Univerzitet u Beogradu
Matematički fakultet

Master rad

Alat za vizualizaciju geoprostornih podataka na vebu

Ivan Merdović

Beograd, 27.08.2014.

Autor Ivan Merdović

Mentor dr Saša Malkov, docent
Univerzitet u Beogradu - Matematički fakultet

Članovi komisije dr Nenad Mitić, vanredni profesor
Univerzitet u Beogradu - Matematički fakultet

dr Mladen Nikolić, docent
Univerzitet u Beogradu - Matematički fakultet

Datum odbrane

Apstrakt

Vizualizacija podataka je sredstvo za efikasnije praćenje, razumevanje i uočavanje određenih pravila u predstavljenim podacima i njihovim promenama. Pregledom postojećih rešenja u domenu prikaza geografski određenih (geoprostornih) podataka na prostoru Republike Srbije uočljiv je prostor za napredak u vizualnom, funkcionalnom i tehničkom smislu. Problem je u velikoj meri zajednički i ne zavisi od tipa podatka. Cilj rada je sagledavanje karakteristika problema i razvoj demonstracionog veb alata koji bi unapredio postojeće stanje na ovom polju i pružio osnovu za dalje unapređenje. U funkcionalnom smislu, podaci se mogu pregledati u različitim prikazima integrisanim u uobičajenu kartografsku notaciju, što je u postojećim rešenjima ili nemoguće ili strogo ograničeno, zbog čega se smisao upotrebe mape svodio samo na prikaz lokacije merne stanice podatka. Alat se oslanja na nove tehnologije prikaza, kombinujući HTML5, JavaScript i SVG za prikaz mapa i podataka ali uzimajući u obzir i različite uređaje i platforme za pregledanje veb sadržaja.

Sadržaj

1 Uvod 6
1.1 Geovizualizacija 7
1.2 Motivacija 8
1.2.1 Primer 1 - Hidrometeorološki zavod Srbije (HMZS) 9
1.2.2 Primer 2 - Državna mreža za automatski monitoring kvaliteta vazduha (SEPA) 10
1.2.3 Alternativa i poboljšanje - WebMap 11
2 Analiza problema i definisanje zahteva 13
3 Idejno rešenje alata WebMap 15
3.1 Arhitektura 15
3.2 Model podataka 16
4 Implementacija alata WebMap 19
4.1 Tehnologije i alati 19
4.1.1 XML 20
4.1.2 JSON i GeoJSON 21
4.1.3 Skalabilni vektorski crteži 22
4.1.4 D3.js 24
4.1.5 Modularni JS 28
4.1.6 PHP i okruženje CodeIgniter 29
4.2 Detalji implementacije 30
4.2.1 Struktura datoteke izmerenih vrednosti 31
4.2.2 Upravljanje SVG mapom i reprezentacijom podataka 32
4.2.3 Modul Google mape 35
4.2.4 Modul korisničkog interfejsa 35
5 Primeri upotrebe 37
5.1 Podaci potrebni za prikaz karte 37
5.1.1 Natural Earth 37
5.2 Primer prikupljanja vrednosti vodostaja sa stranica HMZS 39
5.2.1 Prikupljanje osnovnih podataka o mernim stanicama 39
5.2.2 Prikupljanje vrednosti merenja 40

5.3 Primer mape sa više mernih parametara 41

6 Diskusija i zaključak 42

7 Dodatak 45

7.1 Podešavanje projekcije i crtanje karte 45

7.1.1 Podešavanje projekcije 45

7.1.2 Crtanje karte 46

7.2 Vizualizacija kružićima 47

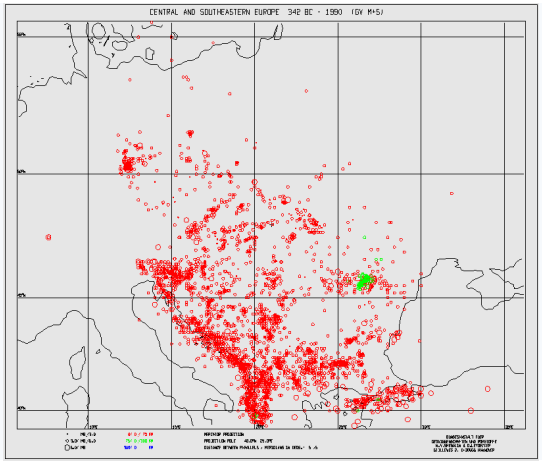
7.3 Vizualizacija pravougaonicima 48

8 Literatura 50

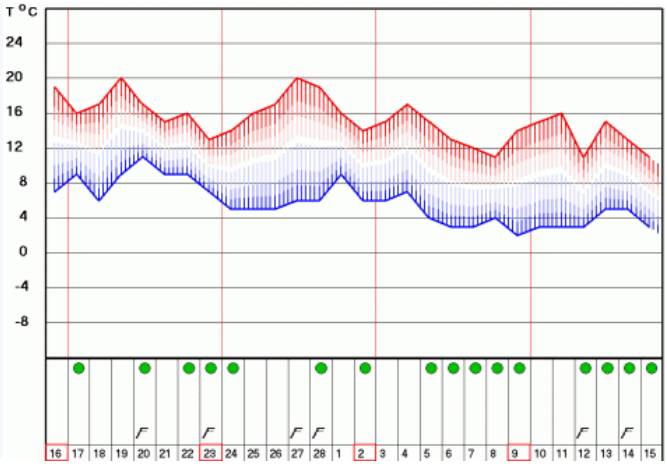
1 Uvod

Slika govori više od hiljadu reči. Može se reći da je vizualizacija podataka direktna potvrda ove široko korišćene i prihvaćene teze. Ljudi se u velikoj meri oslanjaju na svoju sposobnost vizualne percepcije. Ljudski um je sposoban da na različite načine vizualnim putem obradi i prihvati prikazane podatke, da razume odnose, poredak i postojanje veza u zavisnosti od načina na koji su mu podaci prezentovani. Subjektivnost igra veliku ulogu u sferi vizualnog. Različiti ljudi imaju različite poglede na istu stvar. Baš zbog te fleksibilnosti i gotovo neograničenog broja načina vizualne interpretacije podataka, vizualizacija nije u svakom slučaju poboljšanje. U nekim slučajevima je nepotrebna, dok u nekim sem estetskog poboljšanja ne donosi ništa novo u smislu razumevanja samih podataka, što je zapravo i njen najveći motiv. Kao i drugi vidovi prezentovanja informacija i vizualizacija podataka je pogodna za manipulaciju i navođenje na pogrešne ili ciljane zaključke onoga ko podatke predstavlja (namernim ili slučajnim izborom pogrešnog načina prikaza, isticanjem pogrešnih vrednosti i manje bitnih informacija itd.).

Živimo u vremenu u kome smo okruženi različitim informacijama. Primeri vizualizacije su svuda oko nas. Ko će pobediti na izborima, koliko je čist vazduh koji udišemo, koje zemlje da izbegavamo ako se plašimo zemljotresa i kakvo nas vreme očekuje sutra... Sve su to informacije koje ne čitamo red po red, vrednost po vrednost već pogledom na odgovarajuću grafičku predstavu. Neki primeri vizualizacija predstavljeni su na slikama 1 i 2.



Slika 1 - Mapa epicentara zemljotresa centralne i jugoistočne evrope u periodu od 342 PNE do 1990. http://www.bgr.bund.de/EN/Themen/Seismologie/Bilder/Sei_semap_g.html



Slika 2 - Izgledi vremena za Beograd i širu okolinu za period 6.2. - 15.3.2014 http://www.hidmet.gov.rs/ciril/prognoza/beograd.php

Sam proces vizualizacije je po definiciji u izdanju *Visualizing Data* [1] podeljen u sedam segmenata:

1. Prikupljanje podataka
2. Parsiranje
3. Filtriranje
4. Istraživanje podataka (*Data mining*)
5. Reprezentacija
6. Prečišćavanje i usavršavanje reprezentacije
7. Interakcija

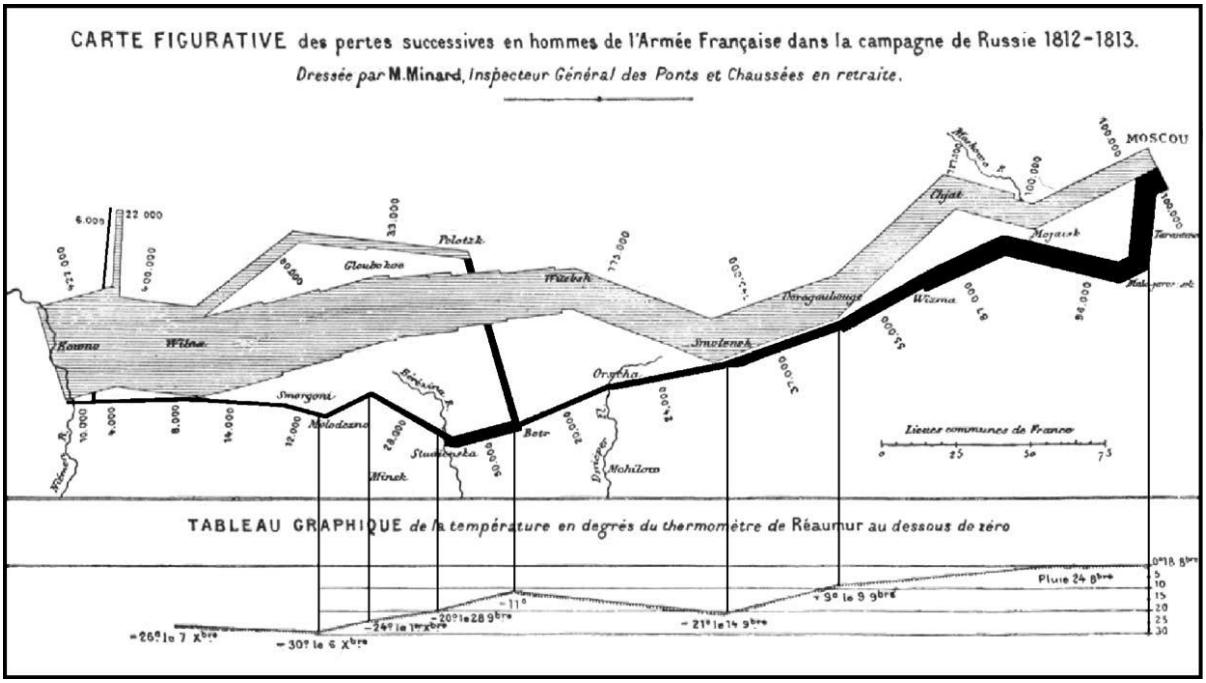
Vizualizacija podataka je proces koji zahteva bar delimično poznavanje različitih disciplina kako bi bio uspešan. Velika slika konačnog rezultata mora postojati na samom početku procesa - prilikom prikupljanja i obrade podataka. Jasno je da neko ko pretenduje da prikaže određene informacije mora prvo da razume šta one predstavljaju, mora da zna kako da te informacije prikupi, pripremi i među njima odabere one koje su relevantne za obradu. Nakon toga, u skladu sa svojom idejom autor mora da osmisli način prikaza, da ga doradi i usavrši u okviru procesa i na kraju omogući krajnjem korisniku informacija da interaguje sa njima.

1.1 Geovizualizacija

Geovizualizacija (od *geografska vizualizacija*) predstavlja skup alata i tehnika za grafičku reprezentaciju i analizu geoprostornih podataka (*geografski određenih podataka*) [2]. Pod pojmom geoprostornih podataka podrazumevaju se podaci koji su usko vezani sa određenom geografskom lokacijom. Prilikom proučavanja geoprostornih podataka često je poželjno postojanje interakcije između korisnika i podataka. Tradicionalne statičke karte i mape fiksnih veličina omogućavaju prikaz koji obično nije najpogodniji svakom korisniku, odnosno sadrže ili premalo ili previše informacija za određenu svrhu. Mogućnost interakcije sa prikazima, kontola veličine karte (*zoom*), prikaz različitih kombinacija podataka i izbor različitih načina prikaza su pogodnosti koje mogu omogućiti lakše razumevanje i pronalaženje potencijalnih pravilnosti među podacima.

Svaki geoprostorni podatak je višedimenzionalan. Odlikuje ga lokacija (jedna ili više koordinata i odgovarajući sistem), odlikuje ga vrednost samog podatka koji se prikazuje ali u većini slučajeva odlikuje ga i vreme za koji se podatak vezuje (najčešće je to vreme merenja).

Istorijski razvoj geovizualizacije počinje verovatno i mnogo pre definisanja samog pojma ali geovizualizacija veliku pažnju privlači tek sa razvojem informacionih tehnologija i geografskih informacionih sistema (GIS). Kao jedan od najčešće pominjanih istorijskih primera geovizualizacije pojavljuje se *Charles Minard*-ova mapa Napoleonovog pohoda na Rusiju 1812. godine [3], opisana kao remek delo i odličan primer prikaza - "*toliko dobro, da posmatrač neretko i ne shvata koliko različitih a usko povezanih dimenzija podataka posmatra u isto vreme*".



Slika 3 - Napoleonov Marš na Moskvu - Prikazani su gubici Napoleonove vojske prilikom ovog marša. Uz geografske lokacije i vremenski period uporedo su prikazani:(1) Broj ljudi u pohodu (numerički i vizualno, šrafirana putanja predstavlja napad, crna povlačenje, širina putanje oslikava broj ljudi), (2)Temperatura, (3) Značajne geografske tačke i nazivi velikih bitaka

Minardova mapa na slici 3 oslikava istinski značaj geovizualizacije i dublje shvatanje grupe usko povezanih podataka. Jasno je prikazan smer napada i smer povlačenja, vremenska promena broja ljudi u maršu, uticaj temperature na ljude u pohodu kao i geografski i istorijski podaci. Sa druge strane prikazani podaci nisu potpuno precizni ali je to uradjeno namerno, sa ciljem da se što bolje vizualno prikaže ono što je bitno, uz odstupanja koja ne utiču na smisao osnovne poente (npr. putanja povlačenja se u mnogo većoj meri poklapala sa putanjom napredovanja, ali je do odstupanja došlo zbog boljeg prikaza).

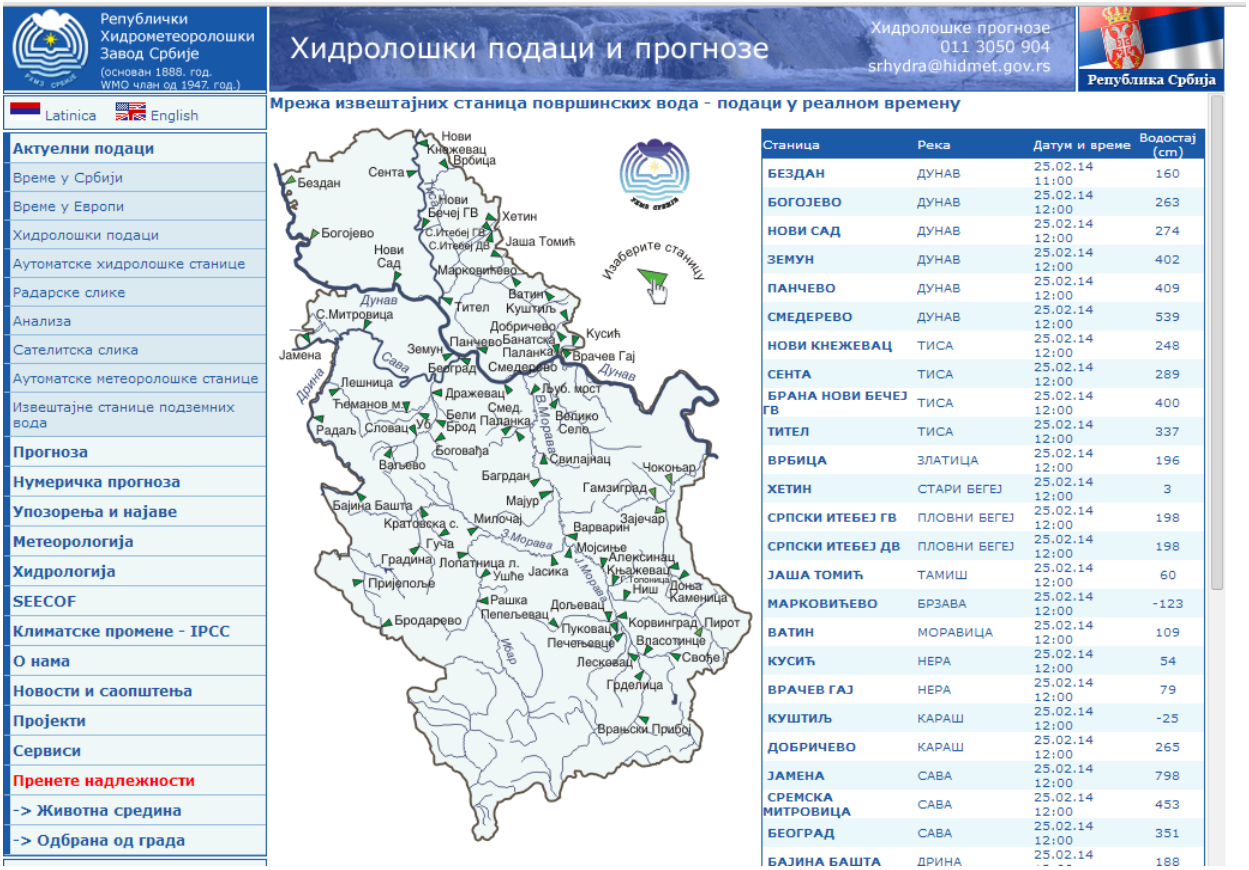
1.2 Motivacija

Jedan od osnovnih motiva ovog rada jeste da čitaocu približi pojam i značaj geovizualizacije. Živimo u vremenu u kom Internet predstavlja osnovni medij, što znatno olakšava pristup različitim informacijama. Okruženi smo aparatima za pregledanje informativnog sadržaja na Internetu počev od kućnih PC računara, mobilnih telefona, tableta pa sve do televizora i drugih aparata. Uopšteno gledano - podatke imamo u izobilju kao i mogućnosti da ih posmatramo i proučavamo na različite načine. Čak i ako suzimo izvore geoprostornih podataka na jedno određeno područje (u ovom slučaju na Republiku Srbiju) osnovnim pregledom javnih podataka možemo sagledati potencijlana poboljšanja po pitanju geovizualizacije istih. U tu svrhu predstavićemo trenutno stanje i trenutni prikaz javnih geoprostornih podataka na dva primera u Republici Srbiji:

1. Hidrometeorološki zavod Srbije (HMZS) - http://www.hidmet.gov.rs/
2. Državna mreža za automatski monitoring kvaliteta vazduha (SEPA) - http://www.sepa.gov.rs/

1.2.1 Primer 1 - Hidrometeorološki zavod Srbije (HMZS)

Iako ova veb strana predstavlja izvor različitih geoprostornih podataka, u ovom radu posmatran je samo podatak koji opisuje vodostaj na vodotokovima u Republici Srbiji. Na stranici "Aktuelni podaci" i podstrani "Automatske hidrološke stanice" predstavljeni su aktuelni podaci vodostaja na rekama u Srbiji.



Slika 4 - Veb stranica HMZS - Prikaz položaja mernih stanica i vrednosti vodostaja u desnom okviru (uz vreme merenja)

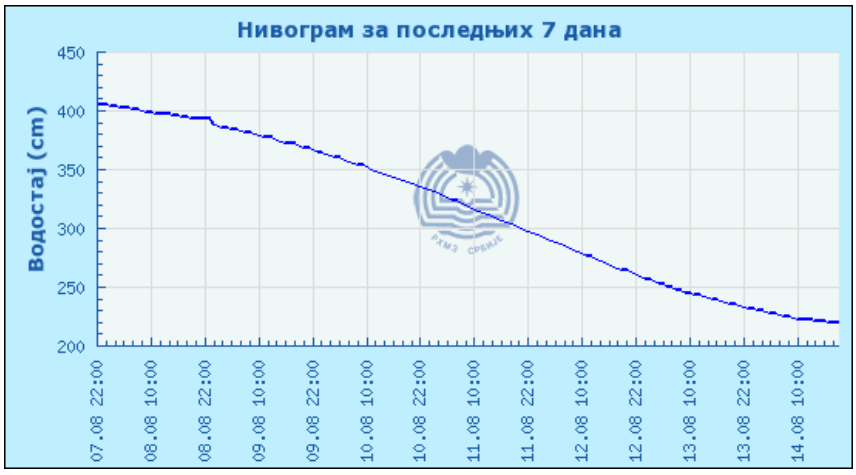
Analizirajmo šta nam je predstavljeno na slici 4:

- Možemo videti granice Republike Srbije i osnovne geografske odlike (vodotokove, nazive mesta odnosno mernih stanica)
- Vide se pozicije mernih stanica
- U okviru sa desne strane vidimo vrednosti i vremena merenja

Ono što nedostaje jeste podatak koji opisuje nivo vode na samoj mapi. Moramo da tražimo podatak u tabeli pa zatim njegovu poziciju na mapi. Ukoliko na primer želimo da pratimo odnose nivoa vode na istom rečnom slivu moramo da pogledamo na mapi koje stanice pripadaju istom slivu pa onda za svaku da tražimo vrednosti u tabeli desno. Postavlja se pitanje - koliko ovakvim načinom usvajanja informacija gubimo u smislu početne ideje? Uz ogradu da je ovakav prikaz informacija dobar za određene načine korišćenja, za veću fleksibilnost i lakše uočavanje podataka i njihovih odnosa na ovoj stranici postoji prostor za unapređenje (bar kao dodavanje nove opcije prikaza).

Stranica prikazana na slici 4 pokazuje, dakle, samo trenutne, najsvežije podatke. Ukoliko želimo da posmatramo istorijat merenih vrednosti na određenoj stanici onda moramo ući na njenu podstranu klikom na naziv željene merne stanice. Ovde možemo videti lep prikaz istorije merenja vodostaja u vidu linijskog dijagrama, predstavljenog na slici 5.

Jedno unapređenje koje bi značajno doprinelo informativnosti prezentacije istorije merenja bi mogla da bude upotreba animacije.



Slika 5 -Dijagram istorije izmerenih vrednosti vodostaja

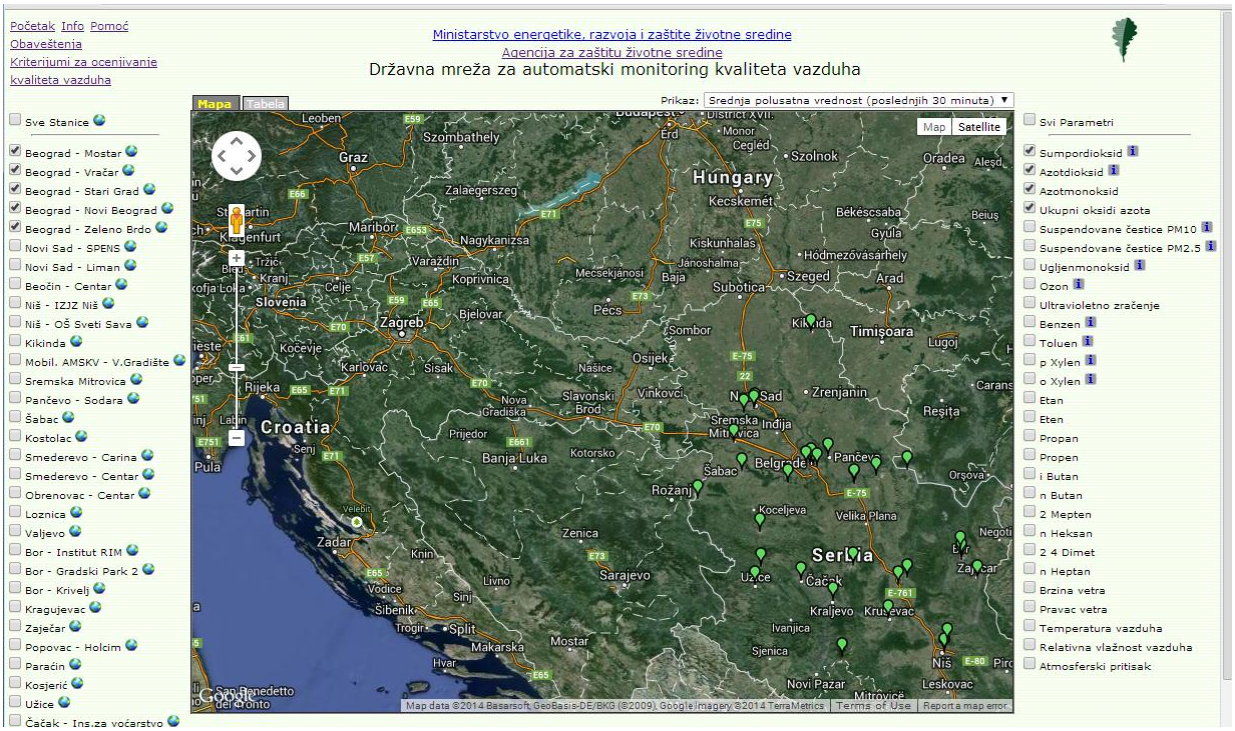
1.2.2 Primer 2 - Državna mreža za automatski monitoring kvaliteta vazduha (SEPA)

Drugi primer na koji ćemo ukazati jeste veb stranica "Državne mreže za automatski monitoring" kvaliteta vazduha (SEPA) koja pruža uvid u podatke koji predstavljaju prisutnost određenih hemijskih elementata i jedinjenja u vazduhu u svrhu kontrole kvaliteta vazduha. Za razliku od prethodnog primera, kad je mapa u suštini statična slika (HMZS primer), u ovom primeru kao podloga za prikaz lokacija stanica koristi se *Google Maps API* [4]. Korišćenjem Google mapa dobijaju se određene prednosti u pregledu u odnosu na prethodni primer sa statičnom slikom, ali na žalost ništa suštinski bitno za same podatke koji se mere i korisnika. Google mapa omogućava uvećanje mape, samim tim u mogućnosti smo da vidimo tačnu lokaciju merne stanice na mapi ili satelitskom snimku što je u suštini jedina prava prednost u odnosu na prethodni primer.

Slično kao sa prethodnim primerom analizirajmo šta nam je predstavljeno na veb stranici SEPA prikazanoj na slici 6:

- Možemo videti granice Republike Srbije i okolnih zemalja i detaljne geografske odlike
- Vide se pozicije mernih stanica, kao čiode na "Google mapi";
- U koloni sa desne strane možemo filtrirati merne podatke;
- U koloni sa leve strane možemo filtrirati merne stanice;

Imamo sličan problem kao u prethodnom primeru. Podaci se i dalje ne vide na samoj mapi. Do podataka stižemo klikom na link koji otvara pogled pod nazivom "Tabela" gde možemo pročitati vrednosti merenih podataka. Vizualnim pregledom ne možemo proceniti da li postoji veza između zagađenja i pojedinih geografskih odlika, da li neki regioni imaju veće koncentracije pojedinih jedinjenja od drugih, da li postoje grupisanje stanica po određenom kriterijumu i slično.



Slika 6 - Veb stranica SEPA - Prikaz položaja mernih stanica i količine jedinjenja izmerenih u vazduhu

1.2.3 Alternativa i poboljšanje - WebMap

Ideja je da se razvije alat primenljiv na različite tipove podataka - jedan alat koji bi u perspektivi mogli da koriste i HMZS i SEPA kao specifičan način prikaza svojih podataka (kao novu opciju ili zamenu za postojeće stanje). Osnovna motivacija razvoja alata WebMap jeste da omogućavi korisniku različite tipove prikaza. U zavisnosti od tipa merenog podatka

odnosno od toga da li su podaci sa HMZS ili SEPA stranice, različiti prikazi će manje ili više odgovarati prirodi samog podatka.

Podloga na kojoj će podaci i stanice biti prikazani ne treba da odvlači pažnju sa samih podataka. U skladu sa tim, podlogu će predstavljati samo kontura regiona, sa osnovnim geografskim odlikama (npr veliki tokovi). Ovo će značajno uticati i na optimizaciju kompleksnosti pri iscrtavanju što će doprineti bržem prikazu i lakšem rukovanju alatom.

2 Analiza problema i definisanje zahteva

U prethodnom poglavlju objašnjen je pojam geovizualizacije i njegov značaj kao i trenutno stanje i potencijalna poboljšanja na konkretnim primerima. Ranije pomenutih sedam segmenata vizualizacije u dobroj meri određuju tok razvoja od postavljenog problema do demonstracionog projekta pod nazivom WebMap. Da se podsetimo i kratko analiziramo navedene korake:

- **Obezbeđivanje i priprema podataka**
    1. Prikupljanje podataka
    2. Parsiranje
    3. Filtriranje
    4. Istraživanje podataka (*Data mining*)
- **Prikaz i interakcija**
    5. Reprezentacija
    6. Prečišćavanje i usavršavanje reprezentacije
    7. Interakcija

Obezbeđivanje i priprema podataka obuhvataju posao definisan u prve četiri faze postupka. U cilju što efikasnije realizacije projekta u fazi razvoja alata podaci mogu da se obezbede direktno sa posmatranih veb stranica metodom koja se popularno naziva "prikupljanje sa veba" [5]. Prikupljanje sa veba predstavlja automatsku obradu i prikupljanje podataka sa javnih veb stranica. Etička ispravnost (pa i legalnost) ovog postupka predmet je stalne rasprave u IT zajednici. Međutim, kako se alat WebMap razvija u akademske svrhe, metod prikupljanja sa veba je najjednostavniji način za obezbeđivanje podataka za prikaz. Ostali podaci potrebni za iscrtavanje mape i geografskih odlika biće obezbeđeni iz javnih izvora na Internetu.

Reprezentacija podataka i interaktivnost čine srž problema. Kako sprovesti uspešnu geovizualizaciju? Kome želimo da prikažemo podatke i na koji način? Kao što je već napomenuto ciljni medij je Internet. Prikaz vizualizacije mora biti što brži, vizualno dopadljiv i pre svega jasan već na prvi pogled. Interaktivnost na različitim uređajima može predstavljati problem ali sam prikaz mora da bude kompatibilan sa većinom današnjih uređaja i veb pregledača. Prilikom izbora tehnika za razvoj sve navedeno se mora uzeti u obzir.

Tri osnovna prikaza koja će biti omogućena alatom WebMap su:

1. **Prikaz merene vrednosti u obliku kruga**, gde boja kruga i prečnik oslikavaju merenu vrednost;

2. **Prikaz merenih vrednosti u obliku povezanih krugova**, gde pored samih vrednosti prikazujemo i eventualnu povezanost dve merne stanice (primer su stanice za merenje vodostaja na istom rečnom slivu);
3. **Prikaz podataka u obliku pravougaonika**, gde visina i boja odgovaraju merenoj vrednosti podatka;

Interakcija omogućava prilagođavanje prikaza potrebama korisnika. Korisnik treba da bude u mogućnosti da pristupi što lakše podacima (i prikazu) koji ga interesuju na način koji mu najviše odgovara. Potrebno je obezbediti osnovne interakcije sa mapom (skaliranje i pomeranje) kao i bar osnovnu interakciju sa prikazanim podacima (detaljniji prikaz podatka).

Poželjno bi bi bilo omogućiti i pregled vrednosti izmerenih u različitim vremenskim trenucima na način koji bi korisniku omogućio da uoči eventualna pravila u promenama vrednosti tokom vremena. Jedno dobro rešenje ovog problema bilo bi animirano menjanje vrednosti tokom vremena.

Generalne smernice razvoja alata:

- Korišćenje javnih, besplatnih podataka i programskih biblioteka;
- Razvoj u tehnikama prilagođenim prikazu na vebu i na različitim uređajima;
- Korišćenje savremenih tehnologija;
- Izbor između više različitih vrsta prikaza;
- Primenljivost na širokom spektru geoprostornih podataka;
- Modularnost projekta, mogućnost lakog unapređivanja i dopunjavanja alata;

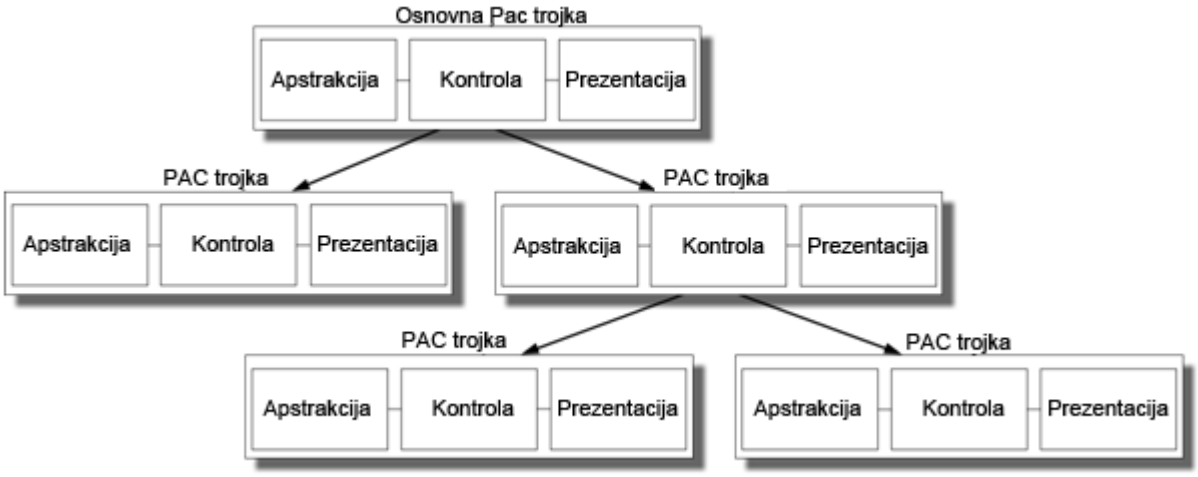
I na kraju, ono što nema konkretne veze sa vizualizacijom ali je vrlo bitno za razvoj alata WebMap, jeste izbor okruženja u kom će se alat razvijati kao i **definisanje modela i arhitekture** alata. Potrebno je izabrati dobru osnovu za razvoj alata, uzimajući u obzir prethodno postavljene ciljeve. Kako je akcenat ipak na korisniku vidljivom delu alata (grafički prikaz) više pažnje je potrebno posvetiti tom delu.

3 Idejno rešenje alata WebMap

3.1 Arhitektura

Uzimajući u obzir prirodu problema i motivaciju rada, akcenat alata je unutar njegovog prezentacionog sloja. Korisniku alata potrebno je prikazati statične podatke kao što su lokacije stanica, granice karte i sl. ali i dinamične podatke izmerenih vrednosti parametara na mernim stanicama u različitim vremenskim trenucima. Poželjno je da ceo koncept bude što je moguće više uopšten kako bi odgovarao većem broju slučajeva mogućeg korišćenja. Imajući prethodno u vidu, ređe menjane (statične) podatke kao što su podaci o stanicama i parametrima merenja možemo čuvati u bazi podataka. Korisnik (sa odgovarajućim pristupom i privilegijama) bi bio u mogućnosti da održava bazu mernih stanica, da briše i menja postojeće ili dodaje nove stanice, slično kao i u slučaju parametara merenja ili bilo kojih drugih statičnih podataka. Ovim bi dobili i klasu klijenata alata koji održavaju gradivne elemente mapa i prikaza, što je svakako poželjno u budućem razvoju.

Za upravljanje statičnim podacima i komunikaciju sa bazom koristiće se PHP okruženje *Codeigniter* [6]. *CodeIgniter* je okruženje sa relativno malim otiskom (*footprint*, obavezna projektna osnova) koje omogućava brži razvoj dinamičkih veb aplikacija. Okruženje pruža osnovne biblioteke koje značajno olakšavaju pojedine poslove (npr. komunikacije sa bazom) ali pruža i osnovu arhitekture pogodne za ovaj projekat. Iako se u mnogim izvorima na Internetu (pa i na zvaničnoj stranici CodeIgniter-a) navodi da je okruženje u osnovi ne-striktna MVC (Model-View-Controller) arhitektura, preciznije je reći da će u konkretnom slučaju implementacije ovog alata arhitektura biti **PAC** (Presentation-Abstraction-Control) predstavljena na slici 7, gde se glavna kontrola i deo za čitanje statičnih podataka u bazi može posmatrati kao glavna (vrhovna) trojka arhitekture alata [7].



Slika 7 - Arhitektura PAC

Direktna komunikacija između modela podataka (apstrakcionog dela) i prikaza (prezentacije) podataka ne postoji. Celokupna komunikacija se vrši preko posrednika odnosno kontrole. Druga PAC trojka vezana za statične podatke mogla bi biti zadužena za izmene nad statičnim podacima odnosno potencijalni "administratorski" deo alata.

Sa druge strane, ako posmatramo dinamične podatke odnosno vrednosti merenja koje želimo da prikažemo jasno je da su to podaci podložni čestim promenama, potencijalno obimni i glomazni za rukovanje i obradu. Poželjno bi bilo omogućiti što lakši i brži pristup ovim podacima - samim tim treba ih čuvati i održavati što bliže klijentu alata. Merene vrednosti biće čuvane u formatu JSON [8] koji je postao jedan od standarda za komunikaciju, održavanje i obradu podataka u današnjim veb pregledačima. Još jedna PAC trojka (ili PAC objekat kako se negde naziva) biće Javascript modul zadužen za merene podatke. Slično kao u prethodnim slučajevima - *abstraction* će predstavljati sami podaci i metode za obradu, *presentation* će sadržati konkretne metode zadužene za prikaz, dok će *control* omogućiti osnovni mehanizam komunikacije dva prethodna ali i komunikacije sa osnovnom PAC trojkom preko kontrole zadužene za statične podatke.

3.2 Model podataka

Jasno se razlikuju dve, po postojanosti različite klase podataka u postavljenoj problematici. Podaci koji su po svojoj prirodi postojaniji čuvaće se u bazi podataka. Sistem za upravljanje bazom podataka koji će biti korišćen za čuvanje statičnih podataka je MySQL. Osnovni prepoznati entiteti baze su **mapa**, merna **stanica** i **parametar** merenja.

Drugu klasu podataka podatke čine konkretne vrednosti merenja. Jasno je da su vrednosti merenja dinamičke, sa čestim promenama kroz vreme, te skladištenje u relacionoj bazi podataka u njihovom slučaju nije najbolje rešenje.

Pre samog razvoja i formirnja modela podataka definisani su osnovni zahtevi i ideje vezane za entitete u alatu:

Karakteristike mernog parametra

1. Naziv
2. Merna jedinica
3. Definisanje pojasa ekstrema (kritične vrednosti)

Karakteristike merne stanice

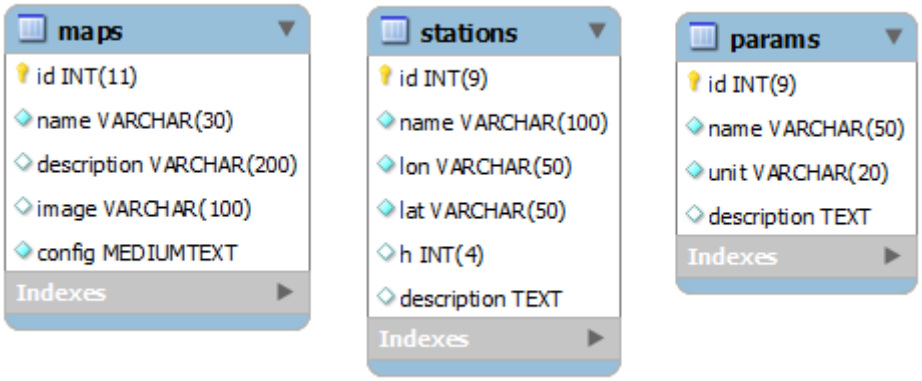
1. Podešavanje osnovnih podataka (naziv, dodatne informacije...)
2. Odabir mernih parametara za baznu stanicu, pravljenje liste. Merna stanica može meriti jedan ili više mernih parametara
3. Definisanje specijalnih informacija (odnosi, susedstva itd.). Odnosi među stanicama su posebno bitni npr. u slučaju merenja vodostaja, zbog odnosa stanica na istom rečnom slivu.

4. Pozicioniranje na Google mapi (kao pomoćni / precizniji prikaz lokacije). Kako su Google mape veoma rasprostranjene i prepoznatljive nije na odmet omogućiti paralelne prikaze lokacija korišćenjem *Google Maps API*. Jedino što nam je potrebno jesu geografska širina i dužina lokacije na kojoj se stanica nalazi.

Karakteristike mape

1. Logo (simbol mape)
2. Opis mape
3. Lista stanica koje su vezane za mapu. Stanice mogu biti deljene za više mapa
4. Uzimajući u obzir tačku pod brojem 3. potrebno je omogućiti izbor vidljivih parametara za određenu mapu.

Osnovni, "statični" entiteti koji se čuvaju u relacionoj bazi prikazani su sa svojim atributima na slici 8.



Slika 8 - Model entiteta baze podataka

Na osnovu sheme prikazane na slici 8 nije očigledno kako su i na koji način ispunjeni prethodno navedeni zahtevi u vezi sa karakteristikama entiteta. To se prvenstveno odnosi na zahteve usmerene ka odnosima među entitetima, vezama koje postoje između mapa i stanica, stanica i parametara itd.

Atribut **config** koji opisuje mapu sadrži informacije o vezama između entiteta. Atribut config predstavlja konfiguracioni XML koji između ostalog povezuje mapu sa stanicama i parametrima u bazi. Razlog zbog kog su relacije entiteta iz relacione baze izmeštene u XML oblik je lakše menjanje konfiguracije ali i mogućnost lakše proširivosti u budućnosti. Pored osnovnih veza, konfiguracioni XML može definisati i dodatne informacije kao što su boje i stilovi prikaza za entitete, globalne i lokalne (vezane za određenu mernu stanicu) kritične vrednosti parametara itd. Primer konfiguracione XML datoteke:

```
<map_config>
<stations_data>
        <stations>
                <station id="38" />
                <station id="40" />
        </stations>
</stations_data>
```

```
<parameters_data>
        <parameters>
        <parameter id="1" color="#FFA4A4">
        <critical_points>
        <critical_point id="ok" name="normal" value="2" />
        <critical_point id="vs" name="alarm" color="#F70000" value="Inf" />
        </critical_points>
        </parameter>
</parameters_data>
</map_config>
```

Pravila za pravljenje konfiguracionog XML-a

- Koreni XML element je `<map_config>`.
- Elementi `<map_config>` mogu biti `<stations_data>`, `<parameters_data>` i `<station_parameters_data>`
    - `<stations_data>` sadrži podatke o mernim stanicama (koje se nalaze u bazi).
    - Elementi `<stations_data>` mogu biti `<stations>` i `<station_relations>`
        - `<stations>` sadrži listu stanica koje mapa koristi (određenih id-em iz baze) u obliku: `<station id="x" />`
        - `<station_relations>` sadrži listu relacija među stanicama u obliku: `<rel a="x" b="y" />`
    - `<parameters_data>` sadrži podatke o parametrima relevantnim za mapu
    - Elementi `<parameters_data>` su `<parameters>`
    - Elementi `<parameters>` su `<parameter>` koji sadrži vezu sa parametrom u bazi podataka preko atributa `id`. Svaki parametar može imati definisanu boju za prikaz.
        - `<parameter>` može sadržati kritične tačke definisane kroz skup kritičnih tačaka u obliku `<critical_point id="ok" name="normal" value="2" />` čiji atributi sadrže naziv, vrednost, i boju prikaza.
    - `<station_parameters_data>` može sadržati vrednosti kritičnih tačaka specifičnih za određeni parametar i mernu stanicu u sledećem obliku:
        - ```
          <station id="x">
                <parameter id="y">
                <critical_point id="ok" value="z" />
                </parameter>
          </station>
          ```

U slučaju podataka koji predstavljaju vrednosti merenja, jasno je da ih određuju statični podaci kao što su stanica i parametar ali i vreme merenja kao i sama izmerena vrednost. U ovom slučaju bitno je definisati standard zapisa kroz koji će alat obrađivati podatke. Više reči o tome biće u poglavlju koje se bavi detaljima implementacije alata WebMap.
          ```

4 Implementacija alata WebMap

4.1 Tehnologije i alati

Prikaz podataka i mogućnost interakcije sa istim svakako predstavlja srž alata WebMap ali i celokupnog rada. U ovom delu biće prikazani detalji implementacije kao i tehnike i alati korišćeni prilikom izrade alata. Podsetimo se osnovnih zatheva i ideja vezanih za prikaz:

1. **Ciljni medij je Internet**
   WebMap je veb alat, osnovu prikaza čine HTML, CSS i Javascript.

2. **Potrebna je vidljivost (bar osnovni prikaz) na što većem broju platfomi i uređaja**
   U opticaju je bilo više različitih načina implementacije prikaza. Jedna od opcija bio je *Adobe Flash* i njegov jezik *ActionScript 3* (za koji je potreban *Flash Player*) ali je u poslednje vreme izgubio podršku na određenim uređajima te je ova mogućnost odbačena. Dilemu su predstavljala dva moguća izbora - HTML5 korišćenjem elementa `Canvas` ili SVG.

Canvas	**SVG**
– 2D podloga za crtanje sa visokim performansama – Performanse su konstantne - sve je piksel, performanse opadaju sa rezolucijom slike. – Sadržaj Canvas elementa može se sačuvati kao jpg ili png (rasterska grafika) – Najbolje rešenje za rastersku grafiku (igre i slično), obradu slika i svega što zahteva obradu piksel po piksel.	– Ne zavisi od rezolucije ekrana sa kog se pregleda - ovo ga čini dobrim kandidatom u slučajevima kad je potrebna kompatibilnost na različitim uređajima – SVG ima dobru podršku za animacije – Elementi u SVG-u su u formi DOM stabla te su lako pristupačni – SVG je u suštini XML dokument te je kao takav standardizovan i predstavlja već dobro upoznatu strukturu na vebu

   SVG više odgovara prirodi ove vrste projekta [9] pa je izabran kao tehnika implementacije prikaza u projektu razvoja alata WebMap.

3. **Što viši stepen nezavisnosti od spoljnih izvora podataka i servisa**
   Konkretno, ovo se odnosi na korišćenje spoljnih servisa kao što su Google mape (npr. za lociranje stanice, za prikaz mape Srbije tj. podloge za podatke itd.). Google mape će biti prisutne u alatu kao opcija ali ne i kao osnova.

4. **Modularnost i proširivost**
   Dizajn alata mora biti modularan. Dodavanje novih oblika vizualizacije mora biti jednostavno za implementaciju. Alat mora biti nezavistan od načina obezbeđivanja podataka za prikaz.

Ono što korisniku alata nije uočljivo a bitno je za osnovu projekta je izbor načina za čuvanje i rukovanje podacima za prikaz. Neki od standarda za komunikaciju i čuvanje podataka koji se danas široko koriste su jezik XML i JSON.

4.1.1 XML

Jezik XML (*EXtensible Markup Language*) [10] služi za čuvanje i transport podataka. XML je jezik za označavanje, slično kao i HTML, s tim što je fokus XML-a na tome šta podaci predstavljaju a ne kako ih prikazati. Nazive oznaka u XML dokumentu definiše korisnik pa ih može potpuno prilagoditi podacima koje želi da sačuva. Sadržaj XML dokumenta (koji je u osnovi tekstualni dokument) može se podeliti na dve klase - oznake i sadržaj (vrednosti).

Oznake (*tags)* počinju sa "<" i završavaju sa ">". Oznake mogu postojati u tri različita oblika:

- Otvarajuće (npr. `<section>`)
- Zatvarajuće (npr. `</section>`)
- Prazne, bez elemenata u sadržaju (npr. `<section />`)

Oznake mogu imati atribute koji ih karakterišu, na primer:

```
<img src="slika.jpg" alt='Ovo je neka slika' />
```

Element XML dokumenta predstavlja ili praznu oznaku (kao što je u prethodnom primeru `img`) ili celinu koja počinje određenom otvarajućom oznakom i završava sa odgovarajućom zatvarajućom oznakom. Pod sadržajem elementa podrazumeva se sve što se nalazi između ove dve oznake (drugi XML elementi ili tekstualni sadržaj). Primer XML dokumenta:

```
<note>
        <to>Tove</to>
        <from>Jani</from>
        <heading>Reminder</heading>
        <body>Don't forget me this weekend!</body>
</note>
```

4.1.2 JSON i GeoJSON

Format JSON (*JavaScript Object Notation),* slično XML dokumentima, služi za opis, čuvanje i transport podataka [8]. Format JSON je tekstualni format, nezavistan od bilo kog programskog jezika te se može koristiti u različite svrhe i u različitim razvojnim okruženjima. Za razliku od jezika XML čiji zapis održava podatke u drvolikoj formi, format JSON čine dve osnovne strukture podataka:

- Kolekcija parova ime-vrednost (što odgovara različitim strukturama podataka, najviše objektima, rečnicima i sl.)
- Uređena lista vrednosti (što u raznim programskim jezicima odgovara nizovnim strukturama, listama itd.)

Objekat u JSON notaciji počinje sa "{" i završava se "}". Parovi imena i vrednosti odvojeni su zarezima, dok se ime i vrednost u okviru para odvajaju sa dve tačke (":"). Nizovi vrednosti se ograničavaju uglastim zagradama, dok se vrednosti niza odvajaju zarezima. JSON omogućava i zapis ugnježdenih struktura.

4.1.2.1 GeoJSON

GeoJSON je otvoren standard za označavanje kolekcija jednostavnih geografskih podataka, odlika i njihovih atributa korišćenjem JSON-a [11]. Geografski podaci koji se predstavljaju su tačke (odnosno adrese i lokacije) i otvoreni i zatvoreni poligoni (odnosno ulice, reke, granice, regije itd.). Geografski podaci se takođe mogu grupisati u odgovarajuće kolekcije.

Objekti standarda GeoJSON

Zapis GeoJSON se uvek sastoji od jedinstvenog objekta. Ovaj objekat može predstavljati geometriju, geografsku odliku ili kolekciju geografskih odlika. Objekat mora imati par ime-vrednost sa imenom `"type"` koji određuje tip objekta. Mogući tipovi objekata standarda GeoJSON su:

- Geometrijski
  - `"Point"`,
  - `"MultiPoint"`,
  - `"LineString"`,
  - `"MultiLineString"`,
  - `"Polygon"`,
  - `"MultiPolygon"`,
  - `"GeometryCollection"`
- Geografske odlike
  - `"Feature"`
  - `"FeatureCollection"`

Svaki geometrijski objekat koji nije "`GeometryCollection`" mora imati člana pod nazivom "`coordinates`" koji je uvek nizovnog tipa i čija struktura zavisi od tipa objekta u kom se nalazi.

Objekti koji predstavljaju geografske odlike moraju imati člana koji definiše njihovu geometriju pod nazivom "`geometry`" ali i "`properties`" koji može predstavljati bilo koji objekat koji opisuje ili karakteriše tu odliku. Primer GeoJSON zapisa:

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [102.0, 0.5]
      },
      "properties": {
        "prop0": "value0"
      }
    },
    {
      "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [102.0, 0.0], [103.0, 1.0], [104.0, 0.0], [105.0, 1.0]
        ]
      },
      "properties": {
        "prop1": 0.0,
        "prop0": "value0"
      }
  ]
}
```

4.1.3 Skalabilni vektorski crteži

4.1.3.1 Osnovni koncepti

Skalabilni vektorski crteži ili SVG (*Scalable Vector Graphics*) [12], je vrsta jezika XML za obeležavanje i predstavljanje vektorskih (grafičkih) sadržaja.

Skalabilnost podrazumeva uniformno povećavanje ili smanjivanje bez narušavanja sadržaja i kvaliteta. U kontekstu grafike, skalabilnost podrazumeva nezavisnost prema pikselima, odnosno nepostojanje neke fiksirane veličine crteža u pikselima. To znači da je format crteža SVG prilagodljiv različitim rezolucijama prikaza te da je moguće prikazivati različite veličine i verzije grafike u zavisnosti od potrebe, bez gubljenja kvaliteta prikaza.

Vektorska grafika sastavljena je od geometrijskih oblika kao što su linije i krive, poligoni i geometrijska tela. Vektorska grafika je u velikoj meri fleksibilnija u odnosu na

rastersku grafiku gde svaki piksel nosi neku informaciju potrebnu za prikaz. Vektorski formati kao SVG mogu takođe da kombinuju i rasterske grafike ukoliko je to potrebno.

XML predstavlja opšteprihvaćen standard za strukturiranje različitih tipova informacija. U slučaju SVG-a, XML je odlično sredstvo za strukturiranje i opis vektorske i mešovite grafike.

4.1.3.2 Struktura SVG dokumenta

Neki od osnovnih strukturnih elemenata SVG crteža:

- **"svg"** - Glavni koreni element SVG-a je čvor "`svg`". Ovaj element sadrži sve ostale elemente SVG-a.
- **"g"** - Element koji predstavlja grupu elemenata. Često je potrebno iz različitih razloga grupisati određene elemente u objedinjen skup.
- **"defs"** - Definicije ponovo upotrebljivog sadržaja (senki, gradijenata...)
- **"image"** - Element koji predstavlja drugi vektorski ili rasterski crtež
- itd.

4.1.3.3 Osnovni oblici

Osnovni oblici predstavljaju se sledećim SVG elementima:

- **"rect"** - pravougaonici (opcionalno sa zakrivljenim uglovima)
  Osnovni opisni atributi:
  - `x` i `y` - predstavljaju položaj levog gornjeg temena pravougaonika u referentnom koordinatnom sistemu
  - `width` i `height` - visina i dužina (paralelne osama koordinatnog sistema)
  - `rx` i `rz` - predstavljaju zakrivljenost ćoškova
- **"circle"** - krugovi
  Osnovni opisni atributi:
  - `cx` i `cy` - predstavljaju koordinate centra kruga
  - `r` - poluprečnik kruga
- **"ellipse"** - elipse
  Osnovni opisni atributi:
  - `cx` i `cy` - predstavljaju koordinate centra
  - `rx` i `ry` - poluprečnici paralelni x i y osama u referentnom koordinatnom sistemu
- **"line"** - prave linije
  Osnovni opisni atributi:
  - `x1,y1` i `x2,y2` - predstavljaju koordinate početne i krajne tačke
- **"polyline"** i **"polygon"** - poligonske linije i oblici
  Osnovni opisni atributi:
  - `points` - skup tačaka

4.1.3.4 Putanje (elementi *Path*)

Element putanje služi za definisanje novih oblika. Putanja predstavlja okvir oblika koji može da se popunjava i uokviruje, da služi kao maska za isecanje ili kao kombinacija nečega od ove tri mogućnosti. Element putanje karakteriše koncept "tekuće tačke". U analogiji sa crtanjem po papiru pod tekućom tačkom može se smatrati trenutni položaj olovke. Položaj olovke se može promeniti i time odrediti trag koji ostavlja bilo da je u pitanju kriva ili prava linija. Putanje predstavljaju geometriju oblika tj. njegove granice (*outline*). Pravljenje putanje vrši se uz pomoć sledećih "komandi" koje se izvršavaju u okviru koncepta "tekuće tačke":

- ***moveto (M)*** - postavi novu tekuću tačku
- ***lineto* (L, H, V)** - nacrtaj pravu liniju
- ***curveto* (C, S, Q, T)** - nacrtaj Bezijerovu krivu drugog ili trećeg stepena
- ***arc (A)*** - nacrtaj eliptični luk
- ***closepath (Z)*** - zatvori oblik crtanjem linije ka poslednjoj tekućoj tački

Moguće je kreirati i takozvane šuplje likove korišćenjem kombinovanih putanja.

Osnovni atribut elementa putanje je atribut "`d`" koji sadrži podatke o okviru oblika. Posmatrajmo sledeći primer koji crta trougao:

```
<path d="M 100 100 L 300 100 L 200 300 Z"
fill="red" stroke="blue" stroke-width="3" />
```

U navedenom primeru koordinate početne tačke crtanja u referentnom koordinatom sistemu su x=100 i y=100 (komanda M 100 100). Sledeća komanda je iscrtavanje linije do pozicije x=300 i y=100 (L 300 100), nakon toga crta se linija do pozicije x=200 i y=300 i zatvara oblik komandom ***closepath*** odnosno oznakom komande Z.

4.1.4 D3.js

D3.js (od "*Data Driven Documents*" ili samo D3) je Javascript biblioteka otvorenog koda koja olakšava pravljenje i rukovanje objektima u veb aplikacijama koje implementiraju neki vid vizualizacije podataka [13]. D3 olakšava povezivanje podataka sa elementima DOM strukture veb stranica. Biblioteka funkcioniše u skladu sa široko rasprostranjenim i prihvaćenim standardima i tehnikama izrade korisničkih interfejsa i interaktivnih sadržaja na Internetu - HTML5, CSS3, Javascript, jQuery, SVG itd.

U osnovi, D3 je skup gotovih funkcija za obilazak i obradu elemenata DOM stabla veb stranice. D3 omogućava dinamičko pravljenje, izdvajanje i obradu elemenata u DOM stablu. Ono što ovu biblioteku čini posebno zanimljivom u kontekstu ovog projekta jeste povezivanje određenih tipova ili klasa elemenata sa odgovarajućom vrstom podataka (različite sematike). Ako, na primer, imamo niz brojeva, korišćenjem biblioteke D3 veoma je lako napraviti HTML tabelu ili SVG grafiku sa pravougaonicima različite dužine.

4.1.4.1 Izdvajanje elemenata u D3

Pod izdvajanjem se podrazumeva niz elemenata izdvojenih iz DOM stabla neke veb stranice. Funkcije biblioteke D3 za izdvajanje elemenata oslanjaju se na tzv. "CSS3 selektore" definisane u *W3C Selectors API* [14]. Selektori su obrasci (*pattern*) koji služe za izdvajanje elemenata u DOM stablu i prihvaćeni su i podržani od strane velike većine današnjih veb pregledača. Selektorski obrasci mogu biti sačinjeni od raznih tipova HTML oznaka - tagova, attributa `id` ili atributa `class`.

Posmatrajmo sledeći problem - potrebno je izdvojiti sve paragrafe i promeniti boju teksta na osnovu nekog korisnički izazvanog ili drugog tipa događaja na stranici. U Javascriptu bi to izgledalo ovako:

```
var paragraphs = document.getElementsByTagName("p");
for (var i = 0; i < paragraphs.length; i++) {
  var paragraph = paragraphs.item(i);
  paragraph.style.setProperty("color", "white", null);
}
```

Deklarativnim pristupom biblioteke D3 i korišćenjem selektora ovaj postupak se svodi na jednu komandu:

```
d3.selectAll("p").style("color", "white");
```

Naravno selekcije mogu biti i pojedinačne, po atributu `id` DOM elementa ili drugim jedinstvenim elementima u dokumentu, na primer:

```
// izdvajanje elementa body i promena boje pozadine
d3.select("body").style("background-color", "black");
// izdvajanje elementa sa id-em "mainContainer"
d3.select("#mainContainer").style("background-color", "black");
```

4.1.4.2 Povezivanje izdvojenih elemenata i podataka

Vrednosti atributa u skupu izdvojenih elemenata ne moraju biti jednake. Naprotiv, poželjno je da vrednosti atributa nekih elemenata budu dinamički određene u zavisnosti od podataka koje želimo da predstavimo.

Pretpostavimo da je potrebno da grafički prikažemo određeni niz celobrojnih vrednosti kao niz horizontalnih linija (HTML element `<hr/>`) različitih dužina. Posmatrajmo sledeći primer korišćenja metode `selection.data([values[,key]])` za povezivanje podataka i izdvojenih elemenata DOM stabla:

```
d3.selectAll("hr")
    .data([40, 80, 150, 160, 230, 420])
```

```
    .style("width", function(d) { return d + "px"; });
```

Prethodnim kodom će prvih šest elemenata `hr` u HTML dokumentu imati širine odgovarajuće vrednostima u nizu zadatom kao argument `data()` metoda koji nudi D3. Prvi argument ovog metoda može biti niz primitivnih tipova ili niz objekata. Jednom povezan podatak ostaje u elementu kao vrednost atributa "`__data__`". Postoje različiti načini kontrole povezivanja vrednosti podatka (*datum*) i elementa. Podaci ne moraju biti povezani sa elementima samo na osnovu poretka u selekciji i poziciji u nizu. D3 nudi različite opcije za kontrolu veze podataka i elemenata preko drugog (opcionog) argumenta "`key`".

Dodavanje novog elementa u DOM

Još jedna značajna mogućnost ove biblioteke jeste dinamičko dodavanje DOM elemenata. Za svaki element iz skupa podataka vezanih za selekciju za koji ne postoji odgovarajući DOM element, metod `enter()` pravi nov, prazan DOM element koji pridodaje već izdvojenim elementima. Rezultat metoda `enter()` se takođe tretira kao skup izdvojenih elemenata. Ovako dobijen skup ipak nudi samo sledeće metode: `append`, `insert`, `select` i `call`, od kojih su najinteresantnije prve dve. Ukoliko posmatramo sledeći kod:

```
d3.select("body").selectAll("div")
        .data([4, 8, 15, 16, 23, 42])
        .enter().append("div")
        .text(function(d) { return d; });
```

Uz pretpostavku da je stranica tj. element `body` inicijalno prazan, rezultat izvršavanja ovog segmenta će biti stranica sa elementom `body` koji ima 6 elemenata `div` koji sadrže vrednost odgovarajućeg podatka iz niza zadatog metodom `data()`:

```
<body>
       <div>4</div>
       <div>8</div>
       <div>15</div>
       <div>16</div>
       <div>23</div>
       <div>42</div>
</body>
```

Izdvajanje nepovezanih elemenata

Metodom `exit()` dobijamo DOM elemente u izdvajanju koji su ostali nepovezani - odnosno nemaju par u definisanom skupu podataka zadatom metodom `data()`. Ovaj skup elemenata je interesantan jer je u određenim slučajevima potrebno ukloniti elemente koji ne prikazuju nikakvu vrednost. Kao što su u slučaju skupa dobijenog nakon `enter()` najinteresantnije metode `append()` i `insert()`, tako je ovde najinteresantniji metod `remove()` koji uklanja elemente iz DOM strukture.

4.1.4.3 D3 i SVG grafike - generatori putanja

Biblioteka D3 nudi odličnu podršku za pravljenje Skalabilnih vektorskih crteža (SVG). Pored osnovnih mogućnosti dodavanja SVG čvorova (*node* elemenata) u SVG crtež (na isti način kao i ostalih DOM elemenata u veb stranicu) D3 nudi i znatno zanimljivije "generatore putanja" (*path generators*) odnosno generatore linija, poligona i oblika sa većom slobodom pravljenja od standardnih SVG primitiva. Putanja je element SVG-a definisan od strane W3C kao ivična linija oblika koji može biti popunjavan, oivičen ili korišćen kao maska isecanja (*clipping mask*). Putanja je definisana konceptom "tekuće tačke". Osnovni atribut elementa putanje je `"d"`.

Primer generatora putanje biblioteke D3 koja služi za pravljenje elementa putanje SVG na osnovu podataka iz niza elemenata sa vrednostima x i y i zadate interpolacije:

```
var line = d3.svg.line()
    .x(function(d) { return d.x; })
    .y(function(d) { return d.y; })
    .interpolate("basis");

g.append("path")
    .attr("d", line);
```

U prethodnom primeru promenljiva `"line"` je u isto vreme i objekat i funkcija. Ukoliko pretpostavimo da je `g` neki postojeći element u pripremljenom SVG crtežu, zatim da je neki niz podataka (tj. koordinata) već vezan za tu grupu `g` (odnosno čvor `g`), onda je rezultat prethodnog koda kriva koja prolazi kroz zadati niz tačaka, određena odgovarajućom izabranom interpolacijom. Moguće interpolacije krive koje nudi D3 su, između ostalih:

- *linear* - linearna kriva - praktično predstavlja iscrtavanje poligona sa zadatim temenima;
- *linear-closed* - iscrtavanje zatvorenog poligona;
- *basis, basis-open, basis-closed* - varijacije B splajna;
- i druge.

4.1.4.4 Čitanje spoljašnjih resursa (izvora podataka)

Ukoliko su spoljašnji skupovi podataka veliki, njihovo asinhrono učitavanje u veb stranicu je od velikog značaja za korisničko iskustvo. Asinhrono učitavanje podataka se vrši preko objekta klase `XMLHttpRequest` (XHR). Prilikom obrade ovih zahteva, učitavanje ostalih resursa se ne blokira, a sve što zavisi od podataka koje očekujemo od asinhronog poziva obrađujemo u takozvanim *callback* funkcijama. D3 nudi skup predefinisanih metoda za različite tipove spoljašnjih resursa: JSON, HTML, XML, CSV. Primer učitavanja JSON datoteke:

```
var data;
d3.json("path/to/file.json", function(error, json) {
  if (error) return console.warn(error);
```

```
  data = json;
  visualizeit();
});
```

4.1.5 Modularni JS

Budući da je dobar deo alata razvijan u Javascript-u (JS) jedno od pitanja o kome se moralo voditi računa bilo je i pitanje organizacije koda. Moduli su integralan deo bilo koje robusne aplikativne arhitekture. Moduli se u Javascript-u koriste kao alternativa klasama koje u Javascript jeziku ne postoje. Moduli imaju određene odlike klasa iz objektno orjentisanih jezika kao što je recimo enkapsulacija, ali suštinski to su funkcije koje se izvršavaju u jednom prolazu. Obrazac "Modularni JS" je poznat način modularizacije aplikacija razvijanih u Javascript-u. Pre objašnjenja samog obrasca potrebno je razumeti i pojam "zatvorenja" funkcije.

4.1.5.1 Zatvorenje

Pojam zatvorenja funkcije odnosi se na samu funkciju ali i na okruženje u kome se ona referiše. Zatvorenje funkcije održava tabelu ne-lokalnih vrednosti referisanih u funkciji prilikom njenog izvršenja [15]. Zatvorenje omogućava da unutrašnja funkcija referiše i koristi promenljive iz spoljašnjeg okruženja (šireg bloka u kom je ona definisana). Predstavljeno na primeru:

```
// Funkcija koja racuna funkciju za dodavanje broja

function addGenerator(num) {
    // Racuna i vraca funkciju za dodavanje broja datom broju
    return function (toAdd) {
        return num + toAdd
    };
}
// addFive je funkcija koja dodaje 5 argumentu
var addFive = addGenerator(5);

alert(addFive(4) == 9);   // Ovo je tacno
```

4.1.5.2 Obrazac otkrivajućeg modula

Postoji više varijacija modularnih JS šablona. U okviru projekta WebMap korišćen je takozvani "otkrivajući" šablon. Posmatrajmo sledeći primer:

```
var EmployeeModule = (function () {
    var employeeList = [];
    return {
        add: function (employee) {
            employeeList.push(employee);
        },
        getAll: function () {
```

```
            return employeeList;
        },
        totalCount: function () {
            return employeeList.length;
        }
    };
})();
```

U prethodnom primeru postoji lista zaposlenih koja se van modula ne vidi. Komunikacija sa modulom definisana je kroz tri funkcije vidljive van modula, koje možemo nazvati javnim funkcijama. Problem sa ovakvom strukturom je u tome što je kod komplikovanijih modula organizacijski teško održiva. Pored toga, svako pozivanje javne funkcije iz neke druge zahteva poziv preko glavnog objekta. Unapređenje u odnosu na ovo je upravo odabrani obrazac "otkrivajućeg" modula: sve promenljive i funkcije definišemo kao lokalne promenljive unutar funkcije a zatim vratimo anonimni objekat sa pokazivačima na funkcionalnost i/ili vrednosi koje želimo da podelimo sa spoljašnim svetom. Na ovaj način uvodi se i određena analogija sa klasama i objektima, odnosno privatnim i javnim atributima i metodima. Prethodni primer napisan u skladu sa obrascem otkrivajućeg modula:

```
var EmployeeModule = (function () {
    var employeeList = [];
    var allEmployee = function () {
        return employeeList;
    };
    var add = function (employee) {
        employeeList.push(employee);
    };
    var totalCount = function() {
        return employeeList.length;
    };
    return {
        add: add,
        getAll: allEmployee,
        totalCount: totalCount
    };
} ());
```

4.1.6 PHP i okruženje CodeIgniter

Jezik PHP je skript jezik za programiranje na strani servera [16]. PHP se jednostavno kombinuje sa HTML jezikom u cilju izrade veb stranica sa dinamičnim sadržajima. Primer dinamičnog sadržaja je sadržaj koji zavisi od određenih podataka smeštenih u bazi podataka na serveru.

Okruženje *CodeIgniter*, razvijeno i održavano od strane kompanije *EllisLab,* omogućava lakši i brži razvoj veb aplikacija sa dinamičnim sadržajem [6]. Okruženje pruža osnovu arhitekture aplikacije u vidu ne-striktnog obrasca MVC. Pored toga, okruženje nudi razne pomoćne biblioteke koje olakšavaju česte "tipske" probleme prilikom razvoja veb aplikacija. Brzo i jednostavno učenje osnovnih mogućnosti razvoja i početnih koraka u ovom okruženju bila je jedna od preporuka u njegovu korist prilikom izbora tehnologije za razvoj severske strane alata WebMap.

4.2 Detalji implementacije

Deo zadužen za vizualni prikaz čine tri u dobroj meri logički odvojene celine:

1. **Upravljanje SVG mapom i reprezentacijom podataka**
   Moduli za rad sa SVG reprezentacijom podataka sadrže najbitniji deo posla - rad sa podacima i prikaz mape kroz različite vizualizacije. Javascript moduli zaduženi za ovaj segment su:
   - Kontrolni modul - `module_svgmap-control.js`
   - Prezentacioni modul - `module_svgmap-presentation.js`
   - Apstrakcioni modul - `module_svgmap-data.js`

2. **Upravljanje Google mapom** - `module_gmap.js`
   Modul koji komunicira sa *Google Maps API*. Namerno je izdvojen jer nije neophodan. Aplikacija može bez problema da funkcioniše i bez njega. Sa druge strane ovaj modul je proširiv u slučaju potrebe.

3. **Upravljanje korisničkim interfejsom** - `module_ui.js`
   Modul zadužen za korisnički interfejs i interakciju

Serverska strana alata i upravljanje podacima iz baze takođe se može podeliti na tri celine:

1. **Priprema podataka za prikaz** - `controllers/main.php`
   Ovu celinu čini PAC trojka zadužena za obezbeđivanje i pripremu podataka za prikaz na klijentskoj strani. Osnovno zaduženje navedenog kontrolera jeste da prikupi statične podatke i parsirani konfiguracioni XML od modela (apstrakcionog dela) pre nego što ih prosledi za prikaz.

2. **Upravljanje podacima u bazi** - `controllers/admin.php`
   Celina zadužena za upravljanje podacima u bazi (menjanje, brisanje i dodavanje). Administrativni deo alata. Navedeni kontroler održava komunikaciju između odgovarajuće prezentacije i apstrakcije podataka.

3. **Prikupljanje podataka** – `controllers/admin_scrapping.php`
   Ovo je demonstrativni segment serverske strane koji predstavlja primer prikupljanja podataka za prikaz.

Kako se rad prvenstveno bavi vizualizacijom, neće se baviti detaljima implementacije sa serverske strane koji se prevashodno odnose na komunikaciju sa bazom.

4.2.1 Struktura datoteke izmerenih vrednosti

Podaci koji predstavljaju vrednosti merenja u različitim vremenskim trenucima moraju biti sledeće strukture u formatu JSON:

Osnovni, koreni objekat karakteriše identifikacioni atribut mape za koju su podaci mereni - atribut "**mapid**"

- atribut "**times**" sadrži niz "**t**" vrednosti, odnosno vremena u kojima su podaci mereni
- atribut "**s**" sadrži niz stanica. Stanica je definisana atributom "**id**"
- Svaka stanica je objekat koji sadrži:
    - vrednost "**p**" koja predstavlja niz objekata koji sadrže izmerene vrednosti parametra u definisanim vremenskim trenucima.
    - svaki element niza "**p**" definisan je svojim atributom "**id**"
    - svaki element niza "**p**" sadrži niz izmerenih vrednosti "**d**" koje odgovaraju vrednostima i uređenju u nizu vremena "**t**"

Primer strukture datoteke JSON sa vrednostima merenja:

```
{
"mapid": "1",
"times": {
        "t": ["2014-08-01 20:00:00", ... ]
        },
"s":  [
{
        "id": "36",
        "p": [
        {
                "id": "3",
                "d":  ["171", "159", ... ]
        }
        ]
},
{
        "id": "37",
        "p": [
                {
                "id": "3",
                "d": ["206", "197", ...]
                }
        ]
        },
        ...
]}
```

4.2.2 Upravljanje SVG mapom i reprezentacijom podataka

1..1.1 Kontrolni modul SVG mape

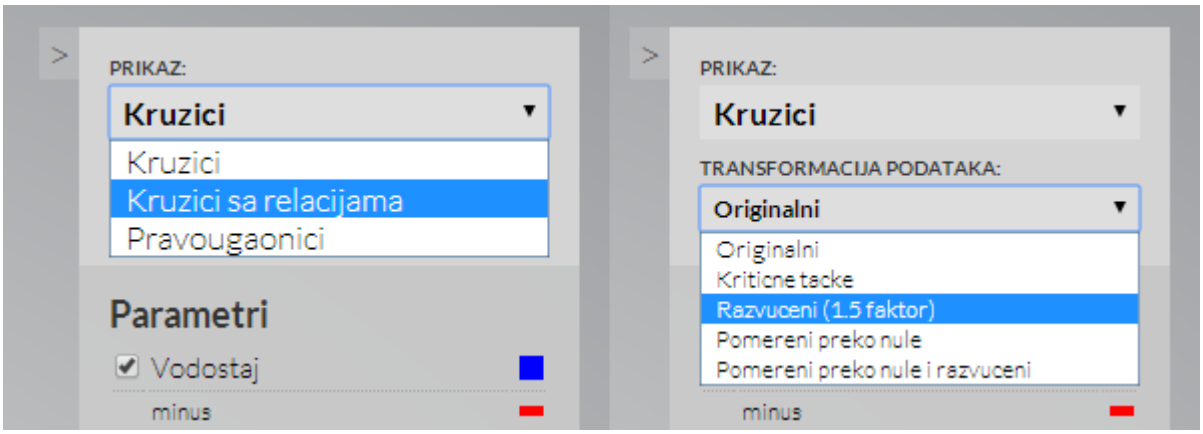
Kontrolni modul upravlja prikazom i podacima kroz komunikaciju sa preostala dva modula zadužena za podatke i njihovu prezentaciju (`module_svgmap-presentation.js`, `module_svgmap-data.js`).

Inicijalizacija

Pozivom `init` funkcije modula inicijalizuju se vrednosti modula i započinje se proces prikaza mape i podataka. Podaci koji inicijalizuju kontrolni modul su podaci dopremljeni iz baze (transformisani u format JSON): spisak stanica za odabranu mapu, spisak parametara, kao i pripremljene kritične vrednosti parametara i ostali podaci iz konfiguracionog XML-a koji karakteriše mapu. Inicijalizacijom ovog modula iniciraju se i moduli za podatke i prezentaciju podataka. Započinje se učitavanje karte, odnosno podloge za prikaz podataka iz GeoJSON datoteke sa graničnim tačkama Republike Srbije i definiše povratna funkcija koja će se izvršiti nakon što se podaci o karti učitaju.

Promena vizualizacije i preslikavanja podataka

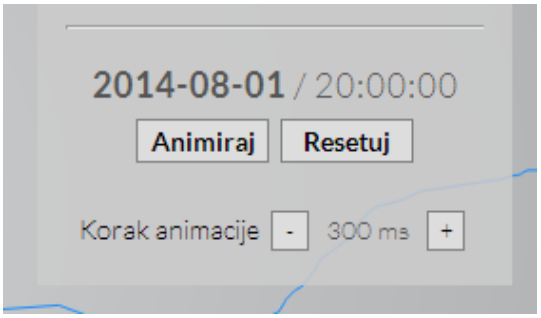
Kontrolni modul obezbeđuje interfejs za promenu vizualizacije i promenu tipa preslikavanja podataka u vrednosti pogodne za prikaz. Izborom vizualizacije ili tipa preslikavanja kroz padajuće liste (predstavljene na slici 9), napravljene u okviru modula zaduženog za korisnički interfejs (`module_ui.js`), izvršava se jedna od dve funkcije (`callVisualization(visName)`, `callNormalization(norm)`) koje sadrže logiku obrade ovih zahteva korisnika.



Slika 9 - Padajuće liste za izbor vizualizacije i tipa preslikavanja vrednosti merenja u vrednosti za prikaz

Kontrola animacije

Animacija se sastoji od sekvencijalnog prikazivanja niza vrednosti podataka izmerenih u različitim vremenskim trenucima. Korisniku je omogućena kontrola brzine animacije kroz izbor dužine pauze pre prelaska na sledeće izmerene vrednosti (prikazano na slici 10). Niz vremena koji definiše animaciju nalazi se u okviru JSON datoteke sa izmerenim podacima. Pre samih podataka u datoteci se moraju navesti vremena merenja.



Slika 10 - Kontrola animacije

1..1.1 Prezentacioni modul SVG prikaza

Crtanje karte regiona

Nakon što se u kontrolnom modulu učitaju tačke koje predstavljaju granice karte, izvršava se metod `prepareRegions(json)`. Granice karte su u formatu GeoJSON. Koordinate tačaka koje određuju granice regiona su u geografskom koordinatnom sistemu te ih treba transformisati za prikaz na ekranu. U okviru pomenute metode definiše se vrsta projekcije, centralna tačka prikaza i optimalna veličina kako bi crtež karte bio proširen na uređaju za prikaz. Projekcija korišćena u implementaciji alata WebMap je *Mercator* projekcija [17]. Detalji implementacije metoda `prepareRegions` mogu se pogledati u dodatku 7.1.1.

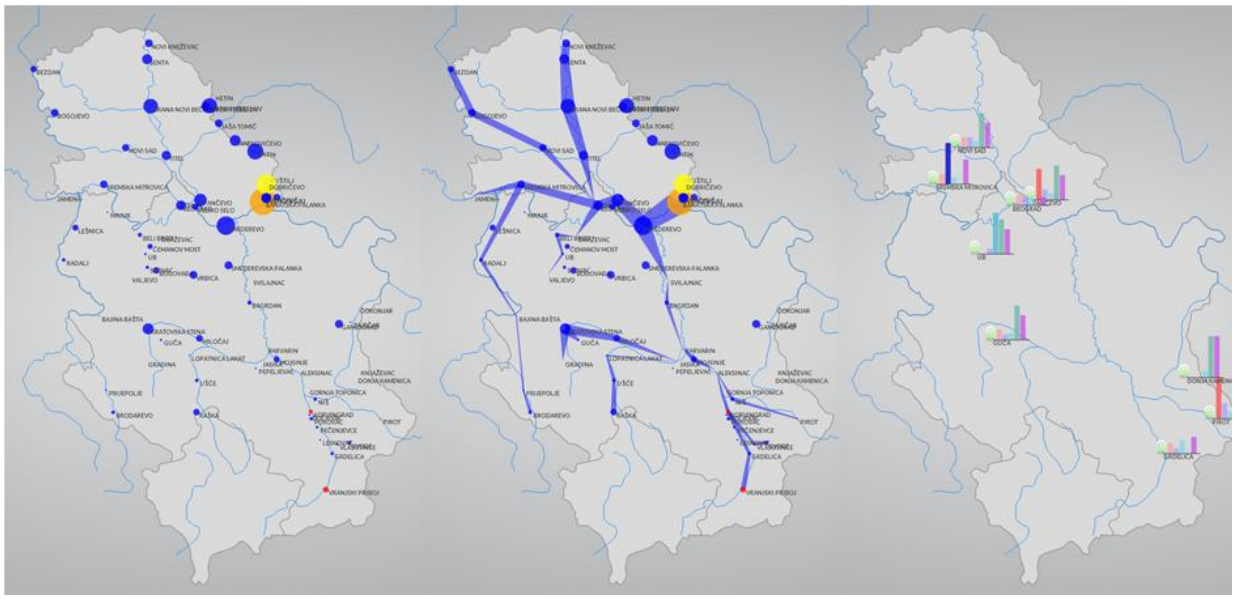
Nakon definisanja projekcije izvršava se funkcija koja crta kartu na osnovu pripremljenih podataka (detalji implementacije u dodatku 7.1.2).

Prikazivanje merenih vrednosti

Pozivom funkcije `callVisualization(visName)` funkcije iz kontrolnog modula u suštini pozivamo jednu od (trenutno) tri moguća tipa vizualizacije u prezentacionom modulu (slika 11):

1. `vCircles()` - podatak je predstavljen poluprečnikom kruga (dodatak 7.2)
2. `vCirclesConnected()` - kao pod 1. ali su susedne merne stanice vizualno povezane
3. `vBars()` - podatak je predstavljen visinom pravougaonika (dodatak 7.3)

Malo detaljnije će biti predstavljena vizualizacija povezanim krugovima jer predstavlja proširenje vizualizacije kružićima i složenija je od vizualizacije pravougaonicima.



Slika 11 - (1) Vizualizacija krugovima (vCircles), (2) Viualizacija povezanim krugovima (vCirclesConnected), (3) Vizualizacija pravougaonicima (vRect)

Vizualizacija povezanim krugovima

Sama vrednost podatka predstavlja se krugom u boji dodeljenoj odgovarajućem parametru. Prečnik kruga odgovara vrednosti podatka. Nakon iscrtavanja krugova potrebno je vizualno prikazati i veze između podataka na susednim stanicama:

1. Za svaku stanicu se ispituje da li postoji njoj susedna stanica
2. Ukoliko postoji susedna stanica pripremaju se koordinate za iscrtavanje veze između dve stanice. Veza se prikazuje kao trapez koji povezuje stanice. Parametri koji određuju tačke trapeza su:
   a. Položaji početne i krajnje stanice
   b. Vrednosti merenog podatka u obe stanice (kako bi se izračunale dužine naspramnih stranica trapeza u okviru susednih mesta).
   c. Ugao između dve susedne stanice

Na osnovu prethodnih podataka računaju se po 4 temena svakog od trapeza koji se iscrtavaju na SVG mapi.

1..1.2 Apstrakcioni modul SVG prikaza

Osnovna uloga ovog modula jeste priprema podataka dobijenih iz baze i učitavanje podataka koji predstavljaju vrednosti merenja iz JSON datoteke. Pored toga, apstrakcioni modul nudi u svom interfejsu i funkciju za normalizaciju, odnosno preslikavanje originalnih merenih vrednosti u vrednosti za prikaz. Postoji 5 različitih vrsta preslikavanja: linearno i kontrastno, zatim linearno i kontrastno sa vrednostima pomerenim preko nule (najmanja

vrednost se preslikava na nulu, ostale se pomere za istu vrednost) i preslikavanje koje se vezuje za vrednosti definisanih kritičnih tačaka.

4.2.3 Modul Google mape

Modul Google mape (`module_gmap.js`) je poprilično jednostavan, ali je zbog bolje strukture izdvojen kao zasebna celina. U funkciji `init` se inicijalizuje mapa i pripremaju se osnovni parametri i elementi prikaza. Funkcija `init` kao argument dobija niz stanica kako bi mogla da ih označi na mapi. Klikom na bilo koju stanicu (na SVG mapi) poziva se funkcija u okviru ovog modula koja pozicionira Google mapu nad odabranom stanicom (slika 12).

Slika 12 - Google mapa pozicionirana nad odabranom stanicom

Vidljivost mape se kontroliše klikom na dugme "Google Maps" u levoj koloni korisničkog interfejsa/navigacije (u levom okviru na slici 13).

4.2.4 Modul korisničkog interfejsa

Korisnički interfejs (`module_ui.js`) ne treba da odvlači pažnju sa mape i podataka, te ne zauzima suviše mesta na ekranu. Elementi korisničkog interfejsa se takođe mogu lako skloniti i prikazati na ekranu, po potrebi korisnika. Dve osnovne celine korisničkog interfejsa pozicionirane su uz levu i desnu ivicu prostora za prikaz (detaljno predstavljeno na slici 13). Sa leve strane su osnovne informacije o mapi koja se posmatra: padajući meni za prikaz i izbor raspoloživih mapa, dugme za prikaz Google mape i glavni linkovi navigacije. Uz desnu stranu ekrana nalaze se opcije vezane za tip vizualizacije, način transformacije podataka kao i legenda mape koja se posmatra.

Po odabiru mape iz levog menija dinamički se formira desna strana interfejsa. Na osnovu podataka za odabranu mapu kreira se legenda prikazanih parametara. Pojedinačni parametri se mogu prikazivati i skrivati po želji korisnika. Ispod liste parametara nalaze se

pomoćni elementi mape (oznake mesta, nazivi mesta, reke) koji se po potrebi takođe mogu skrivati i prikazivati.

Na dnu desne kolone nalazi se kontrola animacije podataka mape. Klikom na dugme "Animiraj" animacija se pokreće i zaustavlja, dok se klikom na "Resetuj" merene vrednosti vraćaju na stanje u početnom trenutku. Tokom animacije se prikazuje i vreme merenja za koje su rezultati prikazani.

1..1.3 Dodatne informacije o merenim podacima

U okviru modula korisničkog interfejsa definišu se dva dodatna načina prikaza podataka:

1. Korišćenjem komponente *tooltip*
2. Korišćenjem komponente *dialog*

Obe komponenete implementirane su korišćenjem jQueryUI biblioteke [18]. Sadržaj se popunjava kroz povratnu (*callback)* funkciju kojoj se prosleđuju odgovarajući argumenti (na primer `id` odabrane stanice i/ili parametra nad kojim se nalazi kursor). *Tooltip* se aktivira prelazom preko reprezentacije vrednosti parametra dok se *dialog* otvara klikom na stanicu (ili bilo koju reprezentaciju merenog parametra koji joj pripada). Na slici 13 prikazan je primer upotrebe komponenti *tooltip* i *dialog*.

Slika 13 - Korisnički interfejs - (1) Izbor mape, osnovne informacije o mapi, glavni navigacija, (2) Izbor vizualizacije i tipa preslikavanja podataka, (3) Legenda, (4) Kontrola animacije, (5) Detaljne informacije u Dialog komponenti, (6) Tooltip

5 Primeri upotrebe

U svrhu primera, konkretna implementacija alata WebMap sadrži dve mape sa vrednostima merenim na teritoriji Republike Srbije. Jasno je da se u potencijalnom budućem razvoju vizualizacija ne ograničava na korišćenje jedne teritorije. Sledeća diskusija vezana za prikaz karte analogno važi i za bilo koju drugu teritoriju čiji su podaci za prikaz obezbeđeni na isti način.

5.1 Podaci potrebni za prikaz karte

Jednu od polaznih tačaka projekta predstavlja obezbeđivanje karte nad kojom će se prikazivati prikupljeni podaci sa stanica ili mernih lokacija. Detaljan prikaz geografskih i topoloških odlika nije neophodan jer je karta samo podloga za ono na čemu je akcenat celog projekta, a to su prikupljeni podaci. Imajući prethodno u vidu, za iscrtavanje karte (podloge) neophodna je samo geometrija granica države (ili regiona) za koju se podaci prikupljaju i prikazuju.

Pronalaženje geometrije tj. skupa poligona granica mape se logično nameće kao prvi problem na putu ka prikazivanju mape. Pored toga, mora se voditi računa o problemu pozicioniranja ostalih podataka na izabranu podlogu. Kako pozicionirati stanicu sa lokacijom određenom geografskom širinom i visinom na mapu države iscrtanu u odgovarajućim proporcijama ali sa tačkama koje nemaju veze sa geografskom širinom i visinom? Neophodna je (ili bar u mnogome olakšava problem) geometrija mape sa tačkama izraženim u geografskoj širini i visini (geografske koordinate) koju možemo projektovati u koordinate za prikaz. Naknadno pozicioniranje stanica i lokacija određenih geografskim koordinatama na taj način ne predstavlja nikakav problem.

Ne postoji mnogo izvora koji nude ove podatke za korišćenje bez nadoknade. Pretragom Interneta kao neki od relevantnih izvora javljaju se ***GeoCommons*** i ***Natural Earth***. Uporednim pregledom onoga što nude oba izvora izabran je ovaj drugi.

5.1.1 Natural Earth

Natural Earth [19] je projekat koji predstavlja proizvod rada kartografa *Nathaniel Vaughn Kelso*-a koji radi za *Washington post* [20]. *Natural Earth* je otvoren skup vektorskih i raster mapa koji održava zajednica okupljena oko ovog projekta. Sajt nudi mape kulturnih i fizičkih karakteristika u različitim razmerama i nivoima detalja. *Natural Earth* pruža mape u formatu *ESRI shapefile* (.shp) koji predstavlja standard za vektorske geo-podatke. Raster mape su u formatu TIFF. Sve mape koje se nalaze na sajtu koriste **"Geografski koordinatni**

sistem" (verzija standarda - **WGS84**) koji takođe koriste i sistemi za **GPS** (*Global positioning system*).

Geografski koordinatni sistem predstavlja skup parametara koji određuju lokacije na zemljinoj površini. Zemlja je predstavljena elipsoidom. Centar koordinatnog sistema je tačka koja predstavlja centar mase Zemlje. Polarna osa geografskog koordinatnog sistema poklapa se sa osom rotacije zemlje (prolazi kroz južni i severni pol). Osnovna (najveća) kružnica normalna na polarnu osu je ekvator.

Položaj lokacije severno ili južno u odnosu na ekvator (ugao između prave koja prolazi kroz lokaciju i centar koordinatnog sistema i ravni ekvatora) naziva se geografska širina ili **latituda** i može imati vrednosti od -90° (južni pol) do +90° (severni pol). Geografska širina ekvatora je 0°.

Meridijani su polu-elipse sa početnom i krajnjom tačkom na severnom i južnom polu. Pod nultim meridijanom obično se podrazumeva *Greenwich* meridijan ali je u slučaju WGS84 nulti meridijan pomeren za oko 100 metara istočno. Geografska dužina ili **longituda** predstavlja ugao između meridijana na kom se lokacija nalazi i nultog meridijana. Geografska širina ka istoku raste do maksimalnih 180°, dok se uglovi sa zapadne strane nultog meridijana predstavljaju negativnim vrednostima (do -180°).

U ovom radu bavimo se teritorijom Republike Srbije pa su korišćeni odgovarajući podaci sa sajta *Natural Earth*. Kombinovani su i kulturni i fizički podaci. Podaci korišćeni za prikaz mape Republike Srbije su (sa naslovima preuzetim sa sajta http://www.naturalearthdata.com - bez prevoda) :

- Kulturni - "*Admin 0 – Details*" - Granice pokrajina, državnih podnivoa
- Fizički - "*Rivers + lake centerlines*" - Poligoni velikih rečnih korita
- Fizički - "*Rivers + lake centerlines - Supplemental data for Europe*" - Dodatni poligoni rečnih korita u Evropi

Konverzija formata *ESRI shapefile* (.shp) koji nudi Natural Earth u GeoJSON na sreću ne predstavlja prevelik problem jer je zajednica koja se bavi ovom problematikom velika i dobro organizovana te postoje besplatni konvertori iz jednog formata u drugi. Biblioteka sa alatima *Geospatial Data Abstraction Library* (GDAL) sadrži alat `ogr2ogr` koji vrši konverziju iz pomenutog formata u GeoJSON.

5.2 Primer prikupljanja vrednosti vodostaja sa stranica HMZS

Kao što je navedeno, podaci za prikaz obezbeđeni su prikupljanjem sa veba. Podaci su ekstrahovani pomoću upitnog jezika XPath [21]. XPath je razvijen kao upitni jezik za izbor i izdvajanje čvorova u strukturi XML. Selektovanje čvorova je moguće po više različitih kriterijuma.

5.2.1 Prikupljanje osnovnih podataka o mernim stanicama

Stranica sa potrebnim informacijama:
http://www.hidmet.gov.rs/latin/osmotreni/nrt_index.php

Slika 14 - Lokacija željenog podatka i odgovarajući html

Primer preuzimanja DOM strukture za obradu pomoću XPath (element `body` je koreni element):

```
$xpath = new DOMXpath($doc);
$tbody = $doc->getElementsByTagName('body')->item(0);
```

Kako bi izdvojili nazive stanica (i `id`-eve koje koristi HMZS zbog reference i kasnijeg korišćenja) formira se sledeći XPath upit:

```
$name = $xpath->query('//table//td[@class="siva75 levo" or
                        @class="bela75 levo"]//a',$tbody);
```

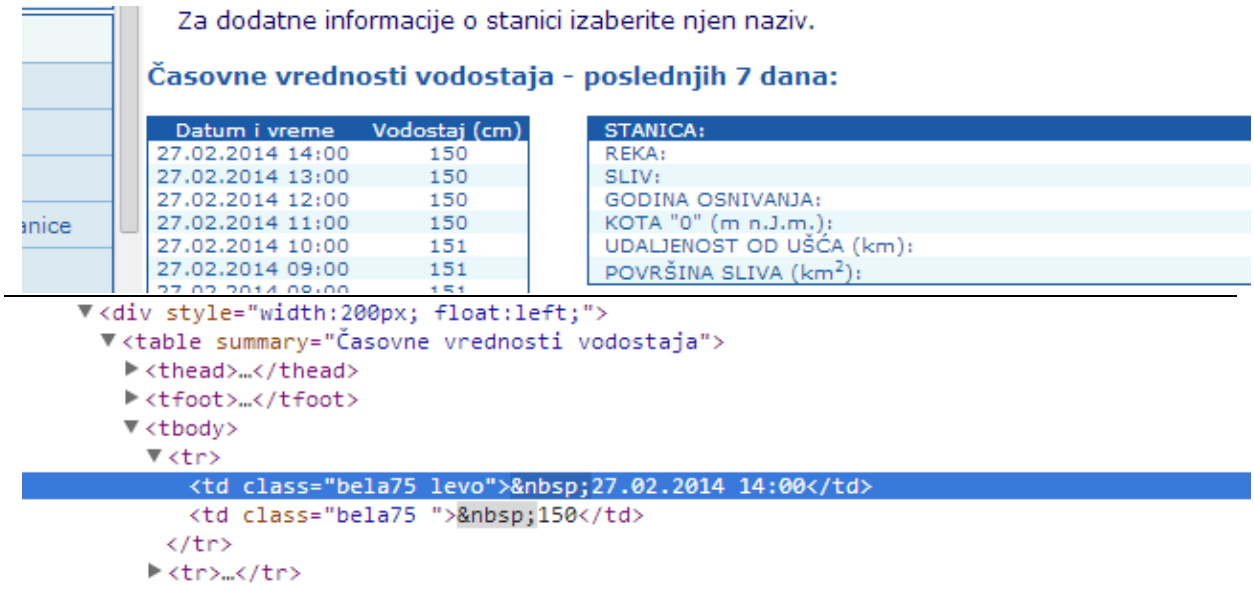
Ovo znači da sa XPath ulazimo u čvorove `<table>`, zatim u `<td>`, i to one sa za zadatom vrednošću atributa `class` i na kraju u čvorove `<a>` koji su nam i potrebni. Na ovaj način izbegavamo preostale tri kolone koje nam u ovom slučaju nisu interesantne. U čvoru `<a>` imamo sve potrebne informacije (pogledati sliku 14).

5.2.2 Prikupljanje vrednosti merenja

Vrednosti merenja se mogu naći na pojedinačnim stranicama mernih stanica, na primer: **http://www.hidmet.gov.rs/latin/osmotreni/nrt_tabela_grafik.php?hm_id=42010&period=7** prikazanoj na slici 15**.**

Stranica sa prethodnom URL putanjom sadrži vrednosti merenja za stanicu "Bezdan" (`id=42010`, sa slike 14). Sa prethodno pripremljenim nizom `id`-eva stanica koje koristi HMZS prolazimo kroz sve ove strane (za svaku stanicu) i prikupljamo željene vrednosti na sledeći način:

```
$time = $xpath->query('//table[@summary!=""]//td[@class="bela75 levo"
        or @class="bela75 "]',$tbody);
```



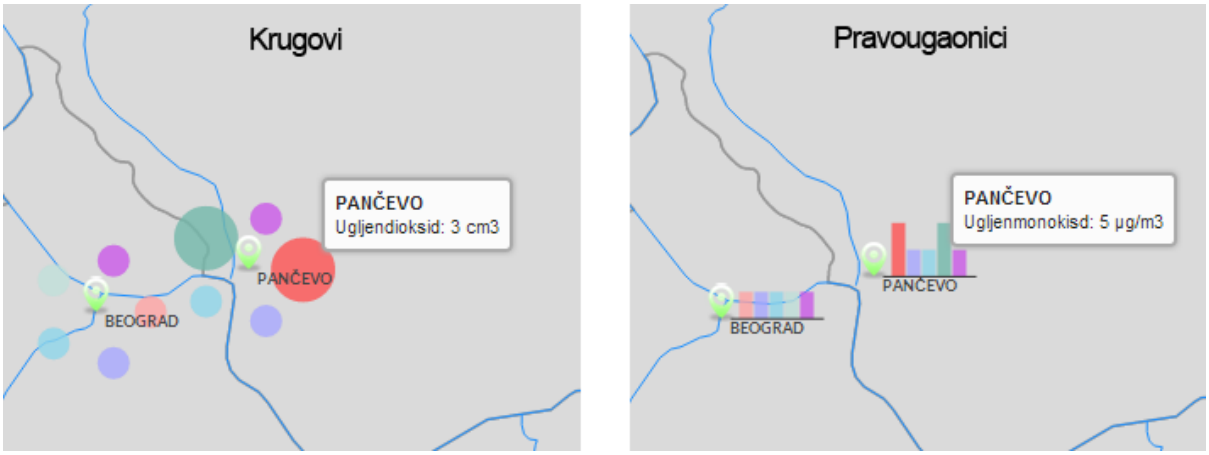
Slika 15 - Lokacija željenog podatka i odgovarajući html

Na osnovu prethodno definisanog upita izdvajaju se naizmenično čvorovi sa vremenom merenja i stvarnom vrednošću vodostaja.

Pripremljena mapa u alatu koja prikazuje nivo vodostaja sadrži merene vrednosti u razmacima od 8 sati u periodu od nedelju dana (od 26.07.2014. do 01.08.2014). Vrednosti nivoa kritičnih tačaka nisu realne (sem na pojedinim stanicama) ali dodavanjem vrednosti u konfiguracioni XML mape mogu se naknadno definisati.

5.3 Primer mape sa više mernih parametara

Mapa koja vizualizuje podatke o kvalitetu vazduha formira svoj prikaz na osnovu proizvoljnih vrednosti. Primer mape kvaliteta vazduha prikazuje kako se više različitih parametara može prikazati oko jedne stanice. Jasno je da vizualizacija povezanih krugova ovde nema neku ulogu, jer veze među stanicama nisu definisane. Sa druge strane primetno je kako prikaz vrednosti pravougaonicima predstavlja dobro rešenje kada se na istoj mapi prikazuje više različitih parametara. Primer vizualizacije više različitih parametara na istoj mapi prikazan je na slici 16.



Slika 16 - Primer vizualizacije više parametara (maa kvaliteta vazduha)

6 Diskusija i zaključak

Projekat razvoja alata WebMap imao je za cilj da istraži i predstavi alternativne načine prikaza geoprostornih podataka na prostoru Republike Srbije. Razvijeni alat predstavlja alternativu konkretnim rešenjima sa sličnom problematikom, koja su pomenuta u radu (HMZS i SEPA). Pored toga, WebMap predstavlja svojevrsnu demonstraciju korišćenja savremenih tehnika i alata u cilju iskorišćavanja potencijala Interneta i pristupačnosti informacija. Ukratko ćemo razmotriti postojeće rešenje i mogućnosti za unapređenje alata.

Prikupljanje podataka za prikaz

Implementiran je mehanizam za prikupljanje javnih podataka sa određenih stranica (HMZS). Trenutno se u okviru alata prikupljaju informacije sa određene stranice za određeni vremenski period. Ovo je segment gde se u eventualnom daljem razvoju objektivno može očekivati promena i poboljšanje. Izabrani podaci su demonstracionog karaktera, nisu podaci iz realnog vremena, ali kako je struktura aplikacije modularna, promena načina prikupljanja podataka neće uticati na ostale segmente aplikacije. Za razliku od trenutnog metoda prikupljanja sa veba, za prikupljanje podataka u budućnosti mogao bi se koristiti poziv ka veb servisu koji dostavlja podatke.

Administratorski deo alata

Alat sadrži osnovni "administratorski" deo koji omogućava korisniku pravljenje novih mapa, stanica i parametara i upravljanje postojećim podacima u bazi. Osnovna funkcionalnost postoji ali ima prostora za unapređenje kako funkcionalnosti tako i zaštite ovog segmenta alata.

Vizualizacije

Postoje tri osnovna tipa vizualizacije koja su dovoljna korisniku alata da razume potencijal koji ovaj segment rada poseduje. Moguće su gotovo nebrojene varijacije prikaza. Pored toga jasno je da se prikazi moraju prilagođavati prirodi podatka i odnosima stanica na kojima se podaci mere. Kao i ostalo, implementacija i dodavanje novih vizualizacija u alat je moguće i predviđeno prilikom određivanja arhitekture rešenja i njegove implementacije.

Interakcija

U alatu je moguće pomerati i menjati veličinu mape do određenih razumnih granica. Klikom na određenu mernu stanicu korisnik dobija detaljnije informacije o onome što gleda i što je relevantno za određenu stanicu. Izveštaji o izabranim stanicama predstavljaju osnovne

informacije, te je ovo takođe segment koji se može unaprediti. U nekom daljem razvoju, pri zahtevu korisnika za detaljnijim informacijama merenja za određenu stanicu, prikazao bi se novi okvir ili nova strana sa pristupom svim detaljima stanice i svim podacima merenja (a ne samo merenja u određenom trenutku).

Animacija promene vrednosti merenja

Moguće je razviti znatno složeniji sistem/modul za animiranje prelaza u toku vremena. Postoji mogućnost potpunog redefinisanja načina animiranja prelaza. Umesto smene diskretnih prikaza na određeni vremenski period moguće je osmisliti i implementirati transformacije stanja iz jedne vremenske tačke u drugu.

Značajno poboljšanje u daljem razvoju alata moglo bi da bude automatsko prikupljanje podataka za prikaz, bez intervencije "administratora alata". Vizualizacija najnovijih podataka prikupljenih sa automatskih mernih stanica značajno bi doprinela razumevanju podataka kroz nadgledanje promena vrednosti u realnom vremenu. Implementacija prethodno navedenog moguća je na više načina. Jedna mogućnost bi bila prikupljanje sa veba preko definisanih zakazanih vremenskih poslova (*Cron*). Drugi način bi bio pozivanje veb servisa stranice sa koje se podaci prikupljaju, takođe na određeni, definisani vremenski interval (u dogovoru sa vlasnicima stranice da se odgovarajući servisi omoguće na korišćenje). Moguće je takođe definisati veb servis u okviru alata koji bi stranice sa merenim podacima pozivom obaveštavale o promeni podataka sa njihove strane itd.

Nakon temeljnijeg sagledavanja implementiranog alata može se doći do zaključka da korišćenje relacione baze nije bilo neophodno te da bi svi podaci (pa i statični kao što su stanice i parametri) mogli biti definisani u okviru datoteka kao što su XML ili JSON. Ovo bi značajno olakšalo distribuciju alata stranicama koje bi potencijalno mogle da ga koriste (npr. u radu navedenim stranicama HMZS i SEPA).

Biblioteka D3 koja je korišćena u razvoju, zajedno sa mogućnostima koje pruža SVG, pokazala se kao zadovoljavajuć alat koji znatno olakšava proces vizualizacije podataka. Biblioteka nudi dobre mogućnosti za rad i sa SVG crtežima i sa geografskim podacima. Ipak, implementirane vizualizacije alata WebMap su poprilično jednostavni primeri prikaza merenih vrednosti. Moguće je formirati znatno složenije vizualizacije koje bi pored samog podatka na jednoj mernoj stanici uzimale u obzir i podatke u okruženju na osnovu čega bi se mogle formirati vizualizacije sa prelazima između tačaka u geoprostoru. Pored toga, moguće je uvesti i treću dimenziju u vizualizacije. Korišćenjem novih tehnika i biblioteka za rad u 3D prostoru na vebu kao što je npr. *WebGL* moguće bi bilo napraviti reljefne vizualizacije (treća dimenzija bi predstavljala vrednost parametra), ali bi i samu kartu mogli prikazati detaljnije, uzimajući u obzir detalje o elevaciji tla.

Stalnim razvojem Interneta kao izvora podataka i multimedijalnog sadržaja razvijaju se i tehnike za što bolje predstavljanje tih podataka korisnicima. Vizualna predstava podataka omogućava nam da informacije sagledamo na drugačiji način. Vizualizacija geoprostornih podataka omogućava onome ko podatke posmatra da istraži i uoči veze između izmerenih

vrednosti i lokacije merenja. U radu su prikazani različiti tipovi vizualizacije podataka kao alternativa postojećim rešenjima u upotrebi u Republici Srbiji. Osnovno unapređenje koje donosi alat WebMap u odnosu na postojeća rešenja navedena u radu predstavlja mogućnost pregleda i lokacije i vrednosti u jednom pogledu. Pored toga moguće je menjati način gledanja na podatke (kroz tri implementirane vizualizacije), animiranje promena vrednosti kroz vreme itd. Uz sve navedeno, WebMap ostavlja prostor ali i dobru osnovu za dalje unapređenje i dalji razvoj.

7 Dodatak

7.1 Podešavanje projekcije i crtanje karte

7.1.1 Podešavanje projekcije

```
/**
/* Prepare for drawing from JSON
 */
this.prepareRegions = function prepareRegions(json) {

        var w = window.innerWidth;
        var h = window.innerHeight;

        // zoom behaviour
        this.zoom = d3.behavior.zoom();

        // first guess for the projection
        var center = d3.geo.centroid(json);
        var scale = 1000;
        var offset = [ w / 2, h / 2 ];

        this.projection =
        d3.geo.mercator().scale(scale).center(center).translate(offset);

        this.path = d3.geo.path().projection(this.projection);

        // get the bounding box and from JSON coords
        // with first guessed projection
        var boundsBox = this.path.bounds(json);

        // calculate appropriate scale based on first guess
        var horScale = scale * w / (boundsBox[1][0] - boundsBox[0][0]);
        var verScale = scale * h / (boundsBox[1][1] - boundsBox[0][1]);
        var scale = (horScale < verScale) ? horScale : verScale;

        var offset = [ w - (boundsBox[0][0] + boundsBox[1][0]) / 2,
                       h - (boundsBox[0][1] + boundsBox[1][1]) / 2 ];

        // fix scale to 90%
        scale *= 0.9;

        // updated projection
        this.projection =
        d3.geo.mercator().center(center).scale(scale).translate(offset);

        this.path = this.path.projection(this.projection);
        this.scale = scale;

        // define zoom behavior according to new projection
        this.zoom
```

```
                .translate(this.projection.translate())
                .scale(this.projection.scale())
                .scaleExtent([ scale / 2, scale * 3 ])
                .on("zoom", zoomed);

        // call zoom on main group
        this.mainG.call(this.zoom);


        // VISUALIZE ALL ELEMENTS WHEN THIS IS DONE
        this.visualizeMap(this.jsonpath, json);

        }
```

7.1.2 Crtanje karte

```
/**
 * VISUALIZE MAP - SUBUNITS, PLACES...
 */
this.visualizeMap = function visualizeMap(jsonpath, json) {

// Show subunits (main) map
        this.mainG.append("g")
        .attr("id", "subunits").selectAll("path")
        .data(json.features).enter()
        .append("path")
        .attr("d", this.path)
        .style("fill", "#dadada").style('stroke', '#999')
        .style('stroke-width', 1.5);

// Add group for global data visualization
        this.mainG.append("g").attr("id", "global-data-visuals");

// define station places group
        this.mainG.append("g").attr("id", "places").selectAll()
        .data(this.stations).enter().append("g")
        .attr("class", "place")
        .attr("transform",
        function(d) {
                return "translate(" +
                        SVGMAPpresentation.projection([ d.lon, d.lat ]) +
                        ")";
        });

// station specific data visualization
        this.mainG.selectAll(".place").append("g")
        .attr("class", "data-visuals");

// add LABELS to places
        this.mainG.selectAll(".place").data(this.stations)
        .append("text").attr("class", "map-element-label")
        .attr("dy", "1em").attr("dx","0.5em")
        .style("fill", "#444")
        .style("font-size", 9.5)
        .style("font-weight", "bold")
        .style("pointer-events", "none")
        .text(function(d) {
```

```
                return d.name;
        });

// add PINS to places
        this.mainG.selectAll(".place").append("svg:image")
        .attr("class", "map-element-pin").attr('x', -10)
        .attr('y', -21)
        .attr('width',20)
        .attr('height', 23)
        .attr("xlink:href",
        SVGMAPmodule.assetspath + "images/greenpin.png");

// add on click handler for a place
        this.mainG
        .selectAll(".place")
        .on("click",function(d) {
                UImodule.showDialog(d);
                GMAPmodule.positionOverPoint(d.lat, d.lon);
        });

// LETS DO SOME OTHER VISUALIZATIONS /////
// draw map features
drawFeatures();

}
```

7.2 Vizualizacija kružićima

```
/**
 * Basic circles visualization
 */
this.vCircles = function vCircles() {

this.clearVisuals();

        MULTIPLIER = Math.min(this.startHeight, this.startWidth) / 25;
        // R for offset xy position of parameter
        if (this.params.length < 2)
                R = 0;
        else
                R = MULTIPLIER / 1.2;

// for each parameter
for ( var i = 0; i < this.params.length; i++) {
        color = this.params[i].color;

        offsetX = R * Math.cos(i * 2 * Math.PI / this.params.length);
        offsetY = R * Math.sin(i * 2 * Math.PI / this.params.length);

        this.mainG
                .selectAll(".data-visuals")
                .data(this.stations)
                .append("circle")
                .attr("class", "param_" + this.params[i].id)
                .attr( "r",
        function(d) {
```

```
        var values =
        SVGMAPmodule.normValues[d.id][SVGMAPmodule.params[i].id];
        if (values == null)
                return 0;
        if (values.length > 0 && typeof values[frame] !== 'undefined')
        return Math.abs(values[frame].value * MULTIPLIER);
        else
                return 0;
                })
                .attr("fill",
        function(d) {
        var values =
        SVGMAPmodule.normValues[d.id][SVGMAPmodule.params[i].id];
        if (values.length > 0 && typeof values[frame] !== 'undefined')
        if (values[frame].color!="")
                return values[frame].color
        else return color;
        })
                .style("fill-opacity", "0.8")
                .attr("cy", offsetY + "px")
                .attr("cx", offsetX + "px")
                .attr("tooltip",function(d) { return "s="+d.id+"&p="+i });
                }

        }
```

7.3 Vizualizacija pravougaonicima

```
/**
 * Basic bars visualization
 */
this.vBars = function vBars() {
this.clearVisuals();

if (this.params.length < 2)
        W = 12;
else if (this.params.length < 6)
        W = 8;
else
        W = 6;

        MULTIPLIER = Math.min(this.startHeight, this.startWidth) / 15;

// for each parameter
for ( var i = 0; i < this.params.length; i++) {
        color = this.params[i].color;
        offsetX = 10 + i * W + i;

        this.mainG
                .selectAll(".data-visuals")
                .data(this.stations)
                .append("rect")
                .attr("class", "param_" + this.params[i].id)
                .attr("width", W + "px")
                .attr("height",
        function(d) {
```

```
        var values =
        SVGMAPmodule.normValues[d.id][SVGMAPmodule.params[i].id];
        if (values.length > 0)
                return Math.abs(values[frame].value * MULTIPLIER) + "px";
        else
                return 0;
        })
                .attr("fill",
        function(d) {
        var values =
        SVGMAPmodule.normValues[d.id][SVGMAPmodule.params[i].id];
        if (values.length > 0 && typeof values[frame] !== 'undefined')
                if (values[frame].color!="")
                        return values[frame].color
                else return color;
        })
                .style("fill-opacity", "0.8")
                .attr("x", offsetX + "px")
                .attr("y",
        function(d) {
        if (typeof SVGMAPmodule.values[d.id] !== 'undefined')
        if (typeof SVGMAPmodule.values[d.id][SVGMAPmodule.params[i].id]
                !== 'undefined') {
        if (SVGMAPmodule.values[d.id][SVGMAPmodule.params[i].id].length>0)
        if(SVGMAPmodule.normValues[d.id][SVGMAPmodule.params[i].id][frame].
value>0)
        return
-SVGMAPmodule.normValues[d.id][SVGMAPmodule.params[i].id][frame].value
* MULTIPLIER + "px";
        else return "0px";
        else return "0px";
        }
        return "0px"
        })
                .attr("tooltip",
        function(d) {
                return "s="+d.id+"&p="+i });
        }

// bottom line
        this.mainG
                .selectAll(".data-visuals")
                .append("line")
                .attr("class", "shared")
        .attr("x1", 5 + "px")
        .attr("y1", 0)
        .attr("x2", this.params.length * W + 20 + "px")
        .attr("y2", 0)
        .style( "stroke", "#333").style("stroke-width", "1px");
        }
```

8 Literatura

1. Fry Ben**, Visualizing Data** (1), *O'Reilly Media, Sebastopol CA*, 2008, 978-0-596-51455-6
2. Wikipedia, **Geovisualization,** *Wikipedia,* `http://en.wikipedia.org/wiki/Geovisualization` (@2013)
3. Kraak Menno-Jan, **Geovisualization illustrated**, *ISPRS Journal of Photogrammetry & Remote Sensing*, ed. *M. Madden, J. Schiewe,* Volume (57), 390-399, 2003.
4. Google, **Google Maps**, *Google developers* `https://developers.google.com/maps/` (@2013)
5. Wikipedia, **Web scraping,** *Wikipedia,* `http://en.wikipedia.org/wiki/Web_scraping` (@2013)
6. EllisLab, Inc. **CodeIgniter**, *CodeIgniter*, `http://ellislab.com/codeigniter` (@2014)
7. Derek Greer, **Interactive Application Architecture Patterns,** *Aspiring Craftsman***,** `http://aspiringcraftsman.com/2007/08/25/interactive-application-architecture/` (@2014)
8. Ecma International**, Introducing JSON,** *JSON,* `http://json.org/` (@2014)
9. Mihai Sucan, **SVG vs Canvas**, *Dev.Opera*, `http://dev.opera.com/articles/view/svg-or-canvas-choosing-between-the-two/` (@2013)
10. W3Schools, **Introduction to XML,** *w3schools*, `http://www.w3schools.com/xml/xml_whatis.asp` (@2014)
11. Howard Butler, Martin Daly, Allan Doyle, Sean Gillies, Tim Schaub, Christopher Schmidt, **The GeoJSON Format Specification,** *GeoJSON,* `http://geojson.org/geojson-spec.html` (@2014)
12. W3C, **Scalable Vector Graphics (SVG),** *W3C,* `http://www.w3.org/TR/SVG/` (@2014)
13. Bostock Mike, **D3,** *GitHub*, `https://github.com/mbostock/d3/wiki` (@2013)
14. W3C, **Selectors API,** *W3C,* `http://www.w3.org/TR/selectors-api/` (@2013)
15. Angus Croll, **Understanding JavaScript Closures**, *Javascriptweblog*, `http://javascriptweblog.wordpress.com/2010/10/25/understanding-javascript-closures/` (@2014)
16. PHP, **PHP - Hypertext Preprocessor**, *PHP*, `http://php.net` (@2014)
17. Bostock Mike, **Geo-Projections,** *GitHub*, `https://github.com/mbostock/d3/wiki/Geo-Projections` (@2014)

18. jQuery, **jQueryUI,** *jQueryUI,*
`http://jqueryui.com/` (@2014)
19. Natural Earth, **Natural Earth,** *Natural Earth Data,*
`http://www.naturalearthdata.com/` (@2013)
20. Nathaniel Vaughn, **Kelso Cartography,** *Kelso Cartography,*
`http://kelsocartography.com/` (@2013)
21. W3C, **XML Path Language,** *W3C,*
`http://www.w3.org/TR/xpath/` (@2014)