false,
    selModel: new GeoExt.grid.FeatureSelectionModel({
        autoPanMapOnSelection: true
    }),
    listeners: {
        rowclick: function(grid, rowIndex) {
                    var record = grid.getStore().getAt(rowIndex);
                    if (record) {
                      Ext.Ajax.request({
                          url: 'data.php',
                          method: 'GET',
                          params: {broj_groba: record.get('id')},
                          success: function(response, options) {
                                  var json = Ext.decode(response.responseText);
                              ukopaniDvStore.loadData(json.ukopani);
                              korisniciDvStore.loadData(json.korisnici);
                               },
                          failure: function (){
                            alert("Pogreška!")
                          }
                      });
                    }
                  }
    }
});
```

- *data_view.js*

```
var ukopaniDvStore;
var ukopaniDVTpl;
var ukopaniDv;
var korisniciDvStore;
var korisniciDVTpl;
var korisniciDv;

ukopaniDvStore = new Ext.data.Store({
        autoLoad: false,
        id: 'ukopaniDvStore',
        reader: new Ext.data.JsonReader({
            root: 'results_ukopani',
            totalProperty: 'total_ukopani',
        },
        [
        { name: 'id', mapping: 'id' },
        { name: 'broj_groba', mapping: 'broj_groba' },
        { name: 'redni_broj', mapping: 'redni_broj' },
        { name: 'ime', mapping: 'ime' },
        { name: 'prezime', mapping: 'prezime' },
        { name: 'spol', mapping: 'spol' },
        { name: 'nadnevak_rodjenja', mapping: 'nadnevak_rodjenja' },
        { name: 'nadnevak_smrti', mapping: 'nadnevak_smrti' }
        ])
});

ukopaniDVTpl = new Ext.XTemplate(
        '<tpl for=".">',
        '<div class="Wrap" id="ukopani_{id}">',
        '<div class="Name">{broj_groba} {prezime} {ime}</div>',
        '<div>',
        '<span class="title">Redni broj: </span>',
        '{redni_broj}',
        '</div>',
        '<div>',
        '<span class="title">Spol: </span>',
        '{spol}',
        '</div>',
        '<div>',
        '<span class="title">Nadnevak rođenja: </span>',
        '{nadnevak_rodjenja}',
```

```
        '</div>',
        '<div>',
        '<span class="title">Nadnevak smrti: </span>',
        '{nadnevak_smrti}',
        '</div>',
        '</div>',
        '</tpl>'
);

ukopaniDv = new Ext.DataView({
        tpl: ukopaniDVTpl,
        store: ukopaniDvStore,
        singleSelect: true,
        itemSelector: 'div.Wrap',
        selectedClass: 'Selected',
        overClass: 'Over',
        style: 'overflow:auto; background-color: #FFFFFF;'
});

korisniciDvStore = new Ext.data.Store({
        autoLoad: false,
        id: 'korisniciDvStore',
        reader: new Ext.data.JsonReader({
            root: 'results_korisnici',
            totalProperty: 'total_korisnici',
        },
        [
        { name: 'id', mapping: 'id' },
        { name: 'ime', mapping: 'ime' },
        { name: 'prezime', mapping: 'prezime' },
        { name: 'broj_groba', mapping: 'broj_groba' },
        { name: 'jmbg', mapping: 'jmbg' },
        { name: 'adresa', mapping: 'adresa' },
        { name: 'osnova_stjecanja', mapping: 'osnova_stjecanja' },
        { name: 'placanje', mapping: 'placanje' },
        { name: 'iznos_uplate', mapping: 'iznos_uplate' }
        ])
});

korisniciDVTpl = new Ext.XTemplate(
        '<tpl for=".">',
        '<div class="Wrap" id="korisnik_{id}">',
        '<div class="Name">{prezime} {ime}</div>',
```

```
        '<div>',
        '<span class="title">ID: </span>',
        '{id}',
        '</div>',
        '<div>',
        '<span class="title">Broj groba: </span>',
        '{broj_groba}',
        '</div>',
        '<div>',
        '<span class="title">JMBG: </span>',
        '{jmbg}',
        '</div>',
        '<div>',
        '<span class="title">Adresa: </span>',
        '{adresa}',
        '</div>',
        '<div>',
        '<span class="title">Osnova stjecanja: </span>',
        '{osnova_stjecanja}',
        '</div>',
        '<div>',
        '<span class="title">Placanje: </span>',
        '{placanje}',
        '</div>',
        '<div>',
        '<span class="title">Iznos uplate: </span>',
        '{iznos_uplate}',
        '</div>',
        '</div>',
        '</tpl>'
);

korisniciDv = new Ext.DataView({
        tpl: korisniciDVTpl,
        store: korisniciDvStore,
        singleSelect: true,
        itemSelector: 'div.Wrap',
        selectedClass: 'Selected',
        overClass: 'Over',
        style: 'overflow:auto; background-color: #FFFFFF;'
});
```

- *login.php*

```
<?php
$conn_string = "host=localhost port=5432 user=postgres password=postgres
dbname=postgis";
?>
```

- *data.php*

```
<?php

include("mb_json_encode.php");
require_once 'login.php';

$conn=pg_connect($conn_string);
if (!$conn) {
   die("Error in connection: " . pg_last_error());
}

$broj_groba = '';
if ( isset($_GET['broj_groba'])){
        $broj_groba = $_GET['broj_groba'];

        if ( $broj_groba != "" ){

                $upit_ukopani = "SELECT * FROM ukopani
                                WHERE broj_groba='$broj_groba'";

                $result_ukopani = pg_query($upit_ukopani);

                $nbrows_u = pg_num_rows($result_ukopani);

                if($nbrows_u>0){
                        while($rec_u = pg_fetch_assoc($result_ukopani)) {
                                $arr_u[] = $rec_u;
                        }
                        $jsonresult_u = mb_json_encode($arr_u);

                        echo
'{"ukopani":{"total_ukopani":'.$nbrows_u.',"results_ukopani":'.$jsonresult_u.'},';
                } else {
                        echo '{"total":0, "results":""}';
                }
```

```
                $upit_korisnici = "SELECT id,ime,prezime,broj_groba,jmbg,adresa,
                                   osnova_stjecanja,placanje,iznos_uplate
                                   FROM korisnici,grobovi_korisnici
                                   WHERE grobovi_korisnici.id_korisnika=korisnici.id
                                   AND broj_groba='$broj_groba'";

                $result_korisnici = pg_query($upit_korisnici);

                $nbrows_k = pg_num_rows($result_korisnici);

                if($nbrows_k>0){
                        while($rec_k = pg_fetch_assoc($result_korisnici)) {
                                $arr_k[] = $rec_k;
                        }
                        $jsonresult_k = mb_json_encode($arr_k);
                        echo
'"korisnici":{"total_korisnici":'.$nbrows_k.',"results_korisnici":'.$jsonresult_k.'
}}';
                } else {
                                   echo '"korisnici":{"total_korisnici":0,
"results_korisnici":""}}';
                }

        // free memory
        pg_free_result($result_ukopani);
        // free memory
        pg_free_result($result_korisnici);
        // close connection
        pg_close($conn);

        }
}

?>
```

Prilog 2. Sadržaj priloženog optičkog medija

Br.	Naziv datoteke	Opis sadržaja
1.	diplomski_avinkovic.doc	Tekst diplomskog rada
2.	diplomski_avinkovic.pdf	
3.	podaci/grobovi.shp	Shape datoteka grobova groblja Velika Trapinska
4.	podaci/6E29-13-DOF.tif	Digitalni ortofoto za područje groblja Velika Trapinska
5.	podaci/groblje.xls	Izvorni podaci o grobovima u *.xls formatu
6.	aplikacija/kod_aplikacije.zip	Cjelokupni programski kod webGIS aplikacije
7.	pg_18042012.backup	Backup baze podataka (PostgreSQL) za datum 18.04.2012.

Životopis

Osobni podaci	
Prezime(na) / Ime(na)	**Vinković Adam**
Adresa(e)	Dobriše Cesarića 47, 10 090 Zagreb, Hrvatska
Telefonski broj(evi)	(385) 98 622 762
E-mail	avinkovic@gmail.com
Državljanstvo	Hrvatsko
Datum rođenja	06. 12.1987
Obrazovanje	
Datumi	2009. – 2012.
Naziv dodijeljene kvalifikacije	Magistar inženjer geodezije i geoinformatike
Ime i vrsta organizacije pružatelja obrazovanja i osposobljavanja	Sveučilište u Zagrebu, Geodetski fakultet, Kačićeva 26, 10 000 Zagreb, Hrvatska
Datumi	rujan 2010. – ožujak 2011.
Naziv dodijeljene kvalifikacije	Studentska razmjena – ERASMUS
Ime i vrsta organizacije pružatelja obrazovanja i osposobljavanja	Tehničko sveučilište u Münchenu, Geodetski fakultet, Arcisstraße 21, 80333 München, Njemačka
Datumi	2006. – 2009.
Naziv dodijeljene kvalifikacije	Sveučilišni prvostupnik i inženjer geodezije i geoinformatike
Ime i vrsta organizacije pružatelja obrazovanja i osposobljavanja	Sveučilište u Zagrebu, Geodetski fakultet, Kačićeva 26, 10 000 Zagreb, Hrvatska
Datumi	2002. – 2006.
Naziv dodijeljene kvalifikacije	Srednja stručnja sprema - prirodoslovno-matematička gimnazija
Ime i vrsta organizacije pružatelja obrazovanja i osposobljavanja	5. Gimnazija, Klaićeva 1, 10 000 Zagreb, Hrvatska
Radno iskustvo	
Datumi	lipanj, srpanj, rujan 2009., rujan 2010., kolovoz, rujan 2011.
Zanimanje ili radno mjesto	Studentska ispomoć u geodetskoj tvrtki
Ime i adresa poslodavca	Geosistem d.o.o. , Masarykova 14/I, 33000 Virovitica, Hrvatska
Datumi	travanj - srpanj 2011.
Zanimanje ili radno mjesto	Studentska praksa u geoinformatičkoj tvrtki
Ime i adresa poslodavca	Intergraph SG&I Deutschland GmbH, Reichenbachstraße 3, 85737 Ismaning, Deutschland
Osobne vještine i kompetencije	
Materinski jezik	Hrvatski

Samoprocjena

	Razumijevanje				Govor				Pisanje	
	Slušanje		Čitanje		Govorna interakcija		Govorna produkcija			
Njemački	C2	Iskusni korisnik	C2	Iskusni korisnik	C2	Iskusni korisnik	C2	Iskusni korisnik	C2	Iskusni korisnik
Engleski	C1	Iskusni korisnik	B2	Samostalni korisnik	B2	Samostalni korisnik	B2	Samostalni korisnik	B2	Samostalni korisnik
Talijanski	A2	Temeljni korisnik	A2	Temeljni korisnik	A1	Temeljni korisnik	A1	Temeljni korisnik	A1	Temeljni korisnik

Tehničke vještine i kompetencije

-sposobnost samostalnog korištenja geodetskog instrumentarija (GPS uređaj, totalna stanica, nivelir)

Računalne vještine i kompetencije

- obrada teksta: Microsoft Word
- tablični proračuni: Microsoft Excel
- prezentacijski alati: Microsoft PowerPoint
- obrada slike: Gimp
- CAD alati: Autodesk AutoCAD (2000-2012, Map),
- GIS alati: Intergraph GeoMedia Professional, Intergraph GeoMedia 3D, Intergraph GeoMedia ResPublica Intranet, Skyline TerraBuilder, Quantum GIS, GeoServer, OpenLayers
- baze podataka: PostgreSQL, PostGIS, Microsoft Access
- programiranje: HTML, JavaScript, PHP

Druge vještine i kompetencije

- 2000.-2008. član košarkaškog kluba 'Zrinjevac', 2008.-2010. član košarkaškog kluba 'Podsused', 2010.-2011. član košarkaškog kluba DJK SB München
- 2005. izabran u košarkašku selekciju grada Zagreba, te sudjelovao na međunarodnom košarkaškom turniru u Choletu (Francuska)

Vozačka dozvola

B kategorija

Dodatne informacije

-od 2006. do 2009. dobitnik državne stipendije za postignuto 4.mjesto na prijamnom ispitu geodetskog fakulteta