# Modeliranje i izrada dimenzionalno prilagodljive web stranice

**Kovač, Marina**

**Undergraduate thesis / Završni rad**

**2018**

*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:200:604836

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-04-20**

*Repository / Repozitorij:*

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**

**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA**

**Stručni studij**

# Modeliranje i izrada dimenzionalno prilagodljive web stranice

**Završni rad**

**Marina Kovač**

**Osijek, 2018.**

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

**Obrazac Z1S: Obrazac za imenovanje Povjerenstva za obranu završnog rada na preddiplomskom stručnom studiju**

**Osijek, 09.07.2018.**

**Odboru za završne i diplomske ispite**

# Imenovanje Povjerenstva za obranu završnog rada na preddiplomskom stručnom studiju

| | |
|---|---|
| **Ime i prezime studenta:** | Marina Kovač |
| **Studij, smjer:** | Preddiplomski stručni studij Elektrotehnika, smjer Informatika |
| **Mat. br. studenta, godina upisa:** | AI4277, 13.10.2017. |
| **OIB studenta:** | 14900243175 |
| **Mentor:** | Doc.dr.sc. Mirko Kohler |
| **Sumentor:** | |
| **Sumentor iz tvrtke:** | |
| **Predsjednik Povjerenstva:** | Izv. prof. dr. sc. Krešimir Nenadić |
| **Član Povjerenstva:** | Doc.dr.sc. Ivica Lukić |
| **Naslov završnog rada:** | Modeliranje i izrada dimenzionalno prilagodljive web stranice |
| **Znanstvena grana rada:** | **Programsko inženjerstvo (zn. polje računarstvo)** |
| **Zadatak završnog rada** | Zadatak ovog rada je napraviti model dimenzionalno prilagodljive web stranice. Također je potrebno obraditi različite načine za prilagođavanje stranice dimenzijama zaslona. Potrebno je implementirati naslovnu stranicu s vijestima, kalendar, forum, prijavu korisnika itd. |
| **Prijedlog ocjene pismenog dijela ispita (završnog rada):** | Izvrstan (5) |
| **Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:** | Primjena znanja stečenih na fakultetu: 2 bod/boda<br>Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda<br>Jasnoća pismenog izražavanja: 2 bod/boda<br>Razina samostalnosti: 3 razina |
| **Datum prijedloga ocjene mentora:** | 09.07.2018. |

| | |
|---|---|
| *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, 18.09.2018.**

| | |
|---|---|
| **Ime i prezime studenta:** | Marina Kovač |
| **Studij**: | Preddiplomski stručni studij Elektrotehnika, smjer Informatika |
| **Mat. br. studenta, godina upisa:** | AI4277, 13.10.2017. |
| **Ephorus podudaranje [%]:** | 3 |

Ovom izjavom izjavljujem da je rad pod nazivom**: Modeliranje i izrada dimenzionalno prilagodljive web stranice**

izrađen pod vodstvom mentora Doc.dr.sc. Mirko Kohler

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** ..... 1
**2. KORIŠTENE TEHNOLOGIJE** ..... 3
**2.1. HTML** ..... 3
**2.2. CSS** ..... 9
**2.3. Bootstrap i JavaScript** ..... 13
**2.4. PHP i SQL** ..... 24
**3. POSTIZANJE RESPONZIVNOSTI** ..... 30
**3.1. Grid (mreža) stranice** ..... 31
**3.2. Fleksibilni multimedia sadržaj** ..... 32
**3.3. Media Queries** ..... 33
**4. IZRADA DIMENZIONALNO PRILAGODLJIVE WEB STRANICE** ..... 35
**4.1. Izrada one page stranice za veterinarsku ambulantu** ..... 35
**4.2. Izrada te pregled Blog stranice i admin panela** ..... 46
**4.3. Kreiranje baze podataka i implementacija PHP kôda** ..... 56
**5. ZAKLJUČAK** ..... 65
**LITERATURA** ..... 66
**SAŽETAK** ..... 67
**ABSTRACT** ..... 68
**ŽIVOTOPIS** ..... 69
**PRILOZI** ..... 70

# 1. UVOD

Zadatak koji se razrađuje u ovom završnom radu je modeliranje i izrada dimenzionalno prilagodljive web-stranice, tematski stranica je namijenjena imaginarnom primjeru veterinarske stanice. Sama funkcionalnost web stranice prilagođena je i omogućava lakšu komunikaciju s veterinarima i poboljšava samo poslovanje veterinarske stanice. Vlasnici kućnih ljubimaca biti će u mogućnosti pristupiti stranici sa svojih mobilnih uređaja, tableta, laptopta i desktop računala te se pri tome sa svakog uređaja jednako dobro snalaziti po korisničkom sučelju stranice, koje se dimenzionalno prilagođava rezoluciji uređaja s kojeg se trenutno pristupa. Također korisnik koji želi koristiti web stranicu neće morati brinuti o izboru web preglednika unutar svog operacijskog sustava jer rad web stranice ne ovisi, niti je ograničen na samo jedan web preglednik. Stranica je izrađena kako bi korisnici u korak s modernim dobom, u kojemu je sve više raširen digitalni marketing i promocija usluga i proizvoda putem Interneta, mogli ugovoriti termine pregleda svojih kućnih ljubimaca bilo gdje i bilo kad ili pak imati još jedan izvor informacija o udomljavanju životinja kojima je hitna skrb potrebna.

Aktivnim korištenjem web stranica svaka tvrtka/djelatnost pospješuje širenje svojih usluga. Činjenica je da je većini ljudi korištenje Interneta veliki dio svakodnevice, a kada bi svatko odlučio svoje poslovanje proširiti i na Internet to bi svakako pridonijelo količini ljudi koji bi znali za određenu tvrtku/djelatnost koju bi svakodnevno pratili te preporučali ostalim ljudima koje znaju. Internet i uporaba web stranica time je izvrsna vrsta samo-promocije, a da pri tome korisnici imaju sve na jednom mjestu što će ih držati ažurnima u području koje ih zanima. Web stranica koja je namijenjena isključivo svrsi veterinarske stanice, imati će osnovne funkcionalnosti kao što je kontak forma pomoću koje korisnik može poslati upit o pregledu ili bilo čemu što ga zanima u vezi svog kućnog ljubimca, blog na kojemu će glavni veterinar objavljivati oglase o udomljavanju životinja te sličnim informacijama, cjenik usluga, upute za dolazak do veterinarske stanice (Google maps prikaz adrese) te sam kontak broj.

Kako bi postigli ovaj cilj biti će potrebno osmisliti iizgled baze podataka u koju će se spremati registrirani korisnici, podaci vezani za cijene usluga te blog postovi. Bootstrap framework doprinijeti će fludnijem, responzivnom i preglednom izgledu stranice, dok će se sam kostur i dizajn stranice provoditi pomoću HTML-a, CSS-a te ponešto JavaScripta.

U slijedećem poglavlju navedene su i opisane tehnologije koje su se koristile prilikom izrade one page stranice, bloga te admin sučelja. Svaka od tehnologija opisana je unutar zasebnog potpoglavlja uz objašnjenja i primjere njezine uporabe. Treće poglavlje opisuje proces postizanja responzivnosti tj. navedeni su ključni elementi pomoću kojih se na stranici postiže fluidan i prilagodljiv sadržaj. Ti ključni elementi su stupćasta mreža stranice, media queries te fleksibilni multimedijski sadržaj. Četvrto poglavlje objašnjava cijeli postupak postavljanja sučelja i izrade one page stranice te javnog i privatnog dijela bloga (admin sučelje). Sam proces izrade poredan je onako kako je izrada projekta tekla po potpoglavljima te se uz opis funkcionalnosti i cilja određenih elemenata nalaze slike kôdova i njihova objašnjena te slike izgleda pojedinih dijelova stranice. Peto, ujedno i zadnje poglavlje predstavlja osvrt na cijeli projekt, ukratko je sumirano ono što se postiglo tokom izrade, iznesena su zapažanja vezana uz primjenu funkcija stranice kao i moguća poboljšanja i rješenja potencijalih problema ukoliko nije bilo moguće ostvariti i implementirati sve osmišljene stavke.

# 2. KORIŠTENE TEHNOLOGIJE

Kako bi se lakše razumio koncept rada te proces njegove izrade ovo je poglavlje namijenjeno navođenju svrha i osnovnih značajki pojedinih tehnologija koje su korištene prilikom izrade ovog rada. Tehnologije koje će biti objašnjene su:

a) HTML
b) CSS
c) Bootstrap i JavaScript
d) Baza podataka, SQL i PHP

## 2.1. HTML

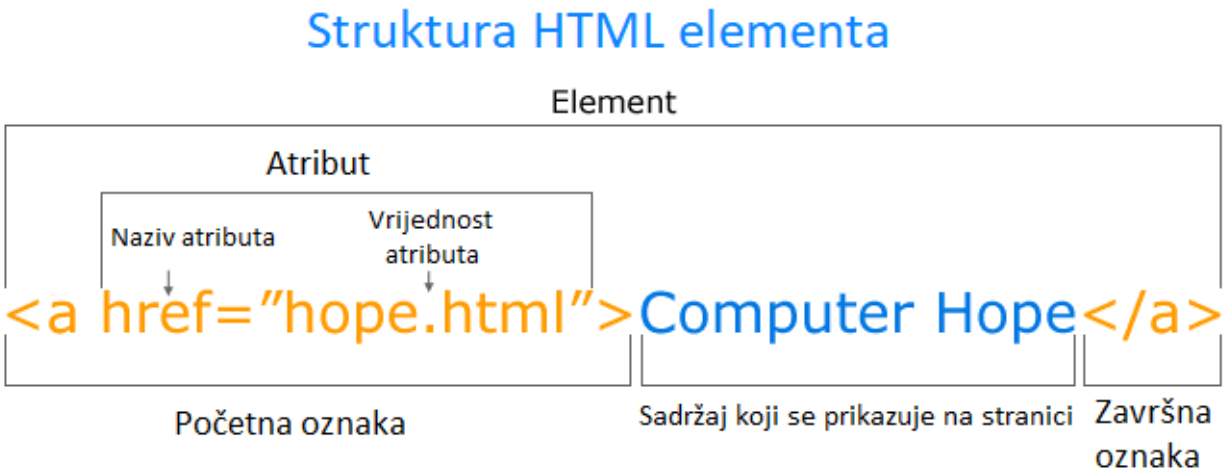
HTML (*HyperText Markup Language)* je jezik koji služi za izradu web stranica te pomoću njega oblikujemo sadržaj. Jednostavnije rečeno HTML je vrsta formata teksta namijenjane Internetu koji potom web preglednici (Chrome, IE, Safari, Mozilla itd.) isčitavaju i omogućuju korisniku vizualizaciju elemenata definiranih u tom dokumentu. HTML nije programski jezik jer njime nije moguće izvoditi nikakve matematičke i slične operacije, ali ga nazivamo jezikom jer ima svoju sintaksu i kodne riječi. HTML dokumenti se od običnih tekstualnih razlikuju samo po ekstenziji koja glasi *.html* ili *.htm*, time se također da zaključiti da HTML kôd možemo pisati i uređivati u programima koje nazivamo tekstualni editori kao što su *Notepad++*, *Sublime Text*, *Atom* itd.

Samu strukturu HTML-a čine *oznake (tags)* kojima uokvirujemo sadržaj koji će se nalaziti na stranici, postoji više vrsta oznaka ovisno o tome kakve funkcionalnosti želimo dodijeliti našem sadržaju. Osnovna struktura svakog HTML dokumenta prikazana je slikom 2.1..

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Naziv stanice koji će biti prikazan u kartici preglednika</title>
5    </head>
6    <body>
7      Sadržaj ide unutar body oznaka
8    </body>
9  </html>
10
```

**Sl. 2.1.** *Osnovna HTML struktura*

Kao što je pokazano u primjeru elementi koje zapisujemo HTML kodom sastoje se od početne i završne oznake, gdje početnu od završne razlikujemo po tome što završna sadrži kosu crtu „/“. <!DOCTYPE html> nije HTML oznaka, već linija teksta koja se uvijek nalazi na samom početku i pruža pregledniku informaciju o inačici HTML-a kojim je stranica napisana. Potom slijedi <html> element kao unutar kojeg smještamo sve ostale elemente. Prvi element u tom nizu je <head> koji služi za navođenje skripti, stilova, meta podataka i naziva stranice te se sadržaj unutar tih oznaka ne prikazuje na stranici. Potom slijedi <body> element unutar kojeg se smješta sav sadržaj koji želimo prikazati na stranici, paragrafi, slike, liste, navigacija…




**Sl. 2.2.** *Osnovna struktura HTML elementa*

Prema slici strukturu HTML elementa možemo rasčlaniti na početnu oznaku, atribut oznake i njegovu vrijednost, sadržaj koji slijedi nakon početne oznake te završnu oznaku, iako nekolicina HTML elemenata nema završnu oznaku. Novi navedeni pojam je atribut koji može imati svaki HTML dokument, dok su kod nekih elemenata atributi nužni da bi se pravilno izvršili. Atributi pružaju dodatnu informaciju o elementu i navode se na početu početne oznake, npr. kod oznake za hiper linkove *<a href=“ https://www.google.com“>Google</a>*. U navedenom primjeru naveli smo href atribut unutar kojeg upisujemo URL stranice na koju ćemo biti preusmjereni ukoliko kliknemo na <a> oznaku. Neki od atributa navedeni su u tablici 2.1..

**Tab. 2.1.** *Atributi i njihovo objašnjenje*

| | |
|---|---|
| alt | Određuje alterativni tekst za sliku ukoliko ju nije moguće prikazati na stanici |
| class | Određuje klasu koju možemo dodijeliti elementima, služi za stiliziranje |
| id | Određuje posebnu identifikaciju koju smo dodijelili elementu |
| href | Određuje URL stranice za link |
| src | Određuje adresu na kojoj se nalazi slika |
| style | Određuje CSS stil elementa unutar HTML-a |
| title | Određuje dodatne informacije o elementu |

HTML uistinu sadrži velik broj oznaka koje korisniku pružaju mnoštvo mogućnosti, osim prethodno navedenih osnovnih oznaka u ovom će se dijelu rada proći kroz ostale oznake koje su često dio svake stranice. Za oblikovanje naslova unutar dokumenta koristimo oznake koje se sastoje iz 6 razina, a raspon im je od <*h1*> do <*h6*>. Prema važnoti pojedinih naslova koristimo odgovarajuće oznake te se za one najvažnije naslove uvijek koristi h1 oznaka koja je ujedno i najveće veličine. Nakon svakog naslova logičnim slijedom idu paragrafi za koje postoji <*p*> oznaka, ukoliko više različitih rečenica uokvirimo ovom oznakom između svake će automatski biti generiran prostor. HTML-om je također moguće tekst prikazati u obliku liste, to ćemo postići s <*ul*> ili <*ol*> oznakom gdje prva označava neorganiziranu listu, a druga organiziranu.

```
8   <ul>
9     <li>Prvi element</li>
10    <li>Drugi element</li>
11    <li>Treći element</li>
12  </ul>
13
14  <ol>
15    <li>Prvi element</li>
16    <li>Drugi element</li>
17    <li>Treći element</li>
18  </ol>
```

- Prvi element
- Drugi element
- Treći element

1. Prvi element
2. Drugi element
3. Treći element

**Sl. 2.3.** *Razlika između <ul> i <ol> lista*

Svaka lista načinjena je od stavki unutar <*li*> oznaka. Za odvajanje teksta tj. sadržaja stranice često se koristi <*div*> koji sam po sebi ne sadrži nikakve predefinirane stilove poput <p> oznake.

Pomoću ove oznake razdvajamo sekcije stranice kako bi im zasebno dodijelili određene stilove i lakše prepoznali od ostatka sadržaja stranice. Slično  oznaci, za podjelu stranice na dijelove možemo koristit i <section>. Čestom uporabom Bootstrap framework-a sve više viđamo <nav> oznaku kao dio stranice, čija je svrha definiranje navigacije koja korisnika vodi na različite dijelove stranice ili na neki drugi dokument. Forme su također zastupljen dio sadržaja, bilo da se radi o kontak formama ili pretplati na novosti, njihova je struktura nešto veća jer se unutar *<form>* oznake nalazi još HTML elemenata koji služe kao interaktivne kontrole za slanje sadržaja serveru.

```
41  <div class="container">
42    <form action="action_page.php">
43
44      <label for="fname">Ime</label>
45      <input type="text" id="fname" name="firstname" placeholder="Vase ime..">
46      <br>
47
48      <label for="lname">Prezime</label>
49      <input type="text" id="lname" name="lastname" placeholder="Vase prezime..">
50      <br>
51
52      <label for="category">Kategorija upita</label>
53      <select id="category" name="category">
54        <option value="australia">HTML</option>
55        <option value="canada">CSS</option>
56        <option value="usa">JavaScript</option>
57      </select>
58      <br>
59
60      <label for="subject">Poruka</label>
61      <textarea id="subject" name="subject" placeholder="Ovdje upisite poruku.." style="height:200px"></textarea>
62      <br>
63
64      <input type="submit" value="Submit">
65
66    </form>
67  </div>
```

**Sl. 2.4.** *HTML struktura forme*

Za kreiranje tablica koristimo *<table>* oznaku koja se sastoji od jedne ili više *<tr>*, *<th>* i *<td>* oznaka. Oznaka *<tr>* definira table row (eng. redak tablice), *<th>* definira table header (eng. zaglavlje tablice), a *<td>* definira table cell (eng. ćelija tablice). Na slici XY možemo vidjeti strukturu jednostavne tablice. Postoji i kompleksnija struktura tablice koja uključuje <thead> za grupiranje sadržaja zaglavlja te unutar ove oznake mora postojati barem jedan redak, a isto vrijedi i za <tbody> i <tfoot> oznaku. Razlika između ove tri oznake je u tome što se <thead> elementi nalaze na samom vrhu tablice, <tbody> elementi su središnji dio (tijelo) tablice, a <tfoot> je samo podnožje tablice. Uporabom ove tri dodatne oznak prikaz tablice se ne mijenja, no lakše je stilizirati tablicu ukoliko je podijeljena na ovaj način.

```
15    <div class="container">
16      <table style="width:50%">
17        <tr>
18          <th>Ime</th>
19          <th>Prezime</th>
20          <th>Godine</th>
21        </tr>
22        <tr>
23          <td>Ivan</td>
24          <td>Ivic</td>
25          <td>50</td>
26        </tr>
27        <tr>
28          <td>Ana</td>
29          <td>Anic</td>
30          <td>24</td>
31        </tr>
32        <tr>
33          <td>Lovro</td>
34          <td>Lovric</td>
35          <td>35</td>
36        </tr>
37      </table>
38    </div>
```

| Ime | Prezime | Godine |
|---|---|---|
| Ivan | Ivic | 50 |
| Ana | Anic | 24 |
| Lovro | Lovric | 35 |

**Sl. 2.5.** *Prikaz osnovnog HTML koda tablice i tablice u pregledniku*

Meta oznaku <meta> zapisujemo unutar <head> oznake čiji se sadržaj ne prikazuje na stranici. Meta oznaka daje podatke o HTML dokumenentu kao što su opis stranice, ključne riječi, autor dokumenta, datum zadnje modifikacije itd. Preglednici koriste te meta podatke tj. podatke o podacima te prema njima prikazuju i ponovno učitavaju sadržaj. Ova oznaka uvijek sadrži dva atributa, name i content. Jedna od ključnih meta oznaka koja se uvijek mora nalaziti u HTML dokumentu te služi kako bi se na svim uređajima stranica dobro prikazivala, zapisuje se kao: <meta name="viewport" content="width=device-width, initial-scale=1.0">

Slijedeća od oznaka koje također nalazimo unutar <head> oznake je <link>. Link oznaka koristi se najčešće za povezivanje s dokumentima koji sadrže CSS ili JS kod. Njegova je namjena povezivanje vanjskih datoteka s HTML dokumentom na kojemu trenutno radimo. Prilikom pisanja našeg koda određene linije možemo zakomentirati ili jednostavno staviti neku napomenu koja će nam koristit u budućnosti. Struktura komentara prikazana je slikom 2.6..

```
13  <!-- Ovo je komentar u jednom redu -->
14
15  <!-- Ovo je komentar
16         u dva reda -->
```

**Sl. 2.6.** *Komentar unutar HTML dokumenta*

Za razliku od meta oznake, <script> oznaka se može staviti unutar <head> ili na kraj <body> oznake. Služi za navođenje vanjskih skriptnih datoteka (JavaScript) ili se unutar <script> oznaka nalazi skriptni kôd.

```
5
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7       <link rel="stylesheet" type="text/css" href="style.css">
8
```

**Sl. 2.7.** *Povezivanje HTML i CSS dokumenta pomoću <link> oznakte, a <meta> oznaka govori pregledniku da prati širinu zaslona uređaja*

Oznaka <footer> koristi se za podnožje stranice u koju se smješta alternativna navigacija stranice, sponzorski logo-i, kontakt i adresa, autorska prava, copyright, mapa lokacije itd.

## 2.2. CSS

CSS (Cascading Style Sheets) je deklaracijski jezik pomoću kojeg određujemo izgled pojedinog elementa stranice. CSS možemo definirati unutar <style> oznake koju smještamo unutar <head> oznake, navođenjem svojsta style HTML elemenata ili pak kreiranjem posebne datoteke s ekstenzijom .css. Sintaksa CSS-a sastoji se od selektora i deklaracijskog bloka kao što je prikazano na slici.

**Sl. 2.8.** *Prikaz osnovne sintakse unutar CSS dokumenta*

Selektor pokazuje na HTML element koji želimo urediti, a unutar vitičastih zagrada nalazi se deklaracijski blok koji sadrži svojstvo i dodijeljenu vrijednost tom svojstvu. Deklaraciju uvijek zaključujemo s točkom i zarezom. Postoji više vrsta selektora no oni najosnovniji su selektor HTML elementa, id selektor (koristi id atribut koji smo dodijelili HTML elementu) te klasni selektor (koristi class atribut koji smo dodijelili HTML elementu). Selektore također možemo i grupirati ukoliko želimo da više različitih elemenata ima jednak stil bez da svaki put navodimo jednaki deklaracijski blok za pojedini HTML element.

CSS je koristan jer nam također omogućuje manipuliranje dimenzijama pojedinih elemenata s čime ostvarujemo responzivan dizajn stranice tj. elemente koje vidimo na svom web pregledniku dok koristimo desktop računalo, moguće je jednako fluidno i lijepo prikazati u drugim dimenzijama koje su prikladne za uređaje manjih rezolujcija i zaslona (mobiteli, tableti, laptopi). Recimo da želimo modificirati npr h1 oznaku koja predstavlja naš naslov odlomka. Potom otvorimo i zatvorim blok s vitičastim zagradama { } unutar kojih ćemo navoditi svojstvo koje želimo mijenjati npr. color: blue; gdje je color svojsto, a blue željena vrijednost. Kako tokom izrade dokumenta nećemo vjerojatno htjeti da nam se na npr. svaki naslov primjenjuje isto svojstvo, pojedinačnim naslovima ili bilo kojem drugom HTML elementu možemo pridružiti

poseban klasni atribut po kojemu će se međuseobno razlikovati. Za deklariranje atributa možemo koristiti HTML element sa klasnim atributom class="naslov1" ili pak s identifikaijskim elementom id="naslov1". Prilikom navođenja treba paziti koji smo od navedenih mogućnosti koristili jer kada zovemo klasni atribut unutar CSS dokumenta pišemo točku (.naslov1), a ako zovemo identifikacijski atribut pišemo hastag (#naslov1). Također veoma je bitno napomenuti da HTML i CSS dokument trebaju biti međusobno povezani kako bi se stil primijenio na HTML elemente, to se postiže tako da slijedeću liniju koda *<link rel="stylesheet" type="text/css" href="nazivCssDokumenta.css">* implementiramo unutar <head> oznaka u HTML dokumentu. Kao i kod HTML-a i u CSS-u možemo zakomentirati određene linije koda pomoću: /* Ovo je komentar */.

**Tab. 2.2.** *CSS selektori i njihovo objašnjenje*

| **Selektor** | **Objašnjenje** |
|---|---|
| * | *Univerzalni selektor*, stil će biti primijenjen na sve elemente stranice |
| #id | *Identifikacijski selektor*, stil se primjenjuje na sve elemente koji imaju određenu id vrijednost |
| .class | *Klasni selektor*, stil se primjenjuje na sve elemente koji imaju određenu klasu |
| X | *Selektor elementa*, X odabire bilo koju HTML oznaku |
| X Y | *Selektor nasljednika*, odnosi se na Y element koji se nalazi unutar X elementa |
| X > Y | *Selektor djeteta*, odabrani su samo oni elementi koji su direktnadjeca od X elementa |
| X + Y | *Susjedni selektor*, Y je element koji prvi slijedi iza X elementa |
| X[attribute] | *Atributni selektor*, samo oni HTML elemenit s određenim atributom |
| X:hover | *Pseudo selektor*, stil se primjenjuje ukoliko se mišem pređe preko X elementa (varijante ovog selektora su :visited, :active, :link |
| X:nth-child(n) | Selektor odabire one elemente koji su djeca X elementa, gdje n označava parametar, npr X:nth-child(2) – označava se svaki drugi element koji je dijete od X elementa |
| X:nth-of-type(n) | Selektor odabire elemente određenog tipa X, gsje je n parametar, npr X:nth-of-type(3) – označava svaki treći X element |

| | |
|---|---|
| X::after | Selektor dodaje sadržaj iza svakog X elementa, postoji i ::before varijanta |
| X[attribute=“value“] | Selektor odabire sve X elemente s određenim atributom |
| X[attribute~=“value“] | Selektor odabire sve X elemente s određenim atributom koji sadrži određenu vrijednost npr. neku riječ |
| X[attribute$=“value“] | Selektor odabire sve X elemente čiji atribut završava s određenom vrijednošću, npr. s nekom ekstenzijom |

**Tab. 2.3.** *CSS svojstva i objašnjenja*

| **Property (svojstvo)** | **Objašnjenje** |
|---|---|
| color | Određuje boju teksta, može se izraziti kao RGB, HEX vrijednost ili pomoću engleskog naziva određene boje |
| letter-spacing | Povećava ili smanjuje razmak između znakova unutar teksta, vrijednosti se izražavaju u cm, mm, px, pt,… |
| text-align | Određuje horizontalno poravnanje teksta unutar elementa, vrijednosti su left, right, center, justify |
| border | Određuje svojstva rubova, širinu, stil, boju |
| z-index | Određuje položaj elemenata po Z osi, ponekad se elementi preklapaju jedan preko drugoga, a z-indeks određuje koji će element biti prikazan ispred kojeg (onaj s većim iznosom će biti prvi prikazan ispred onog s manjim iznosom z-indeks-a) |
| background | Određuje sva svojstva pozadine elemenata u jednoj deklaraciji kao što su color, image, position, size, repeat… |
| box-shadow | Dodaje sjenu elementu, vrijednosti se dodjeljuju za x i y os, blur i boju sjene |
| display | Koristi se za određivanje prikaza elementa tj utječe na način kako se element prikazuje, kao blok, u redu itd. |

| | |
|---|---|
| float | Određuje položaj elementa unutar kontejnera (lijevo, desno) te se odnosi na elemente koji su prikazani kao block |
| font-family | Pomoću font-family svojstva možemo navesti specifičnu vrstu fonta za određeni sadržaj |
| font-weight | Određuje debljinu slova u tekstu |
| font-size | Određuje veličinu fonta |
| opacity | Određuje prozirnost elementa |
| padding | Definira prostor između sadržaja i njegovog ruba tj. prostor u elementu |
| margin | Određuje dodatan prostor oko elementa |

Također jedna od neizostavnih svojstava koja su uvijek prisutna su *padding* i *margine*. Za objašvanjanje margina i padding-a koristi se *model kutije*. CSS model kutije predstavlja kutiju koja obavija HTML element, a sastoji se od margina, rubova (borders), padding-a i sadržaja. Polazimo od sadržaja to je prostor gdje je prikazan tekst ili slika, padding definiramo oko sadržaja on je nevidljiv i predstavlja udaljenost sadržaja od ruba (border), dok margine predstavljaju nevidljivi prostor koji definiramo izvan ruba. Marginama većinom jednu vrstu elementa odjeljujemo od druge vrste elementa. Uzvešni to u obzir, prava širina i dužina HTML elemenata se računa tako da npr. za širinu uzmemo širinu slike te na to dodamo lijevi i desni padding, lijevi i desni rub te lijevu i desnu marginu. Isto bi računali visinu samo bi u obzir uzeli gornju i donju marginu, padding i rub.

**Sl. 2.9.** *Model kutije koji prikazuje odnos sadržaja, padding-a, ruba i margina*

## 2.3. Bootstrap i JavaScript

*Bootstrap* je najpopularniji HTML, CSS i JavaScript framework za razvoj responzivnih stranica, također je besplatan i open source što znači da svaki korisnik postojeći kôd može prilagoditi sebi. Bootstrap uključuje mnoštvo HTML i CSS baziranih predložaka za forme, gumbe, navigaciju, responzivne slike, karusel itd. Bootstrap možemo gledati kao biblioteku svih sadržaja koji su potrebni za funkcionalnost svake stranice. Recimo da dizajner unutar svoje stranice želi uklopiti navigacijsku traku, ukoliko koristi Bootstrap može se poslužiti predefiniranom navbar klasom koja će toj navigacijskoj traci pružiti neki osnovni izgled (pozadinska boja, veličina fonta, raspored linkova, smanjivanje trake na određenim veličinama). Također dizajner može maknuti, prepisati postojeći kôd ili pak dodati još nešto što nije navedeno unutar Bootstrapa. Jednostavan je za uporabu čak i ukoliko je osoba upoznata samo s osnovama CSS-a i HTML-a te ga podržavaju svi moderni preglednici. Kako bi započeli rad s Bootstrapom potrebno je otići na stranicu https://getbootstrap.com/docs/3.3/getting-started/. Nakon što smo skinuli Bootstrap dokumente, smjestimo ih potom u mapu gdje se naš projekt nalazi te unutar <head> oznaka upišemo slijedeće linije koda prikazane slikom

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
```

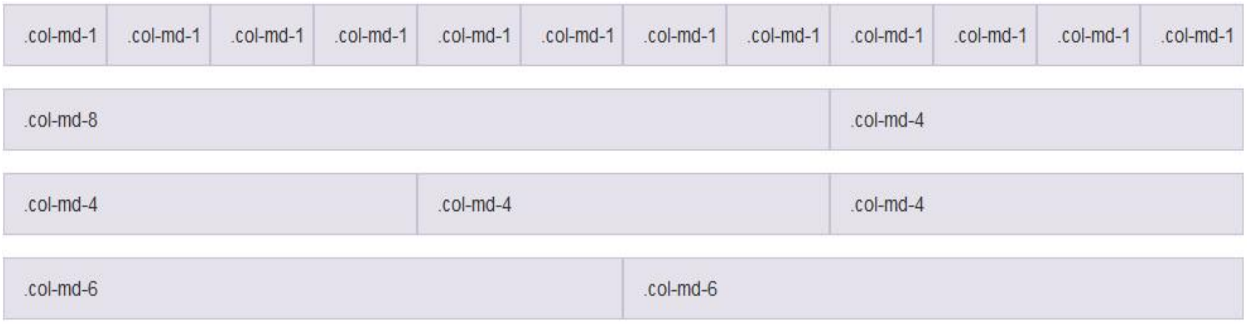
**Sl. 2.3.2.** *Implementacija Bootstrap dokumentacije unutar HTML dokumenta*

Osim uputa kako implementirati Bootstrap unutar projekta, službena Bootstrap stranica nudi i detaljna pojašnjenja za svaki pojedini element koji dolazi s ovim frameworkom. Odlaskom na https://getbootstrap.com/docs/3.3/components/ korisnik može odabrati naziv komponente koja ga zanima te vidjeti od čega se sastoji te kako određenu klasu pozvati. Bootstrap su razvili Mark Otto i Jacob Thornton, dva programera koja su radila za Twitter. Prva inačica njihovog projekta izašla je 2011. godine u kolovozu, a 2014. postala je najpopularniji projekt na GitHub-u. U slijedećim odlomcima biti će navedeni i opisani Bootstrap elementi koji se najviše koriste prilikom izrade projekata.

Započnimo s Bootsrap *gridom* (mrežom), sastoji se od dvanaest stupaca koji čine 100% širine stranice. Stupci su responzivni te će se ponašati u skladu s veličinom zaslona uređaja. Stupce je moguće grupirati kako bismo dobili veće, no njihov zbroj i dalje mora iznositi dvanaest. Mrežu čine četiri klase:

1. **col-xs –** za mobilne uređaje
2. **col-sm –** za tablete
3. **col-md –** za laptope manjih zaslona
4. **col-lg –** za laptope i desktop računala

Svaka od ovih klasa funkcionira na isti način, upravlja veličinom elementa i mijenja njegovu veličinu prema veličini zaslona na kojemu se prikazuje. Npr. ukoliko odaberemo klasu col-sm na zaslonima koji su manji ili jednaki 576px, veličina div elementa maksimalno može doseći 540px. Jednostavnije rečeno, na zaslonima koji su veličine tableta pa do zaslona desktop računala stupci će se povećavati i biti će prikazani u jednom redu, jedan pored drugoga dok će se na mobilnim uređajima ti stupci posložiti jeda ispod drugoga.

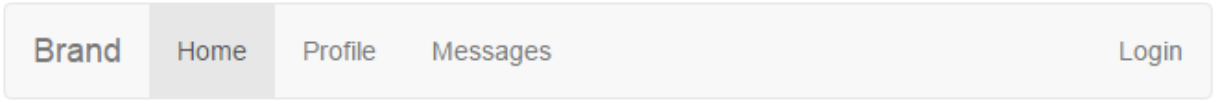



**Sl. 2.10.** *Primjer Bootstrap stupaca po veličini, na slici se radi o md stupcima za laptope manjih zaslona*

*Navbar* komponenta omogućuje kreiranje responzivne navigacijske trake. Osim CSS-a zahtjeva i JavaScript plug-in. Promjenom veličine zaslona poput svih responzivnih elemenata i navbar mijenja svoju veličinu, no navbar ima dodatnu mogućnost koja se zove collapse. Funkcionalnost collapse se definira tako da unutra HTML elementa unesemo *data-toggle="collapse"*. Collapse će raditi samo ako smo implementirali JS i JQuery *CDN (Content Delivery Network). CDN* je besplatan i javan sadržaj koji korisnik može dokučiti s Bootstrap servera, a on uključuje CSS i JS datoteke. Kada bi se kreirani navbar učitao na ekranima manjih uređaja kao što su mobiteli, pomoću collapse funkcionalnosti sadržaj navbar-a bio bi sakriven. Kako bi se prikazao linkovi koji služe kao navigacija na stranici, bilo bi potrebno tapnuti ili kliknuti na dio unutar navigacijske trake te bi se tada u obliku padajućeg izbornika prikazao uneseni sadržaj liste koji predstavlja linkove za navigaciju. Time se postiže preglednost i funkcionalnost na manjih zaslonima. Navigaciji se može dodati nekoliko dodatnih značajki i klasa, ukoliko želimo da uvijek bude na vrhu/dnu stranice tijekom skrolanja dodajemo klasu *.navbar-fixed-top ili .navbar-fixed-bottom,* prilikom dodavanja ove klase valja naglasiti da se mora dodati padding-tom/bottom body elementu kako navicagija ne bi prelazila preko sadržaja. Zadano boja navigacijske trake je bijela, no to se može obrnuti pomoću klase *.navbar-inverse,* također možemo dodati tražilicu, ikonu branda ili pak koristiti određenu klasu za poravnjavanje sadržaja navigacije lijevo (*.navbar-left*) ili desno (*.navbar-right*).

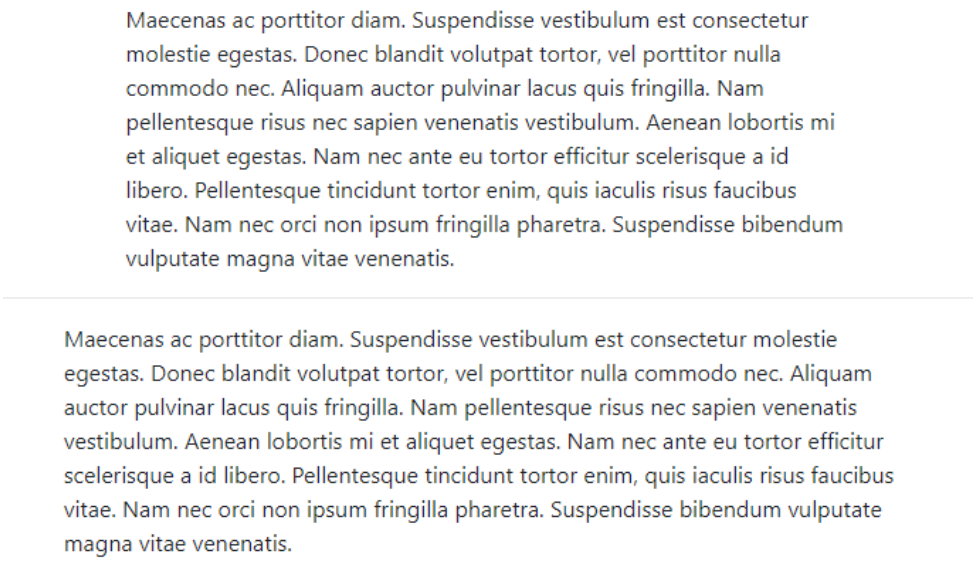
```html
<nav class="navbar navbar-default">
    <!-- Toggle funkcionalnost koja se pojavljuje na manjim ekranima -->
    <div class="navbar-header">
        <button type="button" data-target="#navbarCollapse" data-toggle="collapse" class="navbar-toggle">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
        </button>
        <a href="#" class="navbar-brand">Brand</a>
    </div>
    <!-- Linkovi navigacaije i ostali sadržaj -->
    <div id="navbarCollapse" class="collapse navbar-collapse">
        <ul class="nav navbar-nav">
            <li class="active"><a href="#">Home</a></li>
            <li><a href="#">Profile</a></li>
            <li><a href="#">Messages</a></li>
        </ul>
        <ul class="nav navbar-nav navbar-right">
            <li><a href="#">Login</a></li>
        </ul>
    </div>
</nav>
```

**Sl. 2.11.** *HTML kôd za implementaciju Bootstrap komponente – navbar*




**Sl. 2.12.** *Prikaz navigacije u pregledniku*

*Container* klasa je jako važna prilikom korištenja Bootstrap grid-a. Kontejner je običan div element kojemu je dodijeljena klasa *.container*. Služi kao omotač redaka i stupaca mreže stranice. Ima predefiniran širinu i margine no responzivan je, što znači da mu se širina mijenja ovisno o veličini zaslona. Ukoliko želimo da kontejner ne bude predefiniran već da mu je širina uvijek maksimalna, što bi značilo da uvijek zauzima maksimum mjesta stranice i nema margine, tada koristimo klasu -*.container-fluid.*

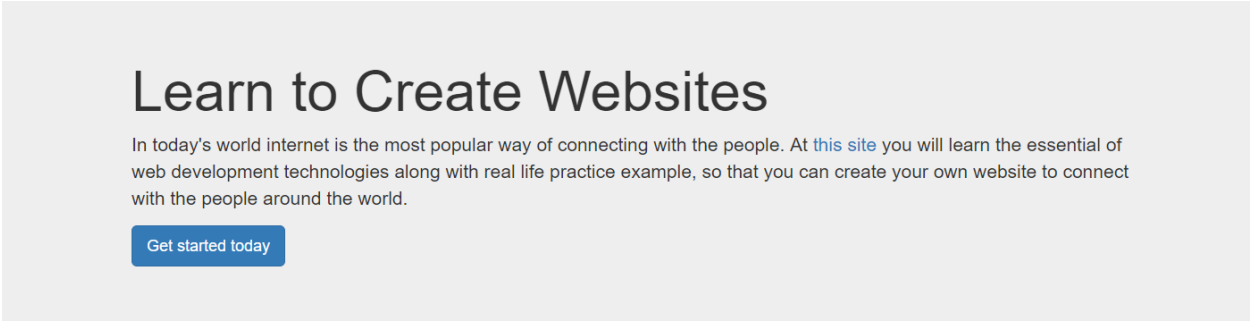



**Sl. 2.13.** *Prvi odlomak smješten je unutar .container, a drugi unutar .container-fluid klase*

Za marketinške svrhe te kao idealna prezentacija proizvoda ili tematike stranice koristi se komponenta *jumbotron.* Jumbotron je također običan div element kojemu je pridružena klasa *.jumbotron.* Jumbotron se najčešće koristi kako bi velikom slikom privukli i zainteresirali posjetitelje stranice. Unutar jumbotrona se također može staviti gotovo svaki HTML element osim slike, kao što je tekst, gumbi, forme… Ukoliko želimo da se jumbotron proteže cijelom širinom stranice, tada ga ne treba smjestiti unutar već izvad div elementa s klasom container.

```
22      <div class="jumbotron">
23        <div class="container">
24          <h1>Learn to Create Websites</h1>
25          <p>In today's world internet is the most popular way of connecting with the people.
26             At <a href="#" target="_blank">this site</a> you will learn the essential of web development
27             technologies along with real life practice example, so that you can create your own website to
28             connect with the people around the world.</p>
29          <p><a href="https://www.tutorialrepublic.com" target="_blank" class="btn btn-primary btn-lg">Get started today</a></p>
30        </div>
31      </div>
32
```

**Sl. 2.14.** *HTML kôd za jumbotron koji se proteže cijelom širinom stranice*




**Sl. 2.15.** *Jumbotron prikazan u pregledniku*

*Forme* su također važan dio interaktivnog sadržaja svake stranice. Bootstrap i za to ima komponentu tj. klasu koja se zove .form-group. Ovom se klasom dodaje predefinirani stil elementima forme kao što su <input>, <textarea> i <select>. Navedeni se HTML elementi omotaju div elementom s klasom .form-group te se svakom od elemenata kao što su <input>, <textarea> i <select> doda klasa .form-control. No ukoliko želimo da su neki elementi forme u istom redu te ćemo komponente dodatno omotati s div elementom klase .form-inline.

```
23    <div class="container>"
24      <form>
25          <div class="form-group">
26              <label for="inputEmail">Email</label>
27              <input type="email" class="form-control" id="inputEmail" placeholder="Email">
28          </div>
29          <div class="form-group">
30              <label for="inputPassword">Lozinka</label>
31              <input type="password" class="form-control" id="inputPassword" placeholder="Lozinka">
32          </div>
33          <div class="checkbox">
34              <label><input type="checkbox">Zapamtii me</label>
35          </div>
36          <button type="submit" class="btn btn-primary">Login</button>
37      </form>
38    </div>
```

**Sl. 2.16.** *Html kôd Bootstrap komponente – form.group*

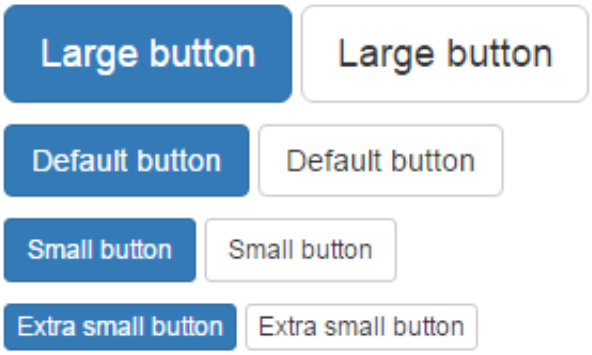
Još jedan interaktivni element su *gumbi*, Bootstrap nudi različite klase koje omogućuju biranje boje i veličine gumba ovisno o njihovoj svrsi. Zanimljivo je to što gumb klasu možemo primijeniti na bilo koji element, no najčešće se koristi za <a>, <input> i <button> elemente. U tablici su prikazane vrste gumba i klase kojima se određene vrste definiraju.

**Tab. 2.4.** *Prikaz Bootstrap gumba po vrsti klase*

| Vrsta gumba | Prikaz |
|---|---|
| btn btn-default | Default |
| btn btn-primary | Primary |
| btn btn-info | Info |
| btn btn-success | Success |
| btn btn-warning | Warning |
| btn btn-danger | Danger |
| btn btn-link | Link |

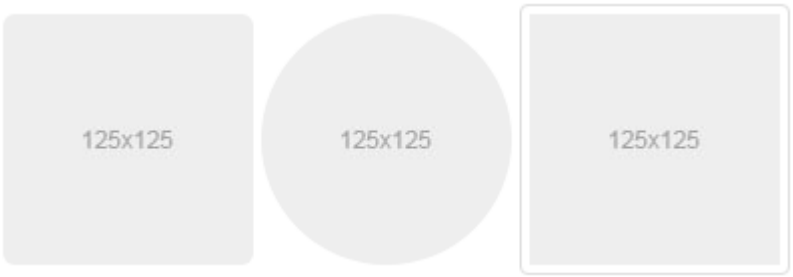
Što se veličine gumba tiče postoje četiri između kojih možemo birati, a to su:

1. btn-lg
2. btn-default ili btn-primary
3. btn-sm
4. btn-xs




**Sl. 2.17.** *Dostupne veličine Bootstrap gumba*

Pomoću Bootstrap-a također možemo oblikovati *slike*, koje su veoma važan dio svake responzivne stranice. Bitno je da se slike ponašaju u skladu s promjenom veličine zaslona, želimo da naše slike budu pregledne i skalabilne. Osim što predefinirana Bootstrap klasa čini slike responzivnim, koristeći određenu klasu slika može poprimiti izgled kruga, može biti zaobljenih rubova ili biti prikazana kao thumbnail (umanjena verzija slike ili videa). Da bismo postigli responzivnost slika, img oznaci treba dodati klasu *.img-responsive*, time će slika dobiti *max-width:100%* te *height:auto* što znači da će mijenjati veličinu u skladu s elementom u kojem se nalazi prilikom promjene veličine zaslona.




**Sl. 2.18.** *Oblici dostupni uporabom Bootstrap klasa za slike - .img-rounded, .img-circle i .img-thumbnail*

**Sl. 2.19.** *Primjer slika unutar roditelj elementa od kojih je na drugoj korištena .img-responsive klasa*

*JavaScript (JS)* je uz Bootstrap također veoma popularan kod web dizajnera. Može se definirati kao programski jezik HTML-a, objektno je orijentiran i namijenjen je stvaranju raznih interaktivnih efekata unutar web preglednika. Ukratko rečeno pomoću JavaScript-a možemo pohraniti vrijednosti unutar varijabli, manipulirati tekstom, prikazati datum, vrijeme, animacije, kreirati kalendar, za određene interakcije na stranici napraviti određeni događaj, npr. iskočni prozor/poruku i sl. Sve zapravo što ne bismo mogli s HTML-om i CSS-om, možemo s JavaScript-om. Uz JS, poznata je i uporaba JavaScript biblioteke pod nazivom *JQuery*. *JQuery* omogućuje lakšu uporabu JavaScript-a, sadrži većinu popularnih zadaća koje bi mogli implementirati unutar web stranice no uz puno linija koda te nam omogućuje njihovo pozivanje tako da se za pozivanje koristi jedna ili manje linija koda. Kao što vrijedi za CSS tako se i JavaScript i JQuery moraju implementirati unutar HTML koda kako bi njihove funkcionalnosti bile vidljive na stranici. Stoga, JavaScript kod možemo pisati unutar <*script*> HTML oznaka

unutar HTML dokumenta. Preporuča se navođenje JavaScript-a unutar <*body*> HTML oznaka, no na samome dnu jer se time pospješuje učitavanje stranice.

Također je moguće kreairati i vanjsku .js datoteku u kojoj ćemo pisati samo isključivo JavaScript kod. Razlog zašto se (ukoliko JS kod stavljamo direkt u HTML dokument) <script> oznaka s JS kodom treba staviti pri dnu HTML dokumenta je taj što se HTML dokumenti unutar preglednika prikazuju po redu kako su zapisani u dokumentu. Ukoliko bi na vrh stavili JS kôd koji treba utjecati na funkcionalnost nekog elementa iz sredine sadržaja stranice, kôd bi mogao ne raditi ispravno ili uopće. Osnovu koda čini definiranje varijabli, njih koristimo da bismo pohranili neku vrijednost a deklariramo ključnom riječi *var* te navedemo željeno ime npr. ***var Sumiranje;*** te obavezno na kraj dodajemo „**;**“.Vrijednost varijable može biti jednaka stringu, broju, polju, boolean vrijednosti ili objektu. Ukoliko želimo zakomentirati jednu ili više linija koda možemo koristi način komentiranja kao u CSS-u ili pomoću dvije kose crte ukoliko komentiramo samo jednu liniju koda:

a) /*
   Ovo je komentar u jednom redu
   */
*b)* // Ovo je također komentar u jednom redu

JS može prikazati podatke na četiri različita načina:

1. Upisivanjem unutar HTML elementa pomoću innerHTML
2. Upisivanjem unutar HTML izlaza pomoću document.write()
3. Upisivanjem unutar iskočnog prozora (alert) pomoću window.alert()
4. Upisivanje unutar konzole preglednika pomoću console.log()

Kôd kojim želimo manipulirati određenim HTML elementima se najčešće zapisuje kao *funkcija, događaj* (event) ili *uvjet* (condition).

*Funkcija* predstavlja neki blok koda koji će izvršiti određenu radnju kada tu funkciju pozovemo. Da bismo definirali funkciju trebamo kôd započeti s ključnom riječi function, dodijeliti joj ime te unutar okruglih zagrada () dodati parametre. Nakon okruglih zagrada otvaramo blok za upisivanje instrukcija pomoću para vitičastih zagrada {}. Funkciju se može izvršiti tako da se određena aktivnost dogodi (korisnik klikne na neki dio HTML-a), pozivanjem ili se može automatski izvršiti. Unutar funkcije često nailazimo i na return izraz, u trenutku kada funkcija dođe do te izjave ona prestaje s izvršavanjem. Return izraz govori pregledniku da vrati vrijednost varijable

izvan funkcije kako bi se kasnije ta ista vrijednost mogla koristiti. Važno je navesti return izraz jer se inače varijable unutar blok kôda funkcije koriste samo unutar te funkcije, a ne izvan nje ukoliko nismo naveli return izraz. JS koristi dvije vrste varijabli, *lokalne* i *globalne*.

*Lokalne* varijable su one koje je moguće koristi samo unutar određene funkcije i samo ta funkcija unutar koje se nalaze ima pravo na korištenje te varijable. One se preiranju prilikom izvršavanja funkcije i brišu nakon što je funkcija izvršena do kraja. Ukoliko želimo definirati *globalnu* varijablu, definirat ćemo njeno ime i vrijednost izvan funkcije.

```
function name(parameter1, parameter2, parameter3) {
    code to be executed
}
```

**Sl. 2.3.13.** *Sintaksa JS funkcije*

```
function setLocation(city) {
  var country = "France";

  function printLocation() {
    console.log("You are in " + city + ", " + country);
  }

  printLocation();
}

setLocation ("Paris");
You are in Paris, France
```

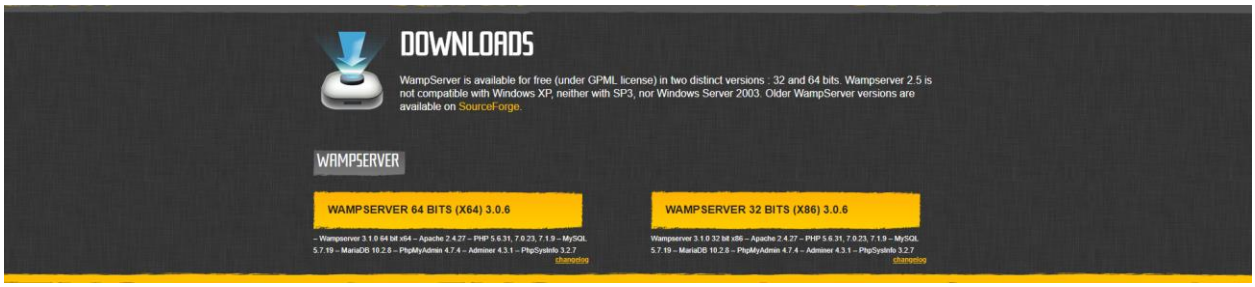
**Sl. 2.20.** *JS funkcija koja ispisuje string unutar Google konzole*

Pomoću *događaja* doprinosimo interaktivnosti stranice. Najjednostavniji primjer JS događaja bio bilo bi klikanje korisnika na određeni element stranice. U tablici ispod navedeni su tipovi događaja i pripadajući primjeri. *Uvjete* pak koristimo da bismo neku radnju izvršili ovisno o odrešenim uvjetima. U JS-u se najčešće pojavljuju slijedeći uvjeti koje također nazivamo *petljama*:

a) *if uvjet* - ukoliko je izjava točna izvršit će se blok kôd
b) *else uvjet* - ukoliko je izjava netočna izvršit će se drugi blok kôd
c) *else if* – određuje se drugi uvjet koji se mora testirati ukoliko je prva izjava netočna
d) *switch* – određuju se alternativni blok kôdovi koji se trebaju izvršiti

## 2.4. PHP i SQL

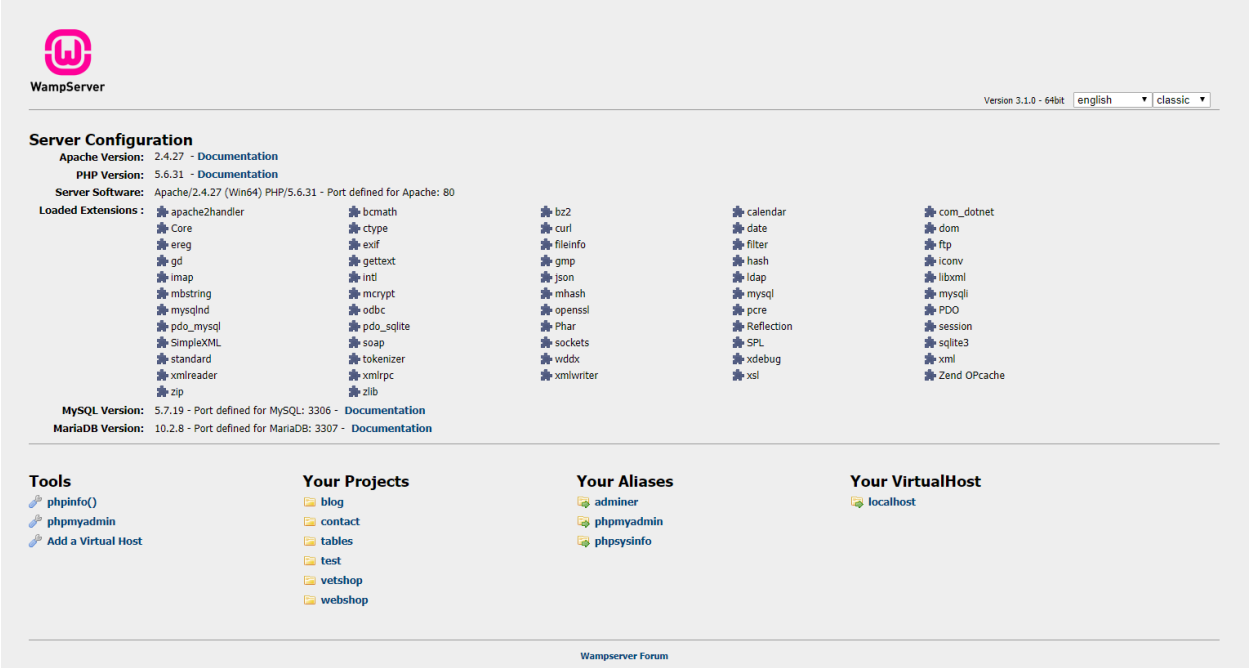
*PHP (Hypertext predprocessor)* je programski jezik koji je danas uveliko raširen u području razvoja web stranica. Sintaksno je vrlo sličan programskom jeziku C te pomoću PHP-a možemo implementirati funkcionalnosti kao što je dohvaćanje i upisivanje podataka u bazu, kontrolirati pristup korisnika stranici (logiranje, registracija), prikupljanje podataka iz formi, enkripcija podataka itd. Programske naredbe PHP-a uključujemo unutar HTML oznaka, početak našeg koda naglašavamo s <*?php*> a kraj s *?*> oznakom. No prije same upotrebe baze i PHP-a, korisniku je potrebno sučelje preko kojeg će se povezivati na bazu te koje će omogućiti izvršavanje PHP koda na serveru, to će se postići uporabom programa kao što je *WampServer*. Instalacijom WampServera korisnik desnim klikom na ikonu koja se nalazi u donjem desnom kutu alatne trake Start može pristupiti *phpMyAdmin* stranici koja će mu omogućiti kreairanje baze te lokalni pristup kreairanim php skriptama. Ukratko rečeno WampServer je softver koji je neophodan za testiranje i pokretanje php skripti koje korisnik kreira te mu omogućuje povezivanje s bazom i to sve na lokalnoj mreži, bez objavljivanja stranice na nekom od Internet servera. Proces instalacije sučelja za rad s PHP-om i bazom podataka prikazan je na slikama 2.21-2.25..




**Sl. 2.21.** *Odlaskom na stranicu www.wampserver.com/en/ moguće je besplatno skinuti software*

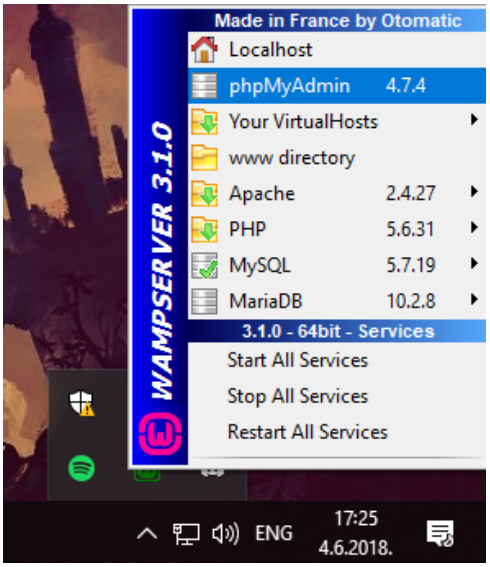
Namještanje WampServer sučelja je jednostavno, instalacija je vrlo brzo gotova i ne zahtjeva nikakve posebne uvjete. Po završetku instalacije mapa pod nazivom *wamp* nalazit će se na našem C disku. Unutar wamp mape treba odabrati mapu pod nazivom *www* te unutar nje kreirati vlastitu mapu u koju ćemo smjestiti projekt.

Da bismo pristupili našem projektu preko lokalnog servera, unutar preglednika ćemo upisati *„localhost“*. WampServer uspješno je podešen i moguće je prikazati projekt ukoliko u pregledniku vidimo slijedeći prikaz:




**Sl. 2.22.** *Prikaz ispravno podešenog lokalnog servera pomoću WampServer software-a*

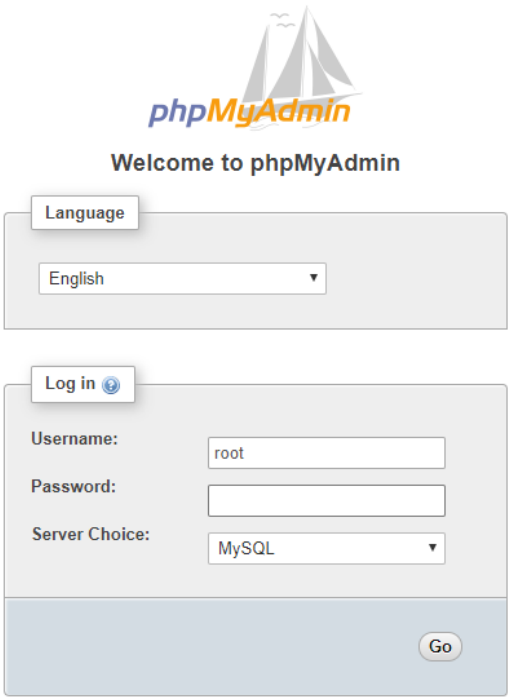
WampServer-om implementirali smo Apache, MySQL, PHP i phpMyAdmin na naše računalo koji su potrebni za izradu dinamičke web stranice. Ukoliko želimo započeti s izradom baze prije implementiranja php koda unutar projekta, pokrenut ćemo preko WampServer-a phpMyAdmin.



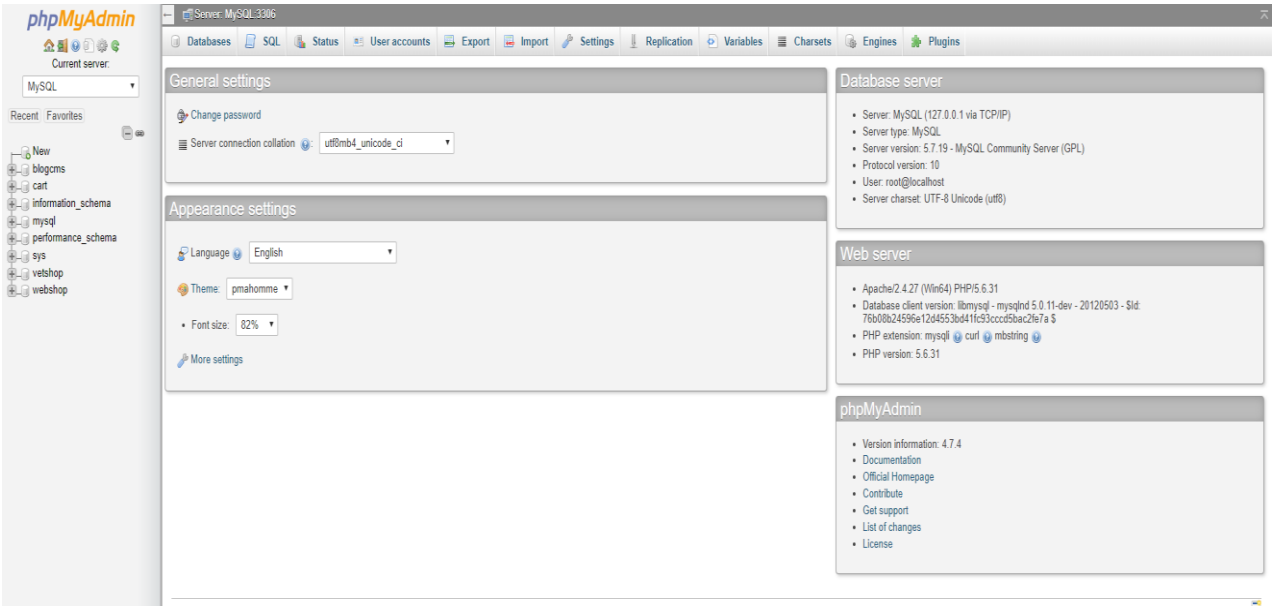

**Sl. 2.23.** *Prikaz pristupa phpMyAdmin-u preko WampServer-a*

Nakon što smo kliknuli na phpMyAdmin naš će nas preglednik odvesti na adresu localhost/phpmyadmin gdje je potrebno upisati podatke za logiranje koji su unaprijed definirani.

Username je uvijek postavaljen na „root“, dok je polje za lozinku prazno ili se može upisati unutar *config.inc.php* datoteke vlastita lozinka za buduće korištenje.




**Sl. 2.24.** *Prikaz pristupa phpMyAdmin-u*




**Sl. 2.25.** *Prikaz sučelja phpMyAdmin-a*

Jedne od najčešće korištenih funkcionalnosti PHP-a prilikom izrade dinamičkih web stranica su *$_GET / $_POST* metode, *$_SESSION*, *$_FILES* te *HTTP* funkcije, iako PHP nudi mnogo više. *$_GET i $_POST* se koriste za prikupljanje podataka iz formi, obje metode kreiraju polje koje sadrži key/value parove, ključevi su imena polja formi a vrijednosti su one koje je korisnik upiso u polja forme. *$_GET* najčešće koristimo kada šaljemo neosjetljive podatke, kada bi korisnik pritisnuo gumb podaci koje je upisao u određena polja bili bi vidljivi u URL adresi. $_GET je uredu koristiti ukoliko trebamo dohvatiti id neke objave jer nam je id potreban kako bismo pravilno linkali gumb na svaku pojedinačnu objavu iz baze podataka. *$_POST* funkcionira kao i $_GET, no upotrebljava se u formama češće jer podaci proslijeđeni preko forme nisu vidljivi svima te nema ograničenje koliko informacija može poslati kao što $_GET ima. *$_FILES* se upotrebljava u PHP sustavu za uploadanje datoteka, npr. slika, pomoću $_FILES možemo dohvatiti ime, vrstu, veličinu datoteke. Te su informacije korisne kada koristimo funkciju za premještanje uploadanih datoteka na novu lokaciju, to se koristi u stiuacijama kada odaberemo neku sliku s našeg računala, uploadamo ju preko forme, a ona će biti premještena na npr. mjesto gdje se naš projekt nalazi. *$_SESSION* varijabla se koristi kada želimo imati podatke kojima mogu pristupiti različite stranice koje čine jednu web stranicu. Sesijom spremamo informacije u varijable kao što su korisničko ime, email, lozinka. Ovaj način pohrane informacija koristan je ukoliko određenim stranicama mogu pristupiti samo registrirane osobe, unutar sesije će biti pohranjeni podacima logirane osobe koja ima pristup određenoj stranici sve dok se ta sesija ne uništi tj. dok se korisnik ne odjavi ili ugasi preglednik.

Zadnja tehnologija koja će se koristiti u izradu ovog rada obuhvaća uporabu *SQL-a* te izradu *baze podataka*. *Bazu podataka* možemo definitari kao kolekciju podataka, a podatke koje upisujemo u nju možemo zamisliti kao tablicu s različitim atributima kao nazivima pojedinih stupaca koji opisuju podatak. Svaki je atribut podatka potom opisan s odgovarajućim formatom. Npr. u bazi želimo stvoriti zapis pomoću kojeg ćemo moći ovisno o vrsti životinje saznati koliko je takvih životinja upisano, njihov spol, dob.. Tada bismo stvorili zapis koji bismo nazvali „životinja“, odredili bi atribute tj. stupce te tablice ovim redom, ID životinje koji je jedinstven i ukoliko unutar tablice imamo 2 životinje iste vrstse i dobi opet bismo ih prema ID-u znali prepoznati. ID bi bio zapravo broj (INT) koji bi se definirao prilikom unosa pojedinog zapisa, npr ako bi imali 5 zapisa, ID bi se kretao od 1-5. Nakon ID-a, mogli bi dodati stupac „vrsta“ (format bi bio STRING), „dob“ (INT) te spol (CHAR). No kako znamo koji format koristiti? Jednostavno, ovisno o tome kakav podatak želimo definirati, brojčani, tekstualni ili pak jedan ili dva simbola.

**Tab. 2.5.** *Prikaz nekih od formata/tipova podataka koji se mogu unositi u bazu*

| **Tip podatka** | **Opis** |
|---|---|
| CHAR(n) | fiksna duljina zapisa, max. 8000 simbola |
| VARCHAR(n) | varijabilna duljina zapisa, max. 8000 simbola |
| INT | cjelobrojni zapis broja od -2^31 do 2^31-1 |
| DECIMAL(p,s) | gdje je p ukupan broj znamenki (od 1 do 38), a s broj decimalnih znamenki (o <= s <= p) |
| DATETIME | od 01.01.1753 do 31.12.9999 |
| SMALLDATETIME | od 01.01.1900 do 06.06.2079 |
| BOOLEAN | pohranjuje TRUE ili FALSE vrijednost |

Najpopularniji sustav za definiranje baze i podataka unutar nje je ***SQL ( Structured Query Language).*** SQL je jezik pomoću kojeg upravljamo vezama tj. relacijama između podataka unutar baze. Upravljanje tj. manipulacija podacima unutar baze moguća je pomoću query-a tj. upita s kojima možemo dohvatiti neki podatak iz baze, unijeti novi, ažurirati postojeći ili pak obrisati jedan od unosa. Slijedećom tablicom biti će prikazane osnovne i najvažnije SQL naredbe.

**Tab. 2.6.** *Naziv i opis SQL naredbi koje koristimo za kreiranje upita prilikom rada s podacima*

| **Naziv naredbe** | **Uporaba** |
|---|---|
| SELECT | Dohvaćanje podataka iz BP |
| UPDATE | Ažuriranje podataka iz BP |
| DELETE | Brisanje podataka iz BP |
| INSERT INTO | Upisa novih podataka u BP |
| CREATE DATABASE | Kreiranje nove BP |
| ALTER DATABASE | Dodavanje, brisanje ili prepravljanje kolumna u postojećoj BP |
| CREATE TABLE | Kreiranje nove tablice |
| ALTER TABLE | Dodavanje, brisanje ili prepravljanje kolumna u postojećoj tablici |
| DROP TABLE | Brisanje tablice |
| CREATE INDEX | Kreiranje indeksa (pomoću indeksa je lakše tražiti podatke unutar tablica) |

| | |
|---|---|
| DROP INDEX | Brisanje indeksa |

Podatak koji nam koristi za identifikaciju upisa unutar baze podataka naziva se *primary key*, a za povezivanje različitih tablica preko upisa koji se nalaze u njima naziva se *foreign key*. *Primary key* se uvijek definira prilikom unosa podataka u tablicu te samo jedan element tablice može ujedno biti primary key. U većini slučajeva primary key je identifikacija (ID) svakog unosa te ga možemo postaviti da se automatski inkrementira svakim novim unosom.

*Foreign key* je pak onaj ključ pomoću kojeg povezujemo tablice unutar baze podataka. Recimo da imamo dvije tablice, gdje je u prvoj primary key ID osobe, a u drugoj ID narudžbe. Ukoliko bi htjeli povezati te dvije tablice tako da prema broju narudžbe znamo kojoj osobi pripada bez zasebnog pregleda tablica, trebamo stvoriti foreign key. To znači da će druga tablica za narudžbe sadržavati redak koji će se zvati ID osobe te će biti naznačen kao foreign key tablice gdje su navedeni podaci naručitelja.

# 3. POSTIZANJE RESPONZIVNOSTI

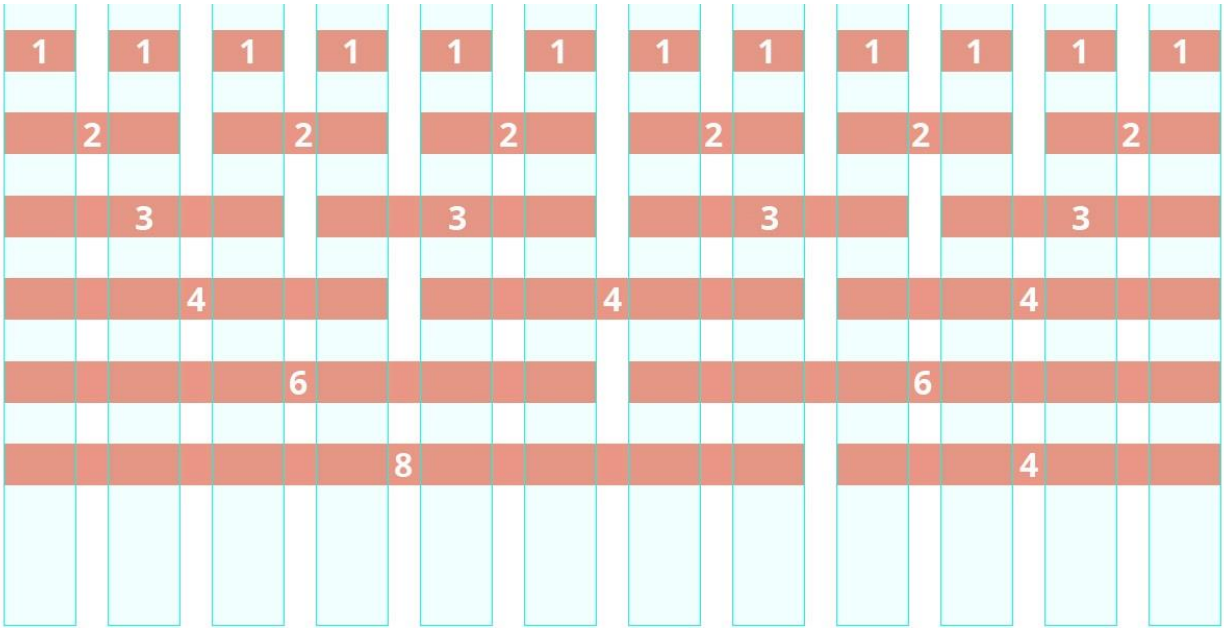
Sve većom raširenošću uporabe Interneta i moderne tehnologije u različitim aspektima poslovanja kao i u neformalnom dijelu života javila se i veća uporaba ostalih uređaja uz računala. Zbog praktične primjene mobilni uređaji, tableti te laptopi postali su važan dio svakodnevice ljudi koji u bilo kojem trenutku žele imati pristup informacijama s Interneta. Upravo zbog sve veće primjene tih uređaja nepraktično je nastaviti s izradom stranica fiksne veličine tj. stranica čiji elementi zauzimaju određen broj piksela na zaslonu uređaja. Elementi fiksne veličine izgledati će drukčije na uređajima različitih rezolucija, jednostavno nije moguće element od npr. 500px širine prikazati jednako na zaslonu stolnog računalu i na mobilom uređaju čiji zaslon seže do 375px širine. Elementi koji bi imali veću širinu nego što je širina određenog uređaja ne bi bili u cjelosti prikazani, već bi bili dijelom odsjećeni.

Koristiti responzivni dizajn prilikom izrade stranice zapravo znači izraditi stranicu koju će biti moguće prikazati na svim veličinama zaslona uređaja. Responzivni elementi će se smanjiti ili povećati ovisno o zaslonu, no neće se javiti problem djelomičnog prikazivanja već će korisnici ostalih uređaja kao i korisnici stolnih računala imati pristup i pregled svega što stranica nudi. Ovisno na kojem se zaslonu prikazuju određeni će se elementi poredati jedan ispod drugog, navigacija stranice će biti padajuća, veličina fonta će se smanjiti itd. Sam pojam responzivnog dizajna osmislio je Ethan Marcotte, on je također i autor knjige pod nazivom Responsive Web Design. Zahvaljujući Ethanovom rješenju više nije posve nužno izrađivati više različitih verzija iste stranice koja će se prikazivati na različitim uređajima. Kada govorimo o responzivnom dizajnu naglasak stavljamo na nekoliko važnih elementa, a to su: raspored elemenata unutar *grida* (eng. mreže), *media queries* i *dimenzionalno prilagodljive slike.*

**Sl. 3.1.** *Prikaz responzivne stranice na različitim uređajima*

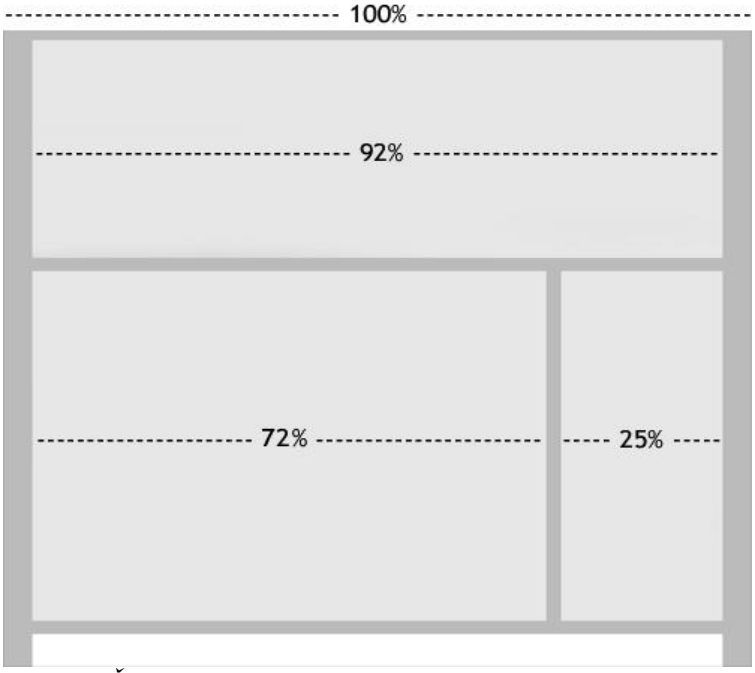
## 3.1. Grid (mreža) stranice

Grid (eng mreža) osnovna je struktura svake stranice, mrežu zapravo čini skupina stupaca koji dijele stranicu na manje dijelove. Koristeći mrežni pregled stranice lakše je smjestiti određene elemente i pozicionati ih tako da budu pregledni korisniku. Najčešće se radi o dvanaest stupćastom mrežnom pregledu stranice, gdje svih 12 stupaca čine 100% ukupne širine stranice.




**Sl. 3.2.** *Prikaz Bootstrap dvanaest-stupćastog grid-a (mreže)*

Prilikom pozicioniranja elemenata na stranici važno je naglasiti da prioritet imaju postotci nad pikselima. Prednost navođenja širine elemenata u postotcima je u tome što će se element tada sam prilagoditi veličini zaslona uređaja umjesto da mi zasebno definiramo veličinu elemenata u pikselima za svaku veičinu zaslona. Recimo da se naša stranica sastoji iz 2 dijela: prostor koji će biti ispunjen glavnim sadržajem te prostor koji će se nalaziti uz glavni dio stranice, no od njega će biti razmaknut za marginu od 20px. Prethodno navedena 2 dijela naše stranice nisu jednakih širina, no recimo da su sadržani u nekom div elementu širine 960px. Odlučili smo da će glavni dio zauzeti većinu div elementa stoga smo mu dodnijelili širinu pod 640px, dok će drugi dio stranice biti širok 300px. Definirane širine elemenata u pikselima nećemo navoditi za svaki uređaj nego ćemo pomoću slijedeće forume dobiti postotke: 960/1920=50% te 640/960=66.66%



**Sl. 3.3.** *Širina elemanata stranice prikazana u postotcima*

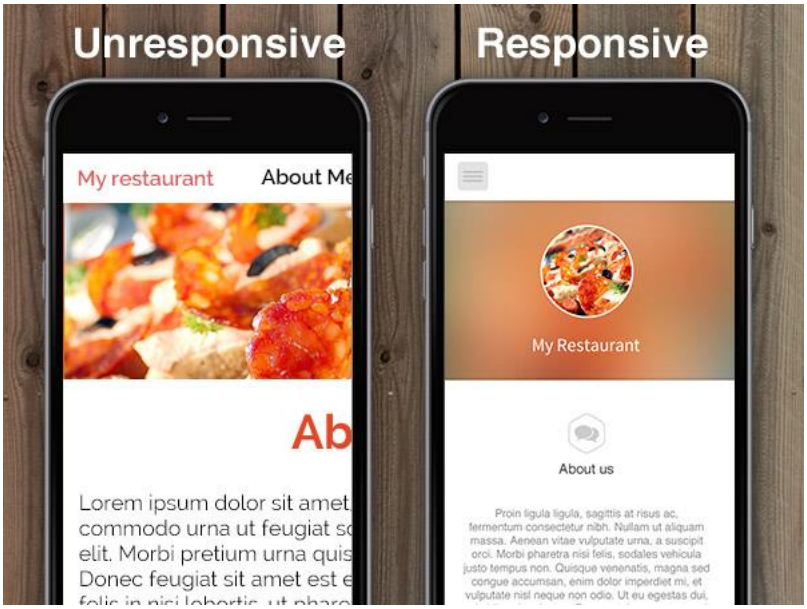
Nakon što smo izračunali omjer veličina elemenata stranice u CSS dokumentu ćemo navesti njihove širine u postotcima uz pripadajuće selektore. Uporabom postotaka umjesto fiksnih vrijednosti ostvarili smo fluidnost tj. elementi će uvijek zauzimati jednak omjer stranice neovisno o veličini zaslona.

## 3.2. Fleksibilni multimedia sadržaj

Kako i sam podnaslov govori pri postizanju responzivnosti također je bitno da su i multimedijski elementi stranice dimenzionalno prilagodljivi veličini zaslona uređaja. U ovu kategoriju spadaju video zapisi, slike te inline frame tj. iframe (okvir u kojemu vidimo pretpleged stranice čiji smo URL naveli unutar iframe oznake). Kao što je objašnjeno u prethodnom potpoglavlju i za ove je multimedijske sadržaje najbolje koristiti postotoke prilikom definiranja veličine elementa.

Za primjer uzmimo sliku koju želimo prilagoditi da zauzima cijeli prostor neke div oznake. Ukoliko bi toj slici dodijelili širinu jednaku *width:100%* iz prve se to čini kao pravo riješenje, no slika će tada (ukoliko je njezina veličina već od veličine naše div oznake) doseći svoju punu širinu i „izaći“ izvan okvira našeg elementa u kojemu bi trebala biti sadržana. Rješenje ovog problema je jednostavno, vrijednost širine ćemo definirati kao *max-width:100%* time ćemo postići da slika ne prelazi granicu elementa u kojemu treba biti sadržana. Smanjenjem ili povečanjem naše div

oznake ovisno o rezoluciji zaslona ujedno će se i slika prilagoditi prostoru u kojemu se nalazi. No drugi problem koji treba riješiti je postavaljanje visine slike, ukoliko bi stavili vrijednot *height:100%* dogodilo bi se isto što i sa širinom slike. Da bi se slika visinom prilagodila elementu unutra kojega se nalazi postavljamo *height:auto*. Također je važno naglasiti da kako bi ova rješenja bila upotrijebljiva bitno je definirati dimenzije prostora elementa u kojemu želimo postaviti sliku.




**Sl. 3.4.** *Prikaz neresponzivnog i responzivnog sadržaja*

## 3.3. Media Queries

Zadnja komponenta kojom postižemo responzivnost, no ujedno i najvažnija je *media query*. Pomoću njih moguće je određeni CSS blok koda primijeniti samo prilikom posebnih uvjeta npr. različiti raspored elemenata stranice ovisno o veličini zaslona ili pogledu uređaja (okomito ili vodoravno). Svaki sadržaj unutar mreže stranice će u određenom trenutku početi prelaziti preko drugog, prelamati se te jednostavno neće više dobro izgledati kao na veličini zaslona za koju je prvobitno bio prilagođen.

Trenutak u kojemu izgled sadržaja stranice više ne izgleda dobro zovemo *breakpoint* (eng. točka prijeloma). Kako bi sadržaj uvijek bio pregledan, trebamo u CSS dokumentu kojeg koristimo za stiliziranje naše stranice navesti niz media query-a unutar kojih ćemo definirati zasebne stilove. Media query sintaksa biti će prikazana na slijedećim slikama.

```
@media screen and (max-width: 900px) and (min-width: 600px), (min-width: 1100px) {
  div.example {
    font-size: 50px;
    border: 8px solid black;
  }
}
```

**Sl. 3.5.** *Media Query, ukoliko je veličina ekrana između 600 i 900px ili iznad 1100px, postavit će se rub i veličina fonta 50px za  element*

Pisanje Media Query-a započinje se s @media, zatim navodimo uvjet koji mora biti ispunjen da bi se query izvršio. Točnije navodi se raspon ili određena veličina zaslona za koje će vrijediti stilovi koji se navode unutar vitičastih zagrada { }. Potom unutar tijela query-a nadovimo HTML element, klasu ili identifikator kojemu ćemo pridružiti željeni stil. Na Internetu je moguće naći opširan popis Media Query-a koji se odnose na mobilne uređaje određenih brandova, za različite veličine zaslona laptopa, tableta te za orijentaciju ekrana. Što je više Media Querya navedeno stranica će imati fluidniji sadržaj. Upravo zbog svih prepreka na koje se nailazi prilikom izrade responzivne stranice valja se držati pristupa koji se naziva „*mobile first*“, prema tome pristupu cijeli se raspored i izgled stranice planira prvo za veličiname zaslona koje odgovaraju mobilnim uređajima.

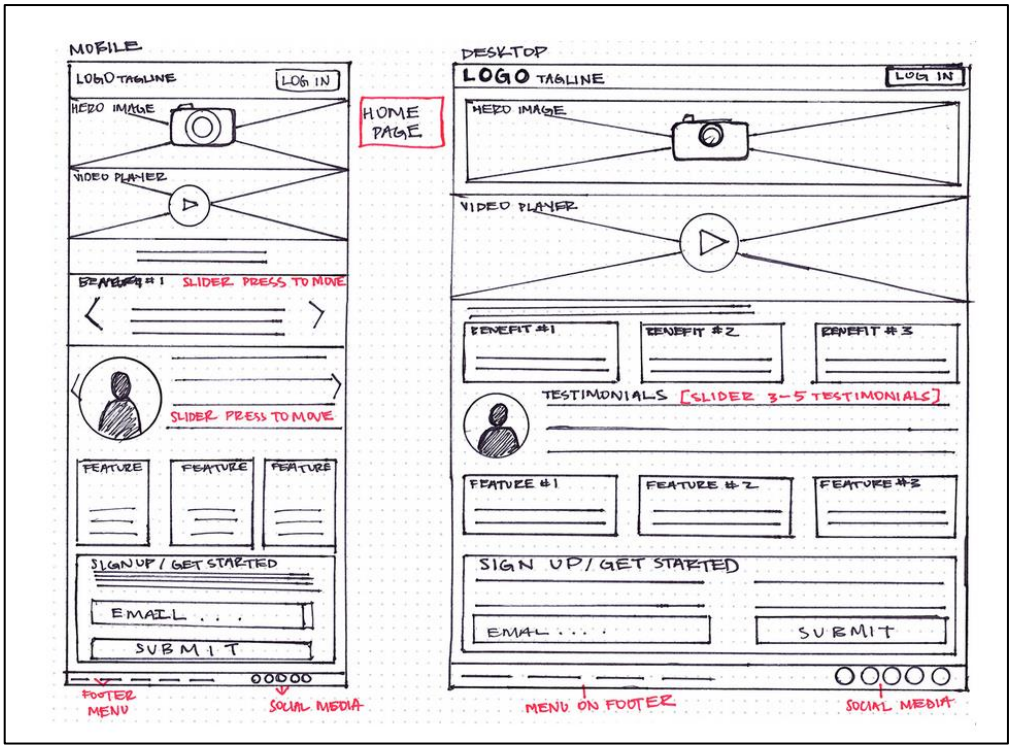
Kako ne bismo uvijek morali izrađivati mrežu stranice i svakome elementu definirati ponašanje prilikom promjena veličina zaslona, možemo se poslužiti Bootstrap *framework-om* koji je *open source,* što znači da svatko može manipulirati njegovim kodom. Mnogi se front-end developeri koriste njime jer olakšava posao izrade responzivnih stranica. Bootstrap možemo gledati kao biblioteku osnovnih elemenata koje bismo pronašli na svakoj stranici. Sadrži HTML i CSS predloške za forme, navigaciju, gumbe i ostale komponete sučelja uz JavaScript ekstenzije koje doprinose interaktivnost elementima.

# 4. IZRADA DIMENZIONALNO PRILAGODLJIVE WEB STRANICE

## 4.1. Izrada one page stranice za veterinarsku ambulantu

One page ili single page stranice postaju sve popularniji tip stranica pomoću kojih se promovira proizvod, poslovanje ili pak služe za demonstriranje portfolija prilikom zapošljavanja. Jednostavnije je ponekad bolje, stoga će stranica, koja je podijeljena na sekcije/dijelove koji predstavljaju najvažnije aspekte nekog poslovanja, više privući pažnju potencijalnom klijentu. Prilikom uporabe ovakve vrste stranica nema čekanja prilikom preusmjeravanja i učitavanja dijelova stranice koji su načinjeni kao zasebna stranica. Stranica se učitava odjednom te prilikom korištenja navigacije korisnik će odmah biti prebačen na dio stranice koji ga zanima. Sadržaj je pregledniji i brzo dostupan, no važno je napomenuti da prilikom izrade single page stranice ne treba dodavati previše sadržaja inače će osoba koja skrola stranicom izgubiti interes. Važno je znati naglasiti ključne riječi i izdvojiti ono najvažnije ukoliko je to moguće, single page stranica ne mora biti rješenje za svako poslovanje. Prije same izrade stranice potrebno je promisliti sadržaj i razraditi kako će sama stranica izgledati. Za takve se svrhe upotrebljava *wireframe*.

**Wireframe** web stranice je zapravo shema tj. nacrt koji predstavlja raspored elemenata na stranici. Wireframe čini kostur stranice, prije samog početka izrade projekta preporuča se izrada wireframe-a kako bi sam proces izrade bio jednostavniji. Unutar nacrta stranice nije uključen dizajn, stil fontova, boje ni grafike jer je centar pažnje na tome koje će funkcionalnosti biti dostupne te kakav će njihov prioritet biti, što se prikazuje poretkom na stranici. Stranica na kraju ne mora izgledati identično kao i njezin nacrt, no lakše je dodavati ili mijenjati elemente na nacrtu nego prilikom izrade stranice stalno alternirati i mijenjati raspored elemenata. Korištenje nacrta ključ je ka boljem korisničkom iskustvu te fluidnijem korisničkom sučelju.




**Sl. 4.1.** *Primjer nacrta (Wireframe-a) stranice za prikaz na ekranu mobitela te na ekranu desktop računala*

Za početak je bilo potrebno stvoriti mapu „projekt“ te otvoriti tekstualni editor Atom i kreirati index.html dokument, koji će kasnije promijeniti ekstenziju u .php. Unutar mape projekta stvoreno je još nekoliko novih pod nazivom css, images, js te fonts. Nakon što je definiran osnovni kostur HTML dokumenta, implementira se Bootstrap framework.

Odlaskom na link http://getbootstrap.com/docs/3.3/getting-started/#download, kopirani su Bootstrap CDN linkovi unutar <head> oznaka index.html dokumenta. Također skinut je Bootstrap zip unutar kojeg se nalaze komprimirane i umanjene CSS, JavaScript i font datoteke. Skinute datoteke svrstavano po mapama projekta ovisno o vrsti (css, js, font). Osim Bootstrap-a prilikom izrade projketa korišteni su Google fontovi koji se mogu preuzeti besplatno na linku https://fonts.google.com/ te je implementirano i nekoliko ikona sa https://fontawesome.com/ stranice.

```
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<!-- Prve 3 meta oznake se uvijek navode prije ostalog sadrzaja -->
<meta name="description" content="veterinary">
<meta name="author" content="Marina Kovac">
<link rel="icon" href="images/favicon.ico">

<title>Veterinarska Ambulanta</title>
<!-- Bootstrap CSS -->
<link rel="stylesheet" href="css/bootstrap.css">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css">

<!-- Glavi Bootstrap CSS -->
<link href="css/bootstrap.min.css" rel="stylesheet">
<!-- CSS dokument za uredjivanje stranice -->
<link href="css/style.css" rel="stylesheet">

<!-- Font Awesome ikone, Google fontovi -->
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.0.6/css/all.css">
<link rel="stylesheet" href="css/fontawesome-all.css">
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Amatic+SC">
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Pontano+Sans">
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Open+Sans">
```

**Sl. 4.2.** *Implemeniranje meta i link oznake unutar head oznaka HTML dokumenta*

Sama stranica sastoji se od nekoliko glavnih sekcija koje su slijedećim redom poslagane:

- *O nama* – dio gdje je postavljen jumbotron te je ukratko navedeno o ambulanti
- *Naš tim* – sekcija gdje su prikazane slike zaposlenika, njihova imena i titule
- *Usluge* – kratak interaktivan popis veterinarskih usluga
- *Kontakt* – kontakt forma preko koje korisnik može poslati upit
- *Blog* – link na blog ambulante
- *Lokacija* – Google Maps prikaz lokacije gdje se ambulanta nalazi te footer s informativnim sadržajem

Svu dokumentaciju koja uključuje kôd i dodatne materijale moguće je **preuzeti** na ovom linku: https://www.dropbox.com/sh/y6vk1fphj217e9p/AABpQw4anoExEQLFkL5wHwEqa?dl=0

**Sl. 4.3.** *Prvi element koji se vidi pri dolasku na stranicu, navigacija, jumbotron te sekcija ukratko o ambulanti*

Na stranici ukupno ima tri sekcije sadržaja koje kao pozadinu imaju slik, a te sekcije su marketinški jumbotron, usluge te blog sekcija. Za te tri sekcije najvažniji dio proizlazi iz CSS-a i glasi:

**background**: url("../images/services-image.jpg") fixed center no-repeat;
**background-size**: cover;

Navedenim svojstvima fokus slike smješten je na njenu sredinu, neće se ponavljati već će zauzeti predefiniran prostor roditeljskog elementa te se neće pomicati zajedno s ostatkom stranice. Cover vrijednost dopušta slici da se širi po elementu u kojemu se nalazi bez da zbog toga izgleda razvučeno. No dio slike biti će „odrezan“ ili vertikalno ili horizontalno. Kada definiramo svojstvo da se slika ne pomiče zajedno s ostatkom sadržaja stranice prilikom skrolanja, ostavlja se dojam dubine stranice, prelazak između sadržaja izgleda fluidnije i preglednije.

```html
<nav class="navbar navbar-fixed-top navbar-default">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
    <a href="#top" class="slide-section"><img id="brand-icon" class="image-responsive" src="images/brand-icon.png"></a>
    </div>
    <div id="navbar" class="collapse navbar-collapse">
      <ul class="nav navbar-nav navbar-right">
        <li><a href="#about-section" class="slide-section" >O nama</a></li>
        <li><a href="#about-team" class="slide-section" >Naš tim</a></li>
        <li><a href="#services" class="slide-section">Usluge</a></li>
        <li><a href="#kontakt-forma" class="slide-section">Kontakt</a></li>
        <li><a href="#location">Blog</a></li>
        <li><a href="#map" class="slide-section">Lokacija</a></li>
      </ul>
    </div><!-- /.nav-collapse -->
  </div><!-- /.container -->
</nav><!-- /.navbar -->
```

**Sl. 4.4.** *HTML kôd potreban za izradu navigacije koja ima toggle funckionalnost na manjim veličinama zaslona (linkovi navigacije se mogu sakriti i prikazati)*

```css
#navbar ul li a {
  font-family: 'Amatic SC', cursive;
  font-size: 25px;
  text-transform: uppercase;
  border-top: 3px solid transparent;
}

#navbar ul li a:hover {
  border-top: 3px solid #4bcebc;
  color: #4bcebc;
}

#navbar ul li a:active {
  border-top: 3px solid #4bcebc;
  background-color: transparent !important;
  color: #4bcebc;
}

@media only screen and (max-width : 768px) {
  .navbar-collapse li a {
    border-top: 3px solid transparent !important;
  }
}

/* Extra Small Devices, Phones */
@media only screen and (max-width : 480px) {
  .navbar-collapse li a {
    border-top: 3px solid transparent !important;
  }
}

/* Custom, iPhone Retina */
@media only screen and (max-width : 320px) {
  .navbar-collapse li a {
    border-top: 3px solid transparent !important;
  }
}
```

**Sl. 4.5.** *CSS kôd koji definira stil navigacije uz media query deklaracije kojima postižemo drukčiji efekt linkova navigavije na manjim zaslonima*

**Sl. 4.6.** *Sekcija koja opisuje vrijednosti ambulante, raspoređena kroz tri col-sm-4 stupca*

Cijela je stranica podijeljena na dijelove, sekcije “O nama”, “Naš tim”, “Kontakt”, kolaž slika te footer (podnožje) sadrže  oznaku s klasom .container kako bi sadržaj bio centriran na svim dimenzijama zaslona. Također svaka od navedenih sekcija unutar navedenog kontejnera sadrži i Bootstrap klasu za stupce čiji zbroj mora biti jednak 12 (Bootstrap mreža sastoji od 12 stupaca koji su dimenzionalno prilagodljivi). Koristi se kombinacija  oznaka s klasom .col-vrijednost-md te, .col-vrijednost-sm te .col-vrijednost-xs jer određeni sadržaj stranice bolje izgleda kada se na nekim veličinama zasona slaže jedan ispod drugog, dok drugio dio izgleda bolje ukoliko je jedan pored drugoga. Bootstrap klasama zapravo određujemo prijelomne točne ponašanja sadržaja ovisno o veliči zaslona.

**Sl. 4.7.** *Sekcija zaposlenika, slike su definirane klasom .img-circle te .img-responsive, time je slikama dodijeljeno svojstvo max-width:100% te height:auto*

Za prikaz sadržaja stranice korišteno je nekoliko različitih načina kroz varijacije. Stoga nije potrebno navesti sav kôd stranice jer jedan primjer objašnjava načine izrade sličnih sekcija. Za preostale sekcije („O nama, „Naš tim“, „Kontakt“ te sekcija kolaža slika) korištena je raspodjela na tri i dva stupca unutar sekcije pomoću Bootstrap kolumni.

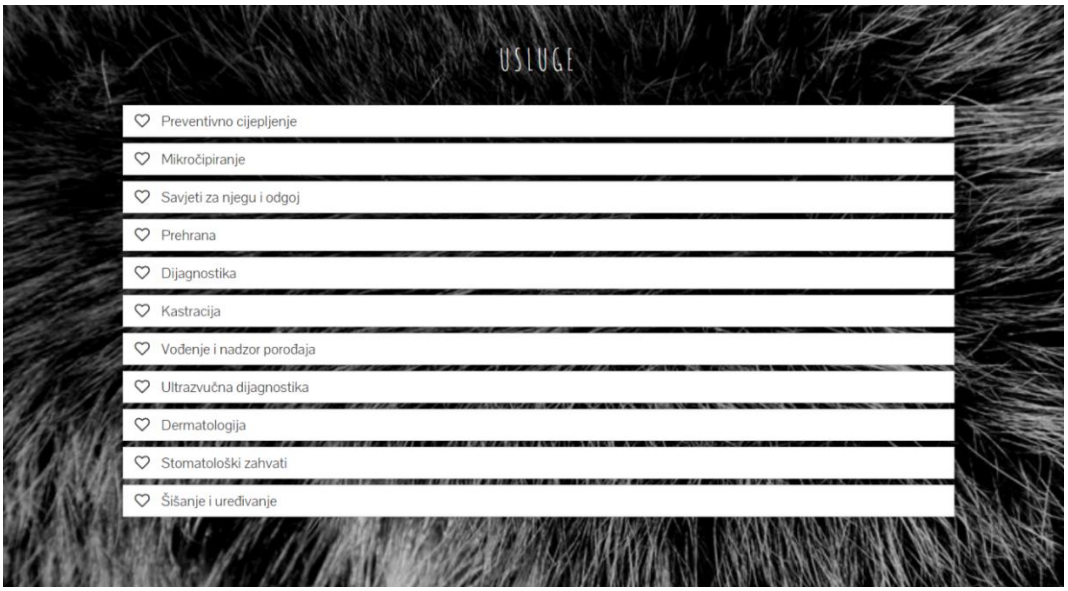
```
<section id="about-team">
    <h2 class="team">Naši zaposlenici</h2>
    <div class="container">

      <div class="col-sm-4">
        <div class="team-member">
          <img src="images/vet1.png" class="img-responsive img-circle" alt="member1">
          <h4>Vanesa Gasparić</h4>
          <p class="titula">Univ. mag. med. vet.</p>
        </div>
      </div>

      <div class="col-sm-4">
        <div class="team-member">
          <img src="images/vet2.png" class="img-responsive img-circle" alt="member2">
          <h4>Darko Šimac</h4>
          <p class="titula">Vlasnik, dr. med. vet.</p>
        </div>
      </div>

      <div class="col-sm-4">
        <div class="team-member">
          <img src="images/vet3.png" class="img-responsive img-circle" alt="member3">
          <h4>Dubravka Vukšić</h4>
          <p class="titula">Univ. mag. med. vet.</p>
        </div>
      </div>
  </section>
```

**Sl. 4.8.** *Prikaz raspodjele sadržaja po responzivnim stupcima, definirao se kontejner unutar kojeg smo smjestili Bootstrap stupce*




**Sl. 4.9.** *Sekcija usluga, korišteni su Bootstrap paneli koji imaju collapse funkcionalnos za prikaz i sakrivanje sadržaja panela*

**Sl. 4.10.** *Prikaz aktivnog panel-a, pritiskom na naziv usluge otvara se skriveno tijelo u kojem se nalazi opis, na isti se način zatvori ili se klikne na neku drugu uslugu*

Prilikom izrade sekcije za usluge, umjesto običnog popisa i cjenika implementirani su nešto kompleksniji Bootstrap *accordion* elementi. *Accordion* se sastoji od *panel* elementa kojemu je dodijeljen *collapse* JavaScript plug-in. Definirali smo prostor za grupu elemenata unutar kojeg će se nalaziti paneli. Svaki panel sastoji se od naziva i tijela unutar kojeg se nalazi opis usluge. Naslovu usluge dodijeljen je *data-toggle* atribut s vrijednosti collapse te atribut *data-target* kojemu se za vrijednost dodijeli CSS selektor kako bi se collapse funkcionalnost primjenila na pravi element, u ovom slučaju na tijelo panel-a. CSS selektor unutar data-target atributa je id  oznake koja predstavlja tijelo usluge (opis), no i tijelu je potrebno dodijeliti klasu .collapse kako bi se prilikom klika na naziv usluge funkcija toggle izvršila te omogućila prikaz ili skrivanje sadržaja tijela. Razlikujemo tri Bootstrap collapse metode:

a) *collapse('toggle')* – prikazuje ili sakriva sadržaj
b) *collapse('show')* – prikazuje sadržaj
c) *collapse('hide')* – sakriva sadržaj

```
<div class="panel-group" id="accordion">
  <div class="panel" id="usluga-panel">
      <div class="panel-heading" data-toggle="collapse" data-parent="#accordion" data-target="#collapseOne">
        <h4 class="panel-title">
          <a class="accordion-toggle"><i id="service-icon" class="far fa-heart"></i>Preventivno cijepljenje</a>
          </h4>
        </div>
        <div id="collapseOne" class="panel-collapse collapse">
          <div class="panel-body">U ovo ubrajamo:
            <p></p>
            <li>Cijepljenje protiv bjesnoće</li>
            <li>Cijepljenje protiv zaraznih bolesti kod pasa (štenećak, parvo virus, zarazni kašalj, leptospiroza)</li>
            <li>Cijepljenje protiv zaraznih bolesti mačaka (panleukopenija, calici, zarazni rinotraheitis u kombinaciji
          </div>
        </div>
    </div>
```

**Sl. 4.11.** *HTML kôd koji prikazuje sintaksu za definiranje jednog collapse panel-a, klasa .panel-heading (naziv usluge) aktivira collapse JS funkcionalnost ukoliko korisnik klikne na naziv*

**Sl. 4.12.** *Sekcija kontakt, unutar kontejnera definirana su dva stupca, .col-md-4 za radno vrijeme te .col-md-8 za kontak formu, više je mjesta pridruženo formi radi preglednosti*

```
<div class="col-md-8">
    <form id="contact" action="<?= htmlspecialchars($_SERVER["PHP_SELF"]) ?>" method="post">
      <div class="row">
          <div class="col-sm-6">
            <input class="form-control" placeholder="Vaše ime i prezime" type="text" name="name" value="<?= $name ?>">
            <span class="error" style="font-size:15px; color:red; margin-top:5px;"><?= $name_error ?></span>
          </div>
          <div class="col-sm-6">
            <input class="form-control" placeholder="Vaša e-mail adresa" type="text" name="email" value="<?= $email ?>">
            <span class="error" style="font-size:15px; color:red; margin-top:5px;"><?= $email_error ?></span>
          </div>
        </div><br>
      <textarea class="form-control" value="<?= $message ?>" name="message" rows="5" placeholder="Ovdje upišite poruku..."></textarea><br>
      <button class="btn btn-default" name="submit" type="submit" id="contact-submit" data-submit="...Sending" style="float:right;">Pošalji</button>
      <div class="success" style="color:#2fb62f; font-size:15px;"><?= $success ?></div>
    </form>
</div>
```

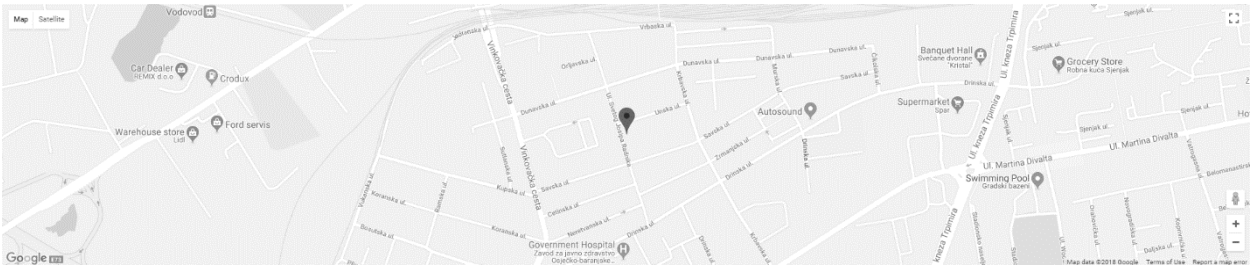
**Sl. 4.13.** *HTML i PHP kôd kontakt forme*

Sama forma je smještena unutar col-md-8 stupca i sastoji od tri dijela. Da bi sve tri komponente forme izgledale pregledno razvrstane su na dva načina. Unutar .col-md-8 stupca definirana je  oznaka s klasom .row jer želimo polje za ime i email staviti u isti red. No da bi polja za ime i email bila jednake veličine treba se opet poslužiti Bootstrap stupcima i voditi se pravilom koje glasi da zbroj stupaca mora biti jednak 12. Stoga je tim poljima dodijeljena klasa .col.sm-6 (sm jer želimo da polja budu jedno ispod drugog na malim dimenzijama ekrana), dok je polje za upis poruke <textarea> smješteno izvan  oznake s klasom .row, kako bi se ono prostiralo kroz prostor definiran klasom .col-md-8. Ovakvim rasporedom polja za ime i email se ne prostiru nepotrebno svom dužinom stupca md-8, dok je za upis poruke idealno imati veći prostor. U slijedećm poglavlju biti će objašnjeni PHP kôdovi za forme i ostatak sadržaja rada koji koristi PHP.

**Sl. 4.14.** *Sekcija za promociju bloga, sadrži pozadinsku sliku te centriranu  oznaku unutar koje se nalazi link na blog*

DR. DOOLITLE
VET. AMBULANTA

**Sl. 4.15.** *Sekcija kolaža i logo-a, gdje je kolaž raspoređen kroz .col-md-4 stupaca, a logo se prostire sredinom  elementa koji je iznad kolaža*

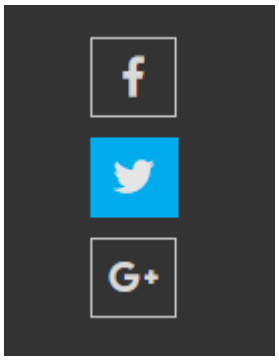



**Sl. 4.16.** *Adresa ambulante prikazana pomoću Google Maps API-a s markerom lokacije*

**Sl. 4.17.** *Footer (podnožje) stranice unutar kojeg su navedene osnovne informacije, navigacija stranice te ikone koje vode na društvene mreže*

Unutar podnožja stranice za ikone društvenih mreža korištene su Font Awesome ikone za Facebook, Twitter i Google+. Da bi se postigao jednostavan no moderan izgled, ikonama je dodan padding koji će dijeliti ikonu od definiranog ruba okvira u kojemu su prikazane. Prije definiranja efekta koji se dogodi prilikom prelaska miša preko ikone, potrebno je definirati tranzicije. Tranzicija u CSS-u omogućuje svojstvu da postepeno kroz određeno vrijeme mijenja vrijednost, npr boju. U ovom slučaju će ikona društvene mreže koja nema ispunu okvira u kojemu se nalazi, nakon prelaska miša preko nje poprimiti pozadinsku boju te će se i boja okvira promijeniti u istu. CSS kôd kojim se to postiže te rezultat prikazani su na slici ispod.

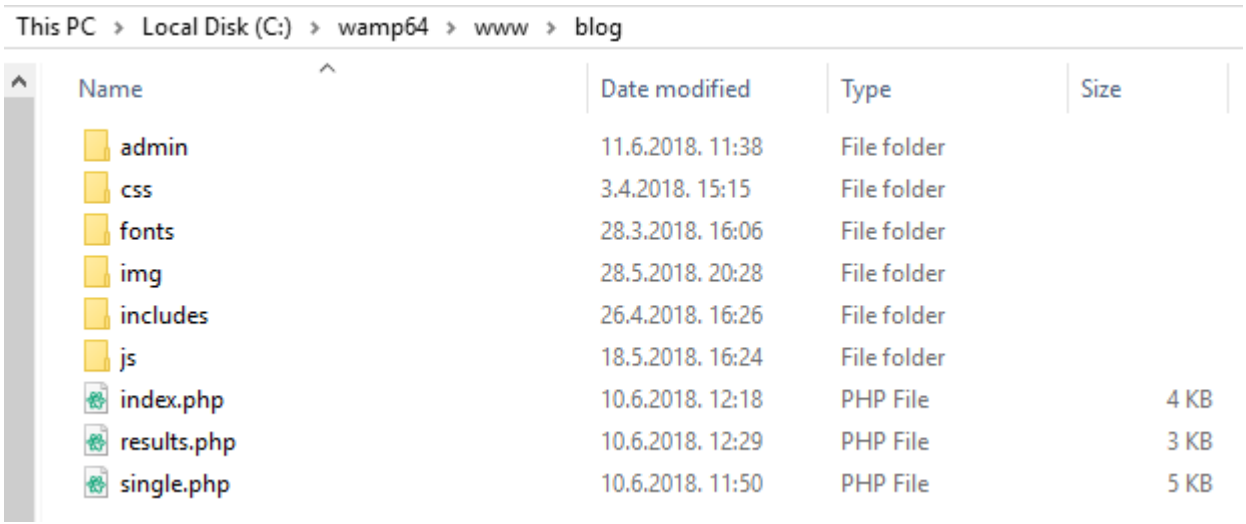
```
554  #social-media a {
555    text-decoration: none;
556    color: #d9d9d9;
557  }
558
559  .fa-facebook-f {
560    border: 1px solid #cccccc;
561    font-size: 20px;
562    padding: 9px 15px;
563    -o-transition:.5s;
564    -ms-transition:.5s;
565    -moz-transition:.5s;
566    -webkit-transition:.5s;
567    transition: .5s;
568  }
569
570  .fa-facebook-f:hover {
571    background-color: #3d5b99;
572    color: #e6e6e6;
573    border: 1px solid #3d5b99;
574  }
```



**Sl. 4.18.** Hover efekt ikona za društvene mreže unutar podnožja stranice

## 4.2. Izrada te pregled Blog stranice i admin panela

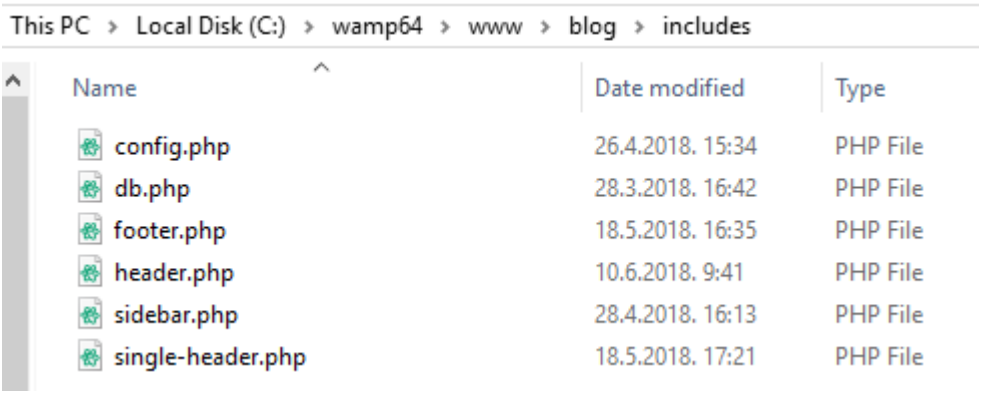
Kao dodatak glavnoj stranici napravljena je još jedna stranica kojoj se može pristupiti preko linka na stranici veterinarske ambulante. Kada korisnik pritisne na link navigacije "Blog", navigacija će ga odvesti do dijela stranice koji promovira blog te klikom na <a> oznaku biti će odveden na zasebnu stranicu. Dodatak bloga stranici koja je namijenjena veterinarskoj ambulanti odličan je dodatak jer doprinosti interaktivnosti. Posjetitelji stranice osim što pomoću kontakt forme mogu poslati email, na blogu mogu pratiti aktualne teme koje ih zanimaju, postavljati pitanja i komentirati s ostalim posjetiteljima objave. Da bi izradili funkcionalan blog, potrebno je koristiti bazu podataka te PHP kôdom implementirati funkcionalnosti kojima će se podaci iz baze prikazivati na stranici. Na početku izrade prvo su napravljeni HTML dokumenti, koji su tek kasnije prebačeni u .php format. Projekt je smješten u www mapu koja se nalazi unutar wamp64 mape jer je izrada projekta vršena na lokalnom serveru. Proces postavljanja lokalnog sučelja za rad s bazom i PHP-om objašnjen je u prethodnom poglavlju. Datoteke bloga podijeljene su u više kategorija i mapa dok stranicu veterinarske ambulante čini samo jedna index.php datoteka te CSS i JS dodaci. Stranica ambulatne PHP koristi samo za postizanje funkcionalnosti slanja email-a, dok se cijeli blog zasniva na PHP-u.




**Sl. 4.19.** *Struktura blog datoteka*

Kreirali smo mapu blog u kojoj će se nalaziti sve potrebne datoteke i mape. Admin sadržavat će sličnu strukturu no u njoj će biti smještene private datoteke kojima će pristup imati samo ovlaštene osobe. Index.php predstavlja prikaz svih objava koje će biti vidljive prilikom ulaska na stranicu bloga. Umjesto navođenja svih HTML i PHP dijelova unutar jednog dokumenta kao što je to napravljeno za stranicu ambulante, ovaj put će dijelovi stranice biti podijeljeni na predloške te će

se naknadno pozivati unutar index.php datoteke. Js, css, image i fonts mape imali smo i za ambulantu, jedino drukčije pri izradi bloga je to što dodajemo mapu includes unutar koje će biti smještene datoteke za povezivanje na bazu te predlošci dijelova stranice.



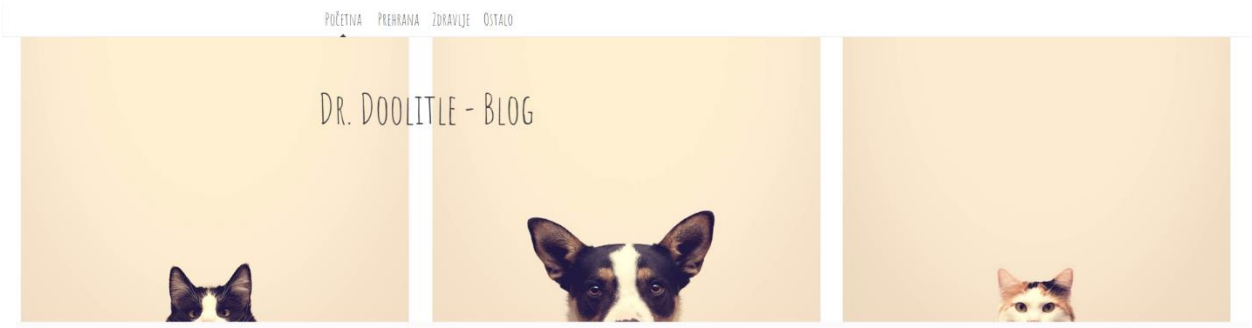

**Sl. 4.20.** *Struktura mape includes*

Na slici je prikazan sadržaj mape *includes*, unutar *header.php* datoteke sadržan je kôd početnih HTML oznaka, jumbotron te početak sekcije za prikazivanje objava. *Sidebar.php* sadrži drugu sekciju stranice koja se nalazi pored glavnog dijela s objavama, unutar te sekcije sadržan je kôd za prikazivanje kategorija, pretraživač, forma za pretplatu, poveznice na društeve mreže te ukratko o ambulanti. *Single-header.php* sličan je header.php datoteci no razlika je u tome što unutar ovog header-a nema jumbotrona sa slikom jer single-header.php služi za prikazivanje pojedinačnih objava te prikaz slike s glavnom prikaza bloga nije potreban jer svaka objava sadrži vlastitu sliku. *Footer.php* drži podnožje stranice gdje su prikazani osnovni infromativni podaci. Dok će datoteke *config.php* i *db.php* biti objašnjene u slijedećem potpoglavlju kada se bude govorilo o izradi baze podataka. Glavna datoteka od koje polazimo nalazi se izvan includes mape, a to je *index.php*. Također je kreirana *results.php* datoteka koja će prikazivati rezultat pretraživanja po ključnim riječima te *single.php* koja predstavlja pogled pojedinačne objave. Unutar index.php pomoću *include* izjave pozvat ćemo datoteke iz includes mape. *Include* uzima sav tekst i kôd koji se nalazi u određenim datotekama i kopira ga unutar datoteke u kojoj su one pozvane. Ovaj način oduzima manje vremena kod većih projekata gdje je potrebno često koristiti iste komponente. Ukoliko je potrebno ažurirati ili izmijeniti neki od tih predložaka pošto u sebi ne sadržavaju sav kôd biti će lakše odraditi taj posao.

```
4        include("includes/config.php");
5        include("includes/db.php");
```

**Sl. 4.21.** *Primjer pozivanja predloška pomoću include izjave*

Na slikama koje slijede biti će prikazana struktura bloga koju vidi svaki posjetitelj. Za pregledavanje bloga i komentiranje nije potrebno biti registrirani korisnik, jedini oblik registracije biti će prikazan za admin sučelje. Blog se sastoji od navigacije, jumbotrona, glavnog dijela unutar kojeg se nalaze objave i straničenje, trake koja se nalazi pored glavnog dijela te podnožja stranice. Unutar header.php datoteke sadržan je kôd za navigaciju, jumbotron te početak  elementa gdje će biti smještene objave. Header je pozvan na početku index.php datoteke pomoću include(„includes/header.php“).



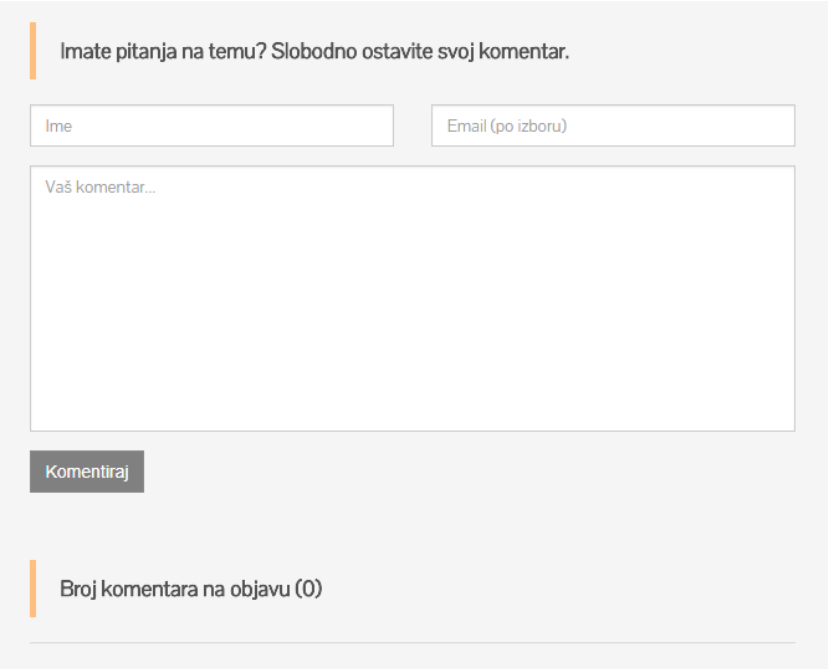

**Sl. 4.22.** *Prikaz header.php kôda u pregledniku, dodavanjem novih kategorija u bazu – unutar navigacije će se pojaviti nove kategorije*

Potom slijedi glavni sadržaj (prikaz svih objava) čiji je kôd unutar index.php-a, objave su smještene unutar  elementa klase col-md-8, što znači da zauzima veći dio mreže stranice. Unutar .col-md-8 stupca nalazi se kompleksija struktura od više  elemenata. Jedan  koji predstavlja čitav prikaz objave, a unutar njega je sadržan  koji drži sliku objave,  koji predstavlja kratki sadržaj objave nakon kojeg slijedi gumb. Pritiskom na gumb naziva „Pročitaj više“ biti ćemo preusmjereni na single.php dokument koji omogućuje cijeli prikaz objave zajedno sa sekcijom gdje korisnik može ostaviti komentar.

**Sl. 4.23.** *Prikaz index.php kôda u pregledniku, prikazuje objave iz baze te straničenje ovisno o broju objava na blogu (na svakoj stranici prikazuje najviše 5 objava)*

**Sl. 4.24.** *Prikaz sidebar.php kôda u pregledniku, sadržaj se prikazuje jedan ispod drugoga, kategorije se „povlače“ iz baze*

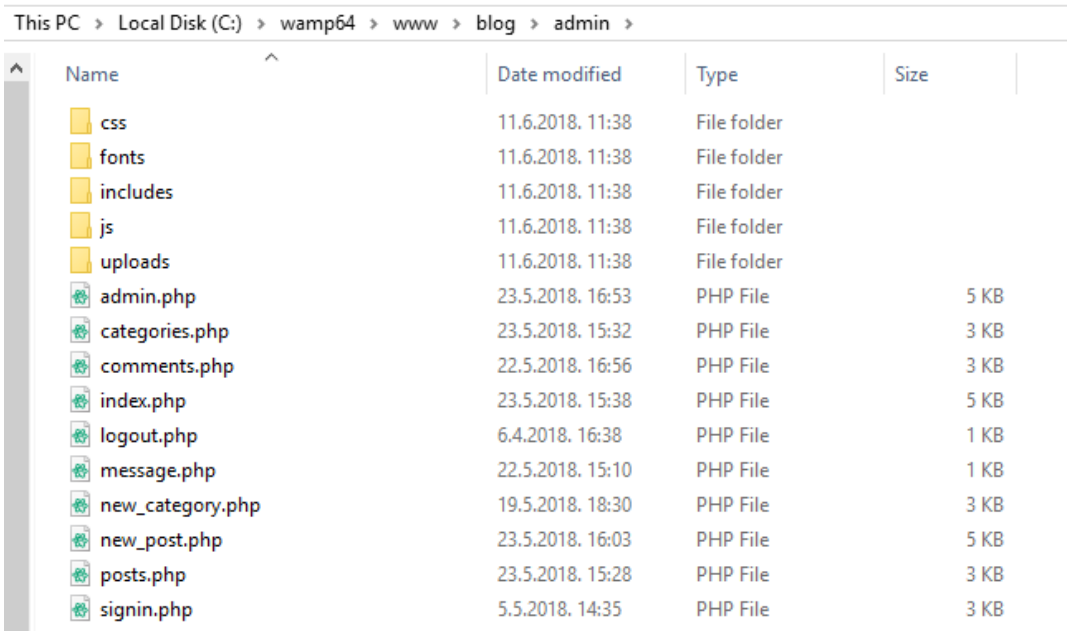
Kao što je prikazano na slici iznad, pomoć sidebar.php datoteke unutar bloga se implementira traka koja zauzima ostatak mreže stranice. Traka je smještena unutar  elementa koji je definiran klasom *col-sm-3 col-sm-offset-1*. Dosad se nije koristila oznaka offset unutar  elemenata, no prilikom definiranja offseta njegova se vrijednost također zbraja prilikom provjere zauzimaj li sadržaj svih 12 stupaca mreže. Koristeći klasu .col-sm-3 col-sm-offset-1  element zauzima jednak prostor na stranici kao i col-sm-4 klasom, jedina razlika je u tome što će razmak između prostora za objave i sidebar-a biti veći. Traka će se prikazivati uz glavni sadržaj bloga sve dok veličina ekrana ne bude odgovarala veličinama koje su jednake ili manje od 734px. Pri tim veličinama traka će se premjestiti ispod sekcije s objavama bloga.




**Sl. 4.25.** *Prikaz prostora za unos komentara na objavu, ukoliko korisnik otvori određenu objavu pri njezinom dnu vidjet će formu koja omogućava unos komentara*

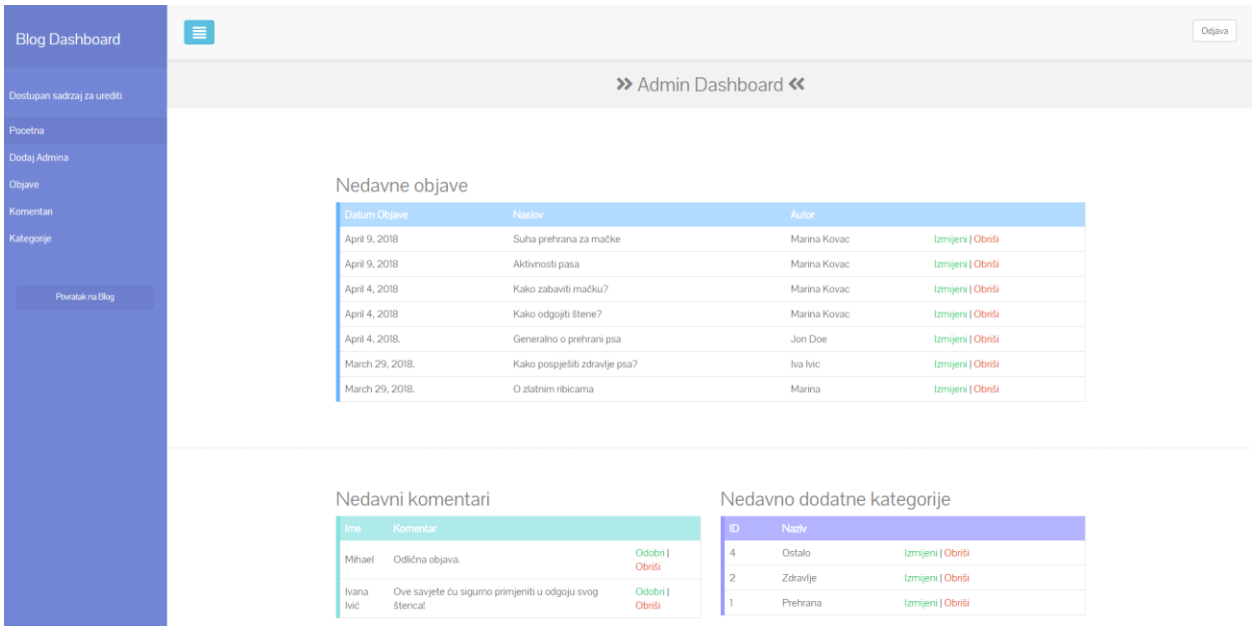
Blog korisnicima također daje mogućnost komentiranja pojedine objave. Ukoliko korisnik želi postaviti pitanje to je moguće pomoću forme koja se nalazi ispod svake pojedine objave. Korisnik se ne mora registrirati kako bi komentirao, forma šalje komentar s podacima osobe koja je komentirala u bazu, a zatim se unutar admin sučelja komentar odobri ili obriše. Komentar će postat vidljiv ispod pripadajuće objave u trenutku kada je odobren. Također ispod forme namijenjene unosu komentara, nalazi se brojač koji prebrojava komentare na objavi.

Nakon kreiranja javnog dijela bloga potrebno je izraditi sučelje za admine. Admin panel tj. sučelje za admine ograničenog je pristupa, samo registrirani administratori mu mogu pristupiti preko forme za logiranje. Ovo je sučelje „posrednik“ bazi podataka, namijenjeno je unosu novih objava i kategorija kao i kontroliranju aktivnosti posjetitelja. Svaki se komentar posjetitelja unutar sučelja mora odobriti ili obrisati ukoliko nije primjeren. Unutar blog projekta smještena je admin mapa u kojoj su smješteni svi pogledi i funkcionalnosti koje ne može vidjeti obični korisnik bloga.



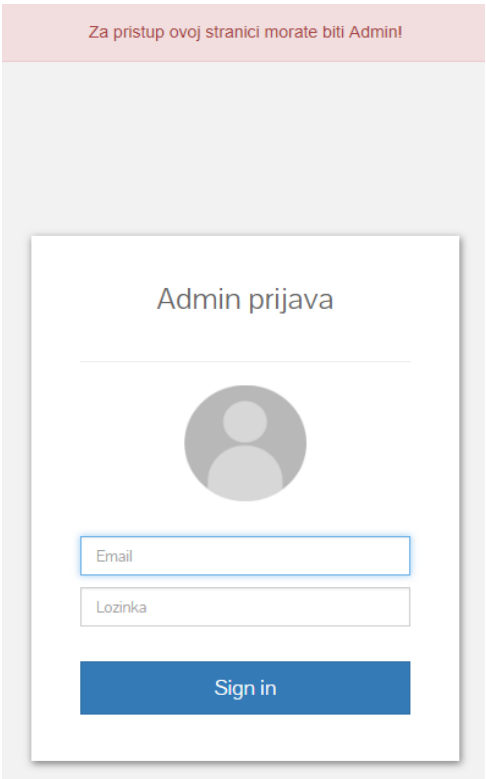

**Sl. 4.26.** *Prikaz strukture datoteka unutar admin mape koje čine admin sučelje*

Kao što je i javni dio bloga podijeljen u poglede tako je podijeljeno i admin sučelje. Index.php ne sadrži cijeli kôd stranice već su dijelovi koji se ponavljaju (navigacija, podnožje, sidebar) prikazani u zasebnim datotekama unutar mape includes. Time se postiže brže učitavanje kôda i stranice. Admin sučelje podijeljeno je u nekoliko dijelova, index.php je cjelokupni prikaz na sve dijelove koji se prate od objava preko komentara do kategorija koje su nedavno dodane u bazu. S index.php stranice moguće je preusmjeriti admina na posts.php, categories.php, comments.php te admin.php. S navedenih pod stranica postoji još dodatnih preusmjeravanja koja vode pristupu formi za unos ili uređivanje postojećih podataka.



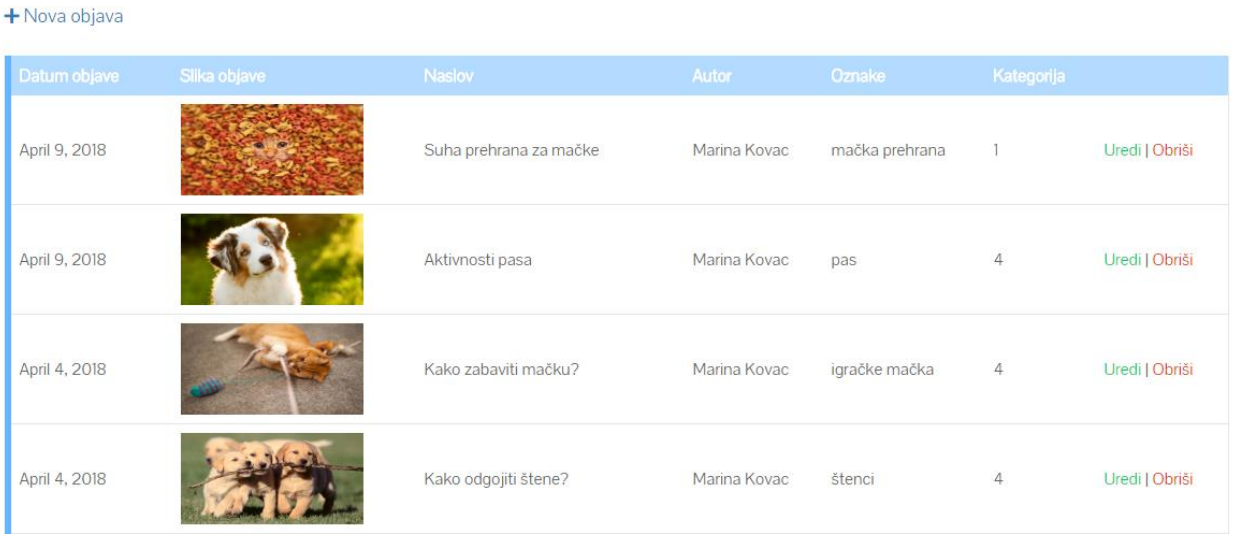

**Sl. 4.27.** *Prikaz admin index.php datoteke u pregledniku*

Da bi prikaz sa slike bio vidljiv osoba koja je ovlaštena za upravljanje funkcijama koje nudi admin sučelje mora se prvo ulogirati. Ukoliko osoba pokuša pristupiti admin sučelju u pregledniku će se prikazati slijedeća forma i upozorenje.




**Sl. 4.28.** *Forma koja omogućava pristup admin sučelju*

Nakon logiranja, admin kao početni prikaz vidi sve nedavne aktivnosti te preko početnog prikaza može odobriti komentare ili ih obrisati. Za pristup ostalim funkcionalnostima koristi se sidebar, traka pored pretpegleda nedavnih aktivnosti. Sidebar također ima collapse funkcionalnost ukoliko se admin sučelju pristupa s uređaja manjih dimenzija zaslona. Traku je moguće sakriti ako smeta administratoru prilikom pregleda sadržaja. S pretpregleda se pomoću sidebar-a možemo preusmjeriti na stranice za dodavanje admina, objava, kategorija i pregled komentara, a ukoliko želimo pregledati kako sadržaj izgleda na blogu to možemo provjeriti klikom na gumb ispod linkova navigacije koji se zove „Povratak na Blog“.




**Sl. 4.29.** *Odlaskom na link „Objave“ admin može pregledati sve objave s bloga, urediti ih, obrisati ili napisati novu objavu*

Prilikom pregleda objava vidljiv je datum objave, slika objave, njezin naslov, autor, ključne riječi kao oznake koje se koriste prilikom pretraživanja te kategorija kojoj objava pripada. Prikaz sadržaja koji se povlači iz baze postignut je pomoću HTML kôda za tablice. Tablice su bile najjednstavnije rješenje za cjelokupni prikaz i grupiranje podataka. Sve tablice (za objave, komentare, kategorije) načinjene su na isti način te imaju jednak stil kojim su uređene, jedina razlika im je u boji radi snalaženja u kategorijama. Slijedeće dvije slike prikazuju strukturu responzivne tablice i pripadajući CSS kôd stila.

```
57    <table class="first-table" style="width: 100%; overflow-x:auto; margin-bottom: 30px;">
58      <tr class="first-row-1">
59        <th>Datum Objave</th>
60        <th>Naslov</th>
61        <th>Autor</th>
62        <th></th>
63      </tr>
64      <?php while($row = $posts->fetch_assoc()) { ?>
65      <tr style="border-bottom: 1px solid #e6e6e6;">
66        <td><?php echo $row['date']; ?></td>
67        <td><?php echo $row['title']; ?></td>
68        <td><?php echo $row['author']; ?></td>
69        <td><a id="edit-link" href="new_post.php?post=<?php echo $row['id']; ?>" class="">Izmijeni</a>
70      </tr>
71      <?php } ?>
72    </table>
73  </div>
```

**Sl. 4.30.** *HTML struktura tablica admin sučelja, podaci su odijeljeni donjim rubom sive boje te tablica na određenim veličinama zaslona dobije horizontalnu scroll traku*

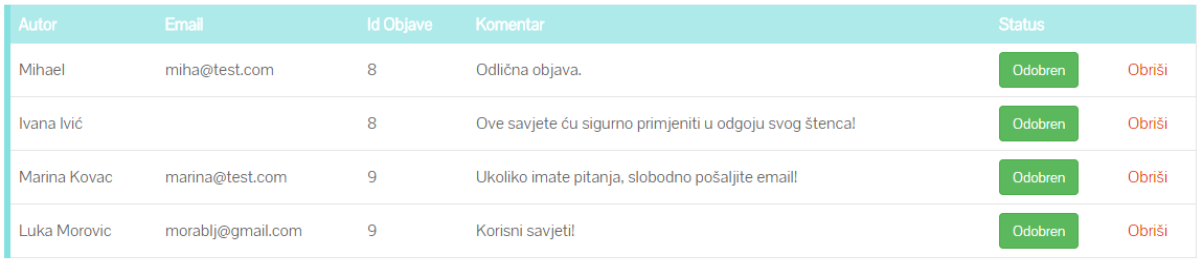
Svakoj je tablici dodana klasa kako bi se mogao primijeniti određeni stil u ovom je primjeru to .first-table. Prvi red <tr> predstavlja zaglavlje tablice gdje su navedeni nazivi za pojedini stupac. Dok se u slijedećem retku tablice nalaze ćelije ispunjenje PHP kôdom koji povlači podatke iz baze. Podaci za svaku objavu izlistavaju se jedan ispod drugoga, prvo će biti izlistani podaci o datumu, naslovu i autoru prvog posta, pa drugog itd.

```
222  table {
223    border-collapse: collapse;
224    width: 100%;
225    color: #666666;
226  }
227
228  th, td {
229    text-align: left;
230    padding: 8px !important;
231  }
232
233  th {
234    color: white;
235  }
236
237  .first-table {
238    border-left: 6px solid #66b3ff;
239    border-right: 1px solid #e6e6e6;
240  }
241
242  .first-row-1 th {
243    background-color: #b3daff;
244  }
```

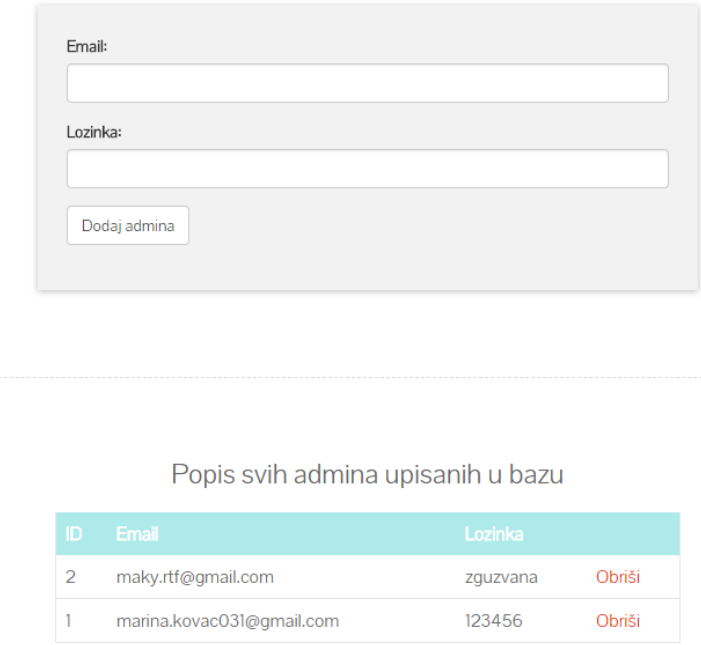
**Sl. 4.31.** *CSS kôd koji određuje stil tablice, tablica će uvijek zauzimati maksimum dostupnog prostora, definiran joj je podebljan lijevi rub, boja pozadine prvog retka (zaglavlja)*

**Sl. 4.32.** *Odlaskom na link „Kategorije“ vidljiv je pregled svih kategorija koje se mogu preurediti, izbrisati ili dodati nova*

Kategorije je moguće preurediti, obrisati ili dodati novu. Svakim dodavanjem nove kategorije u javnom dijelu bloga unutar navigacije i sidebar trake pojavit će ta novo dodana kategorija.
Na stranici namijenjenoj komentarima vidljivi su podaci posjetitelja bloga koji su komentirali objave. Ukoliko je netko postavio pitanje koje zahtjeva duži odgovor, admin može pogledati je li ta osoba navela svoju email adresu te mu tada može putem email-a detaljno objasniti ono što posjetitelja zanima. Također pošto je vidljiv ID objave, admin može otići na blog, pronaći odogovarajući post (na stranici objava vidljiv je naslov te objave s odgovarajućim ID-om) te odgovoriti posjetitelju




**Sl. 4.33.** *Odlaskom na link „Komentari“ vidljivi su pojedini komentari koji su odobreni te identifikacijski broj posta na kojemu su ostavljeni*




**Sl. 4.34.** *Odlaskom na link „Dodaj Admina“ već postojeći admin može naknadno dodati još osoba koje će moći upravljati sučeljem*

## 4.3. Kreiranje baze podataka i implementacija PHP kôda

Prije implementiranja PHP kôda unutar projekta bilo je potrebno kreirati bazu podataka u koju e biti pohranjeni svi podaci koji se prikazuju na blogu. Baza je kreirana preko phpMyAdmin sučelja na lokalnom (localhost) serveru, pod nazivom “blogcms” te se unutar te baze nalazi pet tablica za unos podataka. Tih pet tablica su admin, categories, comments, posts i subscribers, gdje tablica subscribers prikazuje samo jednostavan unos podataka posjetitelja u bazu. Od pet tablica, tablice objava i komentara su međusobno povezane na način da se preko određenog komentara može vidjeti id objave na kojoj je komentar ostavljen, što bi značilo da id objave tablici komentara predstavlja foreign key. Na isti način je postavljen odnos kategorija i objava, svaka objava sadrži naziv kategorije koja joj je dodijeljena, stoga id kategorije predstavlja foreign key unutar tablice objava. Nakon što je baza kreairana potrebno je stvoriti dva ključna .php dokumenta koja ćemo koristiti da bi se spojili na bazu. Prvo je kreirana *config.php* datoteka, unutar nje navedene su četiri konstante koje omogućuju spajanje na bazu te kontakte definiraju ime hosta baze (u ovom slučaju je to na lokalnoj mreži), naziv korisnika i šifra za logiranje u bazu (na phpMyAdmin) te naziv baze na koju se treba spojiti.

```
3    define("DB_HOST", "localhost");
4    define("DB_USER", "root");
5    define("DB_PASS", "marina94");
6    define("DB_NAME", "blogcms");
```

**Sl. 4.35.** *Sadržaj config.php datoteke, varijable potrebne za spajanje na bazu*

Slijedeća je na redu *db.php* datoteka koja služi za povezivanje na bazu. Kreira se varijabla u koju se pohranjuje objekt koji sadrži argumente koji su definirani unutar config.php datoteke. Kreirane konstante iz config.php datoteke služe kao argumenti objekta u db.php datoteci.

```
<?php

 $db = new mysqli(DB_HOST, DB_USER, DB_PASS, DB_NAME);

?>
```

**Sl. 4.36.** *Sadržaj db.php datoteke, prikaz spajanja na bazu*

Ove će se dvije datoteke pozivati na početku svakog dokumenta inače ne bi bilo moguće izvršiti PHP kôd za prikazivanje podataka iz baze podataka na stranici. Kako bi izlistali sve objave koje se nalaze u bazi treba definirati query te ga izvršiti. Definira se varijabla nakon definiranja upita (query-a) te će se u tu varijablu pohraniti rezultat upita tj. svi redovi tablice koja se nalazi u našoj bazi. Da bi te podatke ispisali unutar predviđenog prostora u HTML kôdu potrebno je napraviti if petlju koja provjerava ima li redova unutar tablice podataka te ukoliko ih ima, while petlja prolazi kroz te redove i pomoću fetch_assoc() funckije rezultate iz redaka stavlja u polje što omogućava ispis tih podataka na ekranu. Podaci se ispisuju pomoću echo funkcije. Ovaj način ispisa podataka korišten je za sve ispise objava, kategorija i komentara iz baze podataka.

```php
if(isset($_GET['category'])) {
  $category = mysqli_real_escape_string($db, $_GET['category']);
  $query = "SELECT * FROM posts WHERE category='$category' ORDER BY id DESC";
}elseif(isset($_GET['page'])){
  $page=$_GET['page'];
  if($page==0 || $page<1) {
    $ShowPostFrom=0;
  }else {
    $ShowPostFrom = ($page*5)-5;
  }
    $query = "SELECT * FROM posts ORDER BY id DESC LIMIT $ShowPostFrom,5";
  }

else {
  $query = "SELECT * FROM posts ORDER BY id DESC LIMIT 0,5";
}

$posts = $db->query($query);
```

**Sl. 4.37.** *Na slici je prikazan upit kojim se objave filtriraju prema kategoriji, a dio uokvireno crveno prikazuje uobičajen upit za ispis iz baze te postupak izvršavanje upita*

Slijedeće je bilo bitno napisati kôd kojim bi se kategorije iz baze prikazale u navigaciji i sidebar traci. Kategorije su na blogu uz pretraživanje po ključnim riječima najvažnija funkcija po kojoj se posjetitelj može orijentirati. Ukoliko posjetitelj klikne na jednu od kategorija, link ga mora odvesti na objave koje se nalaze pod tom određenom kategorijom. Prvo je potrebno kreirati i izvršiti upit koji će dohvatiti sve kategorije iz tablice “categories” te će se rezultat našeg upita pohraniti u varijablu $categories. Zatim se u području HTML kôda za navigacijsku traku izvrši prethodno opisan PHP kôd kojim ispisujemo podatke iz baze podataka. Ovim kôdom ostvaren je ispis kategorija, preostalo je još dodati funkcionalno filtriranja prema kategorijama tj. funkcionalnost koja će izlistati određene objave ovisno o kategoriji koja je odabrana. Filtriranje ostvarujemo isset() funkcijom tako da definiramo if else petlju, postavimo isset kojemu predajemo varijablu koju treba provjeriti, a to je u ovom slučaju $_GET[‘category’]. Otvaramo blok unutar kojega definiramo varijablu $category koja će u sebi držati podatak iz URL-a koji ukazuje na ime kategorije. Potom definiramo novi upit koji ćemo također pohraniti u varijablu $query, taj upit govori da treba dohvatiti sve objave pod uvjetom da je njihova kategorija jednaka onoj iz URL-a. Na kraju zatvramo blok kôda te definiramo else uvjet unutar kojeg se nalazi upit kojim se dohvaćaju sve objave iz baze ukoliko u URL-u nema navedene kategorije.

```php
<?php if($posts->num_rows > 0) {
  while($row = $posts->fetch_assoc()) {
 ?>
<div class="blog-post">

    <div class="thumbnail">
      <img src="admin/uploads/<?php echo $row['image']; ?>">
    </div>

    <div class="short-post">

      <p class="blog-post-meta"><?php echo $row['date']; ?>, objava od <span><?php echo $row['author']; ?></span></p>
      <h2 class="blog-post-title"><a href="single.php?post=<?php echo $row['id']; ?>"><?php echo $row['title']; ?></a></h2>
      <div class="post-preview">
        <p>
        <?php $body = $row['body'];
           echo substr(strip_tags($body), 0, 400) ."...";
        ?>
       </p>
      </div>
      <a id="read-more" href="single.php?post=<?php echo $row['id'] ?>" class="btn btn-primary">Pročitaj više</a>
   </div>
</div><!-- /.blog-post -->
```

**Sl. 4.38.** *Postupak kojim se postiže ispis podataka iz baze unutar HTML kôda, prikazana je while petlja, fetch_assoc() te echo funkcija*

Index.php prikazuje pet objava po stranici, no svaka objava ima ukratko ispisan tekst kako objave ne bi zauzimale previše mjesta na stranici jer to također utječe i na brzinu učitavanja bloga. Upravo zbog toga ispod kratkog sadržaja nalazi se gumb koji posjetitelja odvodi na single.php stranicu koja služi kao pregled pojedinačne objave. Da bi gumb radio potrebno je implementirati PHP kôd unutar href atributa, dodaje se naziv dokumenta na koji linkamo, URL varijabla te se pomoću echo funkcije dohvaća id pojedine objave. Na ovaj način ukoliko posjetitelj klikne na gumb ili pak na naslov objave, link će ga odvesti na objavu koja ima određeni id.

```php
<a id="read-more" href="single.php?post=<?php echo $row['id'] ?>" class="btn btn-primary">Pročitaj više</a>
```

**Sl. 4.39.** *Gumb vodi na single.php datoteku, navodi se URL varijabla ?post te se dohvaća id određene objave kako bi bilo moguće u pregledniku vidjeti onu objavu koju je korisnik kliknuo*

Zaseban prikaz objava omogućava nam single.php datoteka, taj se prikaz ostvaruje na sličan način kao i filtriranje objava po kategorijama. Kod u zaglavlju datoteke isti je kao na slici Sl…, samo je potrebno dohvatiti id objave iz URL-a te izvršiti upit koji će dohvatiti sve retke objave (naslov, sadržaj) čiji id odgovara id-u iz URL-a. Unutar single.php datoteke nalazi se i forma za komentiranje na objavu. Komentari su definirani id-om, imenom osobe koja komentira, email-om koji posjetitelj može po želji upisati, tijelom objave te id-om objave na kojoj su ostavljeni. Potrebno je u zaglavlju dokumenta napisati PHP kôd koji provjerava jesu li podaci iz forme

proslijeđeni te gumbu u formi dodijeliti name atribut pomoću kojeg je moguće izvršiti taj kôd. Nakon što posjetitelj klikne gumb "Komentiraj" kôd provjerava jesu li polja forme popunjena, uzima vrijednosti polja te ih pomoću SQL upita pohranjuje u predviđene retke tablice "comments". Kako se komentar ne bi proslijeđivao i ponovno upisivao u bazu nakon što se stranica osvježi potrebno je pozvati header() koji će nas odmah nakon proslijeđenih podataka preusmjeriti na istu stranicu objave na kojoj smo komentirali. Komentare ćemo prikazati tako da izvršimo SQL upit i povučemo sve komentare iz tablice "comments" gdje je vrijednost post jednaka id-u objave. Zatim se pomoću while petlje prolazi kroz sve redove tablice komentara i oni se ispisuju pomoću echo funkcije (ovaj postupak je prethodno objašnjen). Još jedna važna funkcionalnost bloga je pretraživanje objava po ključnim riječima koje definiraju pojedine objave. Rezultat pretrage korištenjem tražilice bloga prikazan je results.php datotekom. Objave se filtriraju ovisno o ključnim riječima definiranjem isset() funkcije tako da kreiramo if else petlju, isset funkciji predamo $_GET['search'] varijablu, gdje je "search" name atribut polja za pretraživanje. Na kraju unutar bloka definiramo upit koji će dohvatiti sve objave kojima je vrijednost keywords iz tablice objava jednaka određenom uzorku znakova koji se prilikom pretrage pojave unutar URL-a.

```php
if(isset($_GET['search'])){
  $page_title = "Rezultati za pojam \"". $_GET['search'] ."\"";
}

include("includes/header.php");

if(isset($_GET['search'])) {
  $keyword = mysqli_real_escape_string($db, $_GET['search']);
  $query = "SELECT * FROM posts WHERE keywords LIKE '%$keyword%' ORDER BY id DESC";
  $posts = $db->query($query);
}
else {
  echo "<p>Nema rezultata za trazeni pojam</p>";
}
```

**Sl. 4.40.** *Kôd za funkcionalnost pretraživanja, dio WHERE keywords LIKE '%keyword%' omogućava pretraživanje točno te riječi u bilo kojoj poziciji*

Nakon funkcionalnosti i izgleda javnog dijela bloga, potrebno je bilo napraviti dio koji neće biti dostupan svakom posjetitelju te mu mogu pristupiti samo ovlaštene osobe. Radi se o admin sučelju (admin dashboard) preko kojega jedan ili više admina mogu kontrolirati komentare posjetitelja, dodavati nove kategorije i objave, izmijenjivati postojeće unose te ih brisati ukoliko je to potrebno. Ovo je sučelje podijeljeno na sličan način kao i blog, unutar pojedinih php dokumenata ne nalazi se cijeli kod već se u mapi includes nalaze one datoteke koje se na svakom prikazu ponavljaju kao

što je sidebar, header, footer. Za početak su u bazu ručno uneseni podaci admina kako bi se moglo pristupiti sučelju. Datoteka koja će biti prva prikazana ukoliko pokušamo pristupiti blog/admin/index.php je *sign.php*. Ova datoteka sadrži formu preko koje se admin može samo logirati. Kada admin popuni tražena polja forme, klikom na gumb koji nosi name atribut login aktivirat će se PHP kôd. Pomoću isset() funkcije provjeravamo podatke koji su proslijeđeni preko forme pomoću $_POST metode te ukoliko se upisani podaci podudaraju s podacima iz baze tj. admin tablice u bazi, admin će biti preusmjeren na početnu stranicu admin sučelja. No unutar datoteke koja se svugdje koristi, a to je header.php potrebno je započeti sesiju i provjeriti ukoliko sesija nije postavljenja (ukoliko unutar nje nije pohranjen email admina) te ako je to točno korisnik će biti preusmjeren na login.php uz poruku greške koja glasi "Za pristup ovoj stranici morate biti Admin!". Također je postavljena poruka greške ukoliko su upisani podaci neispravni ili ne odgovaraju onima iz baze.

```php
if(isset($_POST['login'])){
  $email = mysqli_real_escape_string($db, $_POST['email']);
  $password = mysqli_real_escape_string($db, $_POST['password']);

  $query = "SELECT * FROM admin WHERE email='$email' AND password='$password'";
  $result = $db->query($query);

  if($result->num_rows == 1){
    $_SESSION['email'] = $email;
    header("Location:index.php");
    exit();
  }else{
    header("Location:signin.php?err_msg=Upisani su neispravni podaci.");
    exit();
  }
}
```

**Sl. 4.41.** *Podaci proslijeđeni preko forme spremaju se u varijable te se uspoređuju s postojećima iz baze, ukoliko email odgovara onome iz baze korisnik će biti preusmjeren*

Nakon kreiranja login funkcionalnosti kreiran je gumb koji omogućuje adminu da se izlogira sa sučelja. Unutar logout.php datoteke definiran je početak sesije zatim su uklonjene sve proslijeđene varijable (email, password) te je nakraju sesija koje definira logiranje uništena, a korisnik sučelja biti će preusmjeren na singin.php kada klikne na gumb "Odjava".

```php
<?php

  session_start();

  session_unset();

  session_destroy();

  header("Location:signin.php");

  exit();

?>
```

**Sl. 4.42.** *Da bi se korisnik izlogirao sa sučelja sesija se mora poništiti i uništiti, a korisnik sučelja biti će preusmjeren na singin.php*

Svrha ovog sučelja je mogućnost manipulacije sadržajem bloga. Sadržaj iz baze prikazan je unutar tablica za objave, kategorije i komentare na isti način na koji se prikazuju i objave na javnog dijelu bloga pomoću upita koji je spremljen u varijablu te PHP kôda unutar HTML-a koji prolazi kroz polje redaka i zatim ispisuje rezultat svakog retka tablice. Ono što javni dio bloga nema je mogućnost uređivanja i brisanja. Osim datoteke za pretpregled cjelokupnog sadržaja admin mapa sadrži posts.php, categories.php, comments.php te admin.php gdje je moguće pojedini sadržaj pregledati te s tih stranica preusmjeriti se na druge koje su namijenjene novom unosu i uređivanju postojećeg sadržaja. Pretpregled također sadrži link kraj pripadajućih sadržaja koji odvodi admina na stranicu za preuređivanje. Klikom na "Izmijeni" gumb izvrši se kôd koji provjerava pomoću if petlje prisustvo entity, action i id parametara u URL-u, njihove se vrijednosti pohrane u varijable $entity, $action i $id. Nakon definiranja varijabli unutar već postojeće if petlje definiraju se if else situacije, prvo se definira blok ukoliko je action == delete, tada se unutar tijela te petlje provjerava odnosi li se entity na objavu, komentar ili kategoriju. Unutar svake provjere definiran je SQL upit kojim se briše objava/komentar/kategorija s id-em koji odgovara onome u URL-u ukoliko je admin kliknuo na gumb definiran akcijom brisanja. Na sličan način smo definirali i funkcionalnost odobravanja komentara. Ukoliko admin odluči odobriti komentar izvršit će se unutar iste if petlje upit koji će ažurirat redak tablice "comments" koji se zove status te će vrijednost s 0 staviti na 1. Važno je u upitu naglasiti da se status stupac ažurira samo od onog komentara čiji je id jednak id-u iz URL-a.

```php
if(isset($_GET['entity']) && isset($_GET['action']) && isset($_GET['id'])) {
    $entity = mysqli_real_escape_string($db, $_GET['entity']);
    $action = mysqli_real_escape_string($db, $_GET['action']);
    $id = mysqli_real_escape_string($db, $_GET['id']);

    if($action == "delete"){

        if($entity == "post"){
          $query = "DELETE FROM posts WHERE id ='$id'";
        }else if ($entity == "comment"){
          $query = "DELETE FROM comments WHERE id ='$id'";
        }else{
          $query = "DELETE FROM categories WHERE id ='$id'";
          $q = "UPDATE posts SET category='0' WHERE category='$id'";
        }

    }else {
        $query = "UPDATE comments set status = '1' WHERE id='$id'";
    }
    $db->query($query);
    if(isset($q)){
      $db->query($q);
    }
}
```

**Sl. 4.43.** *Funkcionalnost za brisanje objava/komentara/kategorija*

Brisanje kategorija nešto je drukčije od brisanja objava i komentara. Unutar prethodno objašnjene if petlje navedena je još jedna else grana unutar koje je definiran upit koji će obrisati kategoriju čiji id odgovara id-u iz URL-a te će se definirati nova varijabla koja će ažurirati obrisanu kategoriju na vrijednost 0. Ovo je potrebno iskodirati jer ukoliko se obriše neka kategorija, objava s tom obrisanom kategorijom i dalje će imati postavljenu kategoriju koju smo obrisali, a mi želimo da vrijednost kategorije tog posta bude 0 tj. da objava nema kategoriju.

```php
$query = "UPDATE comments set status = '1' WHERE id='$id'";
```

**Sl. 4.44.** *Kôd za odobravanje komentara, komentar će biti prikazan na javnom dijelu bloga ukoliko admin klikne na „Odobri“ gumb, time se komentaru mijenja vrijednost status retka*

Za dodavanje novog sadržaja stvorene su datoteke new_post.php, new_category.php te admin.php, ove će se datoteke također koristiti prilikom izmjene postojećeg sadraja. Prvo je na svakoj stranici potrebno definirati uobičajeni upit kojim se stvara novi unos u tablicu baze podataka. Sadržaj dodajemo preko formi stoga ćemo name atribut gumba koristiti u PHP kôdu kako bi bazi proslijedili upisane podatke iz polja formi. Kada smo sve podatke iz polja dodijelili varijablama definiramo upit kojim ćemo u npr. tablicu posts dodati vrijednosti varijabla koje smo prethodno definirali te ćemo izvršiti upit.

Kod uređivanja sadržaja kôd je nešto drukčiji. Unutar new_post.php i new_category.php kreira se nova if petlja koja će provjeravati prisustvo npr. URL varijable post za objave. Nakon toga definiramo varijablu u koju ćemo izvršiti upit koji dohvaća objavu čiji je id jednak onome iz URL varijable, a fetch_assoc() funkcija će u obliku polja dohvatiti podatke objave iz tablice u bazi. Preostalo je još PHP kôd implementirati oko polja forme kojim ćemo prolaziti kroz retke i dohvaćati njihove vrijednosti koje želimo izmijeniti. Zadnja izmjena uključuje dodavanje funkcionalnosti koja će ažurirati izmijenjene podatke ukoliko admin pritisne na gumb „Dodaj“. Ovo se također može riješiti pomoću if else petlje tako da se provjeri prisutnost id vrijednosti te se ta vrijednost pohrani u varijablu, a zatim se izvrši upit za ažuriranje pojedinog retka objave (naslov, autor, kategorija, tijelo, keywords, slika). Na isti način se dodaju i izmjenjuju kategorije.

```php
if(isset($_POST['add_post'])){
  $title = mysqli_real_escape_string($db, $_POST['title']);
  $author = mysqli_real_escape_string($db, $_POST['author']);
  $category = mysqli_real_escape_string($db, $_POST['category']);
  $body = mysqli_real_escape_string($db, $_POST['body']);
  $keywords = mysqli_real_escape_string($db, $_POST['keywords']);
  $image = $_FILES['image']['name'];
  $target = 'uploads/'.basename($_FILES['image']['name']);

  if(isset($_POST['id'])){
    $id = mysqli_real_escape_string($db, $_POST['id']);
    $query = "UPDATE posts SET title='$title',author='$author',category='$category',body='$body',image='$image',keywords='$keywords'
    WHERE id='$id'";
    move_uploaded_file($_FILES['image']['tmp_name'],$target);
    header("Location: posts.php");
  }else{
    $d = getDate();
    $date = "$d[month] $d[mday], $d[year]";
    $query = "INSERT INTO posts (title,author,category,body,image,keywords,date)
    VALUES ('$title','$author','$category','$body','$image','$keywords','$date')";
    move_uploaded_file($_FILES['image']['tmp_name'],$target);
    header("Location: posts.php");
  }
```

**Sl. 4.45.** *Prvi dio kôda omogućava kreiranje nove objave preko forme, a drugi omogućava preuređivanje postojećeg sadržaja objave određenog id-a*

# 5. ZAKLJUČAK

Glavni zadatak ovog završnog rada je izrada dimenzionalno prilagodljive web stranice, stoga je najveći fokus stavljen na izradu elemenata stranice koji će se u skladu s promjenom veličine ekrana samoprilagođavati. Uporabom HTML, CSS i JavaScript tehnologija te Bootstrap framework-a postignuta je responzivnost glavne stranice veterinarske ambulante, bloga i admin sučelja. Responzivnost pojedinih dijelova projekta testirana je na stranici http://www.responsinator.com/ te su na svim dimenzijama uređaja koji se nalaze na toj stranici HTML elementi uspješno zadovoljili cilj dobre preglednosti i prilagodbe. Najveći problem pri izradi responzivne stranice je zapravo osmišljavanje rasporeda elemenata. Najviše vremena i testiranja otišlo je na to kako bi bilo najbolje rasporediti određene sekcije po temi, kako bi trebalo izgledati samo zaglavlje stranice te na koji je način najbolje prikazati određene podatke kao što su to vrste usluga koje veterinarska ambulanta nudi. Važno je proučiti što korisnici web stranica smatraju ugodnim oku te kakvi su dizajnovi popularni među modernim stranicama ovisno o području poslovanja za koje se stranica radi. Iz tog razloga odlučeno je prikazati projekt kao one page stranicu koja svojom interaktivnošću korisniku pruža jednostavan i brz uvid u proizvod, kao i pregledan raspored najvažnijih stvari bez da korisnik osjeća preplavljenost informacijama. Uz izradu responzivne stranice implementiran je blog s admin sučeljem kao dodatak završom radu. Blog je bio izvrstan izbor dodatka stranici veterinarske ambulante jer omogućava bolju komunikaciju između profesionalaca i posjetitelja. Pošto glavni fokus nije bio na implementaciji baze i uporabi PHP kôda, napravljen je osnovni primjer blog sustava. Implementiranjem baze podataka, čiji se sadržaj pomoću PHP kôda i SQL upita prikazuje na stranici, postignuta je dinamičnost. Za bolji sustav trebalo bi doraditi postojeći gdje ne bi bilo potrebno svaki put prilikom uređivanja objava uploadati sliku ili pak doraditi kôd vezan za komentare gdje bi preko admin sučelja ovlaštena osoba mogla komentirati na postojeći komentar određene objave kao što je to npr. moguće na Facebook-u. Idealno bi bilo kada bi sustav na temelju email-a prepoznao da se radi o adminu te bi kraj njegovog komentara bila dodana ikona koja razlikuje admina od ostatka posjetitelja koji su ostavili komentare. Sam sustav bi možda bio sigurniji ukoliko bi se uvelo pravilo da osoba ne može ostavljati komentare ukoliko nije kreirala svoj račun/profil na stranici bloga ambulante. Mnogo je faktora kojima bi pospješili funkcionalnost bloga, no pošto se radilo o osnovnoj ideji neke su pogreške i nedostatci zanemarivi. Također bi umjesto običnog textarea elementa koji služi za unos tijela objave bilo bolje implementirati custom tekst editore kakve nalazimo na forumima, gdje posjetitelj ispisani tekst kao i u Word-u može pritiskom na određeni gumb podebljati, poravnati, povećati font itd.

# LITERATURA


[1] W3Schools - https://www.w3schools.com/
- datum zadnje posjete: 24.05.2018.

[2] Tutorial Republic - https://www.tutorialrepublic.com/
- datum zadnje posjete: 19.06.2018.

[4] MDN web docs - https://developer.mozilla.org/en-US/docs/Learn
- datum zadnje posjete: 20.06.2018.

[5] PHP5 Tutorial - http://www.php5-tutorial.com/
- datum zadnje posjete: 18.06.2018.

[6] JavaTpoint - https://www.javatpoint.com/php-tutorial
- datum zadnje posjete: 18.06.2018.

[7] Bootstrap3 - https://getbootstrap.com/docs/3.3/components/
- datum zadnje posjete: 24.05.2018.

[8] Clever Techie YouTube kanal - https://www.youtube.com/watch?v=1CkBsGhux9U&t=1331s
- datum zadnje posjete: 25.05.2018.

[9] Responsinator - http://www.responsinator.com/
- datum zadnje posjete: 21.06.2018.

# SAŽETAK

U ovom radu izrađena je responzivna stranica i blog za veterinarsku ambulantu pomoću HTML, CSS, JavaScript i PHP tehnologija te Bootstrap framework-a. Također je dan opis navedenih tehnologija i objašnjenje najvažnijih linija kôda, dok su slikama prikazani dobiveni rezultati u pregledniku. Glavni fokus stavljen je na rješavanje problema prilagodbe stranice različitim veličinama ekrana. Bootstrap framework najviše doprinosi responzivnosti svojom popularnom stupćastom mrežom koja omogućuje developeru da organizira elemente stranice za svaki uređaj. Korištenjem PHP-a i baze podataka moguće je učiniti blog i admin sučelje dinamičnim. Unutar baze podataka nalaze se informacije o objavama, kategorijama, komentarima na objave te podaci registriranih administratora koje su unesene preko admin sučelja. PHP kôd implementiran je unutar HTML elemenata te je to omogućilo prikazivanje podataka iz baze na pripadajućim dijelovima stranica. U današnje vrijeme ljudi koriste više blog platforme kao što je WordPress, Wix, Weebly itd. umjesto da kreiraju cijeli sustav bloga od samog početka. Razlozi uporabe blog platformi su mogućnost korištenja različitih ugrađenih funkcionalnosti, bolje organiziran sigurnosni sustav, korisnik ne mora imati svoj vlastiti server te može birati između velikog broja predlošaka za izgled svog bloga.

**Ključne riječi:** *baza podataka, blog platforma, Bootstrap, CSS, HTML, PHP, responzivnost*

# ABSTRACT

**Modeling and design of responsive web page**

In this paper a responsive website and a blog for veterinary clinic have been created using HTML, CSS, JavaScript and PHP technologies and the Bootstrap framework. There is also a description of above-mentioned technologies and an explanation of the most important lines of code, while the images show results displayed in the browser. The main focus of this paper is solving the problem of page customization on different screen sizes. What contributes the most to the page responsivness is Bootstrap framework with its popular column grid system, which allows the developer to organize page elements for every device view. By using PHP and database it is possible to make the blog and admin interface dynamic. Database contains information about posts, categories, post comments as well as registered admin data which are submitted via admin dashboard. PHP code that was implemented within HTML elements made it possible to display database data on the appropriate page. Today people usually use blog platforms such as WordPress, Wix, Weebly etc instead of creating a blog system from scratch. Reasons why people choose blogging platforms over creating a blog from scratch is that blogging platforms offer different integrations, functionalities, better security system and user doesn't have to host their own server and can choose from variety of templates.

**Keywords:** *blogging platform, Bootstrap, CSS, database, HTML, PHP, responsive*

# ŽIVOTOPIS

Marina Kovač rođena je 15.10.1994. godine u Osijeku. Svoje je obrazovanje započela u osnovnoj školi „Ivan Filipović“ u Osijeku te se kao odličan đak odlučila za upis u gimnaziju. I. Opću gimnaziju upisala je 2009. godine te je tokom godina pokazala interes za izrađivanje web stranica zbog čega se odlučila po završetku srednje škole, 2013. godine upisati na Elektrotehnički fakultet u Osijeku, točnije stručni smjer informatike. Tijekom fakultetskih godina stekla je osnovna znanja o HTML, CSS, JavaScript i PHP tehnologijama te je ista znanja proširila kroz stručne prakse u Bamboo Lab-u te Mono Software-u u Osijeku. Svoja znanja postepeno je proširivala radom kod kuće, a početkom rada u Shoutem-u 2017. godine upoznala se s marketinškim aspektima IT firme. Uz dobro baratanje frontend tehnologijama na dnevnoj bazi također koristi engleski jezik u govoru i pismu.

Potpis studenta:

_____

# PRILOZI

Na priloženom optičkom disku uz završni rad nalazi se .doc i .pdf verzija završnog rada te kôd web stranice.

SQL upiti za kreiranje tablica iz blog baze podataka:

```
CREATE TABLE IF NOT EXISTS `admin` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`password` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
PRIMARY KEY (`id`))

CREATE TABLE IF NOT EXISTS `categories` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`text` varchar(100) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
PRIMARY KEY (`id`))

CREATE TABLE IF NOT EXISTS `comments` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`name` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
`comment` text COLLATE utf8_unicode_ci NOT NULL,
`post` int(11) NOT NULL,
`email` varchar(200) COLLATE utf8_unicode_ci NOT NULL,
`status` int(1) NOT NULL DEFAULT '0',
`is_admin` int(1) NOT NULL DEFAULT '0',
PRIMARY KEY (`id`))

CREATE TABLE IF NOT EXISTS `posts` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`title` varchar(200) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`category` int(11) NOT NULL,
`date` varchar(100) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`body` longtext CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`author` varchar(100) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
```

```
`image` varchar(255) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`keywords` varchar(200) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
PRIMARY KEY (`id`))

CREATE TABLE IF NOT EXISTS `subscribers` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`name` varchar(100) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`email` varchar(200) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
PRIMARY KEY (`id`))
```