Web aplikacija za vođenje evidencije održavanja osobnog automobila s responzivnim dizajnom

Gal, Sanja

Master's thesis / Diplomski 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:200:466975

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-24**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Sveučilišni studij

WEB APLIKACIJA ZA VOĐENJE EVIDENCIJE ODRŽAVANJA OSOBNOG AUTOMOBILA S RESPONZIVNIM DIZAJNOM

Diplomski rad

Sanja Gal

Osijek, 2018.

Sadržaj

1. UVOD 1
1.1. Zadatak rada 1
2. POVIJEST I RAZVOJ APLIKACIJE 2
2.1. Povijest aplikacija 2
2.2. Struktura aplikacije 3
2.3. Klijent – poslužitelj model 4
3. KORIŠTENE TEHNOLOGIJE 6
3.1. HTML 6
3.2. CSS 7
3.3. JavaScript 8
3.4. PHP 11
3.5. SQL 19
3.6. Bootstrap 20
4. STRUKTURA APLIKACIJE 22
4.1. Struktura baze podataka 22
4.2. Korisničko sučelje 24
5.TESTIRANJE RADA APLIKACIJE 28
6. ZAKLJUČAK 32
LITERATURA 33
SAŽETAK 34
ABSTRACT 35
ŽIVOTOPIS 36

1. UVOD

Ubrzanjem načina života čovjekova potreba za raznim pomagalima se konstantno povećava. U današnje vrijeme teško se može zamisliti život bez pametnog telefona, prijenosnog računala ili tableta. Količina podataka koja se može spremiti ili dobiti pomoću takvih tehnologija čovjeku znatno olakšava organizaciju, orijentaciju, komunikaciju, zabavu i izvršavanje obaveza u svakodnevnom životu. Razvitkom web aplikacija omogućena je dostupnost informacija na zabavan i izgledom lijep način.

Ova aplikacija namijenjena je prosječnom vlasniku automobila i omogućuje korisniku brigu o svome automobilu bez previše razmišljanja i znanja. Korisnik se samo registrira na stranicu i od aplikacije prima obavijesti što treba servisirati na automobilu.

U radu će biti obrađena povijest i razvoj aplikacija, tehnologije koje se koriste za izradu aplikacije, rješenje te tijek izrade aplikacije s opisanim kodom kao i konačan izgled aplikacije.

1.1. Zadatak rada

Zadatak rada je napraviti dinamičku web aplikaciju pomoću koje će registrirani korisnici moći voditi evidenciju o registraciji, servisima i stanju osobnog automobila.

Prvi korak je izraditi model baze podataka za potrebe aplikacije. Pri izradi aplikacije koristit će se responzivni dizajn i mogućnost upozoravanja prijavljenog korisnika na nadolazeće obveze. Pri izradi aplikacije bit će korištene sljedeće tehnologije: HTML5, CSS3, JavaScript (jQuery), PHP/MySQL.

2. POVIJEST I RAZVOJ APLIKACIJE

Sa razvojem informacijskih tehnologija javljali su se noviteti u izradi i korištenju aplikacije. Jedan takav novitet su bile i web aplikacije. U nastavku će biti opisana kratka povijest i razvoj te struktura aplikacija.

2.1. Povijest aplikacija

Početkom 80.-ih dolazi do promjena u razvoju IT aplikacija. Dr. Ted Codd je razvio teoriju izgradnje aplikacija temeljenu na relacijskom sistemu koji je omogućavao usporedno izvršavanje više linija koda odjednom, koristeći strukturni SQL koji može biti implementiran u većinu jezika visoke razine.
Također se počela koristiti objektno-orijentirana tehnologija. C++ i Java su postali popularni i 90.-ih su računalne tvrtke prihvatile ove tehnologije za razvoj aplikacija. IBM-ova softverska grupa je 1998. godine razvila alate koji su postali poznati kao Eclipse.
Tri godine kasnije, u studenom 2001., IBM je odlučio prihvatiti open source operacijski model za ove tehnologije kako bi povećao i ubrzao njihov razvoj. IBM je s osam drugih organizacija ustanovio Eclipse konzorcij i eclipse.org.[1]

Principi konzorcija pretpostavili su da će open source zajednica kontrolirati kod i marketing. To je bio nov i zanimljiv pristup open source modela. Još uvijek se temeljio na otvorenoj, slobodnoj platformi, ali komercijalne tvrtke potiču stvaranje neprofitnih alata. Tako je započelo rođenje IBM® InfoSphere® Optim alata za upravljanje. Ovi alati počinju se razvijati automatizirano prema modelu: [1]

- dizajn
- izgradnja
- razvoj
- rukovanje
- optimizacija.

2.2. Struktura aplikacije

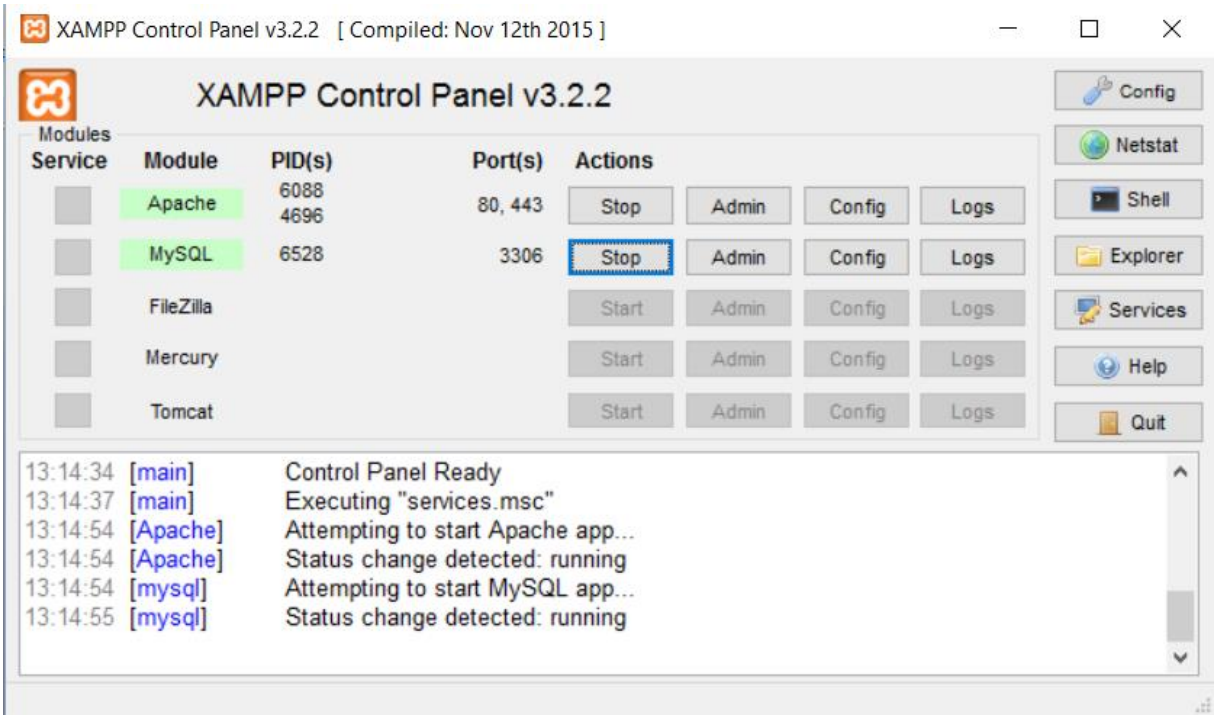
U daljnjem tekstu će biti opisana i objašnjena struktura aplikacije.

Sl.2.1. Struktura aplikacije

Slika 2.1. prikazuje strukturu web aplikacije koja se sastoji od preglednika, poslužitelja i baze podataka.

Baza podataka je zapis podataka kojem računalni program može pristupiti kada mu ti podaci zatrebaju.

Web poslužitelj je računalo na kojem se nalaze web stranice. Kako je pristup svakom računalu određen portovima (ulazima) koji su predstavljeni brojevima, tako je i pristup web poslužitelju određen portom. Port za pristup web poslužitelju je 80. Svaki poslužitelj ima IP adresu, te se prilikom posjećivanja neke stranice, odnosno nekog poslužitelja, zapravo posjećuje IP adresa, pri čemu broj 80 označava da se radi o web poslužitelju. U ovom radu bit će korišten program Xampp koji služi kao simulacija web poslužitelja. Sučelje programa XAMPP je prikazano na slici 2.2.



Sl.2.2. - XAMPP

Slika 2.2. prikazuje XAMPP program. Potrebno je pokrenuti „Apache“ i „MySQL“ za pristupanje aplikaciji.

Preglednik je softverska aplikacija za dohvaćanje, prikaz i pregledavanje informacijskih resursa na World Wide Web-u.

2.3. Klijent – poslužitelj model

Pojam klijent – poslužitelj modela podrazumijeva svaki sustav koji je moguće razdvojit na dva dijela. Prvi dio je klijent, a drugi dio je poslužitelj. Klijent – poslužitelj model je danas jedan od najčešće korištenih modela, a korisnici ga susreću svakodnevno iako toga nisu ni svjesni. Ovaj model je temelj za mnoge aktivnosti – slanje elektroničke pošte, dijeljenje datoteka, pristupanje Internetu, rad sa bazom podataka i sl.

Komunikacije kod klijent – poslužitelj modela se dijeli u tri faze. Prva faza komunikacije između klijenta i poslužitelja je faza gdje klijent šalje zahtjev poslužitelju za obavljanje određene usluge. U drugoj fazi komunikacije poslužitelj obrađuje zahtjev koji je uputio klijent. U zadnjoj, trećoj fazi komunikacije između klijenta i poslužitelja, poslužitelj šalje rezultate obrade klijentu kao odgovor na klijentov zahtjev.

Klijent, a tako i poslužitelj mogu biti računala, tableti, mobilni uređaji pa čak i neki drugi elementi. Primjerice, klijent i poslužitelj mogu biti i programski elementi (programske komponente, moduli unutar iste aplikacije). Okruženje klijent – poslužitelj je heterogeno. To znači da sklopovska oprema i operacijski sustavi klijenta i poslužitelja ne moraju nužno biti isti, što ovaj model čini pogodnim za implementaciju u raznim slučajevima. Na slici 2.3. je prikazan način komunikacije između klijenta i poslužitelja.

Sl. 2.3. – Način komunikacije između klijenta i poslužitelja

3. KORIŠTENE TEHNOLOGIJE

Na tržištu postoji veliki broj tehnologija pomoću kojih se može napraviti ista stvar. Koja tehnologija će biti odabrana često ovisi i o samim navikama razvojnog tima, budžetu pa i mogućnostima određene tehnologije. U nastavku će biti opisane tehnologije korištene u web aplikaciji.

3.1. HTML

HTML je opisni jezik za označavanje koji se koristi za kreiranje dokumenata na World Wide Web-u. HTML se koristi za stvaranje hipertekstualnih datoteka (datoteka koje sadržavaju linkove).

HTML naredbe se pišu u vidu tzv. „tag – ova“, tj. oznaka. Jedan „tag“ je naredba koja govori pregledniku što i kako napraviti tj. na koji način prikazati sadržaj neke stranice. HTML oznake su "case insensitive" tj. svejedno je pišu li se malim ili velikim slovima. Oznake se pišu unutar znaka "<" i ">" (bez znakova navoda) npr: <html>. Ova oznaka se nalazi na početku svakog HTML dokumenta i ona govori pregledniku kako je datoteka koju je upravo počeo učitavati baš HTML dokument te ga kao takvoga treba i prikazati. Na kraj HTML dokumenta se stavlja završna HTML oznaka:</html> koja pregledniku prenosi podatak kako je to kraj HTML dokumenta. Većina oznaka ima i početnu i završnu oznaku. Oznaka završetka se dobiva dodavanjem znaka "/" i označava mjesto na kojem prestaje djelovanje početne oznake.
Postoje i oznake kod kojih se ne mora stavljati oznaka završetka, kao što je <br> koji služi za prelazak u novi red.
Svaki HTML dokument se sastoji od dva dijela: zaglavlja (engl. head) i tijela (engl. body). Zaglavlje se odvaja oznakama <head> i </head>, a tijelo dokumenta oznakama: <body> i </body>. Sve ono što je napisano u zaglavlju dokumenta neće biti prikazano u prozoru preglednika već obično služi samo pružanju neke informacije o samoj stranici. [2]
Na slici 3.1. je prikazan isječak koda sa kojim je definiran izgled stranice pomoću HTML-a.

```
    <div class="timeline-badge"><i class="fa fa-credit-card"></i>
    </div>
    <div class="timeline-panel">
        <div class="timeline-heading">
            <h4 class="timeline-title">Dobar dan <?php echo htmlentities($_SESSION['user']['username'], ENT_QUOTES, 'UTF-8'); ?>!</h4>
        </div>
        <div class="timeline-body">
            <p><?php echo "Vaš auto je: <b>".$rows[0]['auti_id']."</b>"; ?></p>
            <p><?php echo "Model auta je:<b>".$rows[0]['model_id']."</b>"; ?></p>
            <p><?php echo "Prešli ste <b>".$rows[0]['km']." kilometara</b>"; ?></p>
        </div>
    </div>
</li>
<li class="timeline-inverted">
    <div class="timeline-badge warning"><i class="fa fa-car"></i>
    </div>
    <div class="timeline-panel">
        <div class="timeline-heading">
            <h4 class="timeline-title">Mali Servis</h4>
        </div>
        <div class="timeline-body">
            <p><?php echo "Mali servis za vaš auto je svakih: <b>".$rows[0]['mali_servis']."km</b>"; ?></p>
            <?php if ($mali_servis==1) {?>
                <p>Pod hitno napravite mali servis auta</p>
            <?php }else{ ?>
                <p>Vaš auto do malog servisa ima <?php echo $mali_servis_br_km-$x; ?>km</p>
            <?php } ?>

        </div>
    </div>
</li>
<li>
    <div class="timeline-badge danger"><i class="fa fa-car"></i>
    </div>
    <div class="timeline-panel">
        <div class="timeline-heading">
            <h4 class="timeline-title">Veliki Servis</h4>
        </div>
        <div class="timeline-body">
            <p><?php echo "Veliki servis za vaš auto je svakih: <b>".$rows[0]['veliki_servis']."km</b>"; ?></p>
            <?php if ($veliki_servis==1) {?>
                <p>Pod hitno napravite veliki servis auta</p>
            <?php }else{ ?>
                <p>Vaš auto do velikog servisa ima <?php echo $veliki_servis_br_km-$y; ?>km</p>
```

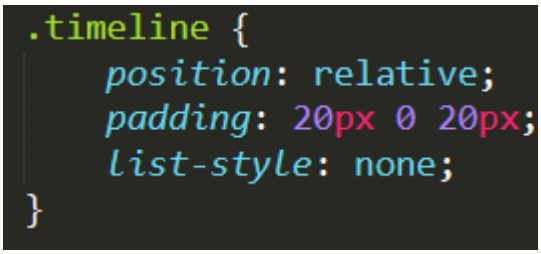
Sl. 3.1. – Izgled stranice definiran sa HTML-om

3.2. CSS

CSS (engl. „*Cascading Style Sheets*“), odnosno kaskadni stilovi, jednostavan su mehanizam za dodavanje stilova: fontova, boja, razmaka između odlomaka, uređivanje tablica. Svojom pojavom CSS je izazvao pravu revoluciju na internetu zahvaljujući nizu prednosti koje ima pred tabličnim „layoutom“ (korištenje tablica za formiranje stranice). Korištenje CSS-a omogućilo je odvajanje prezentacije podataka i dizajna od same strukture podataka. HTML kôd postaje pregledniji i manji te ga je puno lakše kontrolirati, a također je moguće jednostavnom primjenom parametara promijeniti izgled stranice. CSS je donio čitav niz načina za uređivanje prikaza podataka koji do tada nisu postojali u samom HTML-u, a web programeri su razvili korisne tehnike kojima se može uštedjeti dragocjeno vrijeme prilikom izrade internet stranica. Međutim, mogućnosti formatiranja HTML-a prilično su ograničene. Prilikom dizajniranja izgleda stranice u HTML-u mogu se koristiti tablice, različiti fontovi i nekoliko stilova teksta, npr. bold i italic. [2]

Koji stil, odnosno CSS će biti primijenjen na koji element definira se pomoću selektora. Selektori su identifikatori koji aplikaciji govori na koje elemente da primjeni određeni stil. U

ovoj aplikaciji su najviše korišteni klasni selektori. Oni se prepoznaju tako što se ispred imena selektora nalazi točka (.). Primjer korištenja klasnog selektora je prikazan na slici 3.2.



Sl. 3.2. – Primjer korištenja klasnog selektora

Svi elementi koji pripadaju klasi *timeline* se postavljaju na relativnu poziciju. Vrijednosti koje pozicija može poprimiti su jedna od navedenih: *static, relative, absolute, fixed* i *sticky*. U ovome slučaju pozicija je poprimila vrijednost *relative*. Svojstvo *padding* omogućuje generiranje prostora oko elementa koji pripada navedenoj klasi. U ovome slučaju prostor iznad ovog elementa će biti 25px, lijevo i desno će biti 0px, a dolje 20px, dok svojstvo *list-style* označava koji se stilovi koriste za dekoraciju eleemenata iz te klase. U ovom slučaju nema stilova koji su korišteni za dekoraciju spomenutog elementa.

3.3. JavaScript

JavaScript je najpopularniji skriptni jezik na Internetu kojeg podržavaju svi poznatiji preglednici (Internet Explorer, Mozilla, Firefox, Netscape, Opera). Cilj kreiranja JavaScript jezika bio je dodati interaktivnost HTML stranicama.

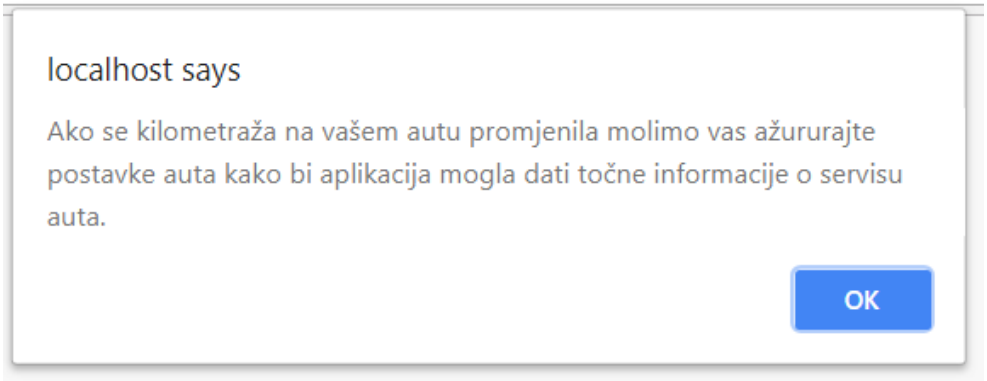
Skriptni jezici su programski jezici manjih mogućnosti, koji se sastoje od izvršnog računalnog koda, obično ugrađenog u HTML stranice. JavaScript je interpreter, što znači da se skripta izvršava odmah naredbu po naredbu, bez prethodnog prevođenja (compiling) cijelog programa i kreiranja izvršne datoteke. JavaScript je javno raspoloživ skriptni jezik (nije potrebna licenca za korištenje). [3]

Što sve može?

- Programiranje u okviru HTML stranica
- Pretvaranje dinamičkog teksta u HTML stranicu – npr. ako se neka varijabla pod nazivom “name” želi ispisati u okviru HTML stranice, može se napisati skriptna naredba: document.write("<h1>" + name + "</h1>")

- Reagiranje na događaje – moguće je postaviti naredbu za izvršavanje skripte kao reakcije na događaj, npr. kada se stranica učita, ili kada korisnik klikne na određeno dugme ili drugi HTML element
- Čitanje i pisanje HTML elemenata - JavaScript može pročitati i promijeniti sadržaj nekog HTML elementa
- Validiranje (provjeru ispravnosti i vjerodostojnosti) podataka - JavaScript može validirati podatke prije nego se pošalju na poslužitelj, čime se poslužitelj oslobađa dodatne obrade
- Detektiranje preglednika kojeg korisnik upotrebljava – na osnovu tog prepoznavanja preglednika JavaScript može učitati drugačiju stranicu ovisno o pregledniku tako da se učita stranica koja je posebno dizajnirana za taj preglednik
- Kreiranje "kolačića" (cookies) - JavaScript može pohraniti i učitati informacije o korisnikovom računalu. [4]

Jedna od primjena JavaScript programskog jezika u ovoj aplikaciji je ispis poruke (engl. alert message) prilikom prijave u aplikaciju. Na slici 3.3. je prikazana ta poruka.



Sl. 3.3. – Poruka prilikom prijave u aplikaciju

Programski kod za ispis poruke sa slike 3.3. nalazi se na slici 3.4. Iz programskog koda vidljivo je da se koristi *localStorage* funkcionalnost iz HTML5 koja omogućuje pohranu određenih podataka u memoriju preglednika. Prilikom prijave u aplikaciju korisniku će se ispisati poruka koja kaže da ažurira postavke automobila i prilikom toga će se u memoriju preglednika pohraniti da je ta poruka prikazana te se više neće prikazivati korisniku sve do iduće prijave u aplikaciju. U datoteci *nav.php* (linija 142) napravljeno je da prilikom odjave klikom na gumb „Odjava" varijabla *localStorage* se postavlja na vrijednost „*no*" čime je omogućeno ponovo prikazivanje poruke prilikom sljedeće prijave korisnika u aplikaciju. Programski kod za ispis *alert message* je prikazan na slici 3.4.

```
<script type="text/javascript">
        var alerted = localStorage.getItem('alerted') || '';
        function kmFunction() {
            if (alerted != 'yes') {
                alert("Ako se kilometraža na vašem autu promjenila molimo vas ažurirajte
                    postavke auta kako bi aplikacija mogla dati točne informacije o servisu auta.");
                localStorage.setItem('alerted','yes');
            }
        }
        kmFunction();
</script>
```

Sl. 3.4. – Programski kod za *alert message*

Isto tako, JavaScript je korišten i za animaciju simbola za obavijesti (zvona). Simbol za obavijesti će imati efekt njihanja sve dokle god korisnik ne klikne na njega kako bi pročitao obavijesti. Za ovaj dio koda nije korišten čisti JavaScript nego biblioteka jQuery. Programski kod za ovaj slučaj je prikazan na slici 3.5.

```
<script>
  $(function() {
    // run the currently selected effect
    function runEffect() {
      // get effect type from
      var selectedEffect = "shake";

      // most effect types need no options passed by default
      var options = {};

      // run the effect
      $( "#shake" ).effect( selectedEffect, options, 5000 );
    };

    runEffect();
  });
</script>
```

Sl. 3.5. – Programski kod za efekt njihanja simbola za obavijesti

Sa slike 3.5. je vidljivo da postoji funkcija *runEffect* koja se poziva prilikom učitavanja datoteke *nav.php*. Unutar te funkcije su deklarirane dvije varijable. Prva varijabla je *selectedEffect* kojoj je dodijeljena vrijednost „*shake*“. Ta vrijednost govori aplikaciji da se simbol za obavijesti treba njihati, odnosno da na njemu primjeni efekt njihanja. Drugoj varijabli je dodijeljen prazan *array* kao vrijednost. Element koji za ID ima vrijednost *shake* (linija 83 u *nav.php*) dobit će efekt njihanja koji se pokreće sa ugrađenom jQuery funkcijom *.effect*. Funkcija *.effect* kao parametre prima vrstu efekta, opcionalni parametar (u ovom slučaju prazan *array*) i brzinu izvođenja efekta (brzina njihanja zvona).

3.4. PHP

PHP (engl. *PHP: HypertextPreprocessor*) je skriptni jezik koji se odvija na poslužitelju i služi za izradu internet stranica. PHP se odvija na poslužiteljskoj strani, što znači da se kod pokreće na poslužitelju i generira HTML te ga šalje korisniku, ali klijent ne može vidjeti ishodišni kod za tu skriptu. PHP je lako umetnuti u HTML kod, a koristi se većinom za izradu dinamičkih materijala, enkripciju podataka, spremanje podataka iz obrasca, slanje i primanje web kolačića, kontrolu pristupa, spremanje i brisanje iz baze podataka te otvaranje, spremanje, čitanje i brisanje podataka sa poslužitelja. [6]

Kako je rečeno PHP jezik služi i za povezivanje aplikacije sa bazom podataka. [7] U nastavku, na slici 3.6. je prikazan dio koda koji služi za povezivanje na bazu podataka.

```php
<?php

    $username = "root";
    $password = "";
    $host = "localhost";
    $dbname = "baza";


        $options = array(PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8');


    try
    {

        $db = new PDO("mysql:host={$host};dbname={$dbname};charset=utf8", $username, $password, $options);
    }
    catch(PDOException $ex)
    {

        die("Greska prilikom spajanja na bazu: " . $ex->getMessage());
    }


    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);


    $db->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);


    header('Content-Type: text/html; charset=utf-8');


    session_start();
```

Sl. 3.6. – Isječak koda za povezivanje na bazu podataka

Definirane su varijable koje sadrže podatke koji se predaju funkciji i spajaju na MySQL bazu. Array $ options se predaju funkciji koja se spaja na bazu i prenosi joj kako se s njom želi komunicirati pomoću UTF-8 character encoding (može dekodirati sve moguće znakove definirane UNICODE-om). Try/catch je uobičajena metoda rukovanja pogreškama u objektno orijentiranom kodu. Prvo PHP izvrši kod s blokom try. Ako se prilikom izvršavanja tog bloka

dogodi neka pogreška zaustavlja se izvršavanje try bloka i izvršava se blok catch. PDO pruža fleksibilno sučelje između PHP i mnogih drugih vrsta poslužitelja baza podataka. Ako se greška dogodi prilikom spajanja na bazu bit će uhvaćena u catch bloku. Skripta će prikazati grešku i prestati raditi.
"$ex--> get Message()" se ne bi tebao prikazivati jer je onda moguće izvući podatke vezane za bazu, te se stoga konfigurira PDO kako bi napravio iznimku prilikom greške (time je omogućeno korištenje try/catch bloka za prikazivanje greške vezane za bazu podataka). Postavljaju se dva atributa. Prvi služi za ispisivanje greške ako ne uspije spajanje na bazu, a drugi služi za vraćanje niza, u kojem vrijednost niza označava naziv stupca u bazi podataka, kako bi se lakše rukovalo podacima i bio omogućen ispis podataka korisniku.

Također, pomoću PHP jezika je ostvarena i prijava korisnika u aplikaciju. Isječak koda sa kojim je to ostvareno prikazan je na slici 3.7.

```php
<?php

    require("db_connect.php");

$spremljeno_korisnicko_ime = '';
$greska=0;

if(!empty($_POST))
{

    $query = "
        SELECT
            id,
            username,
            password,
            salt,
            email
        FROM users
        WHERE
            username = :username
    ";

        $query_params = array(
    ':username' => $_POST['username']
);

    try
    {

        $stmt = $db->prepare($query);
        $result = $stmt->execute($query_params);
    }
```

Sl. 3.7. – Isječak koda za prijavu korisnika u aplikaciju

Prvo je izvršen kod za spajanje na bazu podataka. Zatim je korištena varijabla „$spremljeno_korisnicko_ime="";“ za spremanje korisničkog imena kako bi, ako korisnik pogriješi lozinku, morao ponovno upisati samo lozinku, a ne i korisničko ime. Pomoću if-

naredbe provjerava se je li login forma spremljena. Ako je spremljena, login kod se pokreće, a ako nije ponovno se pojavljuje login forma.

Na slici 3.8. je prikazan isječak koda koji vrši provjeru lozinke i preusmjerava na stranicu.

```
    else
    {
        $greska = 1;

        $spremljeno_korisnicko_ime = htmlentities($_POST['username'], ENT_QUOTES, 'UTF-8');
    }
}
```

Sl. 3.8. – Isječak koda za provjeru lozinke i preusmjeravanje na stranicu

Postavlja se $login-ok varijabla s bolean vrijednosti true/false te je početno definirana kao false, a ako se korisnik uspješno ulogira postavljena je na true. Prilikom izvršavanja upita na bazu ako baza varijablu $row označi kao false, tj. nije nađena u bazi, onda taj korisnik uopće nije registriran. Sljedeća radnja je uspoređivanje lozinke. Korištenjem lozinke koju je korisnik spremio i salt-a (ključ za dešifriranje) iz baze, moguće je provjeriti točnost lozinke. Ako je točna, varijablu se označi true "$login-ok=true;".

Ako se korisnik uspješno logirao, otvara se stranica korisnika koji je logiran. Ako se nije uspio logirati prima poruku o neuspješnom logiranju te mu se ponovno prikazuje forma za logiranje. Naredbom unset sklanjaju se lozinka i salt vrijednosti iz arraya (niza) kako bi ti podatci bili zaštićeni i nitko ih ne bi mogao vidjeti. Uspješno logiran korisnik preusmjerava se na index.php. To je prikazano na 3.9.

```
<?php

    require("db_connect.php");

    unset($_SESSION['user']);


    header("Location: login.php");
    die("Redirecting to: login.php");
```

Sl. 3.9. - Isječak koda za odjavu korisnika iz aplikacije

Izvrši se spajanje na bazu. Korisnik se ukloni iz session-a (varijabla koja sadrži informacije o jednom korisniku, a dostupna je svim stranicama aplikacije) i preusmjeri na login stranicu.

PHP jezik je korišten i za obavljanje registracije novih korisnika. Unutar forme za registraciju korisnik unosi podatke te se ti podaci pomoću PHP jezika spremaju u bazu podataka. Isječak koda koji to omogućuje prikazan je na slici 3.10.

```php
<?php


    require("db_connect.php");

    if(!empty($_POST))
    {
        if(empty($_POST['username']))
        {
            die("Molimo vas unesite korisnicko ime.");
        }

        if(empty($_POST['password']))
        {
            die("Molimo vas unesite password.");
        }


        if(!filter_var($_POST['email'], FILTER_VALIDATE_EMAIL))
        {
            die("Invalid E-Mail Address");
        }


        $query = "
            SELECT
                1
            FROM users
            WHERE
                username = :username
        ";


        $query_params = array(
            ':username' => $_POST['username']
        );

        try
        {
            $stmt = $db->prepare($query);
            $result = $stmt->execute($query_params);
        }
```

Sl. 3.10. – Isječak koda za registriranje novih korisnika

Prvo se vrši spajanje na bazu, zatim se provjerava je li register forma spremljena. Provjerava se je li korisnik unio korisničko ime, lozinku i ispravan e-mail. Isječak koda koji to omogućuje prikazan je na slici 3.11.

```php
$row = $stmt->fetch();

if($row)
{
    die("Ova E-mail adresa je vec zauzeta, molimo vas izaberite drugu E-mail adresu.");
}

$query = "
    INSERT INTO users (
        username,
        password,
        salt,
        email
    ) VALUES (
        :username,
        :password,
        :salt,
        :email
    )
";
    catch(PDOException $ex)
    {
        die("Failed to run query: " . $ex->getMessage());
    }


    $row = $stmt->fetch();


    if($row)
    {
        die("Ovo korisnicko ime je vec zauzeto, molimo vas izaberite neko drugo.");
    }


    $query = "
        SELECT
            1
        FROM users
        WHERE
            email = :email
    ";

    $query_params = array(
        ':email' => $_POST['email']
    );

    try
    {
        $stmt = $db->prepare($query);
        $result = $stmt->execute($query_params);
    }
    catch(PDOException $ex)
    {
        die("Failed to run query: " . $ex->getMessage());
    }
```

Sl. 3.11. – Isječak koda za provjeru zauzetosti e-mail adrese

Provjerava se je li korisničko ime već zauzeto. Sprema se korisničko ime u varijablu. Provjerava se je li došlo do greške pomoću try/catch bloka. Sve se također provjerava i za e-mail.

U aplikaciji je moguće promijeniti i osobne podatke. Isječak koda koji to omogućava prikazan je na slici 3.12.

```php
if(!empty($_POST))
{
    if(!filter_var($_POST['email'], FILTER_VALIDATE_EMAIL))
    {
        die("Neispravna Email adresa");
    }


    if($_POST['email'] != $_SESSION['user']['email'])
    {

        $query = "
            SELECT
                1
            FROM users
            WHERE
                email = :email
        ";


        $query_params = array(
            ':email' => $_POST['email']
        );

        try
        {
            $stmt = $db->prepare($query);
            $result = $stmt->execute($query_params);
        }
        catch(PDOException $ex)
        {
            die("Failed to run query: " . $ex->getMessage());
        }


        $row = $stmt->fetch();

        if($row)
        {
            die("Ova Email adresa je vec zauzeta, molimo vas izaberite drugu Email adresu.");
```

Sl. 3.12. – Isječak koda za promjenu osobnih podatka

Ukoliko korisnik želi, moguće je da promijeni i lozinku. Pri promijeni lozinke se generira i novi ključ za dešifriranje lozinke. Isječak koda koji to radi prikazuje slika 3.13.

```php
if(!empty($_POST['password']))
{
    $salt = dechex(mt_rand(0, 2147483647)) . dechex(mt_rand(0, 2147483647));
    $password = hash('sha256', $_POST['password'] . $salt);
    for($round = 0; $round < 65536; $round++)
    {
        $password = hash('sha256', $password . $salt);
    }
}
else
{
    $password = null;
    $salt = null;
}

$query_params = array(
    ':email' => $_POST['email'],
    ':user_id' => $_SESSION['user']['id'],
);


if($password !== null)
{
    $query_params[':password'] = $password;
    $query_params[':salt'] = $salt;
}


$query = "
    UPDATE users
    SET
        email = :email
";
```

Sl. 3.13. – Isječak koda za generiranje salt-a

Na slici 3.14. je prikazan isječak koda za promjenu podataka o vozilu.

```php
if(!empty($_POST))
{

    if(!empty($_POST['auti_id'])){
        $query_params = array(
            ':auti_id' => $_POST['auti_id'],
            ':model_id' => $_POST['model_id'],
            ':user_id' => $_SESSION['user']['id'],
        );


        $query = "
            UPDATE users
            SET
                auti_id = :auti_id,
                model_id = :model_id
        ";


        $query .= "
            WHERE
                id = :user_id
        ";
```

Sl. 3.14. – Isječak koda za promjenu podataka o vozilu

U prvom dijeli query obavještava se server o izvršenju update-a na tablicu user i podešavanju marke i modela automobila. Drugi dio nosi informaciju na koji red će se to odnositi, tj. kojem korisniku će se podesiti podatci (korisniku koji je logiran). Isto se vrši i za ostale parametre koji se žele podesiti.

Mali servis se obavlja svakih 10 000 do 15 000 km ovisno o modelu automobila. Izračunava se tako što se kilometre koje je korisnik do tada prešao oduzmu od kilometara na zadnjem malom servisu. Dobiveni podatak se uspoređuje s kilometrima propisanim za obavljanje servisa i korisniku se šalje poruka treba li ili ne obaviti mali servis. Na slici 3.15. je prikazan isječak koda za računanje malog servisa.

```
$mali_servis_br_km=$rows[0]['mali_servis'];
$user_mali_servis=$rows[0]['user_mali_servis'];

$x=$km-$user_mali_servis;

if ($x>=$mali_servis_br_km) {
    $mali_servis=1;
}else{
    $mali_servis=0;
}
```

Sl. 3.15. – Isječak koda za računanje malog servisa

Veliki servis se obavlja svakih 100 000 do 120 000 km, ovisno o modelu automobila. Izračunava se tako što se kilometre koje je korisnik do tada prešao oduzmu od kilometara na zadnjem velikom servisu. Dobiveni podatak se uspoređuje s kilometrima propisanim za obavljanje servisa i korisniku se šalje poruka treba li ili ne obaviti veliki servis. Na slici 3.16. je prikazan isječak koda za računanje velikog servisa.

```
$veliki_servis_br_km=$rows[0]['veliki_servis'];
$user_veliki_servis=$rows[0]['user_veliki_servis'];

$y=$km-$user_veliki_servis;

if ($y>=$veliki_servis_br_km) {
    $veliki_servis=1;
}else{
    $veliki_servis=0;
}
```

Sl. 3.16. – Računanje velikog servisa

Pri velikom servisu se mijenjanju određeni dijelovi automobila, a prikaz toga šta je potrebno promijeniti je omogućen kodom sa slike 3.17.

```
<div aria-expanded="false" id="collapseTwo" class="panel-collapse collapse">
    <div class="panel-body">
        <h4 class="panel-title"> Veliki servis automobila- što točno promijeniti? </h4>

        <p>Većina mehaničara pokušati će što više zaraditi na vama, gdje se najčešće nađete u situaciji gdje mijenjate nešto što nije potrebno,
            što se događa ukoliko ste još neiskusan vozač, takvi pristaju na sve. Zbog toga budemo u ovom poglavlju istaknuli one stvari koje
            se moraju mijenjati tokom velikog servisa automobila.</p>

        <p>Razlike između dizelaša i benzinaca nema, veliki servis je jednak za oboje, jedino postoji mala razlika ako auto ima lanac umjesto
            zupčastog remena. Stoga:</p>

        <p>Ako vaš automobil ima lanac, obavezno mijenjajte:</p>
        <div class="table-responsive">
        <table class="table table-striped table-bordered table-hover">
            <tbody>
                <tr>
                    <td>tračni remen</td>
                </tr>
                <tr>
                    <td>rolice,španer</td>
                </tr>
                <tr>
                    <td>vodenu pumpu</td>
                </tr>
            </tbody>
        </table>
        </div>
        <!-- /.table-responsive -->
        Ako vaš automobil ima zupčasti remen, obavezno mijenjajte: zupčasti remen, tračni remen, rolice, špraner, vodenu pumpu
        <div class="table-responsive">
        <table class="table table-striped table-bordered table-hover">
            <tbody>
                <tr>
                    <td>zupčasti remen</td>
                </tr>
                <tr>
                    <td>tračni remen</td>
                </tr>
                <tr>
                    <td>rolice</td>
                </tr>
                <tr>
                    <td>vodenu pumpu</td>
                </tr>
            </tbody>
```

Sl. 3.17. – Isječak koda za prikaz što je potrebno promijeniti na velikom servisu

3.5. SQL

SQL (engl. *StructuredQueryLanguage*) je programski jezik za pristupanje i manipuliranje bazama podataka. [8] SQL služi za kreiranje, brisanje, pretragu i osvježavanje podataka u relacijskim bazama podataka. Kako bi se SQL primijenio na internet stranici, on zahtijeva relacijsku baze podataka (u ovoj web aplikaciji koristi se MySQL) i skriptni jezik koji se odvija na poslužiteljskoj strani poput PHP-a. Jedan od primjera korištenja SQL jezika iz ove aplikacije prikazan je na slici 3.18.

```
SELECT
    users.id, users.datum_registracije, users.km, users.user_mali_servis, users.user_veliki_servis, auti.auti_name as auti_id,
        modeli.model_name as model_id, modeli.mali_servis as mali_servis, modeli.veliki_servis as veliki_servis
FROM users
LEFT JOIN auti ON users.auti_id = auti.auti_id
LEFT JOIN modeli ON users.model_id = modeli.model_id
WHERE
        id = :user_id
```

Sl. 3.18. – Primjer korištenja SQL jezika/SQL upita

U primjeru sa slike 3.18. naredba *SELECT* se koristi za dohvat svih podataka iz tablice koje se nalaze unutar baze podataka. Zatim se preko stupca „auti_id“ vrši pridruživanje korisnikovog automobila sa nazivom automobila iz tablice „auti“. Također, preko stupca „model_id“

pridružuje se model korisnikovog automobila korisniku. Navedene radnje se rade kako bi se korisniku mogli ispisati podaci o njegovom automobilu.

3.6. Bootstrap

Bootstrap je vrlo jednostavan i dostupan set alata za stvaranje moderne internet stranice i internet aplikacija. On nudi podršku za razne web tehnologije kao što su HTML, CSS, JavaScript i mnogo elemenata koji se mogu lako ugraditi u internet stranice. Za korištenje bootstrapa potrebno je osnovno znanje HTML-a i CSS-a. Interaktivni elementi kao što su gumbi, kutije, izbornici i grafički elementi mogu biti umetnuti koristeći samo HTML i CSS.

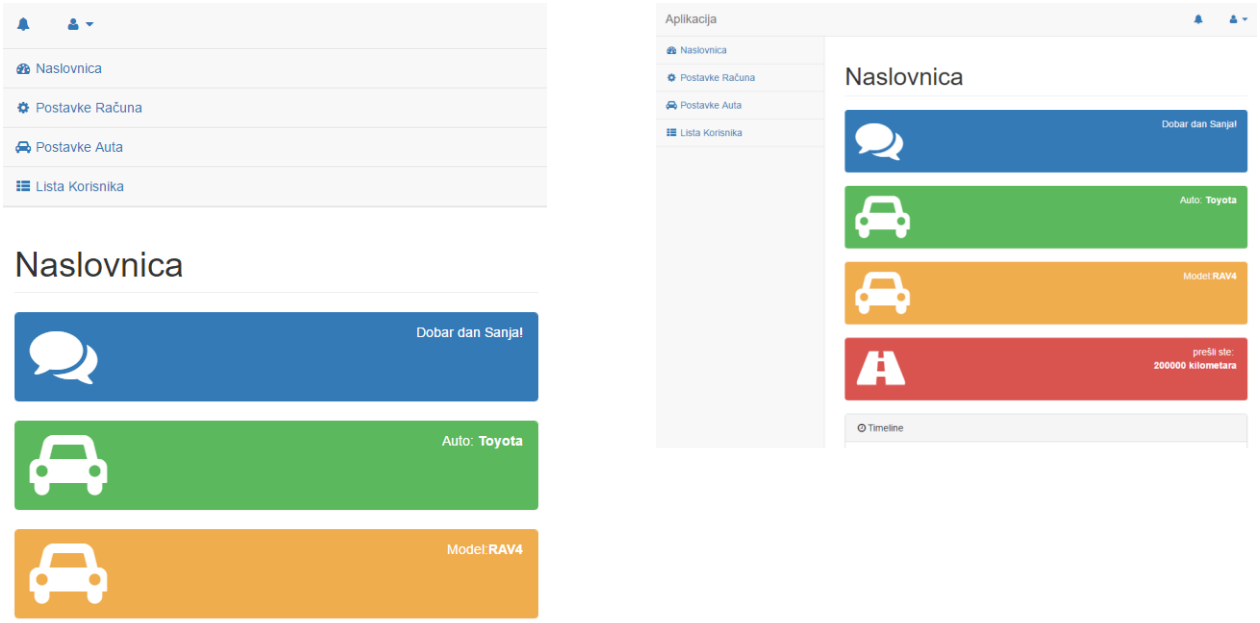
Bootstrap, izvorno nazvan Twitter projektiranje, razvili su Mark Otto i Jacob Thornton kao okvir za poticanje internih alata. Nakon nekoliko mjeseci razvoja od strane male grupe, mnogi programeri na Twitteru su počeli doprinositi projektu u sklopu Hack tjedna, hackathon. Objavljen je kao open source projekt 19. kolovoza 2011.
31. siječnja 2012. godine objavljen je Bootstrap 2. Ovo izdanje je nadopunjeno s „grid layout“ dvanaest stupaca i odgovarajućim dizajnom komponenti, kao i promjenama na mnogim postojećim komponentama. Bootstrap 3 izdanje objavljeno je 19. kolovoza 2013., kada kreće i prvi mobilni pristup. 29. listopada 2014. Mark Otto je najavio razvoj Bootstrap 4. Prva alpha verzija Bootstrap 4 izašla je 19. kolovoza 2015.
Bootstrap je kompatibilan s najnovijim verzijama Google Chrome, Firefox, Internet Explorer, Opera i Safari, iako neki od tih preglednika nisu podržani na svim platformama.
Od verzije 2.0 podržava i prilagodljivi web dizajn. To znači da se izgled web stranica prilagođava dinamički, uzimajući u obzir karakteristike uređaja koji se koristi (desktop, tablet, mobilni telefon). [9]
Kako bi bilo moguće koristit Bootstrap radni okvir (engl. *framework*) u nekoj web aplikaciji prije svega potrebno ga ja uključiti u tu aplikaciju. Uključivanje Bootstrapa u web aplikaciju je prikazano na slici 3.19.

```
<!-- Bootstrap Core CSS -->
<link href="bower_components/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet">
```

Sl. 3.19. – Uključivanje Bootstrap radnog okvira u web aplikaciju

Sa linijom koda (datoteka *header.php*, linija 15) koja je prikazana na slici 3.19. omogućeno je korištenje Bootstrap radnog okvira i svih njegovih dijelova prilikom razvoja i rada web

aplikacije. Bootstrap prilikom izrade web aplikacije omogućuje kreiranje mreže (engl. *grid*) koja dijeli zaslon uređaja na 12 polja. U ovome radu je korištena mreža *.col -lg*. *.col -lg* je klasa koja je namijenjena za desktop uređaje, tj. uređaje koji imaju zaslon veći od 1200 px. U slučaju da je zaslon uređaja manji od navedenog, Bootstrap će prilagoditi prikaz veličini toga zaslona. Primjer prilagođavanja prikaza različitim veličinama zaslona prikazan je na slici 3.20.



Sl. 3.20. - Prikaz prilagođavanja dizajna za ovu web aplikaciju[11]

Bootstrap je „*open source*“ i programeri su ohrabreni na sudjelovanje u projektu i davanje svoga doprinosa na platformi. Bootstrap dolazi s nekoliko JavaScript komponenti u obliku jQuery dodataka. Oni pružaju dodatne funkcije korisničkog sučelja kao što su dijaloški okviri, opisi i carousels. Oni također proširuju funkcionalnost nekih postojećih elemenata sučelja, uključujući, na primjer „auto-complete“ (samoispunjavajuću) funkciju za unos polja. [10]

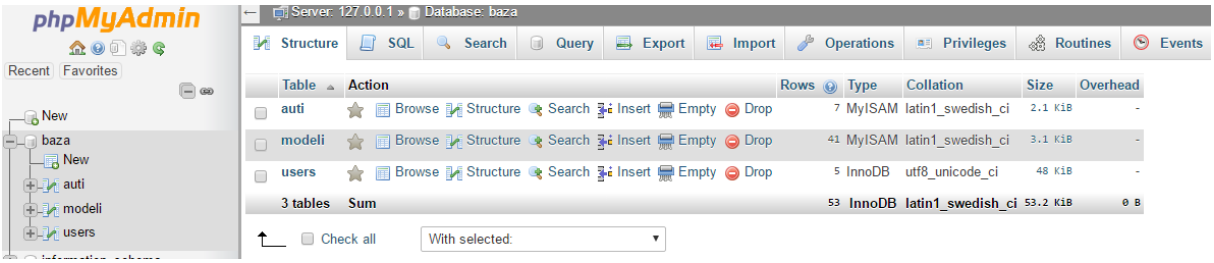
4. STRUKTURA APLIKACIJE

Potrebno je napraviti dinamičku web aplikaciju pomoću koje će registrirani korisnici moći voditi evidenciju o registraciji, servisima i stanju osobnog automobila.

U bazi podataka nalaze se razne marke i modeli automobila koji su povezani preko ID broja. Korisnik treba biti u mogućnosti registrirati se na stranicu sa svojim korisničkim podacima te iz baze podataka odabrati model automobila koji on posjeduje. Dalje treba biti u mogućnosti unijeti podatke o svome automobilu, kao što su prijeđena kilometraža i datum registracije, kako bi aplikacija mogla računati korisnikove nadolazeće obveze. Na kraju kada aplikacija obradi podatke i izvrši izračun ona obavještava korisnika kada i što treba servisirati na automobilu.

4.1. Struktura baze podataka

Baza podataka koja je korištena u ovoj aplikaciji je prikazana na slici 4.1. Sa slike 4.1. vidi se da se baza podataka sastoji od tri tablice – users, modeli i automobili. Unutar navedenih tablica se spremaju podaci o korisnicima i ostali podaci o automobilu.



Sl. 4.1. – Struktura baze podataka

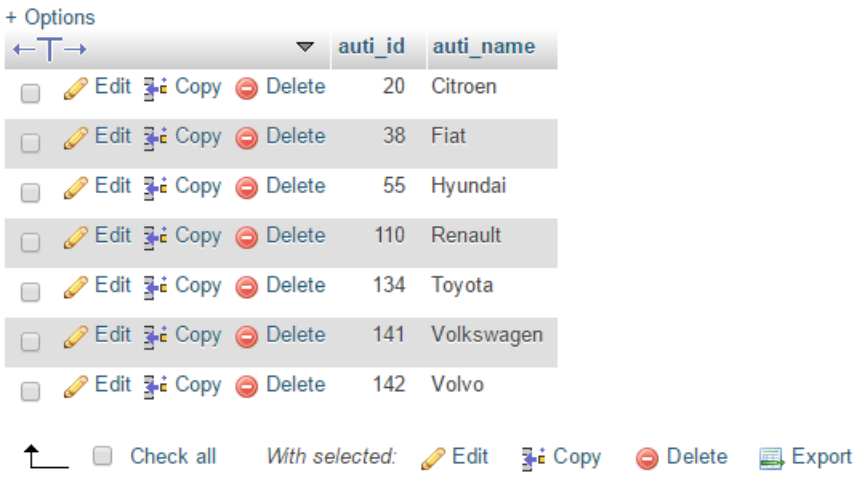
U tablicu „users“ se pohranjuju podaci o korisniku. Ti podaci su ime, lozinka, e-mail, marka automobila, …). Izgled tablice „users“ je prikazan na slici 4.2. ID korisnika je samo redni broj korisnika. Salt je ključ za dekriptiranje lozinke.

	id	username	password	salt	email	auti_id	model_id	km	datum_registracije	user_mali_servis	user_veliki_servis
Edit Copy Delete	6	Ivan	17292dfefd4643eafadcf602b16d155624601d54f7e9a522fc...	7dbb5bca75fe15ea	ivan@yahoo.com	38	486	240000	04/07/2016	225000	120000
Edit Copy Delete	7	Sanja	ae4f73c4e2b464a0ab0c63e17856755652873571e914a1751a...	27b057e675722678	sanja@gmail.com	134	1519	200000	03/01/2016	190000	100000
Edit Copy Delete	8	Mario	851003507156bcc8ac81d7e5687aa1ea2969cc26a4c39b37ce...	433118584d8a1018	mrki@gmail.com	142	1646	190360	02/03/2016	190000	120000
Edit Copy Delete	9	Vedran	0003f7f0cf639c7fc58fd219ee64c2e2c5fd5fa66360d8bf08...	1c5d8c6a6cfb67	vex@etfos.com	55	731	361460	01/05/2016	361000	300000
Edit Copy Delete	10	Martina	729d6b157553499b58f7e098156d84250258e5e4176e883eeb...	e56ebcf48e9dc4d	martina@gmail.com	20	296	198700	12/19/2015	190000	100000

Check all With selected: Edit Copy Delete Export

Sl. 4.2. – Tablica „users“

Druga tablica unutar baze podataka je tablica „automobili“. U njoj su pohranjene marke automobila. Zbog velikog broja automobila na tržištu, u aplikaciju su stavljene samo neke europske marke automobila. ID automobila povezuje automobile sa modelima. Struktura tablice „automobili“ je prikazana na slici 4.3.



+ Options

	auti_id	auti_name
Edit Copy Delete	20	Citroen
Edit Copy Delete	38	Fiat
Edit Copy Delete	55	Hyundai
Edit Copy Delete	110	Renault
Edit Copy Delete	134	Toyota
Edit Copy Delete	141	Volkswagen
Edit Copy Delete	142	Volvo

Check all With selected: Edit Copy Delete Export

Sl. 4.3. – Tablica „automobili“

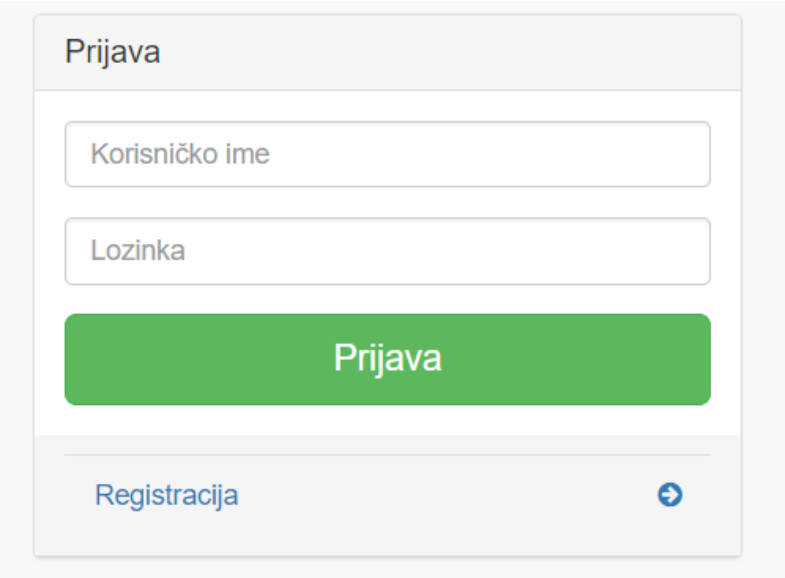
Treća tablica koja se nalazi u bazi podatka je tablica „modeli“. U toj tablici se nalaze modeli automobila od pojedine marke koje se nalaze u tablici „automobili“. Struktura tablice „modeli“ je prikazana na slici 4.4.

	model_id	model_name	auti_id	mali_servis	veliki_servis
Edit Copy Delete	295	C2	20	10000	100000
Edit Copy Delete	296	C3	20	10000	100000
Edit Copy Delete	297	C4	20	10000	100000
Edit Copy Delete	298	C5	20	10000	100000
Edit Copy Delete	300	GS	20	10000	100000
Edit Copy Delete	301	Saxo	20	10000	100000
Edit Copy Delete	483	Brava	38	15000	120000
Edit Copy Delete	484	Bravo	38	15000	120000
Edit Copy Delete	486	Coupe	38	15000	120000
Edit Copy Delete	491	Punto	38	15000	120000
Edit Copy Delete	492	Regata	38	15000	120000
Edit Copy Delete	501	Uno	38	15000	120000
Edit Copy Delete	718	Accent	55	15000	120000
Edit Copy Delete	720	Elantra	55	15000	120000
Edit Copy Delete	728	i30	55	15000	120000
Edit Copy Delete	731	Santa Fe	55	15000	120000
Edit Copy Delete	733	Sonata	55	15000	120000
Edit Copy Delete	736	Tucson	55	15000	120000
Edit Copy Delete	737	Veracruz	55	15000	120000
Edit Copy Delete	1328	Clio	110	15000	120000
Edit Copy Delete	1331	Laguna	110	15000	120000
Edit Copy Delete	1333	Megane	110	15000	120000

Sl. 4.4. – Tablica „modeli“

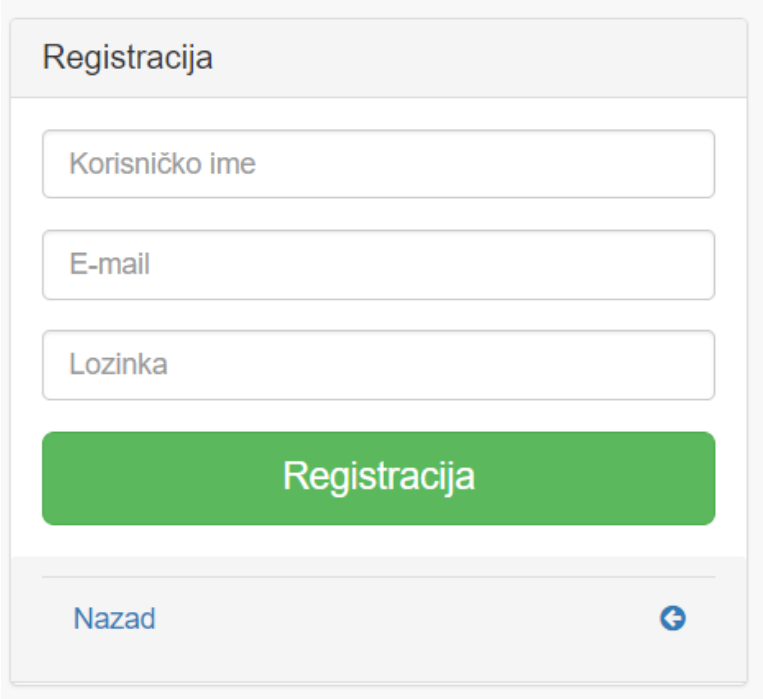
4.2. Korisničko sučelje

Ovo poglavlje prikazuje pojedina sučelja aplikacije. Početno sučelje aplikacije je forma za prijavu u aplikaciju. Forma za prijavu u aplikaciju je prikazana na slici 4.5.



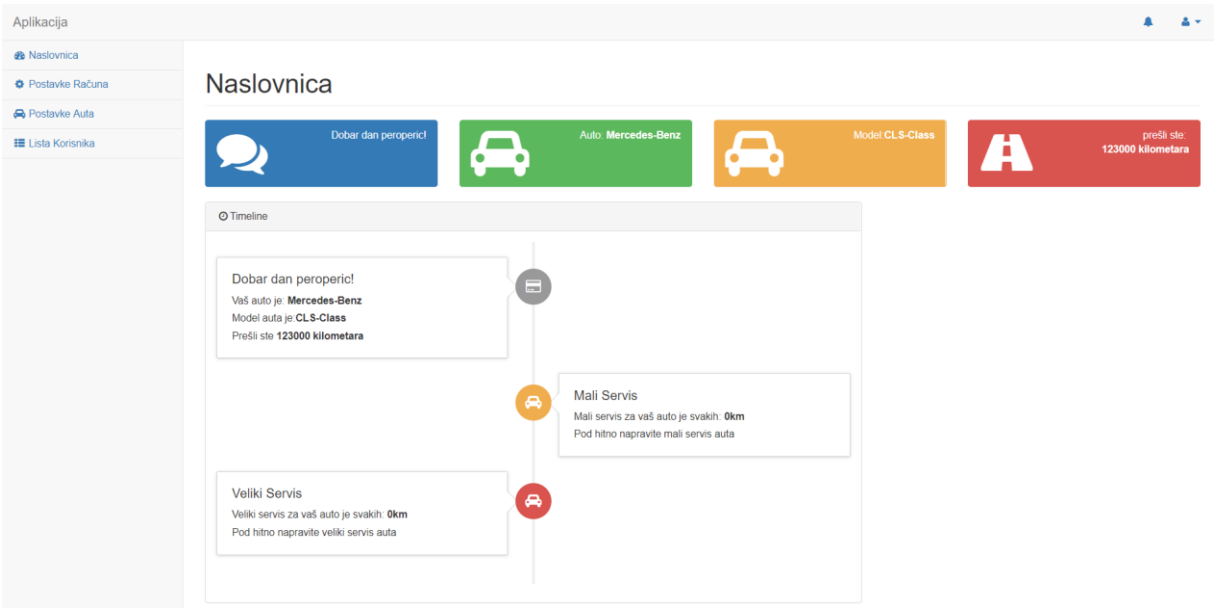
Sl. 4.5. – Sučelje za prijavu u aplikaciju

Ukoliko korisnik nije registriran, prije prijave potrebno je da obavi registraciju. Registraciju je moguće obaviti pomoću forme koja je prikazana na slici 4.6., a ona se otvara klikom na tekst „Registracija“ koji se nalazi ispod gumba za prijavu.



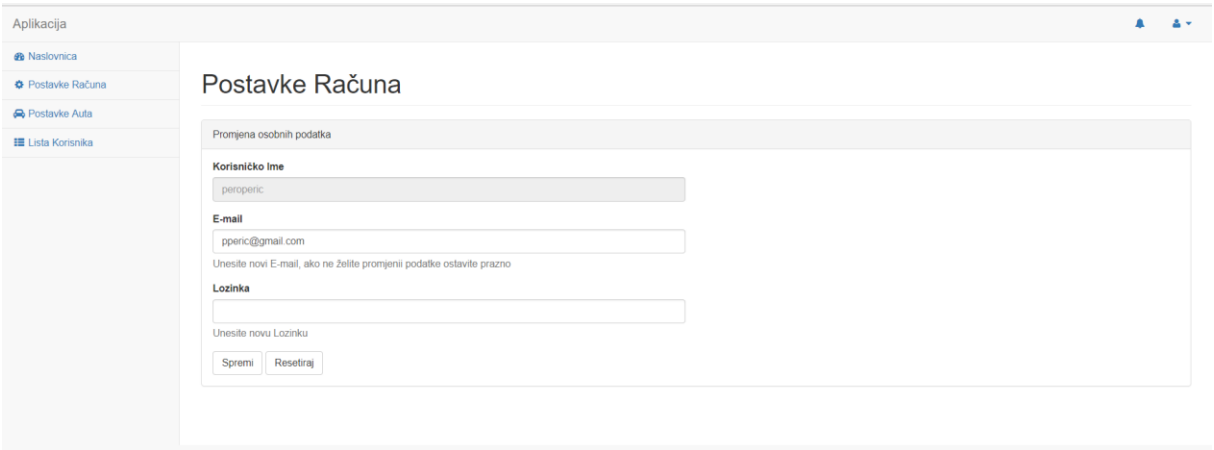
Sl. 4.6. – Sučelje za registraciju korisnika

Nakon prijave (ili potrebne registracije) korisniku se prikaže početno sučelje aplikacije. Početno sučelje aplikacije je prikazano na slici 4.7.



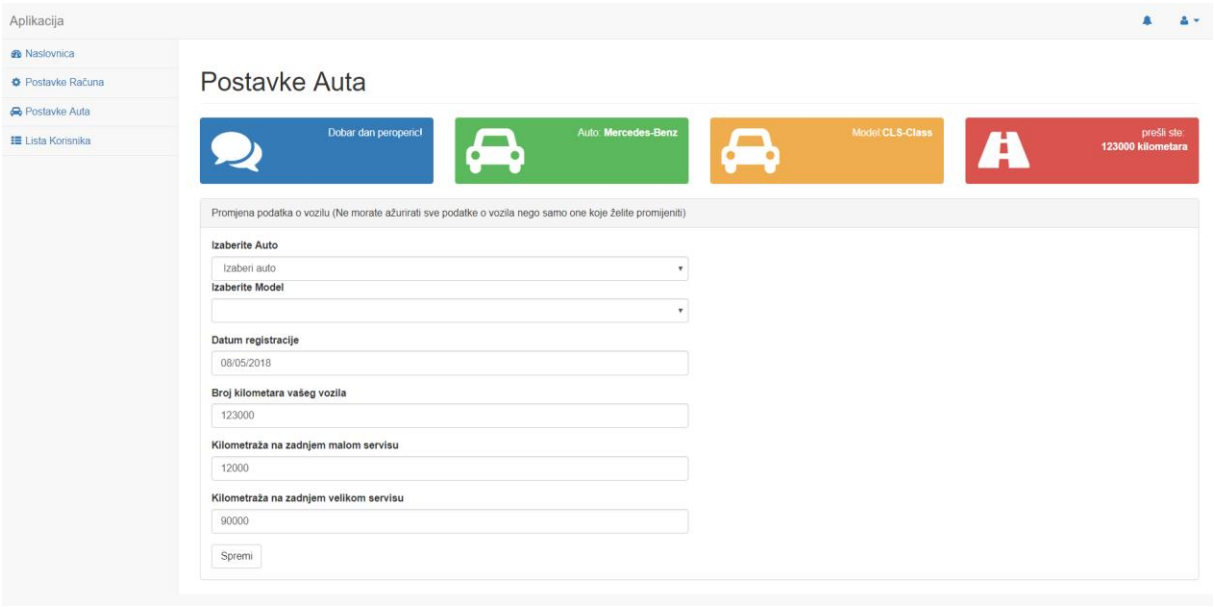
Sl. 4.7. – Početno sučelje aplikacije

U lijevom dijelu zaslona se nalazi izbornik koji nudi opcije promjene postavi računa, promjene podataka o automobilu i pregled svih registriranih korisnika. Sučelje za promjenu postavki računa prikazano je na slici 4.8.



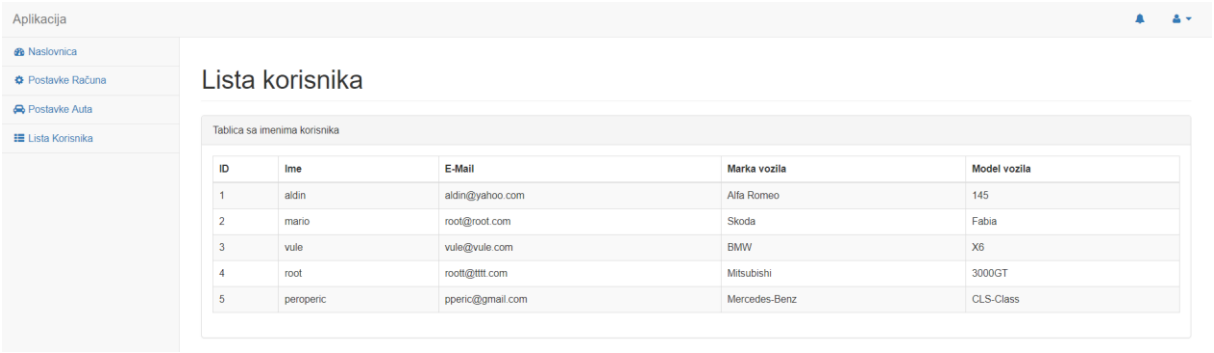
Sl. 4.8. – Sučelje za promjenu postavi računa

U predviđena polja moguće je unijeti nove podatke, te klikom na gumb „Spremi“ ti podaci se pohranjuju u bazu podataka. Druga opcija u izborniku je promjena podataka o automobilu. Sučelje za promjenu podataka o automobilu je prikazano na slici 4.9.



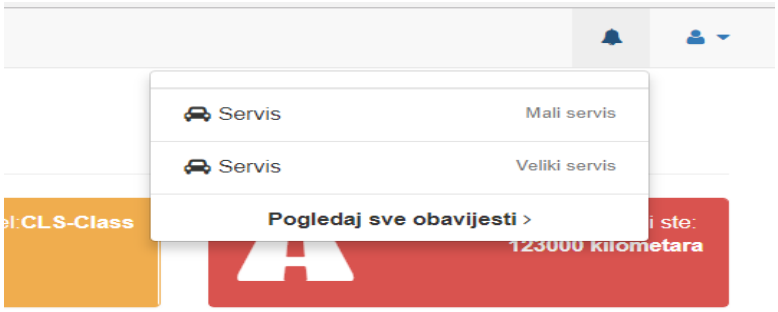
Sl. 4.9. – Sučelje za promjenu podataka o automobilu

Zadnja opcija u izborniku je pregled liste korisnika. Lista korisnika sadrži podatke o korisnicima i njihovim automobilima. Sučelje koje prikazuje listu korisnika se nalazi na slici 4.10.



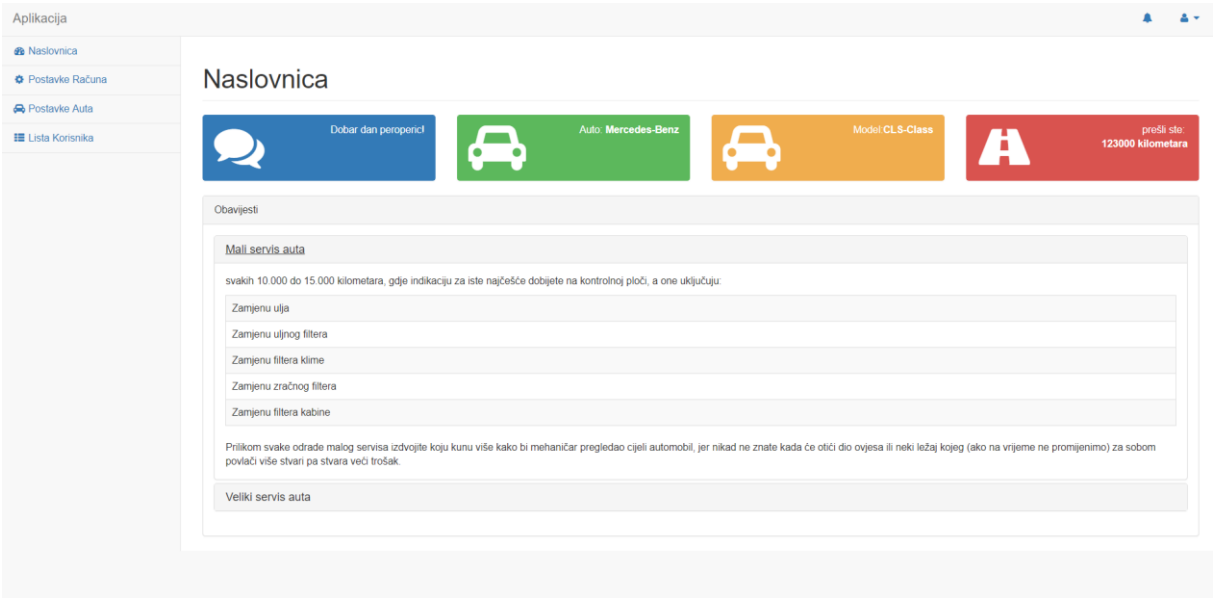
Sl. 4.10. – Sučelje za prikaz liste korisnika

U gornjem desnom kutu zaslona se nalazi simbol zvona koji prikazuje podatke o malom i velikom servisu automobila. Klikom na njega se otvara izbornik u kojem je moguće odabrati mali ili veliki servis. Sučelje za taj izbornik se nalazi na slici 4.11.



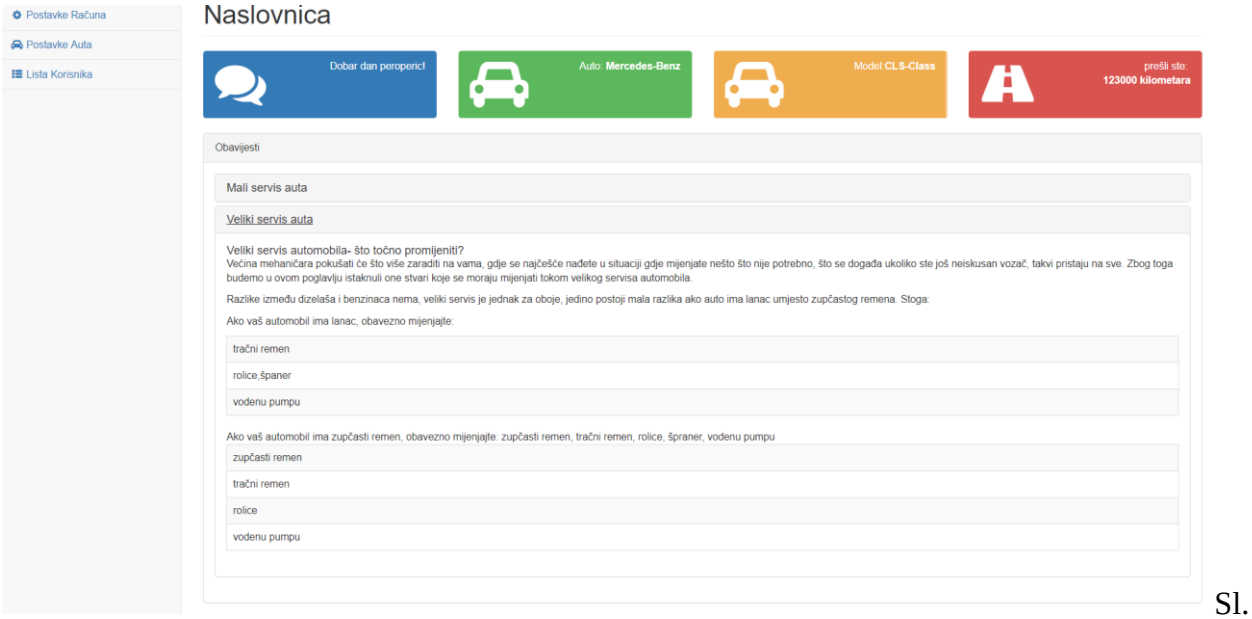
Slika 4.11. – Izbornik sa opcijama „Mali servis“ i „Veliki servis“

Klikom na opciju „Mali servis“ otvara se sučelje kao na slici 4.12.



Sl. 4.12. – Sučelje za opciju „Mali servis“

Klikom na opciju „Veliki servis“ otvara se sučelje kao na slici 4.13.

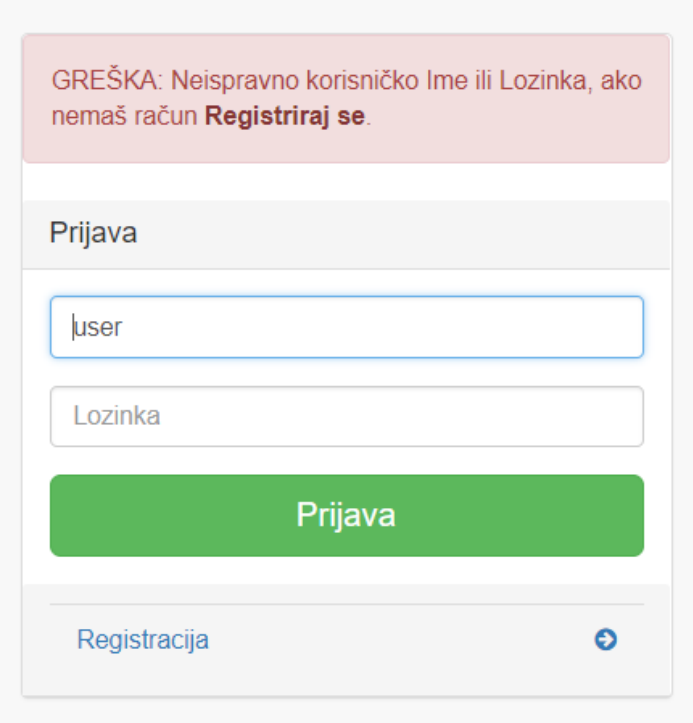


Sl. 4.13. – Sučelje za opciju „Veliki servis“

5. TESTIRANJE RADA APLIKACIJE

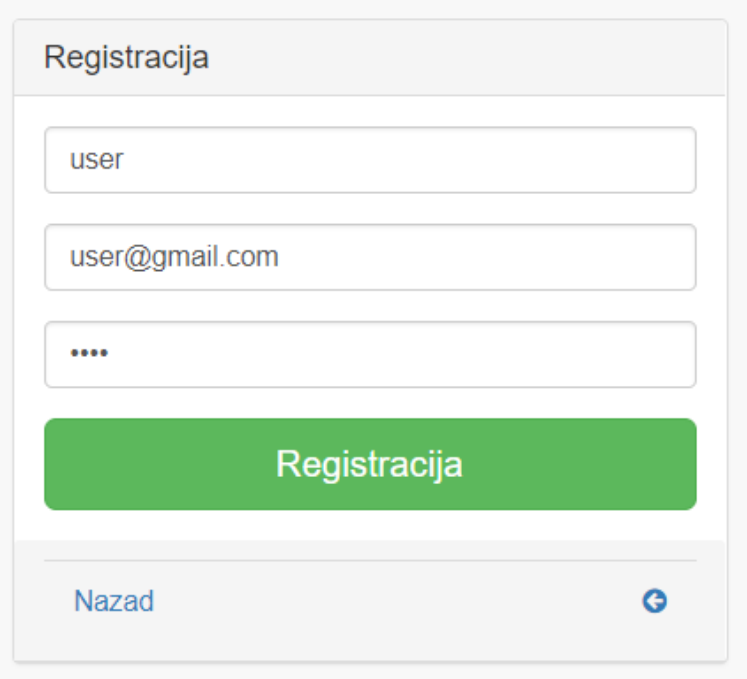
U ovome poglavlju je dan prikaza testiranja aplikacije. Aplikacije je testirana u nekoliko testnih slučajeva. Na osnovu tih testnih slučajeva i prethodnog znanja o tome šta bi se trebalo desiti u aplikaciji izvukao se zaključak je li aplikacija dobro obavila posao koji joj je zadan.

Prvi testni slučaj je prijava u aplikaciju. U aplikaciju je moguće se prijaviti samo ukoliko je korisnik prethodno registriran. U suprotnome, aplikacija će javiti poruku o grešci. Na slici 5.1. je prikazana poruka o grešci u slučaju kada prijava nije moguća.



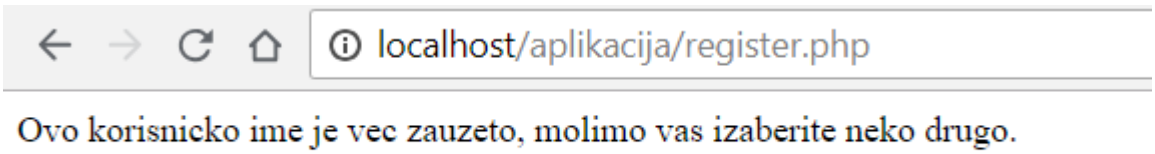
Sl. 5.1. – Poruka o grešci kod neuspjele prijave u aplikaciju

Sa slike 5.1. jasno se vidi da je prije same prijave u aplikaciju potrebno obaviti registraciju. U samoj poruci o greški dan je i link na formu za registraciju. Popunjena forma za registraciju prikazana je na slici 5.2. Nakon unosa traženih podataka, aplikacija preusmjerava korisnika na stranicu za prijavu u aplikaciju.



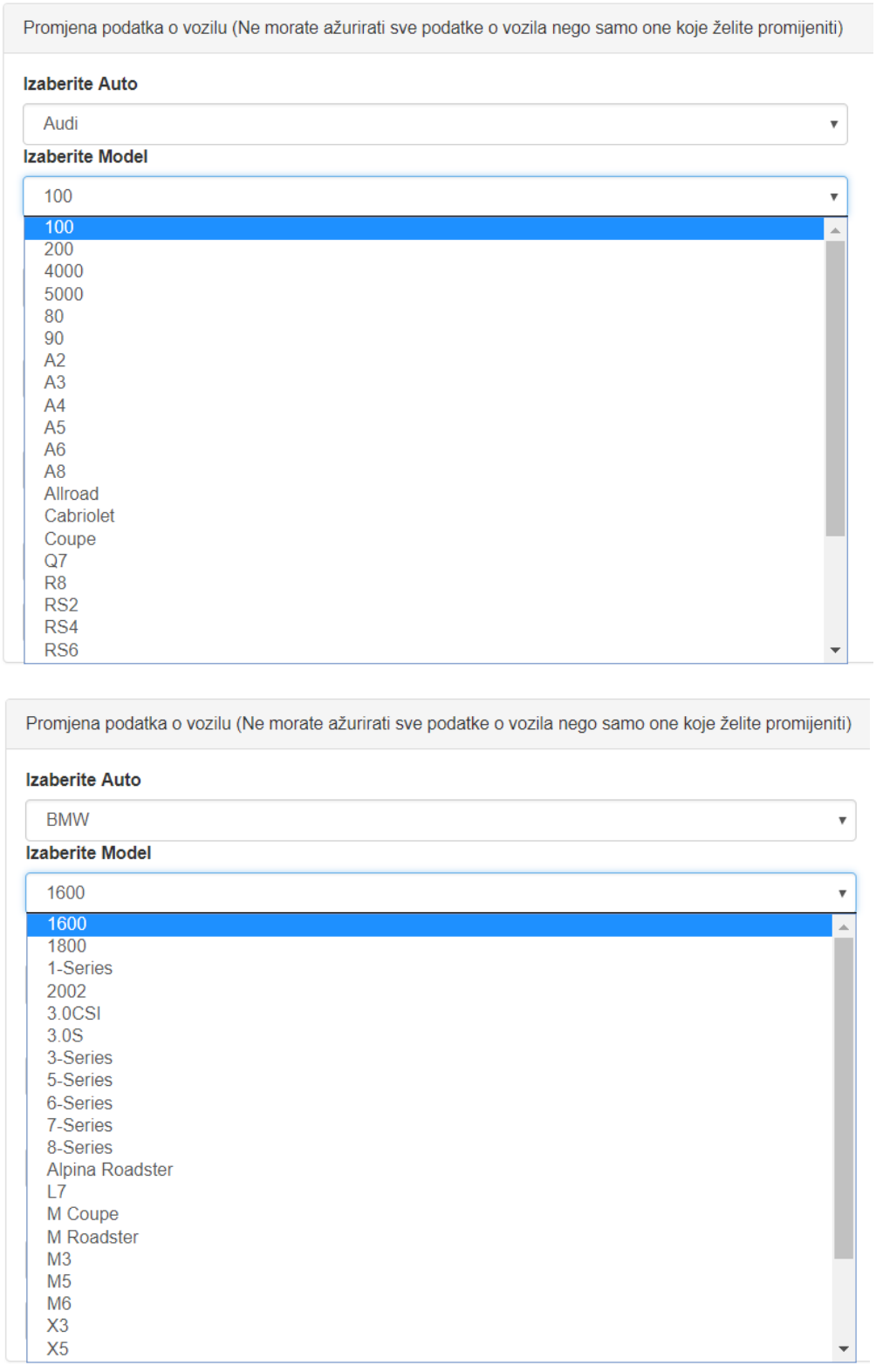
Sl. 5.2. – Popunjena forma za registraciju

Ukoliko je korisnik odabrao korisničko ime koje je već zauzeto, aplikacija će mu javiti poruku o grešci u kojoj piše da odabere neko drugo korisničko ime – prikazano na slici 5.3.



Sl. 5.3. – Prikaz poruke o grešci prilikom registriranja korisnika

Poslije obavljene registracije korisnika, omogućena mu je prijava u aplikaciju. Nakon prijave u aplikaciju korisnik može započeti sa podešavanjem parametara o automobilu. Također, omogućena mu je i promjena osobnih podataka. Forma pomoću koje korisnik podešava parametre svog automobila sadrži dva padajuća izbornika. Prvi padajući izbornik nudi korisniku odabir marke automobila, a drugi padajući izbornik nudi izbor nekog modela od prethodno odabrane marke. Na slici 5.4. su prikazana dva slučaja odabira marke i modela automobila.

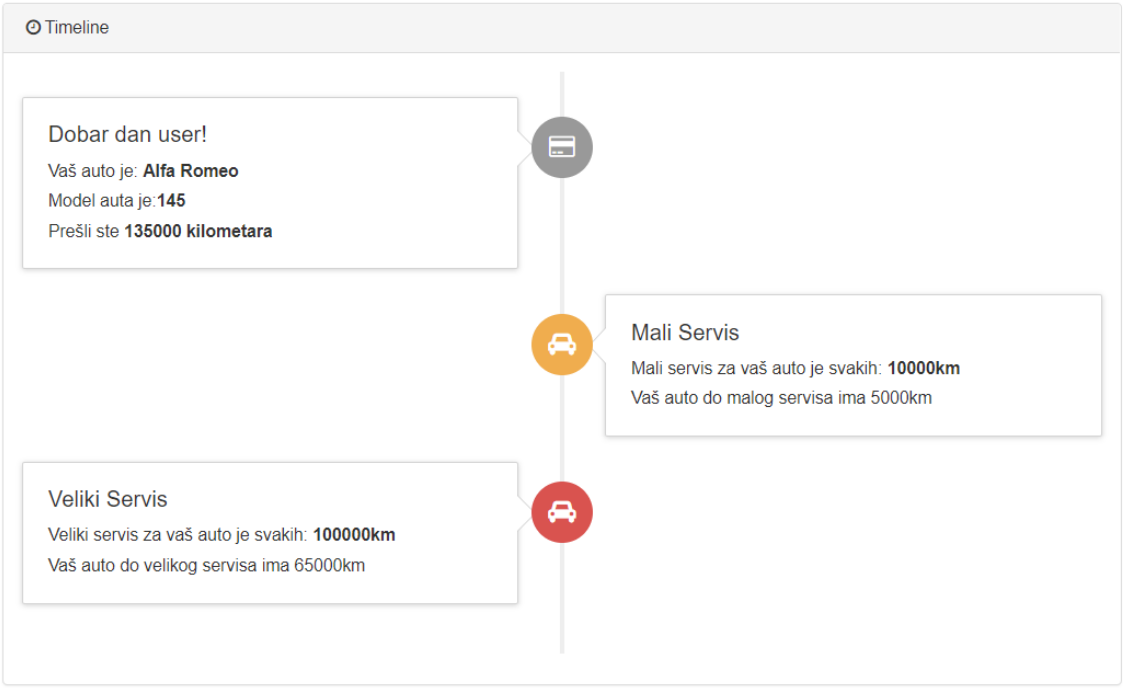


Sl. 5. 4. – Padajući izbornici za odabir marke i modela automobila

Sa slike 5.4. se vidi da ukoliko korisnik odabere marku automobila „Audi“, u drugom padajućem izborniku se nudi odabir nekog od modela automobila koje proizvodi „Audi“ (gornja dio slike).

Također, ista stvar je i ako korisnik odabere marku automobila „BMW“. Tada mu se nude modeli koje je proizveo „BMW“ (donji dio slike).

Nakon unesenih parametara za korisnikov automobil, aplikacije će na osnovu izračuna dati korisniku informacije o potrebom za velikim ili malim servisom. Aplikacije će izračunati koliko je moguće još preći kilometara sa automobilom prije potrebnog servisa. Na slici 5.6. je prikazano sučelje za testni primjer gdje aplikacija daje korisniku informaciju o preostalim kilometrima do servisa.



Sl. 5.6. – Informacije korisniku o potrebi za servisom automobila

Ukoliko je vozilo prešlo određeni broj kilometara potrebnih za servis, aplikacija će mu ispisati poruku da je potrebno napraviti servis. U tome slučaju će pisati da potrebno pod hinto napraviti veliki ili mali servis. [11]

Testiranje ove aplikacije je obuhvatilo osnovne radnje u aplikaciji. Također, testirane su i isprobane funkcionalnosti ove aplikacije. Na osnovu provedenog testiranja može se zaključiti da je aplikacija točno obavljala postavljene zadatke. Jednostavno rečeno, aplkacija radi ono što bi i trebala, odnosno obavlja posao za koji je i napravljena. Pri radu sa aplikacijom tijekom testiranja nije bilo zamjerki na njen rad.

6. ZAKLJUČAK

Razvoj pametnih telefona i njihova svakodnevna upotreba dovode do razvoja aplikacija za pametne telefone za gotovo svaki aspekt života. Osobni automobili su također postali nezamjenjivi u svakodnevnom životu. Kako se o njima potrebno redovito skrbiti, a velik broj vlasnika automobila ne zna što i kada treba raditi kako bi dobro održavali svoje vozilo, stvara se potreba za razvojem aplikacije koja bi im u tome pomogla. Tako nastaje i ova aplikacija koja korisniku omogućava brzo i jednostavno snalaženje sa svojim obvezama oko automobila.

LITERATURA

[1] http://www.ibmbigdatahub.com/blog/brief-history-application-development [ozujak 2016.]

[2] https://bib.irb.hr/datoteka/532594.Skripta_-_Uvod_u_xhtml_html_i_css.pdf [sijecanj 2016.]

[3] Alex MacCaw, "JavaScript Web Applications", 2011.

[4] David Sawzef McFarland, "JavaScript & JQuery", 2008.

[5] Nicholas C. Yakas, "Maintainable JavaScript", 2012.

[6] David Powers, "PHP Solutions", 2010.

[7] http://php.net/manual/en/book.pdo.php [veljaca 2016.]

[8] http://www.durangedesign.com/blog/p/mysql-vehicle-makes-and-models-database-sql- files-up-to-2009 [veljaca 2016]

[9] http://startbootstrap.com/template-overviews/sb-admin-2/ [ozujak 2016.]

[10] https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework) [ozujak 2016.]

[11] http://www.oktan.hr/veliki-servis-automobila-na-sto-sve-pripaziti/ [travanj 2016.]

SAŽETAK

Cilj je bio napraviti aplikaciju koja će korisniku pomagati pri redovitom izvršavanju obveza servisiranja automobila. Napravljena aplikacija omogućuje korisniku uspostavljanje svoga korisničkog računa, logiranje i unos podataka o svojem automobilu te primanje obavijesti o nadolazećim obvezama vezanim uz servis automobila.

Na jednostavan način, nakon logiranja korisnika, korisniku se prikazuje naslovna stranica s podacima o njegovom automobilu te zvonce (alarm) koje se pomiče ukoliko je potrebno obaviti neki od servisa. Također mu se pojavljuju obavijesti što točno treba odraditi na pojedinom servisu kako nešto ne bi previdio ili pak bio prevaren od strane automehaničara.

Ključne riječi: web aplikacija, baza podataka, PHP, HTML, jQuery, JavaScript, MySQL, bootstrap

ABSTRACT

WEB APLICATION TO KEEP RECORDS OF A CAR MAINTENANCE WITH RESPONSIVE DESIGN

The aim of this paper is to design an application that will be used to help a user in keeping regular maintenance of their car. When using this application it is possible to create a user account, log into the account, enter information about a user's car and receive information on due date for the next service as well as a type of a car service needed.

Following the log-in, a home page opens displaying information on a user's car and a bell (alarm) that signals a need to get a car serviced. It also displays information on what the upcoming service should include in order to make necessary changes/repairs and to avoid being cheated by a car mechanic.

Key words: web application, data base, PHP, HTML, jQuery, JavaScript, MySQL, bootstrap

ŽIVOTOPIS

Sanja Gal rođena je u Osijeku 16. listopada 1991. Pohađa Osnovnu školu Dobriše Cesarića, a potom upisuje Prvu gimnaziju u Osijeku. Nakon srednje škole upisuje Elektrotehnički fakultet u Osijeku 2010. godine smjer Računarstvo na Preddiplomskom studiju. Preddiplomski studij završava 2014. godine te iste godine upisuje diplomski studij, smjer Procesno računarstvo.

Sanja Gal