Gradovi u Hrvatskoj

Ivešić, Toni

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni preddiplomski studij računarstva

GRADOVI U HRVATSKOJ

Završni rad

Toni Ivešić

Osijek, 2019.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 05.09.2019.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Toni Ivešić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3779, 18.10.2018.
OIB studenta:	36742453142
Mentor:	Izv. prof. dr. sc. Alfonzo Baumgartner
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Gradovi u Hrvatskoj
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	05.09.2019.
Datum potvrde ocjene Odbora:	11.09.2019.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 11.09.2019.

Ime i prezime studenta:	Toni Ivešić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3779, 18.10.2018.
Ephorus podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: Gradovi u Hrvatskoj**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Alfonzo Baumgartner

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Suvremeni svijet se sve više i više oslanja na postojeće i otkrivanje novih tehnologija i kao što svi znamo tehnologija na dnevnoj bazi napreduje sve više i očekuje se da će uskoro smanjiti odnosno u potpunosti zamijeniti ljudski rad i čovjekovo opažanje u potpunosti.

Svakodnevno se u dosta slučajeva susrećemo s bazama podataka u raznim područjima, to je već postalo i neopaženo na svakome koraku. Nije uvijek potrebno stvaranje novih baza podataka, postoji već puno kreiranih baza podataka od strane drugih korisnika koje su omogućene za korištenje i rad s njima i drugim korisnicima što naravno nije uvijek slučaj. Važno je u izradi baze podataka da se ona ukoliko je moguće napravi bez grešaka jer je nekad ispravljanje grešaka neisplativo, što financijski, što vremenski. Izrada nove baze podataka se dijeli na 5 različitih faza izrade, a to su: analiza potreba, modeliranje podataka, implementacija, testiranje i održavanje.

Cilj ovog završnog rada je uz izrađenu bazu podataka svih gradova u Hrvatskoj, točnije 127 gradova postoji u Republici Hrvatskoj, pokazati njihovu točnu lokaciju na iscrtanoj mapi na našoj web stranici. Web stranicu smo napravili u programu PHPStorm gdje smo iz baze podataka učitali podatke i napravili ono što se od nas tražilo opisom teme završnog rada.

Opis aplikacije i njeno djelovanje opisani su u nekoliko poglavlja, čiji se sadržaj temelji na opisu zadatka koji moramo obaviti, zatim slijedi opis korištenih tehnologija, nakon toga ide opis što se radilo u kojem koraku i kako je napravljena aplikacija, poslije opisa po koracima pravljenja aplikacije se nalaze slike izgleda aplikacije i kako se koristi aplikacija te naposljetku zaključak.

1.1. Opis zadatka završnog rada

Zadatak je napraviti bazu podataka sa svim gradovima u Hrvatskoj. Uz naravno imena gradova, potreban nam je i njihov geo-položaj. Po želji možemo dodati i druge podatke kao što su: županija, država itd. Iz te baze podataka zadatak nam je omogućiti generiranje slike na kojoj će se iscrtati točke s imenima gradova s obzirom na njihovu geo-lokaciju.

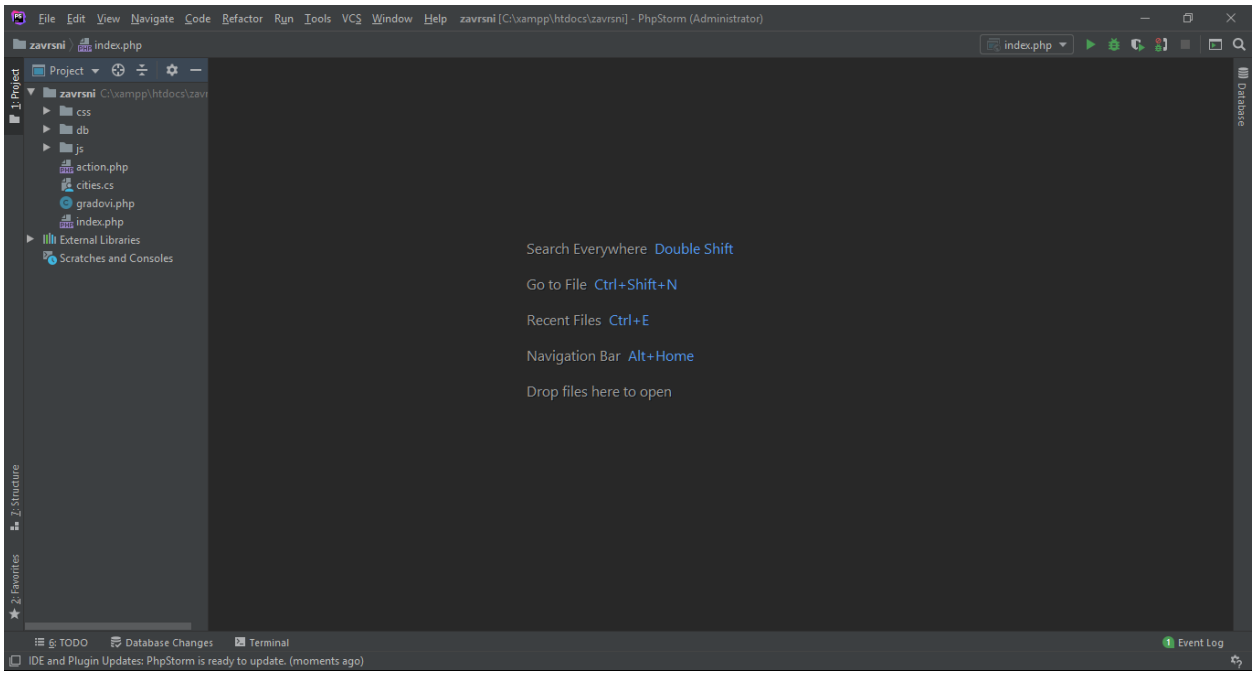
2. OPIS KORIŠTENIH TEHNOLOGIJA

Kako bi shvatili proces nastajanja web aplikacije u kojoj na iscrtanoj karti Hrvatske se pokazuju gradovi Hrvatske označeni markerima potrebno je opisati tehnologije koje se koriste kako bi program uspješno radio.

U ovome poglavlju će biti opisan PhpStorm te njegove najvažnije značajke, zatim XAMPP i što postižemo njegovom uporabom, slijedi MySQL i njegovo korištenje, a također će biti opisani i HTML, PHP, SQL i JavaScript koji su korišteni u izrazi programa.

2.1. PhpStorm

PhpStorm je među najboljima IDE (engl. Integrated Development Enviroment) za PHP izrađen od strane JetBrains-a, koji nam uvelike olakšava i pomaže u radu s kompletiranjem koda, refractoring-om koda, integracijom version control-a (GIT), integracijom WordPressa te Command Prompt-a, također olakšava i integraciju SASS kompajler i Typscripte transpilera, a isto tako i radi debuging. [1]



Sl. 2.1. *Sučelje PhpStorm integriranog razvojnog okruženja*

Neke od najvažnijih značajki PhpStorm-a su:

- Uređivač koda koji je u potpunosti opremljen za PHP
- Usmjeravanje na prednje jezike pomoću dodatnog urednika

- Opcije koje omogućuju jednostavno testiranje i otklanjanje pogrešaka i nude izvođenje na udaljenim poslužiteljima ili lokalno. [2]

Prednost PhpStorma se očituje i u automatskom ažuriranju na novije verzije kako bi bile uključene sve značajke koje korisnici očekuju u novoj verziji. Sa svakom novom inačicom programa dolaze neke nove promjene u sintaksi i etiketi što je vrlo važno kod završavanja koda i njegovog oblikovanja. U PhpStormu također postoji mogućnost odabira stila oblikovanja po želji koja dolazi sa sustavom oblikovanja koda. Program se brine za osiguranje usklađivanja svih skripti u vašem kodu. Kako bi olakšalo korisniku, dovršavanje koda završava klase, ključne riječi i nazive varijabli potpuno automatski. Također u svrsi olakšavanja stvaranja sustava, PhpStorm sadrži urednik za JavaScript, HTML i CSS. [9]

2.2. Xampp

XAMPP je besplatan serverski paket (Open Source) pomoću kojeg se instalira Apache server na računala koja podržavaju operacijske sustave Linux, Window ili OS X. Prije svega, XAMPP je osmišljen za korištenje, odnosno upotrebu u lokalnoj mreži s osnovnom svrhom omogućavanja programerima da izrade server na kojem će moći isprobati svoje web stranice, skripte. Jednostavnije, ukoliko ne posjedujete neku domenu ili hosting, XAMPP će simulirati sve to u off-line načinu rada. [3]

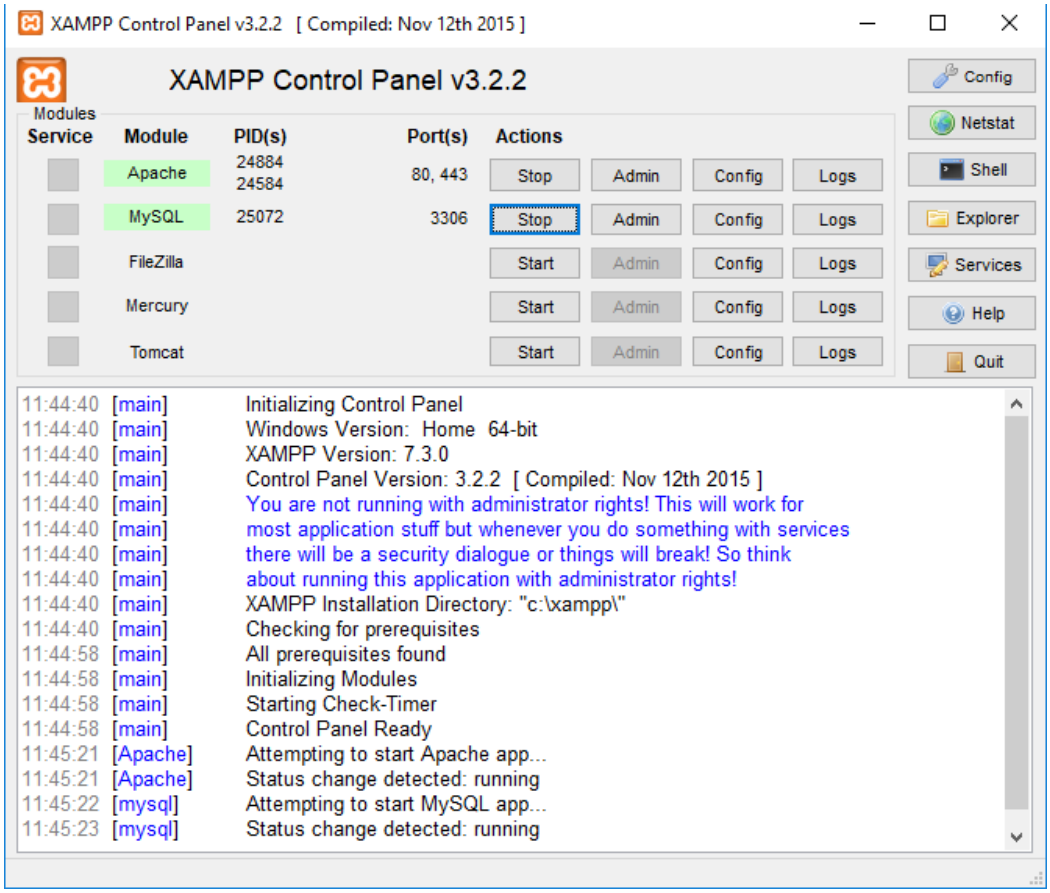
Osnovni paket XAMPP-a sadrži:

- Apache
- DBMS
- PHP
- MariaDB
- Webalizer
- FileZilla FTP poslužitelj
- OpenSSL

Prednosti korištenja XAMPP-a su:

- Potrebni moduli već ugrađeni u bazu
- Radi kao punopravni internetski poslužitelj
- Moguće funkcioniranje i u javnoj mreži uz unaprijed konfiguriranu javnu mrežu

- Sadrži poslužitelje za različite sustave



Sl. 2.2. *Izgled xampp sučelja*

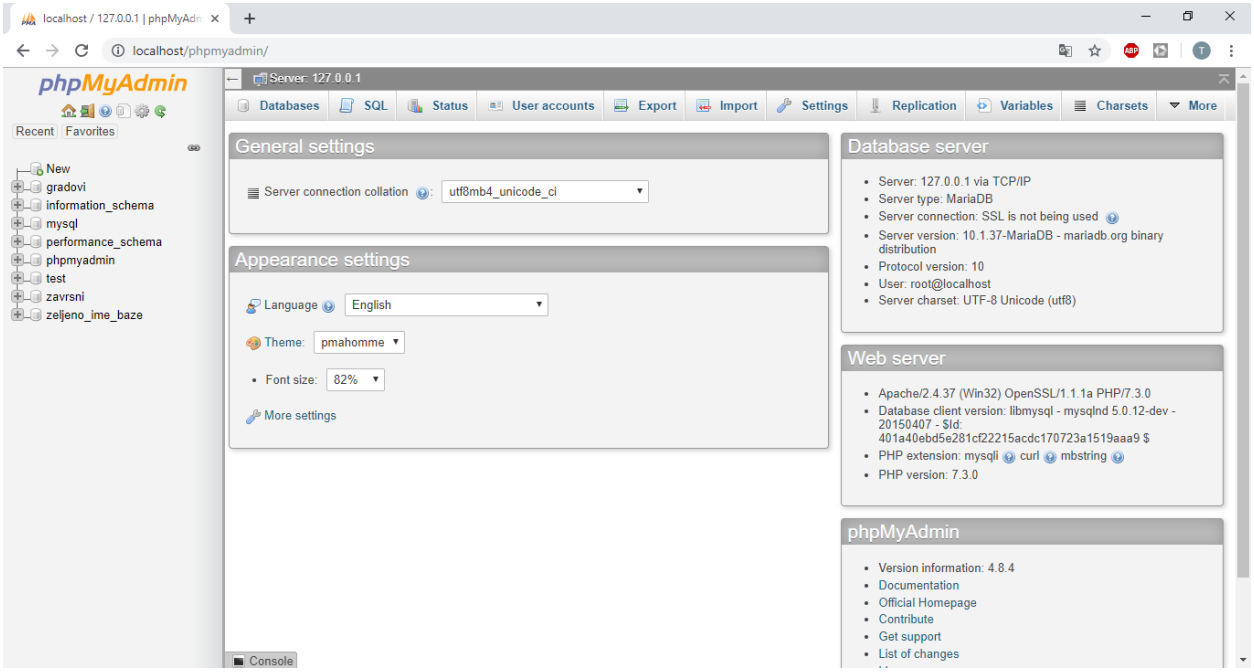
2.3. MySQL

MySQL je sustav za upravljanje bazama podataka. Besplatan je i open source sustav, korisnici ga vrlo često odabiru kod projekata otvorenog koda. Sastavni je dio Linux-a, ali podržava i druge operacijske sustave kao što su Windows, Mac OS X, UNIX, Symbian i drugi. [6]

Relacijski tip MySQL baza je vrlo popularan zbog funkcionalnog načina pretraživanja i spremanja velikog broja podataka, koristi se skoro u svakom informacijskom sustavu upravo zbog toga što korisnici zahtijevaju dostupnost brzih i kvalitetnih informacija. Skalabilnost MySQL-a je također jako bitna značajka uz vrhunsku podršku za development interface elemente. MySQL se bazira na korištenju R-/R+ Tree, Hash i Full-text indexing-u, velika je zajednica MySQL podrške.

MySQL baze su veoma brze, ali stabilne te sadrže podršku brojnih programskih jezika, a neki od njih su: Java, Python, PHP itd. Kako bi upravljali MySQL bazama podataka potreban nam

je phpMyAdmin, koji je besplatan alat izrađen u PHP-u. [10] Za njegovo pokretanje potrebno je uključiti servere Apache i MySQL u XAMPP-u kao što je vidljivo prema slici 2.2.. Nakon što su uključeni spomenuti serveri, u web pregledniku je potrebno samo pozvati localhost: http://localhost/phpmyadmin.



Sl. 2.3. *Sučelje phpMyAdmin*

2.4. Html

HTML (engl. HyperText Markup Language) jest prezentacijski jezik izradu web stranica. Pomoću HTML jezika se stvaraju hipertext dokumenti. [5] Sadržaj hipertext dokumenta te njegove hiperveze se oblikuju pomoću HTML-a. Besplatan je, pa je zbog toga i zbog svoje jednostavnosti jako rasprostranjen i vrlo često korišten. Pošto je glavni cilj HTML jezika omogućiti prikazivanje hipertext dokumenta na web pregledniku, potrebno je učiniti da se dokument ne mijenja, odnosno da izgleda jednako neovisno gdje se prikazuje, na kojem web pregledniku, računalu ili operacijskom sustavu. Bitno je naglasiti da HTML nema veze s programiranjem jer on ne vrši nikakve operacije ni zadaće, nego se koristi samo za opis hipertekstualnih dokumenata.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Naziv stranice</title>
  </head>
  <body>
    <p>Ovdje se unosi sam sadržaj stranice.</p>
  </body>
</html>
</pre>
```

Sl. 2.4. *Primjer jednostavnog HTML dokumenta*

2.5. Php

PHP se koristi za programiranje dinamičnih web stranica. Sadrži dosta široku podršku internet protokola kao i baza podataka te su korisniku dostupne razne programerske knjižnice. PHP je jako sličan programskom jeziku C, većina kodova je jednostavna što mu je velika prednost kod korisnika te je zato jedan od najkorištenijih programskih jezika koji se koriste za programiranje web aplikacija. [4]

Kod izrade složenijih aplikacija programeri koriste web framework-ove kao što su:

- CakePHP
- Yii
- Laravel
- CodeIgniter itd.

Neki od najpoznatjih PHP editora su:

- Windows : Notepad++, ConTEXT, Dreamweaver
- Linux : gPHPEdit, Geany, Bluefish
- Mac OS X : TextMate, Coda, Eclipse
- Multiple OS : PhpStorm, Aptana, Emacs.

```php
<?php
$str_isset = "";
$bol_isset = isset($str_isset);

If ($bol_isset){
   echo "The variable is set";
}
else {
    echo "The variable is not set";
}
?>

```

Sl. 2.5. *Primjer programa napisanog u PHP-u*

2.6. Sql

SQL (Structured Query Language) je programski jezik visoke razine koji se koristi za izradu, ažuriranje, brisanje podataka iz povezanih baza podataka.

Neke od naredbi su:

- Grupiranje/pretraga podataka
  - *GROUP BY*
  - *ORDER BY*
  - *FROM*
  - *WHERE*
- Mijenjanje podataka
  - *DELETE*
  - *INSERT*
  - *UPDATE*
- Definiranje podataka
  - *ALTER*
  - *DROP*
  - *CREATE*
- Kontrola podataka
  - *REVOKE*
  - *GRANT*

```
CREATE TABLE moja_tablica (
  moje_polje1   INT,
  moje_polje2   VARCHAR (50),
  moje_polje3   DATE         NOT NULL,
  PRIMARY KEY (moje_polje1, moje_polje2)
);
```

Sl. 2.6. *Primjer kreiranja tablice i definiranja podataka*

2.7. JavaScript

JavaScript je programski jezik sličan Javi, čisto zbog lakšeg korištenja, međutim temelji se na prototipu, a nije objektno orijentiran kao Java. Razvila ga je tvrtka Netscape. JavaScript sadrži AJAX (Asynchronous JavaScript and XML) tehniku koja omogućava komunikaciju web stranica sa serverskim programom, što olakšava rad web aplikacije i poveća interaktivnost iste.

```
1  <html>
2  <head>
3  </head>
4  <body>
5      <script>
6          var suma = 0; // Početna vrijednost sume mora biti nula
7          for(var i = 1; i < 100; i++)
8          {
9              //
10         }
11     </script>
12 </body>
13 </html>
```

Sl. 2.7. *Primjer jednostavnog programa napisanog u JavaScript-u*

3. IZRADA APLIKACIJE ZA PRIKAZ GRADOVA HRVATSKE

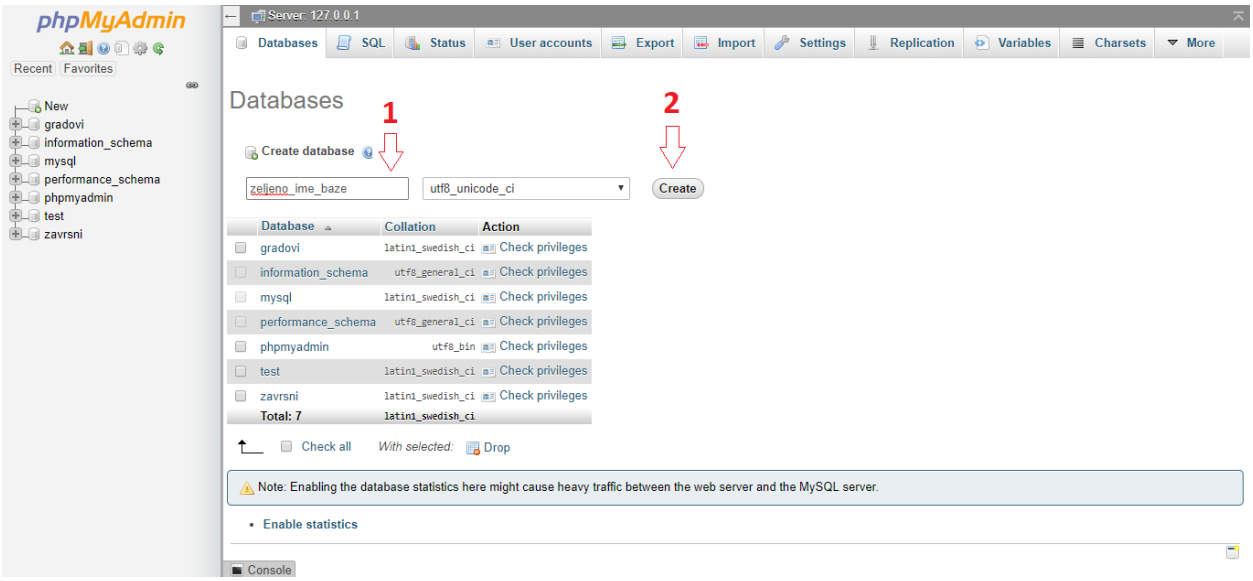
3.1. Stvaranje baze podataka

Za izradu baze podataka koja se koristi u ovoj web aplikaciji koristi se također web aplikacija naziva phpMyAdmin. Kako bi imali pristup web aplikaciji phpMyAdmin, potrebno je također imati pristup nekom MySQL serveru, a to se postiže korištenjem XAMPP Apache web servera zajedno s PHP-om i MySQL-om.

Kako bi pristupili aplikaciji phpMyAdmin, upisujemo *http://localhost/phpmyadmin* u web preglednik. Bazu podataka kreiramo pod *Create new database* gdje se upiše željeno ime baze podataka i zatim *Create*. Baza podataka se također može kreirati i jednostavnim MySQL upitom:

CREATE DATABASE zeljeno_ime_baze.

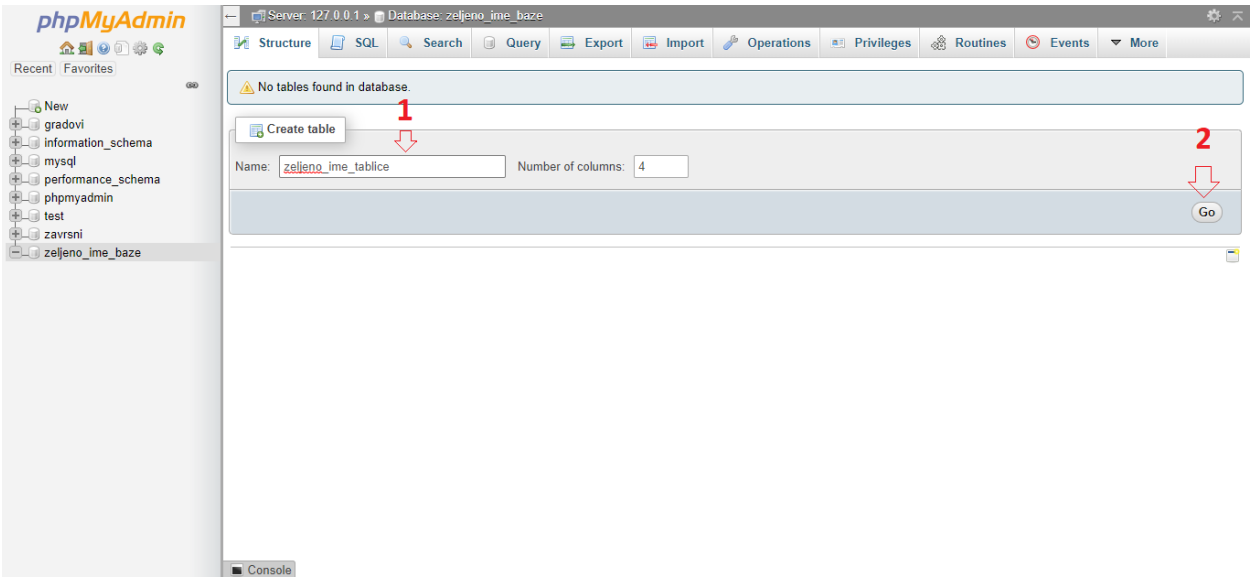
Taj upit se piše u SQL odjeljku do kojeg se dolazi jednostavno klikom na SQL karticu pri vrhu phpMyAdmin aplikacije.



Sl. 3.1. *Izrada baze podataka*

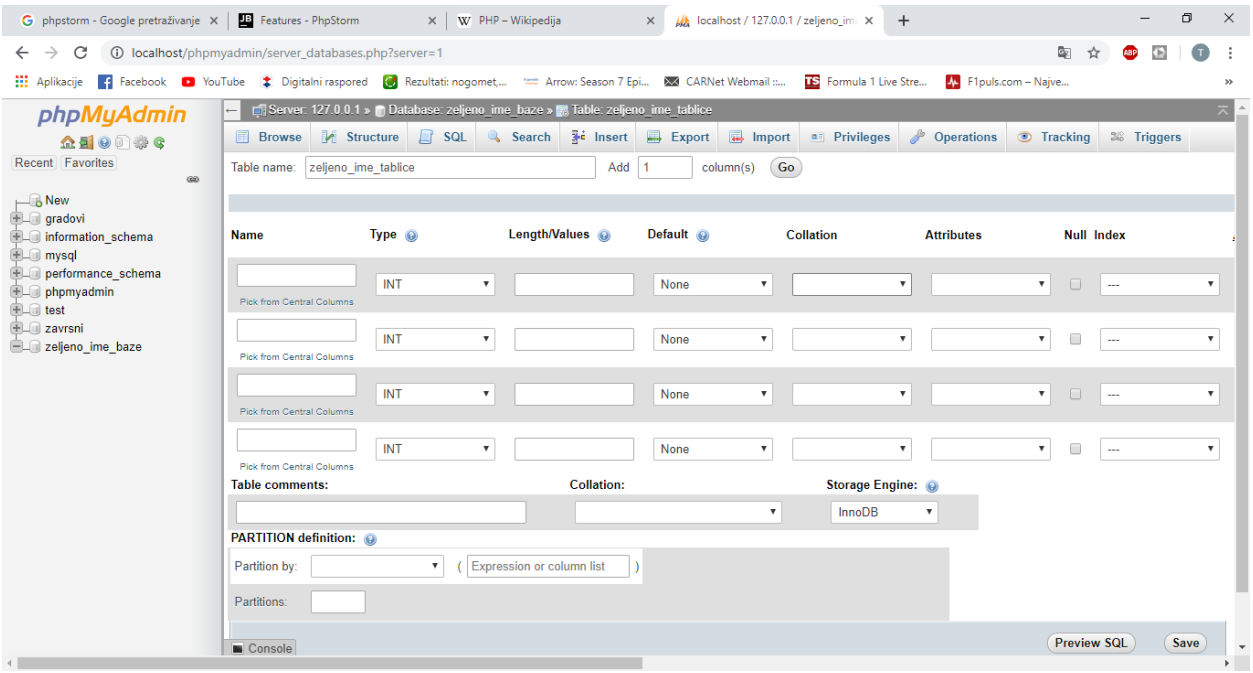
3.1.1. Stvaranje tablice u bazi podataka

Nakon što je izrađena baza podataka, izrađuje se tablica. Tablica se može izraditi jednostavno upisom imena tablice i pritiskom na *Go*, što je ponuđeno čim napravite novu bazu podataka. Također imamo i opciju koliko želimo stupaca želimo u tablici.



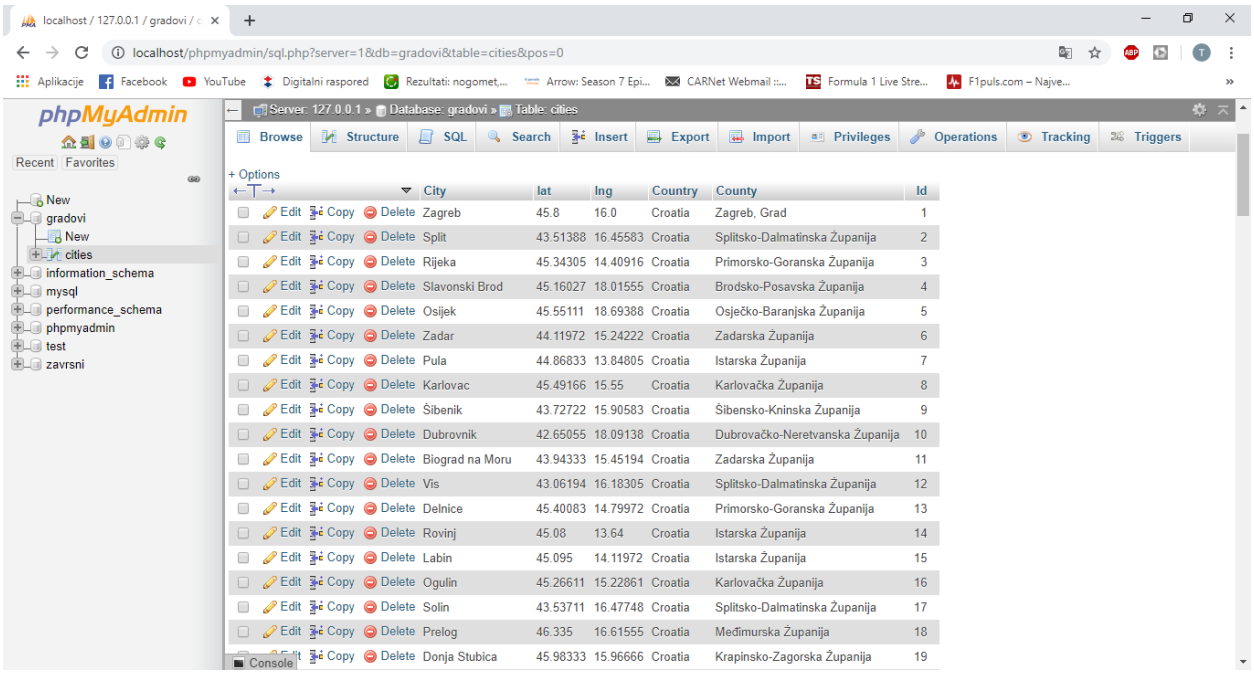
Sl. 3.2. *Izrada tablice u bazi podataka*

Kad smo izradili tablicu, program nas odmah preusmjerava na zadavanje imena stupca, tipa podataka koji će se nalaziti u tim stupcima, upisivanja podataka odnosno vrijednosti podataka i ostalih dijelova strukture tablice.



Sl. 3.3. *Određivanje naziva stupaca, tipa podataka, vrijednosti podataka itd.*

Nakon što se ispune svi podaci potrebni, klikom na *Save* se dobija tablica u bazi podataka koja se dalje koristi za programiranje zadanog zadatka.



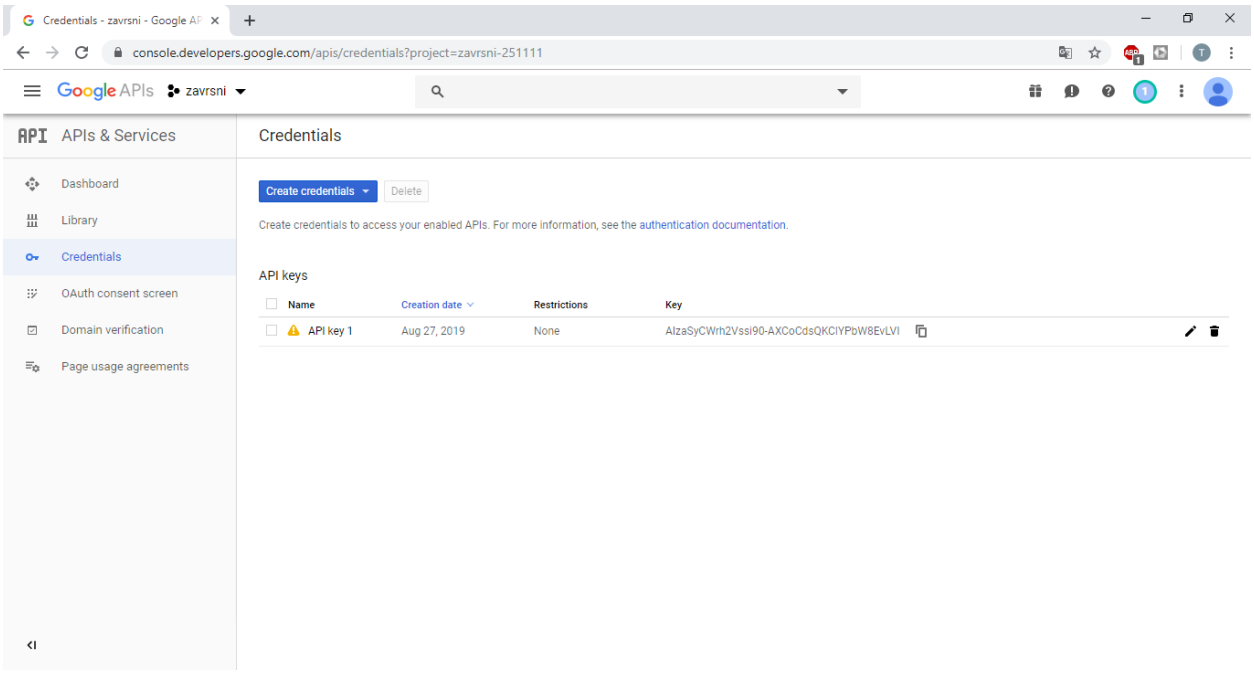
Sl. 3.4. *Dio baze podataka koja se koristi u web aplikaciji za prikaz gradova u Hrvatskoj*

3.2. Generiranje karte na web aplikaciji

Kako bi generirali kartu Hrvatske, potrebno je napraviti Google API ključ kako bi mogli koristiti Google kartu Hrvatske.

3.2.1. Stvaranje Google API ključa

Google API ključ se izrađuje kako bi se dobio pristup kartama, mjestima, geokodiranju i ostalim stvarima povezanima s Google API-jem ukoliko ga korisnik želi koristiti u svom programu. Stvaranje ključa se odvija na internetskoj stranici Google Developers .[7] Potrebno je naći *Google API Console* koji se nalazi pod naslovom *Developer Consoles*. Zatim pod *Credentials* je potrebno samo kliknuti na *Create credentials* i onda *API key*. Dobiveni API ključ se kopira da se može koristiti u generiranju karte.



Sl. 3.5. *Google API ključ*

3.2.2. Iscrtavanje karte

Za iscrtavanje karte potrebno je definirati funkciju za učitavanje karte kojoj se dodaju atributi karte koju je poželjno učitati (definiranje cijele te funkcije se radi pomoću JavaScripta). U toj funkciji se nalazi definicija varijable Hrvatska kojoj su dodane koordinate te varijable (zemljopisna širina i dužina, engl. latitude and longitude). Pomoću varijable *map* se stvara karta na osnovu koordinata varijable Hrvatska, odnosno iscrtava se karta Hrvatske sa centrom točno u koordinatama iste. [8]

```
var map;

function initMap() {
    var hrvatska = {lat: 44.4738, lng: 16.4689};
    map = new google.maps.Map(document.getElementById("map"),
        {
            zoom: 6.8,
            center: hrvatska

        });
}
```

Sl. 3.6. *Funkcija za iscrtavanje karte u Web aplikaciji*

3.3. Povezivanje s bazom podataka

Kako bi pristupili podacima iz baze podataka i radili s njima, potrebno je povezati se s bazom podataka. Zbog toga je potrebno kreirati klasu u kojoj se definiraju parametri iz tablice u bazi podataka, te se definiraju funkcije postavljanja i dohvaćanja tih istih parametara (*set/get* funkcije). Potrebno je također definirati i konstruktor koji će omogućiti povezivanje s bazom podataka.

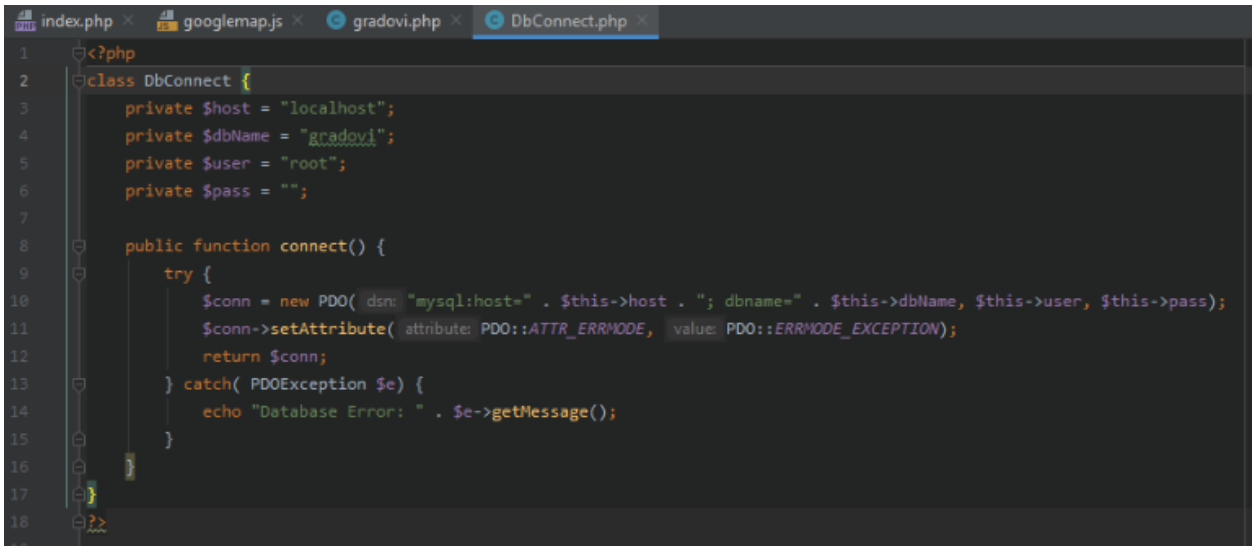
```php
<?php

class gradovi {
    private $City;
    private $lat;
    private $lng;
    private $Country;
    private $County;
    private $Id;
    private $conn;
    private $tableName = "cities";

    function setCity($City) { $this->City = $City; }
    function getCity() { return $this->City; }
    function setLat($lat) { $this->lat = $lat; }
    function getLat() { return $this->lat; }
    function setLng($lng) { $this->lng = $lng; }
    function getLng() { return $this->lng; }
    function setCountry($Country) { $this->Country = $Country; }
    function getCountry() { return $this->Country; }
    function setCounty($County) { $this->County = $County; }
    function getCounty() { return $this->County; }
    function setId($Id) { $this->Id = $Id; }
    function getId() { return $this->Id; }

```

Sl. 3.7. *Klasa s parametrima iz tablice u bazi podataka i funkcije postavljanja i dohvaćanja parametara*

```php
public function __construct() {
    require_once("db/DbConnect.php");
    $conn = new DbConnect;
    $this->conn = $conn->connect();
}
```

Sl. 3.8. *Definicija konstruktora za povezivanje s bazom podataka*

Konstruktor zahtijeva datoteku *DbConnect* koja sadrži sve potrebno za dohvaćanje podataka iz baze podataka odnosno povezivanje s istom. Najvažniji dio koji sadrži ova datoteka je PDO, odnosno PHP Data Objects koji omogućuju nevjerojatno jednostavan pristup i povezivanja s bazom podataka. Temelji se na funkciji *try/catch* koja radi na principu da ukoliko dođe do bilo kakvog problema, ti isti problemi se vraćaju u *catch* blok, odnosno ukoliko dođe do problema s povezivanjem na bazu podataka *PDO()* objekt vraća *PDOException().*



Sl. 3.9. *Datoteka koja sadrži sve potrebno za povezivanje s bazom podataka*

3.4. Geokodiranje

Geokodiranje se odnosi na traže zemljopisne širine i dužine, odnosno koordinata gradova iz baze podataka koji trebaju biti prikazani na iscrtanoj karti. Koordinate grada su potrebne iz razloga da bi preko programa mogli te iste gradove iscrtati na karti. Da bi se pristupilo podacima iz baze prvo treba definirati funkciju koja će iz baze podataka izvući one gradove koji nemaju upisane koordinate (*lat/lng*) u tablici. Podacima se pristupa preko SQL upita.

```
public function getCitiesBlankLatLng() {
    $sql = "SELECT * FROM $this->tableName WHERE lat IS NULL AND lng IS NULL";
    $stmt = $this->conn->prepare($sql);
    $stmt->execute();
    return $stmt->fetchAll( fetch_style: PDO::FETCH_ASSOC);
}
```

Sl. 3.10. *Dohvaćanje gradova koji nemaju upisane koordinate(lat/lng) iz baze podataka*

Nakon što je funkcija dohvatila podatke, te iste podatke treba spremiti u JavaScript kako bi se nad njima moglo izvršiti geokodiranje. Potrebno je u glavnom dijelu programa zahtijevati dio programa gdje se dohvaćaju podatci iz baze podataka, a to se radi pomoću naredbe *require*. Zatim se ti podatci spremaju naredbom *json_encode* kako bi ih se moglo koristiti u JavaScriptu.

```php
<?php
function utf8ize($d) {
    if (is_array($d)) {
        foreach ($d as $k => $v) {
            $d[$k] = utf8ize($v);
        }
    } else if (is_string ($d)) {
        return utf8_encode($d);
    }
    return $d;
}
require 'gradovi.php';
$city = new gradovi;
$Data = $city->getCitiesBlankLatLng();
$encodedData = json_encode(utf8ize($Data));
echo '<div id="Data">' . $encodedData . '</div>';
?>
```

Sl. 3.11. *Spremanje podataka dohvaćenih preko funkcije za korištenje u JavaScriptu*

Nakon što su podaci spremni za rad u JavaScript-u, treba napisati funkciju za geokodiranje koja će pomoću imena grada iz baze podataka naći njegove koordinate. Kada se izvrši geokodiranje, slijedi spremanje koordinata u varijablu koju predajemo funkciji koja na kraju učitava te podatke u bazu podataka, odnosno pridružuje koordinate (*lat/lng*) gradova njihovim imenima.

```javascript
var cdata = JSON.parse(document.getElementById( elementId: "data").innerHTML);
geocoder = new google.maps.Geocoder();
codeAddress(cdata);

function codeAddress(cdata) {
    Array.prototype.forEach.call(cdata, argArray: function(data){
        var address = data.City + ' ' + data.Id;
        geocoder.geocode( { 'address': address}, function(results, status) {
            if (status === 'OK') {
                map.setCenter(results[0].geometry.location);
                var points = {};
                points.Id = data.Id;
                points.lat = map.getCenter().lat();
                points.lng = map.getCenter().lng();
                updateCitiesWithLatLng(points);
            } else {
                alert('Geocode was not successful for the following reason: ' + status);
            }
        });
```

Sl. 3.12. *Geokodiranje*

3.5. Spremanje rezultata geokodiranja u bazu podataka

Ukoliko je geokodiranje uspješno obavljeno rezultati se spremaju pomoću varijable *points* u funkciju koja će ažurirati tablicu na osnovu podataka dobivenih geokodiranjem, a to su koordinate gradova.

```
function updateCitiesWithLatLng(points) {
    $.ajax({
        url:"action.php",
        method:"post",
        data: points,
        success: function(res) {
            console.log(res)
        }
    })

}
```

Sl. 3.13. *Funkcija ažuriranja tablice rezultatima geokodiranja*

Funkcija zahtijeva učitavanje dijela programa koji se odnosi na provjeru uspješnosti ažuriranja tablice rezultatima geokodiranja.

```
<?php
require "gradovi.php";
$gr = new gradovi;
$gr->setId($_REQUEST["Id"]);
$gr->setLat($_REQUEST["lat"]);
$gr->setLng($_REQUEST["lng"]);
$status = $gr->updateCitiesWithLatLng();
if($status == true) {
    echo "Updated...";
} else {
    echo "Failed...";
}
?>
```

Sl. 3.14. *Provjera uspješnog ažuriranja tablice*

3.6. Prikazivanje lokacija gradova na karti

Nakon uspješno odrađenog ažuriranja tablice, potrebno je napraviti funkciju koja će učitati sve gradove (ažurirane) iz tablice da bi ih se moglo prikazati na karti. Za prikaz gradova na karti

koriste se markeri koji na osnovu koordinata gradova pokazuju njihovu točnu lokaciju na karti i klikom na svaki od markera će se pokazati ime grada koji se nalazi na mjestu tog markera.

Prva stvar kod učitavanja gradova jest njihovo dohvaćanje iz baze podataka preko SQL upita. Nakon što se učitaju potrebno ih je spremiti za korištenje u JavaScript-u. Posljednja stvar potrebna za njihovo prikazivanje na karti je funkcija u JavaScript-u koja učitava gradove iz varijable u koju su oni spremljeni pomoću *json_encode* funkcije. Također je potrebno i definirati markere koji će se pojavljivati na mjestima koordinata gradova te ime grada koje će se pokazati prilikom klika na određeni marker.

```php
public function getAllCities() {
    $sql = "SELECT * FROM $this->tableName";
    $stmt = $this->conn->prepare($sql);
    $stmt->execute();
    return $stmt->fetchAll( fetch_style: PDO::FETCH_ASSOC);
}
```

Sl. 3.15. *Funkcija za dohvaćanje ažuriranih gradova iz tablice u bazi podataka*

Nakon što su podaci dohvaćeni iz baze podataka, u glavnom dijelu programa ih je potrebno spremiti u varijablu kako bi ih mogli u JavaScript-u koristiti. Kako bi ih koristili u JavaScript-u treba spremiti rezultate funkcije *getAllCities* u varijablu *allData* kako bi nju proslijedili pomoću *json_encode* funkcije u JavaScript.

```php
<?php
function utf8ize($d) {
    if (is_array($d)) {
        foreach ($d as $k => $v) {
            $d[$k] = utf8ize($v);
        }
    } else if (is_string ($d)) {
        return utf8_encode($d);
    }
    return $d;
}
require 'gradovi.php';
$city = new gradovi;
$allData = $city->getAllCities();
$encodedData = json_encode(utf8ize($allData));
echo '<div id="allData">' . $encodedData . '</div>';
?>
```

Sl. 3.16. *Spremanje podataka za korištenje u JavaScript-u*

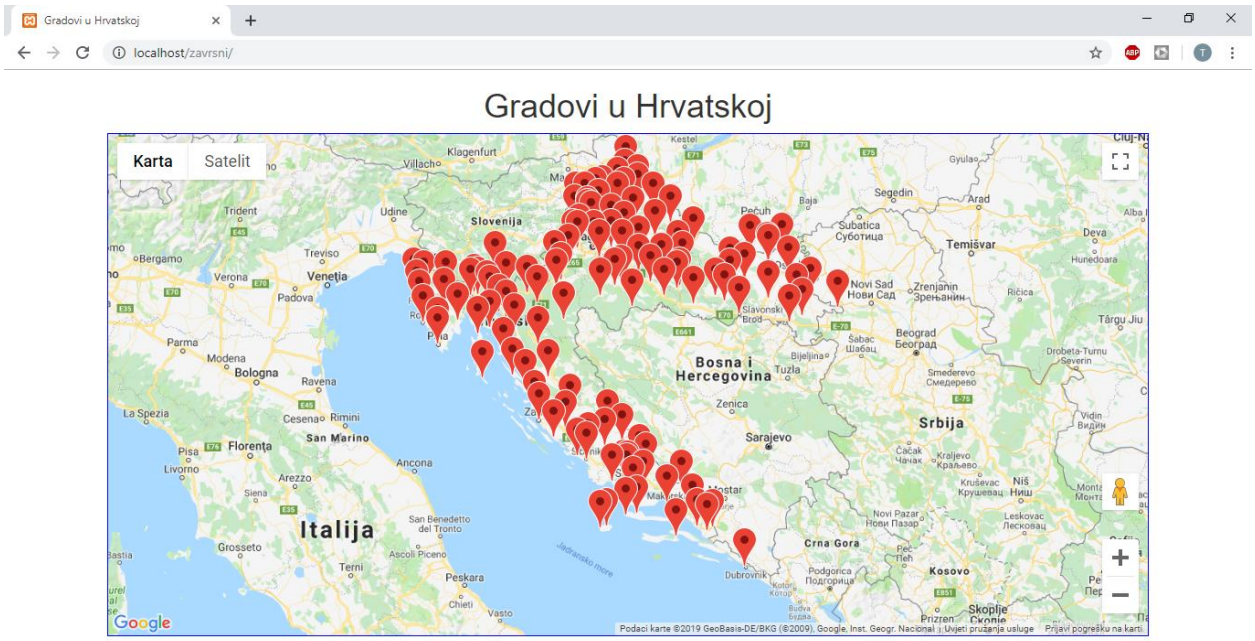
Posljednji korak za prikazivanje lokacija gradova na karti jest izrada funkcije koja će te iste gradove pomoću markera iscrtati na mjestima zadanim njihovim koordinatama u bazi podataka. Učitavanjem podataka spremljenih u varijablu *allData* funkcija stvara markere na osnovu *lat* i *lng* vrijednosti koje se nalaze u bazi podataka. Nakon što su markeri postavljeni na mjestima zadanim koordinatama gradova, korištenjem funkcije za dodavanje *listenera* koji reagira na klik na marker i pokazuje informacije, odnosno ime grada koji se nalazi na lokaciji tog markera.

```
    var someData = JSON.parse(document.getElementById( elementId: 'allData').innerHTML);
    showAllCities(someData)
}

function showAllCities(allData) {
        var infoWind = new google.maps.InfoWindow;
    Array.prototype.forEach.call(allData,  argArray: function(data) {
        var content = document.createElement( tagName: 'div');
        var strong = document.createElement( tagName: 'strong');
        strong.textContent = data.City;
        content.appendChild(strong);

        var marker = new google.maps.Marker({
            position: new google.maps.LatLng(data.lat, data.lng),
            map: map
        });
        marker.addListener( listener: 'click', function () {
            infoWind.setContent(content);
            infoWind.open(map, marker);
        })
    })
}
```

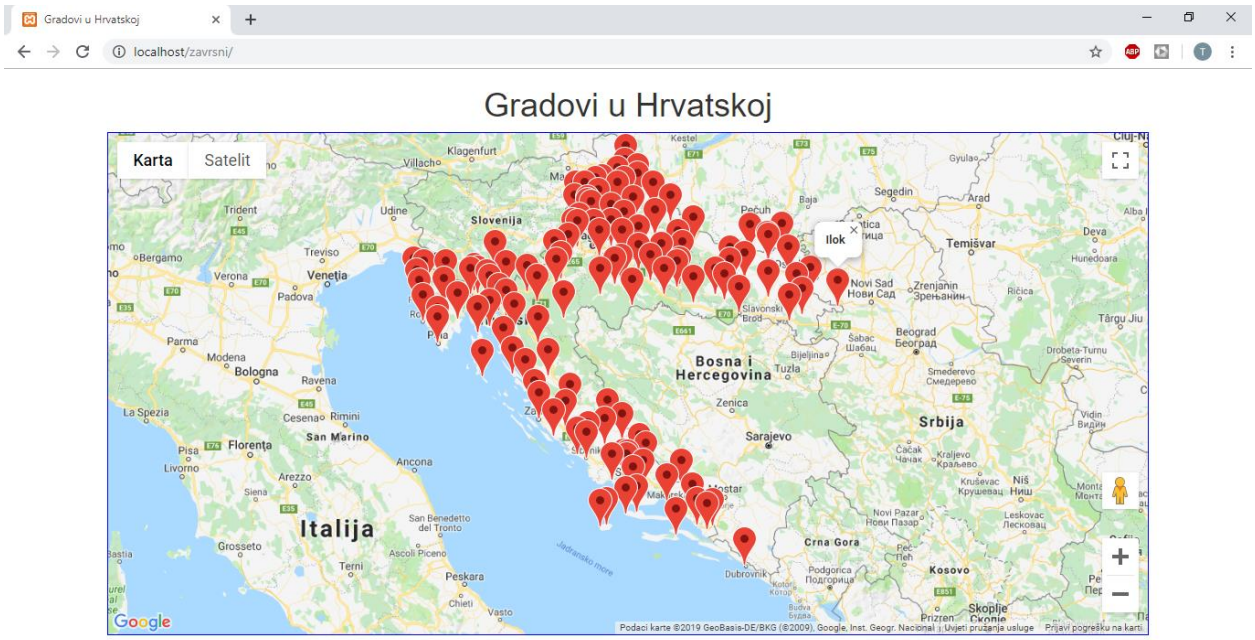
Sl. 3.17. *Funkcija stvaranja markera i pokazivanje imena grada nakon klika na marker*

4. IZGLED APLIKACIJE I NAČIN KORIŠTENJA

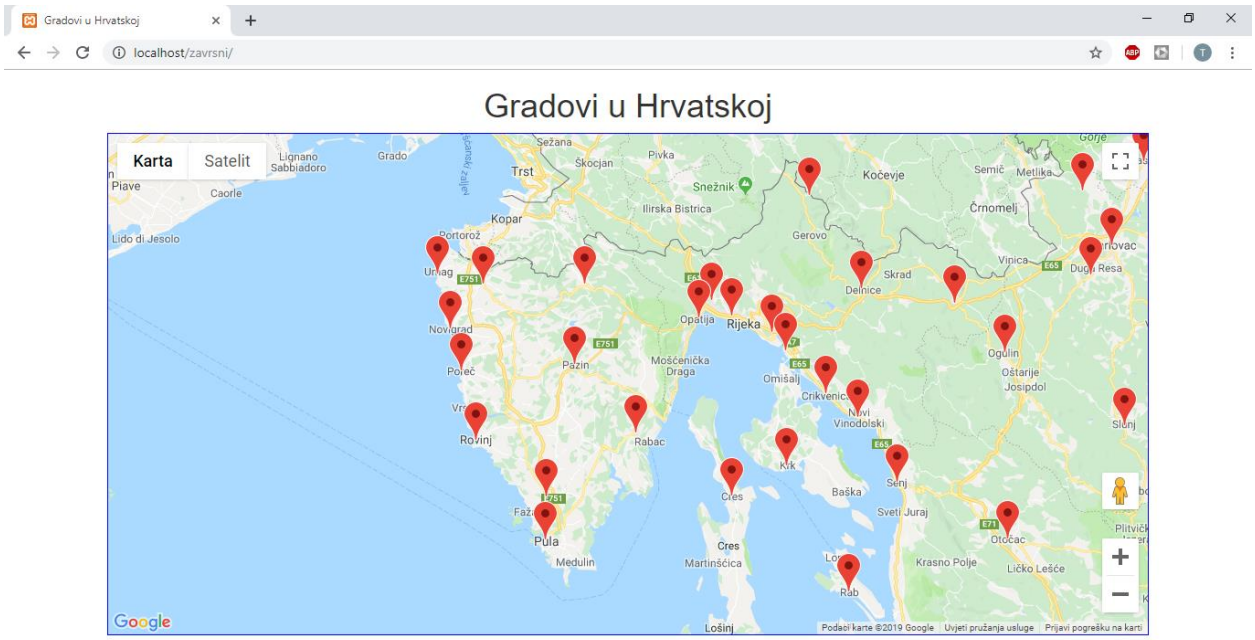
Aplikacija se pokreće jednostavnim ulaskom u web preglednik i upisivanjem lokalne adrese za pokretanje iste. Pokretanjem same aplikacije se automatski pokazuju svi gradovi na karti te se klikom na bilo koji od njih pokazuje ime tog grada. Također je moguće uvećati kartu i tako iz bližega provjeriti o kojem se gradu točno radi.



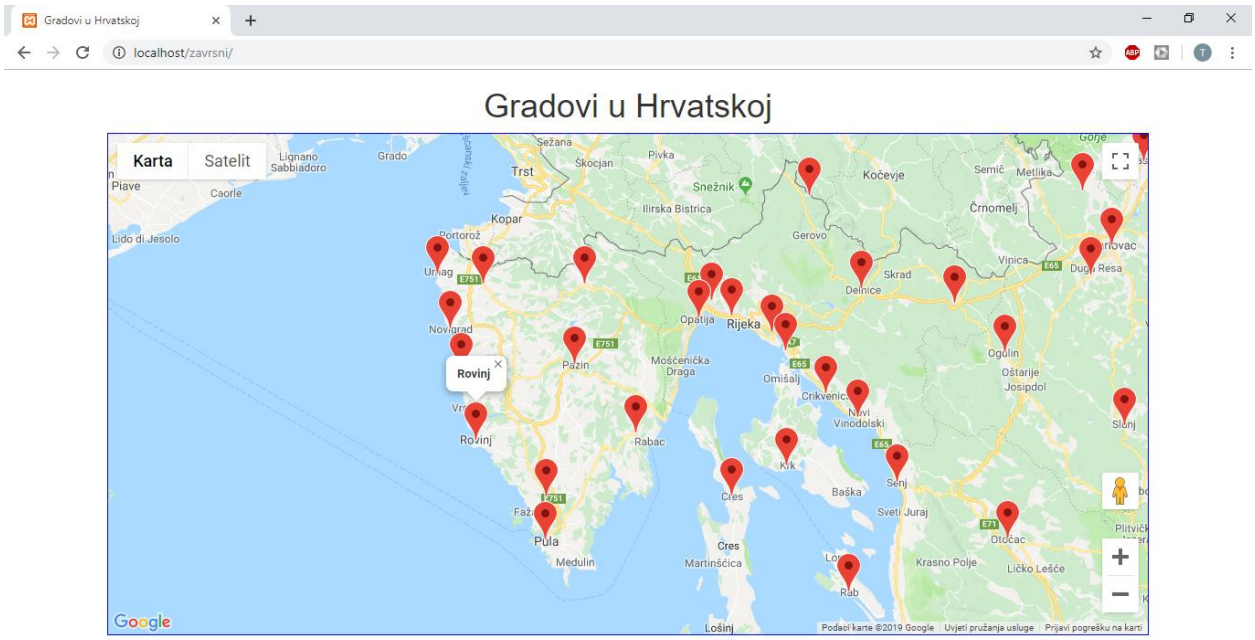
Sl. 4.1. *Izgled nakon učitavanja aplikacije*



Sl. 4.2. *Pojava imena grada nakon klika na marker*



Sl. 4.3. *Uvećan prikaz označenih gradova*



Sl. 4.4. *Klik na marker*

Aplikacija je vrlo jednostavna za korištenje, zapravo ima dvije opcije, a to su uvećavanje karte i druga opcija je ime grada koje se dobije klikom na marker.

5. ZAKLJUČAK

Problem kojim se bavi ovaj završni rad je kako prikazati lokacije gradova na karti. Kako bi uspjeli prikazati lokacije gradova, potrebno je bilo geokodiranjem doći do koordinata gradova kako bi ih spremili u bazu podataka i onda pomoću njih prikazali te iste gradove na karti Hrvatske.

U radu je opisana web aplikacija za prikazivanje gradova Hrvatske na karti uz pomoć markera koji se nalaze na mjestima određenim koordinatama tih gradova. S obzirom da aplikacija ispravno radi i radi ono što je zadano opisom teme završnog rada, možemo zaključiti da je zadatak završnog rada uspješno ispunjen.

Učitavanjem web aplikacije, vidi se da su označeni svi gradovi iz baze podataka, također klikom na bilo koji od tih gradova vidljivo je njihovo ime. Ukoliko se žele izbliza vidjeti neki gradovi, kartu je moguće zumirati i također klikom na marker se vidi ime tog grada.

Aplikacija je napravljena za svakodnevno korištenje ukoliko je potrebno pronaći gdje se nalazi koji grad Republike Hrvatske na karti iste.

LITERATURA

[1] https://en.wikipedia.org/wiki/PhpStorm, pristupljeno: 28.kolovoza, 2019. godine

[2] https://www.jetbrains.com/phpstorm/features/, pristupljeno: 28.kolovoza, 2019. godine

[3] https://wmforum.geek.hr/t/xampp-sve-u-ovoj-temi/27697, pristupljeno: 29.kolovoza, 2019. godine

[4] https://hr.wikipedia.org/wiki/PHP, pristupljeno: 1.rujna, 2019. godine

[5] https://hr.wikipedia.org/wiki/HTML, pristupljeno: 1.rujna, 2019. godine

[6] https://znatko.com/2753/sto-je-mysql, pristupljeno: 1.rujna, 2019. godine

[7] https://developers.google.com/maps/, pristupljeno: 2.rujna, 2019. godine

[8] https://developers.google.com/maps/documentation/javascript/adding-a-google-map#map, pristupljeno: 2.rujna, 2019. godine

[9] http://bolgarna.site/9324116e7a813, pristupljeno: 2.rujna, 2019. godine

[10] https://www.hdonweb.com/programiranje/phpmyadmin-kreiranje-mysql-baze-podataka, pristupljeno: 3.rujna, 2019. godine

SAŽETAK

U ovom završnom radu je obrađena tema izrade web aplikacije za prikaz gradova Hrvatske na karti Hrvatske. Korisnik učitavanjem web aplikacije pokreće učitavanje karte Hrvatske i gradove označene markerima na mjestima njihovih koordinata. U izradi aplikacije korištene su brojne tehnologije za izradu web aplikacija. U radu je opisan način rada aplikacije, te su također dani i primjeri programskog koda.

Ključne riječi: karta, grad, lokacija, koordinate, marker, ime grada, web aplikacija

Title of the final paper: Cities of Croatia

ABSTRACT

This final paper processes the topic of creating a web application for displaying Croatian cities on a map of Croatia. By loading the web application, the user starts loading the map of Croatia and the cities marked with markers at the locations of their coordinates. Numerous web application development technologies have been used in application development. The paper describes how the application works and examples of programming code are also given.

Keywords: map, city, location, coordinates, marker, city name, web application

ŽIVOTOPIS

Toni Ivešić rođen je 4. veljače 1997. godine u Slavonskom Brodu. Pohađa Osnovnu Školu Ivana Filipovića u Velikoj Kopanici te istu završava 2011.godine. Zatim upisuje srednju Tehničku školu u Slavonskom Brodu, smjer Tehničar za računalstvo i završava ju 2015. godine. Po završetku srednje škole upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer Računarstvo na preddiplomskom sveučilišnom studiju, koji trenutno pohađa.

Toni Ivešić

U Osijeku, 2019. _____

PRILOZI

1. Tekst završnog rada u .docx formatu
2. Tekst završnog rada u .pdf formatu
3. Izvorni kod