Dizajn interktivne aplikacije za spremanje podataka koristeći grafički prikaz zemljovida

Ravas, Alen

Undergraduate thesis / Završni rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:460025

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

DIZAJN INTERAKTIVNE APLIKACIJE ZA SPREMANJE PODATAKA KORISTEĆI GRAFIČKI PRIKAZ ZEMLJOVIDA

Završni rad

Alen Ravas

Osijek, 2016.

SADRŽAJ

1. Uvod 1
1.1. Zadatak završnog rada 1
2. TEHNOLOGIJE KORIŠTENE ZA IZRADU APLIKACIJE 2
2.1. HTML 2
2.1.1. Alati za pisanje HTML dokumenta 2
2.1.2. Osnove HTML-a 3
2.2. CSS 6
2.3. JavaScript 7
2.3.1.Tipovi podataka 7
2.3.2. Naredbe za kontrolu toka 8
2.3.3. Operatori 9
2.3.4. Funkcije 9
2.1.3 PHP 10
2.1.3 phpMyAdmin 10
3. Google Maps API 12
3.1 Google Maps Objekti 14
3.2 Kontrole 17
3.3 Vrste karata 19
3.3.1 Karte s nagibom 20
4. IZRADA APLIKACIJE 21
5. ZAKLJUČAK 29
LITERATURA 30
SAŽETAK 32
ABSTRACT 32
ŽIVOTOPIS 33
PRILOG 34

1. UVOD

Tema je ovoga rada dizajn interaktivne aplikacije za spremanje podataka koristeći grafički prikaz zemljovida. U radu se opisuje razvoj aplikacije izrađene pomoću *Google Maps* grafičkog sučelja čije korištenje zahtjeva poznavanje *HTML*-a i *JavaScripta*, a njihove su osnove također opisane u ovom radu.

U prvom su poglavlju opisane tehnologije koje je bitno poznavati da bi se mogla napraviti web aplikacija. Tu čitatelja upoznajemo sa osnovama *HTML*-a , *CSS*-a i *JavaScript*-a čije je poznavanje važno za rad s *Google Maps API*-em. U prvom je poglavlju također opisan *phpMyAdmin* pomoću kojega je napravljena baza podataka u koju će se spremati podatci izrađene aplikacije te su ukratko opisani principi *PHP*-a koji se koristi za povezianje aplikacije s bazom podataka.

Drugo poglavlje obrađuje principe *Google Maps API*-a koji je glavni alat za izradu aplikacije.

Zadnje se poglavlje odnosi na izradu same internet aplikacije koja je zadatak ovoga rada.

1.1. Zdatak završnog rada

Proučiti korištenje *HTML*-a, *Javascript*-a i *Google Maps* grafičkog sučelja (API-a), opisati principe *Google Maps* grafičkog sučelja. Kao primjer korištenja *Google Maps*-a napraviti aplikaciju koja pokazuje kartu i koja omogućuje da se označi na karti neko mjesto. Za označeno mjesto aplikacija omogućuje da se upisuju podaci i da se stavi slika koja se sprema na serveru, ta slika se pokazuje ako se klikne to označeno mjesto.

2. TEHNOLOGIJE KORIŠTENE ZA IZRADU APLIKACIJE

Za izradu aplkacije korišteno je nekoliko tehnologija i programskih jezika. Baza podataka za aplikaciju je relacijska *MySQL* baza podataka kreirana pomoću *phpMyAdmina*.[1] Podatci se unose u bazu preko *PHP*-a koji se prikazuje unutra *HTML* elemenata. Karta je pozvana pomoću *JavaScript*-a tj. *JavaScript* biblioteke *jQuery*.

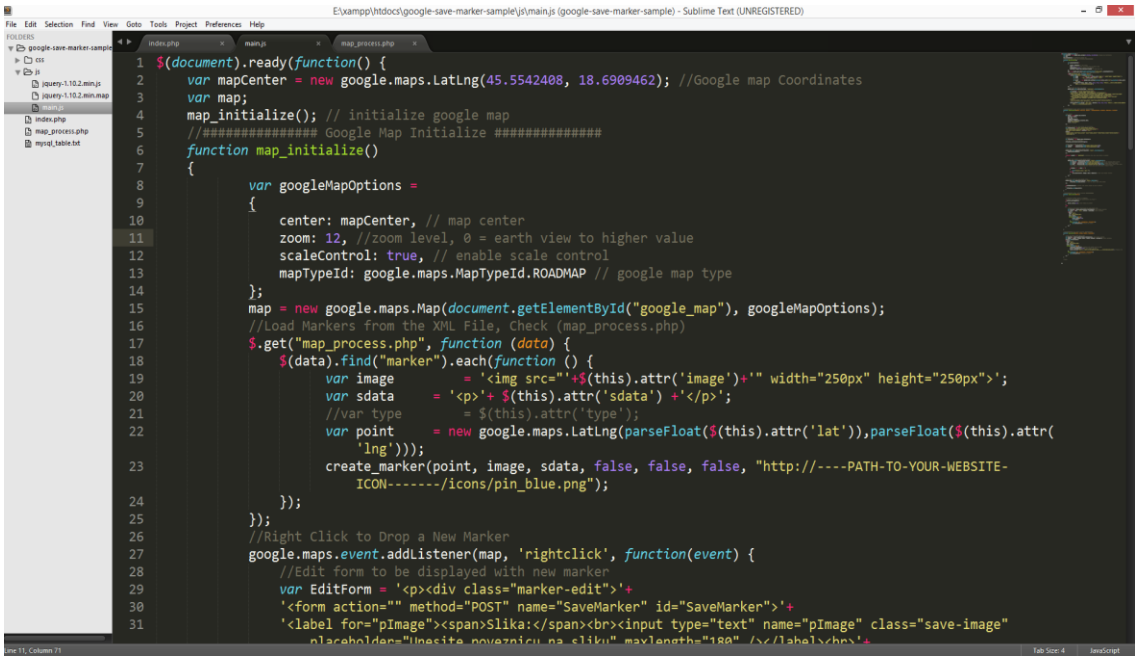
2.1. HTML

Osnovni jezik koji se koristi za izradu internet stranica je *HTML* (što je kratica za HyperText Markup Language). *HTML* jezikom oblikuje se sadržaj i stvaraju hiperveze hiper-tekstualnih dokumenata. Oni se od običnih dokumenata razlikuju po tome što sadrže hiperveze kojima su povezani s drugim hiper-tekstualnim dokumentima. Prikaz hiper-tekstualnih dokumenata omogućuje web preglednik, a temeljna zadaća *HTML* jezika je uputiti internet preglednik kako prikazati hiper-tekst dokument. *HTML* je važno razlikovati od ostalih programskih jezika jer on, za razliku od njih, služi samo za opis hiper-tekstualnih dokumenata i njime nije moguće izvršiti ni jednostavne zadatke poput obavljana osnovnih aritmetičkih radnji. *HTML* opisuje internet stranice koristeći obične tekstualne datoteke kojima je ekstenzija .html ili .htm. [2]

2.1.1. Alati za pisanje HTML dokumenta

Za izradu *HTML* dokumenta dovoljno je imati bilo kakav program za obradu teksta (npr. Notepad) i neki od internet preglednika, kako bismo provjerili izgled stranice. No, ručno pisanje koda u programu za obradu teksta može biti složeno jer korisnik može lako pogriješiti. Danas postoje mnogi napredni alati za pisanje koda koji često imaju mogućnost podcrtavanja sintakse, bojanje i isticanje oznaka, atributa i sadržaja tako da se međusobno razlikuju, uređivanje više različitih linija koda odjednom, automatsko dopunjavanje napisanih naredbi, a tekst se može formatirati na način da se mijenja veličina slova, mijenja stil, umeću tablice, slike, poveznice i multimedija. Neki od naprednih alata su: *Sublime Text* [3] (Slika 2.1) , *Brackets* [4] i *PhpStorm* [5].

Postoje alati poput *Adobe Dreamweaver*-a [6] i *KompoZer*-a [7] koji, osim ručnog pisanja koda, omogućuju vizualnu izradu web stranica i bez poznavanja koda. Oni rade po principu WYSIWYG (What You See Is What You Get) što prevedeno na hrvatski znači *ono što vidiš, to ćeš i dobiti*. Na taj način korisnik bez poznavanja *HTML* koda na jednostavan i intuitivan način može oblikovati sadržaj jer je izgled web stranice odmah vidljiv i dostupan. [8]



Sl. 2.1. *Prikaz podcrtavanja sintakse u Sublime Text uređivaču*

2.1.2.Osnove HTML-a

HTML dokumenti opisani su elementima koji sastoje od *tagova* i sadržaja koji se nalazi u tim tagovima. Oni govore internet pregledniku na koji način prikazati sadržaj internet stranica. Tagovi su ključne riječi koje se zapisuju unutar izlomljenih zagrada (“<>“).

HTML elementi sadrže početne tagove i završne tagove `< tag >` ,`< /tag >`, no postoje i tagovi kod kojih se ne mora stavljati završni tag, a koriste se kod prekida linija i ubacivanja slika (*
, <img>*).

Svaki *HTML* dokument ima određenu strukturu (Slika 2.2). Na početku svakog *HTML* dokumenta postavlja se `<!DOCTYPE html>` tag kojim se definiraju standardi za izradu *HTML* dokumenta. On pomaže internet pregledniku da ispravno prikaže dokument. Nakon njega

postavlja se osnovni `<html>` tag koji internet pregledniku govori da je datoteka koju učitava *HTML* kod. Unutar `<html>` taga nalazi se sadržaj *HTML* dokumenta, tj. ostali *HTML* tagovi.

Meta podatci *HTML* dokumenta pišu se unutar `<head>` taga. Tag `<head>` može sadržavati poveznice na dodatni *CSS* pomoću `<link>` taga, poveznice na *JavaScript* datoteke pomoću `<script>` taga, naslov kartice u pregledniku pomoću `<title>` taga. Meta podatci nisu prikazani na stranici kada stranicu pregledavamo u pregledniku.

Sadržaj koji se prikazuje *HTML* dokumentom stavlja se unutar `<body>` taga te predstavlja tijelo *HTML* dokumenta. Unutar njega se upisuje ono što se želi prikazati u pregledniku.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="style.css">
    <title>Naslov kartice u pregledniku</title>
  </head>

<body>

  <h1>Naslov</h1>

  <p>Paragraf</p>
  <!--html komentar ovo se ne prikazuje -->
  <a href="https://en.wikipedia.org">Hiperveza</a>

  <img src="https://upload.wikimedia.org/wikipedia/commons/thumb/8/84/HTML.svg/266px-HTML.svg.png" alt
  ="alternative text">
  <br>
  <br>
</body>
</html>

```

Sl. 2.2. *Osnovna struktura HTML dokumenta*

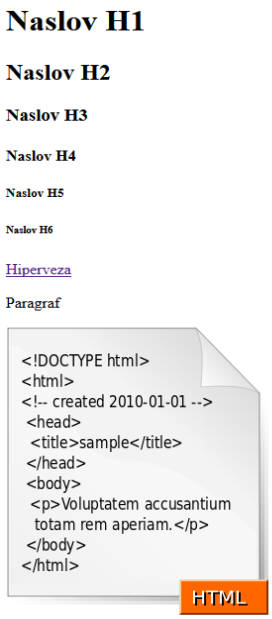
Radi bolje preglednosti i organizacije stranice te odvajanja različitih tema dokmenta koriste se tagovi od `<h1>` do `<h6>`, gdje `<h1>` tag definira najvažniji naslov s najvećom veličinom fonta, a `<h6>` tag najmanje važan naslov s najmanjim fontom (Slika 2.3). Za definiranje novog odlomka, koristi se `<p>` tag koji sadržaj zapisuje u jedan paragraf. Kada se unutar odlomka želi prijeći u novi red koristi se `<br>` tag. Pomoću `<a>` taga definira se hiperveza koja se koristi za povezivanje dvaju stranica, a najvažniji atribut za `<a>` element je *„href“* atribut koji definira kamo vodi hiperveza. Atributi se koriste da bi dodatno opisali tagove, atribut općenito izgleda ovako:

`<tag attribute="value"></tag>.`

Neki od važnijih atributa su *„id“* i *„class“* koji služe za zahvaćanje elemenata preko identifikacijske riječi koja im se dodjeljuje. Razlika između *id*-a i klase je ta što je *id* jedinstven za svaki element i svaki element može imati samo jedan *id*, a klasu može dijeliti više elemenata, te jedan element može imati samo jednu klasu.

Pravila sintakse određuju se tako da vrijednost atributa mora biti u navodnicima. Tablice u *HTML*-u definiraju se `<table>` tagom koji se sastoji od `<tr>` elemenata koji definira redak tablice, `<th>` elemenata koji definira zaglavlje tablice i `<td>` elementa koji definira ćeliju tablice. Liste s oznakama definiraju se pomoću `<ul>` taga, a pojedine stavke unutar liste se definiraju pomoću `<il>` taga. Slike na u *HTML*-u definiramo `<img>` tagom. On mora sadržavati atribut *„src“* koji korisniku pokazuje na lokaciju slike. Prikazani primjer sadrži još i atribut *„alt“* koji nam pruža alternativne informacije o slici ukoliko se slika iz nekog razloga ne može vidjeti (Slika 2.2.).

HTML ima mogućnost postavljanja komentara u izvorni kod, a postavljaju se pomoću taga `<!-- komentar-->`. Komentari se koriste za pojašnjenje koda i posebno su korisni kada ima puno koda jer pomažu pri lakšem snalaženju kada kod treba mijenjati. Komentari su često označeni drugom bojom kako bi se razlikovali od od koda. Komentari se ne prikazuju u internet pregledniku.[9]



Sl. 2.3. *Prikaz elemenata h1-h6, p, a i img u internet pregledniku*

2.2. CSS

CSS je kratica od Cascading Style Sheets. Radi se o stilskom jeziku koji se rabi za opis prezentacije dokumenta napisanog pomoću markup *HTML* jezika. *CSS* se razvio zbog malih mogućnosti prezentacijskih *HTML* elemenata te je dizajniran kako bi omogućio razdvajanje sadržaja *HTML* dokumenta od izgleda dokumenta. *CSS* se koristi za defniranje stilova internet stranica, uključujući i dizajn, izgled i varijacije na zaslonu za različite uređaje i veličine zaslona.[10]

Sintaksa *CSS*-a sastoji se od dva elemenata: selektora i deklaracijskog bloka. Deklaracija se sastoji od jednog ili grupe svojstava koji se odvajaju točka-zarezom, a svako se svojstvo sastoji od naziva, dvotočke i vrijednosti tog svojstva. Selektor označava *HTML* element koji želimo oblikovati, tj. element na koji će biti primjenjen deklaracijski blok.

```
element {  svojstvo: vrijednost; }
```

CSS se u *HTML* dokumentu može pisati unutar samog taga kao atribut *style*="svojstvo:vrijednost", unutar `<style></style>` taga te pozvati kao vanjsku datoteku pomoću `<link>` taga.

```
<link rel="stylesheet" type="text/css" href="putanjaDo/datoteka.css">
```

```
/* width and height of google map */
#google_map {
    height: 100%;
    width: 100%;
    left: 0;
    position: absolute;
    top: 0;
}

/* Marker Edit form */
.marker-edit label{display:block;margin-bottom: 5px;}
.marker-edit label span {width: 100px;float: left;}
.marker-edit label input, .marker-edit label select{height: 24px;}
.marker-edit label textarea{height: 60px;}
.marker-edit label input, .marker-edit label select, .marker-edit label textarea {width: 87%;margin:0px;padding-Left:
    5px;border: 1px solid #DDD;border-radius: 3px;}

/* Marker Info Window */
h1.marker-heading{color: #585858;margin: 0px;padding: 0px;font: 18px "Trebuchet MS", Arial;border-bottom: 1px dotted #
    D8D8D8;}
div.marker-info-win {max-width: 300px;margin-right: -20px;}
div.marker-info-win p{padding: 0px;margin: 10px 0px 10px 0;}
div.marker-inner-win{padding: 5px;}
button.save-marker, button.remove-marker{border: none;background: rgba(0, 0, 0, 0);color: #00F;padding: 0px;
    text-decoration: underline;margin-right: 10px;cursor: pointer;
}
.gm-style-iw div{ overflow: hidden !important; }
.main-title { position: absolute; width: 99%; font-size: 16px; font-family: sans-serif;z-index: 2;opacity: 0.6; }
```

Sl. 2.4. *Primjer CSS dokumenta*

2.3. JavaScript

Javascript je skriptni programski jezik koji dodaje interaktivnost internet stranicama. *JavaScript* je skriptni programski jezik zato što programi napisani u *JavaScriptu* ne trebaju poseban kompajler. Kako bi se rasteretio poslužitelj, moguće je napisati *JavaScript* kod koji će se izvršavati na klijentskoj strani, odnosno u pregledniku. Osim na klijentskoj strani, *JavaScript* se može izvršavati i na poslužitelju pomoću okruženja kao što je *Node.js*.[11]

Krajnji korisnik treba imati omogućeno izvršavanje JavaScripta u postavkama vlastitog internet preglednika, a svi moderni internet preglednici imaju podršku za *JavaScript*. *AJAX* ili Asinkroni *JavaScript* i *XML* povećava interaktivnost internet stranica tako što omogućava komunikaciju s programom koji se izvršava na Web poslužitelju.

JavaScript ne može izvršavati neposredne upite na *MySQL* bazi osim uz pomoć *Node.js-a*. Skriptni programski jezik *JavaScript* ima mogućnost objektno orijentiranog programiranja, ali ne podržava sve značajke tog pristupa. [12]

2.3.1. Tipovi podataka

Tipovi podataka koje *JavaScript* podražava su: brojevi, boolean vrijednosti, zankovna polja, nizovi i objekti. *JavaScript* varijable su dinamičke, a to znači da ista variabla može pohraniti različite tipove podataka [13]. Na slici (Slika 2.5.) prikazana je osnovna sintaksa za deklaraciju navedenih *JavaScript* tipova podataka.

```
1  var broj = 1;
2  var booleanVrijednost = true;
3  var zankovnoPolje = "Znakovi";
4  var niz = ["jedan","dva"];
5  var objekt = {
6      nazivVrijednosti: "vrijednost",
7      nazivVrijednosti2: "vrijednost2",
8      nazivBrVrijednosti: 128
9  };
```

Sl. 2.5. *Sintaksa za deklaraciju tipova podataka u JavaScriptu*

2.3.2. Naredbe za kontrolu toka

Naredbama za kontrolu toka se određuju uvjeti koji se trebaju ispuniti kako bi se dio programa izvršio. Naredbe za kontrolu toka su: *if*, *if elese*, *if else if*, *while*, *do while*, *for* i *switch*. *If* naredbom se ispituje uvjet u zagradi te ako je uvjet ispunjen blok unutar *if* naredbe se izvršava u suprotnom se preskaće. [14]

If naredba može sadržavati više dijelova, tj. proširenje naredbe uvjetom *else* koji se izvršava ako prvi uvjet nije zadovoljen. Proširenje naredbe uvjetom *else if* definira drugi uvjet koji se izvršava ako je taj uvjet ispunjen, a početni *if* nije.

Blok *while* petlje izvršava se sve dok je navedeni uvjet istinit. Razlika između *do while* petlje i *while* petlje je ta što se kod *do while* petlje prvo izvršava blok naredbi, a zatim se ispituje uvjet.

U slučaju *for* petlje koriti se početni izraz, uvjet koji ispituje hoće li se petlja izvršiti i završni izraz.

Switch prima vrijednost i ispituje odgovara li primljena vrijednost definiranim uvjetima. Ako odgovara, blok s odgovarajućim uvjetom se izvršava. U suprotnom se izvršava „default“ zadani blok. Na slici (Slika 2.6.) prikazana je osnovna sintaksa kontrole toka programa *JavaScriptom*.

```
if ( a < b){
    console.log("a je manji od b");
}else if(a > b) {
    console.log("a je veci od b");
}else {
    console.log("a == b");
}

while (a < b){
    console.log("izvrsava se sve dok je uvijet ispunjen")
}

do {
    console.log("izvrsava se prije ispitivanja uvijeta")
}while(a < b);

for (var a=0; a <10; a+=1){
    console.log(a);
    console.log("Ispisuje se sve dok je a < 10. a se povecava za 1 svaki krug");
}

var x = 3;

switch(x){
    case(1):
    console.log("Ispisujem ako je x = 1");
    break;
    case(2):
    console.log("Ispisujem ako je x = 2");
    break;
    case(3):
    console.log("Ispisujem ako je x = 3");
    break;
    default:
    console.log("Ispisujem ako ni jedan od uvijeta nije zadovoljen");
    break;
}
```

Sl. 2.6. *Sintaksa naredbi za kontrolu toka u JavaScriptu*

2.3.3. Operatori

Aritmetički operatori koji su ugrađeni u *JavaScript* slični su operatorima ostalih programskih jezika. *JavaScript* operatori su: operator dodjeljivanja +, operator oduzimanja -, operator množenja *, operator djeljenja / i modulo operator % koji vraća ostatak kod dijeljenja.

Operatori dodjeljivanja ugrađeni u *JavaScript* su: dodjeljivanje vrijednosti varijable =, uvećanje vrijednosti +=, smanjivanje vrijednosti -=, množenje vrijednosti *=, dijeljenje vrijednosti /=, dodjeljivanje modulo vrijednosti %=.

Operatori uspoređivanja i logički operatori ugrađeni u *JavaScript* su: jednakost ==, jednakost vrijednosti i tipa ===, nejednakost !=, nejednakost vrijednosti i tipa !==, veći od >, manji od <, veći ili jednak >=, manji ili jedank <= i terarni operator ?.

2.3.4. Funkcije

Funkcije se pohranjuju kako bi se radnje koje je potrebno izvršiti na više mjesta u programu mogle izvršavati bez da ih se ponovno definira. Željene radnje se pohranjuju u funkciju te određuju koje vrijdnosti funkcija prima i koje se vrijednosti funkciji predaju. Takve funkcije pozivaju se imenom koje im je dodjeljeno i predane su im vrijednosti nad kojima se operacije trebaju izvršiti. Na slici (Slika 2.7.) prikazana je osnovna sintaksa za deklaraciju i pozivanje funkcije.

```
1  //Deklaracija funkcij
2  var nekaFunkcija = function (ime) {
3      alert("Pozdrav od browsera," + " " + ime);
4  };
5  //pozivanje funkcije
6  nekaFunkcija(prompt("Unesite svoje ime"));
```

Sl. 2.7. *Sintaksa za deklaraciju i pozivanje funkcija u JavaScriptu*

2.4. PHP

PHP je skriptni programski jezik koji se izvršava od strane Web poslužitea i najčešće se koristi za razvoj internet stranica s dinamičkim sadržajem te pisanje malih skripti za smanjivanje ponavljajućih poslova na Web poslužitelju.

Pojavio se 1995. godine, a trenutna stabilna verzija je 7, iako su starije verzije još uvijek u širokoj upotrebi.

PHP ima mogućnosti spajanja i manipulacije bazom podataka, rukovanja *HTML* formatima te ispisivanje *HTML* elemenata. Naredba za deklaraciju varijable u *PHP*-u ne postoji. Varijable se kreiraju kada im je dodjeljena vrijednost.

Tipovi podataka koji su omogućeni u *PHP*-u su: nizovi, cijeli brojevi, realni brojevi, bool vrijednosti, redovi, objekti, NULL vrijednosti i resursi. Na slici (Slika 2.8.) prikazana je sintaksa jednostavnog programa napisanog u *PHP*-u.

```php
<?php
// Deklaracija varijable int tipa
$varijabla = 11;

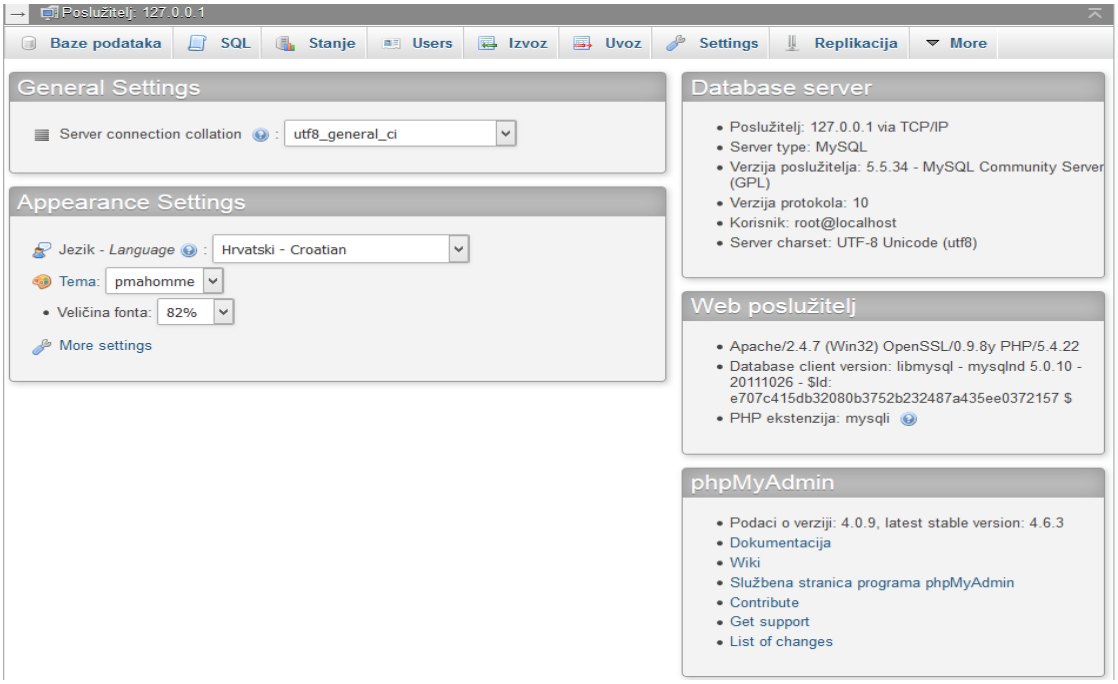
// Grananje
if ($varijabla < 10) {
    echo "Program ispisuje ako je varijabla $varijabla manja od 10";
}else ($varijabla > 10){
    // Petlja
    for ($i = 0; $i < 10; $i++){
        echo "Ovo program ispisuje 10x ako je varijabla $varijabla veca od 10";
    }
}
?>
```

Sl. 2.8. *Osnovna sintaksa skriptnog programskog jezika PHP*

2.5. phpMyAdmin

phpMyAdmin je *open-source* program napisan pomoću php skriptnog programskog jezika i koristi se za kreiranje i manipulaciju bazom podataka. Zbog grafičkog sučelja koje se prikazuje

unutar preglednika znatno olakšava rad na bazi, pregled baze podataka, izvoz baze, uvoz baze, operacije nad bazom i korisnicima baze (Slika 2.9.).



Sl. 2.9. *phpMyAdmin početni zaslon*

3. GOOGLE MAPS API

Google Maps je Googlova tehnolologija digitalnih karata koja obuhvaća cijelu površinu zemaljske kugle i može prikazivati razne oblike karata (Slika 3.1.).

API je akronim za Aplication Programming Interface što označava skup metoda, protokola i alata koji se mogu koristiti za izgradnju softverskih aplikacija. *API* dopušta korisnicma da koriste gotova rješenja i već postojeće projekte za izradu vlastite aplikacije. Mnoge velike web aplikacije imaju svoj *API*, a *Google Maps API* jedan je od najčešće korištenih.

Google API Maps omogućuje svojim korisnicima ugraditi *Google Karte* na web-stranice pomoću *JavaScript*-a. *Google Maps API* dizajniran je za rad na mobilnim uređajima i računalima.

API određuje kao bi softverske komponente trebale komunicirati i kako se programira grafičko korisničko sučelje. Dobar *API* olakšava izradu programa pružanjem svih elemenata koje korisnik spaja u jednu aplikaciju.

Google Maps API dopušta koisnicima da prikazuju *Google Maps* karte na vlastitim internetskim stranicama, da pristupaju njihovim funkcijama za uređivanje, te omogućuje prilagodbu karte i informacija na kartama.

Za pristupanje *Google Maps API*-u potrebo je imati *API* ključ koji dobivamo registracijom na Google preko *Gmail* računa.

Google Maps API je besplatan za korištenje, pod uvjetom da je stranica na kojoj se koristi javno dostupna da ne naplaćuje za pristup i da ne prelazi više od 25000 pristupa stranici u jednom danu. U slučaju da stranica ne ispunjava te uvjete, moguće je koristiti *Google Maps API* za posao koji pruža još neke dodatne mogućnosti, ali on se naplaćuje. [15][16][17]



Sl. 3.1. *Osnovna Google Maps karta*

Za učitavanje *Google Maps API*-a potrebno je definirati osnovnu *HTML* strukturu, unutar *script* taga napisati poveznicu na *API*, inicijalizirati kartu te joj definirati osnovne postavke: mjesto na kojemu je karta centrirana, koliko je karta zumirana te koju vrstu karte želimo prikazivati.

Nakon inicijalizacije, funkciju je potrebno pozvati te odrediti u kojem *HTML* elementu će se karta prikazivati i odrediti *CSS* svojstvima visinu i širinu karte koju prikazujemo (Slika 3.2.).

```
<!DOCTYPE html>
<html>
<head>
<script src="http://maps.googleapis.com/maps/api/js"></script>
<script>
function initialize() {
  var mapProp = {
    center:new google.maps.LatLng(51.508742,-0.120850),
    zoom:5,
    mapTypeId:google.maps.MapTypeId.ROADMAP
  };
  var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);
}
google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>
</body>

</html>
```

Sl. 3.2. *Osnovna postavke za prikazivanje Google Maps karte*

3.1. Google maps objekti

Google Maps API omogućuje korištenje nekoliko metoda za prikazivanje objekata na karti koji su vezani za određene koordinate. Objekti koji se mogu pozvati su: oznake, linije, poligoni, krug i trokut.

Oznake i ostali objekti pozivaju se nakon što je karta učitana. Pomoću oznaka istaknute su željene lokacije na karti. Na slici (Slika 3.3) prikazana je metoda za pozivanje oznake na karti, a na slici (Slika 3.4) prikazan je izgled dodane oznake u pregledniku.

```
var marker=new google.maps.Marker({
position:new google.maps.LatLng(45.549783,18.7044149,14),
});
```

Sl. 3.3. *Metoda za prikazivanje oznake na karti*



Sl. 3.4. *Prikaz metode za označavanje u pregledniku*

Oznakama se može mijenjati slika kojom se prikazuju, te ih je moguće animirati (Slika 3.6.). Na slici (Slika 3.5.) prikazane su metode za izmjenu slike i animiranje oznake.

```
var marker=new google.maps.Marker({
position:new google.maps.LatLng(45.549783,18.7044149,14),
icon: 'https://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Tux.svg/204px-Tux.svg.
    png',
animation:google.maps.Animation.BOUNCE
});
```

Sl. 3.5. *Metoda za izmjenu uobičajne slike oznake i dodavanje animacije*



Sl. 3.6. *Prikaz metode za označavanje u pregledniku s korisnički definiranom slikom i animacijom*

Pomoću nabrojanih metoda omogućeno je prikazivanje linija i geometrijskih oblika između određenih lokacija. Linijama i oblicima omogućena je promjena boje, širina i prozirnost crte kojom je iscrtana linija ili oblik, određvanje boje i prozirnosti ispune oblika.

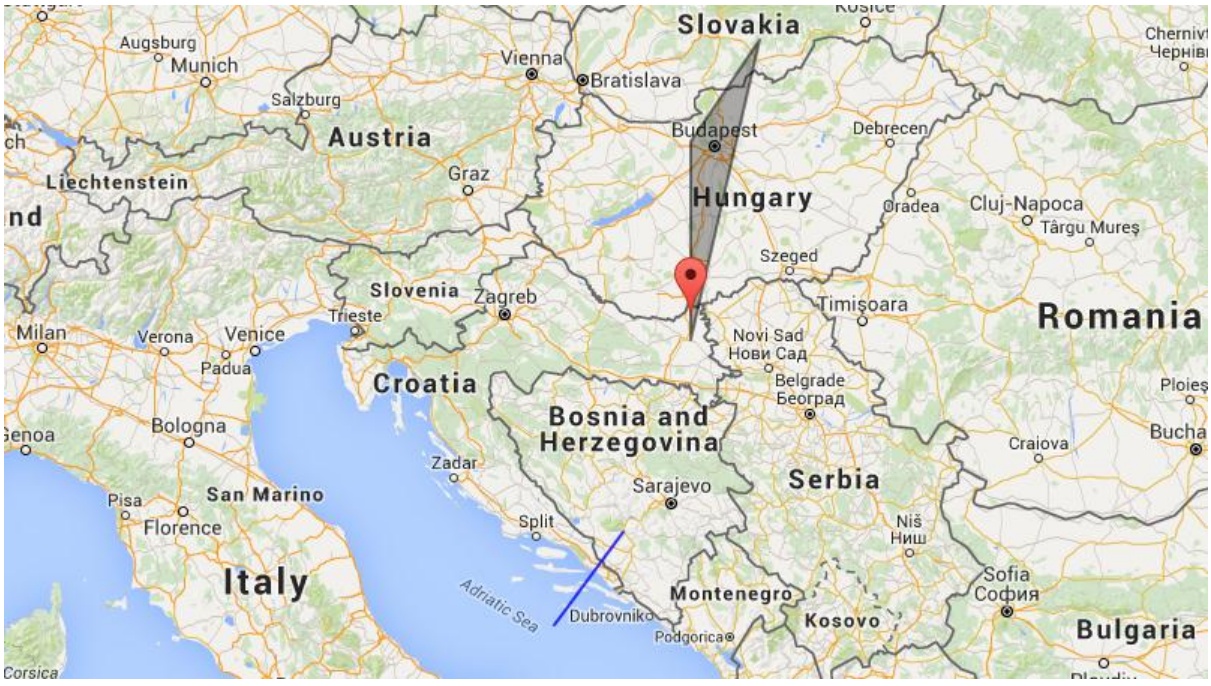
Na slici (Slika 3.7) prikazana je metoda za prikazivanje linije „Polyline“ te metoda za prikazivanje određenog oblika „Polygon“. Liniji je određno svojstvo širine „strokeWeight“, prozirnosti „strokeOpacity“ i boje „strokeColor“ te dvije koordinate koje povezuje. Poligonu je određeno svojstvo širine „strokeWeight“ i prozirnosti linije „strokeOpacity“.[18]

Na slici (Slika 3.8) je prikaz linije i oblika u pregledniku te tri koordinate koje su vrhovi iscrtanih oblika.

```
// Određivanje lokacije za ispisivanje linije
var x = new google.maps.LatLng(43.549783,17.7044149,14);
var y = new google.maps.LatLng(42.549783,16.7044149,14);
//Metoda za ispis linije
var polyline = new google.maps.Polyline({
    path:[x,y],
    strokeColor:"#0000FF",
    strokeOpacity:0.8,
    strokeWeight:2
  });
// Pozivanje linije na kartu
polyline.setMap(map);
// Određivanje lokacije za ispisivanje poligona
var a = new google.maps.LatLng(45.549783,18.7044149,14);
var b = new google.maps.LatLng(47.549783,18.7044149,14);
var c = new google.maps.LatLng(48.549783,19.7044149,14);

// Metoda za ispis poligona
var polygon =new google.maps.Polygon({
    path:[a,b,c],
    strokeOpacity:0.5,
    strokeWeight:2
  });
// Pozivanje poligona na karti
polygon.setMap(map);
```

Sl. 3.7. *Metode za prikaz linije i poligona*



Sl. 3.8. *Prikaz metode za liniju i poligon u pregledinku*

3.2. Kontrole

Google maps API karte sadrže kontrole koje korisniku omogućuju interakciju s kartom. Na standardnoj karti bit će prikazane samo osnovne kontrole. Osim njih postoje i dodatne kontrole koje se ne prikazjuju na zadanim postavkama.

Osnovne kontrole karte su :

„Zoom control“ (kontrola u visine) „+“ i „-“ su tipke za promjenu razine visine na karti nalaze u donjem desnom uglu karte, kada je karta uvećana promjena visine se može kontrolirati klizačem.

„Map type control “ (tip karte) može biti pikazan kao horizontalne tipke ili tipke u padajućem izbiorniku u gornjem lijevom uglu karte. Omogućuava korisniku prebacivanje između ponuđenih tipova karte - satelitski snimak, teren, ceste.

„Pan control“ (kontrola pomicanja) korisniku omogućava da pomoću kursora na mišu ili tipkama na tipkovnici (gore, dolje, lijevo, desno) pomiče kartu u smjerovima (istok, zapad, sjever, jug).

„Street View control“ (pogled s ulice) je ikona „Pegmana“ nalazi se u donjem desnom uglu karte, nju korisnik može povući na željeno mjesto na karti da se na tom djelu omogći pogled s ulice *street view* .

```
6  function initialize() {
7    var mapProp = {
8      center:new google.maps.LatLng(45.5542408, 18.6909462),
9      zoom:5,
10     mapTypeId:google.maps.MapTypeId.ROADMAP,
11     scaleControl:true,
12     streetViewControl:true,
13     overviewMapControl:true,
14     panControl:true,
15     zoomControl:true,
16     mapTypeControl:true,
17     rotateControl:true
18   };
19   var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);
20 }
21 google.maps.event.addDomListener(window, 'load', initialize);
```

Sl. 3.8. *Prikaz svi kontrola koje su aktivirane*



Sl. 3.9. *Prikaz karte na kojoj su sve kontorle aktivirane*



Sl. 3.9. *Prikaz karte s isključenim kontrolama*

Dodatne kontrole su :

„Scale control“ (skala) skala prikazuje omjer između udaljenosti dvije točke na karti i njihove stvarne udaljenosti.

„Rotate control“ (rotacija) prikazuje kružnu ikonu koja omogućuje kombinaciju nagiba i rotacije slike, to je kontrola za karte pod nagibom.

„Overview Map Control“ (pregledna mapa) kontrola koja osim glavne karte prikazuje još jednu umanjenu kartu koja prikazuje šire područje od onoga koje je prikazano na glavnoj karti. [19]

Pomoću varijable *MapOptions* moguće je upravljati kontrolama, a može se odrediti koje kontrole će biti prikazane na karti i kako će izgledati

3.3. Vrste karata

Google Maps API podržava četiri osnovna tipa karte. Postoje mogućnosti izrade prilagođenih karata ili mijenjanje prikaza postojećih karata, ali se korisnici uglavnom, ovisno o potrebama njihove internet stranice, odlučuju za neku od osnovnih karata.

Dostupni tipovi karit na *Google Maps API*-u su :„ROADMAP“ (auto karta) - uobičajena karta na kojoj su istaknute prometnice, „SATELITE“ (satelitska karta) - prikazuje satelitski pogled sniman *Google Earthom*, „HIBRID“ (hibridna karta) - prikazuje kombinaciju auto karte i satelitske karte, te „TERRAIN“ (terenska karta) - prikazuje fizička svojstva terena. [19]

Vrsta karte definira se na dva načina:

```
mapTypeId: google.maps.MapTypeId.ROADMAP
```

ili

```
map.setMapTypeId(google.maps.MapTypeId.ROADMAP);
```

3.4. Karte s nagibom

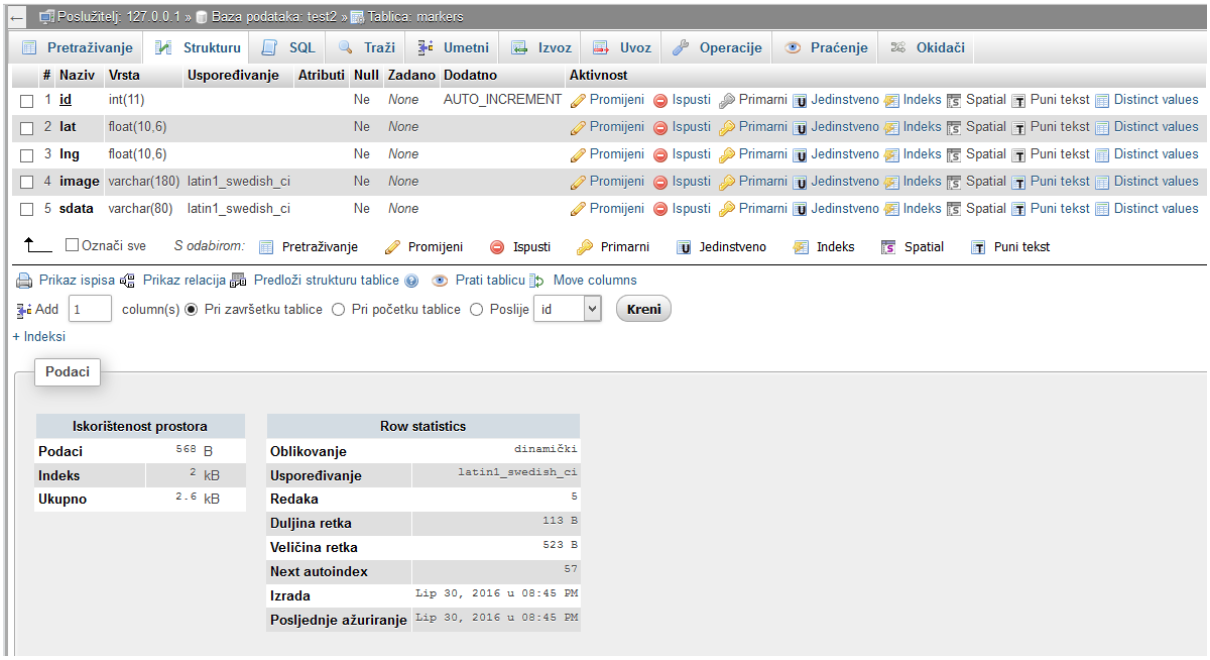
Karte s nagibom podržavaju samo tipovi karata „SATELITE“ i „HYBRID“ i dostupne su samo za neke lokacije. Kada se određena lokacija dovljno približi kontrolom „zoom“, satelitska ili hibridna karta se zamjenjuje fotografijama iz zraka snimljnenim pod kutom od 45°. Tako prikaže pogled na lokaciju sa strane za razliku od prijašnjeg nadzemnog pogleda. Kada se kotrolom „zoom“ prikaz udalji, karta se vraća u nadzemni pogled. Uključuje se s naredbom `setTilt(45)`, a isključuje se s naredbom `setTilt(0)`. [19]



Sl. 3.10. *Prikaz karte s uključenim nagibom*

4. IZRADA APLIKACIJE

Za izvršavanje *PHP* i *MySql* naredbi potreban je Web poslužitelj s instaliranim *PHP*-om i *MySql*-om. Za emuliranje Web poslužitelja koristi se *XAMPP* uz koji dolazi predinstaliran *phpMyAdmin*. *phpMyAdmin* je potreban za kreiranje baze u koju se spremamju podatci unesenih lokacija. U bazi je potrebna jedna tablica koja sadrži četiri atributa: *id, sdata, image, lat i lang*. *Id* atribut je tipa *int* duljine 11 *auto increment* i sadrži primarni ključ u koji se pohranjuje identifikacijski broj lokacije. Atributi *lat* i *lng* su tipa *float* duljine 10,6 u koje se pohranjuju koordinate odabrane lokacije, *image* atribut je tipa *varchar* duljine 80 koji se pohranjuje u lokaciju pohranjene slike, a *sdata* je atribut tipa *varchar* duljine 180 u koji se pohranju upisani podatci. Na slici (Slika 4.1.) prikazana je struktura baze podataka u *phpMyAdminu*.



Sl. 4.1. *Struktura baze podataka u phpMyAdminu*

Za unos i ispis podataka iz baze potrebno se povezati na bazu podataka. Povezivanje s bazom podataka ostvaruje se pomoću *PHP*-a, te je potrebno unijeti korisničko ime baze podataka, lozinku baze podataka, naziv baze podataka i naziv Web poslužitelju na kojem se nalazi. Na slici (Slika 4.2.) prikazano je povezivanje s bazom podataka.

```php

// database settings
$db_username = 'root';
$db_password = '';
$db_name = 'test2';
$db_host = 'localhost';

```

Sl. 4.2. *Povezivanje s bazom podataka*

Nakon unosa korisničkih podataka baze za povezivanje s bazom podataka potrebno je definirati metode za unos i brisanje podataka iz baze. Za unos i brisanje podataka u bazu dohvaća se *HTTP POST* metoda. Ovisno o tome dali se iz *HTTP POST* metode primaju podatci iz forme, podatci se prosljeđuju u bazu. U suprotnom, ako se primi zahtjev za brisanje, podatci u bazi se brišu. Na slici (Slika 4.3.) prikazane su metode za unos i brisanje podataka u bazu podataka.

```php
if($_POST) //run only if there's a post data
{
    //make sure request is comming from Ajax
    $xhr = $_SERVER['HTTP_X_REQUESTED_WITH'] == 'XMLHttpRequest';
    if (!$xhr){
        header('HTTP/1.1 500 Error: Request must come from Ajax!');
        exit();
    }

    // get marker position and split it for database
    $mLatLang   = explode(',',$_POST["latlang"]);
    $mLat       = filter_var($mLatLang[0], FILTER_VALIDATE_FLOAT);
    $mLng       = filter_var($mLatLang[1], FILTER_VALIDATE_FLOAT);

    //Delete Marker
    if(isset($_POST["del"]) && $_POST["del"]==true)
    {
        $results = $mysqli->query("DELETE FROM markers WHERE lat=$mLat AND lng=$mLng");
        if (!$results) {
          header('HTTP/1.1 500 Error: Could not delete Markers!');
          exit();
        }
        exit("Done!");
    }

    $mImage         = filter_var($_POST["image"], FILTER_SANITIZE_STRING);
    $mSdata     = filter_var($_POST["sdata"], FILTER_SANITIZE_STRING);
    //$mType        = filter_var($_POST["type"], FILTER_SANITIZE_STRING);

    $results = $mysqli->query("INSERT INTO markers (image, sdata, lat, lng) VALUES ('$mImage','$mSdata',$mLat, $mLng)");
    if (!$results) {
          header('HTTP/1.1 500 Error: Could not create marker!');
          exit();
    }

    $output = '<img src="'.$mImage.'" width="250px" height="250px"><p>'.$mSdata.'</p>';
    exit($output);
}
```

Sl. 4.3. *Metode za unos i brisanje podataka*

Podatci iz baze podataka ispisuju se pomoću *PHP*a na temelju kojih se generira *XML* dokument. Ispis podataka iz baze i generiranje *XML* dokumenta prikaznao je na slici (Slika 4.4.).

```php
//Create a new DOMDocument object
$dom = new DOMDocument("1.0");
$node = $dom->createElement("markers"); //Create new element node
$parnode = $dom->appendChild($node); //make the node show up

// Select all the rows in the markers table
$results = $mysqli->query("SELECT * FROM markers WHERE 1");
if (!$results) {
    header('HTTP/1.1 500 Error: Could not get markers!');
    exit();
}

//set document header to text/xml
header("Content-type: text/xml");

// Iterate through the rows, adding XML nodes for each
while($obj = $results->fetch_object())
{
  $node = $dom->createElement("marker");
  $newnode = $parnode->appendChild($node);
  $newnode->setAttribute("image",$obj->image);
  $newnode->setAttribute("sdata", $obj->sdata);
  $newnode->setAttribute("lat", $obj->lat);
  $newnode->setAttribute("lng", $obj->lng);
}

echo $dom->saveXML();

```

Sl. 4.4. *Ispis podataka iz baze i generiranje XML dokumenta*

Za prikaz mape s podatcima iz baze kreiran je *PHP* dokument *index.php* u kojemu je *HTML*om definirana osnovna struktura. U zaglavlju dokumenta poziva se *jQuery JavaScript* biblioteka zbog gotovih *JavaScript* funkcija koje *jQuery* ima. Osim *jQuerya* u zaglavlju dokumenta poziva se i glavna *JavaScript* datoteka i *Google Maps API*. U script tagu za pozivanje *Google Maps API*-a potrebno je unjeti i identifikacijski ključ na kraju URL-a koji poziva *Google Maps API*.

Prije kreiranja funkcije za pozivanje *Google Maps* karte potrebno je odrediti mjesto unutar dokumenta gdje će se karta prikazivati. Karta se prikazuje u *div* tagu unutar *body* taga kojem se pomoću *CSS*-a definira veličina. Na slici (Slika 4.5.) prikazan je *HTML* unutar kojeg se poziva *Google Maps* kartu.

```
<!DOCTYPE html>
<html>
<head>
<title>Google Maps API</title>
<meta charset="UTF-8">
<script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
<script type="text/javascript" src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDVDRmH7qpabQV1bpJ0pihQP2vpyHqVv64&
sensor=false"></script>
<script type="text/javascript" src="js/main.js"></script>
<style type="text/css">
/* width and height of google map */
#google_map {
    width: 100%;
    height: 100%;
    left: 0;
    position: absolute;
    top: 0;
}
.gm-style-iw div{
    overflow: hidden !important;
}
.main-title {
    position: absolute;
    width: 99%;
    font-size: 16px;
    font-family: sans-serif;
    z-index: 2;
    opacity: 0.6;
    text-transform: uppercase;
    text-align: center;
}
</style>
</head>
<body>
<div class="main-title"><h1>Pritisnite lijevu tipku miša za dodavanej nove lokacije</h1></div>
<div id="google_map"></div>
</body>
</html>
```

Sl. 4.5. *Osnovni HTML i CSS potreban za prikaz karte*

Google Maps karta se prikazuje na punoj duljini zaslona što je i definirano *CSS* svojstvima: *width:* 100% , *height:*100%, za što nam element treba biti absolutno pozicioniran na poziciji *left:*0 i *top*: 0.

Nakon što je baza kreirana i povezana pomoću *PHP*-a te su *HTML* i *CSS* definirani, potrebno je inicijalizirati kartu. Prilkom inicijalizacije karte, osim što treba unjeti postavke karte, potrebno je ispisati podatke koje nam predaje *map_process.php* datoteka te definirati događaj koji na pritisak desne tipke miša otvara formu za unos podataka. Na slici (Slika 4.6.) prikazna je funkcija za inicijalizaciju karte.

```
function map_initialize()
{
        var googleMapOptions =
        {
            center: mapCenter, // map center
            zoom: 12, //zoom level, 0 = earth view to higher value
            scaleControl: true, // enable scale control
            mapTypeId: google.maps.MapTypeId.ROADMAP // google map type
        };
        map = new google.maps.Map(document.getElementById("google_map"), googleMapOptions);
        //Load Markers from the XML File, Check (map_process.php)
        $.get("map_process.php", function (data) {
            $(data).find("marker").each(function () {
                  var image         = '<img src="'+$(this).attr('image')+'" width="250px" height="250px">';
                  var sdata     = '<p>'+ $(this).attr('sdata') +'</p>';
                  //var type          = $(this).attr('type');
                  var point     = new google.maps.LatLng(parseFloat($(this).attr('lat')),parseFloat($(this).attr('lng')));
                  create_marker(point, image, sdata, false, false, false);
            });
        });
        //Right Click to Drop a New Marker
        google.maps.event.addListener(map, 'rightclick', function(event) {
            //Edit form to be displayed with new marker
            var EditForm = '<p><div class="marker-edit">'+
            '<form action="" method="POST" name="SaveMarker" id="SaveMarker">'+
            '<label for="pImage"><span>Slika:</span><br><input type="text" name="pImage" class="save-image" placeholder="Unesite
                poveznicu na sliku" maxlength="180" /></label><br>'+
            '<label for="pDesc"><span>Podatci :</span><br><textarea name="pDesc" class="save-desc" placeholder="Unos podataka"
                maxlength="150"></textarea></label>'+
            '</form>'+
            '</div></p><button name="save-marker" class="save-marker">Pohrani</button>';
            //Drop a new Marker with our Edit Form
            create_marker(event.latLng, 'Novi unos', EditForm, true, true, true);
        });
}
```

Sl. 4.6.– *Inicijalizacija karte s podatcima iz baze i formom za unos*

U definiranom događaju za unos lokacije s podatcima poziva se funkcija za kreirenje oznake s informacijskim prozorom. U funkciji za kreiranje oznake poziva se *Google Maps* metoda za kreiranje oznake i inforamcijskog prozora te joj se predaju podatci uneseni u formi. Funkcija za kreiranje oznake sadrži i tipke za brisanje i spremanje oznake koje pozivaju funkcije za spremanje i brisanje. Funkcija za brisanje šalje *POST* metodu za brisanje koja preko funkcije u *map_process.php* briše podatke. Na slici (Slika 4.7.) prikazana je funkcija za brisanje oznake.

```
//################ Remove Marker Function ###############
function remove_marker(Marker)
{

    /* determine whether marker is draggable
    new markers are draggable and saved markers are fixed */
    if(Marker.getDraggable())
    {
        Marker.setMap(null); //just remove new marker
    }
    else
    {
        //Remove saved marker from DB and map using jQuery Ajax
        var mLatLang = Marker.getPosition().toUrlValue(); //get marker position
        var myData = {del : 'true', latlang : mLatLang}; //post variables
        $.ajax({
          type: "POST",
          url: "map_process.php",
          data: myData,
          success:function(data){
                Marker.setMap(null);
                alert(data);
            },
            error:function (xhr, ajaxOptions, thrownError){
                alert(thrownError); //throw any errors
            }
        });
    }

}
```

Sl. 4.7. *Funkcija za brisanje oznake*

Funkcija za dodavanje oznake slična je funkciji za brisanje oznake, ali umjesto *POST* metode za brisanje. Funkcija za dodavanje oznake pomoću *POST* metode prosljeđuje podatke koji se zapisuju u bazu podataka. Na slici (Slika 4.8.) prikazana je funkcija za dodavanje oznake.

```
130     //################ Save Marker Function ###############
131     function save_marker(Marker, mImage, mSdata, replaceWin)
132     {
133         //Save new marker using jQuery Ajax
134         var mLatLang = Marker.getPosition().toUrlValue(); //get marker position
135         var myData = {image : mImage, sdata : mSdata, latlang : mLatLang }; //post variables
136         console.log(replaceWin);
137         $.ajax({
138           type: "POST",
139           url: "map_process.php",
140           data: myData,
141           success:function(data){
142                 replaceWin.html(data); //replace info window with new html
143                 Marker.setDraggable(false); //set marker to fixed
144
145             },
146             error:function (xhr, ajaxOptions, thrownError){
147                 alert(thrownError); //throw any errors
148             }
149         });
```

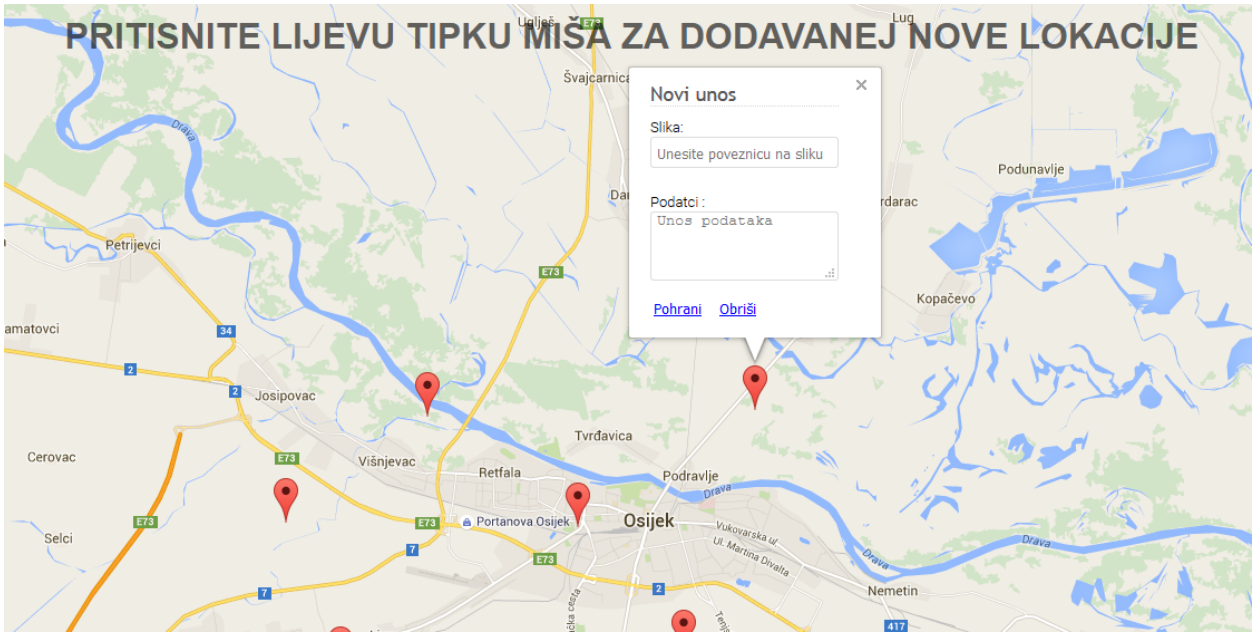
Sl. 4.8. *Funkcija za brisanje dodavanje*

Unosom adrese aplikacije u internet preglednik korisniku se učitava karta unutar preglednika (Slika 4.9.).



Sl. 4.9. *Prikaz aplikacije*

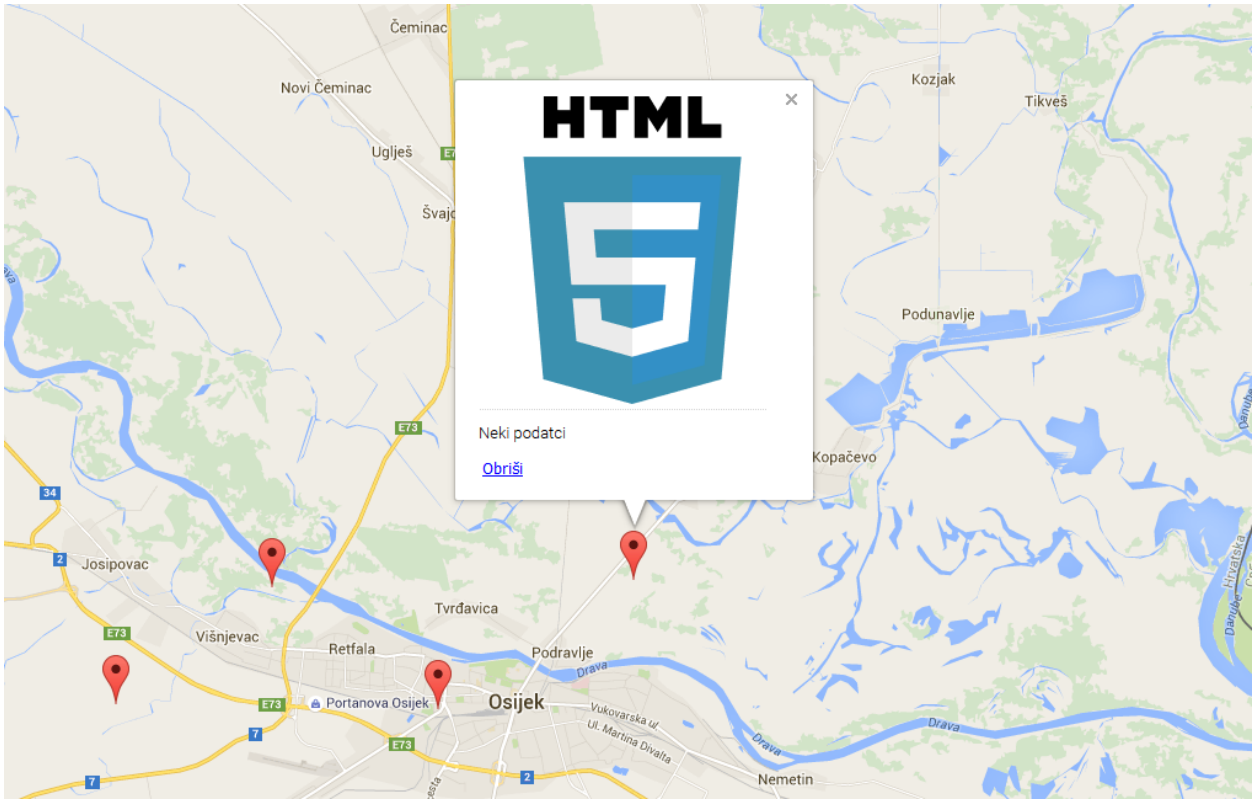
U grafičkom sučelju aplikacije korisnik pritiskom na tipku desnog klika miša kreira oznaku na novoj lokaciji te u forme za unos, unosi lokaciju slike i željene podatke (Slika 4.10.).



Sl. 4.10. *Unos nove lokacije*

Nakon unosa podataka potrebno je kliknuti na gumb *pohrani* za spremanje željene lokacije zajedno sa slikom i podatcima koji su uneseni. Kada je lokacija pohranjena klikom lijeve tipke

miša na unesenu lokaciju prikazuje se informacijski prozor u kojemu je unesena slika i podatci (Slika 4.11.). Ukoliko korisnik želi ukloniti postojeću lokaciju potrebno je klinkuti lijevom tipkom miša na *obriši* i lokacija će se obrisati s karte.



Sl. 4.11. *Prikaz unesene lokacije*

5. ZAKLJUČAK

U ovom zvršnom radu opisan je princip rada *Google Maps API* korisničkog sučelja i pomoću njega izrađena je internet aplikacija koja omogućava označavanje mjesta na karti te unos slike i podataka za označeno mjesto. Opisane su tehnologije potrebne za izradu aplkacije: *HTML, Java Script, phMyAdmin* i *PHP* .

Također se može zaključiti da se pomoću uputa i dijelova koda koje pruža *Google Maps API*-a lako može dizajnirati apikacija interaktivne karte i time omogućiti korisnicima da sami modeliraju kartu prema svojim interesima i potrebama. Primjena takavih apikacija danas je vrlo česta, jer vlasnicima internet stranica na kojima se nalazi karta lako omogućava da unesu željne informacije za one lokacije koje su važne za njihovo poslovanje i promociju. Također tako korisnicima na pregledan i jednostavan način pruža osnovne informacije o lokacijama koje su označene na karti.

LITERATURA

[1] phpMyAdmin, stranica https://en.wikipedia.org/wiki/PhpMyAdmin (pristup stranici 1.7.2016.)

[2] HTML, stranica https://hr.wikipedia.org/wiki/HTML (pristup stranici 25.6.2016.)

[3] Sublime Text, stranica https://en.wikipedia.org/wiki/Sublime_Text (pristup stranici 1.7.2016.)

[4] Brackets, stranica https://en.wikipedia.org/wiki/Brackets_(text_editor) (pristup stranici 1.7.2016.)

[5] Php Storm, stranica https://en.wikipedia.org/wiki/PhpStorm (pristup stranici 1.7.2016.)

[6] Adobe Dreamweaver, stranica https://en.wikipedia.org/wiki/Adobe_Dreamweaver (pristup stranici 1.7.2016.)

[7] KompoZer, stranica https://en.wikipedia.org/wiki/KompoZer (pristup stranici 1.7.2016.)

[8] WYSIWYG, stranica https://hr.wikipedia.org/wiki/WYSIWYG (pristup stranici 25.6.2016.)

[9] Osnove HTML-a, stranica http://www.w3schools.com/html/ (pristup stranici 25.6.2016.)

[10] Definicija CSS-a, stranica https://hr.wikipedia.org/wiki/CSS (pristup stranici 25.6.2016.)

[11] Node.js, stranica https://en.wikipedia.org/wiki/Node.js (pristup stranici 1.7.2016.)

[12] JavaScript, stranica https://en.wikibooks.org/wiki/JavaScript(pristup stranici 25.6.2016.)

[13] JavaScrip tipovi podataka, stranica http://www.w3schools.com/js/js_datatypes.asp(pristup stranici 28.6.2016.)

[14] JavaScript naredbe za kontrolu toka, stranica https://en.wikibooks.org/wiki/JavaScript(pristup stranici 28.6.2016.)

[15] Google Maps API, stranica https://en.wikipedia.org/wiki/Google_Maps#Google_Maps_API (pristup stranici 28.6.2016.)

[16] API, stranica https://en.wikipedia.org/wiki/Application_programming_interface (pristup stranici 28.6.2016.)

[17] API, stranica http://www.webopedia.com/TERM/A/API.html (pristup stranici 28.6.2016.)

[18] Google Maps objekti, stranica http://www.w3schools.com/googleapi/google_maps_overlays.asp (pristup stranici 28.6.2016.)

[19] Google Maps kontrole, vrste karata, karta s nagibom, stranica https://developers.google.com/maps/documentation/ (pristup stranici 28.6.2016.)

SAŽETAK

Definicija Web tehnologija potrebnih za izradu intreaktivne aplikacije za spremanje podataka korisnički odabrane lokacije. Opis načina funkcioniranja *Google Maps API*-a i osnovnih postavki. Izrada baze *MySQL* baze podataka korištenjem *phpMyAdmin*-a u koju se spremaju lokacije, te povezivanje baze s aplikacijom pomoću *PHP*-a. Podešavanje *Google Maps API*-a korištenjem *JavaScript*-a i uređivanje prikaza pomoću *HTML*-a i *CSS*-a.

Ključne riječi:

Google Maps API, HTML, CSS, JavaScript, PHP, phpMyAdmin

ABSTRACT

Design of interactive application for storing data using graphical interface maps

Definition of Web technologies used for development of Interactive application for storing user selected location and data. Description of *Google Maps API* and basic settings. Creating a *MySQL* database using *phpMyAdmin* that stores the location and connection of the base with an application using *PHP*. Adjusting the *Google Maps API* using *JavaScript* and editing views using *HTML* and *CSS*.

Keywords:

Google Maps API, HTML, CSS, JavaScript, PHP, phpMyAdmin

ŽIVOTOPIS

Alen Ravas rođen 9.6.1991. u Osijeku, osnovnu školu završio u Donjem Miholjcu, a srednju Elektorotehničku u Našicama. Upisao sveučilišni preddiplomski studij elektrotehnike, smjer informatikika i komuikacije, na Elektrotehničkom fakultetu u Osijeku.

X

Alen Ravas

PRILOG

Kompletan izvorni kod aplikacije opisane u poglavlju 4.

https://www.sanwebe.com/downloads/61-google-save-marker-sample

Kompletan izmjenjeni kod aplikacije

index.php

```
<!DOCTYPE html>

<html>

<head>

<title>Google Maps API</title>

<meta charset="UTF-8">

<script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>

<script                                          type="text/javascript"
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDVDRmH7qpabQV1bp
J0pihQP2vpyHqVv64&sensor=false"></script>

<script type="text/javascript" src="js/main.js"></script>

<style type="text/css">

h1.heading{padding:0px;margin:     0px     0px     10px     0px;text-
align:center;font: 18px Georgia, "Times New Roman", Times, serif;}


/* width and height of google map */

#google_map {

     height: 100%;

    width: 100%;

    left: 0;

    position: absolute;

    top: 0;

}
```

```
/* Marker Edit form */

.marker-edit label{display:block;margin-bottom: 5px;}

.marker-edit label span {width: 100px;float: left;}

.marker-edit label input, .marker-edit label select{height: 24px;}

.marker-edit label textarea{height: 60px;}

.marker-edit label input, .marker-edit label select, .marker-edit label textarea {width: 87%;margin:0px;padding-left: 5px;border: 1px solid #DDD;border-radius: 3px;}

/* Marker Info Window */

h1.marker-heading{color: #585858;margin: 0px;padding: 0px;font: 18px "Trebuchet MS", Arial;border-bottom: 1px dotted #D8D8D8;}

div.marker-info-win {max-width: 300px;margin-right: -20px;}

div.marker-info-win p{padding: 0px;margin: 10px 0px 10px 0;}

div.marker-inner-win{padding: 5px;}

button.save-marker, button.remove-marker{border: none;background: rgba(0, 0, 0, 0);color: #00F;padding: 0px;text-decoration: underline;margin-right: 10px;cursor: pointer;

}

.gm-style-iw div{ overflow: hidden !important; }

.main-title { position: absolute; width: 99%; font-size: 16px; font-family: sans-serif;z-index: 2;opacity: 0.6;text-transform: uppercase;text-align: center;}

</style>

</head>

<body>

<div class="main-title"><h1>Pritisnite lijevu tipku miša za dodavanej nove lokacije</h1></div>

<div id="google_map"></div>

</body>

</html>
```

main.js

```
$(document).ready(function() {

	var mapCenter = new google.maps.LatLng(45.5542408, 18.6909462); //Google map Coordinates

	var map;

	map_initialize(); // initialize google map

	//################ Google Map Initialize ##############

	function map_initialize()

	{

		var googleMapOptions =

		{

			center: mapCenter, // map center

			zoom: 12, //zoom level, 0 = earth view to higher value

			scaleControl: true, // enable scale control

			mapTypeId: google.maps.MapTypeId.ROADMAP // google map type

		};

		map = new google.maps.Map(document.getElementById("google_map"), googleMapOptions);

		//Load Markers from the XML File, Check (map_process.php)

		$.get("map_process.php", function (data) {

			$(data).find("marker").each(function () {

				var image = '<img src="'+$(this).attr('image')+'" width="250px" height="250px">';

				var sdata = '<p>'+ $(this).attr('sdata') +'</p>';

				//var type = $(this).attr('type');
```

```
                              var point     =                          new
google.maps.LatLng(parseFloat($(this).attr('lat')),parseFloat($(this).
attr('lng')));

                              create_marker(point, image, sdata, false,
false,       false,       "http://----PATH-TO-YOUR-WEBSITE-ICON-------
/icons/pin_blue.png");

                        });

                });

                //Right Click to Drop a New Marker

                google.maps.event.addListener(map,        'rightclick',
function(event) {

                        //Edit form to be displayed with new marker

                        var EditForm = '<p><div class="marker-edit">'+

                        '<form  action=""  method="POST"  name="SaveMarker"
id="SaveMarker">'+

                        '<label
for="pImage"><span>Slika:</span><br><input  type="text"  name="pImage"
class="save-image"    placeholder="Unesite    poveznicu    na    sliku"
maxlength="180" /></label><br>'+

                        '<label                  for="pDesc"><span>Podatci
:</span><br><textarea name="pDesc" class="save-desc" placeholder="Unos
podataka" maxlength="150"></textarea></label>'+

                        '</form>'+

                        '</div></p><button             name="save-marker"
class="save-marker">Pohrani</button>';

                        //Drop a new Marker with our Edit Form

                        create_marker(event.latLng,      'Novi      unos',
EditForm, true, true, true, "http://----PATH-TO-YOUR-WEBSITE-ICON-----
--/icons/pin_green.png");

                });

        }


        //################ Create Marker Function ##############
```

```
function create_marker(MapPos, MapTitle, MapDesc,
InfoOpenDefault, DragAble, Removable, iconPath)
{

	//new marker
	var marker = new google.maps.Marker({
		position: MapPos,
		map: map,
		draggable:DragAble,
		animation: google.maps.Animation.DROP,
		title:"Hello World!",
		//icon: iconPath
	});

	//Content structure of info Window for the Markers
	var contentString = $('<div class="marker-info-win">'+
	'<div class="marker-inner-win"><span class="info-content">'+
	'<h1 class="marker-heading">'+MapTitle+'</h1>'+
	MapDesc+
	'</span><button name="remove-marker" class="remove-marker"
title="Remove Marker">Obriši</button>'+
	'</div></div>');


	//Create an infoWindow
	var infowindow = new google.maps.InfoWindow();
	//set the content of infoWindow
```

```
            infowindow.setContent(contentString[0]);


            //Find remove button in infoWindow

            var removeBtn    =         contentString.find('button.remove-
marker')[0];

            var saveBtn     =           contentString.find('button.save-
marker')[0];


            //add click listner to remove marker button

            google.maps.event.addDomListener(removeBtn,         "click",
function(event) {

                  remove_marker(marker);

            });


            if(typeof saveBtn !== 'undefined') //continue only when save
button is present

            {


                  //add click listner to save marker button

                  google.maps.event.addDomListener(saveBtn,      "click",
function(event) {

                        var   mReplace   =   contentString.find('span.info-
content'); //html to be replaced after success

                        var   mImage   =   contentString.find('input.save-
image')[0].value; //image input field value

                        var  mDesc    =  contentString.find('textarea.save-
desc')[0].value; //description input field value

                        //var  mType   =  contentString.find('select.save-
type')[0].value; //type of marker


                        if(mImage =='' || mDesc =='')
```

```
          {
                         alert("Unesite sliku i opis !");
                    }else{
                         save_marker(marker,      mImage,      mDesc,
mReplace); //call save marker function
                    }
               });
          }

          //add click listner to save marker button
          google.maps.event.addListener(marker, 'click', function() {
                    infowindow.open(map,marker);  //  click  on  marker
opens info window
          });

          if(InfoOpenDefault) //whether info window should be open by
default
          {
            infowindow.open(map,marker);
          }
     }

     //################ Remove Marker Function ##############
     function remove_marker(Marker)
     {

          /* determine whether marker is draggable
```

```
            new markers are draggable and saved markers are fixed */
            if(Marker.getDraggable())

{
                  Marker.setMap(null); //just remove new marker
            }
            else
            {
                  //Remove saved marker from DB and map using jQuery
Ajax
                  var mLatLang = Marker.getPosition().toUrlValue();
//get marker position
                  var myData = {del : 'true', latlang : mLatLang};
//post variables
                  $.ajax({
                    type: "POST",
                    url: "map_process.php",
                    data: myData,
                    success:function(data){
                              Marker.setMap(null);
                              alert(data);
                        },
                        error:function (xhr, ajaxOptions, thrownError){
                              alert(thrownError); //throw any errors
                        }
                  });
            }
```

```
	}



//################ Save Marker Function ##############

	function save_marker(Marker, mImage, mSdata, replaceWin)

	{

		//Save new marker using jQuery Ajax

		var mLatLang = Marker.getPosition().toUrlValue(); //get marker position

		var myData = {image : mImage, sdata : mSdata, latlang : mLatLang }; //post variables

		console.log(replaceWin);

		$.ajax({

		  type: "POST",

		  url: "map_process.php",

		  data: myData,

		  success:function(data){

				replaceWin.html(data); //replace info window with new html

				Marker.setDraggable(false); //set marker to fixed

				Marker.setIcon('http://----PATH-TO-YOUR-WEBSITE-ICON-------/icons/pin_blue.png'); //replace icon

		  },

		  error:function (xhr, ajaxOptions, thrownError){

		      alert(thrownError); //throw any errors

		  }

		});

	}
```

```
      //tipsy
});
```

map_process.php

```
<?php
// database settings
$db_username = 'root';
$db_password = '';
$db_name = 'test2';
$db_host = 'localhost';
//mysqli
$mysqli = new mysqli($db_host, $db_username, $db_password, $db_name);
if (mysqli_connect_errno())
{
      header('HTTP/1.1 500 Error: Could not connect to db!');
      exit();
}
################ Save & delete markers #################
if($_POST) //run only if there's a post data
{
      //make sure request is comming from Ajax
      $xhr = $_SERVER['HTTP_X_REQUESTED_WITH'] == 'XMLHttpRequest';
      if (!$xhr){
            header('HTTP/1.1 500 Error: Request must come from Ajax!');
            exit();
      }
      // get marker position and split it for database
      $mLatLang  = explode(',',$_POST["latlang"]);
```

```
     $mLat          =                          filter_var($mLatLang[0],
FILTER_VALIDATE_FLOAT);
     $mLng          =                          filter_var($mLatLang[1],
FILTER_VALIDATE_FLOAT);


     //Delete Marker
     if(isset($_POST["del"]) && $_POST["del"]==true)
     {
          $results   =   $mysqli->query("DELETE   FROM   markers   WHERE
lat=$mLat AND lng=$mLng");
          if (!$results) {
            header('HTTP/1.1 500 Error: Could not delete Markers!');
            exit();
          }
          exit("Done!");
     }


     $mImage        =                     filter_var($_POST["image"],
FILTER_SANITIZE_STRING);
     $mSdata    = filter_var($_POST["sdata"], FILTER_SANITIZE_STRING);
     //$mType       =                       filter_var($_POST["type"],
FILTER_SANITIZE_STRING);
     $results  =  $mysqli->query("INSERT  INTO  markers  (image,  sdata,
lat, lng) VALUES ('$mImage','$mSdata',$mLat, $mLng)");
     if (!$results) {
            header('HTTP/1.1 500 Error: Could not create marker!');
            exit();
     }
     $output     =     '<img     src="'.$mImage.'"     width="250px"
height="250px"><p>'.$mSdata.'</p>';
```

```
      exit($output);
}
################ Continue generating Map XML #################
//Create a new DOMDocument object
$dom = new DOMDocument("1.0");
$node = $dom->createElement("markers"); //Create new element node
$parnode = $dom->appendChild($node); //make the node show up
// Select all the rows in the markers table
$results = $mysqli->query("SELECT * FROM markers WHERE 1");
if (!$results) {
      header('HTTP/1.1 500 Error: Could not get markers!');
      exit();
}
//set document header to text/xml
header("Content-type: text/xml");
// Iterate through the rows, adding XML nodes for each
while($obj = $results->fetch_object())
{
  $node = $dom->createElement("marker");
  $newnode = $parnode->appendChild($node);
  $newnode->setAttribute("image",$obj->image);
  $newnode->setAttribute("sdata", $obj->sdata);
  $newnode->setAttribute("lat", $obj->lat);
  $newnode->setAttribute("lng", $obj->lng);
}
echo $dom->saveXML();
```