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 Gradbeništvo,
Prometna smer

Kandidat:
Peter Pupovac

Spletna aplikacija za izmenjavo in prikaz prometnih podatkov

Diplomska naloga št.: 3037

Mentor:
doc. dr. Marijan Žura

Somentor:
Iztok Levart

Ljubljana, 3. 11. 2008

IZJAVA O AVTORSTVU

Podpisani **PETER PUPOVAC** izjavljam, da sem avtor diplomske naloge z naslovom:

»SPLETNA APLIKACIJA ZA IZMENJAVO IN PRIKAZ PROMETNIH PODATKOV«

Izjavljam, da prenašam vse materialne avtorske pravice v zvezi z diplomsko nalogo na UL, Fakulteto za gradbeništvo in geodezijo.

Ljubljana, 12.10.2008

BIBLIOGRAFSKO – DOKUMENTACIJSKA STRAN IN IZVLEČEK

UDK:

Avtor: **Peter Pupovac**

Mentor: **doc. dr. Marijan Žura**

Naslov: **Spletna aplikacija za izmenjavo in prikaz prometnih podatkov**

Obseg in oprema: **64. str., 4. pregl., 34. sl.**

Ključne besede: **Google Maps, Datex II, spletni program**

Izvleček:

Hiter napredek spletnih tehnologij, kakor tudi sodobne navigacijske opreme omogoča večjo obveščenost voznika o trenutnem prometnem stanju. Stanje na cesti lahko voznik preverja med vožnjo ob uporabi sodobne GPS navigacijske naprave, lahko pa preveri stanje tudi na internetu preden se poda na pot (preko telefona, PDA-ja ali osebnega računalnika). Diplomsko delo opisuje integracijo spletne tehnologije Google Maps in Evropskega standarda za izmenjavo prometnih podatkov Datex II pri prikazovanju točkovnega dogodka na cesti. Za potrebe diplome je to niz prometnih nesreč na avtocestnem odseku H4 Razdrto – Vrtojba. Diploma zajema prometno-informacijski, programerski in grafični del izdelave takega programa

BIBLIOGRAFIC – DOCUMENTALISTIC INFORMATION

UDC:

Author:	**Peter Pupovac**
Supervisor:	**doc. dr. Marijan Žura**
Title:	**Web application for exchanging and visualisation of traffic data**
Notes:	**64. p., 4. tab., 34. fig.**
Key words:	**Google Maps, Datex II, web application**

Abstract:

The fast progress of web technologies and modern navigation devices is helping the driver to gain real time information about the current traffic situation. The information can be accessed while driving through modern GPS devices or by accessing the internet (via the cell telephone, PDA or personal computer) before he actually hits the road. The following paper describes the integration of web technologies such as Google Maps with the European standard Datex II used for exchanging traffic related information, to display a point event on the road. The paper will describe a series of road accidents on the highway section H4 Razdrto – Vrtojba.

ZAHVALA

Zahvaljujem se družini in prijatelju Marku za podporo in pomoč pri študiju. Zahvalil bi se rad še sledečim profesorjem, mentorju doc. dr. Marijanu Žuri, so-mentorju u.d.i.g. Iztoku Levartu, asist. mag. Robertu Rijavcu in podjetju Traffic Design d.o.o.

Diplomo posvečam svojemu preminulemu dedku Aleksandru Renerju.

1 KAZALO VSEBINE

KAZALO PREGLEDNIC

KAZALO SLIK

1 UVOD

Ko se z avtom odpravljamo na pot, želimo v okviru cestno prometnih predpisov, priti v najkrajšem možnem času do cilja. Poznavanje trenutnega stanja na cesti, je pri planiranju poti bistvenega pomena, saj se tako lahko izognemo cestam na katerih je prepustnost omejena ali celo onemogočena. Situacije kot so dela na cesti, kolone ob urnih konicah so dokaj predvidljive narave, zato želimo poznati njihovo lokacijo, da se jim uspešno izognemo. Alternativna pot, ki jo uberemo je lahko celo daljša, kar pa še ne pomeni, da bomo na koncu prišli do cilja hitreje in porabili manj goriva.

Bolj nepredvidljive narave so prometne nesreče, ki pa jih v nobenem primeru ne morem natančno (če sploh) napovedati. Poznavanje lokacije nesreče nam omogoči, da uberemo pravi izhod na avtocesti, se zapeljemo na vzporedno lokalno cesto in že smo pred kolono vozil, v kateri bi sicer lahko čakali več ur. Pri zbiranju podatkov o zapletih na cestah, se lahko obrnemo na medije kot so radijske postaje, teletekst ali internet. Na slednjem je država Slovenije v okviru prometno informacijskega centra PIC postavila spletni portal www.promet.si, ki je svoj čas ustrezno opravljal delo. S porastom naprednih spletnih tehnologij kot je Google Maps in uvedbo standarda DATEX II pri izmenjavi prometnih podatkov na področju evropske unije, je nastala potreba po novem, sodobnem in preglednejšem spletnem portalu.

V diplomski nalogi bom obravnaval koncepte, načine in metode, ki so potrebni pri izdelavi prometnega portala, ki bo sposoben prepoznati podatke zapisane v obliki, ki jo določa evropski standard za izmenjavo prometnih podatkov DATEX II in prikazati situacijo s tehnologijo Google Maps. Slednja je trenutno najbolj uveljavljena in razširjenja spletna tehnologija na področju digitalnega kartiranja. Njena glavna prednost je prosti dostop, obsežna dokumentacija in uporabniška podpora. Za potrebe diplome se bom omejil na točkovne dogodke, ki predstavljajo prometne nesreče na avtocestnem odseku H4 Razdrto – Vrtojba.

2 OPIS SPLETNEGA PROGRAMA ZA PRIKAZOVANJE PROMETNIH DOGODKOV

2.1 Opis

Program prikazuje točkovne prometne dogodke, kot so prometne nesreče, preko interneta z uporabo spletne tehnologije Google Maps *(Slika 2.1.1)*. Podatki o dogodkih so zapisani v XML datotekah, ki so oblikovane po shemi DATEX II Evropskega standarda za izmenjavo prometnih podatkov.

Vsak dogodek je na Google Maps karti prikazan z lokacijo, ikono in informacijskim oknom, ki prikazuje osnovne podatke o dogodku. Aplikacija se nahaja na spletnem naslovu: http://www.googlemaps.si/Program/diploma.html, ki smo ga pri internetnem ponudniku registrirali za namen diplome.

Slika 2.1.1 *Google Maps karta z vrisanima dvemi prometnimi nesrečami*

2.2 Google Maps API

Google je ustvaril nabor ukazov Google Maps API *(»application programming interface«)* z namenom, da bi olajšal razvijalcem programske opreme integracijo Google-ovih kart v njihove spletne strani. Storitev je brezplačna in trenutno ne vsebuje reklam. V pogojih uporabe je omenjeno, da ima Google pravico vstaviti reklame v prihodnosti, če se bo tako odločil.

Z uporabo Google Maps API, lahko vstavljamo celotno storitev Google Maps na našo (eksterno) spletno stran. Ob registraciji nam bo ponujen API ključ, ki bo veljal samo za eno spletno domeno. Ustvarjanje uporabniško spremenjenih kart temelji na dodajanju Google Javascript kode na svojo spletno stran z namenom prikazovanja podatkov točkovne in linijske narave. Posnetki kart se nalagajo direktno iz Google-ovih strežnikov.

2.3 Pridobivanje podatkov

Podatki o nesrečah so zapisani v obliki XML datotek, ki naj bi jih generiral prometni informacijski center podjetja Traffic design d.o.o. in nam jih naložil v direktorij *xmlInput*. Ta se nahaja na istem strežniku kot Google Maps skripta in naš spletni program. Pogoj iste lokacije podatkov in Javascript kode (*»same origin policy«*), je varnostna omejitev programskega jezika Javascript. To pravilo predstavlja varnostni sistem pred zlorabami, saj skriptam preprečuje dostop do podatkov kot so uporabniška gesla in imena na drugih strežniki. Ista lokacija oziroma isti izvor obstaja takrat, ko imamo isto ime domene, protokol in port. Dve strani pripadajo istemu izvoru le v primeru, če so te tri vrednosti enake.

2.4 Vsebina direktorija programa

Opis datotek in direktorijev:

Vsebina korenskega direktorija `Program`:

- `/images/` ...direktorij z slikami ikon in gradniki informacijskega okna
- `/xmlInput/` ...direktorij, kjer PIC potisne XML datoteke s podatki o nesreči

- `/xmlOutput/` ...direktorij z izhodno XML datoteko, ki vsebuje informaciji o več dogodkih
- `locationTable.xml` ...podatki o lokacijskih točkah
- `polyline.xml` ...zgoščena linija, ki aproksimira cesto
- `trans_conf.xml` ...datoteka z nastavitvami transformator.php datoke
- `transformator.php` ...skripta, ki bere podatke iz xmlInput in ustvari datoteko `dogodki.xml`
- `seconds2HMS.js` ...skripta za pretvarjanje sekund v format ure:minute:sekunde
- `extinfowindow.js` ...Google-ova skripta, ki definira obliko informacijskega okna

Vsebina direktorija images:

- `/corners/` ...direktorij z grafičnimi elementi informacijskega okna
- `accident0, accident1 in accident2` ...ikone marker-ja
- `shadow-Accident0, shadow-Accident0, shadow-Accident0` ...sence marker-ja

Vsebina poddirektorija corners:

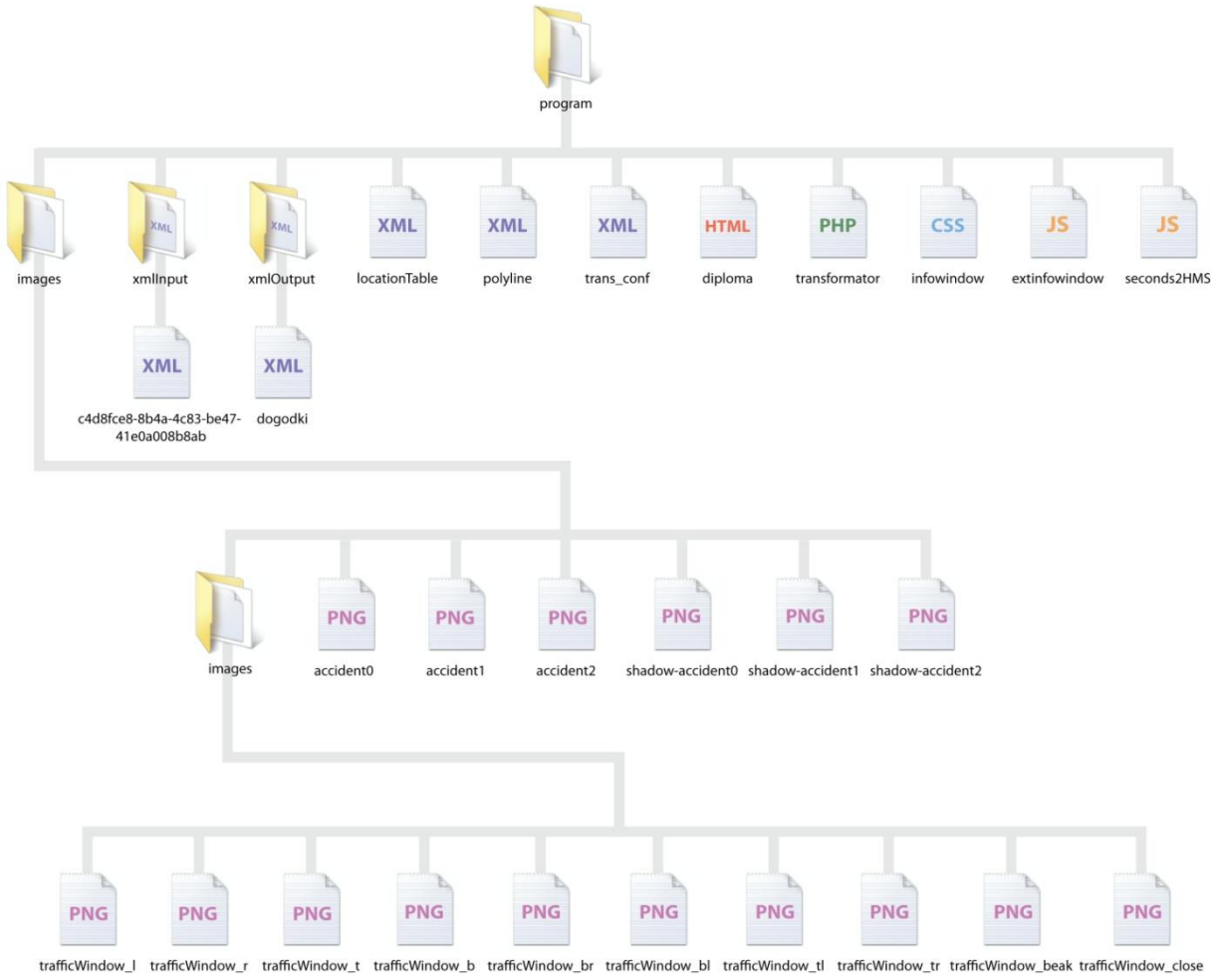
`trafficWindow_b, trafficWindow_t, trafficWindow_l, trafficWindow_r, trafficWindow_tr, trafficWindow_tl, trafficWindow_bl, trafficWindow_br` ...elementi informacijskega okna

Vsebina direktorija `/xmlInput/`:

- XML datoteke z dogodki na cesti. Datoteke imajo naključno generirana imena.

Vsebina direktorija `xmlOutput`:

- `dogodki.xml`, iz katere preberemo končno situacijo na cesti



Slika 2.4.1 *Drevesna struktura spletnega programa (vir: lasten vir)*

3 NASTAVITEV KARTE GOOGLE MAPS

3.1 Pridobitev API ključa

Preden se začnemo ukvarjati z Google Maps aplikacijo je prvo potrebno pridobiti API (»application program interface« - programski vmesnik aplikacije) ključ. To lahko storimo pod pogojem, da se strinjamo z Google Maps pogoji uporabe, ki jih dobimo na internetnem naslovu http://code.google.com/apis/maps/terms.html.

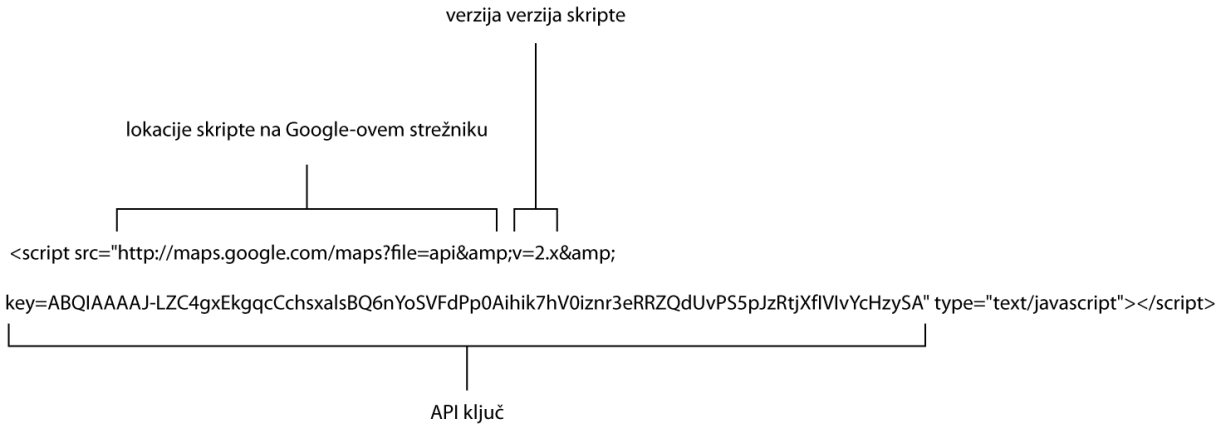
Google izda toliko ključev kolikor jih potrebujemo, edina omejitev je, da več različnih ključev na isti domeni (in poddpomenah) ne moremo uporabiti. Ključu je potrebno določiti korenski naslov naše domene, to storimo na naslovu http://code.google.com/apis/maps/signup.html, kjer se tudi prijavimo za pridobitev ključa. V našem primeru je korenski naslov na strežniku http://www.googlemaps.si.

Primer ključa:

```
ABQIAAAAJ-LZC4gxEkgqcCchsxalsBQ6nYoSVFdPp0Aihik7hV0iznr3eRRZQdUvPS5p
JzRtjXfIVIvYcHzySA
```

3.2 Definiranje Google Maps karte v HTML datoteki

Za uporabo nabora ukazov Google Maps API, je potrebno v HTML datoteki, določiti pot do Google-ovega strežnika, kjer se skripta nahaja. Nastaviti je potrebno tudi različico skripte, ki naj se izvaja. Za primer te diplome smo jo nastavili na različico `v = 2.x`, ki zajema vse najnovejše posodobitve od različice 2.0 naprej.



Slika 3.2.1 *Klic skripte Google Maps API s ključem in verzijo skripte (vir: lasten vir)*

Glavni pogoj za delovanje Google Maps karte je brskalnik z omogočenim Javascript-om. Če ta ni vklopljen karte ni mogoče zagnati.

Kompatibilnost brskalnika z Google Maps karto preverimo s pogojem `if ( GBrowserIsCompatible())`, če je ta veljaven (vrednost `true`) koda ustvari objekt `GMap2`, ki predstavlja karto.

Karto v HTML kodi predstavlja `DIV` element, ki mu pripišemo lastnost `id = "map"`. Dimenzije karte so poljubne. Za naš primer določimo z velikostjo DIV elementa:

- Dolžina: 800 pikslov
- Širina: 800 pikslov

Urejamo lahko tudi ostale lastnosti kot so barva in debelina obrobe ter postavitev glede na ostale elemente na spletni strani.

3.2.1 Primer:

```
<div id="map"
     align="center"
     style="border:2px solid green;
            width: 800px;
            height: 800px">
</div>
```

3.3 Objekt `GMap2`

Karto definiramo v obliki spremenljivke `map`, ki mora biti objekt tipa `GMap2`.

```
map = new GMap2(document.getElementById("map"))
```

Objekt `GMap2` ustvari Google Maps karto v `DIV` elementu (v našem primeru element z lastnostjo `id = "map"`).

3.3.1 Metoda `setCenter()`

Metoda `setCenter()` objekta `GMap2`, določa začetno pozicijo in stopnjo povečave. Začetno pozicijo določimo z objektom `GLatLng`, medtem ko stopnja povečave zavzema vrednosti od 0 do 19. Kjer 0 predstavlja najmanjšo možno povečavo, 19 pa največjo.

3.3.2 Prime**r**:

```
map.setCenter(new GLatLng(45.917683, 13.718296), 9);
```

V tem primeru je:

- Latituda enaka ...45.917683
- Longituda enaka ...13.718296
- Stopnja povečave ...12

3.4 Pripomočki za pregledovanje karte

Google maps karta premore več različnih tipov pripomočkov s katerimi si olajšamo njeno pregledovanje. Tej obsegajo merilo, gumbe za premikanje po karti, drsnik za povečavo karte, gumbi za vklop terenskega, satelitskega in zemljevidnega načina *(Slike 3.4.1.1, 3.4.1.2, 3.4.1.3, 3.4.1.4 in 3.4.1.5).* Pripomočki so definirani v Google Maps API kot objekti. Vsak objekt ima svoje lastnosti (metode), ki jim lahko še dodatno spreminjamo lastnosti.

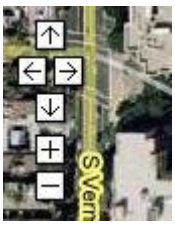
Kontrole dodajamo objektu `GMap2` preko metode `addControl()`.

3.4.1 Orodja za premikanje, pomanjševanje in premikanje po karti

- Za povečavo in pomanjšanje povečave karte samo z ukazi (+) in (-) imamo objekt: `GSmallZoomControl()`

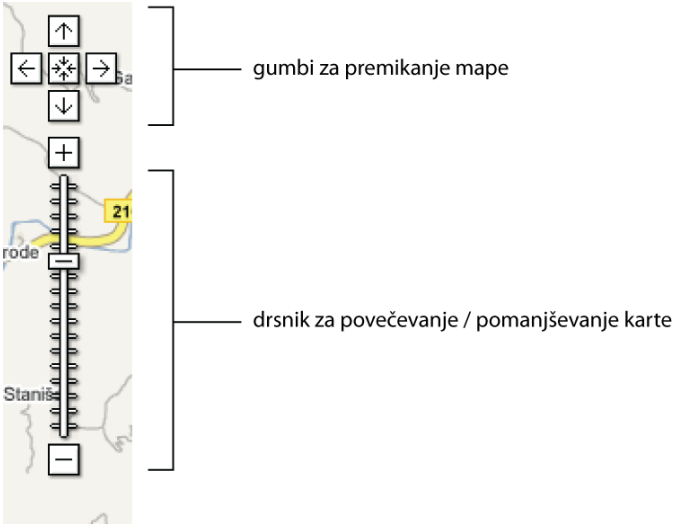
Slika 3.4.1.1 *Gumbi za povečavo (+) in pomanjšanje karte (-) (vir: lasten vir)*

- Za povečavo in pomanjšnje povečave karte z ukazi (+) in (-) ter gumbi za premik po karti imamo objekt `GSmallMapControl()`



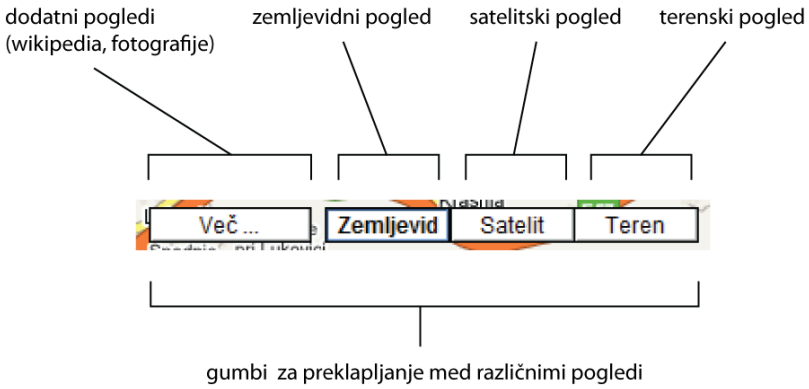
Slika 3.4.1.2 *Gumbi za premikanje, povečavo (+) in pomanjšanjemo karte (-) (vir: lasten vir)*

- Za povečavo in pomanjšanje z ukazi (+) in (-) in drsnikom ter gumbi za premik po karti imamo objekt `GLargeMapControl()`



Slika 3.4.1.3 *Gumbi za premikanje, povečavo (+) in pomanjšanje karte (-) z drsnikom (vir: lasten vir)*

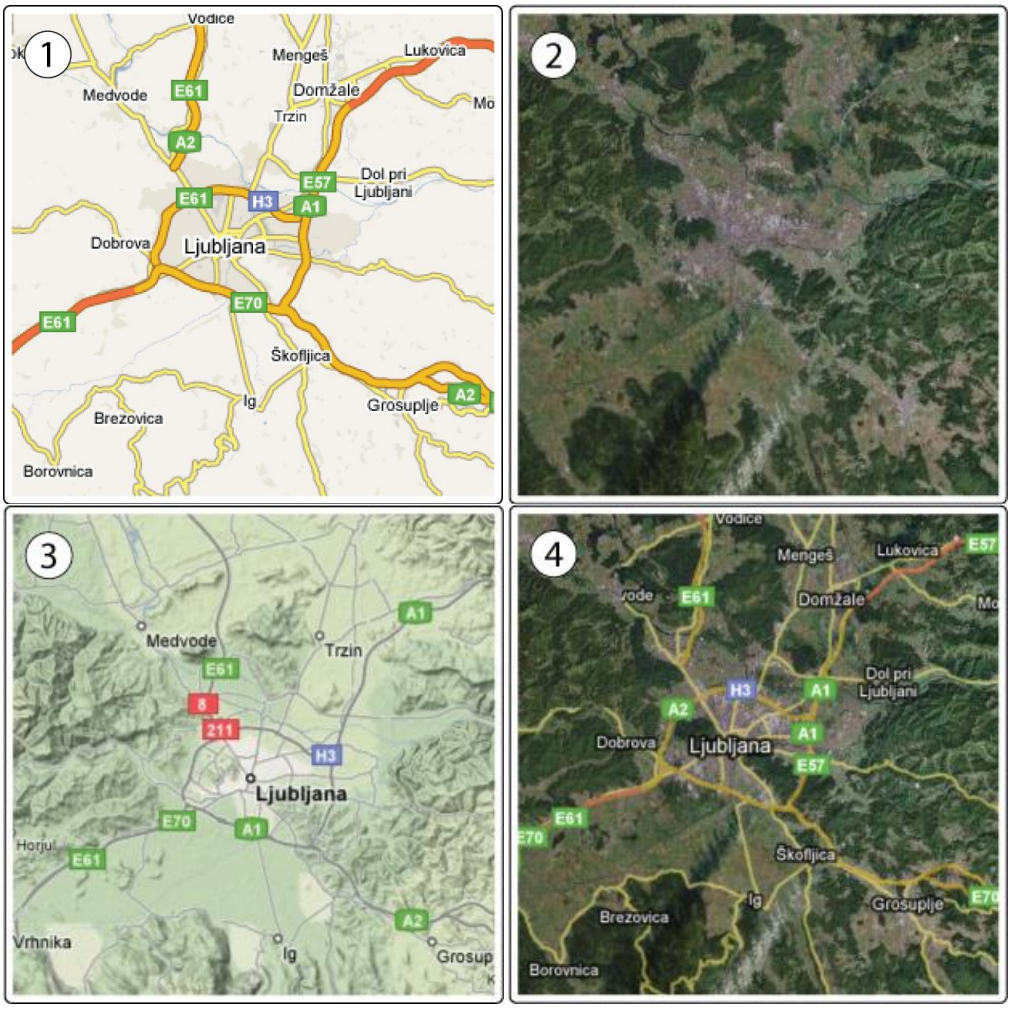
- Preklapljanje med različnimi pogledi omogoča objekt `GMapTypeControl()`.



Slika 3.4.1.4 *Gumbi za preklaplanje med različnimi pogledi (vir: lasten vir)*

Karto lahko prikazujemo na štiri različne načine (*Slika 3.4.1.4*):

1. **Zemljevid** ...zemljevid z vrisanimi cestami, ulica, stavbami
2. **Satelit** ... satelitska slika brez nobenih oznak
3. **Teren** ...vrisane so ceste, ulice, stavbe, kakor tudi višinske plastnice
4. **Hibridna** ...združuje lastnosti zemljevida in satelitske slike



Slika 3.4.1.5 *Tipi kart: 1.) Zemljevid 2.) Satelit 3.) Teren 4.) Hibridna (vir: lasten vir)*

Trije osnovni pogledi (zemljevid, satelit in hibridni pogled) so že zajeti z objektom `GMapTypeControl()`. Posamezno jih dodajamo kot metode objekta `GMap2`:

1. `map.setMapType(G_NORMAL_MAP)` ...za zemljevid
2. `map.setMapType(G_SATELLITE_MAP)` ...za satelitsko karto
3. `map.setMapType(G_HYBRID_MAP)` ...za hibridno karto
4. `map.setMapType(G_PHYSICAL_MAP)` ...za terensko karto

Določimo lahko tudi kateri bo privzeti tip karte:

`map.setMapType(G_PHYSICAL_MAP)` ...za privzeto terensko karto

- Za prikaz merila imamo objekt `map.addControl(new GScaleControl())`

Slika 3.4.1.5 *Merilo (vir: lasten vir)*

3.5 Funkcija `setupMap()`

Funkcija `setupMap()` se zažene, ko se naloži element `<body>` , to storimo z ukazom `onload`. Ko zapustimo stran, pa težimo k tem, da sprostimo pomnilnik karte. To dosežemo s funkcijo `GUnload()`, ki jo pokličemo ob dogodku `onunload`. Enkrat, ko je ta funkcija zagnana vsi `map` objekti, ki smo jih predhodno ustvarili postanejo neuporabni.

```
<body onload="setupMap()"; onunload="GUnload()">
```

3.5.1 Primer funkcije `setupMap()`:

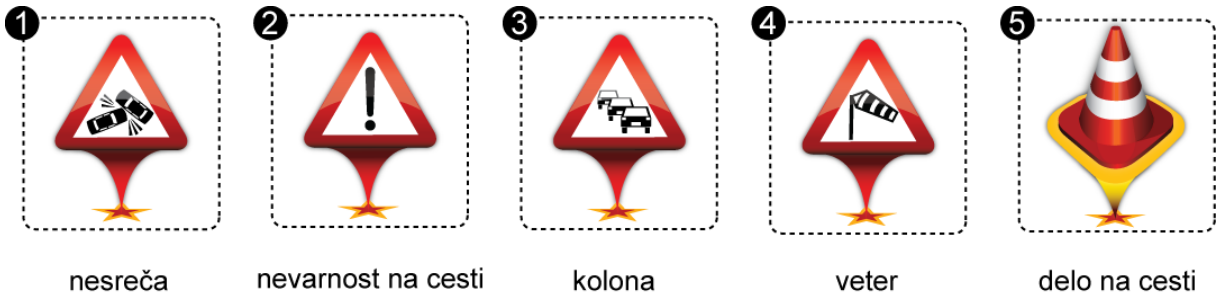
```
function setupMap() {

      if (GBrowserIsCompatible()) {
            map = new GMap2(document.getElementById("map"));
            map.setCenter(Ljubljana, 9);
            map.addControl(new GLargeMapControl());
            map.addControl(new GMapTypeControl());
            map.addControl(new GScaleControl());
   }
```

4 MARKER

4.1 Opis

Če hočemo prikazati natančno prikazati mesto določenega točkovnega dogodka (npr. prometne nesreče, zastoja, dela na cesti, itd. *(Slika 4.1.1*) to storimo z elementom marker *(Slika 4.1.2),* ki je objekt tipa `GMarker`.

***Slika 4.1.1** označevanje prometne nesreče z marker-jem (vir: lasten vir)*



***Slika 4.1.2** Različni markerji za prikaz različnih dogodkov na cesti (vir: lasten vir)*

Prvo ga je potrebno definirati:

```
var marker = new GMarker(point,icon)
```

Objekt `GMarker` ima dva vhodna parametra `point`, ki je objekt tipa `GLatLng` in `icon`, ki je objekt tipa `GIcon`. Marker sestavlja:

- slika ikone marker-ja (GIF ali PNG format)
- točka na katero se pritrdi informacijsko okno (objekt `Gpoint`)
- točka s katero se marker pritrdi na karto (objekt `GPoint`)
- slika senca marker-ja (GIF ali PNG format)

4.2 Objekt `GLatLng`

Objekt `GLatLng` predstavlja dvojico zemljepisnih koordinat, torej latitudo in longitudo, ki ustrezajo Mercatorjevi projekciji.
Definiramo ga tako:

```
var point = new GLatLng(45.917683615569096, 13.718291342278551)
```

kjer prva vrednost predstavlja latitudo druga pa longitudo.

Če imamo že obstoječi `GLatLng` objekt, iz njega pa hočemo pridobiti latitudo ali longitudo, uporabimo metodo `lat()` in `lng()`:

```
var latituda = point.lat()
var longituda = point.lng()
```

4.3 Objekt `GIcon`

Objekt `GIcon` predstavlja sličico ikone marker-ja. Za poznavanje delovanja objekta `GIcon` je potrebno prvo poznati delovanje objektov `GPoint` in `GSize`.

4.3.1 Objekt `GPoint`

`GPoint` je objekt, ki predstavlja dvojico koordinat X in Y, ki so merjene v pikslih.
Koordinatni sistem lokalen in je vezan na objekt preko katerega ga kličemo.

```
var anchor = new GPoint(20,12)
```

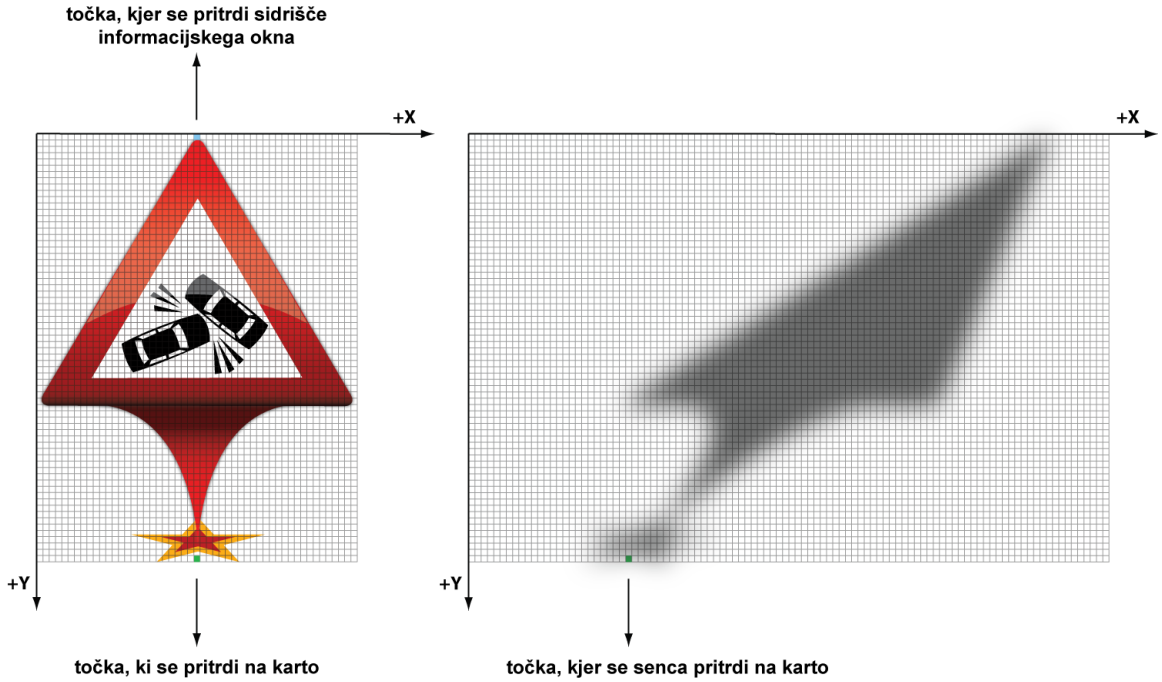
4.3.2 Objekt `Gsize`

`GSize` je objekt, ki predstavljajo dolžino in širino v pikslih objekta (npr. marker-ja, sence marker-ja).

```
var size = new GSize (50,100)
```

Sedaj lahko podrobno pogledamo metode objekta `GIcon`

- `image` ...lokacija slike marker-ja na strežniku, ki je v GIF ali PNG formatu
- `iconSize` ...velikost ikone marker-ja v pikslih, objekt je tipa `GSize`
- `iconAnchor` ...lokalne koordinate točke (glede na koordinatni sistem marker-ja, *(Slika 4.3.2.1*), kjer se marker pritrdi na karto. Vhodni parameter je objekt tipa `GPoint`
- `infoWindowAnchor` ...lokalne koordinate točke (glede na koordinatni sistem marker-ja *(Slika 4.3.2.1*), kjer se marker pritrdi na karto. Vhodni parameter je objekt tipa `GPoint.`
- `Shadow` ...lokacija slike marker-ja na strežniku, ki je v GIF ali PNG formatu
- `shadowSize` ...velikost slike sence v pikslih, objekt je tipa `GSize`
- `shadowAnchor` ...lokalne koordinate točke (glede na koordinatni sistem sence *(Slika 4.3.2.1*), kjer se marker pritrdi na karto. Vhodni parameter je objekt tipa `GPoint`.



Slika 4.3.2.1 *lokacija sidrišč marker-ja, informacijskega okna in sence marker-ja (vir: lasten vir)*

Primer definiranja objekta `GIcon` z vsemi metodami:

```
var icon = new GIcon();
    icon.image = 'images/accident.png'
    icon.iconSize = new GSize(66,84)
    icon.iconAnchor = new GPoint(33,84)
    icon.shadow = 'images/shadow.png'
    icon.shadowSize = new GSize(66,108)
    icon.shadoAnchor = new GPoint(33,84)
```

V našem primeru ustvarimo marker s funkcijo `createMarker()` prikažemo pa ga z objektom `markermanager`.

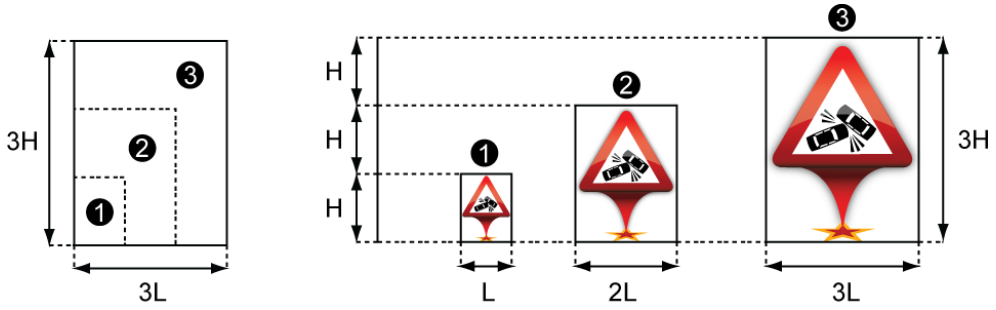
5 PREGLEDNOST KARTE

5.1 Opis

Prikazovanje večjega števila marker-jev je pri manjših povečavah karte nerazločno *(Slika 5.1.1)*, saj se ikone marker-jev prekrivajo. Da je situacija spet razločna, je potrebno karto približati, kar pa je le začasna rešitev.

***Slika 5.1.1** nepreglednost zaradi prevelikih marker-jev pri majhni povečavi karte (vir: lasten vir)*

Rešitev predstavlja več ikon različnih velikosti za različne stopnje povečav. Ko se od karte oddaljujemo se ikone pomanjšujejo, ko se ji približujemo pa povečujejo. Ikone so treh različnih velikosti. Njihove dimenzije so tako izbrane, da predstavljajo večkratnike dimenzij najmanjše ikone *(Slika 5.1.2)*. Ta sistem izberemo zato, da je bolj priročen za uporabo v zanki.

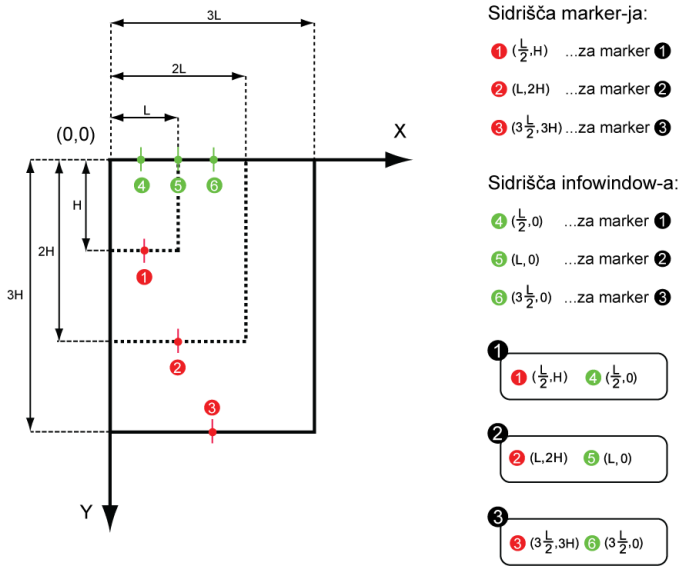


Slika 5.1.2 *dimenzije ikone marker-jev (vir: lasten vir)*

Velikost	**Najmanjša ikona**	**Srednja ikona**	**Največja ikona**
Ime ikone	`Icon0.jpg`	`Icon1.jpg`	`Icon2.jpg`
Dolžina L	22 pikslov	22 * 2 = 44 pikslov	22 * 3 = 66 pikslov
Višina L	28 pikslov	28 * 2 = 56 pikslov	28 * 3 = 84 pikslov

Preglednica 5.1.1: Pregled parametrov za tri veličine ikon

S spreminjanjem velikosti ikon se spreminja tudi položaj sidrišč informacijskega okna in informacijskega okna. Ker je položaj lokalnega koordinatnega sistema pri ikoni marker-ja, vedno na istem mestu (skrajni zgornji - levi piksel ikone) nam to omogoči, da sestavimo z uporabo zanke takšno kodo, ki bo »ulovila« sidrišča pri vseh velikostih marker-ja *(Slika 5.1.3)*.



Slika 5.1.3 *Položaj sidrišč pri vseh velikostih marker-ja (vir: lasten vir)*

Sedaj je potrebno določiti poseben algoritem, ki bo zajel točke sidrišč informacijskega okna, marker-ja in imena datotek za različne velikosti marker-ja. To najlažje storimo z uporabo `for` zanke za števec `i = 0,1,2` in objektom `markermanager`, ki ga pokličemo iz datoteke *markermanager.js.*

5.2 Objekt `markermanager`

Objekt `markermanager` oziroma njegova uradna (ampak zaenkrat manj stabilna verzija) `GmarkerManager` , služi za prikazovanje večjega števila marker-jev z upoštevanjem stopnje povečave.

Prvo ga je potrebno zagnati:

```
mgr = new MarkerManager(map, {maxZoom: 19})
```

Parametri:

- `map` ...ime spremenljivke karte na kateri deluje
- `maxZoom` ...predstavlja največjo povezavo pri kateri markermanager še deluje

5.2.1 Metoda `addMarkers`

`addMarkers(markers, minZoom, maxZoom)` ...uporabimo, če želimo dodati množico marker-jev.

Parametri:

- `markers` ...množica marker-jev. Vhodni parameter mora biti tipa `array`.
- `minZoom` ...spodnja meja povečave pri kateri se bojo prikazovali marker-ji.
- `maxZoom` ...zgornja meja povečave pri kateri se bojo prikazovali marker-ji.

5.2.2 Metoda `refresh`

`refresh()` ...če uporabljamo metodo `addMarkers()` je potrebno z `refresh` ukazom osvežiti stanje za vsako stopnjo povečave.

5.2.3 `For` zanka za tri različne velikosti marker-jev

Prva zanka za `j = 0,1,2` zajame ime ikone marker-ja, velikost markerja in položaj sidrišča informacijskega okna ter marker-ja.

```
var numbatches = 3

for (var j = 0; j < numBatches; j++) {
      icon[j] = new GIcon();
      icon[j].image = 'images/accident' + j + '.png';
      icon[j].iconSize = new GSize(l*(j+1), h*(j+1));
      icon[j].iconAnchor = new GPoint((h/2)*(j+1), h*(j+1));
      icon[j].infoWindowAnchor = new GPoint((h/2)*(j+1), h*(j+1));

      markerBatches[j] = [];
}
```

Druga zanka pa za `j = 0,1,2` potisne v prazno spremenljivko tipa `array` z imenom `markerBatches[j]` objekt tipa `GMarker`, ki je rezultat funkcije `createMarker()`.

- `markerBatches[0]` ...vsebuje `GMarker` z ikono `icon0.png`
- `markerBatches[1]` ...vsebuje `GMarker` z ikono `icon1.png`
- `markerBatches[2]` ...vsebuje `GMarker` z ikono `icon2.png`

`markermanager` tako za stopnje povečave 12,13,14 ustvari tri skupine markerjev različnih velikosti.

```
var numbatches = 3

for (var j = 0 ; j < numBatches ; j++){
      markerBatches[j].push(createMarker(point, icon[j], situVars));
      mgr.addMarkers(markerBatches[j], 12+j, 12+j);
}
```

Ker `markermanager` deluje samo znotraj mej povečave, ki smo mu jih določili (v našem primeru od 9 do 11), bo ob povečavah večjih od 11 ali manjših od 9 prenehal delovati. Posledica bo, da se marker-ji ne bojo izrisali. Težavo rešimo tako, da omejimo povečave iz prvotnih 19 stopenj na 3. To moramo storiti za vsak tip karte posebej:

a.) Za terensko karto:

```
G_PHYSICAL_MAP.getMinimumResolution = function () { return 9 };
G_PHYSICAL_MAP.getMaximumResolution = function () { return 11 };
```

b.) Normalno karto:

```
G_NORMAL_MAP.getMinimumResolution = function () { return 9 };
G_NORMAL_MAP.getMaximumResolution = function () { return 11 };
```

c.) Satelitsko karto:

```
G_SATELLITE_MAP.getMinimumResolution = function () { return 9 };
G_SATELLITE_MAP.getMaximumResolution = function () { return 11 };
```

d.) Hibridno karto:

```
G_HYBRID_MAP.getMinimumResolution = function () { return 9 };
G_HYBRID_MAP.getMaximumResolution = function () { return 11 };
```

***c** Majhne ikone ob majhni povečavi (vir: lasten vir)*

Slika 5.2.3.2 *Srednje in velike ikone ob srednji povečavi (vir: lasten vir)*

6 PODATKI O DOGODKIH NA CESTI

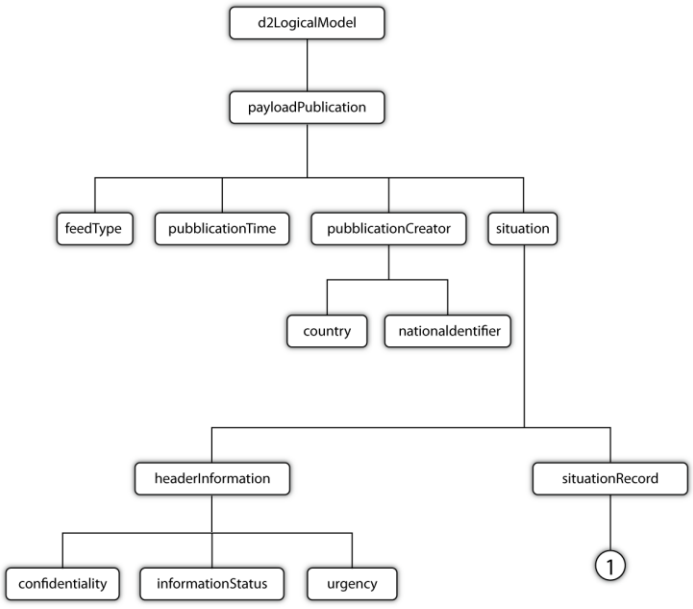
6.1 Splošno

V trenutku, ko se zabeleži nek dogodek na cesti (v našem primeru je to prometna nesreča), se na strežniku generira nova XML datoteka s ključnimi podatki. Datoteke imajo naključno generirana imena:

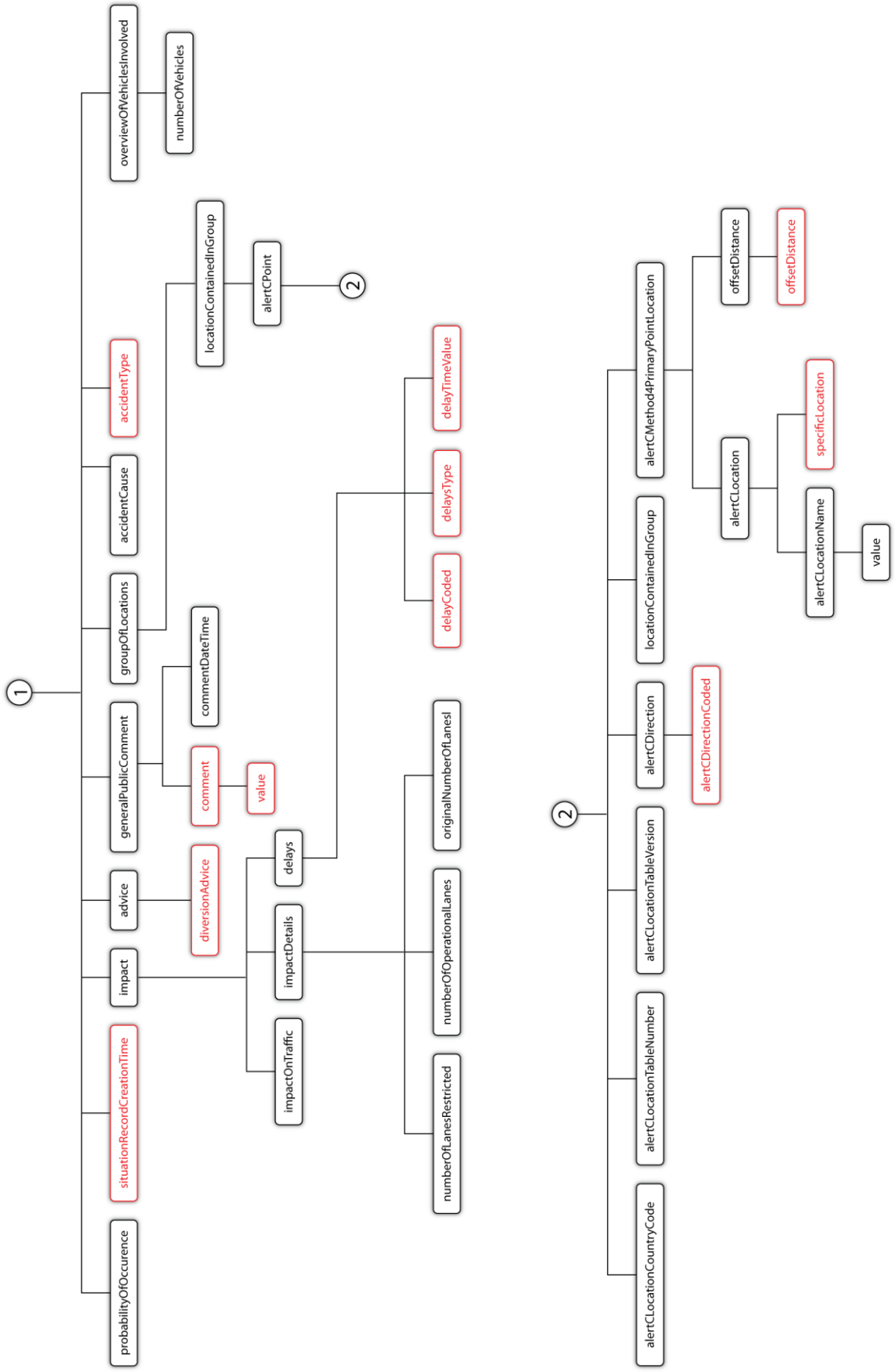
c4d8fce8-8b4a-4c83-be47-41e0a008b8ab.XML

Sestavljena so po XML shemi standarda DATEX II. V našem primeru smo se omejili na dogodke, ki so zapisani z načinom *AlertCMethod4Point*. To pomeni, da imamo opravka s točkovnim dogodkom. Lokacija je definirana s pomočjo točke iz predefinirane lokacijske tabele in linearnim odmikom od nje v smeri prometnega toka.

Struktura XML datoteke *c4d8fce8-8b4a-4c83-be47-41e0a008b8ab.XML* je zelo kompleksna. Za lažjo predstavo lahko pogledamo kako ležijo elementi v drevesni strukturi. Z rdečo barvo so označeni podatki, ki bi nas utegnili zanimati, pri objavi podatkov o nesreči (*Slika 18*).



***Slika 6.1.1** XML po shemi DATEX II za metodo AlertCMethod4Point (vir: lasten vir)*


1
probabilityOfOccurence
situationRecordCreationTime
impact
advice
generalPublicComment
groupOfLocations
accidentCause
accidentType
overviewOfVehiclesInvolved
numberOfVehicles
locationContainedInGroup
alertCPoint
2
commentDateTime
comment
value
diversionAdvice
impactDetails
impactOnTraffic
delays
delayCoded
delaysType
delayTimeValue
numberOfOperationalLanes
originalNumberOfLanes
numberOfLanesRestricted
2
alertCLocationCountryCode
alertCLocationTableNumber
alertCLocationTableVersion
alertCDirection
alertCDirectionCoded
locationContainedInGroup
alertCMethod4PrimaryPointLocation
alertCLocation
offsetDistance
offsetDistance
specificLocation
alertCLocationName
value

Dejanski izgled XML datoteke s podatki o nesreči:

```
<?xml version="1.0" encoding="utf-16"?>
<d2LogicalModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" modelBaseVersion="1.0"
 xmlns="http://datex2.eu/schema/1_0/1_0">
  <payloadPublication xsi:type="SituationPublication" lang="SL">
    <feedType>SituationPublication</feedType>
    <publicationTime>2008-04-08T15:18:15.719875+02:00</publicationTime>
    <publicationCreator>
      <country>si</country>
      <nationalIdentifier>SI</nationalIdentifier>
    </publicationCreator>
    <situation id="d4808f14-6340-4087-b189-42911fd23d6a">
      <headerInformation>
        <confidentiality>noRestriction</confidentiality>
        <informationStatus>real</informationStatus>
        <urgency>normalUrgency</urgency>
      </headerInformation>
      <situationRecord xsi:type="Accident" id="b969b9bb-4dd9-40f7-9127-7eb48c6bb3d7">
        <situationRecordCreationTime>2008-04-08T15:18:15.719875+02:00</situationRecordCreationTime>
        <probabilityOfOccurrence>certain</probabilityOfOccurrence>
        <impact>
          <impactOnTraffic>impossible</impactOnTraffic>
          <impactDetails>
            <numberOfLanesRestricted>1</numberOfLanesRestricted>
            <numberOfOperationalLanes>2</numberOfOperationalLanes>
            <originalNumberOfLanes>3</originalNumberOfLanes>
          </impactDetails>
          <delays>
            <delayCoded>delayBetweenOneHourAndThreeHours</delayCoded>
            <delaysType>delays</delaysType>
            <delayTimeValue>0</delayTimeValue>
          </delays>
        </impact>
        <advice xsi:type="Diversion">
          <diversionAdvice>followSigns</diversionAdvice>
        </advice>
        <generalPublicComment>
          <comment>
            <value lang="SL" />
          </comment>
          <commentDateTime>2008-04-08T15:18:15.719875+02:00</commentDateTime>
        </generalPublicComment>
        <groupOfLocations>
          <locationContainedInGroup xsi:type="Point">
            <alertCPoint xsi:type="AlertCMethod4Point">
              <alertCLocationCountryCode>SI</alertCLocationCountryCode>
              <alertCLocationTableNumber>1</alertCLocationTableNumber>
              <alertCLocationTableVersion>1</alertCLocationTableVersion>
              <alertCDirection>
                <alertCDirectionCoded>both</alertCDirectionCoded>
              </alertCDirection>
              <alertCMethod4PrimaryPointLocation>
                <alertCLocation>
                  <alertCLocationName>
                    <value lang="SI">MPP ŠKOFIJE</value>
                  </alertCLocationName>
                  <specificLocation>1007</specificLocation>
                </alertCLocation>
                <offsetDistance>
                  <offsetDistance>333</offsetDistance>
                </offsetDistance>
              </alertCMethod4PrimaryPointLocation>
            </alertCPoint>
          </locationContainedInGroup>
        </groupOfLocations>
        <accidentCause>unknown</accidentCause>
        <accidentType>accident</accidentType>
        <overviewOfVehiclesInvolved>
          <numberOfVehicles>0</numberOfVehicles>
        </overviewOfVehiclesInvolved>
      </situationRecord>
    </situation>
  </payloadPublication>
</d2LogicalModel>
```

Slika 6.1.2 *Vsebina datoteke XML s podatki o prometni nesreči (vir: lasten vir)*

Od množice podatkov, ki jih pridobimo iz XML datoteke, je samo par pomembnih za nadaljnje delo. Te delimo na podatke o lokaciji dogodka in podatke , ki opišejo značilnosti spremenjenega cestnega režima (relevantno za voznike). Krepko odtisnjeni podatki so obvezne narave in se vsakič pojavijo pri publikaciji dogodka. Ostali niso obvezne narave.

Podatki o lokaciji:

- **število LCD lokacijske točke... `specificLocation`**
- **zamik od LCD točke (»offset«)... `offsetDistance`**
- **smer zamika... `alertCDirectionCoded`**

Podatki o dogodku (prometni nesreči):

- **čas nastanka dogodka ...`situationRecordCreationTime`**
- **narava dogodka... `accidentType`**
- **navodila za obvoz ... `diversionAdvice`**
- komentar k dogodku... `comment`
- kodirani časovni zaostanek, glede na normalne prometne razmere... `delayCoded`
- časovni zaostanek, glede na normalne prometne razmere... `delayTimeValue`

6.2 Uporaba PHP skripte *transformator.php*

Problem nastane, ko se generira preveč dogodkov sočasno. V tem primeru nastane več XML datotek, ki jih je potrebno prikazati na karti. Te imajo bistveno preveč nepotrebnih podatkov in tako zasedejo preveč prostora na strežniku, da bi jih lahko hitro nalagali preko obstoječih internetnih povezav. V primeru, da bi aplikacijo uporabljalo več uporabnikov hkrati, bi tako še dodatno preobremenili promet na strežniku.

Za lažje manipuliranje večje količine dogodkov na cesti uporabimo posebno PHP skripto z imenom *transformator.php*, ki je naložena na strežniku. Njena naloga je da v določenem časovnem intervalu (ki ga lahko poljubno nastavimo - privzeta vrednost je 5 minut) prebere celotno vsebino direktorija .../*xmlInput/*, ki vsebuje dogodkovne XML datoteke in iz njih izbere ključne podatke o posameznem dogodku. S temi podatki nato sestavi

novo XML datoteko *dogodki.xml* v direktoriju .../*xmlOutput/*. Takšna datoteka tako vsebuje ključne podatke o nizu prometnih nesreč.

Na *Sliki 6.2.1* imamo primer datoteke *dogodki.xml* za dve prometni nesreči. Kakor opazimo je podatkov bistveno manj kot prej:

```
<datexModel>
  dogodek 1
  <situation>
      <situationRecord type="Accident" id="b969b9bb-4dd9-40f7-9127-7eb48c6bb3d7"></situationRecord>
      <situationRecordCreationTime>2008-04-08T15:18:15.719875+02:00</situationRecordCreationTime>
      <delayCoded>delayBetweenOneHourAndThreeHours</delayCoded>
      <delayTimeValue>0</delayTimeValue>
      <numberOfLanesRestricted>1</numberOfLanesRestricted>
      <numberOfOperationalLanes>2</numberOfOperationalLanes>
      <diversionAdvice>followSigns</diversionAdvice>
      <specificLocation>1036</specificLocation>
      <offsetDirection>negative</offsetDirection>
      <offsetDistance>200</offsetDistance>
      <comment><value lang="SL"/>Komentar!</comment>
  </situation>
  dogodek 2
  <situation>
      <situationRecord type="Accident" id="b969b9bb-4dd9-40f7-9127-7eb48c6bb3d7"></situationRecord>
      <situationRecordCreationTime>2008-04-08T15:18:15.719875+02:00</situationRecordCreationTime>
      <delayCoded>delayBetweenOneHourAndThreeHours</delayCoded>
      <delayTimeValue>0</delayTimeValue>
      <numberOfLanesRestricted>1</numberOfLanesRestricted>
      <numberOfOperationalLanes>2</numberOfOperationalLanes>
      <diversionAdvice>followSigns</diversionAdvice>
      <specificLocation>1035</specificLocation>
      <offsetDirection>positive</offsetDirection>
      <offsetDistance>500</offsetDistance>
      <comment><value lang="SL"/>Komentar!</comment>
  </situation>
</datexModel>
```

***Slika 6.2.1** vsebina datoteke dogodki.xml za primer dveh prometnih nesreč (vir: lasten vir)*

V primeru, da bi želeli vključiti druge podatke ali če želimo spremeniti lokacije in imena datotek in direktorij XML datotek, to lahko storimo v datoteki *trans_conf.xml*.

6.3 Datoteka *trans_conf.xml*

V datoteki *trans_conf.xml* lahko spreminjamo katere podatke naj skripta *transformator.php* izvleče iz XML datotek o nesreči.

Določanje nastavitev:

```
<settings>
```

```
        <output_encoding>UTF-8</output_encoding>
        <input_directory>xmlInput/</input_directory>
        <input_file_pattern>/.xml/</input_file_pattern>
        <output_directory>xmlOutput/</output_directory>
        <output_filename>dogodki.xml</output_filename>
        <root_tagname>datexModel</root_tagname>
    </settings>
```

- `output_encoding` ...določa jezikovno kodno tabelo (UTF-8 za Slovenščino)
- `input_directory` ...ime direktorija z vhodnimi XML datotekami
- `input_file_pattern` ...določa tip vhodne datoteke – v našem primeru XML
- `output_directory` ...ime direktorija z izhodno XML datoteko
- `output_filename` ...ime izhodne XML datoteke – v našem primeru *dogodki.xml*
- `root_tagname` ...ime glavnega elementa

Določimo lahko tudi katere podatke hočemo izbrati iz XML datoteke o nesrečah:

```
<xml_transforms type="situation">
    <xml_transform>
        <input_xpath> ... </input_xpath>
        <output_xpath> ... </output_xpath>
    </xml_transform>
</xml_transforms>
```

`input_xpath` določa pot do podatka npr.:

```
d2LogicalModel/payloadPublication/situation/situationRecord/
advice/diversionAdvice
```

`output_xpath` določa ime elementa v *dogodki.xml* datoteki npr.:

```
diversionAdvice
```

Rezultat našega primera za dva dogodka bi v datoteki *dogodki.xml* izgledal takole:

```
<datexModel>
    <situation>
```

```
            <diversionAdvice> followSigns </diversionAdvice>
      </situation>
      <situation>
            <diversionAdvice> followSigns </diversionAdvice>
      </situation>
</datexModel>
```

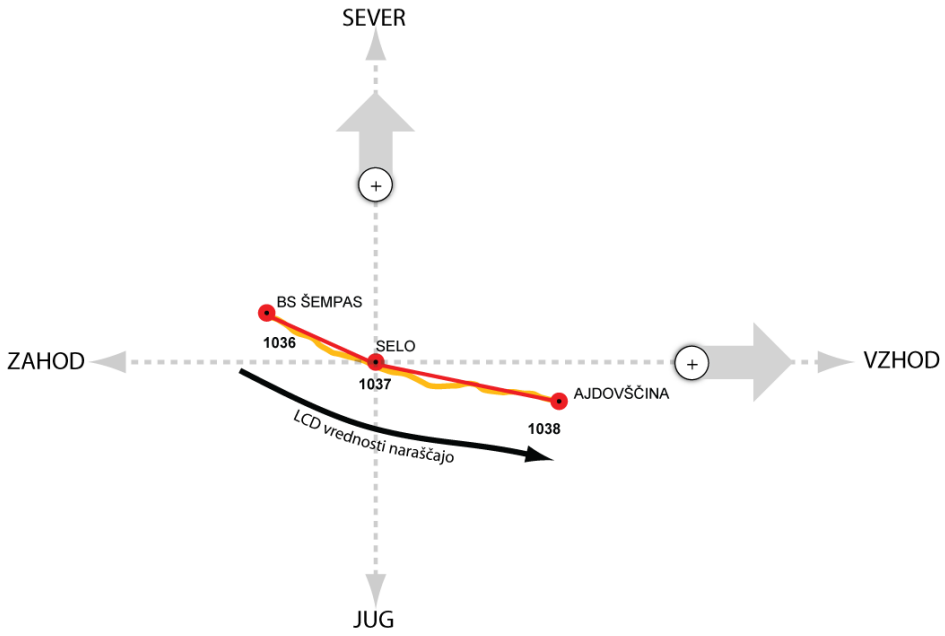
7 LOKACIJSKE TOČKE

7.1 Definicija

Lokacijske točke predstavljajo fizične objekte na cesti (bencinske črpalke, policijske postaje, priključne ramp, odcepe, križišča), glede na katere se orientiramo pri določevanju položaja dogodkov na cesti po standardu DATEX II.

7.2 Enumeracija lokacijskih točk

Standard DATEX II predlaga (ni pa obvezno), da uredimo enumeracijo točk na takšen način, da vrednost LCD točk narašča v smeri od juga proti severu in od zahoda proti vzhodu. Lokacijske točke in njihovo enumeracijo je za potrebe naše diplome določilo podjetje Traffic design d.o.o. *(Slika 7.2.1).*

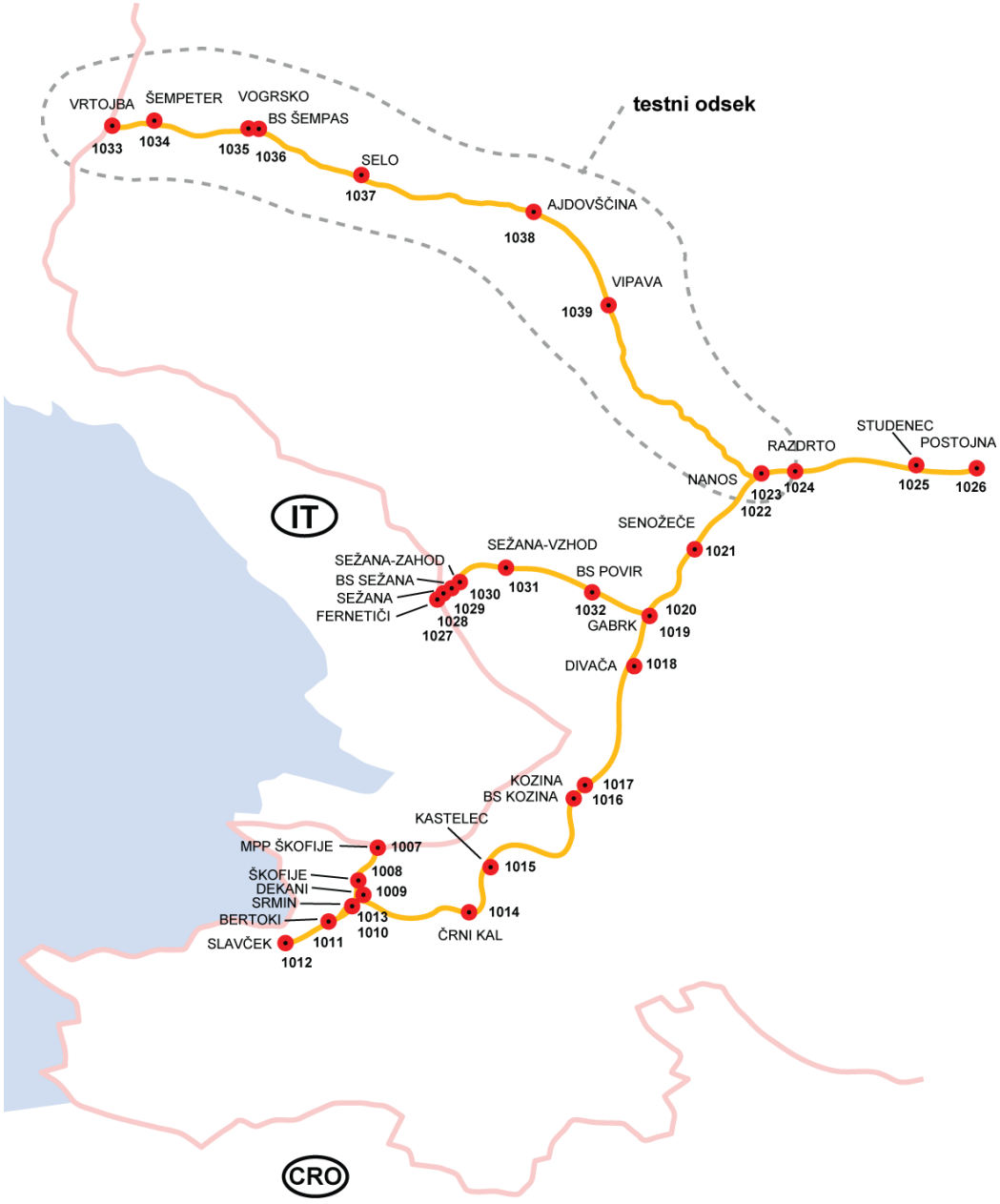


Slika 7.2.1 *Smer kodiranje ceste po predlogu standarda DATEX II (vir: lasten vir)*

Pupovac, P. 2008. Spletna aplikacija za izmenjavo in prikaz prometnih podatkov
Diplomska naloga – UNI. Ljubljana, UL, FGG, Oddelek za gradbeništvo, Prometna smer.

7.3 Trenutno stanje

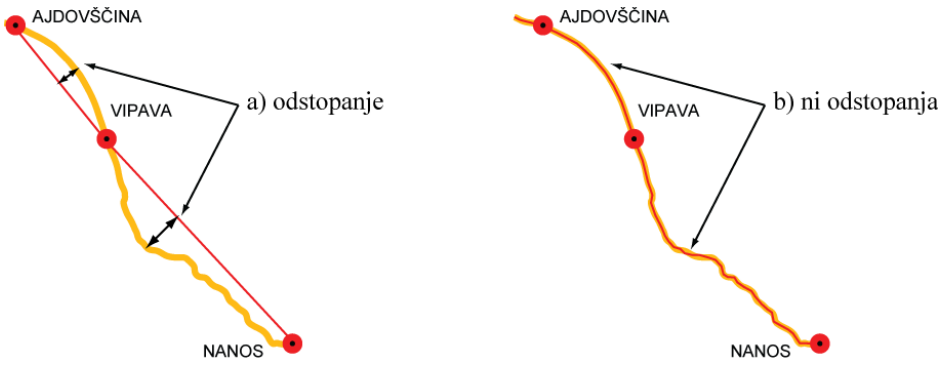
Dokončna ureditev lokacijskih točk za RS zaenkrat še ni določena. Podjetje Traffic design d.o.o. je kot prvo podalo predlog za postavitev lokacijskih točk *(Slika 7.3.1).*



Slika 7.3.1 *Lokacijske točke za Primorsko regijo po predlogu podjetja Traffic Design d.o.o. (vir: lasten vir)*

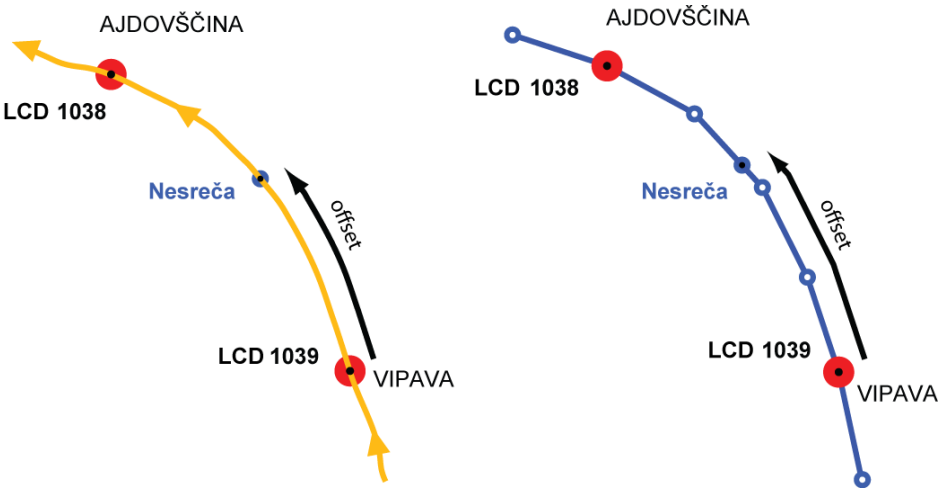
7.4 Zgostitev obstoječih lokacijskih točk

Obstoječe lokacijske točke so za potrebe prikazovanja preko Google Maps postavljene na prevelikih medsebojnih razdaljah, da bi jih lahko direktno uporabili za projiciranje situacije na karto. Njihova gostota je zaenkrat takšna da ne ujame pravilno poteka ceste *(Slika 7.4.1)*.



Slika 7.4.1 *Odstopanja od linije ceste pri aktualni gostoti lokacijskih točk a) in zgostitvi b) (vir: lasten vir)*

Kot rešitev zgostimo obstoječi cestni odsek z dodatnim poligonom na katerega bomo izrisovali dogodke *(Slika 7.4.1)*. Dodatne lokacijske točke niso enakomerno postavljene (recimo na 50 m) ampak so tako postavljene, da čimbolj prilegajo osi ceste.

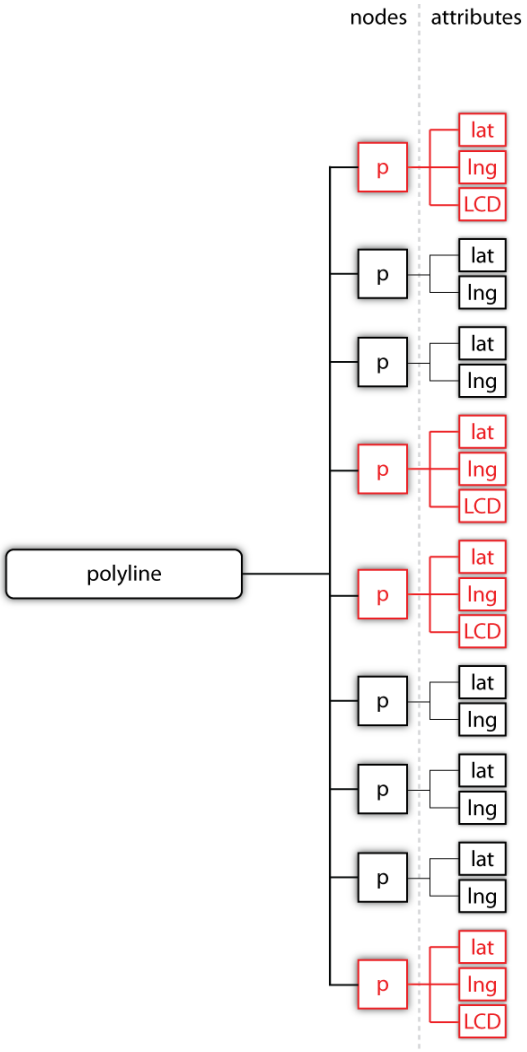


Slika 7.4.1 *Zgostitev obstoječih lokacijskih točk (vir: lasten vir)*

Podatke o poligonu shranimo v datoteko XML z imenom *polyline.xml.*

Sturktura datoteke *polyline.xml*

- osnovno vozlišče *(»root node«)* `<polyline>`,
- lokacijske točke so definirane kot podelementi *(»childnodes«)* `<p>`,
- lokacijske točke vsebujejo atribute *(»attributes)* kot so latituda (`lat`), longituda (`lng`) in LCD (`LCD`), medtem ko so dodatne točke označene samo z latitudo in longitudo *(Slika 7.4.2)*



***Slika 7.4.2** Struktura datoteke polyline.xml (vir: lasten vir)*

7.5 Funckija `loadPOLY()`

Lokacijske točke naložimo v funkcijo `loadPOLY()`, s pomočjo Google Maps API objekta `GXmlHttp`, ki z metodo `create()` ustvari `XMLHttpRequest` Javascript objekt. Komunikacija s strežnikom preko tega objekta predstavlja pri t.i. AJAX (Asynchronous JavaScript And XML) metodi vspostaljanja povezave s strežnikom. Tip povezave za primer datoteke *polyline.xml* je sihnroni, kar pomeni, da program ne bo poslušal strežnik za morebitne spremembe v datoteki, saj je ta nespremenljivega značaja. Asinhrohni tip povezave bomo kvečjemu uporabili pri dogodkih na cesti, saj se taka datoteka ob vsak novemu dogodku spremeni.

7.6 Objekt `XMLHttpRequest`

Če hočemo pridobiti ali poslati podatke na strežnik z uporabo tradicionalnega Javascript programiranja, potem moramo ustvariti HTML obrazec (»form«) in uporabiti GET / POST metodo prejemanja / pošiljanja podatkov. Uporabnik bo moral pritisniti »Pošlji« gumb za prejemanje / pošiljanje podatkov, počakati na odgovor strežnik in nato se bo naložila nova stran z rezultati iskanja.

Ker strežnik ustvari novo stran vsakič, ko uporabnik pošlje zahtevo, pride do tega, da se aplikacije zaganjajo počasneje in so tako uporabniško manj prijazne.

Z AJAX metodo program napisan v Javascripte komunicira direktno s strežnikom preko `XMLHttpRequest` objekta. Na takšen način pridobimo odgovor s strežnika, ne da bi se spletna stran ponovno nalagala. Uporabnik bo ostal na isti spletni strani in ne bo opazil, da je skripta imela kakršnokoli interakcijo s strežnikom.

7.6.1 Lastnost `onreadystatechange`

Ko je poslana zahteva na strežnik, potrebujemo funkcijo, ki bo podatke sprejela. To storimo z uporabo lastnosti `onreadystatechange`, ki bo shranila tako funkcijo, s katero bomo procesirali odgovor s strežnika.

7.6.2 Lastnost `readyState`

Sedaj je potrebno še definirati lastnost `readyState`, ki bo shranila status odgovora s strežnika. Vsakič, ko se readyState spremeni, bo `onreadystatechange` zagnala novo funkcijo.

Tukaj imamo spisek vrednosti, ki jih lahko vrne `readyState` lastnost:

Stanje	Opis
0	Zahteva je inicializirana
1	Zahteva je bila nastavljena
2	Zahteva je bila poslana
3	Zahteva se procesira
4	Zahteva je končana

Preglednica 7.6.2.1 *vrednosti lastnosti* `readyState`

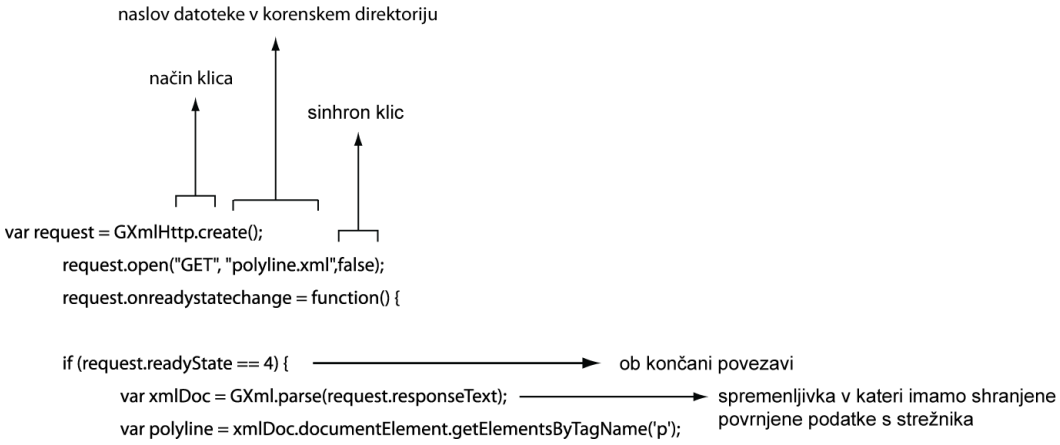
7.6.3 Lastnost `responseText`

Podatke, ki nam jih vrnil strežnik lahko pridobimo z uporabo lastnosti `responseText`. V našem primeru se bojo vsi podatki, ki jih bomo potrebovali za nadaljnjo uporabo shranili v spremenljivko z imenom `xmlDoc`.

Za poslati zahtevo na strežnik uporabljamo metodi `open()` in `send()`. Prvo metodo uporabljamo, ko pošiljamo zahtevo na GET ali POST način. Drugi argument določa URL naslov skripte, ki se nahaja na strežniku (*»server-side script«*). Tretji argument nam pove ali bo zahteva poslana sinhrono ali asinhrono. Metoda `send()` polje zahtevo na strežnik.

7.6.4 Primer

Povezavo shranimo kot spremenljivko `request`:



***Slika 7.6.4.1** Klic datoteke polyline.xml z objektom `GXmlHttp` (vir: lasten vir)*

Za nadaljno obdelavo podatkov, moramo s `for` zanko za vrednosti `i = 0, 1, 2, ...,polyline.length` prebrati spremenljivko `polyline` po vozliščih `p` (`polyline.length` predstavlja število elementov `p` v spremenljivki `xmlDoc`). Vsak element vsebuje atribute `lat` in `lng`, lokacijske točke pa vsebujejo tudi atribut `LCD`.

```
for (var i = 0; i < polyline.length ; i++) {

        var LCD = parseFloat(polyline[i].getAttribute('LCD'));

           LCDarray.push(LCD);

        var aXysLat = parseFloat(polyline[i].getAttribute('lat'));
        var aXysLng = parseFloat(polyline[i].getAttribute('lng'));
        var vertex = new GLatLng(aXysLat,aXysLng);

        aXys.push(vertex);
}
```

***Slika 7.6.4.2** Iteriranje po spremenljivki `polyline` (vir: lasten vir)*

Za vsak i smo torej ustvarili sledeče spremenljivke:

- `LCD` ...števila LCD za lokacijske točke. Tam kjer ni lokacijske točke je izpisana vrednost `NaN` (*»Not a Number«* – ni število)
- `aXysLat` ...latituda posamezne točke
- `aXysLng` ...longituda posamezne točke
- `vertex` ...`GLatLng` objekt sestavljen iz spremenljivk `aXysLat` in `aXysLng`

Spremenljivko LCD z metodo `push()`, potisnemo za vsako vrednost i, na konec objekta tipa `Array()` z imenom `LCDarray`. Podobno storimo s spremenljivko vertex, ki jo shranimo v `Array()` spremenljivko z imenom `aXys`.

Podatki so sedaj pripravljeni, potrebno je le še zapreti povezavo z ukazom `request.send(null)`.

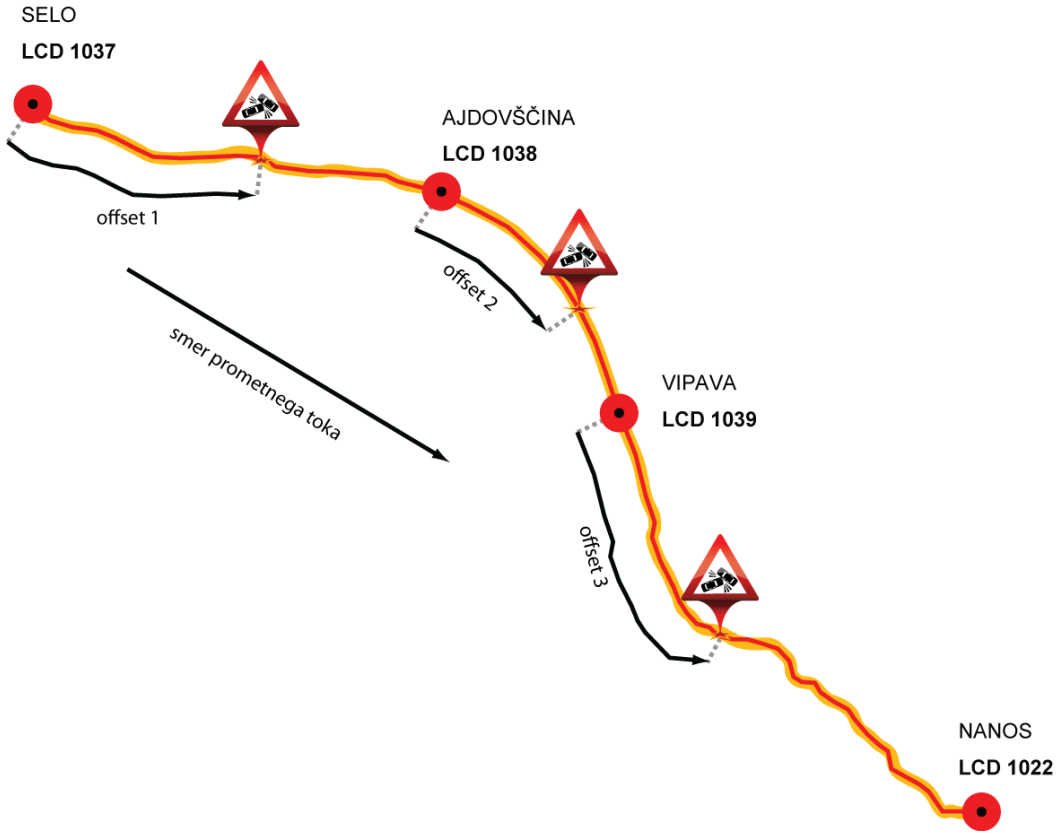
Pupovac, P. 2008. Spletna aplikacija za izmenjavo in prikaz prometnih podatkov
Diplomska naloga – UNI. Ljubljana, UL, FGG, Oddelek za gradbeništvo, Prometna smer.

8 DOLOČITEV LOKACIJE TOČKE

8.1 Metoda *AlertCMethod4Point*

Za namen diplome se bomo omejili na prikaz točkovnih dogodkov po metodi *AlertCMethod4Point*, ki jo določa standard DATEX II. Po definiciji ta metoda določa položaj dogodka na cestni mreži s pomočjo reference glede na točko v predefinirani Alert C lokacijski tabeli in odmika (*»offset«*), ki ima smer prometnega toka (*Slika 8.1.1*).

Podatek, `alertCDirectionCoded`, nam še pove v kateri smeri poteka prometni tok na katerega se dogodek navezuje. Zajema lahko vrednosti: *positive* (v smeri naraščanja LCD vrednosti točk), *negative* (v smeri padanja LCD vrednosti točk) in *both* (dogodek zajema obe smeri) ter *unknown* (ni podatka o smeri).



Slika 8.1.1 *Določitev lokacije nesreče s pomočjo metode AlertCMethod4Point (vir: lasten vir)*

8.2 Podatki potrebni za določitev lokacije dogodka

Podatke o lokaciji dogodka dobimo preko funkcije `loadSITUATION()`, ki nam naloži XML datoteko *dogodki.xml*. Ta vsebuje podatke za prikaz dogodka, ki so bili generirani preko metode *AlertCMethod4Point*, medtem ko so predefinirane Alert C lokacijske točke skupaj z zgoščeno mrežo shranjene v datoteki *polyline.xml*, ki jo naložimo v funkciji `loadPOLYLINE()`.

Kot primer vzemimo prometno nesrečo, ki se je zgodila 2 kilometra od Ajdovščine v smeri Vipave. V tem primeru je so podatki sledeči:

- Podatek `specificLocation` je enak `1038`
- Podatek `offsetDistance` je enak `2000`
- Podatek `alertCDirectionCoded` je enaka `positive`.

8.3 Funkcija `loadSITUATION()`

Podobno kot smo priklicali datoteko *polyline.xml* v prejšnjem poglavju, sedaj to storimo z datoteko *dogodki.xml*. Razlika je le v tem, da bomo sedaj povezavo odprli na asinhroni način, saj se datoteka spremeni vsakič, ko nastopi nov dogodek. V tem primeru bo prednost metode AJAX prišla do izraza.

Slika 8.3.1 *Asinhrono pošiljanje zahteve na strežnik (vir: lasten vir)*

Za nadaljnjo obdelavo podatkov, moramo s `for` zanko za vrednosti `i = 0, 1, 2, ...,datexModel.length` prebrati spremenljivko `datexModel` po vozliščih `situation` (`datexModel.length` predstavlja število elementov `situation` v spremenljivki `xmlDoc`).

Vsak element vsebuje množico podatkov o dogodku na cesti, ki jih shranimo v spremenljivko `situVars`. Ta je tipa asociativni `Array()`.

8.4 Funkcija `truncatePoly()`

Za uporabo funkcije `truncatePoly()` moramo imeti dve vnaprej pripravljeni spremenljivki tipa `Array`, ki nam jih ustvari funkcija `loadSITUATION()`. To sta spremenljivki `aXys` in `LCDarray`. V primeru, da je `alertCDirectionCoded`, negativen moramo obema spremenljivkama obrniti vrstni podatkov z ukazom `aXys.reverse()` in `LCDarray.reverse()`. V tem primeru bo vrednost pod ključem 0 ustrezala LCD točki 1023.

Vrednost `aXys` in `LCDarray`:

aXys		
ključ	**LCD**	**Vrednost**
0	1033	(45.9193840386111, 13.6215025)
1	/	(45.92106501, 13.64089966)
2	1034	(45.921065325, 13.6500694902778)
3	/	(45.91939316, 13.66355896)
4	/	(45.91700472, 13.67145538)
5	/	(45.91509389, 13.67832184)
6	/	(45.91676587, 13.6882782)
7	/	(45.91819895, 13.69411469)
8	1035	(45.9181799286111, 13.7111952972222)
9	1036	(45.9174327938889, 13.7207911)
10	/	(45.91485503, 13.72913361)
11	/	(45.91246639, 13.73394012)
12	/	(45.91031653, 13.7411499)
13	/	(45.90840547, 13.74664307)
14	/	(45.90362753, 13.75556946)
15	/	(45.90147733, 13.7607193)
16	/	(45.90004381, 13.77170563)
17	/	(45.89837133, 13.77925873)
18	1037	(45.8955192972222, 13.7882653972222)
19	/	(45.89359253, 13.79505157)
20	/	(45.89311463, 13.80191803)

LCDarray
vrednost
1033
NaN
1034
NaN
NaN
NaN
NaN
NaN
1035
1036
NaN
NaN
NaN
NaN
NaN
NaN
NaN
NaN
1037
NaN
NaN

21	/	(45.89120297, 13.81118774)	NaN
22	/	(45.88737947, 13.82045746)	NaN
23	/	(45.88642356, 13.82972717)	NaN
24	/	(45.88737947, 13.83590698)	NaN
25	/	(45.88857435, 13.8451767)	NaN
26	/	(45.88666254, 13.85444641)	NaN
27	/	(45.88475066, 13.86165619)	NaN
28	/	(45.88498965, 13.87504578)	NaN
29	1038	(45.8800145, 13.9022946194444)	1038
30	/	(45.88044871, 13.90079498)	NaN
31	/	(45.87758055, 13.91143799)	NaN
32	/	(45.87351708, 13.92208099)	NaN
33	/	(45.86825802, 13.93066406)	NaN
34	/	(45.85965122, 13.93993378)	NaN
35	/	(45.8536735, 13.94508362)	NaN
36	1039	(45.8355660597222, 13.9543751777778)	1039
37	/	(45.83908514, 13.95298004)	NaN
38	/	(45.83501886, 13.95469666)	NaN
39	/	(45.82832079, 13.95847321)	NaN
40	/	(45.82329672, 13.96087646)	NaN
41	/	(45.81827219, 13.96224976)	NaN
42	/	(45.81396508, 13.96327972)	NaN
43	/	(45.81085419, 13.96379471)	NaN
44	/	(45.80582854, 13.96585464)	NaN
45	/	(45.80355488, 13.9682579)	NaN
46	/	(45.80152048, 13.96945953)	NaN
47	/	(45.79804985, 13.97117615)	NaN
48	/	(45.79673335, 13.97340775)	NaN
49	/	(45.79517744, 13.97478104)	NaN
50	/	(45.79481838, 13.97872925)	NaN
51	/	(45.79386087, 13.98130417)	NaN
52	/	(45.79242458, 13.98405075)	NaN
53	/	(45.79194581, 13.98714066)	NaN
54	/	(45.7920655, 13.98954391)	NaN
55	/	(45.79158673, 13.99074554)	NaN
56	/	(45.79038977, 13.99177551)	NaN
57	/	(45.79050947, 13.99332047)	NaN
58	/	(45.79086856, 13.99520874)	NaN
59	/	(45.79015038, 13.99641037)	NaN
60	/	(45.78919279, 13.99726868)	NaN

61	/	(45.78787608, 13.99915695)	NaN
62	/	(45.78763667, 14.00018692)	NaN
63	/	(45.78727756, 14.00138855)	NaN
64	/	(45.78548198, 14.00190353)	NaN
65	/	(45.78488344, 14.0029335)	NaN
66	/	(45.78392576, 14.00344849)	NaN
67	/	(45.7833272, 14.00533676)	NaN
68	/	(45.78356663, 14.0073967)	NaN
69	/	(45.7823695, 14.00859833)	NaN
70	/	(45.78165121, 14.01082993)	NaN
71	/	(45.78057376, 14.01185989)	NaN
72	/	(45.77973572, 14.01323318)	NaN
73	/	(45.7790174, 14.0149498)	NaN
74	/	(45.77889768, 14.01666641)	NaN
75	/	(45.77829907, 14.01769638)	NaN
76	/	(45.77734127, 14.01906967)	NaN
77	/	(45.77602428, 14.02044296)	NaN
78	/	(45.77458753, 14.02147293)	NaN
79	/	(45.77350994, 14.02233124)	NaN
80	/	(45.77219286, 14.02456284)	NaN
81	/	(45.77075601, 14.0278244)	NaN
82	/	(45.76991783, 14.02902603)	NaN
83	/	(45.76716372, 14.03125763)	NaN
84	/	(45.76584649, 14.03125763)	NaN
85	/	(45.76381071, 14.03863907)	NaN
86	/	(45.7616551, 14.04224396)	NaN
87	/	(45.75806222, 14.04584885)	NaN
88	/	(45.75710408, 14.0496254)	NaN
89	/	(45.75770292, 14.05374527)	NaN
90	1023	(45.7573850166667, 14.0540697055556)	1023

Preglednica 8.4.1 *Vrednosti spremenljivk* `LCDarray` *in* `aXys`

Ključ predstavlja pozicijo podatkov v spremenljivki in je enak za obe spremenljivki, saj sta enako dolge. Z rdečo barvo so označene LCD točke, medtem ko so črno barvo označene pomožne točke. Vrednost `NaN` predstavlja *»Not a Number«* (ni število).

8.5 Funkcija `truncatePoly ()`

Namen funkcije, da ustvari novo spremenljivko z imenom `smallPoly` tipa `Array()`, ki predstavlja niz točk oblike `GLatLng`. Njena dolžina predstavlja odsek (interval) med dvema lokacijskima točkama, med katerima se je zgodila prometna nesreča.

Funkcija prebere vhodni podatek `situVars`, ki vsebuje podatke o nesreči. Pri ključu "`LCD point A`" prebere vrednost lokacijske točke na katero se dogodek navezuje in jo zapiše v spremenljivko `start`. V našem primeru je to `1038`. To vrednost nato uporabi, da z ukazom `LCDarray.find(start)` preišče spremenljivko `LCDarray` in na mestu, kjer zapisana vrednost `1038` zapiše zaporedno število ključa, t.j. `29`. Ukaz `find()` ni sestavni del nabora ukazov v Javascriptu zato ga definiramo v datoteki *prototype.js.* Naslednjo lokacijsko točko, določimo tako da poiščemo ključ v spremenljivki `LCDarray`, ki pripada LCD točki, ki je za vrednost ena večja od začetne. Pri nas je to `1038+1=1039`, kar ustreza ključu `36`. V primeru negativnega odmika imamo opravka z spremenljivkami z obrnjenim vrstnim redom podatkov, zato moramo vrednostim kvečjemu odšteti 1, torej: `1038-1=1037`

Za primer nesreče, ki se je zgodila 2 kilometra od Ajdovščine v smeri proti Vipavi, imamo interval v spremenljivki `aXys` s ključi v mejah od 29 do 36. Sedaj uporabimo metodo `slice()`, ki pri teh mejnih vrednostih izreže `aXys` in podatke, ki ležijo znotraj intervala shrani v spremenljivki `smallPoly`. Slednjo lahko končno uporabimo za delo z objektom `GPolyline`.

Primer uporabe metode `slice`():

```
smallPoly = aXys.slice (29,36)
```

smallPoly
(45.8800145, 13.9022946194444)
(45.88044871, 13.90079498)
(45.87758055, 13.91143799)
(45.87351708, 13.92208099)
(45.86825802, 13.93066406)
(45.85965122, 13.93993378)

(45.8536735, 13.94508362)
(45.8355660597222, 13.9543751777778)

Preglednica 8.5.1: *Vrednosti* `smallPoly` *spremenljivke:*

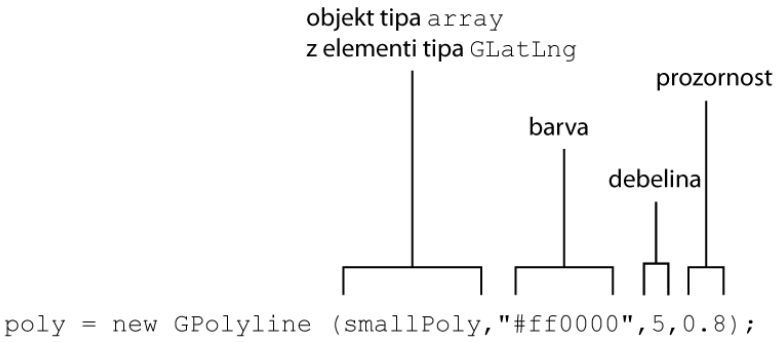
Slika 8.5.1 *Generiranje polilinije smallPoly (vir: lasten vir)*

Problem nastopi pri LCD vrednostih v točkah, kjer se cesta priključi na cesto višje kategorije. V tem primeru ima zadnja lokacijska točka v nizu drugačno vrednost, saj ta ustreza enumeraciji ceste višje kategorije na katero se priključuje. Na odseku Vrtojba – Nanos lokacijske točke naraščajo od 1033 do 1039 vse do Nanosa, temu pa pripada vrednost 1023, kar s prejšnjo metodo povzroči težave. V tem primeru postavimo pogoj, ki preveri, če spremenljivka `LCDarray` vsebuje vrednost `LCD+1`, ki jo shranimo v spremenljivko `nextLCD`. Sedaj jo je potrebno le še preveriti. Uporabiti moramo funkcijo `isNaN()`, ki testira, ali spremenljivka ni enaka objektu tipa `Number()`. V našem primeru vrednost `1040` ne obstaja, tako funkcija `isNaN(nextLCD)` vrne vrednost `true`. Program je ugotovil, da se nesreča nahaja med lokacijskima točkama `1039` in `1023`. Za konec intervala bomo tako izbrali največji možni ključ v spremenljivki `aXys`, ki predstavlja konec odseka lokacijsko točko Nanosa. Določimo ga na podlagi dolžine spremenljivke `aXys` z ukazom `aXys.length`, kar povrne vrednost `90`.

Pupovac, P. 2008. Spletna aplikacija za izmenjavo in prikaz prometnih podatkov
Diplomska naloga – UNI. Ljubljana, UL, FGG, Oddelek za gradbeništvo, Prometna smer.

8.6 Objekt `GPolyline()` in funkcija `createMarker()`

Za praktično in natančno prikazovanje dogodkov na cesti bomo te izrisovali preko objekta `GPolyline`, ki predstavlja objekt polilinije v Google Maps API naboru ukazov.



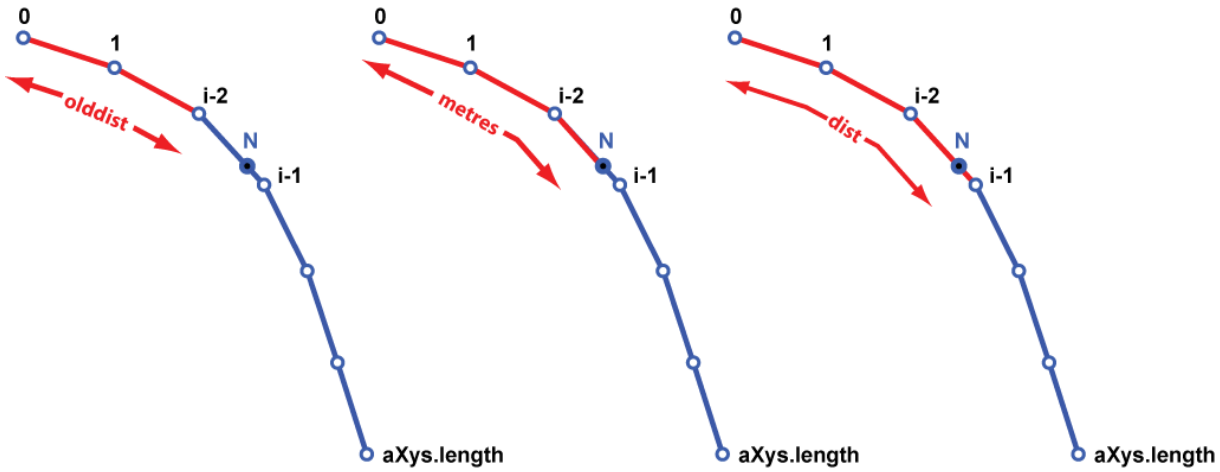
***Slika 8.6.1** Objekt GPolyline (vir: lasten vir)*

Za naše potrebe je pomemben samo vhodni podatek `smallPoly`, ki predstavlja Javascript objekt tipa `Array()` z elementi tipa `GLatLng`. Ti predstavljajo vozlišča polilinije in so linearno povezani. Objekt `GPolyline` ustvarimo v funkciji `drawPoly()`.

V situaciji je potrebno sedaj uporabiti podatek `offsetDistance`, ki predstavlja odmik v metrih od lokacijske točke v smeri kilometraže. Google Maps API zaenkrat ne premore ukaza, ki bi računal oddaljenosti po polilinijah. Potrebno bo definirati novo metodo objekta `GPolyline`, ki jo bomo imenovali `GetPointAtDistance()`. Ta bo ob podatku odmika v metrih od začetne točke polilinije, vrnil objekt tipa `GLatLng`, ki bo predstavljal lokacijo prometne nesreče.

Delovanje metode `GetPointAtDistance()`, opišemo z naslednjimi parametri:

- `metres` ...predstavlja odmik v metrih od začetka polilinije
- `olddist` ...predstavlja razdaljo od začetka polilinije do i-2, ki se nahaja takoj pred nesrečo
- `dist` ...predstavlja razdaljo od začetka polilinije do i-1, ki se nahaja takoj po nesreči



***Slika 8.6.2** Dolžine `olddist, metres` in `dist` (vir: lasten vir)*

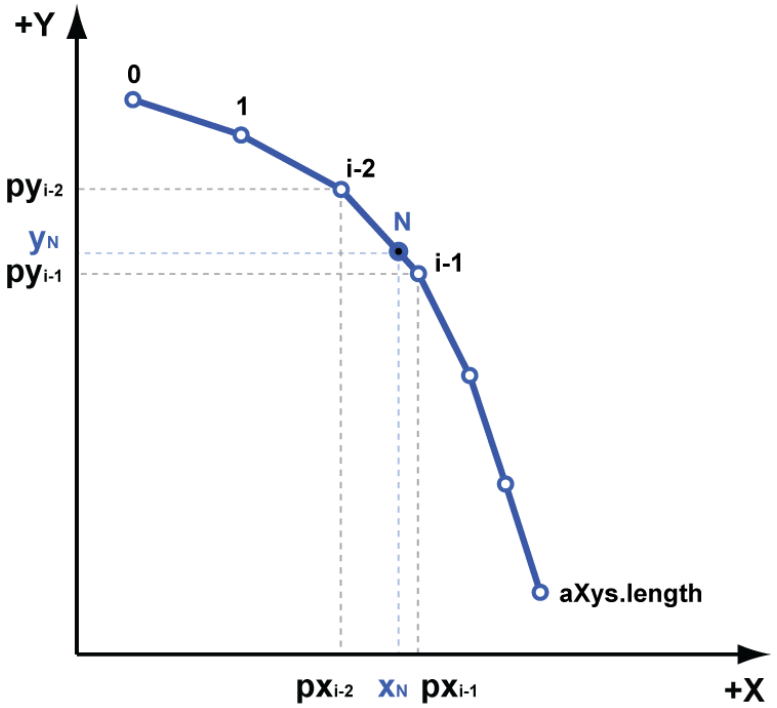
- Kvocient razdalje od i-2 do N, glede na celotno dolžino segmenta, ki sega od točke z indeksom i-2 do točke z indeksom i-1:

- p_1 je točka, ki predstavlja objekt tipa `GLatLng` na na poliliniji z indeksom i-2:

```
var p1= this.getVertex(i-2);
```

- p_2 je točka, ki predstavlja objekt tipa `GLatLng` na poliliniji z indeksom i-1:

```
var p2= this.getVertex(i-1);
```



Slika 8.6.3 *Graf s koordinatami točk, ki so potrebne za določitev lokacije nesreče (vir: lasten vir)*

- Koordinate nesreče (X_N,Y_N):

Za določitev koordinat nesreče smo posebej računali latitudo in longitudo. Razdalji `olddist` smo torej prišteli dolžino segmenta na katerem se je zgodila nesreča pomnoženega z kvocientom `m`.

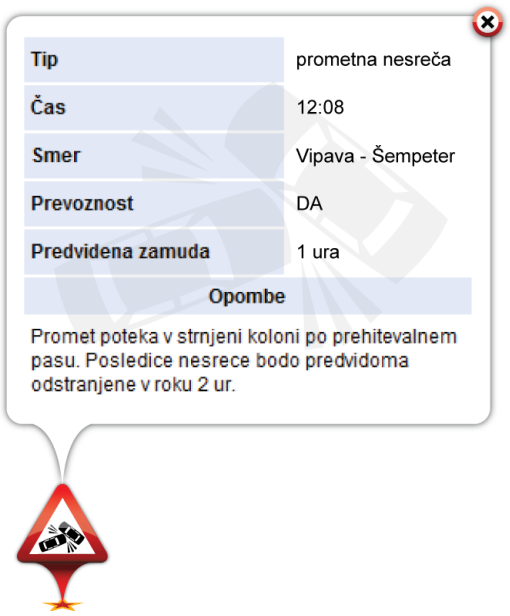
Podatke o nesreči bomo shranili v spremenljivko `point` z naslednjim ukazom:

```
var point = poly.GetPointAtDistance(offset)
```

9 INFORMACIJSKO OKNO – *INFOWINDOW*

9.1 Opis

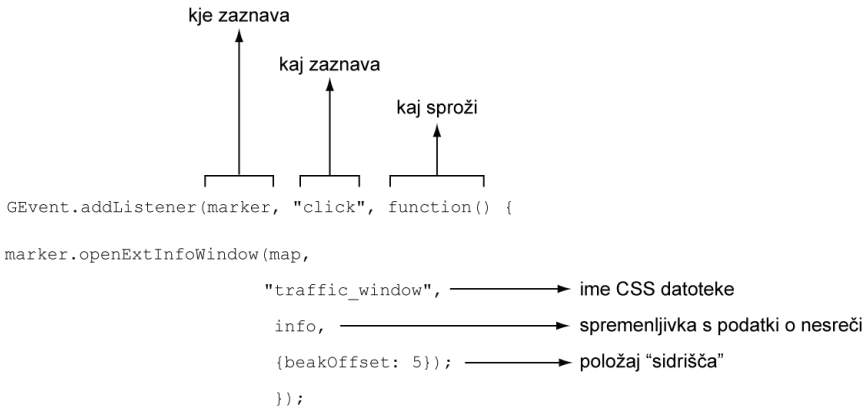
Podatke o prometni nesreči bomo prikazovali v t.i. informacijskem oknu (*»infowindow«*). Ob prikazu dogodkov na karti, okno ni odprto. Če hočemo, da se okno prikaže je potreben klik na marker *(Slika 9.1.1)*.



Slika 9.1.1 *Marker z vklopljenim infowindow-om (vir: lasten vir)*

Za prikaz informacijskega okna je v programerskem smislu potrebno definirati t.i. *»event listener«* , ki zaznava, če uporabnik klika po marker-ju. Ko je klik zaznan sproži funkcijo, ki izriše informacijsko okno.

Primer kode:



***Slika 9.1.2** Event listener (vir: lasten vir)*

9.2 Vsebina informacijskega okna

Podatki o prometni nesreči, ki se bojo prikazovali v informacijskem oknu so shranjeni v spremenljivki vrste `Array()` z imenom `situVars`.

Vsebina spremenljivke `situVars`:

```
situVars = {"situation type": [type],
            "publication time": [time],
            "delay interval": [delayInterval],
            "delay time": [delayTime],
            "restricted lanes": [laneRestricted],
            "operational lanes": [laneOperational],
            "diversion": [diversion],
            "LCD point A": [LCDpointA],
            "offset distance": [offsetDis],
            "offset direction": [alertCDirectionCoded],
            "comments": [comment]};
```

`situVars` je asociativen `array` objekt, kar pomeni, da so njegove vrednosti shranjene pod ključi, ki so `string` oblike. Torej če želimo dobiti podatek `delayTime`, to naredimo na sledeči način:

```
var delay = situVars["delay time"]
```

Vsebina spremenljivke `delay` bo enaka spremenljivki `delayTime`.
Zapis podatkov preko standarda DATEX II je za povprečnega uporabnika nerazločen in zakompliciran, zato moramo dodati par funkcij in datotek, ki nam bojo pomagale pri urejanju podatkov.

9.2.1 Primeri:

1.) Dodaten čas vožnje (`delayTime`), ki ga bomo potrebovali glede na normalne razmere je zapisani v sekundah (npr. 13435 sekund), kar je nečitljivo. Za pretvorbo v format ure : minute pokličemo Javascript datoteko *seconds2HMS.js,* ki vsebuje funkcijo `convertHMS()`. Ta pretvori podatek 13435 sekund v format 03:43.

2.) Čas publikacije dogodka je prvotno zapisan v formatu 2008-04-08T15:18:15 .719875+02:00, spremenimo ga v 15:18.

3.) Smer *offset*-a tudi ne pove uporabniku v kateri smeri se je nesreča zgodila. Podatek smo tako priredili, da se v primeru pozitivnega *offset-a* izpiše smer Vipava – Šempeter, v primeru negativnega pa Šempeter – Vipava.

4.) Če je zapisno v XML datoteki, da tip dogodka enak `'Accident'`, bo v informacijskem oknu, pod razdelkom »Tip« izpisalo »Prometna nesreča«

5.) Če bo podatek `numberOfOperationalLanes`, ki nam pove število prevoznih pasov bil večji od 0, bo izpisalo »da« v razdelku »Prevoznost« in »ne« v primeru, če bo enak 0.

Pupovac, P. 2008. Spletna aplikacija za izmenjavo in prikaz prometnih podatkov
Diplomska naloga – UNI. Ljubljana, UL, FGG, Oddelek za gradbeništvo, Prometna smer.

9.3 Oblikovanje informacijskega okna s pomočjo CSS datotek

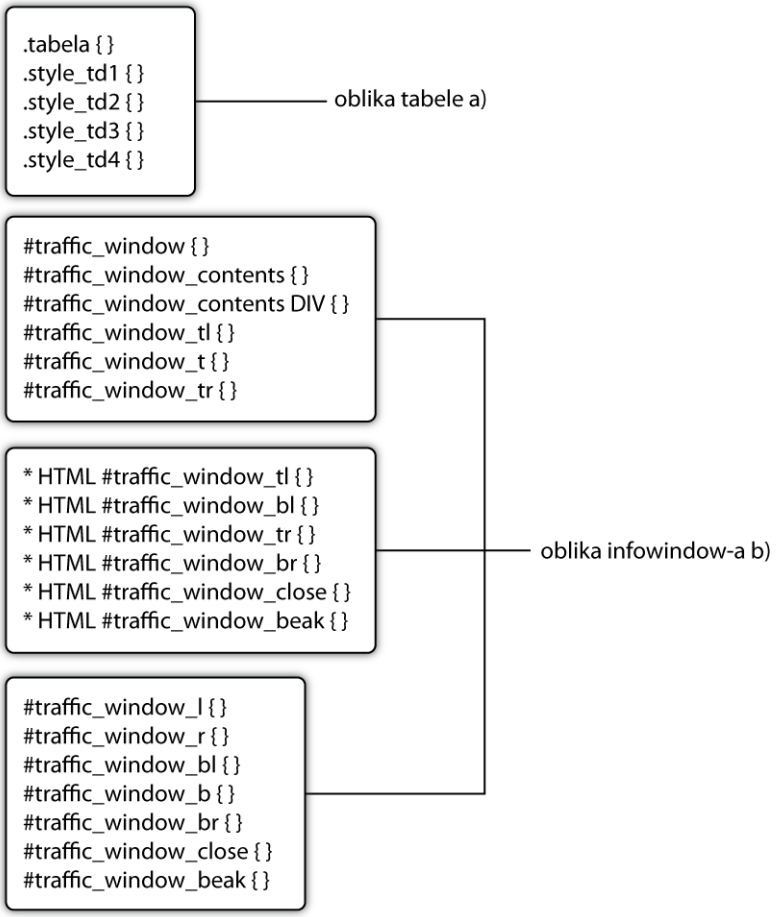
Oblika informacijskega okna je poljubna. Osnovni izgled, ki ga predlaga Google je precej enostaven, zato sem se odločil da ga priredim za potrebe prikazovanja prometnih informacij.

Kompleksnejše oblikovanje informacijskega okna zahteva datoteko CSS (»cascading style sheets«), v kateri definiramo položaj grafičnih elementov informacijskega okna *(Slika 34).* V našem primeru to naredimo v datoteki *infowindow.css.*

Struktura datoteke *infowindow.css*, ki določa položaje in obliko elementov okna lahko v splošnem razdelimo na elemente:

a) tabele

b) informacijskega okna



Slika 9.3.1 *Poenostavljena struktura datoteke infowindow.css (vir: lasten vir)*

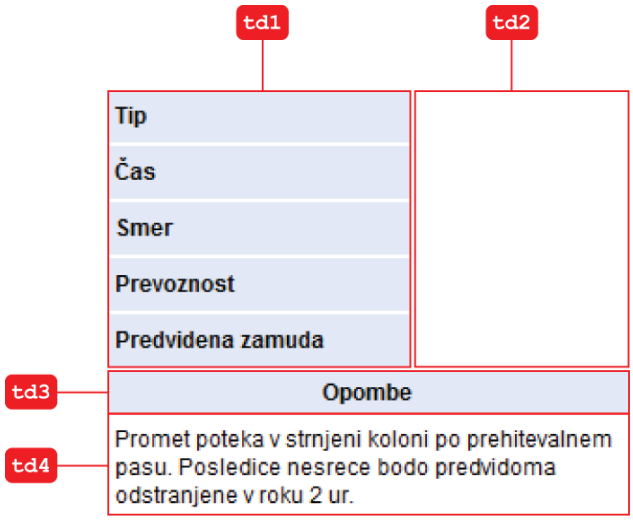
Pupovac, P. 2008. Spletna aplikacija za izmenjavo in prikaz prometnih podatkov
Diplomska naloga – UNI. Ljubljana, UL, FGG, Oddelek za gradbeništvo, Prometna smer.

a) Tabela

Tabelo prvo razgradimo na osnovne gradnike *(Slika 9.3.2)*. Te predstavljajo elementi kot so vrstice, stolpci in tekstovna vsebina. Vsakemu izmed njih dodelimo lastnosti kot so barve, prozornost, debelina robov.

Primer za element `td1`:

```
.style_td1 {
      background-color: #E2E7F5;
      height: 17px;
      font-weight: bold;
      width: 45%; }
}
```



***Slika 9.3.2** CSS gradniki tabele (vir: lasten vir)*

b) Informacijsko okno

Informacijsko okno vsebuje več gradnikih, ki se po razporeditvi zelo razlikujejo. V prvo skupino ločimo elemente, ki se pojavijo samo enkrat. To so elementi, ki nastopajo v kotih:

- `tl` ... zgornji levi kot
- `tr` ... zgornji desni kot
- `bl` ... spodnji levi kot

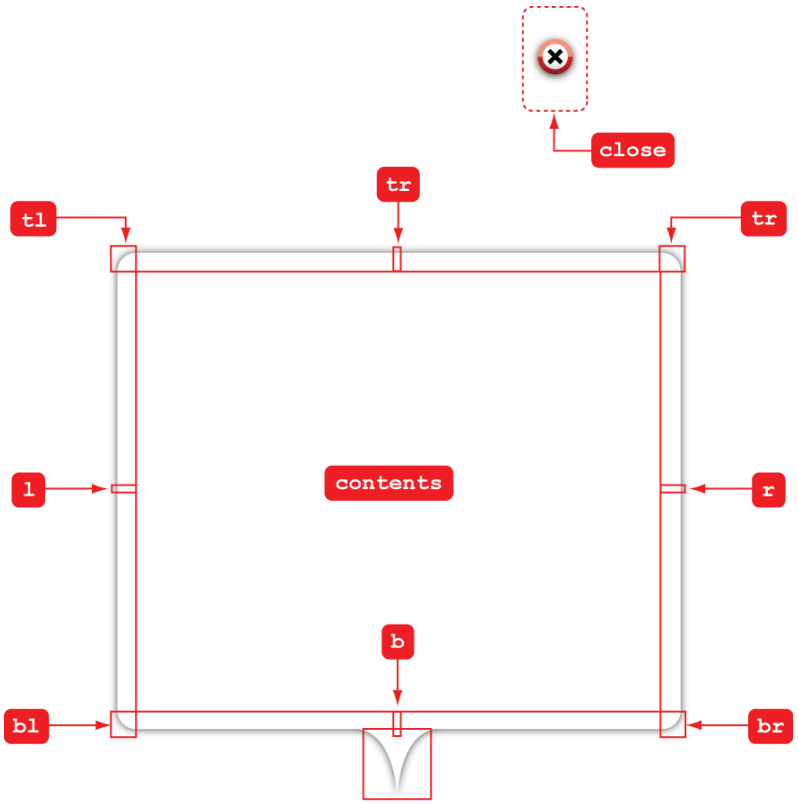
- `br` ... spodnji desni kot

V drugo skupino ločimo elemente, ki predstavljajo robove:

- `l` ... levi rob
- `r` ... desni rob
- `t` ... zgornji rob
- `t` ... spodnji rob

V tretjo skupino pa ostale elemente

- `close` ... gumb za zapiranje
- `beak` ... sidrišče
- `contents` ... vsebina



Slika 9.3.3 *CSS gradniki informacijskega okna (vir: lasten vir)*

10 ZAKLJUČEK

V programu so bile prikazane različne funkcije in objekti, ki smo jih sestavili za potrebe prikazovanja informacij o prometnih nesrečah. Enake koncepte bi lahko uporabili pri izdelavi pravega prometnega portala, ki operira z bistveno večjo količino podatkov. Na strežniku bi postavil bazo podatkov MySQL, v kateri bi shranil vse lokacijske točke za področje Slovenije. Določeni del kalkulacij, kot je določanje pozicije glede na os polilinije, bi prestavil na strežnik z uporabo jezika PHP, medtem ko končni izračun lokacije in osnovne podatke o nesreči, bi še vedno potekale v programskem jeziku Javascript. Takšen način bi vzel bistveno manj dodatnega časa pri nalaganju spletne strani, saj bi Javascript koda, ki se izvaja v brskalniku, imela opravka z preračunavanjem manjše količine spremenljivk kot sicer.

Na relaciji avto – osebni računalnik bo potrebno v prihodnosti tehnologijo še dodatno integrirati za delo z naprednimi GPS navigacijskimi napravami, ki sprejemajo podatke v živo preko RDS-TMC tehnologije. Tako bi ustvarili zaokrožen sistem delujoč v skladu s standardom DATEX II, ki bi na območju evropske unije ponujal podatke ne glede nato ali se trenutno nahajamo v pisarni pred računalnikom, avtu, ali nekje, kjer je edini vir informacij GSM / GPRS mobilni telefonski signal.

VIRI

Literatura

- Sas Jacobs, l. 2006, Beginning XML with DOM and Ajax From Novice to Professional, Berkley, Apress, 455 strani
- Christian Heilmann, l. 2006, Berkley, Beginning JavaScript with DOM Scripting and Ajax From Novice to Professional, Apress, 510 strani
- Michael Purvis, Jeffrey Sambells in Cameron Turner, l. 2006, Berkley, Beginning Google Maps Applications with PHP and Ajax From Novice to Professional, Wiley publishing, 383 strani
- Martin C. Brown, l. 2006, Indianapolis, Hacking Google Maps and Google Earth, 401 strani
- Danny Goodman, l. 2001, New York, Javascript Bible Gold edition, Hungry Minds, 2177 strani
- John Resig, l. 2006, Berkley, Pro JavaScript Techniques, Apress 380 strani

Internetni viri

- Google Maps Groups 23.10. 2008:

  http://groups.google.com/group/Google-Maps-API

- Google Maps API reference 23.10. 2008:

  http://code.google.com/apis/maps/documentation/reference.html

- Google Maps API tutorial 23.10. 2008:

  http://econym.googlepages.com/index.htm

- Some experiments with Google Maps 23.10. 2008:

  http://maps.forum.nu/

11 PRILOGE

11.1 *diploma.html*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
<title> Diplomska naloga Google Maps </title>
<link type="text/css" rel="Stylesheet" media="screen" href="infowindow.css"/>
<script src="http://maps.google.com/maps?file=api&amp;v=2.x&amp;key=ABQIAAAAJ-
LZC4gxEkgqcCchsxalsBQ6nYoSVFdPp0Aihik7hV0iznr3eRRZQdUvPS5pJzRtjXfIVIvYcHzySA"
type="text/javascript"></script>
<script src="http://gmaps-utility-
library.googlecode.com/svn/trunk/markermanager/release/src/markermanager.js"></scri
pt>
<script src="seconds2HMS.js"></script>
<script src="extinfowindow.js"></script>
<script src="epoly.js" type="text/javascript"> </script>
<script src="prototype.js"></script>
<script type="text/javascript">

var map = null;
var mgr = null;

var l = 22;
var h = 28;

var GEOSS = new GLatLng(46.119944, 14.815333);
var Ljubljana = new GLatLng(46.051314066826876,14.504013061523438);

var aXys = new Array();
var smallPoly = new Array();
var LCDarray = new Array();



var numBatches = 3;
var markerBatches = [];
```

```
var icon = new Array();
var point = null;
var situVars = null;

function setupMap() {

      if (GBrowserIsCompatible()) {

            map = new GMap2(document.getElementById("map"));
            map.setCenter(Ljubljana, 9);
            map.addControl(new GLargeMapControl());
            map.addControl(new GMapTypeControl());
            map.addControl(new GScaleControl());

            map.addMapType(G_PHYSICAL_MAP);
            map.setMapType(G_PHYSICAL_MAP);

            G_PHYSICAL_MAP.getMinimumResolution = function () { return 9 };
            G_NORMAL_MAP.getMinimumResolution = function () { return 9 };
            G_SATELLITE_MAP.getMinimumResolution = function () { return 9 };
            G_HYBRID_MAP.getMinimumResolution = function () { return 9 };

            G_PHYSICAL_MAP.getMaximumResolution = function () { return 11 };
            G_NORMAL_MAP.getMaximumResolution = function () { return 11 };
            G_SATELLITE_MAP.getMaximumResolution = function () { return 11 };
            G_HYBRID_MAP.getMaximumResolution = function () { return 11 };

            mgr = new MarkerManager(map, {maxZoom: 19});


            logoFGG = new GScreenOverlay('images/logoFGG.png',
                      new GScreenPoint(0.15, .005, 'fraction', 'fraction'),
                       new GScreenPoint(0, 0),
                       new GScreenSize(383, 62));

            map.addOverlay(logoFGG);

             for (var j = 0; j < numBatches; j++) {

                  icon[j] = new GIcon();
                  icon[j].image = 'images/accident' + j + '.png';

                  icon[j].iconSize = new GSize(l*(j+1), h*(j+1));
```

```
                    icon[j].iconAnchor = new GPoint((l/2)*(j+1), h*(j+1));
                    icon[j].infoWindowAnchor = new GPoint((l/2)*(j+1) -5, 0);
                    icon[j].shadow = 'images/shadow-accident' + j + '.png';
                    icon[j].shadowSize = new GSize(37*(j+1), 28*(j+1));

                    markerBatches[j] = [];
        }
   }
  loadPOLY();
  loadSITUATION();
   }


      function loadPOLY() {

            var request = GXmlHttp.create();
                  request.open("GET", "polyline2.xml",false);
                  request.onreadystatechange = function() {

                  if (request.readyState == 4) {
                        var xmlDoc = GXml.parse(request.responseText);

var polyline = xmlDoc.documentElement.getElementsByTagName('p');

      for (var i = 0; i < polyline.length ; i++) {

            var LCD = parseFloat(polyline[i].getAttribute('LCD'));
             LCDarray.push(LCD);


            var aXysLat = parseFloat(polyline[i].getAttribute('lat'));
            var aXysLng = parseFloat(polyline[i].getAttribute('lng'));
            var vertex = new GLatLng(aXysLat,aXysLng);

            aXys.push(vertex);
      }
              }
                  }
        request.send(null);
}

  function loadSITUATION() {
```

```
        var request2 = GXmlHttp.create();
        request2.open("GET", "dogodki2.xml", true);
        request2.onreadystatechange = function() {

if (request2.readyState == 4) {

      var xmlDoc = GXml.parse(request2.responseText);
      var datexModel = xmlDoc.documentElement.getElementsByTagName('situation');

      for (var i = 0; i < datexModel.length; i++) {


                           type =
datexModel[i].getElementsByTagName('situationRecord')[0].getAttribute('type');
                           time =
datexModel[i].getElementsByTagName('situationRecordCreationTime')[0].childNodes[0].
nodeValue;
                           delayInterval =
datexModel[i].getElementsByTagName('delayCoded')[0].childNodes[0].nodeValue;
                           delayTime =
datexModel[i].getElementsByTagName('delayTimeValue')[0].childNodes[0].nodeValue;
                           laneRestricted =
datexModel[i].getElementsByTagName('numberOfLanesRestricted')[0].childNodes[0].node
Value;
                           laneOperational =
datexModel[i].getElementsByTagName('numberOfOperationalLanes')[0].childNodes[0].nod
eValue;
                           diversion =
datexModel[i].getElementsByTagName('diversionAdvice')[0].childNodes[0].nodeValue;
                           LCDpointA =
datexModel[i].getElementsByTagName('specificLocation')[0].childNodes[0].nodeValue;

                           offsetDis =
datexModel[i].getElementsByTagName('offsetDistance')[0].childNodes[0].nodeValue;
                           offsetDir =
datexModel[i].getElementsByTagName('offsetDirection')[0].childNodes[0].nodeValue;
                           comment =
datexModel[i].getElementsByTagName('comment')[0].childNodes[0].nodeValue;

                           situVars = {"situation type": [type],
                                       "publication time": [time],
                                        "delay interval": [delayInterval],
                                        "delay time": [delayTime],
```

```
                                        "restricted lanes": [laneRestricted],
                                        "operational lanes": [laneOperational],
                                        "diversion": [diversion],
                                        "LCD point A": [LCDpointA],
                                        "offset distance": [offsetDis],
                                        "offset direction": [alertCDirectionCoded],
                                        "comments": [comment]};
                                truncatePoly();
                         }
                }
        }
       request2.send(null);
 }

function truncatePoly() {
       var start = parseFloat(situVars["LCD point A"]);
       if( situVars["offset direction"] == 'negative') {
              LCDarray.reverse();
              aXys.reverse();

              var startLCD = LCDarray.find(start);
              var nextLCD = isNaN(parseFloat(LCDarray.find(start - 1)));

              if (nextLCD == false) {
                     var endLCD = LCDarray.find(start - 1);
              }
              else {
                     var endLCD = LCDarray.length;
              }
       }

       else {
              LCDarray;
              aXys;

              var startLCD = LCDarray.find(start);
              var nextLCD = isNaN(parseFloat(LCDarray.find(start + 1)));

              if (nextLCD == false) {
                     var endLCD = LCDarray.find(start + 1);
              }

else {
```

```
                var endLCD = LCDarray.length;
            }
 }
                smallPoly =  aXys.slice(startLCD, endLCD);
                drawPoly();
 }

function drawPoly() {

    poly = new GPolyline(smallPoly,"#ff0000", 5, 0.8);
    map.addOverlay(poly);
    getPointAtOffset();
}

function getPointAtOffset() {
            var offset = situVars["offset distance"];
            point = poly.GetPointAtDistance(offset);
            markerResizer();
    }

function markerResizer(){
     for (var j = 0 ; j < numBatches ; j++) {
           markerBatches[j].push(createMarker(point,icon[j],situVars));
           mgr.addMarkers(markerBatches[j], 9+j, 9+j);
  }
    mgr.refresh();
}

function createMarker(point, markerIcon, situVars) {

    var comment = situVars["comments"];

    if ( situVars["situation type"] == 'Accident') {
           var type ='Prometna nesreča';
    }
    if (situVars["offset direction"] ==    'positive') {
           var direction = 'Vipava - Šempeter';
    }
    else {
           var direction = 'Šempeter - Vipava';
    }
    if (situVars["operational lanes"] > 0 ) {
           var operational ='da';
```

```
	}
	else {
		var operational ='ne';
	}
	if (situVars['delay time'] != 0) {
	var delayTime = convertHMS(situVars['delay time']).toString().slice(0,5);
	}
	else {
		var delayTime = 'ni podatka';
	}
	var time = situVars["publication time"].toString().slice(11,16);
	var info =  '<table class="tabela">'
			+ '<tr>'
				+ '<td class="style_td1">Tip</td>'
				+ '<td class="style_td2">' + type + '</td>'
			+ '</tr>'
			+ '<tr>'
				+ '<td class="style_td1">Čas</td>'
				+ '<td class="style_td2">' + time + '</td>'
			+ '</tr>'
			+ '<tr>'
				+ '<td class="style_td1">Smer</td>'
				+ '<td class="style_td2">' + direction + '</td>'
			+ '</tr>'
			+ '<tr>'
				+ '<td class="style_td1">Prevoznost</td>'
				+ '<td class="style_td2">' + operational + '</td>'
			+ '</tr>'
			+ '<tr>'
				+ '<td class="style_td1">Predvidena zamuda</td>'
				+ '<td class="style_td2">' + delayTime + '</td>'
			+ '</tr>'
			+ '<tr>'
				+ '<td class="style_td3" colspan="2">Opombe</td>'
			+ '</tr>'
			+ '<tr>'
				+ '<td class="style_td4" colspan="2">' + comment + '</td>'
		+ '</tr>'
	+ '</table>'

var marker = new GMarker(point, markerIcon);
	GEvent.addListener(marker, "click", function() {
	 marker.openExtInfoWindow(map, "traffic_window", info, {beakOffset: 5});
```

```
        });
       return marker;
       }
</script>

       </head>
              <body onload="setupMap()"; onunload="GUnload()">
                     <div id="map" align="center" style="border:2px solid green;
width: 1000px; height: 800px"></div>
              </body>
</html>
```

11.2 funkcija `GetPointAtDistance()`

```
GPolygon.prototype.GetPointAtDistance = function(metres) {
  if (metres == 0) return this.getVertex(0);
  if (metres < 0) return null;
  var dist=0;
  var olddist=0;

  for (var i=1; (i < this.getVertexCount() && dist < metres); i++) {
    olddist = dist;
    dist += this.getVertex(i).distanceFrom(this.getVertex(i-1));
  }
  if (dist < metres) {return null;}

  var p1= this.getVertex(i-2);
  var p2= this.getVertex(i-1);
  var m = (metres-olddist)/(dist-olddist);

  return new GLatLng( p1.lat() + (p2.lat()-p1.lat())*m,
                      p1.lng() + (p2.lng()-p1.lng())*m);
```

11.3 ukaz `find()`

```
Array.prototype.find = function(searchStr) {
  var returnArray = false;
  for (i=0; i<this.length; i++) {
    if (typeof(searchStr) == 'function') {
```

```
      if (searchStr.test(this[i])) {
        if (!returnArray) { returnArray = [] }
        returnArray.push(i);
      }
    } else {
      if (this[i]===searchStr) {
        if (!returnArray) { returnArray = [] }
        returnArray.push(i);
      }
    }
  }
  return returnArray;
}
```