ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU

IMPLEMENTACIJA SAJTA ZA PRIKAZ GEOLOKACIJA ZNAMENITOSTI

– DIPLOMSKI RAD –

Kandidat:

Zlatković Nikola 2006/0099

Mentor:

doc. dr Zoran Čiča

Beograd, Septembar 2015.

SADRŽAJ

SADRŽAJ **2**

1. UVOD **3**

2. KORIŠĆENI ALATI PRI IZRADI SAJTA **4**

2.1. HTML, CSS 4
2.2. JAVASCRIPT, JQUERY 5
2.3. PHP, SQL 6

3. PRIKAZ MOGUĆNOSTI SAJTA **8**

3.1. STRUKTURA SAJTA 8
3.1.1. Verifikacija korisnika *9*
3.1.2. Unos lokacije *10*
3.1.3. Prikaz lokacije znamenitosti *11*

4. OPIS REALIZACIJE SAJTA **13**

4.1. IZGLED SAJTA 13
4.2. BAZA PODATAKA SA TABELAMA 13
4.3. VERIFIKACIJA KORISNIKA 14
4.3.1. Login strana *14*
4.3.2. Registracija novih korisnika *16*
4.4. UNOS LOKACIJE 17
4.4.1. Verifikacija pomoću JavaScript jezika *18*
4.4.2. Verifikacija pomoću jQuery-a *19*
4.4.3. Verifikacija pomoću HTML jezika *19*
4.5. PRIKAZ LOKACIJE ZNAMENITOSTI 20

5. ZAKLJUČAK **22**

LITERATURA **23**

1. UVOD

U današnje doba gotovo je nezamislivo živeti, raditi i obavljati svakodnevne poslove bez nekog vida korišćenja Interneta i Internet tehnologija. Sada već možemo reći da se u poslednjih desetak godina gotovo sasvim podrazumeva da većina stanovništva jedne zemlje ima mogućnost pristupa Internetu bilo preko personalnih računara, laptopova, pametnih telefona ili sličnih uređaja. Brzina i lakoća razmene informacija, velika baza podataka iz svih oblasti, mogućnost zarade, mogućnost marketinga, društvene mreže su samo neki aspekti koji su doveli do popularizacije Interneta na globalnom nivo, tako da smo danas svedoci masovne eksploatacije ove mreže.

Jedna od ključnih oblasti, koja verovatno obuhvata i najveći broj sajtova napravljenih u komercijalne svrhe jeste turizam, tj. sajtovi posvećeni turizmu. U ovoj tezi obradiću jedan od načina prikaza lokacije znamenitosti. Prikaz lokacije biće na digitalnoj karti (Google Maps) uz prateći opis same lokacije. Takođe, posetioci sajta biće u mogućnosti da sami dodaju zanimljive lokacije, i tako omoguće drugim korisnicima da ih vide i pročitaju nešto o njima.

Sam rad možemo podeliti u nekoliko većih celina. U drugom poglavlju biće predstavljeni alati koji su potrebni za realizaciju sajta. Biće opisani HTML5,CSS, JavaScript, jQuery, PHP, MySQL. U trećem poglavlju rada biće dat detaljan pregled mogućnosti sajta, uputstvo za korišćenje sajta i potrebna objašnjenja za popunjavanje formulara za unos lokacije. U četvrtom poglavlju rada biće detaljno objašnjena realizacija sajta, praćena samim kodom i objašnjenjenjima istog. U petom poglavlju dat je kratak rezime rada i osvrt na dalji mogući razvoj i unapređenje sajta. Kompletan kod kojim je sajt realizovan je priložen u elektronskoj formi.

2. KORIŠĆENI ALATI PRI IZRADI SAJTA

U ovom poglavlju bavićemo se alatima potrebnim za izradu sajta. Svaki od korišćenih alata biće opisan u smislu funkcije, načina instalacije, implementacije itd. Ono što treba reći je da se kod sajta sastoji iz nekoliko delova. Gruba podela bila bi na deo koda koji se izvršava na server strani (*server-side*), i deo koda koji se izvršava na klijent strani (*client-side*). U ovom radu za realizaciju koda koji se izvršava na strani servera korišćen je PHP jezik (*Hypertext Preprocessor*), dok su za realizaciju funkcionalnosti na strani klijenta korišćeni JavaScript, jQuery, i HTML5. Baza podataka koja je takođe neophodna realizovana je pomoću SQL alata (MySQL).

2.1. HTML, CSS

HTML (*HyperText Markup Language*) je jezik koji nam služi za strukturni prikaz podataka, slika, teksta i drugih sadržaja na veb stranama. Najnovija verzija ovog jezika razvija se u dva pravca, preporuka je pratiti verziju iza koje stoji zvanično regulatorno telo W3C (*World Wide Web Consortium*), koje je u oktobru 2014. godine i objavilo najnoviju verziju pomenutog jezika. U bliskoj budućnosti najverovatnije će doći do ujedinjenja zvanične i nezvanične verzije jezika, tako da će potencijalni problemi oko standarda biti izbegnuti.

U HTML jeziku, strukturna podela veb strane vrši se pomoću tagova. Svaki tag može opisivati drugačiji sadržaj strane. Sam jezik nije preterano zahtevan za učenje, konceptualno je dobro rešen, takođe postoji i odlična literatura koja se može pronaći i u pisanoj i u elektronskoj formi. Postoji veliki broj alata (tj. editora) u kojima se može pisati kod, jedan od najpoznatijih a besplatnih alata jeste Notepad++. U nastavku je priložen kratki HTML kod gde se mogu uočiti osnovni tagovi:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="stil.css"/>
<title>Page Title</title>
</head>
<body>
<h1>Naslov</h1>
<div id="box">
<p id="tekst">Tekst u okviru paragrafa.</p>
</div>
</body>
</html>.
```

Ono što korisnik može videti na ekranu, tj. vidljivi deo koda nalazi se u okviru `<body>` taga. Može se primetiti da postoji tag za naslov `<h1>`, tag za paragraf `<p>` , zatim `<div>` tag koji povezuje naslov i paragraf u jednu celinu. U `<head>` tagu izmedju ostalog može se naći i tag `<link>`. U okviru ovog taga, na gore prikazani način može se pridodati CSS (*Cascading Style Sheets*) fajl koji omogućava da se izgled strane kreira i prikazuje na željeni način. CSS fajl sadrži skup potrebnih

parametara (pravila). Kompletan spisak referenci vezanih za HTML jezik može se pronaći na http://www.dev.w3.org/html5/html-author/.

CSS (*Cascading Style Sheets*) je dizajniran kako bi se rešio problem sa primenom različitih stilova veb strana, načina na koji će neki elementi biti prikazani itd... Umnogome olakšava i ubrzava proces izrade sajta, sva pravila i stilovi se nalaze na jednom mestu, dakle sama izmena ili dodavanje novih stilova može se izvesti na brz i jednostavan način. Uobičajen način korišćenja ovog alata je taj da se u posebnoj datoteci nalazi dokument sa .css ekstenzijom, koji sadrži skup svih definisanih pravila i stilova. Način na koji se ovaj dokument može priključiti kodu je prikazan u gornjem primeru HTML koda.

2.2. JavaScript, jQuery

Može se reći da je JavaScript jezik koji funkcioniše zajedno sa HTML jezikom, suštinska razlika je u tome što je HTML jezik zadužen za statički prikaz veb strane, a JavaScript jezik za dinamički prikaz veb strane. JavaScript kod se izvršava na računaru korisnika, prevashodno služi sa definisanje ponašanja veb strana prilikom njihovog korišćenja. U osnovi, ukoliko je omogućeno izvršavanje JavaScript koda, veb strane postaju interaktivne, tako da se korisnici svakodnevno susreću sa ovim programskim jezikom, a da nisu svesni toga. Svaki poznati pregledač (brauzer) ima podršku za JavaScript, a osnovna podešavanja podrazumevaju da je izvršavanje koda omogućeno.

JavaScript jezik, je podrazumevani skript jezik u okviru HTML jezika, tako da sve željene funkcionalnosti postižemo tako što u okviru HTML koda otkucamo željeni JavaScript kod u okviru `<script>` taga. Primer JavaScript koda:

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">JavaScript moze promeniti tekst HTML elementa.</p>

<button type="button" onclick="document.getElementById('demo').innerHTML =
'JavaScript kod je upravo promenio tekst!'"> Klikni! </button>.
```

U ovom primeru kada korisnik klikne na dugme koje se prikaže na ekranu, umesto inicijalnog teksta koji se nalazi u okviru paragrafa ispisaće se novi tekst. U praksi se najčešće koriste funkcije koje su napisane JavaScript jezikom, a koje su smeštene ili u nekom delu HTML koda ili u zasebnim datotekama. Kada su funkcije smeštene u okviru HTML koda, JavaScript kod pišemo u okviru `<script>` taga. Kada su funkcije smeštene u zasebnim datotekama, JavaScript kod se piše u dokumentu koji se čuva sa ektenzijom .js a koji se uključuje u HTML kod u okviru taga `<script>`. Na ovaj način se postiže velika modularnost samog programskog koda, jasniji i čistiji izgled koda što je veoma pogodno ukoliko na nekom projektu radi veći broj ljudi. Primer kada je JavaScript funkcija sačuvana u nekoj spoljašnjoj datoteci, a koju uključujemo u HTML kod je sledeći:

```
<script src="JavaScript/validation.js" type="text/javascript"></script>
```

gde se može videti da je JavaScript fukcija *validation.js* smeštena u datoteku sa nazivom JavaScript. U principu ne postoji zvanična verzija JavaScript jezika, u najvećoj meri zato što je svaka firma koja poseduje neki pregledač (brauzer) razvijala i implementirala podršku za JavaScript u isti.

Dobar izvor referenci i uputstava za ovaj skript jezik može se naći na adresi http://www.w3schools.com/js/default.asp .

jQuery je jedna od JavaScript biblioteka koja je napravljena sa ciljem da se u što je moguće većoj meri pojednostavi sam JavaScript kod. Selektori u okviru jQuery biblioteke su metode pomoću kojih pronalazimo DOM (*Document Object Model*) elemente, nad kojima kasnije sprovodimo željene akcije. Ova biblioteka se priključuje sajtu na više načina, među kojima se ističu sledeća dva: preuzimanje biblioteke sa Interneta i postavljanje na server, i pomoću GCDN (*Goodle Content Distribution Network*). Može se reći da je prvi način standardan, a da se u poslednje vreme sve češće kotristi drugi način koji je nešto bolji u smislu performansi sajta. Biblioteci se pristupa pomoću poziva funkcije jQuery, ili pomoću '$' znaka koji je zamenjuje, a samim tim i olakšava i skraćuje kod. Nakon znaka '$', sledi selektor pomoću koga se bira DOM element, zatim akcija i na kraju parametri koji se odnose na akciju koja se sprovodi nad odabranim elementima. Primer jQuery funkcije:

```
<script>
$(document).ready(function(){
    $("p").click(function(){
        $(this).hide();
    });
});
</script>
```

gde se može videti da se inicijalna funkcija nalazi u okviru druge funkcije. jQuery biblioteka bi trebalo da se aktivira nakon što se učitaju svi elementi na veb strani (u ovom slučaju selektorom selektujemo sve paragrafe), i to se praktično rešava na gore prikazani način. U današnje vreme nezamislivo je baviti se veb programiranjem, a ne koristiti ovu biblioteku koja se svakim danom sve više i više dorađuje i unapređuje.

2.3. PHP, SQL

Kao što je već pomenuto **PHP (*Hypertext Preprocessor*)** je jezik koji se izvršava na strani servera. Svoju popularnost je stekao zahvaljujući svojoj jednostavnosti, u većoj ili manjoj meri već poznatoj sintaksi baziranoj na sintaksi C jezika, kao i činjenici da je besplatan. Sa aspekta stila programiranja, može se reći da postoje dve osnovne varijante, proceduralni stil i objektno orijentisan stil. U principu treba težiti objektnom načinu programiranja koje je svakako zastupljenije, a sam način programiranja omogućava i olakšava rad na projektima gde je uključeno više ljudi. Aktuelna verzija PHP jezika trenutno je verzija PHP 5.0 (5.6.13 podverzija u trenutku pisanja ovog rada). Svaka dinamička veb strana koja u svom kodu ima neku PHP skriptu koja se izvršava treba da ima ekstenziju .php kako bi veb server znao da postoji PHP kod koji će se izvršiti. Da bi prezentovali mogućnosti PHP jezika potrebno je da se instalira odgovarajuća verzija PHP jezika, server, kao i alat kojim ćemo kreirati bazu podataka a zatim upisivati i iščitavati podatke iz iste. Instalacija svih potrebnih komponenti može se uraditi pojedinačno uz odgovarajuću konfiguraciju ili se može preuzeti neko razvojno okruženje koje je unapred konfigurisano i koje sadrži sve potrebne komponente. U zavisnosti od platforme, poznata razvojna okruženja izmedju ostalih su WAMP (WampServer) za Windows operativni sistem, LAMP za Linux platformu, a takođe postoji i varijanta razvojnog okruženja za Linux koja je potrabilna. Sam PHP kod može se kucati u bilo kom tekst editoru, a jedan od najpoznatijih a besplatnih je Notepad++.

Primer PHP koda:

```
<?php
echo "Prva skripta napisana u PHPu!";
?>
```

gde možemo videti da se naš PHP kod nalazi unutar niza znakova `<?php` koji predstavljaju početak koda i `?>` koji predstavljaju kraj koda. Ukoliko je instaliran WAMP, da bi se ova skripta izvršila, potrebno je da bude sačuvana u datoteci pod nazivom WWW (najčešće na C particiji hard diska), a samo izvršavanje zadaje se tako što se u pregledaču (brauzeru) ukuca localhost/Ime_skripte.php.

SQL (*Structured Query Language*) je jezik za pristupanje i manipulisanje bazama podataka. Pomoću ovog jezika može se pristupati sistemima kao što su MySQL, SQL Server, Access, Oracle, Sybase, DB2... Pomoću ovog jezika, tj. niza komandi može se uraditi sve što je u interesu, a vezano za bazu podataka počev od kreiranja, brisanja i modifikacije baze, tabela, zatim čitanja podataka i drugo. Za većinu akcija koje se sprovode nad bazom, tabelama u bazi, podacima itd. postoje već definisane komande (*SQL statements*). Neke od bitnijih SQL komandi su SELECT, UPDATE, DELETE, INSERT INTO itd. SQL komande će detaljnije biti objašnjene u poglavlju četiri kroz objašnjavanje koda projekta.

3. PRIKAZ MOGUĆNOSTI SAJTA

Ideja koja se javila prilikom realizacije ovog sajta bila je ta da se na jednom mestu (tj. sajtu) skupi što je moguće više zanimljivih lokacija znamenitosti i da se prikažu na mapi kako bi posetioci sajta bili detaljno informisani o lokacijama na kojima se znamenitosti nalaze. Takođe ukoliko posetioci žele, mogu uneti lokacije koje su oni već imali priliku da posete kako bi i drugim ljudima omogućili uvid u te lokacije. Za unos podataka o lokaciji znamenitosti potrebno je da korisnici budu ulogovani na sajt, tako da je omogućena i registracija novih korisnika.

3.1. Struktura sajta

U osnovi sajt se sastoji iz nekoliko stranica, spojenih u jednu celinu, pružajući korisnicima sve potrebne funkcionalnosti i informacije. Navigacija kroz sajt realizovana je pomoću padajućeg menija, gde se mogu videti sledeći elementi:

- **Naslovna –** strana sajta koja sadrži osnovne informacije o samom sajtu.
- **Unos lokacije** – strana na kojoj je moguće izvršiti unos zanimljive lokacije, s tim da je potrebno biti registrovani korisnik
- **Lokacije** – omogućava izbor znamenitosti čija će lokacija biti prikazana na posebnoj strani
- **O meni i Kontakt –** stranice posvećene autoru sajta

Izvan padajućeg menija može se videti i dugme koje korisnicima omogućava verifikaciju ili registraciju (Login dugme). Pomenuti elementi mogu se videti na slici 3.1.1. U narednim odeljcima biće detaljnije prikazane pomenute strane uz odgovarajuće slike i objašnjenja.

Slika 3.1.1 Padajući meni i Login dugme.

3.1.1. Verifikacija korisnika

U gornjem desnom uglu ekrana, nalazi se dugme **Login**, gde se nakon klika otvara forma za upis korisničkog imena i šifre, i alternativno ukoliko korisnik nije registrovan postoji mogućnost da klikom na link **Registracija** (link je lociran u samoj formi), korisnik izvrši registraciju na sajtu (slika 3.1.2).

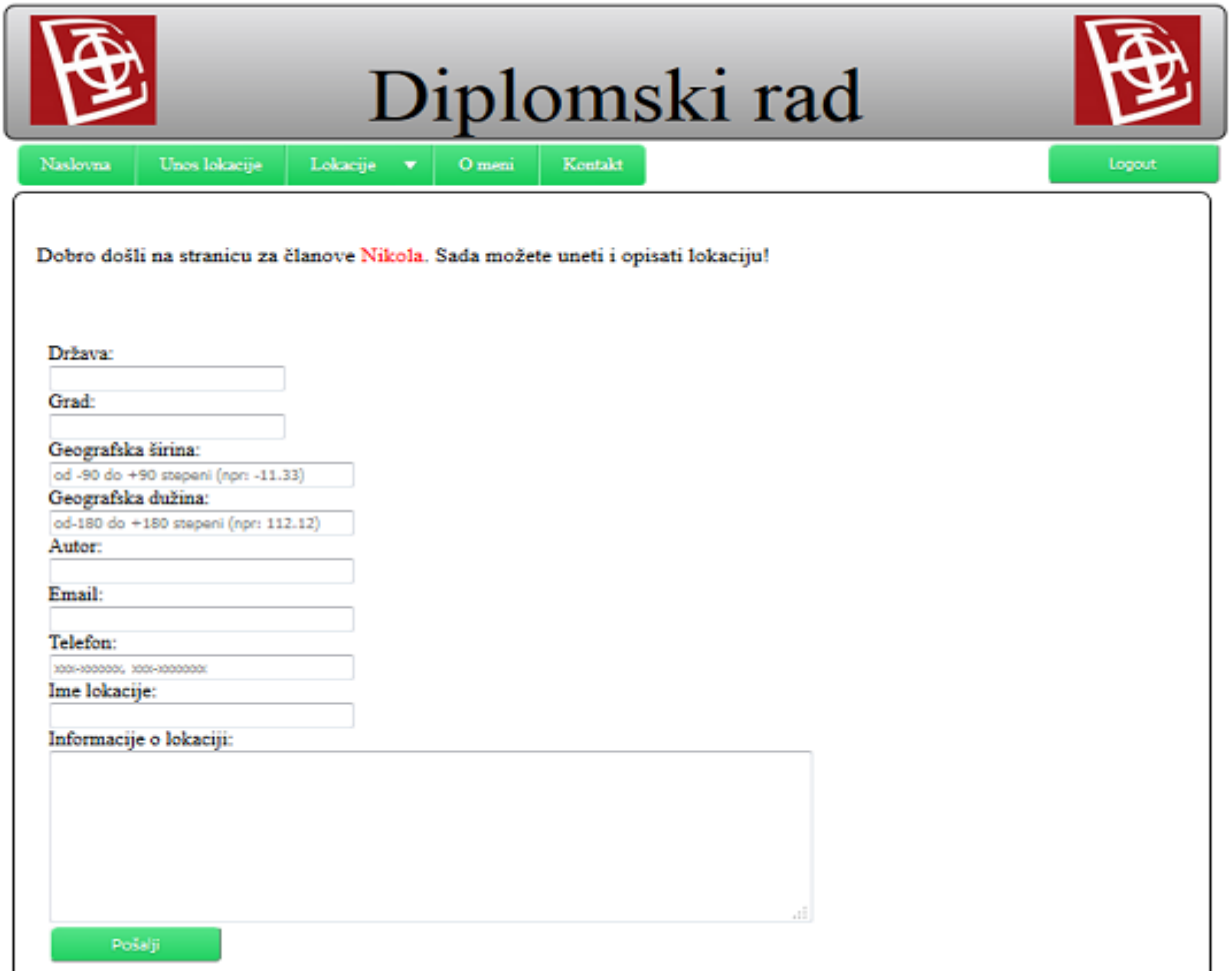
Slika 3.1.2 Login forma nakon kilka na Login dugme.

Autorizacija i/ili registracija korisnika potrebna je ukoliko oni žele da unesu nove lokacije, dok je pregled već unetih lokacija moguće vršiti i bez toga. Na slici 3.1.3 možemo videti izgled strane na kojoj se korisnici mogu registrovati.

Slika 3.1.3 Forma za registraciju korisnika.

3.1.2. Unos lokacije

Kao što je već pomenuto, nove lokacije mogu unositi samo korisnici koji imaju svoj nalog, i koji su logovani. Kada je korisnik logovan, a želi da unese lokaciju, klikom na odgovarajuću karticu padajućeg menija prikazuje se formular za unos nove lokacije (slika 3.1.4).



Slika 3.1.4 Forma za unos nove lokacije.

Prilikom popunjavanja formulara, potrebno je poštovati određena pravila tj. zadovoljiti zahtevani format podataka. Ukoliko neki format podataka nije odgovarajući, korisniku će u zavisnosti od tipa verifikacije podataka biti ispisano upozorenje na ekranu da izvrši korekciju.

3.1.3. Prikaz lokacije znamenitosti

Klikom na odgovarajuću karticu padajućeg menija, korisnici mogu izabrati neku od unetih znamenitosti. Nakon odabira željene znamenitosti, na novoj stranici prikazaće se mapa sa odgovarajućim markerom koji označava lokaciju odabrane znamenitosti, kao i kratak opis sa najbitnijim informacijama vezanim za tu znamenitost. Na sledećim slikama je prikazan odabir znamenitosti kao i mapa sa lokacijom iste (slike 3.1.5 i 3.1.6).

Slika 3.1.5 Odabir željene znamenitosti

Slika 3.1.6 Mapa sa prikazom lokacije znamenitosti.

4. OPIS REALIZACIJE SAJTA

Nakon instalacije neke od WAMP distribucija, zatim instalacije editora za kucanje koda Notepad++ stiču se svi uslovi za izradu sajta. U narednim potpoglavljima i odeljcima rada biće detaljno objašnjena realzacija sajta, praćena delovima samog koda sa objašnjenjima.

4.1. Izgled sajta

U poglavlju dva u kojem je dat pregled alata korišćenih za izradu sajta napisano je da je za statički izgled sajta zadužen HTML jezik. U CSS fajlu se nalaze dodatna podešavanja koja upotpunjuju izgled sajta, počev od korišćenih fontova, zatim stila html elemenata, korišćenih boja itd. Sama podela sajta realizovana je pomoću div elemenata (div-HTML element, moguća je podela i na bazi tabela), koji sajt dele na logične celine, a zatim je preko identifikatora html elemenata, ili na osnovu klasa kojima određeni elementi pripadaju primenjen stil koji se nalazi u **s1.css** i **s2.css** fajlovima. Već je napomenut način na koji se CSS fajl priključuje strani u potpoglavlju 2.1.

4.2. Baza podataka sa tabelama

Za svaki sajt najčešće postoji baza podataka sa odgovarajućim tabelama u kojoj se čuvaju svi podaci vezani za sajt, a koji se na odgovarajući način povezuju sa samim sajtom. Nakon instalacije WAMP distribucije (www.wampserver.com), programski jezik koji je predvidjen za rad sa bazama podataka jeste SQL (MySQL). Za ovaj projekat kreirana je jedna baza podataka (geo_lokacije), u kojoj su kreirane tri tabele (members, podaci i pomocna). Sledećim linijama koda vršimo konekciju na MySQL server:

```
$servername = "localhost";
$username = "root";
$password = "";
$conn = mysqli_connect($servername, $username, $password);.
```

Sada kada je formirana konekcija pristupamo kreiranju baze podataka i odgovarajućih tabela. U ovom pododeljku biće prikazan kod kojim se kreira baza podataka pod imenom geo_lokacije i tabela pod imenom podaci, dok se preostale dve tabele analogno kreiraju. Kod za kreiranje baze podataka je:

```
$sql = "CREATE DATABASE geo_lokacije";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully! <br> ";
} else {
    echo "Error creating database: " . mysqli_error($conn);
}.
```

Možemo videti da se istovremeno vrši upit za kreiranje baze i provera da li je ishod uspešan tj. da li je baza kreirana. Kod za kreiranje tabele :

```
$db_selected = mysqli_select_db($conn,'geo_lokacije');
$sql = "CREATE TABLE IF NOT EXISTS podaci (
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
```

```
  drzava VARCHAR(60) NOT NULL ,
  grad VARCHAR(80) NOT NULL ,
  lat FLOAT(10,6) NOT NULL ,
  lng FLOAT(10,6) NOT NULL,
  autor VARCHAR(60) NOT NULL ,
  email VARCHAR(60) NOT NULL ,
  telefon VARCHAR(60) NOT NULL ,
  ime_lokacije VARCHAR(60) NOT NULL ,
  opis VARCHAR(300) NOT NULL
  )";
```

Treba napomenuti da se prilikom kreiranja tabele, najpre mora selektovati baza podataka u kojoj će se tabele nalaziti što se postiže prvom linijom koda koji se odnosi na kreiranje tabele: `$db_selected = mysqli_select_db($conn,'geo_lokacije');`. Takođe, zgodna je praksa da se najpre proveri da li tabela pod željenim imenom već postoji, kako ne bi bila prebrisana, što se postiže drugom linijom koda: `$sql = "CREATE TABLE IF NOT EXISTS podaci (`, nakon čega se pristupa definisanju imena kolona, zatim tipova podataka koji će biti upisivani u tabelu itd. Sledeće što je potrebno uraditi da bi tabela bila kreirana je izvršiti upit (query):

```
if (mysqli_query($conn, $sql)) {
    echo "Table podaci created successfully! <br> ";
} else {
    echo "Error creating table: " . mysqli_error($conn);
};.
```

Na gore prikazani način, izvršavamo upit i proveravamo da li je upit uspešno izvršen. Važno je napomenuti da na svim PHP stranama koje koriste podatke iz baze, potrebno je imati kod koji će vršiti konekciju na server, zatim konekciju ka bazi, sa odgovarajućim parametrima u smisu imena servera, imena baze, zatim korisničkog imena i šifre. Najčešća praksa je da se svi podaci vezani za pomenutu konekciju ka bazi nalaze u zasebnom PHP fajlu (u ovom projektu podaci vezani za konekciju smešteni su u fajlu **dblogin.php**), koji se kanije komandama include ili require priključi odgovarajućim stranicama.

4.3. Verifikacija korisnika

Pod verifikacijom se podrazumeva proces u kom korisnici koji su registrovani vrše logovanje na sajt, ili eventualno ukoliko nemaju kreiran nalog vrše registraciju naloga, a zatim logovanje. Privilegija koju imaju logovani korisnici sastoji se u tome što im je omogućeno da i sami unose podatke vezane za lokaciju znamenitosti koju bi želeli da predstave drugim korisnicima. U narednim odeljcima biće predstavljen način realizacije strana za logovanje i kreiranje naloga tj.registraciju, praćen odgovarajućim kodom.

4.3.1. Login strana

Na slici 3.1.2 može se videti izgled login forme, koji je postignut zahvaljujući stilu koji se nalazi u CSS fajlu, kao i zahvaljujući HTML kodu. Odgovarajući HTML kod forme je:

```
<div id="prikaz">
   <div class="login">
       <div class="login_side">
           <div class="login_inside">
               <h2>LOGIN</h2>
                   <a href="Registracija.php">Registracija</a>
           </div>
       </div>
       <form id="login_fields" method="post" action="checklogin.php">
          <label>Login Name</label>
               <input class="user_pass" type="text" name="username" />
          <label>Password</label>
          <input class="user_pass" type="Password" name="password"/>
          <input class="submit" name="submit" type="submit" value="Login"/>
       </form>
   </div>
</div>.
```

Prvi div element (`<div id="prikaz">`) koristimo da bi pomoću jQuery biblioteke i funkcija omogućili da se login forma pojavi ili nestane ukoliko se klikne na login dugme ili negde van forme, respektivno. Prikaz koda koji omogućava ovu funkcionalnost:

```
$(document).mouseup(function (e)
{
    var container = new Array();
    container.push($('#prikaz'));

    $.each(container, function(key, value) {
        if (!$(value).is(e.target) // ako meta klika nije conteiner ili...
            && $(value).has(e.target).length === 0) // ... potomak conteinera,
onda sakrivamo konteiner.
        {
            $(value).hide();
        }
    });
});
```

Nakon što korisnik upiše korisničko ime i šifru i potvrdi unos, aktivira se provera unetih podataka. Ukoliko su podaci tačni vrši se redirekcija ka stranici za unos nove lokacije. Ukoliko su uneti podaci neispravni, ispisuje poruka da su podaci pogrešni. Kod kojim se regulišu ove funkcionalnosti smešten je u fajlu **checklogin.php**. Podrazumeva se da je u fajlu sadržan kod za konekciju na server, sa odgovarajućim parametrima. U nastavku odeljka biće priloženi delovi koda, sa objašnjenjima.

```
session_start();
include ("php_pomocni_fajlovi/dblogin.php");
$conn = mysqli_connect($servername, $username, $password, $dbname);

      if (isset($_POST['username']) and isset($_POST['password'])){

                  $username = $_POST['username'];
                  $password = $_POST['password'];
                  $pass=md5($password);.
```

Preko superglobalne promenljive $_POST uzimaju se podaci koje korisnik unosi u polja predviđena za unos korisničkog imena i šifre. Dalje proveravamo da li u tabeli **members** postoji egzaktno to korisničko ime i šifra. Promenljivoj `$row` se dodaju podaci u vidu asocijativnog niza. `$i` je brojač koji kada se izvrši linija koda `$i = mysqli_num_rows($result);`, uzima numeričku vrednost broja redova koji su dodati promenljivoj `$row`.

```
$sql = "SELECT * FROM members WHERE username='$username' AND password='$pass'";
            $result = mysqli_query($conn, $sql);
            $row = mysqli_fetch_assoc($result);
            $i = mysqli_num_rows($result);
                  if ($i == 1){
                        $_SESSION['username'] = $username;
                  }else {
                        echo "Netačni Login podaci.";
                  }
```

Pošto je jedino ispravno da u tabeli postoji samo jedan par korisničkog imena i šifre vrši se provera da li promenljiva `$i` ima vrednost jedan i ukoliko je to tačno formira se sesija. Dok god sesija postoji, znači da je dati korisnik logovan, samim tim on ima privilegiju da vrši unos novih lokacija. Treba napomenuti da ukoliko se podaci koriste i prenose preko sesija potrebno je da pre svake instance u kojoj će se sesija koristiti staviti funkciju **`session_start();`**, kojom se sesija započinje.

4.3.2. Registracija novih korisnika

Forma koja se popunjava podacima vezanim za registraciju novih korisnika data je na slici 3.1.3. HTML kod ove forme je:

```
 <div id="kontejner_registracija">
    <div class="registracija">
            <div class="registracija_side">
                  <div class="registracija_inside">
                        <br><br><br><h2>Registracija</h2>

                  </div>
            </div>

<form id="registracija_desno" method="post" action="Registracija.php">
  <label>Login Name</label>
  <input class="registracija_polja" type="text" name="username" /><br><br>
  <label>Password</label>
  <input class="registracija_polja" type="Password" name="password"  />

  <input class="submit_registracija" name="submit" type="submit"
  value="Potvrdi"/>
</form>

   </div>
</div>.
```

Ono što je bitno obezbediti, a tiče se funkcionalnosti, jeste to da sva korisnička imena budu jedinstvena. PHP kod koji obezbeđuje sve potrebne funkcionalnosti vezane za registraciju novih korisnika nalazi se u fajlu pod nazivom **Registracija.php**. U nastavku biće priloženi najznačajniji delovi koda sa odgovarajućim objašnjenjima, podrazumeva se da je obezbeđena konkcija ka serveru.

```
$username = mysqli_real_escape_string($conn, $_POST['username']);
$password = md5(mysqli_real_escape_string($conn, $_POST['password']));
$sql="SELECT * FROM members WHERE username = '$_POST[username]'";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
      echo "<div class='registracija_poruka'>
      <p>*Izabrano korisničko ime već postoji, molimo Vas pokušajte
      ponovo.</p></div>";
}     else {
            if(isset($_POST["submit"])){
              $sql1="INSERT into members (username, password) VALUES
              ('$username', '$password')";
                  if (mysqli_query($conn, $sql1)) {
                    echo "<div class='registracija_poruka'><p>Uspešno ste se
                    registrovali, možete se ulogovati!</p><div>";
                  } else {
                        echo "Error: " . $sql1 . "<br>" . mysqli_error($conn);
                    }
            };
      };
```

Najpre treba istaći da se funkcija *mysqli_real_escape_string()* koristi kako bi se sprečilo da se potencijalni maliciozni kod doda SQL upitu. Suština je u tome da ova funkcija doda jedan karakter pre potencijalno opasnog koda i taj karakter služi praktično za izlaz, tako da ne postoji mogućnost da maliciozan kod dospe u bazu podataka i slično. Kao što se može videti, podaci se selektuju iz tabele **members**, a zatim se vrši SQL upit kojim podatke smeštamo u promenljivu `$result`, ukoliko odgovarajući podaci postoje. Dalje se vrši provera da li promenljiva `$result` sadrži podatke. Ukoliko se utvrdi da `$result` sadrži podatake, to znači da korisničko ime već postoji i zahteva se pokušaj novog unosa, u suprotnom prelazi se na **`else`** granu koda u kojoj ubacujemo podatke (korisničko ime i šifru) u bazu podataka.

4.4. Unos lokacije

Na slici 3.1.4 može se videti forma pomoću koje je logovanim korisnicima omogućen unos lokacija znamenitosti. U narednim odeljcima ovog potpoglavlja biće prikazan HTML i PHP kod najznačajnijih elemenata sa odgovarajućim objasnjenjima. Među najbitnijim delovima koda svakako spadaju delovi kojima je realizovana funkcionalnost elemenata kod kojih postoji verifikacija. Od korisnika se zahteva da unos podataka bude određenog formata. Polja u kojima se zahteva verifikacija se odnose na polja gde se unose geografska dužina i širina, zatim email i telefon korisnika. Za geografsku širinu je zahtevano da unos bude izmedju -90 i +90 stepeni, za geografsku dužinu unos mora biti izmedju -180 i +180 stepeni, takođe unos ne mora biti ceo broj. Kad je u pitanju telefon podržana su dva formata brojeva: xxx-xxxxxx i xxx-xxxxxxx. Da bi obezbedili verifikaciju potrebno je da imamo svojevrsni obrazac, niz znakova, koji će obuhvatiti sve moguće korektne unose, a sa kojim ćemo uporediti unos korisnika i proveriti njegovu ispravnost. Obrasci za navedena polja su:

-geografskadužina: `/^(\+?-?)((?:1[0-7]|[1-9])?\d(?:\.\d{1,})?|180(?:\.0{1,})?)$/`

-geografska širina: `/^(\+?-?)([1-8]?\d(?:\.\d{1,})?|90(?:\.0{1,6})?)$/`

-telefon: `/^\d{3}-\d{6}$|^\d{3}-\d{7}$/`

-email: `/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/`.

Verifikacija formulara je odrađena na tri načina koja će biti predstavljena u narednim odeljcima.

4.4.1. Verifikacija pomoću JavaScript jezika

U prethodnom delu predstavljeni su odgovarajući obrasci koji će nam služiti za proveru ispravnosti unetih podataka. JavaScript kod koji obezbeđuje verifikaciju nalazi se u fajlu **validation.js** (u folderu JavaScript), a on se priključuje strani **Unos lokacije.php** (strana koja se bavi unosom novih lokacija) na sledeći način:

```
<script src="JavaScript/validation.js" type="text/javascript"></script>.
```

U ovom dokumentu je JavaScript funkcija koja će se izvršiti nakon što korisnik potvrdi svoj unos. HTML kod forme za unos lokacije je:

```
<form  name="f1"   method="POST" onsubmit="return val();">
      Država:<br>
      <input      name="drzava" type="text" value="" size="30" ><br>
      Grad:<br>
      <input      name="grad" type="text" value="" size="30" ><br>
      Geografska širina:<br>
      <input      name="lat" type="text" value="" placeholder="od -90 do +90
                  stepeni (npr: -11.33) "size="40"  ><br>
      Geografska dužina:<br>
      <input      name="long" type="text" value=""placeholder="od-180 do +180
      stepeni (npr: 112.12)"  size="40"  ><br>
      Autor:<br>
      <input      name="autor" type="text" value="" size="40"  ><br>
      Email:<br>
      <input      name="email" type="text" value="" size="40"  ><br>
      Telefon:<br>
      <input      name="telefon" type="text" value="" placeholder="xxx-xxxxxx,
            xxx-xxxxxxx"size="40" ><br>
      Ime lokacije:<br>
      <input      name="ime_lokacije" type="text" value="" size="40" ><br>
      Informacije o lokaciji:<br>
      <textarea   name="opis" rows="7" cols="70"></textarea><br >
      <input id="submit_button" name="submit" type="submit" value="Pošalji"  >
</form>
```

gde možemo videti da je atributom `onsubmit="return val();` obezbeđeno da se funkcija izvrši nakon što korisnik potvrdi unos. Kod JavaScript funkcije koja obezbeđuje verifikaciju je priložen u nastavku s tim što će biti naveden samo kod vezan za unos geografske širine a za ostala polja je analogno.

```
      if(f1.lat.value==""){
            alert("Molim Vas, unesite geografsku širinu!");
            f1.lat.focus();
            return false;
      };
      var latitude = /^(\+?-?)([1-8]?\d(?:\.\d{1,})?|90(?:\.0{1,6})?)$/;
      if(latitude.test(f1.lat.value) == false) {
            alert ("Format geografske širine nije dobar!\nKoordinata mora biti
izmedju -90 i +90 stepeni!");
            f1.lat.focus();
            return false;
      };
```

Možemo videti da se pored funkcionalnosti verifikacije takođe vrši i provera da li je zahtevano polje popunjeno.

4.4.2. Verifikacija pomoću jQuery-a

Prethodno je objašnjeno da je jQuery jedna od biblioteka JavaScript jezika. Sam način verifikacije forme za unos lokacije se ne razlikuje preterano od verifikacije JavaScript jezikom, osim u tome što je nešto drugačija sintaksa tj. selektovanje elemenata nad kojima se sprovode akcije. Naravno, da bi jQuery biblioteka mogla da se koristi potrebno je priključiti je kodu na neki od opisanih načina u potpoglavlju 2.2. Kod koji omogućava tražene funkcionalnosti nalazi se u fajlu **val_jquery.js** (u folderu JavaScript). U nastavku biće priložen i objašnjen kod za proveru ispravnosti unetih podataka za geografsku širinu, a za ostala polja je analogno tome.

Deo koda koji omogućava da određena polja budu zahtevana, tj. da moraju biti popunjena (u uglastim zagradama su vrednosti atributa `name` html forme za odgovarajuća polja):

```
required = ["drzava", "grad", "lat","long", "autor", "email", "telefon", "ime_lokacije", "opis"];.
```

FOR petljom prolazi se kroz sva polja čija se imena nalaze u uglastim zagradama u prethodno prikazanom kodu i vrši se provera uslova: da li polja imaju neku vrednost ili vrednost `emptyerror` identiteta. Ako je uslov ispunjen, dodaje im se klasa `needsfilled` (parametri su definisani u CSS fajlu) tj. jasno se ističu ova polja i u njih se upisuje poruka koja je smeštena u identitet `emptyerror` i koju korisnik vidi. Ukoliko uslov nije ispunjen, dodela klase se poništava.

```
lat = $("#lat")

errornotice = $("#error");

emptyerror = "Popunite ovo polje.";

laterror= "Molimo vas unesite ispravnu vrednost!";

for (i=0;i<required.length;i++) {
      var input = $('#'+required[i]);
      if ((input.val() === "") || (input.val() == emptyerror)) {
            input.addClass("needsfilled");
            input.val(emptyerror);
            errornotice.fadeIn(750);
      } else {
              input.removeClass("needsfilled");
            }
}
```

Kod kojim se vrši provera ispravnosti formata unetih podataka je:

```
      if (!/^(\+?-?)([1-8]?\d(?:\.\d{1,})?|90(?:\.0{1,6})?)$/.test(lat.val())) {
                  lat.addClass("needsfilled");
                  lat.val(laterror);
      }
```

gde se ukoliko je uslov ispunjen (tj. ukoliko obrazac ne obuhvata unete podatke) dodaje klasa odgovarajućem polju i upisuje poruka smeštena u identitet `laterror`.

4.4.3. Verifikacija pomoću HTML jezika

Aktuelna verzija HTML jezika jeste HTML5 verzija. Niz novina koje donosi ova verzija između ostalog odnosi se i na verifikaciju podataka koje korisnici unose u odgovarajuća polja forme predviđene za unos podataka. Celokupna verifikacija na veoma jednostavan način sprovodi se dodavanjem odgovarajućih atributa elementima (poljima) forme. Treba napomenuti da u ovoj verziji HTML (HTML5) jezika, postoje unapred definisani tipovi podataka koji se unose npr. za email, što se može videti u sledećoj liniji koda:

```
<input id='email' name="email" type="email" value="" size="40"  required><br>,
```

čime je verifikacija ovakvih i slični polja veoma olakšana. U nastavku odeljka biće priložen HTML kod dela forme za unos podataka koji se odnosi na verifikaciju geografske širine, dok se za ostala polja verifikacija vrši po analogiji:

```
Geografska širina:<br>
<input id='lat' name="lat" type="text" value="" placeholder="od -90 do +90
stepeni (npr:-11.33)" size="40"

pattern="(\+?-?)([1-8]?\d(?:\.\d{1,})?|90(?:\.0{1,6})?)" required> <br>.
```

Atribut `placeholder` definiše kratko uputstvo koje opisuje očekivanu vrednost u datom polju za unos podataka. Atribut `required` označava da je polje obavezno tj. da se mora uneti vrednost. Atribut `pattern` sadrži obrazac sa kojim se upoređuju uneti podaci i vrši provera ispravnosti unetih podataka. Može se zaključiti da je ovaj način verifikacije (pomoću HTML5 jezika) najjednostavniji, i najbolji što se tiče performansi.

4.5. Prikaz lokacije znamenitosti

Kada se govori o prikazu lokacija na elektronskoj mapi prva asocijacija jeste gugl servis *Google Maps*. Takođe, u najnovijoj verziji HTML jezika postoji svojevrsna implementacija elektronske mape koja je vezana za lokaciju samog korisnika (ukoliko korisnik dozvoli prikaz lokacije). U ovom radu implementacija elektronske mape i sam prikaz lokacije bazira se na Google Maps JavaScript API (*Application Program Interface*) servisu. Da bi API za elektronske mape mogao da se koristi potrebno je priključiti ga željenoj stranici na sličan način kao što se priključuje npr. jQuery biblioteka. U nastavku ovog potpoglavlja biće detaljno predstavljen kod koji omogućava prikaz lokacije na elektronskoj mapi, sa adekvatnim objašnjenjim.

```
<script>
var myCenter=new google.maps.LatLng(<?php include
("php_pomocni_fajlovi/koordinate.php")?>);

function initialize()
{
var mapProp = {
  center:myCenter,
  zoom:15,
  mapTypeId:google.maps.MapTypeId.ROADMAP
  };
var map=new google.maps.Map(document.getElementById("mapa"),mapProp);
var marker=new google.maps.Marker({
  position:myCenter,
  });
marker.setMap(map);
}
google.maps.event.addDomListener(window, 'load', initialize);
</script>.
```

Priloženi kod, koji se bazira na objektno orijentisanoj paradigmi, omogućava tražene funkcionalnosti. Promenljiva `myCenter` služi za centriranje mape, dakle mora sadržati koordinate centra mape. `LatLng` je objekat kome putem PHP koda prosledjujemo tražene koordinate. Kod skripte koja vrši prosleđivanje koordinata nalazi se u fajlu **koordinate.php** (u folderu php_pomocni_fajlovi) je:

```php
<?php
	$lat=$_GET['lat'];
	$long=$_GET['long'];
	echo $lat . "," . $long;
?>
```
,
a same koordinate se ovoj skripti prosleđuju putem URL adrese. Promenljiva `mapProp` sadrži osobine mape vezane za poziciju centra mape, uvećanje, tip mape itd. Kreira se nova mapa (novi objekat) , potrebni parametri su html element u kome će mapa biti smeštena a koji biva selektovan pomoću `getElementById` metode, i osobine mape smeštene u promenljivoj `mapProp`. Nakon kreiranja mape može se kreirati marker (novi objekat) koji će biti postavljen na lokaciji koju definišu koordinate smeštene u promenljivoj `myCenter`. Metod `setMap()` služi da bi marker dodali tj. postavili na mapu. Poslednja linija priloženog koda :

`google.maps.event.addDomListener(window, 'load', initialize);`,

u stvari pokazuje kako se definiše trenutak kada će se JavaScript kod izvršiti. Sintaksa metode `addDomListener` je :

`addDomListener(instance:Object, eventName:string, handler:Function)`. Dakle nakon što se se završi sa učitavanjem, izvršava se zadata funkcija. Sve reference vezane za elektronske mape mogu se pronaći na https://developers.google.com/.

5. ZAKLJUČAK

U prethodnim poglavljima prikazana je realizacija sajta za prikaz geolokacija znamenitosti. Objašnjeni su korišćeni alati pomoću kojih je sajt realizovan, priloženi su delovi koda sa odgovarajućim objašnjenjima i opisane su mogućnosti sajta. Takođe, prikazana su tri načina verifikacije forme za unos podataka pomoću JavaScript jezika, jQuery biblioteke i HTML5 jezika. Kada se uporede tri predstavljena načina verifikacije, može se zaključiti da je verifikacija pomoću HTML5 jezika najjednostavnija za realizaciju i najbolja po pitanju performansi.

Dalji razvoj sajta može da teče u smeru razvoja sajta koji će biti svojevrsna navigacija za pronalaženje lokacija znamenitosti, koje su najbliže posetiocima sajta u trenutku posete. Optimizacija sajta za Android i iOS platforme takođe može biti predmet daljeg razvoja. Što se samog prikaza lokacije tiče, mogu se implementirati novi tipovi mapa za prikaz lokacije. Baza podataka unetih lokacija se takođe može znatno proširiti, kako bi se korisnicima omogućio veći izbor lokacija znametnitosti.

LITERATURA

[1] R.Nixon, "Learning PHP, MySQL & JavaScript with jQuery, CSS and HTML5", 2014

[2] https://developers.google.com/maps/documentation/javascript/

[3] *www.w3schools.com*