Univerza
v Ljubljani
Fakulteta
za gradbeništvo
in geodezijo

Jamova 2
1000 Ljubljana, Slovenija
telefon (01) 47 68 500
faks (01) 42 50 681
fgg@fgg.uni-lj.si

Univerzitetni program Geodezija,
smer Prostorska informatika

Kandidat:
Marko Zore

Izdelava spletnega tematskega portala z interaktivno karto

Diplomska naloga št.: 869

Mentor:
izr. prof. dr. Radoš Šumrada

Ljubljana, 29. 9. 2011

POPRAVKI

IZJAVA O AVTORSTVU

Podpisani Marko Zore izjavljam, da sem avtor diplomske naloge z naslovom: »Izdelava spletnega tematskega portala z interaktivno karto«.

Izjavljam, da je elektronska različica v vsem enaka tiskani različici.

Izjavljam, da dovoljujem objavo elektronske različice v repozitoriju UL FGG.

Ljubljana, 13.9.2011 Marko Zore

BIBLIOGRAFSKO – DOKUMENTACIJSKA STRAN IN IZVLEČEK

UDK: **528.9:519.6(043.2)**

Avtor: **Marko Zore**

Mentor: **izr. prof. dr. Radoš Šumrada**

Naslov: **Izdelava spletnega tematskega portala z interaktivno karto**

Obseg in oprema: **52 str., 7 sl.**

Ključne besede: **spletni portal, interaktivna karta, tehnologija Ajax, prostorski podatki za spletni prikaz**

Izvleček:

Naloga opisuje izdelavo tematskega spletnega portala z interaktivno karto. Izdelani spletni portal poizkuša uravnotežiti in smiselno povezati tekstovno in slikovno vsebino z dinamičnim prostorskim prikazom iste tematike. Za dosego dinamičnega prehajanja med različnimi prikazi je uporabljeno dinamično odpiranje podstrani z metodo Ajax. Opisana je problematika prikazovanja vsebine dinamičnih strani v spletnih iskalnikih in reševanje le-te. Naloga opisuje tudi zajem, pripravo in metode prikaza prostorskih podatkov v spletni obliki.

BIBLIOGRAPHIC – DOCUMENTALISTIC INFORMATION

UDC: **528.9:519.6(043.2)**
Author: **Marko Zore**
Supervisor: **Assoc. Prof. Radoš Šumrada, Ph. D.**
Title: **Making 3D-city model for web use**
Notes: **52 p., 7 fig.**
Key words: **web portal, interactive map, Ajax technology, spatial data for web use**

Abstract:

The thesis describes the making of a web portal containing an interactive map. The main objective of the portal is to balance and logically connect the text and the images with the dynamic spatial display of the same topic. To achieve a fluid dynamic transgression between various kinds of display only dynamic subpage opening is used, via the Ajax method. Described is also the problematic of the display of content of dynamic pages in different web browsers, and several solutions to the problem. The overview, preparation and methods of displaying spatial data in web form are also described in the thesis.

KAZALO VSEBINE

KAZALO SLIK

OKRAJŠAVE IN SIMBOLI

CSS	Stilske predloge (ang. Cascading Style Sheets)
DOM	Aplikacijski programski vmesnik za delo z objekti pri označevalnih jezikih (ang. Document Object Model)
GIS	Geografski informacijski sistem
HTML	Markirni ali označevalni jezik (ang. Hypertext markup language)
HTTP	Protokol za prenos nadbesedila (ang. HyperText Transfer Protocol)
IP	Internetni protokol (ang. Internet Protocol)
ISO	Mednarodna organizacija za standardizacijo (ang. International organization for standardization)
IT	Informacijska tehnologija
SEO	Optimizacija spletnih strani (ang. Search Engine Optimization)
SLD	(ang. Styled Layer Descriptor)
TCP	Protokol za nadzor prenosa (ang. Transmission Control Protocol)
URI	Enotni identifikator virov (ang. Uniform resource identifier)
URL	Enotni lokator virov (ang. Uniform resource locator)
URN	Enotno ime virov (ang. Uniform resource name)
WCS	Spletni servis za prostorske podatkovne sloje (ang. Web Coverage Service)
WFS	Objektni spletni servis (ang. Web Feature Service)
WMS	Kartografski spletni servis (ang. Web Map Service)
XML	Standardizirani jezik za zapis elektronskih dokumentov (ang. Extensible markup language)

1 UVOD

Pojav spletnega portala Google Maps v letu 2005 je pomenil trenutek začetka vzpona za dve spletni tehnologiji, ki sta danes že vsakdanjost. Prva je tehnologija Ajax (asynchronous JavaScript and XML), ki je prekinila potrebo po čakanju nalaganja strani ob vsaki spremembi, ki smo jo sprožili med ogledom spletne strani. Posledica je pojav dinamičnih spletnih strani. Druga tehnologija pa se odraža v interaktivnih kartografski spletnih portalih. Interaktivna spletna karta je namreč z združitvijo dinamičnosti in povezavo z medmrežjem uporabniško precej napredovala od klasične »analogne« tematske kartografije močno občutljivo na staranje podatkov. Interaktivna karta je sedaj poljubno velika, saj se odvečna območja preprosto ne prikažejo. Dosega poljubno merilo, saj so podatki shranjeni v več nivojih. Omogoča več vsebine, saj se odvečni podatki preprosto skrijejo. S povezavo v medmrežje pa zagotavlja ažurnost in širok spekter podatkovnih virov. So pa seveda tudi slabe plati. Zanesljivost in kakovost široke množice podatkov je lahko hitro vprašljiva.

Motiv te diplomske naloge je uporabiti zgoraj opisane tehnologije v konkretnem primeru, to je v izdelavi spletnega portala z interaktivno karto. Je tudi poizkus uravnoteženja statične opisne vsebine z interaktivnim prostorskim prikazom. V spletu lahko namreč obiščemo ogromno statičnih opisnih enciklopedij, ki jim manjka dinamičnosti. V drugi skrajnosti se pojavljajo zmogljivi kartografski portali z veliko prostorskimi sloji, a z premalo opisne vsebine, da bi se uporabnik dovolj dobro seznanil z predstavljeno tematiko. Kljub temu, da ima portal določeno tematiko - vojaško zgodovino, se lahko rešitev uporabi modularno za različna področja, ki zahtevajo prostorski prikaz in opisujejo preteklost, sedanjost ali prihodnost.

Izdelava interaktivnega portala pa prinaša tudi nekaj izzivov in težav pri izvedbi in uporabi. Interaktivne spremembe se namreč v URL-vrstici odražajo brez sodelovanja strežnika oz. povedano drugače, zahteva po vsebini ne gre prek URL-zahtevka. Tu nastane problem »enolične identifikacije« prikazane trenutne vsebine, kar povzroča težave uporabnikom, še posebej pa trpi optimizacija strani za iskalnike ali SEO (Search Engine Optimization), ki za indeksacijo spletne vsebine še vedno prisega na klasične URL-povezave.

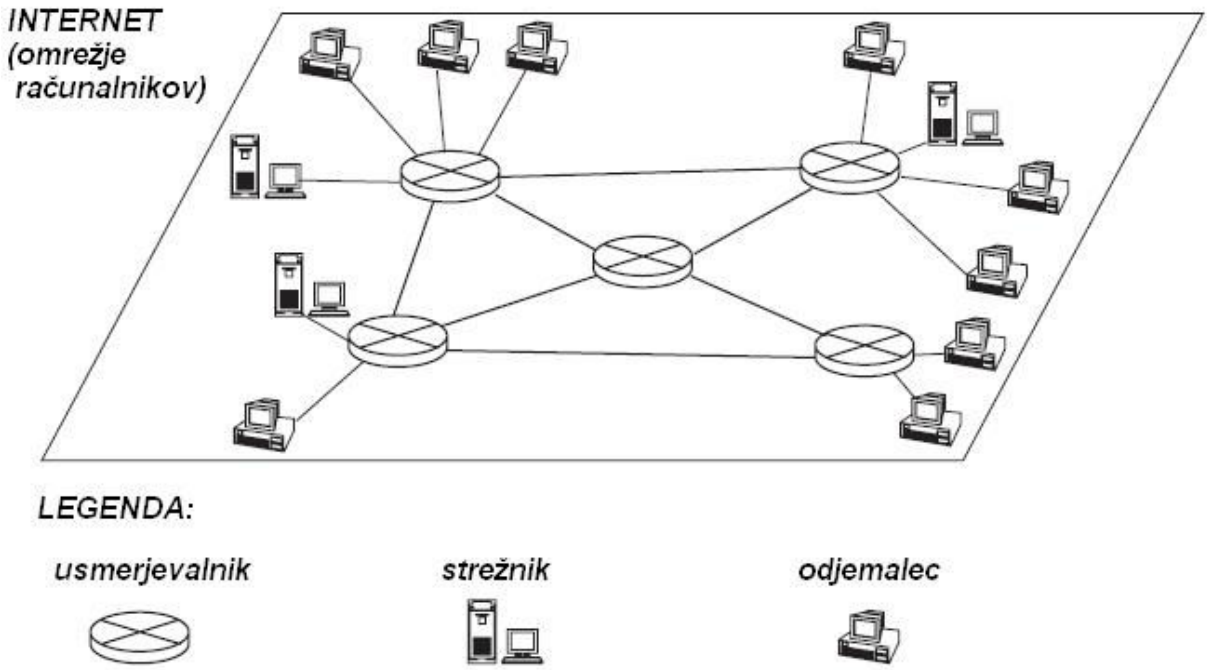
Nekaj malega pozornosti je namenjenega tudi vidiku ekonomičnosti izdelave portala. Vse uporabljene aplikacije pri izdelavi so na voljo brezplačno. Plačljive aplikacije namreč lahko zahtevajo bistven delež začetnega kapitala za vzpostavitev takega portala. Predvsem kvalitetno predstavljen portal je možno tudi tržiti z oglasi ali sponzorji, kar pokriva stroške vzpostavitve in vzdrževanja, lahko pa prinaša celo prihodek.

2 UPORABLJENA INFORMACIJSKA TEHNOLOGIJA

2.1 Internet in svetovni splet

Internet ali medmrežje je računalniško omrežje, ki povezuje množico privatnih, javnih, akademskih in tržnih računalniških omrežij. Gre za najvišji možni nivo računalniškega omrežja, v katerega se je možno povezovati z različnimi elektronskimi, brezžičnimi in optičnimi tehnologijami. Povezovanje znotraj interneta je omogočeno z internetnim skladom protokolov TCP/IP (Transmission Control Protocol/Internet protocol). Sklad protokolov je bil

definiran že leta 1974, gre pa za dva ločena protokola, ki pa se največkrat uporabljata skupaj. Internet omogoča prenos in dostop do različnih informacijskih virov in uslug, kot so svetovni splet, elektronska pošta in neposredni klepet. V zadnjem času se v obliki novih protokolov in pristopov internetu prilagajajo tudi tradicionalni elektronski mediji, kot so telefon, radio in televizija.



Shrema računalniškega omrežja – Interneta
(Hercog Drago: Svetovni splet)

Začetki interneta sodijo v šestdeseta leta prejšnjega stoletja. Takrat so bila računalniška omrežja precej bolj primitivna in občutljiva. Kot pri mnogo stvareh, je tudi na področju interneta raziskave sprožila vojska. Obrambno ministrstvo ZDA ustanovi agencijo DARPA (Defence Advanced Research Projects Agency), ki si je kot cilj zadala vzpostaviti omrežje, po katerem bi med posameznimi strateškimi točkami na ozemlju ZDA potekali podatki v digitalni obliki. Skozi raziskave so vzpostavili ARPANET mrežo, katero so skozi čas vedno bolj izpopolnjevali in razvijali. V sedemdesetih letih je DARPA v razvoj vključila tudi posamezne univerze in podjetja. Glavni rezultat raziskav je bila uvedba sklada protokolov TCP/IP v letu 1982, ki je temelj internetne povezave še danes.

Svetovni splet je hipertekstni sistem, ki deluje v okviru interneta. Informacije v svetovnem spletu so predstavljene v obliki hipertekstnih dokumentov, ki jih pregledujemo s spletnim brskalnikom. Besedilni spletni dokumenti se imenujejo spletne strani, smiselno povezanim spletnim mestom pa spletišče. Glavna lastnost hiperteksta ali nadbesedila je sicer označevanje vsebinskih elementov z povezavami na druge dokumente ali dele besedila znotraj istega dokumenta.

Vsak računalnik ali naprava povezana v medmrežju ima dodeljeno svojo lastno oznako oziroma številko. To je bilo omogočeno z uvedbo IP-številk, oz. IP-naslovov. Za dodeljevanje številk skrbi organizacija IANA (Internet Assigned Numbers Authority), ki preko pooblaščenih organizacij dodeljuje IP-naslove po posameznih regijah. Na nivoju posameznih uporabnikov se IP-naslove dodeljuje statično ali dinamično. Statična dodelitev IP-naslova pomeni, da se ob vsaki vključitvi v medmrežje računalniku dodeli enak naslov. To je pogoj pri strežnikih, pri katerih je URL-naslov neposredno povezan z IP-naslovom. Dinamična

dodelitev IP-naslova, pa ob vsakem dostopu do medmrežja računalniku dodeli različen IP-naslov. Prednost dinamičnega naslova je večja varnost, saj na primer zmanjšuje verjetnost vdora ali kakega drugega nadlegovanja uporabnika.

Po trenutno veljavnem standardu IPv4, ki uporablja desetiški sistem, naslov sestavljajo štiri trimestne številke v razponu od 0 do 255 ločene s piko (npr. 195.3.46.3). Ker se število računalnikov vključenih v medmrežje nezadržno širi, število kombinacij IP-števila po IPv4 standardu kmalu več ne bodo zadoščale za enolično označevanje računalnikov. V letu 2011 so bili namreč podeljeni zadnji prosti sklopi kombinacij teh števil. Za novo označevanje pa je že pripravljen nov šestnajstinski (heksadecimalni) sistem po standardu IPv6. Gre za osem štirimestnih šestnajstiških števil (torej števila od 0-9 in črke od A-F) ločenih s piko (npr. 2101:170:0000:0000:0000:0000:0000:0116 ali skrajšano 2101:170::116). Ta sistem omogoča bistveno večje število enoličnih kombinacij, deloval pa bo hitreje in bolj učinkovito. Naslednja leta bodo tako zaznamovana s prehodom na nov sistem IP označevanja.

Za spletnega uporabnika pa je IP-naslov težaven za uporabo, saj le ta ne asociira na vsebino, ki jo strežnik ponuja na določenemu IP-naslovu. Zato je bil uveden URI (ang. Uniform Resource Identifier) oz. »enotni identifikator virov«. Gre za enoten način za poimenovanje dokumentov in drugih virov na internetu ali v lokalnem omrežju v obliki črk in besed. Te pa uporabniku veliko več povedo o računalniku po katerem poizvedujejo. URI lahko uporabljamo v dveh oblikah: kot URL (ang. Uniform Resource Locator) oz. »»enotni lokator virov«, ali kot URN (ang. Uniform Resource Name) oz. »enotno ime virov«. URL in URN imata podoben pomen, vendar delujeta na drugačen način.

URL (npr. http://www.vzorec.si) je sestavljen iz dveh ključnih delov: iz lokacije vira (www.spletna-stran.si) in metode, s katero lahko do vira dostopamo (http://). Tak način označevanja vira je zelo praktičen, saj URL običajno vsebuje vse podatke, ki jih potrebujemo za odkritje na tak način označenega dokumenta. URL je relativno enostavno uporabljati, zato je v današnjem času pri spletni uporabi precej razširjen.

URL povezuje vir in lokacijo v nerazdružljivi povezani obliki. To pa včasih ni najbolje, saj vir in lokacija nista ista stvar. Problem nastane pri selitvi ali kopiranju nekega vira na drugo lokacijo, vir mora tedaj spremeniti svoj URL. Če ponujamo enolično datoteko na več lokacijah, bo imela ta datoteka toliko različnih URL, kot bo kopij teh datotek. To težavo lahko reši URN, ki datoteko ustrezno poimenuje, nič pa ne govori o lokaciji in metodi dostopa in je tako neodvisna od lokacije. Uporaba URN pa je prav zaradi neodvisnosti od lokacije za uporabo bolj zapletena in zahtevna, zato se v medmrežju ni uveljavila tako izrazito (Kozierok, 2003-2005)

Za splet je pomemben tudi HTTP-protokol. HTTP je komunikacijski protokol med strankami in strežniki. Je glavna metoda za prenos informacij na spletu in je primarno namenjen prenosu HTML-strani. V vlogi HTTP-stranke je največkrat spletni brskalnik, ki začne zahtevo z vzpostavitvijo TCP-povezave na izbrani priklop na oddaljenem gostitelju (strežniku). Stranka ima po priklopu v okviru HTTP-protokola na voljo več različnih zahtevkov. Najpogostejši je zahtevek »GET«, ki predstavlja zahtevo, za pridobitev vsebine datoteke. Primer zahtevka HTTP-stranke, ki od HTTP-strežnika, delujočega na domeni www.vzorec.si zahteva osnovno HTML datoteko oziroma osnovno spletno stran:

```
GET /index.html HTTP/1.1
Host: www.vzorec.si
```

Gostitelj (strežnik) na zahtevek odgovori z posredovanjem vsebine zahtevane datoteke, če pa ta ne obstaja, pa s sporočilom o napaki.

Za prenos podatkov in komuniciranje prek interneta uporabljamo različne pristope. Lahko »brskamo« po spletu, oz. si ogledujemo in uporabljamo spletne strani, lahko pošiljamo elektronsko pošto, se neposredno pogovarjamo z drugimi uporabniki prek spletnih klepetalnic, si lahko izmenjujemo datoteke različnih vrst, itd. Vsak pristop uporablja nekoliko drugačen pristop poimenovanja računalnika s katerim stopamo v stik. Za spletno pošto uporabljamo »naslov elektronske pošte« (npr. marko@elposta.si), v spletni klepetalnici preprosto kliknemo na »vzdevek« osebe, ki nas poveže z njegovim računalnikom, in podobno.

2.2 HTML

HTML (Hyper Text Markup Language) je označevalni jezik za izdelavo spletnih strani. Kot označevalni jezik se je HTML-jezik zgledoval po SGML (Standard Generalized Markup Language) ISO-standardu, a tekom razvoja ni zadoščal vsem kriterijem standarda in se je razvijal vzporedno. HTML brskalnikom omogoča interpretiranje in branje tekstovne slikovne in druge vsebine v ustrezno obliko prijazno uporabnikom.

HTML je razvil fizik Tim Berners-Lee v letu 1991, ko je poleg jezika razvil tudi brskalnik in strežniški program. HTML je bil plod njegovih 10-letnih izkušenj na področju oblikovanja tekstovnih spletnih datotek. Leta 1994 je Tim Berners-Lee ustanovil World Wide Web Consortium (W3C), katerega namen je bil razvoj in razširanje standardov spletnih tehnologij, začenši prav z HTML. Prvi HTML-standard HTML 2.0 je bil izdan leta 1995, sledila je verzija HTML 3.2 v letu 1997, na katerega sta imela velik vpliv tudi ponudnika brskalnikov Netscape in Microsoft. V letu 1999 je bila izdan zadnji standard HTML 4.01, v katerega je bilo vključeno tudi povezovanje s predlogami CSS. V letu 2000 je izšel standard XHTML 1.0. XHTML je verzija HTML-jezika, ki uporablja XML-sintakso. XHTML-jezik je ohranil vse HTML-elemente, le zapis mora ustrezati XML-standardu. To XHTML-jeziku omogoča, da ga je možno razvijati, preverjati in tvoriti tudi z XML programsko opremo.

Osnovne sintaktične enote HTML so značke. Sintaksa značke je ime značke obdana z lomljenim oklepajem (<ime_ značke>). Imena značk morajo biti zapisana z malimi črkami. Značke praviloma nastopajo v parih – z začetno in končno značko. Izjema so tiste značke, ki jih zaključimo že s prvo značko in sicer tako, da pred končnim oklepajem dodamo poševnico (npr.
). Končna značka se od začetne razlikuje po dodani poševnici pred imenom (</ime_značke>). Nekatere značke v HTML se ne zaključuje. V XHTML pa morajo biti zaradi pravil XML zaključene vse značke. Med začetno in končno značko vstavimo ustrezno vsebino za prikaz. Tako tvorjen zapis imenujemo HTML-element. Pomembno HTML-pravilo je gnezdenje. Določen element lahko vsebuje druge HTML-elemente. Med začetno in končno značko enega elementa lahko vstavimo drug element. Pomembno pa je, da se notranji element konča, še preden se konča zunanji. Primer ustreznega gnezdenja:

```
<zunanji>
      <notranji>
      </notranji>
</zunanji>
```

Primer neustreznega gnezdenja:

```
<zunanji>
<notranji>
```

```
</zunanji>
</notranji>
```

Zgornji primer je napačen, in ga brskalniki ne bodo znali pravilno interpretirati.

Za pripombe in drugo vsebino, ki je ne želimo prikazati znotraj HTML-dokumenta uporabimo posebno oznako:

```
<!-- tega brskalniki ne bodo prebrali -->
```

HTML pozna precej prikaznih elementov. Elemente lahko glede na njihove lastnosti združimo v več skupin. Najpomembnejši sta dve skupini, **skupina bločnih** in **skupina vrstičnih elementov**. Za skupino bločnih elementov je značilno, da se začnejo v novi vrstici in lahko obsegajo več vrstic. Zavzamejo torej neko območje. Za skupino vrstičnih elementov pa je značilno, da se lahko začnejo in končajo sredi vrstice. Zavzamejo torej poljubno dolgo zaporedje teksta.

Najbolj pogosti **bločni elementi** so:

Element , ki označuje neko zaključeno območje, ki ga želimo oblikovno ali fizično ločiti od druge vsebine.

Element <table> oz. tabela, ki omogoča tabelarično razvrščanje podatkov, v nekaterih primerih pa je uporaben tudi za izgradnjo pravilne strukture spletne strani ali njenih delov.

Element <p> označuje tekst, ki ga tvorimo v odstavek. Tekst označen s to značko se bo začel v novi vrsti. Po zaključni znački elementa se bo vsak naslednji element pričel v novi vrsti.

Najbolj pogosti **vrstični elementi** so:

Element <a href="http://www.domena.si/podstran1.html" > s katerim definiramo povezavo na drugo spletno stran. Besedilo, ki je označeno s to značko ima že privezo označeno obliko za besedilo za povezavo. Besedilo je običajno modre barve in podčrtano. Kurzor miške se ob prekritju elementa iz puščice spremeni v obliko roke, kar že vizualno nakazuje možnost klika. Povezava ima definirano lastnost »href« oz. povezavo, kateri bomo ob kliku sledili.

S klikom na povezavo lahko odpremo novo HTML-stran. Prejšnjo stran običajno s tem zapremo, lahko pa jo odpremo v novem oknu (povezavi definiramo lastnost target=«blank_«). Poznamo absolutne in relativne povezave. Absolutna povezava je sestavljena iz celotnega URL:

http://www.domena.si/podstran1.html

Prikazuje lahko poljubno spletno stran na medmrežju, vključno z podstranmi lastne spletne strani.

Relativna povezava pa je zapisana v skrajšani obliki URL-ja in se predstavlja lokacijo ciljne datoteke glede na trenutno stran, na kateri je povezava objavljena. Če želimo na primer na spletni strani:

http://www.domena.si/podstran1.html

objaviti povezavo na neko drugo podstran:

http://www.domena.si/mapa/podstran2.html

lahko to povezavo zapišemo v relativni obliki:

mapa/podstran2.html

kjer je z »mapa« označena mapa na strežniku (velja za klasično strukturo spletnih strani), v kateri je HTML-datoteka »podstran2«. Ker sta obe mapi na istem strežniku, lahko oznako z domeno, ki kaže na strežnik, izpustimo. Pravimo, da smo uporabili relativno povezavo.

Z značko <b> označimo tekst, ki ga želimo prikazati krepko, z značko <i> pa tekst, ki ga želimo prikazati poševno.

Značko največkrat uporabimo, ko želimo spremeniti CSS-lastnost točno določenega niza teksta znotraj drugih HTML-elementov.

Značko
 uporabimo za prelom vrstice.

Z značko v dokument ustavimo sliko. V lastnosti »src« definiramo povezavo do slike.

2.3 CSS

CSS (Cascading Style Sheets) so predloge, ki opisujejo oblikovno semantiko (izgled in oblikovanje) označevalnih jezikov. CSS omogoča prilagoditev vsebine v obliki označevalnega jezika za različne tipe naprav, kot so majhni ali veliki zasloni, tiskalniki, itd. CSS se danes uporablja predvsem za oblikovanje HTML spletnih strani. Je pa CSS od HTML povsem neodvisen in se lahko uporablja tudi za oblikovanje XML-dokumentov in nekaterimi drugimi jeziki, ki temeljijo na XML.

CSS je bil razvit predvsem z namenom ločitve vsebine in oblike dokumenta, kar bi omogočilo prikaz iste vsebine na različne oblikovne načine in za različne izhodne naprave zgolj z zamenjavo CSS-predloge. Prvo specifikacijo predloge CSS je razvil W3C leta 1996. Leta 1998 je izšel standard CSS2 z novo dodanimi lastnostmi, z namenom prilagoditvi obstoječim brskalnikom pa je izšel popravljen standard CSS2.1. Ta verzija je še danes aktualna, čeprav se že celo zadne desetletje pripravlja standard CSS3.

CSS-predloge se v povezavi z HTML imenujejo tudi kaskadne predloge, saj je te možno vključevati v HTML-dokument v treh hierarhičnih stopnjah.

CSS-predlogo lahko v HTML-datoteko vključimo v obliki **zunanje datoteke**. Uporabimo HTML-značko »link« in ji definiramo ustrezne lastnosti:

```
<link rel="StyleSheet" type="text/css" href="oblika.css" />
```

Oblika, definirana v zunanji datoteki, bo veljala za celotni HTML-dokument. Vključevanje zunanje datoteke omogoča, da lahko isto CSS-predlogo uporabljajo tudi drugi HTML-dokumenti.

CSS-predlogo lahko v HTML-datoteko vključimo z uporabo HTML-značke »style«:

```
<style type="text/css">
...
</style>
```

CSS-vsebina bo umeščena med začetno in končno značko »style« in tudi v tem primeru bo veljala za cel dokument. V primeru, da ima HTML-dokument definirano CSS-predlogo v obeh oblikah hkrati, najprej uveljavi navodila zunanje CSS-datoteke, zatem pa uveljavi še navodila označena z značko »style«. Če se navodila nanašajo na isti HTML-element v obeh CSS-predlogah ima prednost slednja.

CSS-lastnost pa lahko uveljavimo HTML-elementu neposredno z definiranjem lastnosti style:
`<img style="..." />`

To CSS-navodilo velja samo za HTML-element v katerem je definiran in ima prednost pred vsemi prej opisanimi načini določanja CSS-oblike.

CSS-predloge določajo HTML-elementom obliko v obliki stilov. Vsaka oblikovna lastnost ima svoje ime in vrednost. Navedimo primere stilov za določitev barve, velikosti pisave in določitev velikosti, oblike ter barve okvirja:

```
color:red; font-size:10pt; border:5px solid red;
```

Vidimo, da lahko eni oblikovni lastnosti določimo več vrednosti različnega tipa tudi hkrati. Lastnost ima lahko torej več vrednosti različnega tipa.

Vsaka od teh lastnosti ima definiran končen (left, right, top...) ali neskončen (20px, 10pt...) nabor za posamezen tip vrednosti. Vrednost je lahko tudi URL-povezava na sliko ali kak drug element.

Če je CSS-stil definiran neposredno v HTML-znački, se nanaša na točno tisti HTML-element. Če pa so stili zapisani v zunanji datoteki ali nanizani znotraj značke »style«, pa je potrebno uvesti tako imenovane **selektorje**. Selektorji določajo, za katere HTML-elemente velja navedeni stil. Selektor, lastnost in vrednost zapišemo na naslednji način:

```
selektor { lastnost: vrednost; ... }
```

Lastnost in vrednost, ki pripišemo selektorju, torej definiramo znotraj zavitega oklepaja. Selektorje lahko glede na ciljno skupino elementov in njihove lastnosti zapisujemo na več načinov. Kot selektor lahko izberemo **posamezen HTML-element**:

```
p { font-family: Arial }
```

V tem primeru bo CSS določil obliko vsem elementom »p«, ki se pojavijo v HTML-dokumentu. Če pa ima posamezen element definiran svoj lasten stil z enako lastnostjo a drugačno vrednostjo, se bo po hirarhiji nastavila tista in element bo drugačen od drugih. To

sicer velja za vse oblike selektorjev. Če želimo nastaviti stil za prav vse elemente v dokumentu, uprabimo selektor »*«:

```
* { color: blue }
```

HTML-elementom lahko definiramo tudi določene omejitve, glede na katere se definira izbor točno določenih elementov. Enemu ali več HTML-elementom lahko nastavimo lastnost »class« s poljubnim nazivom. Na ta razred se lahko sklicuje selektor tako, da uporabimo ime razreda in mu pred imenom dodamo piko ».«:

```
.novo { color: red }
```

Zgornji stil bo veljal za vse HTML-elemente, ki imajo definirano lastnost »class« z nazivom »novo«.

Podobno lahko HTML-elementu nastavitmo tudi lastnost »id«, vendar s to razliko, da mora biti naziv te lastnosti identičen za vsak element posebej. Selektor za tako označen element označimo z nazivom lastnosti »id«, pred naziv pa dodamo znak »#«:

```
#opis { color: red }
```

Selektor tako nastavi stil točno določenemu elementu.

Selektor lahko ločuje tudi **stanje** v katerem se HTML-element nahaja. Posameznemu elementu v tem primeru dodamo še določeno stanje, vmes pa dodamo dvopičje:

```
a:link { color: blue }
a:visited { color: magenta }
p:hover { text-decoration: underline }
```

Selektor bo zgornjim primerom dodal določen stil le v primeru, da je element še neobiskana povezava (link), da je element že obiskana povezava (visited), ali da je nad elementom miškin kurzor (hover).

Selektorje lahko ločujemo tudi na nivo HTML-elementov, glede na njihovo potomstvo. Če želimo na primer izbrati vse HTML-elemente »p«, ki se pojavijo znotraj HTML-elementa »div«, to označimo z **verigo** selektorja:

```
div p { font-style: italic }
```

Potomce elementa zapišemo takoj za osnovnim selektorjem tako, da ju ločimo z presledkom. V verigo lahko umestimo poljubno število potomcev.

Na podoben način označujemo selektorje za vse tiste elemente, katerim določamo enak stil, le da jih medseboj ločimo z vejico:

```
a, p { font-style: italic }
```

Vsem elementom »a« in »p« v dokumentu lahko definiramo enak stil hkrati.

2.4 XML

XML (Extensible Markup Language) je internetni standard razvit leta 1998. Razvila ga je organizacija World Wide Web Consortium (W3C) z namenom poenostaviti dotedanji meta označevalni jezik SGML (Standard Generalized Markup Language). SGML je namreč preveč obsežen in kompleksen jezik, da bi na njegovi podlagi lahko razvijali različnim področjem prilagojene označevalne jezike. V letu 2004 je izšla novejša verzija standarda - XML 1.1, vendar njegova uporaba še ni tako močno razširjena, kot prvotna verzija.

XML predstavlja ogrodje za definiranje označevalnih jezikov prilagojenih najrazličnejšim področjem. Omogoča enostaven in univerzalni način shranjevanja različnih tekstovnih podatkov. Izmenjavo in obdelavo XML-dokumentov podpira mnogo aplikacij različnih področij in namenov. Aplikativne postopke za delo z XML-dokumenti je lahko razvijati, saj je XML odprt, brezplačni standard.

XML pravzaprav ni označevalni jezik, temveč meta označevalni jezik, ki določa pravila za tvorjenje označevalnih jezikov. Posledično XML-jezik nima definirane knjižnice značk. Pri razvoju označevalnega jezika je potrebno definirati nabor značk, prilagojenega tematiki novega označevalnega jezika. XML označevalni jeziki so tako na primer: XHTML, RSS, MathML, GraphML, SVG, MusicXML in še mnogo drugih.

2.5 Spletni skriptni jeziki

2.5.1 Strežniški spletni jeziki in spletni jeziki na uporabnikovi strani

Za uvedbo dinamičnosti in interaktivnosti v spletne prikaze so na voljo nekateri zmogljivi spletni jeziki. Če HTML predstavlja osnovno statično strukturo strani in CSS oblikovno razsežnost spletnih strani, so jeziki, ki jih sedaj opisujemo namenjeni dinamičnim spremembam prikazanih spletnih vsebin.

Spletne jezike ločimo na tiste, ki se izvajajo na strani uporabnika (ang. client-side) in na tiste, ki se izvedejo, še preden so poslani uporabniku (ang. server-side). V prvo skupino med najbolj uporabljanimi sodi JavaScript, v drugo skupino pa predvsem PHP in ASP. Delovanje spletnih jezikov na strani uporabnikov je sledeče: uporabnik, ki si želi ogledati dinamično spletno stran, pošlje zahtevek URL prek brskalnika. Brskalnik se s strežnikom poveže s pomočjo HTTP-protokola. Strežnik pošlje nazaj HTML-datoteko spletne strani in druge datoteke, ki sestavljajo zaokroženo celoto spletne strani (npr. rastrske podobe, glasba...). Skupaj z naštetimi elementi pa je lahko uporabniku poslana tudi JavaScript koda, ki je bodisi del HTML-datoteke, ali pa samostojna datoteka. Če ima uporabnikov brskalnik vključeno možnost vključitve JavaScript kode v HTML-dokument, se le ta izvede ob inicializaciji spletne strani v uporabnikovem brskalniku. Nekateri parametri in funkcije so ves čas ogleda strani v pripravljenosti in se lahko sprožijo z vnaprej predvidenim dogodkom na strani (običajno s klikom na gumb ali povezavo).

Delovanje spletnih jezikov na strani strežnika je drugačno. Uporabnik, ki si želi ogledati spletno stran, pošlje zahtevek URL prek brskalnika. Brskalnik se s strežnikom poveže s pomočjo HTTP-protokola. URL v tem primeru ne zahteva datoteke v HTML-obliki, temveč v taki obliki, da strežnik, preden jo pošlje uporabniku, pripravi HTML-dokument po vnaprej določeni kodi. Da se pred oddanim HTML-zapisom uporabniku le ta pripravi na podlagi PHP-kode, je potrebno HTML-dokument skupaj s PHP-kodo shraniti s končnico ».php«. Strežnik sedaj ve, da mora tako oblikovano datoteko najprej poslati PHP-procesorju, ki izvede kodo in

vrne »čisto« HTML-stran. To stran sedaj strežnik pošlje uporabniku. Ker se je koda PHP že izvedla, ni potrebe, da bi jo pošiljali uporabniku. Zato ta ostane uporabnikom pri uporabi skrita, saj ni več del poslanega HTML-dokumenta.

Za jezike na strani strežnika je značilno, da ne morejo povzročiti dinamičnih sprememb, brez posredovanja strežnika. Zato je za te jezike značilna uporaba dodanih parametrov v URL-zahtevku, največkrat v obliki »URL query stringa« označenega z »?«:

http://www.vzorec.si?parameter=32

Parametri so bolj uporabljana metoda tovrstnih jezikov, saj so na podlagi različnih parametrov lahko izdelane strani v najrazličnejših kombinacijah in izvedbah, kar zagotavlja dinamičnost. Lastnost tovrstnih jezikov je torej, da mora vsak zahtevek po spremembi na strani potovati do strežnika, kjer se ta zahtevek izvede.

Iz različnih potreb pri prikazovanju spletnih strani izhajajo tudi različne stopnje in razmerja glede na njihovo uporabnost. Zaradi svoje različne izvedbe pa lahko izpostavimo tudi naslednje prednosti in slabosti med obemi vrstami spletnih jezikov:
Prednosti spletnih jezikov na strani uporabnika:

- koda se izvaja na strani uporabnika, zato je strežnik manj obremenjen. Izvedene spremembe v brskalniku uporabnika so zato hitrejše ali celo sočasne s pregledovanjem spletne strani,
- vse dinamične spremembe se izvedejo na strani uporabnika in ni potrebno posredovanje strežnika.

Slabosti spletnih jezikov na strani uporabnika:

- koda je uporabnikom vidna in uporabljenih rešitev fizično ni mogoče zaščititi pred kopiranjem.
- Nekateri uporabniki imajo lahko zaradi različnih razlogov izključeno možnost uporabe JavaScript jezika na brskalniku. Delovanje spletne strani je v tem primeru okrnjeno ali pa sploh ne deluje.
- Ker za dinamičnost ni potrebna komunikacija s strežnikom, kljub tekstovnim spremembam tako prikazane strani nimajo lastnega URL-ja, zato jih spletni iskalniki težje evidentiranjo.

Prednost jezikov na strani strežnika:

- koda je uporabnikom skrita in zaščitena pred kopiranjem. V kodi se lahko ohrani tudi morebitne zaznamke, ki ohranjajo preglednost kode, a na delovanje ne vplivajo.
- Vsaka sprememba na strani je zaznamovana z različnim URL-jem poslanem strežniku, katere spletni iskalniki lažje evidentirajo.

Slabost jezikov na strani strežnika:

- vsako spremembo, ki zahteva posredovanje kode spletnega jezika, je potrebno zahtevati od strežnika prek URL-ja, ta pa vsakič znova pripravi novo stran s spremembami. Posledično je strežnik procesno in omrežno bolj obremenjen.

2.5.2 Skupni klasični elementi skriptnih programskih jezikov

Kljub temu, da obstaja mnogo programskih jezikov različnih namenov, vrst in generacij, so nekatere metode, postopki in elementi programiranja pri vseh podobni ali celo enaki. Navedimo nekaj klasičnih skupnih elementov in faz programiranja.

Spremenljivke so z identičnim imenom označeni sklopi shranjenih podatkov. Spremenljivko uvedemo z definiranjem. To je običajno prvo poimenovanje spremenljivke in pripis vrednosti, ki jo obdržijo shranjeno. Spremenljivke se običajno ločijo na globalne (veljajo na celotnem sklopu kode) ali lokalne (veljajo le za zaokrožen sklop kode, na primer v funkciji, kjer je bila spremenljivka definirana). Spremenljivke se običajno ločijo tudi po tipu pripisanega podatka (npr. tekst, celo število, realno število, datum, itd...).

Spremenljivke nosijo shranjeno vrednost od takrat, ko ji jo določimo, pa vse dokler je ne spremenimo. Če spremenljivke ne spreminjamo več, le ta ohranja shranjeno vrednost, dokler se koda celotnega dokumenta ne izvede (npr. PHP) ali dokler ne prekinemo stanja pripravljenosti (npr. JavaScript, ko zapustimo spletno stran).

Spremenljivke z indeksi so spremenljivke, ki lahko shranijo več vrednosti (teoretično neskončno mnogo). Spremenljivke z indeksi si lahko predstavljamo kot vektorje in matrike poljubnih dimenzij.

Funkcije so določena zaokrožena zaporedja kode, ki jih sprožimo ob klicu funkcije. Funkcije se poimenuje z identičnim imenom, s katerim se jih tudi kliče. Nekatere funkcije so že vgrajene v programski jezik, lahko pa definiramo svoje lastne. Funkcije imajo lahko določene vhodne in izhodne spremenljivke. Splošen primer funkcije vsote dveh števil:

```
funkcija vsota(a,b)
{ c = a + b
vrni c}
```

Zgornja funkcija ima dve vhodni spremenljivki, za kateri izračuna vsoto. Le to vrne kot neko tretjo spremenljivko. Tako funkcijo lahko kličemo takole:

```
d=2
e=3
seštevek = vsota(d,e)
```

V spremenljivko »seštevek« želimo shraniti vsoto števil 2 in 3. Funkcija izračuna vsoto, vrednost vrnjene spremenljivke pa se pripiše spremenljivki »Seštevek«.

Zanke omogočajo izvedbo enake kode v več ponovitvah (iteracijah). To je uporabno takrat, ko imamo seznam več vrednosti, nad katerimi želimo izvesti enak postopek. Zanko uporabimo tudi, če želimo v seznamu najti določeno vrednost, ki ustreza podanemu pogoju in v tem primeru lahko zanko tudi prekinemo.

If stavek omogoča logično odločanje glede postavljenih pogojev. Uporabimo ga za razlikovanje stanj ali vrednosti in pogosto pomeni vozlišče v zaporedju kode (glede na ugotovljeno stanje izvedemo različno zaporedje kode z različnimi funkcijami). Primer splošnega if stavka:

```
If (rezultat < 50)
{ocena = negativno}
Else
{ocena = pozitivno}
```

Zgornji if stavek glede na vrednost rezultata spremenljivki »ocena« pripiše ustrezno vrednost.
Komentarji so posebej označeni deli teksta, ki pomagajo razvijalcu pri razvoju kode z namigi in povečujejo preglednost. Komentarji so označeni s posebnimi znaki, katerih vsebina je pri izvedbi kode spregledana. S komentarji lahko opišemo manj jasne vrstice kode, opišemo na novo sestavljeno funkcijo ali pa pregledno ločujemo zaokrožene sklope kode.

2.5.3 PHP

PHP je skriptni jezik s širokimi možnostimi uporabe. Še posebej pa je primeren in uporabljan za razvoj spletnih strani z vključevanjem v HTML. Precej se uporablja v strežniških okoljih in za izdelavo dinamičnih spletnih strani. Deluje na strani strežnika. Strežnik po PHP-navodilih sestavi HTML-obliko spletne strani in jo posreduje uporabniku.

PHP je razvil kanadčan Rasmus Lerdorf, ki je želel dopolniti nekaj skript za preprost dostop do podatkovnih zbirk. V začetku se je PHP imenoval še PHP/FI (Personal Home Pages/Form Interpreter), ki izhaja iz namena avtorja po predstavitvi in izmenjavi osebnih podatkov. Nova kratica PHP danes pomeni »Hypertext Preprocessor«, torej neke vrste pretvornik hiperteksta. Pripravljenih je bilo že več različic PHP-ja, zadnja stabilna verzija nosi oznako 5.3.2 (julij 2011).

Za PHP velja, da ima zelo podobno sintakso in semantiko kot JavaScript. Zato se je, s predznanjem JavaScript jezika, jezik PHP relativno lahko naučiti. PHP uporablja dinamično tipizacijo spremenljivk ravno tako kot JavaScript. Tip spremenljivke je avtomatsko določen že z določitvijo same vrednosti spremenljivki. Tako kot JavaScript je PHP veliko bolj obvladljiv kot večina drugih podobnih jezikov. K temu pripomorejo tako avtomatsko tipizacija spremenljivk, kot tudi delo z tekstovnimi nizi (strings) in spremenljivkami z indeksi (arrays). Spremenljivke z indeksi v PHP-ju so kombinacija podobnih programskih jezikov in asociativnih spremenljivk z indeksi. PHP premore tudi bogat izbor funkcij za delo s spremenljivkami z indeksi. PHP podpira tako proceduralno kot objektno programiranje. Tako kot pri JavaScriptu tudi pri PHP-jeziku za pripravo in uporabo kode ne potrebujemo plačljivih aplikacij. (Sebesta, 2010)

PHP-kodo uporabimo tako, da jo vgradimo neposredno v kodo HTML. Kodo vgnezdimo v oznako

```
<?php ... ?>
```

V tem primeru moramo vrsto datoteke iz ».html« spremeniti v ».php«. Zahtevana datoteka se posledično ne pošlje uporabniku takoj, ampak le to prej obdela PHP-procesor. Če v njej odkrije PHP-kodo jo inpretira in izvede, rezultate pa kot HTML-zapis pošlje uporabniku.

Spremenljivke se v PHP označi z enoličnim imenom, pred katerega se postavi dolarski znak »$«. Tip spremenljivke se nastavi avtomatsko z definiranjem vrednosti spremenljivki:

```
$sprem_tekst = 'poljubno besedilo';
$stevilo = 3;
```

Spremenljivke z indeksi oz. polja se v PHP-ju definirajo z izrazom »**array**«. Primer definiranja in klicanja polja:

```
$niz = array('ena','dva','tri');
$niz[2];
```

V prvem primeru je polje definirano, v drugem pa je klicano tretje mesto v polju. Rezultat je besedilo »tri«.

Funkcije se v PHP-jeziku označuje na klasičen način:

```
function plus2($parameter)
{
$rezultat = $parameter + 2;
echo $rezultat;
}
```

V zgornjem primeru je definirana kratka funkcija. Klicali bi jo na naslednji način:

```
$stevilo = 3;
plus2($stevilo);
```

Klic zgornje funkcije bi v HTML-dokument zapisal število 5.

Zanke se v PHP-jeziku formira na naslednji način:

```
$niz = array(1,2,3);
foreach( $niz as $array1){
      echo $array1;
}
```

Zgornja zanka bi v HTML-dokument izpisala celotno vsebino polja $niz.
If stavek tvorimo na naslednji način:

```
if ($dan=="petek")
{echo "Imejte lep vikend!";}
```

Zgornji stavek bi v HTML-dokument izpisal vpisano besedilo, če je spremenljivka $dan enaka besedilu »petek«. Opazimo operator sestavljen iz dveh enačajev. Ta operator se uporablja v primeru primerjave dveh vrednosti. Za določitev vrednosti spremenljivki se uporablja enojni enačaj. Pri primerjavah pa se uporablja tudi trojni enačaj »===«, ko se poleg primerjave vrednosti primerja tudi tip spremenljivke.

Komentarje v php-ju označimo z dvojnima poševnicama »//« za eno vrstico ali z začetnim »/*« in končnim »*/«, pri čemer se kot komentar upošteva vse med začetnim in končnim znakom:

```
$sprem = 3; //spremenljivki smo definirali vrednost 3
$sprem = 3; /*
```

```
$sprem = 2;
Rezultat spremenljivke bo še vedno tri, saj bo PHP procesor vse med znakoma
za komentar spregledal!
*/
```

2.5.4 JavaScript

JavaScript je objektni skriptni programski jezik. Kljub temu, da se uporablja tudi izven spletnega okolja in kot jezik na strani strežnika, je JavaScript najbolj poznan po svojem osnovnem namenu – skrbi za dinamičnost HTML-spletnih strani s pomočjo brskalnika in na uporabnikovem računalniku. Deluje torej na strani uporabnika. Glavna lastnost je torej iterakcija z brskalnikom (uporabnikom) in neposredno umeščanje v HTML-zapis. Lastnost jezika je tudi, da ne podpira objektnega načina programiranja in ne pozna razredov.

JavaScript je razvil Netscape in se je najprej imenoval LiveScript. Leta 1995 so ga preimenovali v današnje poimenovanje. Jezik je v tistih letih precej napredoval in konec devetdesetih je izšel standard ECMA-262 (European Computer Manufacturers Association) za ta jezik. Jezik je potrdila tudi ISO-organizacija s standardom ISO-16262. Uradno se JavaScript po standardu imenuje ECMAScript, vendar ta izraz ni pogosto uporabljen.

Čeprav je bil JavaScript predviden kot programersko orodje tako na strani uprabnika kot strežnika, se je uveljavil le del na strani uporabnika. »Strežniški« del jezika se uporablja bistveno redkeje. Del JavaScript na strani uporabnika sicer omogoča izvajanje nekaj strežniških nalog, vendar ne more nadomestiti dela na strani strežnika. Strežniški del JavaScript na primer omogoča delo z datotekami in bazami podatkov, medtem ko uporabniški del tega ne zmore. Prednost JavaScripta je tudi ta, da se je mogoče jezik relativno hitro naučiti. Lastnost JavaScripta je tudi dobra podpora miškinih ukazov, kar pripomore k interakciji z uporabnikom. JavaScript tako zaznava tudi lego kurzorja in elementom na tisti legi sočasno določa ustrezne lastnosti.

Sposobnost JavaScripta se je izboljšala tudi z uvedbo tehnologije DOM (Document Object Model), ki omogoča spreminjanje CSS-lastnosti in vsebine kateregakoli HTML-elementa v dokumentu in vse to bistveno pripomore k dinamičnosti spletne strani. JavaScript je večinoma vezan na neke dogodke na strani (klik miške, gumba, povezave...), ki jih sproži uporabnik, kar tudi povečuje interaktivnost med uporabnikom in spletno stranjo.

JavaScript izvaja naloge, ki jih opišemo v izjavah oz. vrsticah kode. Kodo lahko v HTML-datoteko umestimo na dva načina. Koda je lahko zapisana kot del HTML-dokumenta. JavaScript kodo od HTML-zapisa ločimo z začetno in končno <script> značko:

```
<script type="text/JavaScript">
alert(»to je koda JavaScript«);
</script>
```

Ta pristop ima nekaj pomankljivosti:

- zapis več pomensko različnih sklopov JavaScript kode zmanjšuje preglednost dokumenta.
- Če HTML-kodo in JavaScript razvijata različni osebi, težko svoja rezultata usklajujeta v enem dokumentu.

Da se temu izognemo, lahko JavaScript-kodo shranimo v ločeno datoteko s končnico ».js« in jo v kodo vključimo z uporabo značke <script>, ki ima za lastnost definirano povezavo na JavaScript datoteko:

```
<script src="openlayers/OpenLayers.js" type="text/JavaScript"></script>
```

Lastnost tega pristopa je tudi, da je JavaScript koda posredno skrita in ni vidna v kodi osnovnega HTML-dokumenta (oz. spletne strani).

Eksplicitno JavaScript kodo običajno umeščamo v glavo HTML-dokumenta. Tja sodi koda, ki je pomembna samo ob uvodnem prikazu dokumenta, ali pa koda, ki je lahko klicana od drugod. V primeru, da želimo JavaScript kodo sprožiti med samo uporabo dokumenta, to proženje omogočimo z zapisom JavaScript tudi znotraj HTML-vsebine (znotraj body elementa). To običajno storimo z definiranjem lastnosti HTML-elementa:

```
<a href="JavaScript:karta()">karta</a>
```

Zgoraj je primer HTML-elementa - povezave, ki ima znoraj svoje lastnosti zapisano JavaScript kodo. Ob kliku na HTML-povezavo se sproži JavaScript funkcija z imenom »karta«. Kodo funkcije bi v eksplicitnem primeru umestili v glavo dokumenta, v implicitnem pa znotraj zunanje JavaScript datoteke.

JavaScript prepoznava HTML-elemente kot objekte. Objekti imajo svoje lastnosti, metode in »rokovalnike dogodkov« (ang. Event handlers). Tipične lastnosti objektov so npr. barva, višina in širina, itd. Metode so vnaprej definirane procedure, ki manipulirajo s pripadajočim objektom. Rokovalniki dogodkov pa se sprožijo, če je objekt na primer kliknjen, ali pozvan na drug način.

```
<button onclick=«karta()«>karta</button>
```

Zgoraj je primer HTML-gumba, ki ima definiran rokovalnik dogodkov. Ob kliku na gumb se bo sprožila v prejšnjem primeru definirana funkcija »karta«.

Spremenljivke se v JavaScript ustvarijo z ukazom »var«:

```
var spremenljivka;
```

Lahko jim tudi takoj pripišemo vrednost:

```
var spremenljivka = 3;
```

Možno je tudi ustvariti spremenljivko direktno s pripisom vrednosti brez ukaza:

```
spremenljivka = 3;
```

V tem primeru je spremenljivka ustvarjena samodejno.

JavaScript ločuje med globalnimi in lokalnimi spremenljivkami. Če spremenljivko ustvarimo znotraj funkcije, bo ta lokalna. Če želimo ustvariti globalno spremenljivko, moramo to storiti izven funkcij.

Za tvorbo oznak spremenljivk moramo upoštevati dvoje pravil:

- JavaScript razlikuje velike in male črke v imenu (x in X sta dve različni spremenljivki),
- spremenljivka se mora začeti s črko ali podčrtajem.

Spremenljivke z indeksi se definira zelo podobno kot v PHP-jeziku:

```
var sprem_ind = new Array('ena', 'dve', 'tri');
```

Iz zgornje spremenljivke drugo mesto kličemo kot:

```
stevilo = sprem_ind[1];
```

Tudi z **funkcijami** operiramo praktično enako kot v PHP-jeziku:

```
Function karta()
{
Alert(»karta ni na voljo«);
}
```

S klicem funkcije, bi se pojavilo prikazno okno z definiranim besedilom.

Zanke v JavaScript se uporabi na naslednji način:

```
for (i=0;i<=5;i++)
{
document.write("Število je: " + i);
}
```

Zgornja zanka teče od 0 (i=0) do 5 (i<=5), s prištetjem ene enote (i++). Pri zankah je možno uporabiti tudi ukaz »while«, pri čemer zanka teče, dokler je izpolnjen nek pogoj. Zanko je možno prekiniti tudi znotraj nje same z ukazom »break«.

If stavek se tvori na podoben način kot pri PHP:

```
if (stevilo<10)
  {
  document.write("Število je manjše od deset!");
  }
```

Komentarje v JavaScript se označuje enako kot v PHP, z dvema poševnima črtama »//« za eno vrstico oz. začetnim znakom »/*« in končnim »*/« za več vrstic skupaj.

2.6 DOM (Document Object Model)

DOM je aplikacijski programski vmesnik (API – Application Programming interface), ki omogoča povezavo med HTML-dokumenti in aplikacijami. Gre za abstrakten model, saj mora ustrezati vrsti programskih jezikov. DOM pri prikazovanju HTML-dokumentov ne igra nobene vloge. Pomemben je za povezavo JavaScript jezika z HTML-elementi, kar omogoči dinamičnost spletne strani. Prek DOM je možno z JavaScript ustvarjati HTML-dokumente, spreminjati njihovo strukturo ter spreminjati, dodajati ali brisati HTML-elemente in njihovo vsebino.

DOM razvija organizacija W3C že od sredine devetdesetih let prejšnjega stoletja. Aktualna verzija je DOM 3. Osnovna motivacija za izdelavo je bila izdelati specifikacijo, ki bi skriptnim jezikom, kot je JavaScript, omogočila dostop do HTML-kode v različnih brskalnikih. Takoimenovana DOM 0 verzija se je že pojavila v brskalnikih Netscape 3.0 in Internet Explorer 3.0, čeprav to še ni bila specifikacija. Prva specifikacija je izšla leta 1998 in se je osredotočila na HTML in XML-jezike. Verzija DOM 2, ki je izšla leta 2000, je omogočila še dostop do stilskih predlog. DOM 3, ki je izšel v 2004 obravnava predvsem XML-dokumente.

Zanimivo je, da DOM 0 podpirajo vsi brskalniki, ki podpirajo tudi JavaScript. DOM 2 je skoraj v celoti podpiral že brskalnik Firefox 2, na drugi strani pa se Internet Explorer 7 pri nekaterih elementih DOM 2 bodisi ne drži standardov ali pa nekaterih elementov sploh ne omogoča. Verzija DOM 3 za splet ne prinaša bistvenih novosti, oziroma se jih načeloma ne uporablja.

2.7 AJAX

Cilj Ajax tehnologije (oz. skupine spletnih metod) je omogočiti spletnim aplikacijo tisto stopnjo interakcije, ki jo uporabnikom omogočajo že namizne aplikacije. Motivacija za to je bilo veliko zanimanje po bogatih spletnih aplikacijah (RIA – Rich Internet Applications). Te aplikacije uporabljajo vmesnik, ki zahteva pogoste interakcije med uporabnikom in strežnikom. Hitrost teh interakcij pa določa uporabnost teh aplikacij. (Sebesta, 2010)

Pred uporabo Ajax-a je uporabnik poslal zahtevo strežniku prek URL. Strežnik je glede na zahtevo pripravil stran in jo poslal uporabniku. Za morebitno spremembo na spletni strani se je brskalnik »zaklenil« za uporabo in zahteva je bila zopet poslana strežniku in stran se je v celoti osvežila. Uporabnik je moral počakati, da se je ta postopek izvršil do konca. Z uvedbo Ajax-a, pa so omogočene asinhrone zahteve strežniku brez osvežitve celotne strani in brez uporabnikovega čakanja, ki lahko tekom procesa povezave s strežnikom še vedno nemoteno spremlja spletno stran. Ker z Ajax zahtevo običajno spreminjamo le majhen del spletne strani, lahko strežnik veliko hitreje vrne zahtevano vsebino, kar prida občutek hitre odzivnosti tehnologije.

Ideja za Ajax je nastala z nastankom HTML-elementa »iframe«, ki znotraj prikazanega HTML-dokumenta prikaže drug HTML-dokument v predpisanem okvirju. Sčasoma so prišli do ideje, da širino in višino okna nastavijo na nič, zahtevek po vsebini strežniku pa ostane. Rešitev pa vseeno ni bila najbolj elegantna, čeprav je delovala. Microsoft je potem uvedel ActiveX komponente v Internet Explorer 5, ki so predstavljale asinhrone zahtevke strežniku med samo uporabo spletne strani. Te rešitve so danes združene v obliki »XMLHttpRequest« objektu, ki je del večine brskalnikov in se precej uporablja.

Nekaj razvijalcev je pred letom 2005 že uporabljalo Ajax, a tehnologija do tega leta še ni vzbujala širšega zanimanja. Tega leta pa je Google predstavil svojo novo storitev »Google Maps«, ki je temeljila na Ajax tehnologiji. Google Maps prikazuje karto z hitro zamenjavo pravokotnih delčkov karte, katere pridobi z asinhronimi zahtevami strežniku. To omogoča navigacijo uporabniku po karti, ne da bi se naložila celotna karta znova. Aplikacija je tako vzbudila veliko zanimanja za tehnologijo Ajax. Poleg tega je v istem letu Ajax tudi dobil svoje sedanje ime, kar naj bi ravno tako pripomoglo k zanimanju za tehnologijo. (Sebesta, 2010) Množična interaktivnost se je tako pravzaprav začela prav zaradi interaktivnih spletnih kartografskih prikazov.

3 SPLETNI FORMATI IN SPLETNI SERVISI PROSTORSKIH PODATKOV

Razvoj informacijske tehnologije (IT) v zadnjih desetletjih ni zaobšel slehernega področja in vede, še več, razvoj IT pogojuje razvoj marsikaterega področja. Tudi široko področje upravljanja in načrtovanja rabe prostora je z IT pridobila nove, prej neslutene možnosti. Koraki v razvoju IT v povezavi s prostorskimi vedami so bili veliki, a hitri. Od prvih rastrskih GIS-aplikacij, ki so se pojavile že relativno zgodaj, prek baz podatkov in CAD-aplikacij, do nastanka zmogljivih GIS-aplikacij. Tekom razvoja aplikacij so se razvijale tudi nove tehnologije zapisovanja in shranjevanja prostorskih podatkov. Uveljavili sta se dve tehnologiji obdelave prostorskih podatkov – rastrska in vektorska.

Gonilna sila razvoja področja je bila pogosto vojska, ki je zaznala pomen razvoja IT zelo zgodaj. Novodognane tehnologije so počasi prehajale v civilno sfero in te so sprožale podjetniško pobudo med podjetji. Nekateri izmed teh so čez čas postali pomembni ponudniki tovrstne tehnologije in storitev. Le ti so tekom razvoja GIS-področja razvili nove zmogljive aplikacije in te so potrebovale na novo razvite formate podatkov. Pomemben dejavnik je bil tudi razvoj interneta. Na ta način dosežena dostopnost ter povezljivost med aplikacijami in bazami podatkov je še bolj kot sicer vzbudila potrebo po standardizaciji prostorskih podatkov. Rezultat tega je bilo uvajanje različnih standardov za prostorske podatke. Za formalno standardizacijo skrbijo organizacije za standardizacijo kot so ISO, CEN, ANSI, slovenska SIST, itd... Že zelo zgodaj pa so potrebo po usklajevanju začutili tako uporabniki kot proizvajalci, ki so že v začetku devetdesetih ustanovili organizacijo »OGC«.

3.1 Tehnologija OpenGIS

Leta 1994 je dozorela ideja o standardu, ki bi poskrbel za povezljivost in medopravilnost različnih prostorskih aplikacij. Zagnali so projekt OpenGIS, v okviru katerega so pričeli iskati medopravilne rešitve. Projekt je še istega leta združil sedem neprofitnih organizacij in podjetje Intergraph v organizacijo »Open GIS Consortium«, ki so jo v letu 2004 preimenovali v »Open Geospatial Consortium« (OGC) . OGC je mednarodna neprofitna organizacija oziroma industrijsko združenje, ki danes združuje več kot 400 podjetij, strokovnih, vladnih in raziskovalnih organizacij ter akademskih ustanov.

Razvojni program OpenGIS se deli na tehnološki in medopravilni razvojni program. Tehnološki razvojni program predstavlja zlasti razvoj raznih splošnih softverskih in izvedbenih rešitev za orodja GIS. Medopravilni razvojni program pa spremlja razvojne pobude, povratne odzive na specifikacije in testira orodja GIS glede skladnosti z OpenGIS rešitvami. Program OpenGIS se je izkazal za tehnološko zelo obsežen in v izvedbenem smislu zahteven, zato se je precej zavlekel. (Šumrada)

Rezultat OpenGIS razvojnega programa je OpenGIS tehnologija, ki je »odprta tehnologija, ki omogoča izmenljivo obdelavo prostorskih podatkov (geoprocesiranje) ter podpira sposobnost pregledne deljivosti različnih prostorskih podatkov in drugih virov v porazdeljenem omrežnem in zlasti medmrežnem (internet) okolju.« (Šumrada, 2005, str.207)

3.2 Spletni servisi OGC

Geografski spletni servis je aplikacija, ki s pomočjo spletnega strežnika omogoča dostop uporabnikom oz. njihovim aplikacijam do prostorskih podatkov. Omogoča tudi dostop do izvedene izbrane obdelave prostorskih podatkov, in sicer bodisi opisnih, položajnih, ali časovnih, oziroma ustrezno kombinirano sestavo navedenih podatkovnih oblik. (Šumrada, 2005). Uporabniki pošiljajo zahtevo enako, kot je to običaj v spletnem okolju, prek URL zahteve. Tako kot pri običajni spletni uporabi, je tudi v tem primeru najprej naveden URL naslov ali domena, kjer se strežnik nahaja, namesto HTML-datotek pa so navedeni parametri, na podlagi katerih se oblikuje natančna zahteva za strežnik. OGC je standardiziral naslednje vrste spletnih servisov:

WMS (Web Map Service) je servis, ki uporabniku na podlagi poslanih parametrov oblikuje karto in mu jo vrne v določenem formatu. Prva verzija 1.0 je bila izdana leta 2000, naslednja 1.1 leto kasneje, v januarju leta 2002 pa je izšla verzija 1.1.1, najnovejša verzija 1.3 pa je na voljo od začetka leta 2004.

WMS predvideva dva obvezna zahtevka (GetCapabilities in GetMap) ter tri neobvezne (GetFeatureInfo, DescribeLayer in GetLegendGraphic). Prvi zahtevek, ki ga bo uporabnik najverjetneje uporabil, je »GetCapabilities«. HTTP-zahtevek, katerega del je WMS-zahtevek, se oblikuje na naslednji način:

```
http://www.vzorec.si/wms?
service=wms&
version=1.1.1&
request=GetCapabilities
```

HTTP-zahtevek vsebuje lokacijo strežnika (http://www.vzorec.si /wms), vprašaju pa, kot je to določeno pri tvorbi URL, sledijo parametri namenjeni strežniku. V tem primeru so to trije parametri, ki določajo vrsto storitve (wms), verzijo (1.1.1) in WMS-zahtevek (GetCapabilities).

Na to zahtevo uporabnik dobi odgovor v XML-formatu z opisom strežnika ter vseh slojev in možnimi parametri, s katerimi uporabnik oblikuje zahtevo.

V primeru, da zahteva ni zasnovana povsem pravilno in jo strežnik ne razume, le ta odgovori z XML-datoteko v kateri je obrazložena napaka.

Naslednji pomemben zahtevek WMS-servisa je »GetMap«. Na podlagi tega zahtevka strežnik pripravi karto v obsegu, obliki in formatu, ki ga je zahteval uporabnik v zahtevku. Primer HTTP-zahtevka z vključenim WMS-zahtevkom:

```
http://www.vzorec.si/ wms?bbox=-130,24,-
66,50&styles=population&Format=image/png&request=GetMap&layers=topp:states&
width=550&height=250&srs=EPSG:4326
```

V HTTP-zahtevku so navedeni parametri o obsegu karte, stilu, formatu, velikosti in koordinatnem sistemu zahtevanega sloja. Kot odgovor uporabnik prejme rastrsko podobo navedenih dimenzij v projekciji, ki je navedena v parametru »srs«.

Poleg teh dveh osnovnih zahtevkov, ki sta obvezna, se neobvezno lahko vključijo še zahtevki »GetFeatureInfo«, »DescribeLayer«, in »GetLegendGraphic«. »GetFeatureInfo« zahtevek

smo uporabili tudi v okviru te naloge, ko smo za klik na določen element prikazanega sloja zahtevali njegove atributre. Ko strežnik odgovori na zahtevek, se odgovor prikaže v prikaznem oknu karte.

WFS (Web Feature Service) in **WCS (Web Coverage Service)** sta v nasprotju z WMS, uporabniško usmerjeni storitvi. V primeru WMS končni izgled karte izdela že strežnik. V primeru WFS in WCS pa uporabnik zahteva po surovih vektorskih in rastrskih podatkih, ki jih bo sam oblikoval. Storitvi tako ob prenosu ne vsebujeta podatka o oblikovanju in sta v tem primeru podobna na primer shape datoteki.

Glavni trije zahtevki WFS so:

- »Get Capabilities«,
- »Describe Feature Type«,
- »Get Feature«.

Zahtevki imajo podobno vlogo, kot v primeru WMS.

3.3 Spletni prostorski podatkovni formati

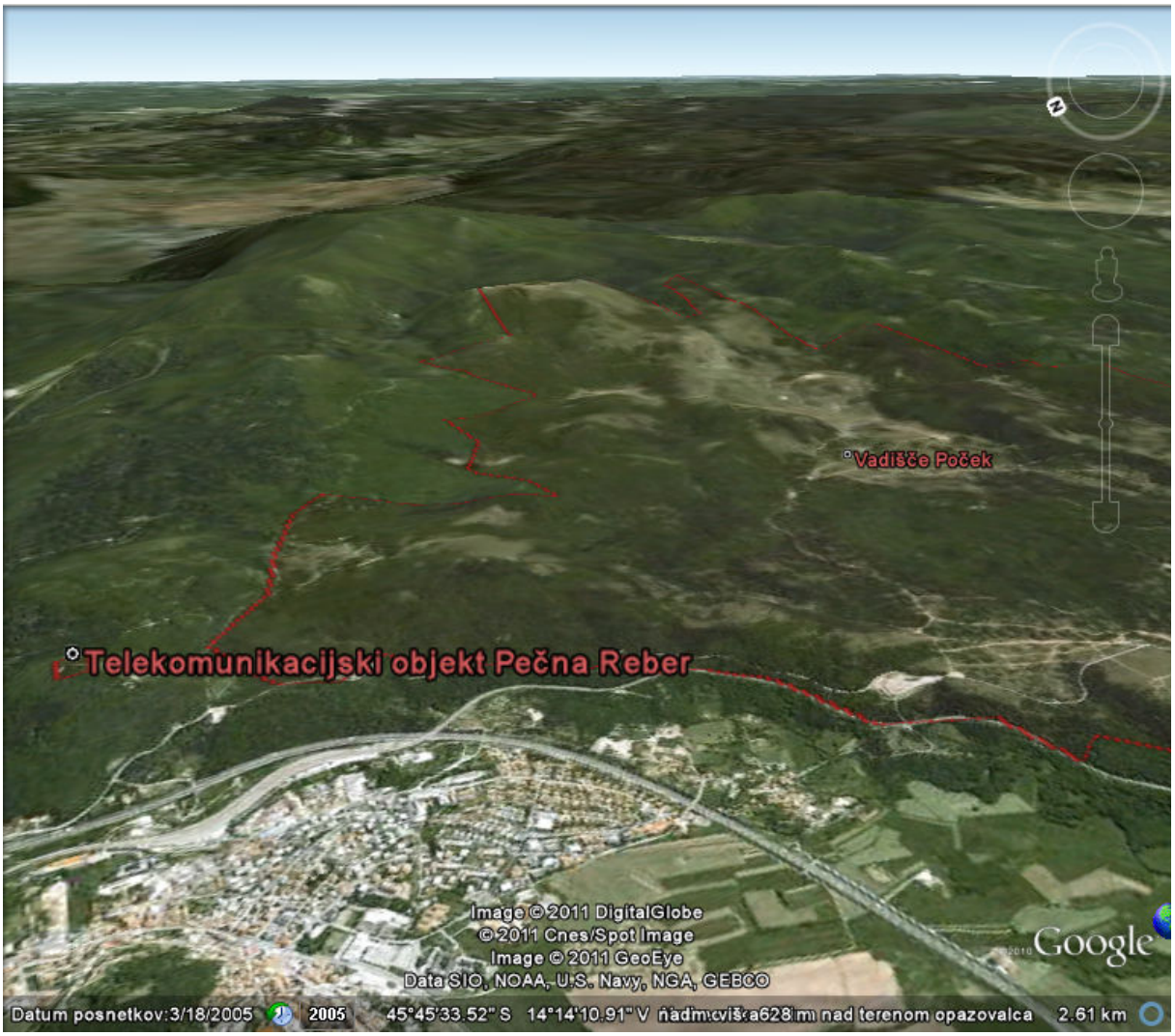
3.3.1 GML

GML (Geography Markup Language) je standard OGC in ISO in predstavlja odprt izmenjevalni format prostorskih podatkov. Tekom obstoja GML so razvili več verzij formata. Verziji GML 1.0 in GML 2.0 temeljita na osnovi treh glavnih gradnikov - točki, liniji in poligonu. Z verzijo GML 3.0 je dodana možnost vključitve rastrskih slojev, kot so na primer satelitski posnetki površja. Zadnja potrjena verzija je GML 3.2.1, ki je bila potrjena v letu 2007, v teku pa je razvoj verzij GML 3.3 in GML 4.

Sprva je GML model temeljil na RDF (Resource Description Framework) specifikacijah, kasneje pa je OGC v GML strukturo uvedel XML-shemo, kar je omogočilo lažje kodiranje prostorskih podatkov iz najrazličnejših področij. Različne organizacije uporabljajo sklope prostorskih podatkov, ki so omejene in prilagojene določeni panogi ali tematskemu področju. Te organizacije (oz. katerikoli uporabnik) lahko v okviru GML-jezika ustvarijo svojo GML-shemo prilagojeno svojemu področju (ang. application schema). Tako nastane nov GML-format, ki temelji na osnovni strukturi GML, a vsebuje lastne definirane elemente in strukturo.

3.3.2 KML

Pot KML (Keyhole Markup Language) je nekoliko drugačna od poti ostalih prostorskih standardiziranih formatov. Bil je razvit znotraj podjetja Keyhole, ki ga je v letu 2004 prevzel Google. Google je format uporabil v svojih spletnih prostorskih vsebinah, predvsem v aplikaciji Google Zemlja in povzročil širšo priljubljenost formata pri uporabnikih. Kot takega ga je kot standard potrdil tudi OGC. Tudi KML temelji na XML, podpira pa zgolj projekcijo EPSG:4326.



Slika 1: Prikaz objektov Slovenske vojske v formatu KML v aplikaciji Google Zemlja

3.3.3 SLD

SLD (Styled Layer Descriptor) je standard OGC za oblikovanje slojev pri WMS servisih. Gre za oblikovno razširitev standarda WMS, ki omogoča definiranje simbolov in oblikovanje posredovanih slojev. Definiran je kot XML-schema in ima sledeč izgled:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
  <StyledLayerDescriptor version="1.0.0"
xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescriptor.xsd"
xmlns="http://www.opengis.net/sld" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <NamedLayer>
   <Name>Point as graphic</Name>
   <UserStyle>
   <FeatureTypeStyle>
  <Rule>
  <PointSymbolizer>
  <Graphic>
 <ExternalGraphic>
  <OnlineResource xlink:type="simple"
xlink:href="policija.png" />
```

```
<Format>image/png</Format>
</ExternalGraphic>
<Size>32</Size>
</Graphic>
</PointSymbolizer>
</Rule>
</FeatureTypeStyle>
</UserStyle>
</NamedLayer>
</StyledLayerDescriptor>
```

SLD za oblikovanje podpirata odprtokodna prostroska strežnika GeoServer in Mapserver, in nekateri drugih programi.

4 UPORABLJENE RAČUNALNIŠKE APLIKACIJE

4.1 Brezplačne aplikacije

Brezplačne (freeware), pa tudi odprtokodne (opensource) aplikacije so zanimiv fenomen sodobnega sveta. V času, ko je materializem in navezanost na denar na žalost visoka človeška vrednota, so gibanja, ko skupina navdušencev izdeluje nekaj, za kar ne bodo zahtevali denarnega povračila, res nenavaden, a dobrodošel pojav. Čeprav se pojav na nek način zdi utopičen in nerealen, pa deluje, saj so tovrstne aplikacije vse boljši nadomestek komercialnih rešitev in jih v mnogih primerih že prekosijo.

Če je še nekaj let nazaj veljalo, da so tovrstne aplikacije v nasprotju s plačljivimi aplikacijami, kjer ob plačilu zagotovijo tudi ustrezno podporo in razvoj, nezanesljive, se ta dejavnik vse bolj izboljšuje v korist brezplačnih aplikacij. Nekatere bolj dovršene brezplačne aplikacije so po odpravljenemu številu možnih napak že povsem izenačene s komercialnimi aplikacijami, podporne oddelke komercialnih aplikacij pa nadomeščajo podporne spletne strani in forumi, kjer uporabniki sporočajo morebitne napake, le te pa se tekom razvoja popravljajo in z izidom nove verzije odpravijo. K temu prispeva tudi koncept odprtokodnih aplikacij, kar je sicer ožji pojem, saj brezplačna aplikacija še ni nujno odprtokodna. Uporabnikom je v tem primeru omogočen vpogled v kodo aplikacije, tako da lahko tudi sami odkrivajo napake, izdelujejo izboljšave in nove funkcionalnosti, ki jih, v primeru njihove zanesljivosti in uporabnosti, vključijo že v novi naprednejši verziji aplikacije.

Uporaba brezplačnih programov z ekonomskega vidika lahko pomeni znižanje potrebnega začetnega kapitala nekega podjetja ali projekta, kar ob visoki ceni nekaterih zmogljivejših komercialnih aplikacij sploh ni nezanemarljivo. Čeprav uporaba takega koncepta predstavlja neko tveganje s strani podpore in zanesljivosti, ki ni zagotovljena, po drugi strani pa v primeru odprtokodnih aplikacij omogoča podjetju sodelovanje pri razvoju aplikacije z izdelavo funkcij prilagojenim prav njemu. Navsezadnje lahko ob uspešni uporabi aplikacije podjetje nadaljni razvoj aplikacije podpre tudi z donacijami skupnosti, ki bdi nad razvojem aplikacije.

4.2 Spletni strežniki

Spletni strežniki so programi, ki pošiljajo dokumente brskalnikom, ki so poslali zahtevo po njih. Strežniki so »suženjski« programi, saj se odzivajo le na zahteve brskalnikov z drugih računalnikov na internetu.

4.2.1 XAMPP

XAMPP je brezplačen odprtokodni namestitveni paket, ki ga sestavljajo naslednje najpopularnejše brezplačne strežniške aplikacije:

- Spletni strežnik **Apache HTTP,**
- SQL baza **MySQL,**
- Procesor **PHP** jezika,
- Podpora za **Pearl** jezik.

Naštete aplikacije lahko štejemo med nepogrešljive pripomočke pri strežniški podpori spletnih strani. Ker se pogosto uporabljajo skupaj, jih je bilo smiselno združiti v enoten paket. Brez XAMPP paketa bi bilo potrebno namestiti vsak program posebej in jih potem uskladiti med seboj. XAMPP pa omogoči namestitev programov v eni sami namestitvi, po kateri so vsi programi usklajeni in pripravljeni za uporabo.

4.2.2 Apache HTTP strežnik

Strežnik Apache je zagotovo najbolj poznan in uporabljen spletni strežnik, ki je odigral ključno vlogo tudi pri razvoju in rasti spletnega omrežja. Apache je bil prva alternativa Netscapovim spletnim strežnikom. Je odprtokodni strežnik, katerega razvija skupnost »Apache Software Foundation«. Strežnik je možno namestiti na veliko različnih operacijskih sistemov. Strežnik je že od leta 1996 najbolj uporabljen strežnik, trenutno pa zavzema dvotretjinski delež izmed vseh strežnikov v uporabi.

Razlog za njegovo popularnost je ta, da združuje tako hitrost kot zanesljivost. Je tudi odprtokodna aplikacija, je zastonj, za razvoj pa skrbijo prostovoljci. Apache omogoča precej širšo uporabnost, kot le osnovno pošiljanje dokumentov. Ko se Apache zažene, prebere konfiguracijske nastavitve iz datoteke in nastavi prebrane parametre. Te nastavitve so ob prvem zagonu nastavljene v osnovnem tipičnem načinu, katere je potem možno prilagoditi lastnim potrebam. Zaradi zgodovinskih razlogov ima Apache tri konfiguracijske datoteke: »http.conf«, »srm.conf« in »access.conf«.

4.2.3 GeoServer

GeoServer je odprtokodni strežnik, ki omogoča spletno prikazovanje in urejanje prostorskih podatkov. Kot skupinski prostovoljni projekt so ga izdelale različne skupine prostovoljcev in organizacij, napisan pa je v java jeziku. GeoServer podpira WMS (Web Map Service), WFS (Web Feature Service) in WCS (Web Coverage Service) standarde, ki jih je izdal OGC (Open Geospatial Consortium). Prek omenjenih storitev strežnik omogoča izdelavo kart v različnih izhodnih formatih. V GeoServer je integrirana tudi OpenLayers knjižnjica, ki omogoča prikaze dinamičnih kart. GeoServer je zgrajen na osnovi »Geotools« orodjih (odprtokodna GIS-orodja napisana v Java jeziku). GeoServer podpira prikaze vseh priljubljenih kartografskih ordojih, kot so Google Maps, Google Earth, Yahoo Maps in Microsoft Virtual

Earth. GeoServer je povezljiv tudi s tradicionalno GIS-arhitekturo (na primer podjetja ESRI), z različnimi bazami podatkov, itd.

4.3 JavaScript knjižnice

OpenLayers je JavaScript knjižnica za prikazovanje kartografskih podatkov. Deluje v večini modernih brskalnikih in omrežno deluje na strani uporabnika. Omogoča izdelavo dinamičnih kart na spletnih straneh. OpenLayers na podlagi lastnega JavaScript API (aplikacijski programski vmesnik) omogoča bogato spletno kartografsko aplikacijo podobno Google Maps in MSN Virtual Earth API s to razliko, da je OpenLayers knjižnica brezplačna.

jQuery je JavaScript knjižnica, ki olajšuje delo s HTML-dokumenti, animacijo in Ajax interakcijami pri delu z JavaScript. Knjižnica je bila razvita leta 2006 in je danes najbolj popularna JavaScript knjižnica. Omogoča tudi izdelavo vtičnikov, to je samostojnih JavaScript datotek, ki jih lahko vključimo v spletno stran. Na spletu je na voljo že precej raznovrstnih jQuery vtičnikov s področja animacije, dinamičnih prikazov podatkov, slikovnih galerij in mnogo drugih namenov.

4.4 Druge aplikacije

Zmogljivejša brezplačna aplikacija, ki je vredna omembe, je zagotovo **Quantum GIS** (QGIS). QGIS predstavlja zmogljivo GIS-aplikacijo, ki omogoča ustvarjanje in urejanje prostorskih podatkov. Podpira vrsto številnih prostorskih formatov kot so ESRI-jeve shape datoteke, izmenjevalni formati (GML/KML), postGIS, GPS formati, formati različnih programskih orodij, navezave na spletne servise, itd... Poleg osnovnih urejevalnih in oblikovalnih orodij združuje tudi vedno več analitično usmerjenih orodij. Vsebuje tudi knjižnico koordinatnih sistemov in omogoča transformacije med njimi.

Izdelavo QGIS je kot projekt izvedla organizacija Open Source Geospatial Fundation, ki pravno združuje gibanje za odprtokodno GIS-tehnologijo in podatke. QGIS je izdelan v C++ jeziku in omogoča uporabo lastnih vtičnikov zapisanih v C++ ali python jeziku. Prednost QGIS je predvsem v manjšem procesnem obremenjevanju računalnika v primerjavi s podobnimi komercialnimi aplikacijami. Posledično se lahko QGIS uporablja tudi na manj zmogljivih računalnikih. Procesno zahtevnejše opravila z na primer veliko sloji pa opravi hitreje.

Pri spletnem programiranju in oblikovanju so uporabne še številne druge brezplačne aplikacije. Na področju dela z programskimi jeziki in spletnimi formati je uporabna aplikacija Notepade++, ki z oblikovnim ločevanjem med deli kode izboljšuje preglednost kode, omogoča pa tudi različne funkcije, ki so uporabne pri delu z različnimi formati. Na področju grafike zagotovo ne moremo mimo programa Gimp, ki z bogato knjižnico grafičnih metod povsem zadovolji potrebe pri osnovnem spletnem oblikovanju.

5 O SPLETNIH STRANEH IN PORTALIH

5.1 Spletne strani in spletni portali

Težko je najti natančno definicijo za spletni portal. Delna razlaga bi lahko bila, da je spletni portal obsežnejša spletna stran oziroma sklop spletnih podstrani z zaokroženo vsebino. Spletni portal je lahko tudi stičišče obiskovalcev z podobnim zanimanjem in interesi. Nadalje je lahko spletni portal obsežnejši spletni servis novic, blogov, forumov, povezav, iskalnikov, itd. Zasledimo lahko tudi definicijo, da je spletni portal »obsežno spletno mesto ali servis, katerega namen je izhodišče vsakega sprehoda po spletu.« (Pavlič, 2007)

Z vidika urednika naj bi spletni portal omogočal tehnologijo, ki omogoča ločitev vsebine od uporabniškega umesnika. Portali naj bi omogočali uredniku lažje dodajanje in posodabljanje vsebin ter modularno zasnovo spletišča. Tovrstno tehnologijo sicer poznamo pod izrazom CMS (Content Management System), ali po slovensko sistem za upravljanje vsebin, ki omogoča urejanje in vzdrževanje spletnih vsebin brez znanja spletnih jezikov.

Glavna razlika med spletno stranjo in portalom naj bi bila v orientiranosti. Če je spletna stran namenjena predstavljanju neke organizacije, avtorja ali teme, ki definira njeno vsebino, naj bi se spletni portal prilagajal predvsem uporabnikom, ki bi sami imeli več nadzora nad tem, kaj želijo videti. Če ima spletna stran za vse uporabnike enako vsebino in obliko podanih informacij, je za spletni portal značilno, da lahko zadovolji željo po podobni vsebini na različne oblike. Do podobnih informacij se lahko uporabnik dokoplje po različnih poteh in izkustvih. Pomembna značilnost portalov je torej prilagodljivost uporabnikov, iz katere izhaja tudi interaktivnost samih vsebin.

Portal naj bi glede na spletno stran bistveno bolje poznal svoje uporabnike. Za portal so tako običajni registracije uporabnikov, pošiljanje rednih novic o portalu na elektronski naslov, ipd., s čemer uredniki spremljajo navade in želje svojih uporabnikov. Značilno je tudi spremljanje statistike obiskov, za katero je na spletu na voljo dovolj dobro zmogljive programske podpore. Ker so portali ves čas premišljeno vodeni, ažurni in prilagodljivi, so dolgoročno načeloma privabiti večje število stalnih obiskovalcev kot spletne strani. Taki portali so zato tudi tržno zanimivi, zato so zanje značilni tudi prostori za reklamiranje.

5.2 Različne vsebinske oblike portalov

Portale lahko glede na vsebino razdelimo na horizontalne in vertikalne portale. Horizontalni portali ciljajo na širše množice ljudi in ponujajo popularno vsebino, ki je blizu večini spletnih uprabnikov. Taki portali so tudi najbolj množično obiskani. Tak primer sta na primer portala 24ur.com in siol.net. Vsi tovrstni portali na prvi strani objavljajo najnovejše novice, z osrednjo glavno novico, ki mora biti čim bolj ažurna in hkrati dovolj zanimiva. Portali preko povezav v menijih vodijo globje v široko in raznovrstno ponudbo najrazličnejših vsebin, prikazov in storitev, ki zadovoljijo najširšo množico ljudi.

Vertikalni portali se po drugi strani osredotočijo na določeno tematiko in predstavljajo široko vsebinsko ponudbo glavne tematike. Omogočajo tematske novice, forume, klepetalnice, spletne trgovine, članke z nasveti in testi, itd. Primeri takih portalov so na primer bicikel.com, avtomobilizem.com, gradimo.com, itd.

Spletne portale lahko razdelimo tudi glede na aktivni in pasivni pristop upravljanja, kar pogojuje tudi ekonomske, tehnične in organizacijske lastnosti portala.

Aktivno usmerjeni portali so portali z aktualno tematiko, kot na primer portali z raznovrstnimi novicami. Tovrstna spletna stran zahteva neprestano posodabljanje spletne strani. Potrebuje

ekipo ljudi, ki pridobiva in posreduje podatke na splet praktično brez prestanka. Taka spletna stran zahteva dodaten uporabniški vmesnik, ki ga lahko uporabljajo tudi računalniški laiki, za redno objavljanje vsebine na portal. Tak portal pritegne več obiskovalcev in obiskovalci ga obiskujejo bolj pogosto, tudi večkrat dnevno. Tovrstni portali so bolj zanimivi za trženje, vendar prinašajo tudi več stroškov za delovanje. Zahtevajo tudi močnejše strežnike in povezavo za veliko množico obiskovalcev hkrati.

Nasprotje so pasivni portali, na primer portali z zgodovinsko tematiko. Glavna vsebina spletne strani so zgodovinske teme, ki se v času praktično ne spreminjajo. Taka spletna stran po vzpostavitvi ne potrebuje veliko posodabljanja, deluje lahko dalj časa brez kakršnega koli posega. Taka stran je stroškovno veliko cenejša, vendar privablja manjše število ljudi, torej je tudi tržni izplen manjši. Obiskujejo jo predvsem občasni naključni gostje, ki jih zanima konkretna tematika. Stran najdejo preko spletnega iskalnika ali povezave z druge spletne strani s podobno tematiko. Ni pa nujno, da so portali tovrstnih vsebin vsi pasivno usmrejeni. Tudi zgodovinski portal je lahko do neke mere aktivno usmerjen, če mu nekdo posveča dovolj pozornosti in vsebine.

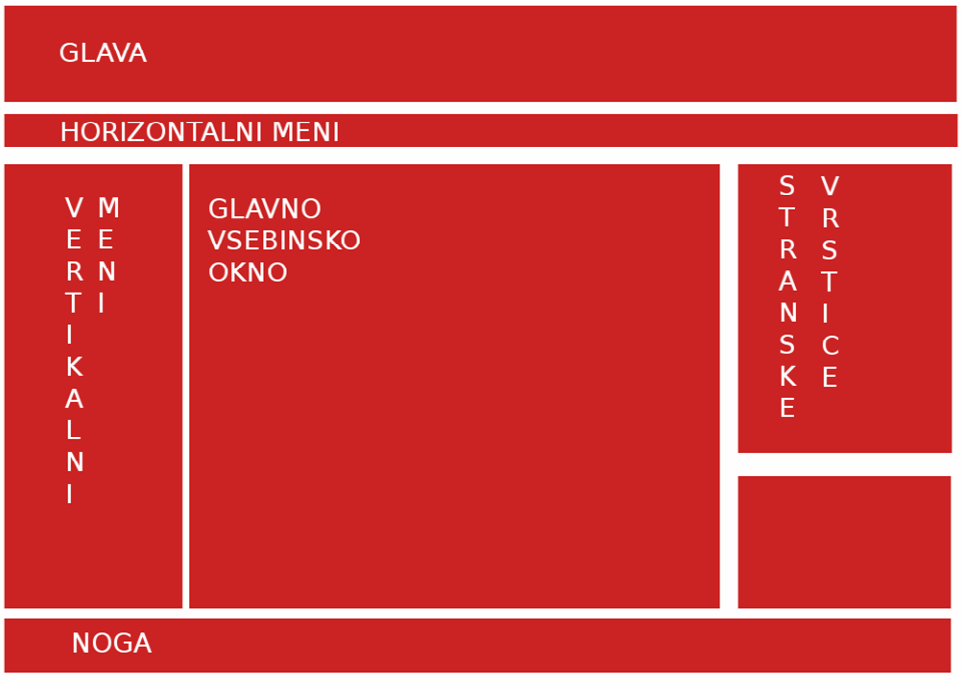
5.3 Elementi osnovne spletne strani

Ključen element spletne strani ali portala je uvodna spletna stran. Čeprav danes veliko obiskovalcev zaradi spletnih iskalnikov obišče spletni portal neposredno na podstran, ki vsebuje vsebino, po kateri je uporabnik poizvedoval, je uvodna stran še vedno ključen dejavnik, ali bo uporabnika pritegnil in bo na portalu ostal dalj časa in se na njega tudi vračal, ali pa ga bo slaba izkušnja za vedno odvrnila od portala.

Spletna stran ima nekatere značilne sestavne elemente:

- V **glavo** strani običajno umestimo naslov in logotip strani, v njej lahko najdejo svoje mesto tudi iskalnik po strani, gumbi za izbiro jezika, itd.
- Blizu glave je običajno **horizontalni meni**. Stran ima lahko namesto horizontalnega menija tudi **vertikalni meni**, običajno umeščen ob levi rob strani, lahko pa stran vsebuje oba menija. V meniju so predstavljene povezave na podstrani portala. Meni je lahko statičen ali dinamičen. Če dinamični meni prekrijemo z miško ali nanj kliknemo, se pokaže še več možnih podstrani, pri horizontalni menijih je pogost na primer viseči meni.
- V **nogi** strani se običajno navede podatke o lastniku ter izdelovalcu strani in avtorske pravice. Lahko se ponovi tudi meni v skrčeni obliki.
- **Stranske vrstice** so namenjene dinamičnim in slikovnim predstavitvam podatkov, opozorilom, različnim informacijskim vtičnikom za novice ali vreme, sponzorjem, itd.
- **Sponzorske ali oglasne vrstice** se pojavljajo med ostalimi elementi strani in so namenjene objavi oglasov. Ti pa so namenjeni zaslužku, oziroma pokrivanju stroškov portala. Sponzorske vrstice tako ne smejo biti preveč vsiljive in moteče, ker le te odvračajo uporabnike, hkrati pa ne smejo biti preveč skrite, da vzbudijo uporabnikom interes.
- **Glavno vsebinsko okno** ima glede na zgoraj naštete elemente sredinsko vlogo. V klasični obliki strani leži njegov levi zgornji kot pod glavo in horizontalnim menijem

ter desno od vertikalnega menija ali stranske vrstice. Tam se koncentrira največ uporabnikove pozornosti, saj se začenja osrednja vsebina strani.



Slika 2: Shema klasične postavitve značilnih elementov spletne strani

5.4 Pogoji za uporabnost in marketinški potencial

V tehničnem smislu je pomemben pogoj za uporabnost zmogljiv strežnik in zmogljiva medmrežna povezava. V oblikovnem smislu pa je pomembno uporabljati ustrezne barvne kombinacije, upoštevati vidnost črk (velikost, oblika, barva, ozadje...), saj neprijetna oblikovanost vsebine uporabnika hitro odvrne. Spletna stran ne sme dajati občutka zastarelosti, kot na primer stran zgolj z osnovnimi HTML elementi. Stran mora vsebovati moderne prijeme, kar jo naredi privlačnejšo in ažurnejšo, a hkrati mora biti enostavna za uporabo, brez odvečnih dinamičnih funkcij, ki so same sebi v namen.

Pomembno je, da izdelovalec strani upošteva potrebe uporabnika. Lahko se vživimo v vlogo uporabnika, ki išče točno določeno informacijo. Na ta način lahko ugotovimo, ali mu osnovna spletna stran sploh omogoča logično in hitro iskanje skritih podstrani in podobno. Priporočljivo je, da si spletno stran ogledamo z nekom, ki jo vidi prvič. Brez vmešavanja opazujemo, kako se na strani znajde, kako logični so mu meniji, si določene stvari in poteze drugače interpretira, kot smo si sami.

Priporočljivo si je ogledati tudi najbolj obiskane portale in analizirati njihov prikaz in delovanje. Zgledovati se je dobro tudi po drugih spletnih straneh in od njih povzemati boljše rešitve in se izogibati njihovim napakam.

Če postane spletni portal dovolj obiskan in seveda tehnično (povezava in strežnik) omogoča obisk večjega števila obiskovalcev, se izplača razmišljati tudi o oglaševanju spletne strani. S tem lahko zmanjšamo stroške spletnega portala, ali pa (ob pogoju, da nudimo kvalitetno vsebino in delovanje spletne strani) lahko razmišljamo celo o dobičku.

5.5 Pogoji za obiskanost in promocija spletnega portala

Za ustrezen nivo obiskanosti je potrebno skrbeti za promocijo portala. Glede na vstop obiskovalcev ločimo tri glavne načine vstopa :

- prek povezav spletnih iskalnikov,
- prek posrednih povezav, objavljenih na drugih spletnih straneh,
- neposreden vpis URL strani v brskalnik.

Veliko truda v zadnjih letih izdelovalci strani usmerjajo v optimizacijo spletnih strani za visoko pojavnost v spletnih iskalnikih (SEO – Search Engine Optimization). Za mnoge uporabnike namreč iskalnik pomeni vstopno točko v svet interneta. Uporabo vrstice za vnos URL naslovov počasi izpodrivajo neposredna okna za iskanje v iskalnikih, saj je tu za uspešen rezultat potrebnega manj tipkanja in doslednosti. To lahko vidimo v googlovem brskalniku Chrome, kjer je URL vrstica že združena z iskalnikom in ima tako dvojno funkcijo.

Filozofija iskalnika se je iz spletnega imenika, kjer so bile spletne strani nanizane po smiselnih sklopih, preselila v primerjanje vpisanih ključnih besed z vsebinami spletnimi stranmi. S kratkim vpisom ključnih besed uporabnika v iskalnik, le ta ponudi seznam spletnih strani, ki najbolje ustrezajo vnešenim ključnimi besedam. V ospredje iskalnik potiska strani tudi glede na njihovo priljubljenost in obiskanost, saj je že statistično večja verjetnost, da je uporabnik iskal priljubljeno stran, kot stran z nizkim obiskom.

Za to visoko pojavnost v spletnih iskalnikih pa se je potrebno truditi že pri izdelavi strani in po izdelavi pri ažurnosti. Pri sestavljanju vsebine strani je potrebno posvečati pozornost ključnim besedam, ki se morajo ves čas smiselno pojavljati v tekstu. Za iskalnike po slikah ustrezno poimenujemo tudi slike. Pri odprtju nove strani le to vpišemo v spletni iskalnik. Vsebino naše strani prebere aplikacija spletnega iskalnika imenovana »pajek« in jo zabeleži v svojo bazo. S tem je omogočena pojavnost v spletnih iskalnikih, ki pa jo je potrebno tekom obstoja strani ves čas dvigovati in ohranjati visoko mesto.

Visoko pojavnost v spletnih iskalnikih omogočijo tudi objavljene povezave spletne strani na drugih spletnih straneh. Lahko stran promoviramo na vsebinsko sorodnem forumu, objavimo predstavitveni film na video portalih (youtube), objavimo predstavitev na socialnih omrežjih (facebook) in podobno. Največji uspeh pa zagotavljajo samodejna posredovanja povezave uporabnikov drugim, saj to pomeni, da jim je bila stran všeč in da jo želijo predstaviti tudi drugim.

Za promocijo lahko skrbimo tudi s objavljanjem spletnega naslova v nedigitalni obliki. Štejejo pravzaprav vse metode marketinga nedigitalnih oblik (časopisni oglasi, posetnice, nalepke, majice, reklamni panoji...)

6 POTEK IZDELAVE SPLETNEGA PORTALA

6.1 Izgradnja statičnih elementov spletne strani (HTML in CSS)

Izgradnjo spletnega portala pričnemo z osnovnim jezikom namenjenemu spletnim stranem, to je HTML-jezikom. Ustvarimo novo tekstovno datoteko, ki jo nato shranimo s končnico ».html«. Pripravljen imamo prazen HTML-dokument. V dokument vstavimo osnovno ogrodje HTML-dokumenta:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
      <title>To je HTML dokument</title>
</head>
<body>
</body>
</html>
```

Povsem na začetku HTML-dokumenta vpišemo HTML-deklaracijo. Ta sicer ne sodi med HTML-značke. Gre za navodila spletnemu brskalniku, v kateri verziji HTML prikazujemo dokument. Deklaracija vsebuje povezavo na DTD (ang. Document Type Definition), to je na definicijo tipa dokumenta. Le ta vsebuje pravila določene verzije HTML, tako da lahko brskalnik vsebino pravilno prikaže.

Sledi prva značka HTML - značka <html>, ki začenja vsak dokument in ga na koncu dokumenta s končno značko </html> tudi končuje. Na ta način značka uokvirja celotno HTML-vsebino dokumenta, kar je lastnost gnezdenja.

Sledi značka <head>, ki predstavlja glavo dokumenta. V glavo vnesemo tiste parametre dokumenta, ki vplivajo na dokument, vendar niso neposredno prikazani v besedilu dokumenta. Znotraj glave je tudi značka <title>, s katero opišemo naslov HTML-dokumenta. Le ta je med ogledom viden v naslovni vrstici brskalnika, pa še kje.

V glavi dokumenta definiramo tudi JavaScript kodo in CSS-zapis, ki ju uporabimo v dokumentu. JavaScript kodo označimo z značko <script type="text/JavaScript">. Znački script definiramo lastnost »type, s katero definiramo tip opisane kode. JavaScript kodo zaključimo s končno značko </script>. V primeru, da se v dokumentu sklicujemo na zunanji JavaScript dokument, le to označimo v obliki: <script src="openlayers/OpenLayers.js" type="text/JavaScript" />, kjer smo kot lastnost dodali povezavo do JavaScript dokumenta.

CSS-zapis označimo z značko <style type="text/css">. Tudi tu lastnost opisuje vrsto dokumenta. V primeru, da se sklicujemo na zunanji CSS-dokument, pa le to označimo z značko: <link rel=" jquery.fancybox-1.3.4.css" type="text/css" media="screen" /> Ta ima sedaj poleg povezave in vrste datoteke definirano tudi izhodni medij, za katerega datoteka CSS velja. V našem primeru je to zaslon. Dokument bi namreč lahko oblikovali drugače za tiskanje, prikaz na dlančnikih, ipd.

Glavo HTML-dokumenta zaključimo z končno značko </head>. To moramo storiti preden začnemo z glavnim HTML-elementom za prikazovanje - telesom dokumenta, ki ga definiramo z značko <body>. V telo dokumenta vstavimo vse HTML-elemente, ki neposredno prikazujejo vsebino HTML-dokumenta.

Pripravljena HTML-struktura pa deluje neurejeno in suhoparno. Za lepši izgled za posamezne elemente definiramo ustrezne CSS-stile, ki jih v spletno stran vključimo kot povezavo na zunanjo datoteko:

```
<link rel="stylesheet" href="js/stili.css" type="text/css" media="screen"
/>
```

HTML-struktura spletne strani, se tako oblikuje po CSS-navodilih, ki so pripravljeni v datoteki stili.css. V datoteki pripravimo ustrezno oblikovanje za naslednje elemente:

- glavi spletne strani definiramo sliko ozadja,
- ustrezno oblikujemo naslov spletne strani,
- oblikujemo dinamično besedilno vrstico (marquee),
- oblikujemo ustrezne zamike in oblikovanje leve menujske vrstice,
- oblikujemo povezave,
- itd.

V spletno stran sta sicer vključeni še dve CSS-zunanji datoteki in sicer »style.css« in »jquery.fancybox-1.3.4.css«, ki sta del vključenih JavaScript knjižnic. Prva datoteka skrbi za ustrezno oblikovanje OpenLayers dinamične karte, druga pa za oblikovanje slikovnega okvirja za prikaz slik knjižnice Fancybox.

6.2 Konflikt med dinamičnimi spletnimi stranmi in spletnimi iskalniki

Klasična postavitev HTML-spletne strani je sestavljena iz glavne oz. uvodne strani in posameznih podstrani, do katerih dostopamo z objavljenimi HTML-povezavami na glavni strani. Glavna stran in posamezne podstrani so običajno samostojne HTML-datoteke, katere spletni brskalnik prebere in prikaže vsakič znova.

Klasično postavitev HTML-spletne strani upoštevajo in dajejo prednost tudi spletni iskalniki. Spletni iskalniki so za spletni portal, ki je namenjen širši javnosti, zelo pomembni, saj so pogosto glavni vir obiskovalcev. Če želimo, da spletni iskalnik prikazuje tudi naš portal, le tega v iskalnik vpišemo. Iskalnik spletni portal večkrat avtomatično pregleda s t.i. pajki, najpogosteje pa upošteva sistem podstrani portala v klasični HTML-obliki.

S pojavom Ajax-a in dinamičnih spletnih strani pa se je pojavila metoda, ki ne slede tem klasičnim pravilom. Pri klasični metodi menjava vsebine podstrani pomeni osvežitev celotne vsebine in zamenjavo URL-naslova. Vsaka podstran in njena vsebina je tako povezana z enoličnim URL-jem, kar izkoriščajo tudi spletni iskalniki. Dinamične spletne strani z uporabo Ajax-a, pa omogočajo komunikacijo z strežnikom brez menjave URL-naslova. To pomeni, da lahko spreminjamo vsebino dinamično, ne da bi se osvežila celotna stran.

Metoda dinamičnih strani je koristna pri spletnih straneh, ki za delovanje uporabljajo obsežno JavaScript kodo. Tovrstne strani bi zaradi obsežnosti programske kode lahko imenovali kar spletne aplikacije. Če bi tovrstne spletne aplikacije povezali z klasično obliko podstrani, bi to pomenilo za uporabnika precej čakanja, saj bi se morala JavaScript koda izvesti ob vsaki menjavi podstrani, to pa je za uporabnika, ki ne tolerira čakanja na podatke, nesprejemljivo in taka spletna stran je posledično neuporabna.

Ta konflikt v veliki meri zadeva tudi to nalogo. Povezati želimo namreč zmogljivo dinamično karto, ki zahteva dinamično metodo prikaza, z vsebino, ki želimo, da je visoko rangirana v spletnih iskalnikih, kar pa zahteva klasično metodo prikaza. Reševanje tega konflikta je opisano v naslednjih poglavjih.

6.3 Metode reševanja konflikta med dinamičnimi stranmi in iskalniki

Kot smo omenili, ima v klasični obliki vsaka podstran svojo enolično URL-povezavo. Sistem dinamične strani, ki s pomočjo Ajax-a spreminja vsebino dinamično, brez spremembe URL-ja, pa izgubi to možnost. Spletni programerji so se zato zatekli k URL-parametru označenem z lojtro »#« imenovanem »fragment identifier« ali tudi »URL-hash«. Ta parameter je v klasični obliki spletnih strani namenjen za označevanje elementov znotraj enega HTML-dokumenta, pa tudi navigaciji znotraj HTML-dokumenta, saj v tem primeru sprememba »#« parametra v URL-ju ne zamenja podstrani in ne osveži spletne strani. Prav te lastnosti pa so skupne tudi dinamičnim stranem, zato se je URL-parameter lojtra dodobra uveljavil pri Ajax dinamičnih spletnih straneh.

6.3.1 Sistem označevanja googlovega iskalnika za Ajax dinamične strani

Google, ki trenutno upravlja tudi z najbolj uporabljanim spletnim iskalnikom, se zaveda tega konflikta, kar omenijo tudi na svoji spletni strani (http://code.google.com/intl/sl-SI/web/ajaxcrawling/docs/learn-more.html). Interes googlovega iskalnika je, da so v njegovih iskalnih bazah na voljo vsi relevantni spletni viri in vsebine, tako tudi vsebine dinamičnih strani. Zato je Google objavil svojo verzijo rešitve za ustrezno prikazovanje dinamičnih spletnih strani v svojem iskalniku.

Googlov iskalnik spletne strani občasno avtomatsko obišče z aplikacijo imenovano »pajek« in prebere vsebino spletne strani. Besedilo, podatke o slikah in drugih dokumentih shrani v bazo in s tem omogoči kasnejši dostop z iskanjem po ključnih besedah, ki jih uporabniki vnesejo v iskalnik. Pajek obišče uvodno stran in vse podstrani, katerih povezave so objavljene na uvodni strani. Pogoj so seveda povezave objavljene v klasični obliki. Dinamičnih povezav, ki prožijo na primer JavaScript ali Ajax dogodke, iskalnik interpretira manj uspešno, ali pa take povezave sploh ne upošteva.

Kljub temu Google teži k temu, da bi v iskalniku prikazal tudi vsebino dinamičnih strani, ki jih klasičen način avtomatskega »branja« iskalnika ne zazna tako uspešno. Google je zato razvil možnost avtomatskega »branja« s pajki tudi Ajax dinamičnih strani, katere je potrebno označevati z določenimi pravili.

Da googlovi pajki prepoznajo dinamični sistem podstrani, je URL-parametru lojtra »#« potrebno dodati še klicaj. Googlov iskalnik bo tako označene povezave na podstrani ustrezno upošteval. Ker googlov iskalnik ne more pognati Ajax kode, pajek ne vidi ustrezne vsebine, če sledi povezavam podstrani. Zato je Google predvidel, da mora upravljalec tovrstne spletne strani poskrbeti, da strežnik omogoča izdelavo t.i. »HTML-posnetka« (angl. »HTML-snapshot«). HTML-posnetek strani omogoča, da je vsebina strani posredovana z Ajax-om pajku dostopna, saj ta HTML-kode ne more prebrati na klasičen način.

Poenostavljen opis poteka je sledeč: Ko googlov pajek obišče uvodno spletno stran, zazna oznako »#!« v povezavi. Pajek to povezavo spremeni tako, da oznako »#!« spremeni v »?_escaped_fragment_=«. Pajek torej spremeni parameter »#«, ki ne omogoča interakcijo s strežnikom v »?«, ki pa to omogoča. S to spremembo si je pajek zagotovil dostop do strežnika spletne stran. Sedaj lahko pošlje zahtevo po »HTML-posnetku«, to je HTML-koda, ki se prikaže ob konkretni povezavi. Za izdelavo HTML-posnetkov je potrebno dodatno priskrbeti

aplikacije, ki to omogočajo. Ena izmed njih je na primer Crawljax (http://crawljax.com/). Ko pajek prejme HTML-posnetek te podatke shrani, poveže pa jih z prvotno obliko URL-ja opremljeno z lojtro. V taki obliki ga bodo tako ves čas uporabljali tudi uporabniki.

Tovrstni sistem ima nekaj slabosti. Ker gre za googlov sistem, deluje le na njihovem iskalniku. Sistem uvaja značilen parameter »#!«, ki slabi uporabniku prijazno obliko povezave. Po spletu so opazne tudi kritike, da je to zasilna rešitev in da je boljša alternativa t.i. »Hijax« sistem spletnih strani, ki omogoča uporabo dinamične spletne strani za tiste, ki omogočijo uporabo JavaScripta v svojih brskalnikih, HTML-vsebino pa prikažejo tudi tistim, ki uporabe JavaScripta ne omogočijo (torej tudi pajkom iskalnikov).

6.3.2 Rešitev »Hijax«

Dinamične strani so vezane na uporabo JavaScript jezika. Dekodiranje JavaScript jezika vključenega v spletne strani je v današnjih dneh bolj samoumeven kot v preteklosti. Vsi večji brskalniki že v osnovi podpirajo dekodiranje JavaScript jezika, omogočajo pa funkcijo ogledovanja spletnih strani brez zaganjanja JavaScript kode. Veliko bolj kot uporabniki pa so danes za spletne strani pomembni pajki iskalnikov, ki tudi nimajo sposobnosti zagona JavaScript kode. Pajki torej ne zaznajo dinamičnih sprememb na spletni strani.

Na tej podlagi se je razvila tehnologija »Hijax«, ki sledi tehnologiji imenovani »progressive enhancement«. Bistvo te tehnologije je zgraditi isto spletno stran vzporedno tako, da lahko do enake vsebine dostopajo ljudje z različno »tehnološko opremljenostjo« svojega brskalnika. Tehnologija »Hijax« tako omogoča dostop do istih spletnih vsebin uporabnikom z ali brez podpore JavaScript. Seveda bodo tisti brez dinamične podpore brskalnika prikrajšani za hitrost in funkcionalnost, ki jo omogoča Ajax, vendar bodo še vedno lahko dostopali do objavljene vsebine. Kot rečeno je to bistvenega pomena v primeru spletnih isklanikov.

6.4 Izgradnja sistema dinamičnih elementov spletne strani

Zgoraj omenjani konflikt zadeva tudi izgradnjo spletne strani v okviru te naloge. Želja je namreč zgraditi spletni portal z uravnovešeno dinamično karografsko in tekstovno vsebino. Za predstavitev tematike v tekstovni obliki s statičnimi slikovnimi elementi bi zadostovala spletna stran v klasični obliki. Ker pa dinamična karta zahteva podporo velike JavaScript knjižnice, ki se mora ob vsakem odprtju strani inicializirati, bi vsakokratno nalaganje ustrezne podstrani upočasnilo delovanje. Zato tudi v tem primeru uporabimo dinamično spreminjanje vsebine s pomočjo Ajax-a. Ker poleg dinamične strani zahtevamo tudi prikazovanje celotne vsebine v iskalnikih, se poslužimo metode »Hijax«.

Zgrajen sistem navigacije spletne strani lahko ločimo na zunanje in notranje povezave spletne strani, ki uporabljajo različen sistem.

6.4.1 Sistem notranjih povezav

Sistem notranjih povezav pomeni uporabo povezav na spletni strani, prek katerih uporabnik dinamično spreminja vsebino spletnih strani. Zapisane so v obliki HTML-elementa za povezavo:

```
<a class="sub_podmeni" id="SV/enoteSV/45_oklepni_bataljon" href="1110/">
```

Povezava ima definirane tri lastnosti. Lastnost »class« je oznaka nivoja podmenija. Na to oznako se sklicuje stilska predloga »stili.css«, ki povezavo ustrezno oblikuje. Na lastnost »id« se sklicuje Ajax koda, ki sproži dinamično spremembo vsebine na strani z sledečo JavaScript funkcijo:

```
function applyEvents(){

        var names = [   "SV",
                                "SV/enoteSV",
                                "SV/objektiSV",
                                "SV/enoteSV/45_oklepni_bataljon",
                                "SV/enoteSV/460_artilerijski_bataljon",
                                "SV/enoteSV/sola_za_podcastnike",
                                "SV/enoteSV/center",
                                "SV/enoteSV/muzej",
                                "SV/enoteSV/raziskave_simulacije",
                                "SV/objektiSV/vojasnica_po",
                                "SV/objektiSV/vojasnica_pi",
                                "SV/objektiSV/pocek",
                                "SV/objektiSV/bac",
                                "SV/objektiSV/polhova_jama",
                                "SV/objektiSV/pecna_reber",
                                "SV/objektiSV/mackovec"
                                ];
            for(var j in names)
            {
       if(document.getElementById && document.getElementsByTagName){
        var arrAllLinks = document.getElementsByTagName("a");
        var oLink;
        for(var i=0; i<arrAllLinks.length; i++){
                        oLink = arrAllLinks[i];
                        if(oLink.id.search(names[j]) != -1){
                               oLink.onclick = function (oEvent){
                                 var oEvent = (typeof oEvent !=
"undefined")? oEvent : event;
                                        oEvent.returnValue = false;

                                        if(oEvent.preventDefault){
                                                oEvent.preventDefault();
                                        }
                                        tarca = oEvent.srcElement?
oEvent.srcElement : oEvent.target;
                                    geslo = tarca.id;
                                odpri(geslo, status);
                                  }}}}}}
```

Funkcija applyEvents() ob zaznanem kliku na poljubno povezavo prestreže klasično navigiranje in dinamično spremeni ustrezno vsebino. To stori tako, da najprej prebere lastnost id povezave, ki je hkrati kar relativna URL-povezava na prikazano vsebino. Ta funkcija povsem na koncu kliče funkcijo »odpri«, ki ima definirani tudi dve spremenljivki »page« in »status«. Ta funkcija skrbi za pravilno dinamično odpiranje podmenijev, zavihkov in za odprtje ustrezne podvsebine. Funkcija se kliče vsakokrat, ko je potrebno nastaviti ustrezno podstran, torej tudi ob prihodu uporabnikov od »zunaj« in ne samo ob kliku na notranjo povezavo. Funkcija vsakič glede na odprto vsebino ustrezno spremeni tudi URL-parameter (za znakom lojtra »#«), ki ne povzroči osvežitve strani. Parameter vsebuje obe spremenljivki (»page« in »status«), ki ju uporabi funkcija odpri. Namen tega parametra je, da se dinamično spreminjanje vsebine odrazi tudi na spremembi URL-ja. Bistveno vlogo pa ima v primeru, da uporabnik URL spletne strani shrani ali posreduje. V tem primeru je shranjen tudi parameter

lojtra. Ko bo uporabnik čez čas sledil shranjeni povezavi, bo parameter lojtro ponovno prebrala funkcija »odpri« in nastavila točno tisto vsebino, ki jo je uporabnik shranil. S tem zagotovimo enolično označevanje vsebine s strani uporabnika. Žal pa seveda enaka metoda ne deluje za spletne iskalnike, saj se pajkom iskalnikom nastavitvena funkcija ne sproži.

Tudi zato je objavljena tretja lastnost povezave »href«. To je klasična lastnost elementa HTML-povezave, v katerem je shranjena relativna ali absolutna povezava na podstran, na katero bi se ob kliku uporabnik navigiral, če ga nebi prestregla koda Ajax. Ta povezava pa kljub temu ni »slepa«, ampak je po metodi »Hijax« uporabna za tiste uporabnike (in predvsem pajke iskalnikov), katerim JavaScript ne deluje in Ajax ne more prestreči navigiranja v klasični obliki. V zgoraj navedeni HTML-povezavi pa vidimo, da »href« lastnost ni v obliki klasične URL-povezave (kot npr. »SV/enoteSV/45_oklepni_bataljon.html«), ampak v obliki numerične šifre.
Razlog za uvedbo te šifre je izgled URL. Če uporabljamo URL v klasični obliki in mu dodamo parameter lojtro, bi URL izgledal nekako takole:

```
SV/enoteSV/45_oklepni_bataljon.html#SV/enoteSV/45_oklepni_bataljon&enciklop
edija
```

Vidimo, da je URL zapleten in neprivlačen. Razlog za to je vzporedno vodenje strani po metodi »Hijax«, ki zahteva podvojene povezave za klasični in dinamični del. Z uvedbo numerične šifre za klasični del URL-ja, lahko URL skrajšamo in vizualno olepšamo:

```
1110/#SV/enoteSV/45_oklepni_bataljon&enciklopedija
```

Tekstovni del spletne strani je tako opisan samo enkrat in privlačnejši za ogled. Da pa to številčno »kodo« lahko sploh uvedemo, uporabimo funkcijo strežnika Apache »mod_rewrite«.

6.4.2 Apache mod_rewrite

Strežnik Apache zagotavlja zmogljiv in zelo uporaben modul »mod_rewrite«. Modul ob prejemu URL-zahtevka le tega po vnaprej določenih pravilih uporabnika spremeni, preusmeri ali zaustavi. Modul se največkrat uporablja bodisi za definiranje datotek, ki jih uporabniki ne smejo odpirati neposredno, ali pa v primerih, ko strežnik za prikaz spletne strani potrebuje veliko parametrov, ki pa so uporabnikom in spletnim iskalnikom v neprijazni obliki. Modul tako omogoča, da neprivlačen URL-zahtevek poln parametrov, ki ga potrebuje strežnik, kot je na primer tale izmišljen primer:

```
spletnastran.com/podstran.html?param=SV&mode=enoteSV&oblika=enciklopedija
```

preoblikujemo v uporabnikom in spletnim iskalnikom privlačno obliko:

```
spletnastran.com/podstran/SV/enoteSV/enciklopedija
```

Nov URL je sedaj krajši in bistveno bolj pregleden. Uporabniki bodo ves čas obiska spletne strani uporabljali privlačnejšo obliko URL. Ko uporabnik pošlje privlačno obliko zahtevka URL strežniku, le tega prestreže modul mod_rewrite. Ta ga po vnaprej določenih navodilih preoblikuje v manj privlačno obliko URL z dodanimi parametri, ki jih zahteva strežnik in prikaže ustrezno stran. Tako olepšan URL sicer označujemo z izrazom »canonical URL«.

Kot rečeno, ta modul uporabimo tudi sebi v prid, za olepšanje URL-naslova. Da strežniku Apache definiramo ustrezna navodila za preoblikovanje URL-jev, moramo v osnovni mapi strežnika (kjer je umeščena uvodna spletna stran) ustvariti dokument ».HTACCESS«. V njej definiramo naslednje vrstice:

```
Options +FollowSymLinks
RewriteEngine On
RewriteRule ([0-9]+)/$ spletnastran.php?mode=$1 [NC]
```

V prvih dveh vrsticah modulu naročimo naj omogoči spreminjanje povezav. V tretji vrstici pa konkretneje definiramo, kako želimo URL preoblikovati. Vrstica modulu naroča naj poljubno zaporedje števil pred znakom »/« v »olepšanem« URL-ju (([0-9]+)/) , spremeni v »slabši« url s tem, da se zaporedje števil zapiše kot URL-parameter »mode«. Konkretno se bo povezava:

```
1110/
```

ki smo jo lahko videli kot klasično href lastnost v HTML-povezavah spletne strani, spremenila v:

```
spletnastran.php?mode=1110
```

Za branje parametra »mode«pa zadolžimo PHP.

6.4.3 Vloga PHP in sistem zunanjih povezav

Vloga PHP je glede na njegove zmogljivosti v našem primeru zelo skromna. PHP uporabimo pri sledenju klasičnih povezav. Ko uporabnik brez JavaScript podpore ali pajek iskalnika sledi klasični spletni povezavi v »lepi« obliki, spremeni modul »mod_rewrite« URL v »slabo« obliko. V našem primeru je to povezava na uvodno spletno stran portala z dodanim parametrom »mode«, ki nosi številsko »kodo« iz »lepe« oblike URL-ja.

PHP kodo bomo dodali pred osnovno HTML-strukturo uvodne spletne strani. V klasičnem »Hijax« načinu bi klasične povezave vodile na podstrani. V našem primeru pa PHP-koda omogoča, da se uvodni strani spletnega portala fizično naloži vsebina podstrani in se v primeru vključenega JavaScript naložijo tudi ustrezni parametri. Zakaj pa je to pomembno, če klasičnim povezavam sledijo le uporabniki brez vključenega JavaScript?

To pravzaprav ne drži v kar precej primerih. Ko pajek iskalnika obišče spletno stran, sledi zgoraj opisanem postopku in iskalnik prebere ustrezno podstran, ki mu jo pripravi PHP-koda. Pajek iskalnika pa bo v svoji bazi shranil klasično in ne dinamično povezavo! Ko bo torej naključni uporabnik, ki bo prek iskalnika našel spletno stran ali eno od podstrani, bo v bistvu sledil klasični povezavi, čeprav bo imel vključen JavaScript in s tem možnost dinamičnega prikaza. PHP ima zato vlogo, da hkrati poskrbi za oba primera – bodisi pripravi ustrezno vsebino podstrani za prikaz brez JavaScript in za prikaz z JavaScript podporo.

V sami PHP-kodi definiramo nekaj opravil, ki omogočijo ustrezno odprtje vsebine podstrani. PHP najprej poizkuša prebrati parameter »mode« iz zahtevanega URL. Če le ta ne obstaja, se PHP-koda ne izvrši. Če pa ta obstaja (v primeru klasične povezave), najprej številčno kodo iz parametra preoblikuje v ustrezno povezavo URL. Koda je sestavljena iz štirih številk, pri čemer prva izmed številk pomeni glavni meni, druga drugi podmeni itd. Če je število 0, ta podmeni ni odprt, če je število od 1 do 9, je odprt eden izmed ustreznih menijev ali podmenijev. PHP na podlagi dobljenega URL odpre ustrezno HTML-datoteko podstrani in iz

nje prebere vsebino, metapodatke, itd. V primeru, da uporabnik, ki je kliknil na klasično povezavo ima podporo JavaScript, nastavi tudi ustrezne JavaScript nastavitve. Rezultat PHP-kode je ustrezno odprta vsebina podstrani, neglede na to, ali uporabnik ima ali nima podpore za JavaScript.

7 PRIPRAVA VSEBINE IN KARTOGRAFSKIH PRIKAZOV

7.1 Iskanje virov prostorskih podatkov

Vsebina portala izdelanega v okviru te naloge je vojaška zgodovina. Za tovrstno tematiko obstaja nekaj spletnih in veliko knjižnih virov iz katerih je možno črpati gradivo.
Vsebinski cilj spletnega portala je:

- zbrati vse pisne in slikovne vire o temi na enem mestu,
- dogodkom in vsebini pridati prostorski prikaz v obliki dinamičnih kart in s tem uporabniku omogočiti dodatno uporabniško izkušnjo, prostorsko preglednost in boljšo razumljivost tematike.

Za vsebino, ki bi jo želeli predstaviti tudi v kartografski obliki, je potrebno pridobiti ustreze prostorske podatke. Nekateri uporabni prostorski podatki morda že obstajajo in jih lahko pridobimo:

- prek prostih spletnih virov,
- z naročilom podatkov Geodetske uprave RS,
- pri občini, podjetju ali organizaciji, ki je povezana z vsebino,
- z izmero na terenu z ročnim GPS sprejemnikom z informativno natančnostjo.

Prostorsko umeščeni podatki so lahko posredno navedeni v pisnih ali spletnih virih:

- na izdelanih kartah v različnih projekcijah in koordinatnih sistemih,
- opisno ali poimensko omenjeni znotraj same vsebine.

V slovenskem pa tudi tujem spletu je možno najti najrazličnejše uporabne spletne vire, kot so na primer:

- brezplačni podatki Geodetske Uprave RS (http://e-prostor.gov.si/index.php?id=263&no_cache=1&tx_simpltabs_pi1[tab]=561#tabs)
- brezplačni podatki Arhiva Slovenije, rastrski sloji starih katastrov iz 19. Stoletja (http://sigov3.sigov.si/cgi-bin/htqlcgi/arhiv/enos_isk_kat.htm)
- skenirane zgodovinske karte Digitalne knjižnice Slovenije (http://www.dlib.si/v2/Browse.aspx?type=zemljevidi)
- okoljski prostorski podatki Agencije RS za okolje, ki omogoča tudi spletni servis WFS: (http://gis.arso.gov.si/geoportal/catalog/main/home.page)
- okoljski podatki omrežja EIONET: (http://nfp-si.eionet.europa.eu/)

- vektorski podatki območij varstva narave: (http://www.zrsvn.si/sl/informacija.asp?id_meta_type=62&id_informacija=705)
- vektorski podatki Statističnega urada RS: (http://www.stat.si/gis/)
- vektorski in rastrski podatki kartografskih gradiv državnih prostorskih načrtov: (http://www.mop.gov.si/si/delovna_podrocja/prostorski_nacrti/drzavni_prostorski_nacrti/)
- itd.

Bolj uporabni podatki Geodetske uprave, ki jih je možno naročiti, so na primer:

- ortofoto posnetki,
- podatki modelov višin ,
- topografski podatki različnih meril,
- državne topografske karte,
- državne pregledne karte,
- itd.

Za nekomercialne namene so ti navoljo brezplačno oz. je potrebno plačati stroške posredovanja, za komercialne namene pa se uporabo podatkov plača po objavljenem ceniku.

7.2 Zajem podatkov iz pisnih in slikovnih virov

Predvsem pri zgodovinskih, pa tudi drugih tematikah je precej prostorskih podatkov navedeno opisno ali slikovno v obliki skic, kart in zemljevidov, največkrat v neznani projekciji ali koordinatnem sistemu. Potrebujemo torej ustrezno GIS-aplikacijo, ki omogoča:

- obdelavo vektorskih in rastrskih tipov podatkov sočasno,
- definiranje koordinatnega sistema, v katerem bodo zajeti podatki,
- georeferenciranje rastrske karte iz neznanega v ciljni koordinatni sistem,
- transformacije med koordinatnimi sistemi,
- zajem vektorskih podatkov na podlagi rastrskih podatkov,
- itd.

Za zmogljive komercialne GIS-aplikacije, ki omogočajo naštete naloge, je bilo običajno potrebno odšteti precej denarja. S pojavom dovolj zmogljivih brezplačnih aplikacij tudi na GIS-področju, pa je reševanje tovrstnih nalog s področja prostorskih podatkov postalo dostopno tudi širši množici.

7.3 Uporaba interaktivnih podlag brezplačnih spletnih ponudnikov

Na spletu se je v zadnjih letih pojavilo veliko brezplačnih kartografskih spletnih aplikacij, ki poleg pregledovanja prostorskih podatkov omogočajo tudi vpogled v razvito API-kodo teh aplikacij in s tem vključevanje in prikazovanje prikazov v lastne namene, pod pogojem, da so te vsebine dostopne uporabnikom brezplačno. Nekateri tovrstni ponudniki so na primer: Google maps, Virtual Earth, OpenStreetMap in Yahoo Maps.

Razlog, zakaj sploh uporabiti tovrstne interaktivne podlage je predvsem v tem, da prikazovanje lastnih rastrskih slojev zahteva veliko prostora na diskih, procesno sposobnost strežnika in zmogljivo povezavo s spletom. Za kvaliteten prikaz karte pa je nujna zapolnitev karte z vsebino, saj so karte z manj podatki (veliko beline) manj vizualno privlačne. Vključitev teh vsebin v lastno interaktivno karto pa omogoči hiter prikaz zahtevnih rastrskih podlag, ki jih posreduje strežnik te storitve, z lastnega strežnika pa se prenesejo le še dodani tematski prostorski sloji.

Ima pa tovrstno vključevanje gostujočih podlag tudi nekatere pomankljivosti. Izgled, barvne sheme in kartografski znaki kart je kljub izbiri več slojev možno spreminjati le v omejenem obsegu. Z lastno tematsko vsebino se moramo tako prilagajati shemam gostujočih podlag. Ker so tovrstni ponudniki brezplačni, omogočajo izdelavo kart množici bolj in manj izkušenim uporabnikom. Na spletu je tako zelo veliko prikazov z vključenimi tovrstnimi interaktivnimi podlagami in uporabnik utegne pri pogostejši uporabi spleta začne tovrstne podlage dojemati kot cenene. Interaktivne spletne karte z lastnimi podlagami pa delujejo sveže ter privlačno in tako bolj pritegnejo začetno zanimanje uporabnika. Enostavnejše oblike portalov bodo tako »obsojene« na izrabo gostujočih interaktivnih prostorskih slojev. Bolj dognani portali, namenjeni višji obiskanosti, pa bodo zahtevali bolj zmogljivo opremo in posledično tudi večji finančni vložek, kar pa bo omogočalo prikaz lastnih rastrskih podlag in tovrsten portal bo bolj avtentičen in verodostojen.

7.4 Koordinatni sistemi

7.4.1 O označevanju koordinatnih sistemov in projekcij

V svetu se je skozi zgodovino razvilo veliko število različnih koordinatnih sistemov, katerih razvoj je bil odvisen od matematičnih pristopov in projekcij, različnih elipsoidov, prilagajani so bili za različne celine in države, itd. V zadnjih letih koordinatne sisteme definirajo že komercialne potrebe, svoj koordinatni sistem za prikaz svojih storitev ima na primer tudi Google.

Z velikim številom koordinatnih sistemov se je pojavila tudi potreba po enotnem označevanju. Te problematike so se najbolj zavedali tisti, ki so se z njo soočali pri svojem delu. Med te zagotovo sodijo tudi iskalci naftnih zalog, ki so se po vsem svetu soočali s prostorskimi podatki v najrazličnejših koordinatnih sistemih. Organizacija EPSG (European Petroleum Survey Group), ki je obstajala od leta 1986 do 2005 - po tem letu je postala sestavni del OGP (International Association of Oil and Gas Producers) - je razvila »EPSG Geodetic Parameter Set« oz. zbirko geodetskih parametrov EPSG, ki predstavlja široko bazo elipsoidov, geodetskih datumov, geografskih in projekcijskih koordinatnih sistemov in enot. Ta baza je seveda uporaben zbir koordinatnih sistemov tudi za druga najrazličnejša področja.
Baza označuje posamezne koordinatne sisteme v bazi z oznako »EPSG«, dvopičjem in kodo koordinatnega sistema (npr. EPSG:2170). Bazo in sistem označevanja sta prevzeli tudi brezplačni aplikaciji Quantum GIS in GeoServer. Za definiranje ustreznega koordinatnega sistema moramo zato poznati bazo in sistem kodiranja EPSG.

Za nas uporabnejši koordinatni sistemi imajo naslednje EPSG-oznake:

- EPSG:3787(državni koordinatni sistem D48, enota metri)
- EPSG:3794 (novi državni koordinatni sistem D96, enota metri)
- EPSG:4326 (koordinatni sistem sistema GPS, enota stopinje)

Parametre vsakega koordinatnega sistema se za uporabo zapisuje v standardno obliko zapisa. Ena takih oblik je WKT (Well known Text). Primer za državni koordinatni sistem EPSG:3787:

```
PROJCS["MGI / Slovene National
Grid",GEOGCS["MGI",DATUM["Militar_Geographische_Institute",SPHEROID["Bessel
1841",6377397.155,299.1528128,AUTHORITY["EPSG","7004"]],TOWGS84[577.326,90.
129,463.919,5.137,1.474,5.297,2.4232],AUTHORITY["EPSG","6312"]],PRIMEM["Gre
enwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.0174532925199433,AUTHOR
ITY["EPSG","9108"]],AUTHORITY["EPSG","4312"]],UNIT["metre",1,AUTHORITY["EPS
G","9001"]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin
",0],PARAMETER["central_meridian",15],PARAMETER["scale_factor",0.9999],PARA
METER["false_easting",500000],PARAMETER["false_northing",-
5000000],AUTHORITY["EPSG","3787"],AXIS["Y",EAST],AXIS["X",NORTH]]
```

V WKT-datoteki so med drugim zabeleženi tudi parametri elipsoida in projekcije, pod parametrom »TOWGS84« pa je zabeleženih 7 parametrov sedem parametrične transformacije, ki jih aplikacija uporabi, če želimo podatke prikazovati v kateri izmed projekcij z WGS-84 elipsoidom (npr. GPS-koordinatni sistem), ti podatki pa so projecirani na podlagi drugačnega elipsoida.

Tudi ESRI v okviru svojih aplikacij uporablja, sicer nekoliko spremenjeno, WKT-obliko zapisa. Ta zapis se shrani v .prj datoteko, ki je (neobvezni) del datoteke shape. Primer ESRI WKT-oblike enakega koordinatnega sistema:

```
PROJCS["MGI / Slovene National
Grid",GEOGCS["MGI",DATUM["D_MGI",SPHEROID["Bessel_1841",6377397.155,299.152
8128]],PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433]],PROJECTIO
N["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["centr
al_meridian",15],PARAMETER["scale_factor",0.9999],PARAMETER["false_easting"
,500000],PARAMETER["false_northing",-5000000],UNIT["Meter",1]]
```

Pogosta je tudi PROJ4 oblika zapisa. PROJ4 je sicer brezplačna knjižnica koordinatnih sistemov avtorja Geralda Evendena. Del te knjižnice je tudi programska koda, ki omogoča transformacije med različnimi koordinatnimi sistemi. Knjižnica je brezplačna in na voljo za številne spletne in programske jezike, med drugimi tudi za JavaScript (Proj4js), PHP, (proj4php), postGIS, itd., v svoje programsko okolje pa jo je vključilo že mnogo brezplačnih aplikacij. Osnovna spletna stran je dostopna na: http://trac.osgeo.org/proj. Proj4 datoteka se za koordinatni sistem EPSG:3787 zapiše v naslednji obliki:

```
+proj=tmerc +lat_0=0 +lon_0=15 +k=0.9999 +x_0=500000 +y_0=-5000000
+ellps=bessel +towgs84=577.326,90.129,463.919,5.137,1.474,5.297,2.4232
+units=m +no_defs
```

Tovrstna oblika sicer izhaja iz konzolne uporabe programa za transformacijo.

Koordinatne sisteme in njihove zapise v različnih formatih pregledno združuje spletna stran http://spatialreference.org/

7.4.2 Izbor koordinatnega sistema

Za prikaz interaktivne karte moramo izbrati ustrezen koordinatni sistem, kateremu morajo ustrezati vsi prikazani podatki. Na izbor lahko vpliva vprašanje ali bomo uporabniku omogočili uporabo koordinat (branje koordinat in/ali iskanje po koordinatah...). Če želimo

uprabniku omogočiti uporabo koordinat, bomo izbrali tisti koordinatni sistem, za katerega bomo predvideli, da bo za uporabnika bolj uporaben. Lahko se na primer odločimo za državni GK državni sistem, vendar ta s pojavom GPS- naprav pri uporabnikih ni več tako pogosto uporabljan kot koordinatni sistem sistema GPS. Bolj zmogljivi portali s spletno karto omogočajo operiranje tudi s koordinatami več koordinatnih sistemov, vendar teh rešitev v našem primeru ne bomo obravnavali. Če je spletni portal z interaktivno karto bolj predstavitveno naravnan, upravljanje s koordinatami niti ni nujno, ključno in potrebno, kar pušča ponudniku karte bolj odprte možnosti pri izdelavi.

7.5 Priprava prostorskih podatkov z aplikacijo Quantum GIS

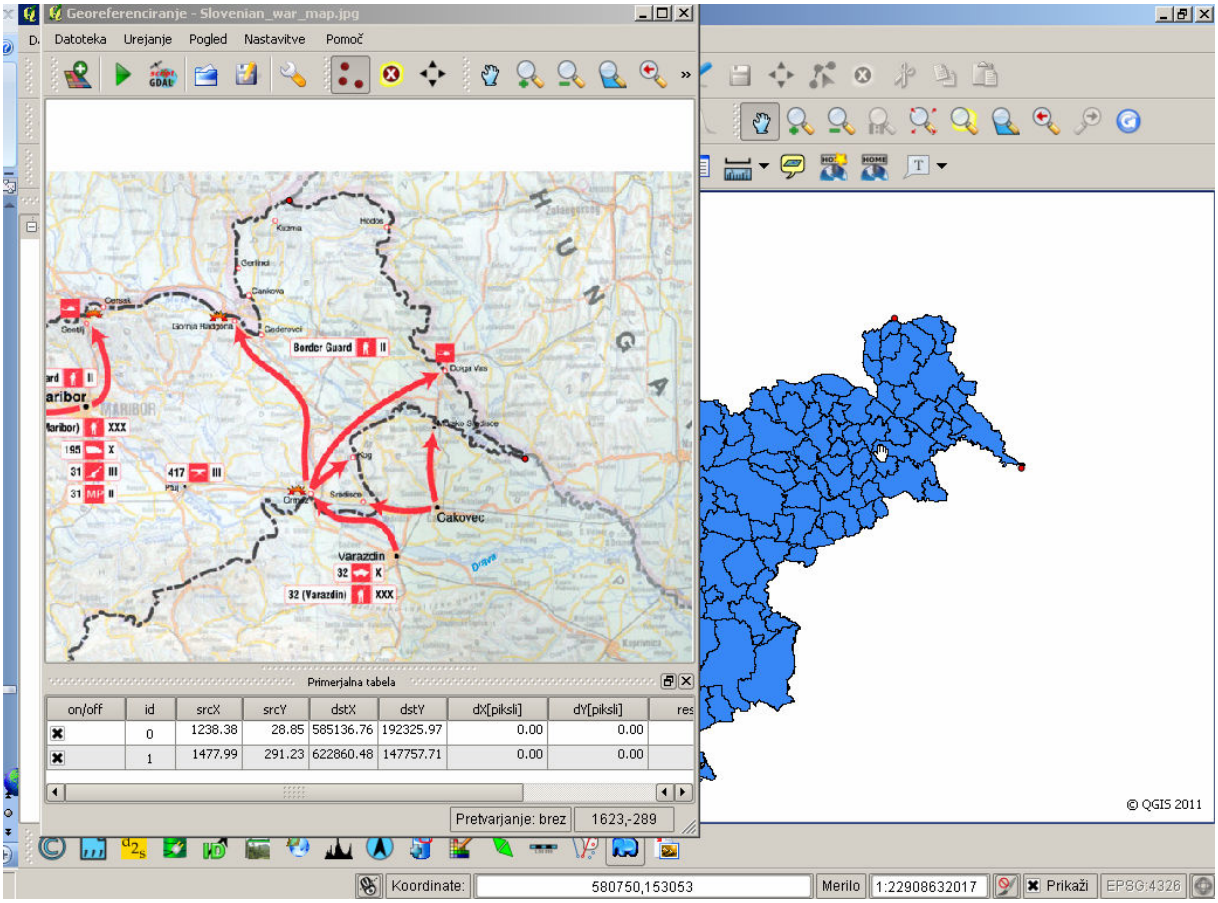
Quantum GIS je lahko uporaben pripomoček pri pripravi prostorskih podatkov. Kot vsako zmogljivo GIS-orodje omogoča ustvarjanje in editiranje slojev, tako rastrskih kot vektorskih. Opisani postopki veljajo za verzijo 1.7.0 »Wroclaw«, prevedeno v slovenski jezik.

7.5.1 Georeferenciranje slojev

Georeferenciranje slojev je uporaben postopek, ko želimo karti v navadni rastrski obliki (rastrska podoba) določiti koordinatni sistem in jo tako spremeniti v rastrski sloj (podobo shranimo v format v kateremu ima vsak piksel koordinato v določenem koordinatem sistemu). Postopek je uporaben za zajem kart, ki imajo bodisi znan ali neznan koordinatni sistem. Če ima podoba znan koordinatni sistem in prikazano kartografsko mrežo, lahko z določitvijo točk na presekih mreže precej natančno določimo lego karte. Če karta nima kartografske mreže in morda tudi neznan koordinatni sistem, moramo karto primerjati z nekim drugim, prostorsko že določenim slojem. Na obeh slojih definiramo točke, ki so skupne na obeh podobah.

Za izvedbo georeferenciranja v Quantum GIS orodju, v orodni vrstici izberemo ikono za zagon vtičnika »Georeferencer«. Odpre se novo okno vtičnika. V njem s klikom na ustrezno ikono odpremo rastrsko podobo, ki jo bomo georeferencirali. Za primer smo izbrali tematsko karto CIE (http://upload.wikimedia.org/wikipedia/commons/7/78/Slovenian_war_map.jpg), ki prikazuje potek desetdnevne vojne za Slovenijo.

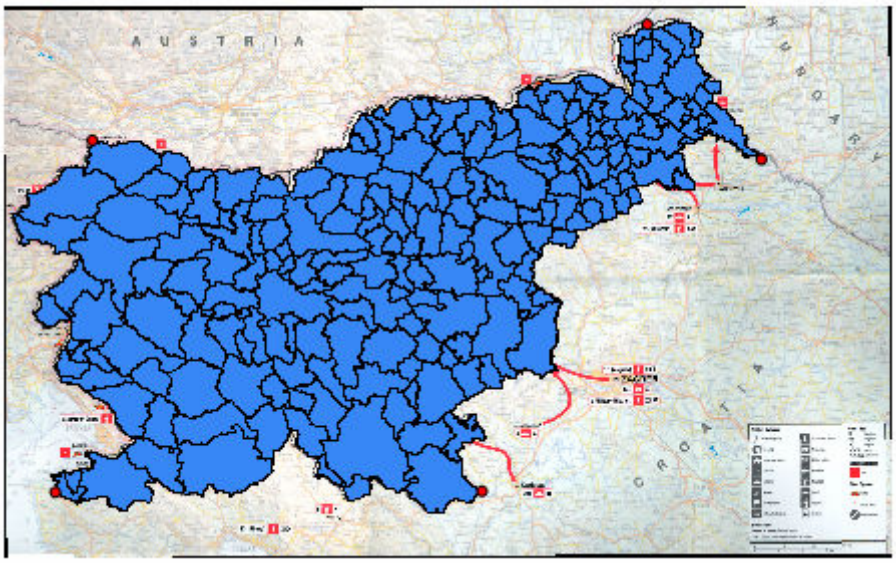
Karta ima poleg tematske vsebine vrisano tudi slovensko državno mejo, ki jo lahko uporabimo za povezavo z vektorskim slojem, ki je že prikazan v pravilnih koordinatah. Za vektorski sloj vzamemo prikaz slovenskih občin, ki je prosto dostopen prek spleta. Zunanja meja obmejnih občin je hkrati tudi državna meja, kar uporabimo za definiranje skupnih točk. Vektorski sloj občin odpremo v standardnem oknu aplikacije izven okna georeferenciranja.



Slika 3: Postopek georeferenciranja v programu Quantum GIS

Postopek georeferenciranja je sledeč. V oknu za georeferenciranje kliknemo na ikono »vstavi točko«. Na rastrski karti najdemo točko, ki jo bomo zlahka našli tudi na vektorski karti. Nanjo kliknemo in v pogovornem oknu izberemo možnost »Iz prikaza zemljevida«. Če bi poznali koordinate te točke, bi jih v istem pogovornem oknu lahko vnesli. Okno za georeferenciranje se v našem primeru umakne, saj je potrebno isto točko najti še na vektorskem sloju. Ko jo najdemo, nanjo kliknemo. Ponovno se odpre okno georeferenciranja z pogovornim oknom, v katerem kliknemo gumb »OK«. Novodoločena točka se pojavi tudi v primerjalni tabeli znotraj okna za georeferenciranje. Ta prikazuje slikovne koordinate rastra in koordinate v koordinatnem sistemu vektorskega sloja, kjer smo kliknili na skupno točko. Po postopku georeferenciranja se izračunajo še odstopanja posameznih točk.

Ko smo končali z zajemom točk (poizkušamo zajeti čimveč skupnih točk za boljši rezultat, potrebne pa so najmanj štiri točke), s klikom na ikono »Začni georeferenciranje« sprožimo postopek georeferenciranja. Odpre se pogovorno okno, kjer lahko izbiramo med različnimi vrstami transformacije, metodami prevzorčenja, morebitno stiskanje, izhodni sloj, koordinatni sistem za izvoz in nekatere druge parametre. Parametre potrdimo in izdela se nov georeferenciran rastrski sloj, ki se sedaj ujema s vektroskim slojem občin, na katerega smo se sklicevali s skupnimi točkami.



Slika 4: Primerjava vektorske in rastrske podobe kot rezultat georeferenciranja v aplikaciji Quantum GIS

7.5.2 Zajem vektorskih slojev iz rastrske podlage

Na podlagi zgoraj georeferencirane rastrske podlage želimo zajeti tematsko vsebino karte v vektorsko obliko. Zajemali bomo začetni položaj vojaških enot v točkovni sloj in premike enot v vektorski sloj. Nov vektorski sloj dodamo z klikom na ikono »nov sloj (shp)«. Pojavi se okno, v katerem izberemo tip sloja (točka, linija, poligon), koordinatni sistem sloja in sloju dodamo poljubne atribute. V našem primeru najprej izberemo točkovni tip v državnem koordinatnem sistemu (EPSG:3794), ter dodamo atribute za ime, tip in velikost vojaške enote, ki jo bomo zajeli. Nastavitve potrdimo in shranimo nov sloj, ki se že prikaže v zavihku z sloji aplikacije.

Ko je sloj ustvaren pričnemo z zajemom. Označimo novoustvarjeni sloj v levem drevesnem meniju, kjer so nanizani sloji. Iz menijske vrstice izberemo gumb »Omogoči urejanje«. Sedaj postanejo aktivni gumbi za urejanje, izmed katerega izberemo »capture point«. S tem smo omogočili funkcijo za zajem točk iz rastrske podlage. Za zajem točke kliknemo na željeno mesto na rastrski podlagi. Odpre se nam pogovorno okno za vnos ustreznih atributov točke, ki jih vnesemo. Po potrditvi je nova točka dodana.

Če želimo označiti več enot na isti lokaciji, lahko uporabimo funkcijo »lepljenja« (snapping). Ta sicer deluje avtomatsko, če pa ga ne želimo upoštevati, moramo ob kliku tiščati tipko »ctrl« na tipkovnici. Nastavitve lepljenja lahko spremenimo tako, da v meniju kliknemo na »nastavitve« in potem »možnosti«. Odpre se pogovorno okno, v katerem najdemo zavihek »risanje«. V okviru »lepljenje« nastavimo poljubne nastavitve.

Ko zajamemo vse zajete enote, vnos shranimo z klikom na ikono »shrani spremembe«. S ponovnim klikom na »omogoči urejanje« prekinemo geometrijsko urejanje slojev in nov sloj je ustvarjen in pripravljen za prikaz ali nadaljne delo.

Za zajem linijskih objektov sledimo podobnemu postopku, le da namesto točk zajemamo zaporedne točke med segmenti linije (lomljenke). Zajem posamezne linije zaključimo z desnim klikom, s čemer označimo končni točko linije.

7.6 Uvoz in priprava prostorskih podatkov v aplikaciji GeoServer

V okviru naloge je uporabljen GeoServer verzije 2.0.0. GeoServer je v svoji osnovi strežnik, zato ga je potrebno pred uporabo zagnati »v ozadju«. Ko je GeoServer zagnan, lahko do administratorskega vmesnika dostopamo prek spletnega brskalnika z uporabo ustrezne URL-povezave (npr: http://localhost:8080/geoserver/web/ za lokalni strežnik). Za vstop potrebujemo uporabniško ime in geslo, saj je kot stran strežnika administratorski vmesnik dostopen tudi iz spleta. Ob začetni uporabi do spremembe sta uporabniško ime in geslo »admin« in »geoserver«.

Ob vstopu v administratorski vmesnik v levem meniju opazimo glavne skupine funkcij strežnika. Pod zavihkom »Server« nastavljamo funkcije strežnika, pod zavihkom »Services« nastavljamo funkcije spletnih servisov, pod zavihkom »Data« pa upravljamo s podatki. Zavihka »Demos« in »Layer preview« omogočata ogled zgledov in predogled pripravljenih prostorskih podatkov.

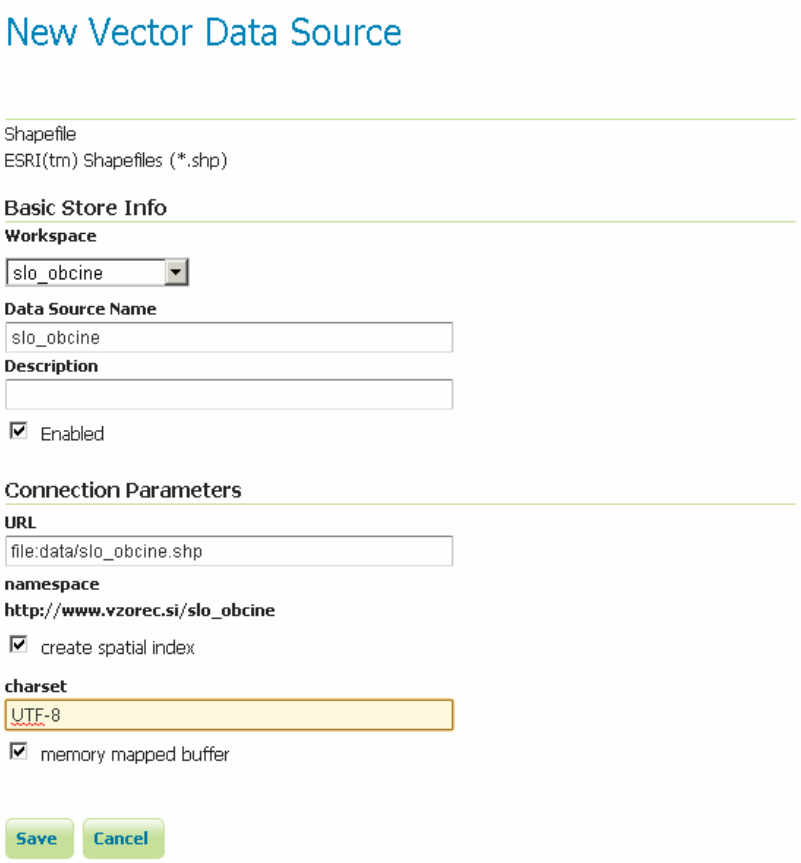
Posamezni uvoženi prostorski sloji so hierarhično zloženi v »kontejnerje« - Workspaces in »Stores«. Kontejnerji so namenjeni združevanju sorodnih slojev, na primer v okviru skupnega projekta, namenjeni so torej organiziranju in urejenosti podatkov. Kontejner vsebuje lastnost URI (Uniform Resource Identifier), ki jo definiramo, ko ustvarimo nov kontejner. URI običajno sestavlja URL-povezavo ali domeno strežnika, na katerem je dostopen Geoserver (npr. www.vzorec.si), kateri dodamo enolično ime kontejnerja, običajno kar naziv.

»Stores« pa so tudi uporabni konstruktorji, ki združujejo sloje, ki se večinoma uporabljajo istočasno (na primer za prikaz iste karte). Konstruktor uporabi enake parametre povezave za podrejene sloje, kar omogoča, da se uporabnik poveže do določenega sklopa slojev zgolj v eni zahtevi in mu ni potrebno te zahteve ponavljati. Sloj se mora tako nujno sklicevati na nek kontejner »stores«, vsak kontejner vrste »stores« pa mora obvezno biti del natanko enega kontejnerja vrste »workspace«.

Kot primer uvoza sloja v GeoServer bomo uvozili vektorski prostorski sloj v formatu shape (.shp) slovenskih občin v državnem koordinatnem sistemu (EPSG:3787). Za uvoz novega sloja najprej izdelamo »workspace« kontejner, v katerega bomo kasneje namestili konstruktor in sam sloj. V levem meniju pod zavihkom »Data« kliknemo na »Workspaces«. Prikaže se seznam obstoječih kontejnerjev. Za izdelavo novega kontejnerja kliknemo na zgornjo povezavo »Add new workspace«. Odpre se vnosno okno, ki zahteva navedbo enoličnega naziva kontejnerja in URI povezavo. Navedemo naziv »slo_obcine_workspace« in URI »http://www.vzorec.si/slo_obcine«. (vnešen URI je izmišljen, ta naj bi sicer kazal na strežnik ali domeno, na katerem deluje GeoServer). URI naj po svojem pomenu nebi kazal na nobeno specifično datoteko, glavna zahteva je le, da je URI enoličen za vsak kontejner. V zahtevi strežniku GeoServer, ki se nanaša na nek sloj, moramo navesti identični kazalec na sloj. Ta kazalec sestavlja naziva konstruktorja »workspace«, dvopičje (:) in naziv sloja. (npr. workspace:sloj). GeoServer bo naziv konstruktorja samodejno zamenjal z URI parametrom, kar bo vidno v našem predstavitvenem primeru kot»http:/www.vzorec.si/slo_obcine:slo_obcine« (tudi sloj bomo namreč poimenovali z nazivom»slo_obcine«).

Ko je kontejner »workspaces« ustvarjen, ustvarimo še kontejner »Stores«. V levem meniju izberemo »Stores«. Pojavi se nam seznam obstoječih kontejnerjev te vrste. Nov kontejner dodamo z klikom na povezavo »Add new Store«. Pojavi se seznam tipov vektorskih in

rastrskih podatkov, ki jih kontejner lahko vsebuje. Izmed vektorskih tipov izberemo tip »Shapefile«. Odpre se nova stran za nastavitev parametrov. Izberemo kontejner »workspace«, na katerega želimo navezati kontejner »store«, v našem primeru »slo_obcine«. Vnesemo še naziv kontejnerja. V sklopu »Connection Parameters« pod parametrom URL nastavimo pot do sloja. Če sloj umestimo v podmapo »data_dir\data« aplikacije GeoServer, potem lahko zapišemo pot do sloja kot »file:data/slo_obcine.shp«. Če v osnovni mapi ustvarimo še kako podmapo, jo ustrezno dodamo v URL. Pomembno je, da povsem na dnu spremenimo »charset« iz »ISO-8859-1« v »UTF-8« (oziroma ga uskladimo s shape datoteko), saj nam v nasprotnem črk s strešicami pri uporabi spletne karte ne bo prikazovalo pravilno (slika 3).



Slika 5: Vmesnik aplikacije GeoServer za uvoz shape datoteke

Kliknemo na »save« in shranimo nov kontejner. Aplikacija nas preusmeri še na nastavitev v URL-povezavi definiranega sloja. V sklopu »Coordinate Reference Systems« preverimo ali je prebran ustrezni koordinatni sistem. V obeh SRS okvirih je označen EPSG:3787, kar je pravilno. V sklopu »bounding boxes« s klikom na »Compute from data« in »Compute from native bounds izračunamo okvir največjih in najmanjših vrednosti v metrih (native) in v geografski širini in dolžini (Lat/Lon Bounding Box). Izračun shranimo. Prikaz sloja si lahko v različnih izvoznih formatih ogledamo v »Layer preview«.

Če želimo v GeoServerju ponastaviti drugačne parametre transformacije, ki je sicer shranjena kot del knjižnice koordinatnih sistemov, moramo v mapi »data_dir« in nadalje v mapi »user_projections« ustvariti datoteko »epsg_overrides.properties«. V tej datoteki vpišemo EPSG-kodo koordinatnega sistema, ki ga želimo modificirati, pripišemo enačaj in izpišemo WKT koordinatnega sistema v novi obliki:

```
3787=PROJCS["MGI / Slovene National
Grid",GEOGCS["MGI",DATUM["Militar_Geographische_Institute",SPHEROID["Bessel
1841",6377397.155,299.1528128,AUTHORITY["EPSG","7004"]],TOWGS84[577.326,90.
129,463.919,5.137,1.474,5.297,2.4232],AUTHORITY["EPSG","6312"]],PRIMEM["Gre
enwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.0174532925199433,AUTHOR
ITY["EPSG","9108"]],AUTHORITY["EPSG","4312"]],UNIT["metre",1,AUTHORITY["EPS
G","9001"]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin
",0],PARAMETER["central_meridian",15],PARAMETER["scale_factor",0.9999],PARA
METER["false_easting",500000],PARAMETER["false_northing",-
5000000],AUTHORITY["EPSG","3787"],AXIS["Y",EAST],AXIS["X",NORTH]]
```

Pod lastnostjo TOWGS84 zamenjamo parametre z novimi. Aplikacijo znova zaženemo in sloju ponovno izračunamo koordinatni okvir in izračun shranimo. Sloj se bo prikazoval transformiran po novih parametrih.

7.7 Uporaba JavaScript knjižnice OpenLayers

7.7.1 Odločitev o formatih prikaza

Ko imamo pripravljeno vso vsebino v vektorskih in rastrskih slojih v ustreznih koordinatnih sistemih, lahko pričnemo razmišljati o vključitvi prostorskih podatkov v interaktivno karto. Interaktivnost spletne karte omogočimo z JavaScript knjižnico OpenLayers. OpenLayers omogoča branje različnih prostorskih podatkov, ki jih dinamično prikazuje v prikaznem oknu znotraj spletnega brskalnika. Znotraj prikaznega okna lahko:

- prek orodnih gumbov ali gumbov na miški spreminjamo globino pogleda (zoom),
- prek orodnih gumbov ali gumbov na miški spreminjamo lego pogleda,
- prek orodnega menija prikazujemo ali skrivamo sloje,
- izpisujemo vsebinske in matematične parametre karte in slojev (npr. koordinate, atribute...),
- odpiramo predstavitvena okna podatkov (pop-up) na klik,
- in uporabimo še mnogo drugih dinamičnih funkcij.

OpenLayers knjižnica omogoča branje široke palete različnih tipov spletnih servisov prostorskih podatkov in datotek prostorskih podatkov. Omenimo le oblike branja prostorskih podatkov, ki zadevajo to nalogo:

- povezava na API-knjižnice spletnih ponudnikov spletnih kartografskih podlag (Google maps...)
- povezava na WMS in WFS-spletne servise prostorskih podatkov (posredovane z npr. GeoServer aplikacijo)
- branje GML in KML-datotek

Kljub široki paleti podprtih formatov knjižnice OpenLayers, velja podpora le standardnim prostorskim izmenjalnim formatom, ne pa tudi formatu kot je na primer shape format za vektorske podatke, v katerem smo pripravljali podatke v Quantum GIS aplikaciji. Za pretvorbo pripravljenih podatkov v spletu primerne podatke uporabimo eno izmed dveh ali obe možnosti:

- v programu Quantum GIS izvozimo vektorske prostorske podatke v GML ali KML-datoteke,
- vektorske shape datoteke uvozimo v program GeoServer, in jih prikazujemo v obliki spletnega servisa.

Za enostavnejšo obliko spletnega portala je zagotovo bolj uporabna prva rešitev. Z neposrednim prikazovanjem GML/KML-datotek se izognemo uporabi GeoServerja, ki je zmogljiva aplikacija, ki kot strežnik pripravlja različne oblike prostorskih prikazov v izmenjalnih formatih po navodilih, ki jih je prejel prek zahteve uporabnikov. GML/KML-datoteke namreč uporabnik pridobi iz strežnika, in jih procesno preoblikuje na svojem računalniku v okviru funkcij OpenLayers. Za grafično pretvorbo je tako uporabil zmogljivosti svojega računalnika. V primeru rešitve z GeoServer strežnikom pa uporabnik zgolj pošlje zahtevo po določenem prostorskem podatku določene vrste in oblike, GeoServer pa jo s pomočjo strojne zmogljivosti strežnika pripravi in vrne uporabniku v končni obliki. Slednje seveda zahteva večje procesne sposobnosti strežnika. GeoServer tako ni rešitev, ki bi jo uporabili zaradi spletne strani same, temveč bolj zaradi obsežnejših baz podatkov, ki bi jih želeli posredovati v spletno okolje. GeoServer pa v tem kontekstu vseeno omenjamo, saj s svojimi servisi omogoča razširjeno možnost uporabe.

7.7.2 Osnovna vzpostavitev karte

OpenLayers knjižnico umestimo v spletno stran z HTML-značko »script«, kot je običajno za vključitev JavaScript kode:

```
<script src="openlayers/OpenLayers.js" type="text/JavaScript" />
```

Če uporabimo celotno knjižnico z vsemi funkcijami, zavzame knjižnica relativno veliko prostora, kar podaljša njeno inicializacijo. Zato je možno knjižnico tudi skrčiti na zgolj tiste funkcije, ki jih v interaktivni karti predvidimo za uporabo.
Karto v HTML-dokumentu aktiviramo z naslednjo kodo:

```
map = new OpenLayers.Map({
            div: "map",
            controls: [new OpenLayers.Control.Navigation(),
                        new OpenLayers.Control.PanZoomBar(),
                  new OpenLayers.Control.MousePosition(),
                  new OpenLayers.Control.ZoomBox()
                  ],
                  numZoomLevels: 8,
            minZoomLevel: 10,
            maxExtent: new OpenLayers.Bounds(1570000, 5720000, 1590000,
5754375)
        });
```

V kodi definiramo, da naj se karta prikaže v HTML-elementu z id-parametrom »map«, omogočimo funkcije za karto. Določimo lego in globino začetnega pogleda na karto ter meje karte, do koder je možno karto pregledovati.

Karti nato dodamo poljubne sloje. Najprej definiramo sloj Google Maps, ki bo predstavljal podlago naši vsebini. Knjižnica ima že pripravljen objekt za vključitev googlovih kart, zato ga lahko enostavno definiramo takole:

```
var gphy = new OpenLayers.Layer.Google(
```

```
                "Google fizična karta",
                {type: G_PHYSICAL_MAP, 'sphericalMercator': true}
            );
var gsat = new OpenLayers.Layer.Google(
                "Google hibridna karta",
                {type: G_HYBRID_MAP, 'sphericalMercator': true}
            );
```

Definirali smo dva sloja googlovih kart, za vsak tip karte (fizična in hibridna) svoj sloj. Nato definiramo vsebinski sloj v obliki KML-datoteke:

```
sundials = new OpenLayers.Layer.Vector("Objekti Slovenske vojske", {
                projection: map.displayProjection,
                strategies: [new OpenLayers.Strategy.Fixed()],
                protocol: new OpenLayers.Protocol.HTTP({
                    url: "kml/objekti_SV.kml",
                    format: new OpenLayers.Format.KML({
                        extractStyles: true,
                        extractAttributes: true
                    })
                })
            });
```

Izdelali smo nov vektorski sloj na podlagi prebrane KML-datoteke, prevzeli pa smo tudi v datoteki nastavljene atribute.

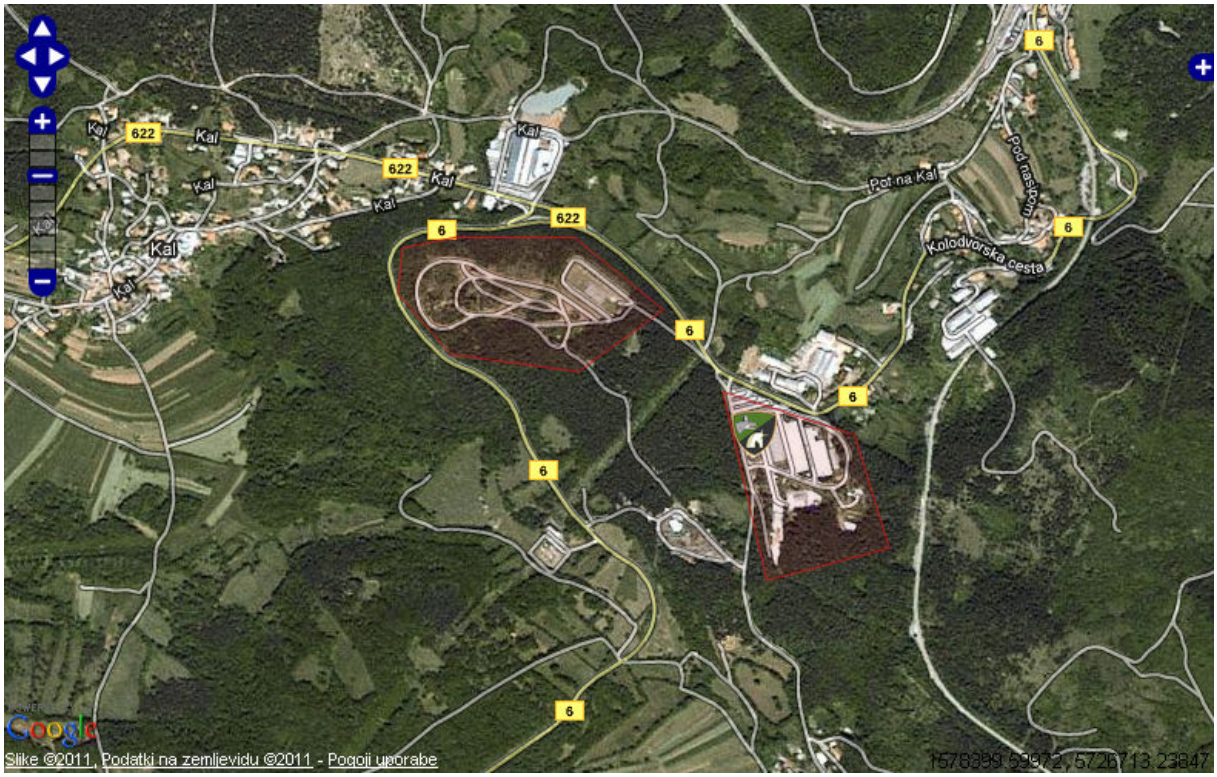
Naslednji vsebinski sloj definirajmo še v tretji obliki povezave:

```
var oklepni_bataljon = new OpenLayers.Layer.WMS("Vojaški muzej",
            "http://www.vzorec.si/geoserver/wms",
            {'layers': 'slovenska_vojska:oklepni_bataljon', transparent:
true, format: 'image/gif'},
            {isBaseLayer: false}
        );
```

Za definiranje sloja smo uporabili WMS-servis, prek katerega smo zahtevali vnaprej pripravljen sloj. Zahtevo smo poslali strežniku Geoserver, ki je na zahtevo pripravil sloj v obliki podobe v gif-formatu. Definirane sloje dodamo v ustvarjeno mapo z naslednjim ukazom:

```
map.addLayers([gsat, gphy, sundials, oklepni_bataljon]);
```

Ob zagonu spletne strani se nam znotraj strani prikaže dinamična karta (slika 4).



Slika 6: Prikaz okna OpenLayers s prikazanimi sloji

Na sliki 4 lahko prepoznamo vse dodane elemente. Karta je prikazana torej smo jo uspešno ustvarili. Dodana so orodja za navigacijo zgoraj levo in funkcija za izbor slojev zgoraj desno (bel »+« na modri podlagi, ki se razširi ob kliku). Kot podlago prepoznamo googlovo hibridno karto, ki prikazuje ceste in ulice na satelitskih podobah. Viden je tudi rdeč ploskovni sloj dodan iz KML datoteke in točkovna oznaka matične vojašnice oklepnega bataljona oblikovana kot grb enote.

7.7.3 Dinamična povezava karte z vsebino

Ker so zgolj slikovni elementi na karti premalo informativni, omogočimo na karti pojavna okna (popup). Želimo si, da bi ob kliku na vsebinski sloj uporabnik dobil povratno informacijo o prikazanem objektu v obliki pojavnega okna z vsebino in povezavami za nadaljne branje. To storimo z vključitvijo nekaj funkcij in objektov knjižnice v kodo. Ustvarimo objekt »popup« in definiramo dinamične funkcije objektom (klik na sloj, zaprtje prikaznega okna...). Definiramo tudi vsebino pojavnega okna v HTML-obliki. Ta vsebina je sestavljena iz HTML-elementa iframe, ki prikaže HTML-datoteko, ki se nahaja izven odprtega dokumenta. Vsebine HTML-dokumentov pojavnih oken, ki se odprejo znotraj elementa iframe pripravimo vnaprej. Pri izdelavi sloja poskrbimo, da ima vsak element sloja definiran atribut »name«. Te atribute označimo enako kot HTML-dokumente, ki se odprejo v pojavnih oknih. Z dinamično izdelavo elementa iframe tako zagotovimo, da se ob kliku na določen element sloja odpre njemu pripadajoč HTML-dokument v pojavnem oknu:

```
var vsebina = "<iframe width='300px' height='200px' scrolling='no'
src='strani/slovenska_vojska/" + feature.attributes.name + ".html'
></iframe>"
```

Definira se različen HTML-element iframe glede na to, na kateri element sloja smo kliknili.

Slika 7: Pojavno okno znotraj interaktivne karte

Ker želimo dinamično karto bolj povezati z celotno spletno stranjo, omogočimo dinamične spremembe na karti tudi s pomočjo dinamičnih povezav, ki se nahajajo v meniju spletne strani. Na ta način omogočimo, da prikaz na karti sledi navigiranju uporabnika po spletni strani. Če uporabnik iz menija izbere določeno vsebinsko tematiko, karta samodejno prikaže sloje obravnavane tematike. Če uporabnik v okviru določene vsebinske tematike izbere točno določen objekt na karti, se karta s pogledom osredotoči nanj. S tem zagotovimo vsaki vsebinski podtemi svoj pogled na karto. Omenjeno dinamičnost dosegamo z funkcijami prikazovanja in skrivanja slojev, ter centriranje karte:

```
if (kart == "SV")
                {
                        sundials.setVisibility(true);
                        enote_SV.setVisibility(true);
                        map.setCenter(new OpenLayers.LonLat(1580000
,5737187.5),1);
                }
```

8 ZAKLJUČEK

Koncept rešitve interaktivne strani optimizirane za spletne iskalnike v okviru te naloge ne gre jemati v smislu edine in najboljše možne rešitve, saj bi zagotovo našli bolj dognane in dovršene koncepte dinamičnih spletnih strani. Koncept rešitve je le grobo zastavljen model delovanja, ki je lahko podlaga za nadaljno optimizacijo koncepta še predvsem v smislu lažje uporabnosti, enostavnosti in bolj modularne izvedbe. Kljub vsemu koncept dobro oriše širok spekter težav, s katerimi se še sooča tehnologija dinamičnih strani in nakazuje nekatere možne rešitve.

Pristop izenačevanja vsebinskih spletnih strani in interaktivnih spletnih prikazov je sicer konkretno predstavljen v vojaško-zgodovinski tematiki, a je lahko uporaben za vrsto drugih tematskih vsebin, in časovnih obdobij, bodisi da gre za preteklost, sedanjost in prihodnost, kot na primer:

- turistične vsebine (portal turizma nekega območja),
- tehnična dediščina (razvoj železnic, prikaz območjih značilne tehnične dediščine),
- naravna dediščina (portal narodnih, krajinskih in drugih območij narave),
- ljubiteljski portali (kolesarski, motoristični, izletniški, planinski, gobarski portali...),
- prikaz prihodnosti (prikaz razgrnitev, poteka tras...),
- itd.

Uporabljen pristop izenačevanje nivoja vsebine in interaktivnih spletnih prikazov je težko ocenjevati brez dejanske izvedbe koncepta v praksi. Težko je oceniti, ali bi bil tovrsten pristop uporabniku bolj prijazen in privlačen od obstoječih tematskih spletnih strani z vsebinsko bogatimi, a statičnimi vsebinami in na drugi strani zmogljivih interaktivnih kartografskih portalov, ki se ne usmerjajo v izdelavo vsebine. Tovrstni portali so pogosto namenjeni zagotavljanju kartografskih podlag in orodij v podporo razvijalcem drugih spletnih vsebin, pogosto omogočajo tudi vsebinsko sodelovanje in kreativnost obiskovalcem, ali pa so le podlaga najrazličnejšim prostorskim podatkom, ki pa ostanejo brez jasne vsebinske podpore in temeljitejše razlage.

Dejstvo je, da je razkorak med bogatimi statičnimi vsebinskimi portali in zmogljivimi dinamičnimi kartografskimi portali še vedno občuten, kar predstavlja nišo za bodoče ponudnike spletnih storitev in vsebin. Opisan predlog v tej nalogi je morda eden od bodočih konceptov, ki bi ob združevanju že obstoječih uporabniških kvalitet ustvaril nove in morda še bolj napredne spletne rešitve.

VIRI:

Apache strežnik. 2011.
http://www.apache.org/ (pridobljeno avgusta 2011)

Bilke, P. 2002. Spoznajmo PHP in MySQL. Nova Gorica, Flamingo: 112 str.

Bride, M. 2003. JavaScript. London, Hodder Headline: 179 str.

Davis Scott. 2007. GIS for web developers. Raleigh, Pragmatic Bookshelf: 254 str.

Geoserver. 2011
http://geoserver.org (pridobljeno avgusta 2011)

HTTP protokol. 2011.
http://sl.wikipedia.org/wiki/HTTP (pridobljeno september 2011)

Keith Jeremy. 2006. Hijax: Progressive Enhancement with Ajax. Amsterdam, XTech 2006.
http://domscripting.com/presentations/xtech2006/ (pridobljeno julija 2011)

Kovačič Iztok. 2005-2006. Uvod v podatkovni standard XML, študijsko gradivo pri predmetu Avtomatska obdelava podatkov.

Kozierok M. Charles. 2003-2005. The TCP/IP Guide.
www.tcpipguide.com/free/t_TCPIPApplicationLayerAddressingUniformResourceIden.htm (pridobljeno avgust 2011)

Making Ajax Applications Crawlable. 2011.
http://code.google.com/intl/sl-SI/web/ajaxcrawling/docs/learn-more.html (pridobljeno avgusta 2011)

Open Geospatial Consortium. 2011.
http://www.opengeospatial.org/ (pridobljeno avgusta 2011)

OpenGeo, GeoServer concepts.
http://workshops.opengeo.org/suiteintro/geoserver/intro/concepts.html (pridobljeno septembra 2011)

Pavlič Luka in drugi. 2002. Zakaj in kako izgraditi spletni portal? Zbornik Elektrotehniške in računalniške konference ERK: 147 – 150
http://164.8.251.136:8080/lp/pages/sl/publics/jetspeed/jetspeed.pdf (pridobljeno julija 2011)

PHP. 2011.
www.php.net (pridobljeno avgusta 2011)

Rolih R. 2000. Kako dobičkonosno poslovati preko interneta. Ljubljana, Lisac & Lisac d.o.o, 114f.

Sebesta R.W. 2010. Programming the World Wide Web 2009, Upper Sadle River, Pearson: 752 str.

Spletni GIS servisi.
http://webhelp.esri.com/arcgisserver/9.3/java/index.htm#what_can_you_publish.htm
(pridobljeno septembra 2011)

Šumrada R. 2005. Tehnologija GIS. Ljubljana, Univerza v Ljubljani, Fakulteta za gradbeništvo in geodezijo: 330 str.

Šumrada R. 2011. Prosti standardni spletni servisi OGC za prostorske podatke. Geod. vestn., 55, 1: 46-56.

W3schools. 2011.
http://www.w3schools.com/ (pridobljeno avgusta 2011)

XAMPP aplikacija. 2011.
http://www.apachefriends.org/en/xampp.html (pridobljeno avgusta 2011)

Zaveršnik Matjaž. CSS.
http://zaversnik.fmf.uni-lj.si/Gradiva/CSS/ (pridobljeno septembra 2011)

Zaveršnik Matjaž. HTML.
http://zaversnik.fmf.uni-lj.si/gradiva/html/index.php (pridobljeno septembra 2011)