

Web aplikacija za objavu oglasa

Udovičić, Dragana

Master's thesis / Diplomski rad

2020

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://um.nsk.hr/um:nbn:hr:200:462140>

Rights / Prava: [In copyright](#) / [Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2025-03-09**

Repository / Repozitorij:

[Faculty of Electrical Engineering, Computer Science and Information Technology Osijek](#)



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

Sveučilišni studij

WEB APLIKACIJA ZA OBJAVU OGLASA

Diplomski rad

Dragana Udovičić

Osijek, 2020.

SADRŽAJ

1. UVOD	1
1.1. Zadatak diplomskog rada	1
2. PREGLED PODRUČJA.....	2
3. OPIS PRIMIJENJENE TEHNOLOGIJE.....	3
3.1. NodeJS	3
3.2. ExpressJS	4
3.3. MongoDB.....	6
3.4. Clouinary NodeJS.....	9
3.5. EJS	10
3.6. CSS.....	10
3.7. Bootstrap	11
3.8. ZingChart	11
4. PRIMJENA OPISANIH TEHNOLOGIJA U IZRADI WEB APLIKACIJE.....	13
4.1. Registracija i prijava korisnika.....	13
4.2 Kreiranje objave, brisanje i izmjena oglasa i komentara	16
4.3. Pretraživanje oglasa	24
4.4. Grafički prikaz podataka.....	25
5. OPIS FUNKCIONALNOSTI I IZGLEDA WEB APLIKACIJE.....	26
6. ZAKLJUČAK.....	33
LITERATURA	34
SAŽETAK.....	36
ABSTRACT	37
ŽIVOTOPIS.....	38
PRILOZI.....	39

1. UVOD

Tema ovog rada je izrada web aplikacije za objavu oglasa, kao primjer kreirana je web aplikacija za prodaju automobila „AutoProdaja“. Korisnici aplikacije mogu biti prijavljeni i neprijavljeni. Mogućnosti pregledavanja i pretraživanja oglasa imaju prijavljeni i neprijavljeni korisnici, a samo prijavljeni korisnici imaju mogućnosti grafičkog prikaza pojedinih podataka automobila, izrade novog oglasa, izmjene starog oglasa, komentiranje oglasa, te brisanje navedenog. U aplikaciji također korisnici su podijeljeni prema ulogama. Postoje dvije vrste uloga: obični korisnici i admin. Svi novi korisnici prema zadanoj vrijednosti su obični korisnici. Prijavljeni korisnici pri izradi novog oglasa moraju ispuniti osnovni obrazac za oglas u kojem upisuju naziv, cijenu automobila, kontakt telefon, vrstu motora, boju, marku, model, prijeđene kilometre, godinu proizvodnje, sliku, kratki opis oglasa. Važno je upisati jasne informacije jer pomoću ovih informacija drugi korisnici mogu jednostavnije pronaći željeni oglas u tražilici. Korisnici mogu pretraživati automobile koji su u prodaji, dok prodane automobile je moguće vidjeti na vlastitom korisničkom profilu. Korisnici s admin ulogom jedini imaju pristup svim informacijama te prava izmjene i brisanja svih podataka.

U prvom poglavlju navest će se glavne razlike ove web aplikacije u odnosu na slične aplikacije na tržištu. Zatim u drugom dijelu ovog rada opisat će se tehnologije koje su korištene pri izradi ove web aplikacije poput NodeJS, ExpressJS, EJS, MongoDB. U trećem poglavlju detaljno će biti objašnjen proces izrade i sam izgled aplikacije, a u zadnjem poglavlju osvrnut će se na konačni projekt, njegovu svrhu i mogućnosti poboljšanja.

1.1. Zadatak diplomskog rada

Zadatak diplomskog rada je izrada web aplikacije koja ima mogućnosti prijave korisnika, pregledavanje, pretraživanje i objavljivanje oglasa. Prijavljeni korisnik ima mogućnosti dodavanja novog oglasa, izmjene ili brisanja vlastitog oglasa, dok neprijavljeni korisnik ima samo mogućnost pregledavanja i pretraživanja oglasa. Korisnici se razlikuju prema ulogama, a mogu biti prijavljeni kao obični korisnik ili kao admin. Korisnici s ulogom admina imaju veća prava od običnih korisnika.

2. PREGLED PODRUČJA

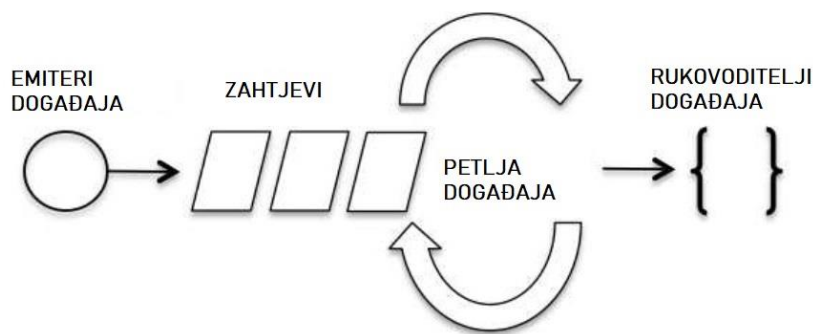
Na tržištu postoje mnogobrojne web aplikacije za prikazivanje oglasa. Većina je namijenjena širem oglašavanju proizvoda. Takve aplikacije su često nepregledne jer ne sadrže mogućnosti pretraživanja po pojedinim poljima ili sadrže mnogobrojne vrste proizvoda i time otežavaju pronalazak ciljanog sadržaja. „AutoProdaja“ je primjer web aplikacije za oglašavanje ciljanog tipa proizvoda, već prema tome je jednostavnija za upotrebu od većine drugih aplikacija za oglašavanje automobila. Samo sučelje aplikacije je vrlo jednostavno i lako razumljivo, te u samo par klikova moguće je objaviti željeni oglas. Osim objave oglasa, upisivanjem traženih specifikacija u predviđena polja aplikacija omogućuje pretraživanje oglasa. Specifičnost aplikacije je u tome što pruža grafički prikaz glavnih podataka o automobilu (cijene, prijeđenih kilometara, godine proizvodnje) za sve oglase na jednom mjestu. Preko grafičkog prikaza korisnici mogu uspoređivati cijene, prijeđene kilometre i godinu proizvodnje automobila iz različitih oglasa. Na ovaj način korisnici mogu brže odlučiti koji od navedenih oglasa ih zanima, te prema tim parametrima odabrati koji oglas žele detaljnije pogledati.

3. OPIS PRIMIJENJENE TEHNOLOGIJE

U ovom poglavlju bit će opisane glavne tehnologije koje su korištene pri izradi ovog projekta.

3.1. NodeJS

NodeJS je *open-source*¹ platforma na strani poslužitelja koja služi za izradu brzih *real-time*² aplikacija, a razvio ju je Ryan Dahl 2009. godine. NodeJS aplikacije su napisane u JavaScript programskom jeziku koji je glavni predstavnik klijentskih programskih jezika, odnosno jezika gdje se kod interpretira na poslužitelju kako je navedeno u [1].



Slika 3.1. Prikaz načina rada NodeJS, prema [1]

Na slici 3.1. prikazan je način rada NodeJS-a. Svi zahtjevi koje korisnici šalju obrađuju se unutar jedne niti. NodeJS temelji se na događajima koji su pokrenuti asinkrono, zahtjevi se obrađuju jedan za drugim, tj. jedna radnja ne blokira drugu. Glavna značajka načina rada NodeJS-a je postojanje glavne petlje događaja koja osluškuje događaje. Asinkronost i baziranost na događaje su velike prednosti NodeJS-a te rezultiraju brzim izvršavanjem koda bez čekanja.

Uz instalaciju NodeJS-a dolazi alat zvan NPM (engl. *Node Package Manager*). NPM je alat koji omogućuje repozitorij NodeJS paketa, jednostavno instaliranje paketa, te predstavlja standard za definiranje ovisnosti o ovim paketima. Svi paketi su dostupni na <http://npmjs.org/> službenoj stranici. Kako bi koristili ove pakete u vlastitom projektu potrebno je napisati samo jednu liniju koda za instalaciju (npr. `npm install express`) te dodatno u kodu zatražiti pristup (npr.

¹ *Open-source* platforma je platforma otvorenog izvora koju je moguće je uređivati i poboljšati. Prema tome NodeJS je besplatni alat koji je svima dostupan.

² *Real-time* aplikacijski program je program koji funkcionira unutar vremenskog okvira koji korisnik smatra neposrednim ili trenutnim.

`require("express"))`). Ovako instalirani paketi su instalirani lokalno u projektu i nalaze se u `node_modules` direktoriju. Uz lokalno instaliranje paketa moguće je paket instalirati i globalno (npr. `npm install express -g`) gdje će paket biti spremljen na jednom mjestu. Preporučuje se instalacija paketa lokalno kako bi sve aplikacije koje ga koriste mogle koristiti različitu verziju koja im odgovara. U NodeJS aplikaciji sve informacije o paketima nalaze se u direktorijima `package.js` i `package-lock.json`. Ove pakete dodajemo u projekt s narednom „`npm init`“. Direktorij `package.js` sadrži listu paketa o kojima ovisi projekt, verzije paketa, te omogućuje jednostavnije ponovno instaliranje paketa koje projekt koristi uz pomoću naredbe „`npm install`“. `Package-lock.json` se automatski osvježi nakon što se dogode promjene u `node_modules` ili `package.js` direktoriju, a omogućuje prikazivanje promjena u direktorijima, prema [3].

NodeJS je izdan pod MIT licencom koja dozvoljava upotrebu softvera za kopiranje, izučavanje i pretvaranje u vlastiti softver, odnosno moguće je služiti se nečijim kodom i tako izmijenjeni kod nije potrebno dijeliti s osobom koja je napisala originalni kod. Zbog navedenih pogodnosti, ova platforma je vrlo zastupljena u današnjem informatičkom svijetu, a neke od poznatih tvrtki koji ju koriste su Microsoft, IBM, LinkedIn, Paypal, Netflix, kao što je navedeno u [1].

3.2. ExpressJS

ExpressJS je NodeJS okruženje za razvoj web aplikacija (engl. *framework*), a njegova svrha je olakšati kreiranje web aplikacije. Osnovna mu je karakteristika što je minimalan, tj. nije kompliciran te pruža samo najpotrebnije. Upravo zbog tog svojstva je vrlo fleksibilan te omogućuje implementiranje različitih rješenja, prema [4]. Osnova svojstva ovog okruženja su: postavljanje posredničkog softvera za odgovaranje na HTTP zahtjeve, definira usmjeravanje (engl. *routing*) različitih radnji na temelju HTTP metode i URL-a i omogućuje dinamičko prikazivanje HTML stranica na temelju proslijeđenih argumenata u predlošku, prema [5]. Jednostavno instaliranje ExpressJS-a je već objašnjeno u prethodnom poglavlju, a primjer jednostavne aplikacije koja ispisuje poruku prikazan je na slici 3.2. Aplikacija se pokreće na 3000 portu, te ispisuje navedenu poruku.

```

var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('Jednostavna način kreiranja ExpressJS aplikacije');
})

var server = app.listen(function (req,res,next) {
  res.locals.curruser=req.user;
  res.locals.error=req.flash("error");
  res.locals.success=req.flash("success");
  next();

  console.log("Server is running on port 3000...", host, port)
})

```

Slika 3.2. *Primjer jednostavne aplikacije*

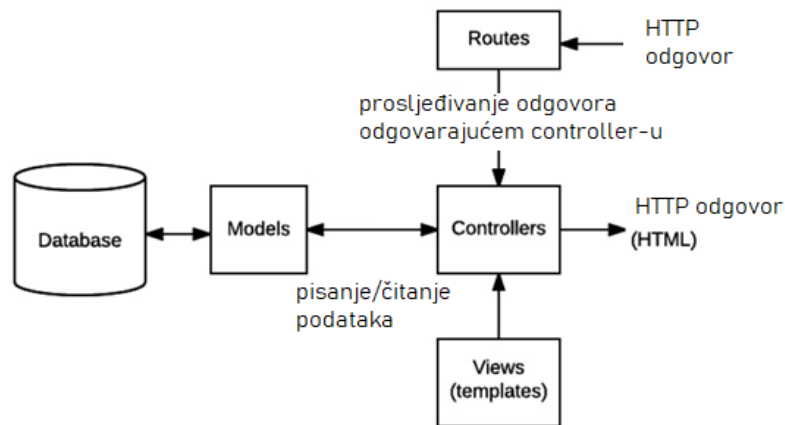
Na slici 3.2. možemo primijetiti *callback*³ funkciju koja ima parametre *req* i *res*. *Req* predstavlja HTTP zahtjev (engl. *request*), dok *res* parametar predstavlja odgovor koji ExpressJS aplikacija šalje kada dobije HTTP zahtjev (engl. *response*). Rezultat koda iz slike 3.2. je ispisivanje željene poruke na početnoj stranici (engl. *homepage*) koristeći metodu GET. GET metoda jedna je od HTTP metoda, a primjeri nekih drugih metoda su POST, DELETE, PUT. Koristimo različite HTTP metode u ovisnosti o tome želimo li slati, dohvaćati ili brisati sadržaj, prema [5]. Struktura NodeJS-a i ExpressJS-a nije strogo definirana, no kao osnovnu strukturu može se uzeti ovakav oblik:

- *controllers* - direktorij gdje se definiraju rute aplikacije,
- *helpers* - direktorij gdje se nalazi kod koji se primjenjuje u više različitih dijelova projekta,
- *middlewares* - direktorij gdje se nalaze middlewares koji obrađuju dolazne zahtjeve prije nego što ih obrade prema rutama,
- *models* - direktorij koji predstavlja podatke, implementira logiku i upravlja pohranom,
- *public* - direktorij koji sadrži sve statičke datoteke poput slika i stilova. ExpressJS pruža ugrađeni *express.static* međuprograme koji poslužuju statičke datoteke, a koristi se jednostavno tako što mu se proslijedi ime direktorija gdje se statički podatci nalaze npr. *app.use(express.static('public'))*; prema [5],
- *views* - direktorij koji pruža predloške koji se generiraju i poslužuju na rutama,
- *tests* - direktorij gdje se testira sve što se nalazi u ostalim direktorijima,
- *app.js* - datoteka koja inicijalizira aplikaciju,
- *package.json* - datoteka koja sadrži informacije o svim paketima, prema [7].

Prema ovoj strukturi mogu se izdvojiti tri ključna dijela: *models*, *views*, *controllers* tj. takozvani MVC (engl. *Model-View-Controller*) obrazac softverske strukture čiji je način rada predstavljen

³ *Callback* funkcija omogućuje da se funkcija proslijedi kao parametar te tako omogućuje da se može pozvati prema potrebi [6].

na slici 3.3.



Slika 3.3. MVC struktura, prema [8]

Prema slici iznad vidi se kako upravljači (engl. *controllers*) služe kako bi povezali modele (engl. *models*) i poglede (engl. *view*) te upravljali korisničkim zahtjevima. Modeli predstavljaju podatke za interakciju s bazom podataka, dok pogledi predstavljaju prikaz prethodno modeliranih podataka [8]. Ova navedena svojstva poput MVC strukture, robusnosti API-a koji olakšava upravljanje rutama, izuzetno brzog I/O (engl. *Input/Output*), asinkronosti te upotrebe jedne niti, predstavljaju neke od glavnih razloga zašto ExpressJS ima veliku prednost među ostalim okruženjima za razvoj web aplikacija.

3.3. MongoDB

MongoDB je *open-source* dokumentno orijentirana baza podataka. Podatci u MongoDB bazi zapisani su u dokumentima koji predstavljaju strukturu podataka koja sadrži parove polja i vrijednosti. Skupina MongoDB dokumenta čini kolekciju (engl. *collection*). Kolekcija u MongoDB-u ekvivalenta je pojmu tablice u relacijskim bazama podataka, prema [9]. Dokumentno orijentirane baze podataka pripadaju najpopularnijoj vrsti NoSQL baza podataka. U odnosu na relacijske baze podataka ove baze podataka imaju dinamičke sheme tj. svaki dokument u kolekciji može imati različitu strukturu. Ovakav tip baze podataka ima hijerarhijsku strukturu gdje jedan dokument može biti dio drugog, npr. slika 3.4. koja predstavlja složeniju strukturu jednog dokumenta, prema [10].

```

{
  "_id": {
    "$oid": "5f6cf1f38b4217774061f6bc"
  },
  "text": "admin komentira",
  "author": {
    "id": {
      "$oid": "5f6cc142f0cae15960b1583d"
    },
    "firstname": "admin"
  },
  "createdAt": {
    "$date": "2020-09-24T19:22:27.447Z"
  },
  "__v": 0
}

```

Slika 3.4. *Primjer strukture dokumenta*

Postoji normalizirani i denormalizirani način povezivanja dokumenata, prema [11]. Način rada normaliziranog povezivanja prikazan je na slici 3.5.

```

_id: ObjectId("5f5e1160fa11773b7c973cf2")
> comments: Array
name: "mercedes benz s"
image: "https://res.cloudinary.com/uniquecloudn
price: 893000
contact: 923566789
description: "Dostupna verzija samo sa 4.7 litar
user_id: ObjectId("5d8711f061fd81f7042f37ab")
createdAt: 2020-09-13T12:32:32.409+00:00
__v: 5
status: "prodaja"
color: "siva"
location: "Osijek"
engine: "diesel"
km: 160000
yearmanufacture: "2017"
carbrand: "mercedes"

```

Slika 3.5. *Normalizirani način povezivanja dokumenata*

Normalizirani način povezivanja dokumenata može se gledati kao povezivanje s vanjskim ključem u relacijskim bazama. Na primjeru iz slike 3.5. vidi se kako je dokument *car* povezan s *user* dokumentom preko polja *user_id*. Denormalizirani način povezivanja prikazan je na slici 3.6. koja prikazuje kako unutar *car* dokumenta postoje veza na *user* dokumente (objekt *owner*). Razlika između ova dva pristupa je u tome što je normalizirani pristup fleksibilniji, ali izvršava više upita, dok denormalizirani pristup izvršava samo jedan upit prilikom dohvaćanja podataka, ali može utjecati na performanse. Normalizirani način bolje je koristiti prilikom veza više-više, dok denormalizirani kod veza jedan-više, prema [11].

```

    _id: ObjectId("5f5e1160fa11773b7c973cf2")
  > comments: Array
    name: "mercedes benz s"
    image: "https://res.cloudinary.com/uniqueclou
    price: 893000
    contact: 923566789
    description: "Dostupna verzija samo sa 4.7 li
  < owner: Object
    id: ObjectId("5d8711f061fd81f7042f37ab")
    firstname: "dragana"
    createdAt: 2020-09-13T12:32:32.409+00:00
    __v: 5
    status: "prodaja"
    color: "siva"
    location: "Osijek"
    engine: "diesel"
    km: 160000
    yearmanufacture: "2017"
    carbrand: "mercedes"

```

Slika 3.6. Denormalizirani način povezivanja dokumenata

Baza podatka zasnovana na dokumentima omogućuje mnoge prednosti kao što su:

- podržava indeksiranje koje služe za brzu obradu upita,
- podržava replikaciju (čuvanje podataka),
- podržava particioniranje koje omogućuje veću efikasnost (podatci su raspodijeljeni na više različitih servera),
- podržava agregaciju za transformiranje dokumenta pomoću filtriranja i grupiranja,
- podržava specijalne tipove kolekcija i pruža spremište za datoteke,
- podržava API za postavljanje upita i mijenjanje podataka, prema [10].

JavaScript se koristi za rad s MongoDB bazom podataka. Primjer upita kako bi dobili sve automobile koji imaju cijenu manju od tražene prikazan je na slici 3.7.

```

Cars.find( {$and: [
  {price: {$lte : req.query.pricee }}]

```

Slika 3.7. Primjer upit na bazu podataka

MongoDB baza podataka pohranjuje podatke u BSON formatu (engl. *Binary JSON*), koji je vrlo sličan JSON formatu. BSON je izumljen kako bi se popravili nedostaci JSON-a, optimiziran je, fleksibilniji i sadrži više vrsta podataka [12]. Svaki dokument u MongoDB-u sadrži *_id* 12-bajtni niz znamenki koji predstavlja jedinstveni ključ. Zbog navedenih svojstava MongoDB baza podataka je jednostavna, fleksibilna i lako dostupna za korištenje, prema [13].

3.4. Cloudinary NodeJS

Cloudinary NodeJS SDK (engl. *Software Development KIT*) skup je računalnih alata koji omogućuju jednostavniji prijenos i transformaciju slika i videozapisa na oblak [14]. Kako bi koristili Cloudinary NodeJS potrebno je instalirati paket preko naredbe: `npm install cloudinary`. Uz instalaciju paketa potrebno je napraviti račun na <https://cloudinary.com/>, gdje se trebaju konfigurirati parametri `cloud_name`, `api_key` i `api_secret`. Postavljanje ovih parametara može se odraditi programski u pozivu Cloudinary metode ili globalno upotrebom metode `config` npr. slika 3.8.

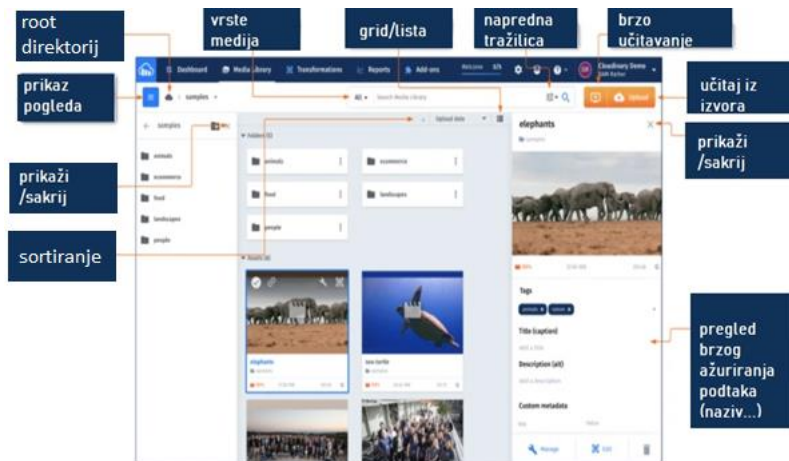
```
32 cloudinary.config({
33   cloud_name: 'xxx',
34   api_key: 'xxx',
35   api_secret: 'xxx'
36 });
```

Slika 3.8. Globalno definiranje Cloudinary parametara, [14]

Cloudinary arhitektura izgrađena je kako bi podržala veliko opterećenje i rukovala neograničenom količinom resursa. Mreža za dostavu sadržaja isporučuje podatke brzo i učinkovito. Prijenos slika izvršava se izravno iz preglednika uz pomoć jQuery dodatka. Cloudinary je sigurno rješenje zasnovano na oblaku. Neke od sigurnosnih značajka koje posjeduje su automatsko sigurno kopiranje resursa na sekundarno zaštićeno mjesto, kompletna kontrola nad pristupom resursima, ograničen pristup resursima osnovan na određenim transformacijama, kontrola pristupa više korisnika, mogućnost dvofaktorske provjere autentičnosti (2FA⁴), prema [15].

Cloudinary sučelje pruža sveobuhvatno upravljanje svim digitalnim resursima s vlastitog računa, a prikazano je na slici 3.9.

⁴ 2FA (eng. *Two Factor Authentication*) provjera autentifikacije korisnika koja uz korisničko ime i lozinku traži i treću dodatnu informaciju za pristup podatcima.



Slika 3.9. Cloudinary sučelje, [16]

3.5. EJS

EJS (engl. *Embedded JavaScript*) jednostavni je jezik za predloške koji omogućuje generiranje HTML-a s jednostavnim JavaScript kodom. Svojstva EJS su: brza kompilacija i prikazivanje, jednostavnost predložaka, statičko predmemoriranje predložaka, udovoljava sustavu ExpressJS aplikacije. Instalacija ovog paketa u projektu radi se preko naredbe *npm install ejs* [17]. Izgled EJS predložaka može se definirati uključivanjem zaglavlja (engl. *header*) i podnožja (engl. *footer*), a primjer strukture jedne ejs datoteke prikazuje slika 3.10.

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>AutoProdaja</title>
5     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
6     <link rel="stylesheet" href="/stylesheets/app.css">
7   </head>
8   <body>
9
10  </body>
11 </html>

```

Slika 3.10. Primjer strukture EJS datoteke, [17]

3.6. CSS

CSS (engl. *Cascading Style Sheets*) stilski je jezik koji se koristi za opis izgleda HTML ili XML jezika. Korisničko sučelje većine web stranica napravljeno je upravo upotrebom CSS zajedno s HTML-om i JavaScript-om. Ovaj stilski jezik pruža mogućnost jednostavne i brze izmjene

izgleda web stranice uz pomoć oznaka npr. za font, boju, veličinu, poravnavanje. CSS sastoji se od dva dijela: selektora i deklaracijskog dijela, slika 3.11. Selektor označava na kojem HTML elementu se želi primijeniti stil, a deklaracijski blok sadrži dio koji deklarira naziv svojstva (engl. *property*), te vrijednosti koje se pridodjeljuju navedenom svojstvu, prema [18].



Slika 3.11. Prikaz dijelova CSS

3.7. Bootstrap

Bootstrap je popularno HTML, CSS, JavaScript okruženje za jednostavnije i brže razvijanje responzivnih⁵ web i mobilnih aplikacija. Uključuje CSS i HTML predloške za dizajn obrazaca, tablica, navigacije itd. Uz razne komponente Bootstrap također sadrži jQuery dodatke. Kako bi margine stranice bile ispravno postavljene potrebno je koristiti sadržaj u *div* HTML elementu s klasom *container*, a ako se želi sadržaj raspodijeliti unutar stupaca, tada se koristi klasa *rows*. Bootstrap komponente mogu se lako prilagoditi željenom izgledu stranice. Kompatibilan je za većinu preglednika npr. Chrome, Firefox, Safari, Opera, Internet Explorer, prema [19].

3.8. ZingChart

ZingChart JavaScript je komercijalni alat za prikazivanje grafova i interaktivnih karti, no dostupan je za besplatno isprobavanje, prema [20]. Instalacija ovog alata je jednostavna, može se instalirati preko *npm* naredbe ili izravno uključiti preko linka. Ovaj alat pruža različite vrste grafova i mogućnosti prikazivanja podataka. ZingChart pruža vlastite predloške za prikaz podataka, no sami predlošci mogu se mijenjati prema potrebama aplikacije. Glavna metoda za prikazivanje grafova je *zingchart.render()* (slika 3.12.) u kojoj se definiraju različite opcije grafa. Osnovni podatci koji se moraju navesti u ovoj metodi su id elementa grafa (engl. *id*) i podatci (engl. *data*), prema [20].

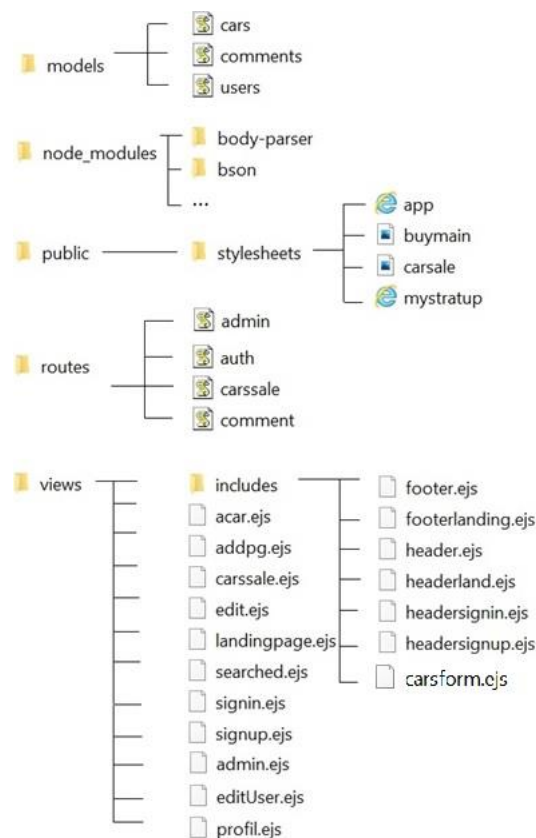
⁵ Responzivna web stranica je stranica koja se automatski može prilagoditi svim uređajima.

```
zingchart.render({  
  id:"myChart",  
  width:"100%",  
  height:400,  
  data:={}  
  
});  
};
```

Slika 3.12. Zingchart metoda prikazivanja grafova

4. PRIMJENA OPISANIH TEHNOLOGIJA U IZRADI WEB APLIKACIJE

Naredno poglavlje rada detaljnije će obraditi način na koji su primijenjene tehnologije koje su opisane u prethodnom poglavlju u izradi web aplikacije. Ova web aplikacija predstavlja primjer aplikacije za objavu oglasa, a služi za objavljivanje oglasa o prodaji automobila i nazvana je „AutoProdaja“. Na slici 4.1. prikazana je struktura napravljenog projekta.



Slika 4.1. Struktura projekta „AutoProdaja“

4.1. Registracija i prijava korisnika

Ovoj aplikaciji mogu pristupiti anonimni i prijavljeni korisnici. Anonimni korisnici imaju samo mogućnost pregledavanja oglasa i upotrebe tražilice, dok za sve ostale funkcionalnosti potrebno je obaviti prijavu korisnika. Predložak za registraciju korisnika prikazan je na slici 4.2. prema kojoj se vidi da za uspješnu registraciju korisnika potrebno je unijeti podatke kao što su: ime, prezime, email, korisničko ime i lozinka. Na ovoj slici u sedmom redu nalazi se HTTP POST metoda koja služi za kreiranje novih podataka. Na slici 4.3. prikazana je kako ova POST metoda

uzima upisane podatke i prosljeđuje ih u *users* kolekciju. Za provjeru identiteta instaliran je paket *passport*, a na slici 4.3. retku 36 nalazi se *authenticate()* funkcija koja služi za provjeru autentičnosti zahtjeva. Prije nego se pozove funkcija za provjeru autentičnosti, potrebno je odrediti strategiju koju koristi aplikacija, te prema slici 4.4. konfigurirati *passport.initialize()*. *Passport* pruža funkciju serijalizacije i deserijalizacije korisnika, pridodjeljujući mu jedinstveni ID i omogućuje pronalaženje korisnika po tom ID-ju, prema [21].

```
1 <% include includes/headersignup.ejs %>
2
3 <div class="container" id="signup">
4   <div class="row">
5     <h1 style="text-align:center;">Registriraj se!</h1>
6     <div style="width: 30%; margin:20px auto;">
7       <form action="/signup" method="POST">
8         <div class="form-group">
9           <label>Ime</label>
10          <input class = "form-control" type="text" name="firstname" placeholder="ime">
11        </div>
12
13        <div class="form-group">
14          <label>Prezime</label>
15          <input type="text" class="form-control" name="lastname" placeholder="prezime">
16        </div>
17
18        <div class="form-group">
19          <label>Email</label>
20          <input type="email" class="form-control" name="email" placeholder="email">
21        </div>
22
23        <div class="form-group">
24          <label>Korisničko ime</label>
25          <input class = "form-control" type="text" name="username" placeholder="korisničko ime">
26        </div>
27
28        <div class="form-group">
29          <label>Lozinka</label>
30          <input type="password" class="form-control" name="password" id="exampleInputPassword1" placeholder="lozinka">
31        </div>
32
33        <div class="form-group">
34          <button class="btn btn-lg btn-default btn-block">Registriracija!</button>
35        </div>
36      </form>
37      <p>Prijavite se <a href="/signin">Log In</a></p>
38    </div>
39  </div>
40 </div>
41
42 <% include includes/footer.ejs %>
```

Slika 4.2. Predložak za registraciju korisnika, *signup.ejs*

```

18 router.get("/signout",function(req,res){
19   req.flash("success","Uspješno ste odjavljeni!");
20   req.logout();
21   res.redirect("/");
22 });
23
24 router.post("/signup",function(req,res){
25   users.register(new users(
26     {
27       firstname:req.body.firstname,
28       lastname:req.body.lastname,
29       email:req.body.email,
30       username:req.body.username
31     }),req.body.password,function(err,user){
32     if(err){
33       req.flash("error","Greška prilikom registracije, molim probajte ponovno.");
34       return res.render("signup");
35     }
36     passport.authenticate("local")(req,res, function(){
37       req.flash("success","Dobrodošli "+user.firstname+" na stranicu AutoProdaja!");
38       res.redirect("/carsale");
39     });
40   });
41 });

```

Slika 4.3. Metode za registraciju korisnika

```

31 app.use(passport.initialize());
32 app.use(passport.session());
33 app.use(methodOverride("_method"));
34 app.use(flash());
35 app.locals.moment = require('moment');
36
37 passport.use(new LocalStrategy(users.authenticate()));
38 passport.serializeUser(users.serializeUser());
39 passport.deserializeUser(users.deserializeUser());
40

```

Slika 4.4. Passport

Users kolekcija predstavlja jedan od modela ovog projekta. Modeli se definiraju pomoću sheme koja omogućuje definiranje pojedinačnih polja unutar svakog dokumenta [22] (slika 4.5). Primjer MongoDB dokumenata za *users* kolekciju prikazan je na slici 4.6.

```

1 var mongoose = require("mongoose"),
2   passportLocalMongoose = require("passport-local-mongoose");
3
4 var usersSchema = new mongoose.Schema({
5   firstname:String,
6   lastname:String,
7   email:String,
8   username:String,
9   password:String,
10  role:{ type:String, default:"basic"},
11 });
12
13 usersSchema.plugin(passportLocalMongoose);
14 module.exports = mongoose.model("users",usersSchema);

```

Slika 4.5. Shema users modela

```

{
  "_id": {
    "$oid": "5f6d933bf4ea2ca01c87df27"
  },
  "role": "basic",
  "firstname": "primjer",
  "lastname": "primjer",
  "email": "primjer@gmail.com",
  "username": "primjer",
  "salt": "cf2b22822234932e678b0b7d8d9f4fbf82ea0a643a2a497cdc1595dece5893e1",
  "hash": "931920514699e1454d85a03eb49254312b20f813c2f459823137898d6ee1f0be9983788183b10d40791671eb4921",
  "__v": 0
}

{
  "_id": {
    "$oid": "5f728d8fc19edf6300b42247"
  },
  "role": "basic",
  "firstname": "ivica",
  "lastname": "ivica",
  "email": "ivica@example.com",
  "username": "ivica",
  "salt": "c367075e0f29b22e3a49f730f5f987f0a220776be99ce66b5692a6e8e0a3129f",
  "hash": "a0391fc4c5a51a9d3b5af7caefb12c7c85abf83040007cb25ee446e41279c2d663625be137b85c393240278119d4",
  "__v": 0
}

```

Slika 4.6. *Primjer prikaza korisničkih podataka u MongoDB*

Korisnik pri registraciji dobije zadanu vrijednost za ulogu, a to je „*basic*“ tj. obični korisnik. Nakon što je korisnik registriran, može se prijaviti preko „*SignIn*“ opcije i početi kreirati nove objave i pisati komentare pod objavama. Tijekom prijave postoji provjera autentičnosti korisnika s *authenticate* funkcijom, nakon koje se korisnik preusmjeri dalje ovisno je li uspješno prošla autentifikacija, ako nije uspješno prošla pojavi se obavijest da je došlo do greške.

```

43 router.post("/signin",passport.authenticate( "local", {
44     successRedirect:"/carssale",failureRedirect:"/signin",failureFlash: true
45   } ) ,function(req,res){
46
47   });

```

Slika 4.7. *Metoda za prijavu korisnika*

4.2 Kreiranje objave, brisanje i izmjena oglasa i komentara

Nakon što korisnik obavi prijavu dobiva pristup kreiranju novih objava. Predložak za kreiranje novih objava prikazan je na slici 4.8. Za kreiranje nove objave potrebno je upisati informacije poput naziva, cijene automobila, kontakt telefona, vrste motora, boje, marke, modela, prijeđenih kilometara automobila, godine proizvodnje, kratkog opisa oglasa te slike. MongoDB omogućuje validaciju prilikom kreiranja nove objave, a neke od ugrađenih validacijskih opcija koje su korišteni u predlošku za kreiranje nove objave su: *min,max* vrijednosti za brojčano polje nazvano „*carcontact*“, *minlength* za tekstualno polje „*descriptionform*“ koje ograničava da upisana vrijednost u polju za opis mora imati najmanje 50 znakova. Metoda koja se koristi za dodavanje

nove objave je POST metoda, slika 4.9.

```
1 <% include includes/header.ejs %>
2 <div class="container">
3   <div class="row">
4     <div style="text-align:center;">Dodaj oglas auta!</div>
5     <div style="width: 30%; margin:20px auto;">
6       <form action="/carsale" method="POST" id="add" enctype="multipart/form-data">
7         <% include includes/carsform.ejs %>
8         <p>Snaga motora (u kW): </p> <input class="form-control" type="Number" name="enginepower" placeholder="snaga motora (u kW)" max=1000 min=0>
9         <p>Maksimalna CO2 emisija (g/km):</p> <input class="form-control" type="Number" name="cod" placeholder="maksimalna CO2 emisija (g/km)" max=1000 min=0>
10      </div>
11      <button class="btn btn-lg btn-default btn-block">Objavi!</button>
12    </div>
13  </form>
14  <a href="/carsale">Vrati se nazad</a>
15 </div>
16 </div>
17 </div>
18 <% include includes/footer.ejs %>
```

Slika 4.8. Predložak za kreiranje nove objave

```
142 router.post("/carsale", isLoggedIn, upload.single('carimage'), function(req, res){
143   cloudinary.uploader.upload(req.file.path, function(result) {
144     var name = req.body.carname;
145     var price = req.body.carprice;
146     var image = result.secure_url;
147     var contact = req.body.carcontact;
148     var description = req.body.descriptionform;
149     var engine = req.body.engine;
150     var km = req.body.km;
151     var yearmanufacture = req.body.yearmanufacture;
152     var carbrand = req.body.carbrand;
153     var location = req.body.location;
154     var color = req.body.color;
155     var model = req.body.model;
156     var typeofdrive = req.body.typeofdrive;
157     var gas = req.body.gas;
158     var airconditioning = req.body.airconditioning;
159     var gearshift = req.body.gearshift;
160     var numberOfgears = req.body.numberofgears;
161     var numberofdoors = req.body.numberofdoors;
162     var numberofseats = req.body.numberofseats;
163     var ecocategory = req.body.ecocategory;
164     var numberofowner = req.body.numberofowner;
165     var audio = req.body.audio;
166     var airbags = req.body.airbags;
167     var payment = req.body.payment;
168     var enginepower = req.body.enginepower;
169     var co2 = req.body.co2;
170     var owner = {
171       id: req.user._id,
172       firstname: req.user.firstname
173     }
174     var newpg = {
175       name: name,
176       image: image,
177       price: price,
178       contact: contact,
179       description: description,
180       engine: engine,
181       km: km,
182       yearmanufacture: yearmanufacture,
183       carbrand: carbrand,
184       location: location,
185       color: color,
186       model: model,
187       owner: owner,
188       typeofdrive: typeofdrive,
189       gas: gas,
190       airconditioning: airconditioning,
191       gearshift: gearshift,
192       numberOfgears: numberOfgears,
193       numberofdoors: numberofdoors,
194       numberofseats: numberofseats,
195       ecocategory: ecocategory,
196       numberofowner: numberofowner,
197       audio: audio,
198       airbags: airbags,
199       payment: payment,
200       enginepower: enginepower,
201       co2: co2,
202     }
203
204     Cars.create(newpg, function(err, car) {
205       if (err) {
206         console.log("ERROR");
207         return res.redirect('addpg');
208       }
209       req.flash("success", "Uspješno ste dodali auto!");
210       res.redirect('/carsale/');
211     });
212   });
213 });
```

Slika 4.9. POST metoda za dodavanje novog oglasa

Prema slici 4.9. prije nego što je moguće objaviti novi oglas provjerava se je li korisnik prijavljen, što radi funkcija *isLoggedIn* (slika 4.10.) metodom *isAuthenticated()*.

```
355 function isLoggedIn(req,res,next){
356   if(req.isAuthenticated()){
357     return next();
358   }
359   req.flash("error","Morate biti prijavljeni!");
360   res.redirect("/signin");
361 }
```

Slika 4.10. *Provjera autentičnosti korisnika*

Za kreiranje novog oglasa potrebno je učitati jednu sliku formata *jpeg* ili *png*. Učitana slika sprema se u oblak uz pomoć Cloudinary rješenja, slika 4.11. Za učitavanje slike koristi se posrednički (engl. *middleware*) softver za obradu podataka *multer*. Kad se šalju datoteke preko poslužitelja, zahtjev za *req.body* je prazan, a upotrebom *multer*-a stvara se objekt *req.file* koji omogućuje učitavanje datoteke. U ovom slučaju korištena je *multer* [23] *upload.single(fieldname)* metoda uz pomoću koje se učitava jedna slika. Učitane slike spremaju se na oblak. Prema slici 4.9. koristi se *Cloudinary.uploader* metoda koja sprema učitane datoteke. Kako bi se uspostavila konekcija s oblakom, bilo je potrebno konfigurirati cloudinary parametre (slika 4.11.).

```
11 var storage = multer.diskStorage({
12   filename: function(req, file, callback) {
13     callback(null, Date.now() + file.originalname);
14   }
15 });
16 var imageFilter = function (req, file, cb) {
17   if (!file.originalname.match(/\.(jpg|jpeg|png|gif)$/i)) {
18     return cb(new Error('Učitaj sliku!'), false);
19   }
20   cb(null, true);
21 };
22 var upload = multer({ storage: storage, fileFilter: imageFilter})
23
24 cloudinary.config({
25   cloud_name: 'uniquecloudname',
26   api_key: 'xxx',
27   api_secret: 'xxx'
28 });
```

Slika 4.11. *Učitavanje slike na cloud*

Prilikom kreiranja novog oglasa upisane vrijednosti su mapirane s nazivima polja za *cars* model, a samom metodom *create* kreira se novi oglas. Slika 4.12. predstavlja *cars* model podataka, prema kojoj vidimo kako je *cars* kolekcija povezana s *users* kolekcijom (veza jedan na više, više oglasa može pripadati jednom korisniku, te više komentara može pripadati jednom oglasu).

```

1 var mongoose = require("mongoose");
2
3 var carsSchema = new mongoose.Schema({
4   name: String,
5   image: String,
6   status: { type:String, default:"prodaja"},
7   color: String,
8   description:String,
9   price:Number,
10  location:String,
11  contact:Number,
12  engine:String,
13  model:String,
14  km:Number,
15  yearmanufacture: String,
16  carbrand:String,
17  typeofdrive:String,
18  gas:String,
19  airconditioning:String,
20  gearshift:String,
21  numberofgears:String,
22  numberofdoors:String,
23  numberofseats:String,
24  ecocategory:String,
25  numberofowner:String,
26  audio:String,
27  airbags:String,
28  payment:String,
29  enginepower:Number,
30  cod:Number,
31  owner:{
32    id:{ type: mongoose.Schema.Types.ObjectId,
33      ref:"users"
34    },
35    firstname:String
36  },
37  createdAt: { type: Date, default: Date.now },
38  comments:[{
39    type: mongoose.Schema.Types.ObjectId,
40    ref: "Comments"
41  }]
42 }
43 );
44
45 var Cars = mongoose.model("Cars",carsSchema);
46 module.exports = Cars;

```

Slika 4.12. *Shema modela cars*

Osim kreiranja novi objava prijavljeni korisnici imaju i mogućnost pisanja komentara pod objavljenim oglasom. Shema *comments* prikaza je na slici 4.13. gdje je jedan komentar povezan s jednim korisnikom i objavom, a sam korisnik može imati više komentara. Metoda koja se koristi za kreiranje komentara također je POST metoda, a prikaza na je na slici 4.14.

```

1  var mongoose = require("mongoose");
2
3  var commentsSchema = new mongoose.Schema({
4    text:String,
5    author:{
6      id:{ type: mongoose.Schema.Types.ObjectId,
7        ref:"users"
8      },
9      firstname:String
10   },
11   createdAt: { type: Date, default: Date.now }
12 });
13 });
14
15 var Comments = mongoose.model("Comments",commentsSchema);
16 module.exports = Comments;

```

Slika 4.13. *Shema modela za komentare*

```

>
6  router.post("/carsale/:id/comment",ownership,function(req,res){
7    var commentcomp = {
8      text:req.body.text,
9      author:{
10       id:req.user._id,
11       firstname:req.user.firstname
12     }
13   }
14   Comments.create(commentcomp,function(err,comment){
15     if(err){
16       console.log("error!");
17     }
18     else{
19       Cars.findOne({_id:req.params.id},function(err,car){
20
21         if(err){
22           console.log("error!!");
23         }
24         else{
25           car.comments.push(comment);
26           car.save(function(err,data){
27             if(err){
28               console.log("error!");
29             }
30             else{
31               Cars.findById(req.params.id).populate("comments").exec(function(err,detailcar){
32                 if(err){
33                   console.log("ERROR: ");
34                   console.log(err);
35                 }
36                 else{
37                   req.flash("success","Uspjesno dodan komentar!");
38                   res.redirect("/carsale/"+detailcar._id);
39                 }
40               });
41             }
42           });
43         }
44       });
45     }
46   });
47 });

```

Slika 4.14. *Metoda za kreiranje komentara*

```

283 function ownership(req,res,next){
284   if(req.isAuthenticated()){
285     cars.findById(req.params.id,function(err,car){
286       if(err){
287         req.flash("error","Nije pronadeno!");
288         res.redirect("back");
289       }
290       else{
291         if((car.owner.id.equals(req.user._id)) || (req.user.role='admin')) {
292           next();
293         }
294         else{
295
296           req.flash("error","Nemate prava!");
297           res.redirect("back");
298         }
299       }
300     });
301   }
302   else{
303     req.flash("error","Prijavite se!");
304     res.redirect("back");
305   }
306 }

```

Slika 4.15. Funkcija *ownership*

Prilikom kreiranja komentara provjerava se nad kojim objektom se kreira komentar, te također pridodjeljuje se vrijednost tko je vlasnik kreiranog komentara. Funkcija *ownership* provjerava ima li trenutni korisnik ima prava, tj. je li trenutni korisnik vlasnik oglasa ili je admin. Kako bi mogli pregledati navedene oglase i komentare, upotrebljava se HTTP GET metoda, slika 4.16. Prema slici 4.16. u retku 220 nalazi se pomoćna metoda *findbyId* koja pronalazi podatke za željeni objekt uz pomoću njegovog jedinstvenog id-a. Na početnoj strani za oglase prikazani su samo automobili koji su još uvijek u prodaji (slika 4.17.), a automobile koje su prodani mogu vidjeti samo vlasnici tih objava na osobnom profilu.

```

219 router.get("/carssale/:id",function(req,res){
220   cars.findById(req.params.id).populate("comments").exec(function(err,detailcar){
221     if(err){
222       console.log("ERROR: ");
223       console.log(err);
224     }
225     else{
226       res.render("acar",{carinfo:detailcar});
227     }
228   });
229 });
230

```

Slika 4.16. Primjer GET metode za dohvaćanje pojedinačnog oglasa


```

30 <h4 class="pull-left">Automobili na prodaji:</h4>
31 <div class="row text-right" style=" flex-wrap:wrap;
32 float: left;
33 width: 100%;
34 height: 100%;
35 object-fit: cover;
36 ">
37 <% pginfo.forEach(function(item){ %>
38 <% if(
39 (item.status=="prodaja")
40 ) { %>
41 <div class="col-md-3 col-sm-6">
42 <div class="thumbnail">
43 <a href="/carssale/<%= item._id %>">
44 
45 </a>
46 <div class="caption">
47 <h4> <%= item.name %> </h4>
48 </div>
49 <div class="caption" style="display: none;">
50 <h4> <%= item.description %> </h4>
51 </div>
52 </div>
53 </div>
54 <% } %>
55 <% }); %>
56 </div>

```

Slika 4.17. Prikaz automobila koji su u prodaji

Za čitanje oglasa i komentara nije potrebno biti registriran. Dok za pregledavanje oglasa nije potrebno biti prijavljen, za uređivanje i brisanje podatka potrebno je prethodno obaviti prijavu. Na slici 4.18. i 4.19. nalaze se DELETE i PUT metode, PUT metoda je vrlo slična POST metodi, no prilikom uređivanja podataka provjerava se je li korisnik vlasnik tih podataka.

```

332 router.delete("/carssale/:id",ownership,function(req,res){
333   Cars.findByIdAndRemove(req.params.id, function(err,car){
334     if(err){
335       res.redirect("/carssale")
336     }
337     else {
338       Comments.deleteMany({_id:{$in:car.comments}}, function(err){
339         if (err) {
340           console.log(err);
341         }
342         else{
343           req.flash("success","Uspješno ste izbrisali ovu objavu!");
344           res.redirect("/carssale")
345         }
346       });
347     }
348   });
349 });
350 });
351

```

Slika 4.18. DELETE metoda za brisanje objave

```

243 router.put("/carsale/:id",ownership,upload.single('carimage'),function(req,res){
244   const carimage = req.file;
245   if (!carimage) {
246     var data = {
247       name:req.body.carname,
248       price:req.body.carprice,
249       contact:req.body.carcontact,
250       status:req.body.status,
251       color:req.body.color,
252       engine:req.body.engine,
253       km:req.body.km,
254       yearmanufacture:req.body.yearmanufacture,
255       carbrand:req.body.carbrand,
256       location:req.body.location,
257       image:"https://res.cloudinary.com/uniquecloudname/image/upload/v1608866328/
258       description:req.body.descriptionform,
259       model:req.body.model,
260       typeofdrive:req.body.typeofdrive,
261       gas:req.body.gas,
262       airconditioning:req.body.airconditioning,
263       gearshift:req.body.gearshift,
264       numberofgears:req.body.umberofgears,
265       numberofdoors:req.body.numberofdoors,
266       numberofseats:req.body.numberofseats,
267       ecocategory:req.body.ecocategory,
268       numberofowner:req.body.numberofowner,
269       audio:req.body.audio,
270       airbags:req.body.airbags,
271       payment:req.body.payment,
272       enginepower:req.body.enginepower,
273       co2:req.body.co2,
274     }
275     Cars.findOneAndUpdate({_id:req.params.id},data,function(err,updated)
276     {
277       if(err){
278         res.redirect("/carsale");
279       }
280       else{
281         req.flash("success","Uspješno ste uredili podatke!");
282         res.redirect("/carsale/"+req.params.id);
283       }
284     });

```

Slika 4.19. PUT metoda za osvježavanje objave

Uz brisanje objava dodana je i opcija brisanja kreiranih komentara. Opcija za brisanje komentara je ponuđena samo vlasnicima tog objekta i adminu, slika 4.20.

```

58 router.delete("/carsale/:id/comment/:comment_id",ownership,function(req,res){
59   var data = {
60     text:req.body.text
61   }
62   Comments.findByIdAndRemove({_id:req.params.comment_id},data,function(err,updated){
63     if(err){
64       res.redirect("/carsale/"+req.params.id+"/comment/"+req.params.comment_id);
65     }else{
66       req.flash("success","Uspješno obrisani komentar!");
67       res.redirect("/carsale/"+req.params.id);
68     }
69   });
70 });

```

Slika 4.20. Kreiranje opcije brisanja komentara

U ovom dijelu opisane su CRUD (engl. *Create,Read,Update,Delete*) operacije nad podacima. Na slici 4.8. mogu se primijetiti kodovi za definiranje izgleda stranice kao što su širina (engl. *width*), margine (engl. *margin*) itd. Uz definiranje stilova unutar samog *ejs* dokumenta, stilovi su definirani i u posebnoj *css* datoteci *app.css*. Za bolji izgled web aplikacije korišten je i Bootstrap koji omogućuje responzivnu aplikaciju. Bootstrap je uključen u zaglavlje stranice na način prikazan na

slici 4.21.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>AutoProdaja</title>
5     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css" integrity="
6     <link rel="stylesheet" href="/stylesheets/app.css">
7     <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
8     <script src="https://cdn.zingchart.com/zingchart.min.js"></script>
9
10    <script>
```

Slika 4.21. Bootstrap

4.3. Pretraživanje oglasa

Ova aplikacija pruža mogućnost pretraživanja oglasa prema poljima: naziva, maksimalnoj cijeni, maksimalnim prijeđenim kilometrima, lokaciji, godini proizvodnje, boji, marki, modelu automobila, vrsti motora, pogona i mjenjača, prema broju sjedala i vrata, ekološkoj kategoriji automobila, a sva polja po kojima je moguće pretraživanje prikazana su na slici 4.22. Tražilica je napravljena upotrebom *regex* operatora [23] i *find* metode. *Regex* operator u MongoDB bazi podatka se koristi za pretragu nizova slogova unutar polja.

```
45 router.get("/search",function(req,res){
46   try {
47     Cars.find( {$and: [
48       {price: {$lte: req.query.carprice}},
49       {"$or":[
50         { name: null }, { name:'' }, {name:{$regex:req.query.carname, $options: 'i' }}
51       ]},
52       {"$or":[
53         { color: null }, { color:'' }, {color:{$regex:req.query.color }}
54       ]},
55       {"$or":[
56         { location: null }, { location:'' }, {location:{$regex:req.query.location, $options: 'i' }}
57       ]},
58       {"$or":[
59         { yearmanufacture: null }, { yearmanufacture:'' }, {yearmanufacture:{$regex:req.query.yearmanufacture, $options: 'i' }}
60       ]},
61       {"$or":[
62         { carbrand: null }, { carbrand:'' }, {carbrand:{$regex:req.query.carbrand }}
63       ]},
64       {"$or":[
65         { engine: null }, { engine:'' }, {engine:{$regex:req.query.engine }}
66       ]},
67       {"$or":[
68         {km: {$lte: req.query.km }}
69       ]},
70       {"$or":[
71         { model: null }, { model:'' }, {model:{$regex:req.query.model, $options: 'i' }}
72       ]},
73       {"$or":[
74         { typeofdrive: null }, { typeofdrive:'' }, {typeofdrive:{$regex:req.query.typeofdrive, $options: 'i' }}
75       ]},
76       {"$or":[
77         { gas: null }, { gas:'' }, {gas:{$regex:req.query.gas, $options: 'i' }}
78       ]},
79       {"$or":[
80         { airconditioning: null }, { airconditioning:'' }, {airconditioning:{$regex:req.query.airconditioning, $options: 'i' }}
81       ]},
82       {"$or":[
83         { gearshift: null }, { gearshift:'' }, {gearshift:{$regex:req.query.gearshift, $options: 'i' }}
84       ]},
85       {"$or":[
86         { numberofgears: null }, { numberofgears:'' }, {numberofgears:{$regex:req.query.numberofgears, $options: 'i' }}
87       ]},
88       {"$or":[
89         { numberofdoors: null }, { numberofdoors:'' }, {numberofdoors:{$regex:req.query.numberofdoors, $options: 'i' }}
90       ]},
91       {"$or":[
92         { numberofseats: null }, { numberofseats:'' }, {numberofseats:{$regex:req.query.numberofseats, $options: 'i' }}
93       ]},
94       {"$or":[
95         { ecocategory: null }, { ecocategory:'' }, {ecocategory:{$regex:req.query.ecocategory, $options: 'i' }}
96       ]},
97       {"$or":[
98         { numberofowner: null }, { numberofowner:'' }, {numberofowner:{$regex:req.query.numberofowner, $options: 'i' }}
99       ]},
100      {"$or":[
101        { audio: null }, { audio:'' }, {audio:{$regex:req.query.audio, $options: 'i' }}
102      ]},
103      {"$or":[
104        { airbags: null }, { airbags:'' }, {airbags:{$regex:req.query.airbags, $options: 'i' }}
105      ]},
106      {"$or":[
107        { payment: null }, { payment:'' }, {payment:{$regex:req.query.payment, $options: 'i' }}
108      ]},
109      {"$or":[
110        {enginepower : { $gt : req.query.enginepowerMin, $lt : req.query.enginepowerMax}}]
```

Slika 4.22. Tražilica napravljena primjenom *find* metode i *regex* operatora

Struktura *regex* operatora sastoji se od vrijednost polja koja mu je predana i dodatnih mogućnosti (engl. *options*) za pretraživanje kao što je u ovom slučaju „i“ koji predstavlja neosjetljivost na velika i mala slova.

4.4. Grafički prikaz podataka

Za grafički prikaz podataka koristi se alat ZingChart, koji na jednostavni način prikazuje graf pomoću metode *zingchart.render()*. Prema slici 4.23. napravljen je graf koji prikazuje na x-osi id oglasa, na y-osi s lijeve strane su vrijednosti polja cijene i prijeđenih kilometara, a s desne strane y-osi prikazane su godine proizvodnje.

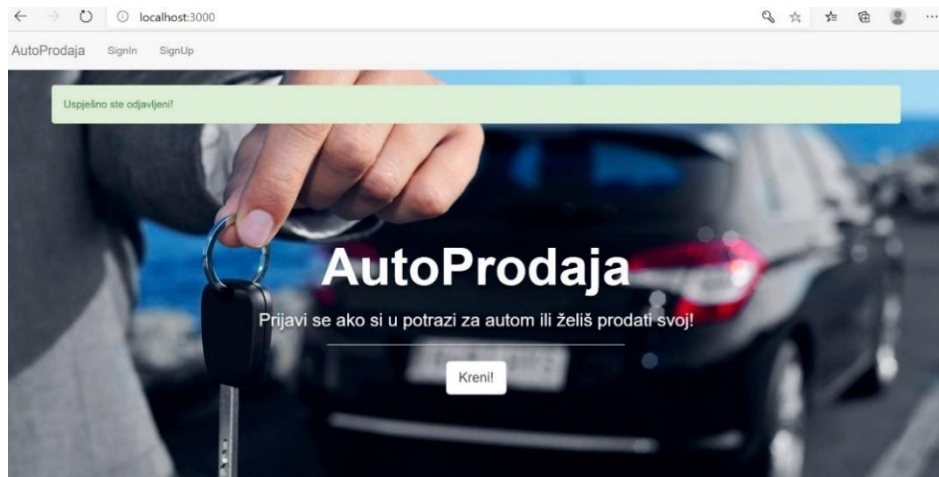
```
59 <% if( (curruser) ){ %>
60     <div class="thumbnail">
61
62         <script>
63         function httpGet(Url){
64             var xmlhttp = new XMLHttpRequest();
65             xmlhttp.open("GET", Url, false);
66             xmlhttp.send(null);
67             return xmlhttp.responseText;
68         }
69
70         window.onload=function(){
71             var aData = JSON.parse(httpGet('http://localhost:3000/test'));
72
73             var seriesData = {
74                 price: [],
75                 km: [],
76                 yearmanufacture: [],
77             };
78
79             for(var n = 0; n < aData.length; n++){
80                 seriesData['price'].push(aData[n]['price']);
81                 seriesData['km'].push(aData[n]['km']);
82                 seriesData['yearmanufacture'].push(aData[n]['yearmanufacture']);
83             }
84
85             zingchart.render({
86                 id:"myChart",
87                 width:"100%",
88                 height:400,
89                 data:{=
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Slika 4.23. Dohvaćanje MongoDB podataka i prikazivanje na grafu

Prema slici iznad podatci iz MongoDB baze za vrijednosti polja „*price*“, „*km*“, „*yearmanufacture*“ dohvaćaju se s pomoću *httpGet* funkcije i spremaju se u polja. Zatim se takvi podatci prosljeđuju *zingchart.render* metodi unutar *data* polja. Unutar *data* polja postoje mnogobrojne definirane značajke koje utječu na način prikazivanja i interakciju grafova.

5. OPIS FUNKCIONALNOSTI I IZGLEDA WEB APLIKACIJE

Sljedeće poglavlje opisat će upotrebu ove web aplikacije uz priložene slike web aplikacije. Na slici 5.1. prikazana je početna stranica aplikacije, gdje su ponuđene opcije u navigacijskoj traci prijave i registracije.

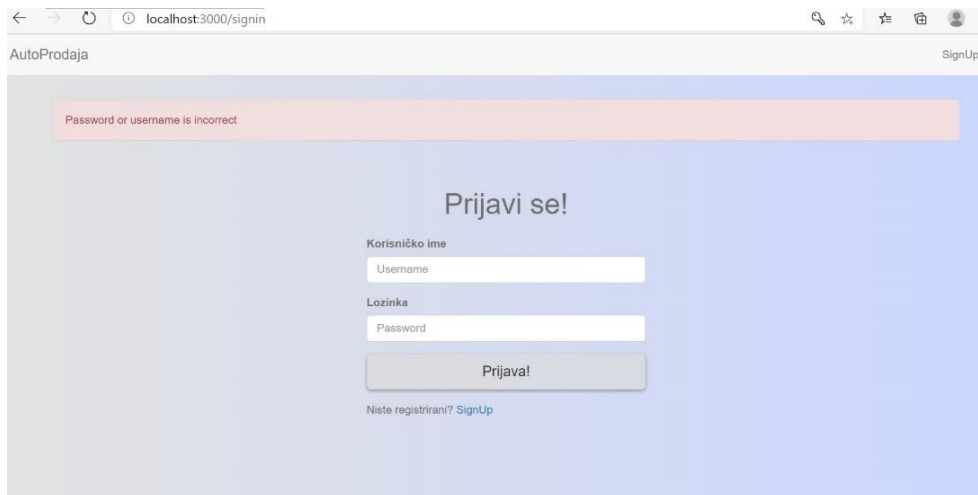


Slika 5.1. Početna stranica web aplikacije „AutoProdaja“

Neregistriran korisnik ne može objaviti oglas, prvo će se morati registrirati upisivanjem potrebnih podataka prema obrascu prikazanom na slici 5.2. Registriran korisnik može se prijaviti odabirom opcije „SignIn“ (slika 5.3.). Uneseni pogrešni podatci za prijavu rezultirat će prikazivanjem odskočnog prozora s porukom kako lozinka ili korisnikovo ime nisu ispravni.

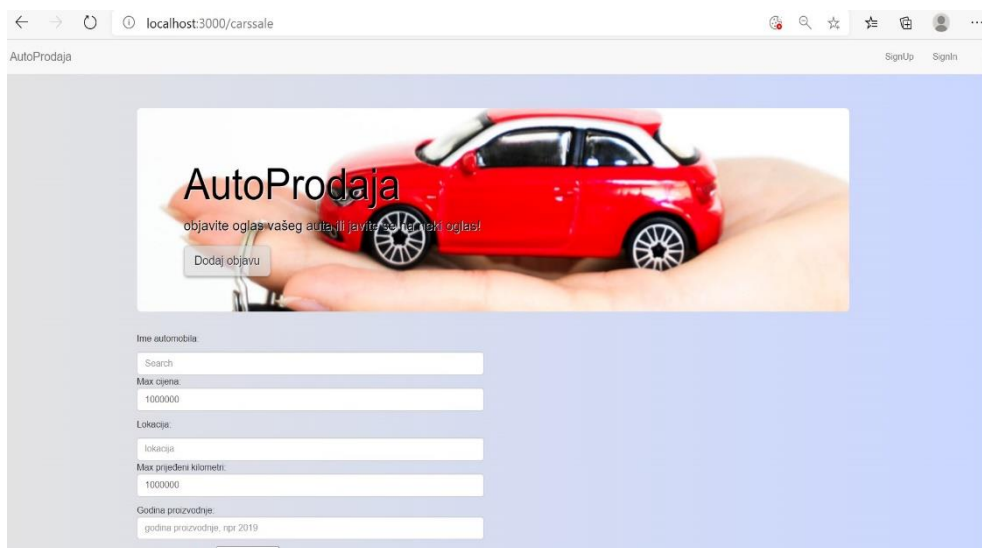
A screenshot of a web browser showing the registration form for 'AutoProdaja'. The browser's address bar shows 'localhost:3000/signup'. The page has a navigation bar with 'AutoProdaja' and 'SignIn' links. The main content area has a light blue background and is titled 'Registriraj se!'. The form contains several input fields: 'Ime' (with 'ime' as a placeholder), 'Prezime' (with 'prezime' as a placeholder), 'Email' (with 'email' as a placeholder), 'Korisničko ime' (with 'korisničko ime' as a placeholder), and 'Lozinka' (with 'lozinka' as a placeholder). Below these fields is a button labeled 'Registriraj!'.

Slika 5.2. Obrazac za registraciju korisnika

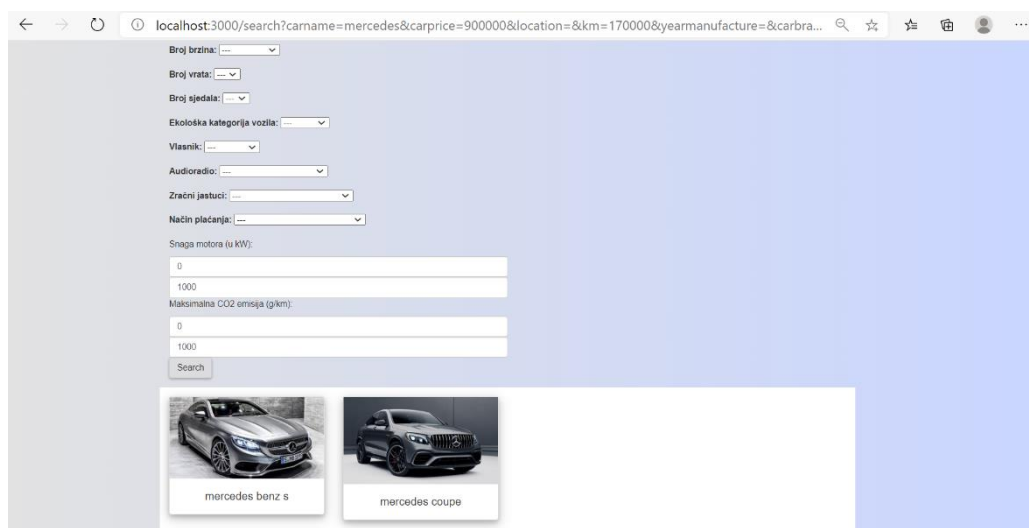


Slika 5.3. *Obrazac za prijavu korisnika*

Odabirom opcije „Kreni!“ prijavljeni i neprijavljeni korisnici preusmjereni su na stranicu s oglasima (slika 5.4.). Korisnici mogu pregledati i pretraživati oglase koji nisu prodani. Slika 5.5. prikazuje pretraživanje oglasa prema riječi „mercedes“. U tražilicu nije potrebno upisati sve vrijednosti po kojima je moguće pretraživanje, za tekstualne vrijednosti uzima se prazan slog tj. sve vrijednosti, a za brojčane vrijednosti uzima se maksimalna vrijednost, a prema ovome principu prikazuju se svi odgovarajući oglasi.

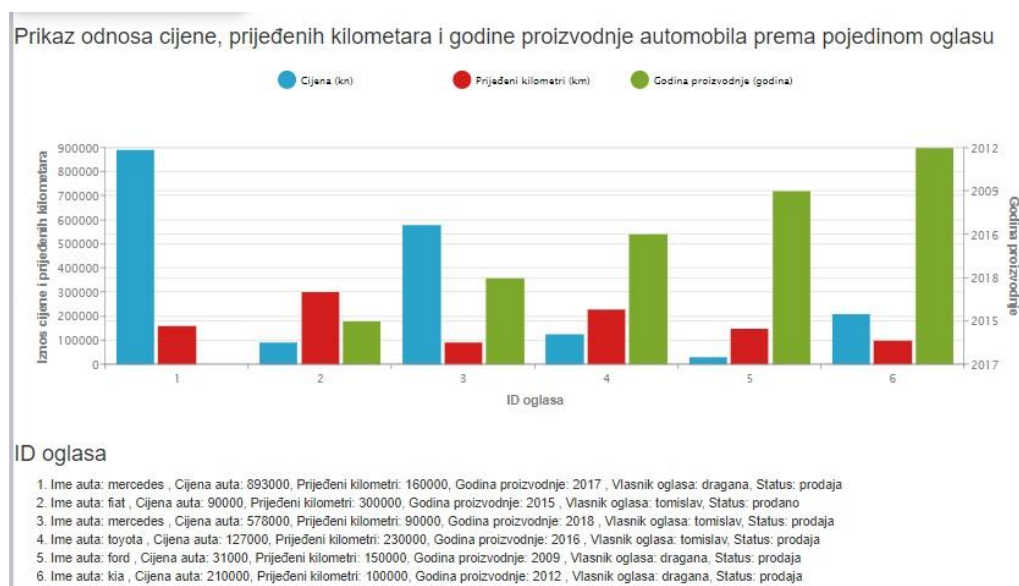


Slika 5.4. *Stranica s oglasima automobila*



Slika 5.5. Rezultati pretraživanja s riječi „mercedes“

Osim tražilice, prijavljeni korisnici na stranici s oglasima imaju i prikazan grafički prikaz podataka o cijeni, prijeđenim kilometrima i godini proizvodnje svih oglasa (slika 5.6.).



Slika 5.6. Stranica s oglasima, grafički prikaz informacija automobila prema pojedinom oglasu

Na slici 5.4. nalazi se gumb „Dodaj objavu“ koji prijavljenog korisnika preusmjerava na stranicu za kreiranje nove objave (slika 5.7.), ako korisnik nije prijavljen, preusmjerava ga na stranicu za prijavu. Prilikom dodavanja oglasa nije potrebno učitati sliku, a ako korisnik ne doda svoju sliku, učitat će se predefinirana slika. Osim opcije za kreiranje, prijavljeni korisnici imaju pravo na uređivanje, brisanje vlastitih objava te pisanja komentara pod objavama (slika 5.8.).

Dodaj oglas auta!

Ime automobila:

Max cijena:

Lokacija:

Max prijeđeni kilometri:

Godina proizvodnje:

Marka automobila:

Motor:

Boja:

Vrsta pogona:

Plin:

Klima:

Mjenjač:

Broj brzina:

Broj vrata:

Broj sjedala:

Ekološka kategorija vozila:

Vlasnik:

Audioradio:


Zračni jastuci:

Način plaćanja:

Snaga motora (u kW):

Maksimalna CO2 emisija (g/km):

Slika 5.7. Kreiranje novog oglasa



mercedes benz s objavio/la @dragana kontakt broj: 923586789 Cijena: 893000

- Boja: siva
- Lokacija: Osijek
- Marka automobila: mercedes
- Prijazeni kilometri: 180000
- Motor: diesel
- Godina proizvodnje: 2019
- Model: s
- Vrsta pogona: prednji
- Prit: na
- Klima: da
- Mjenjač: mehanički mjenjač
- Broj brzina:
- Broj vrata: 4
- Broj sjedala: 4
- Emission kategorija vtela: Euro 6
- Voznik: prvi
- Audiovideo: autoradio/CD/MP3/DVD
- Zračni jastuci: vozači + suvozači + bočni
- Snaga motora (u kW): 380
- Maksimalna CO2 emisija (g/km): 4
- Način paljenja: gasovins

Dostupna verzija samo sa 4.7 litarskim V8 benzincem s dva turbo punjača - 455KS i 700 Nm obrtnog momenta. Osim u verziji s pogonom na zadnje kotače, S 500 Coupe će se moći kupiti i s 4Matic pogonom na sva četiri kotača (S 500 Coupe 4Matic). S-Class Coupe dugačak je 5027, širok 1899, a visok 1411 mm. Međuosovinsko razmak iznosi 2943 mm. Najavljene su i verzije poput S63 AMG Coupea s 5.5 litarskim V8 agregatom od 599KS i 900 Nm, te S 600 Coupea i S65 AMG Coupe.

Objavljeno 10 days ago

Uredi Obrisi

Napisi komentar

Dodaj komentar!

admin 5 days ago
admin komentira

dragana 3 days ago
draganin komentar

Obrisi

Slika 5.8. Prikaz jednog oglasa

Na prethodnoj slici prikazan je primjer jednog oglasa čiji je vlasnik korisnik „dragana“, trenutno prijavljeni korisnik, zbog toga ovaj korisnik ima ponuđenu opciju uređivanja objave i brisanja objave, te brisanja vlastitog komentara. Odabirom opcije „Uredi“ korisnik će biti preusmjeren na stranicu za uređivanje objave sa slike 5.9.

Ažuriraj mercedes benz s

Marka automobila: mercedes ▼
 Odaberi model: ▼
 Motor: diesel ▼
 Status: prodaja ▼
 Boja: siva ▼

Učitaj sliku
 No file chosen
 format: jpg, png

Dostupna verzija samo sa 4.7 litarskim V8 benzincem s dva turbo punjača – 455KS i 700 Nm obrtnog momenta. Osim u verziji s pogonom na zadnje kotače, S 500 Coupé

Vrsta pogona: prednji ▼
 Pilo: da ▼
 Klima: da ▼
 Mjenjač: mehanički mjenjač ▼
 Broj brzina: ▼
 Broj vrata: 4 ▼
 Broj sjedala: 4 ▼
 Ekološka kategorija vozila: Euro 6 ▼
 Visenik: prvi ▼
 Audioradio: autoradio/CD/MP3/GVD ▼
 Zračni jastuci: vozački + suvozački + bočni ▼
 Način plaćanja: gotovina ▼

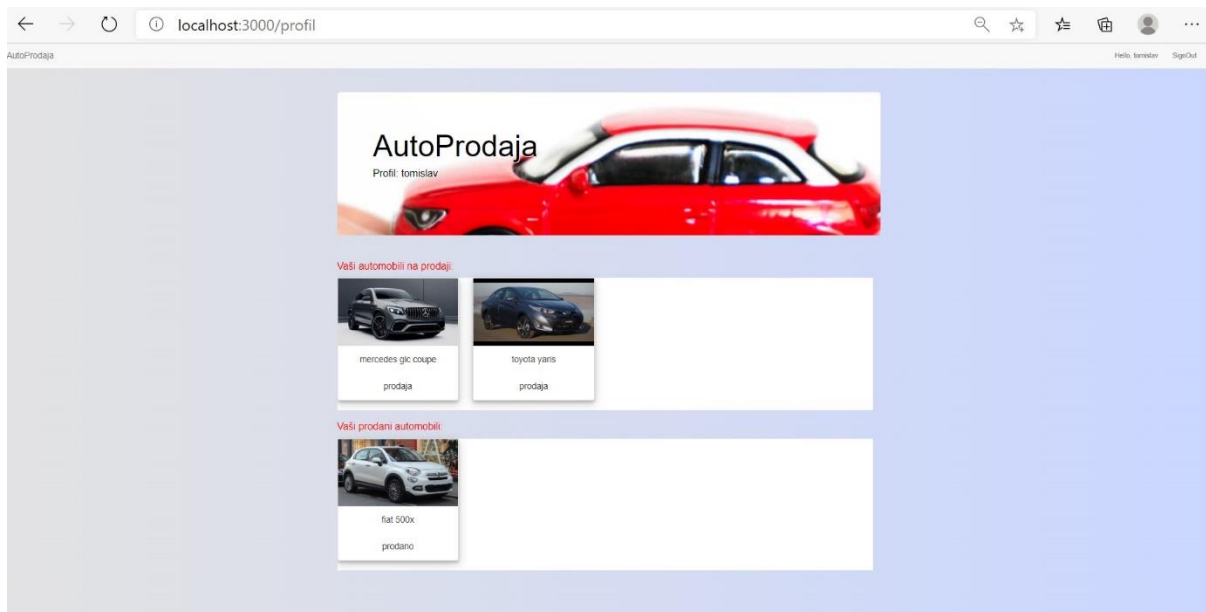
Snaga motora (u kW):

 Maksimalna CO2 emisija (g/km):

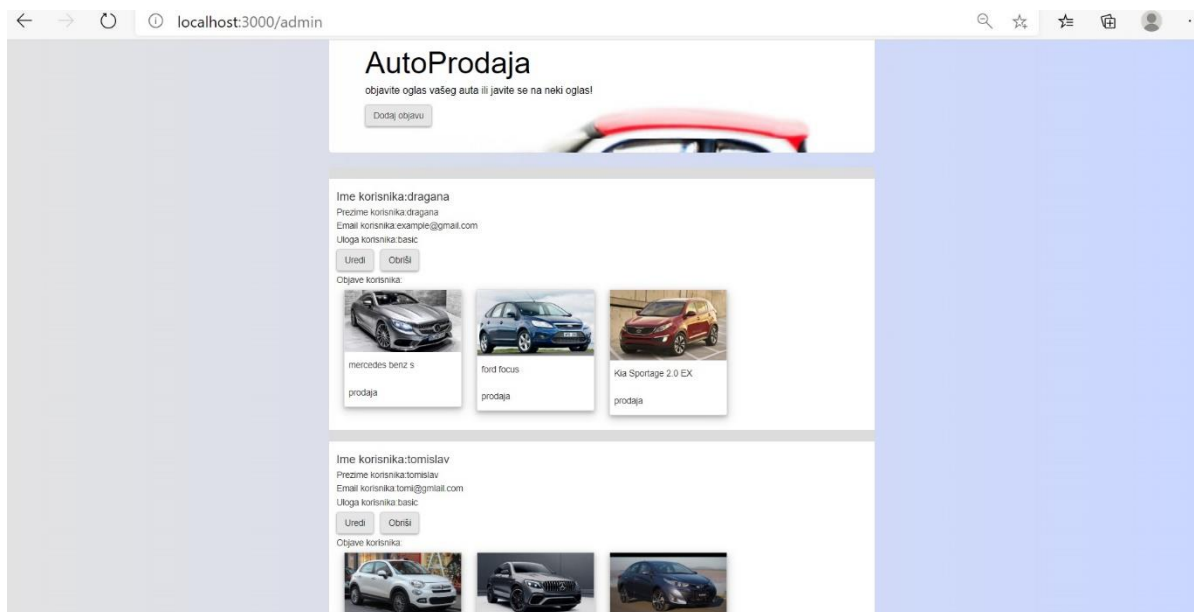
[Nazad](#)

Slika 5.9. Izgled web stranice za uređivanje objave

Prijavljeni korisnici imaju i vlastiti profil (slika 5.10.) gdje mogu pregledati vlastite prodane i neprodane automobile. Osim profila običnog korisnika, postoji i profil admina (slika 5.11.). Profil admina se razlikuje u odnosu na profil običnog korisnika jer ima prava na pregled svih kreiranih korisnika i njihovih objava, također adminu je dozvoljeno mijenjanje i brisanje svih podataka.



Slika 5.10. *Prikaz profila običnog korisnika*



Slika 5.11. *Profil admin*

6. ZAKLJUČAK

Ovaj diplomski rad imao je za zadatak objasniti kako koristeći tehnologije poput NodeJS-a, ExpressJS-a i MongoDB baze podataka na jednostavan način moguće kreirati funkcionalnu web aplikaciju. U radu je objašnjeno kako je napravljena web aplikacija za objavljivanje oglasa „AutoProdaja“ čija je namjena oglašavanje prodaje automobila. Za ovu aplikaciju napravljeni su osnovni elementi prijave, registracije, autorizacije korisnika, te CRUD operacije nad objektima. Aplikacija sadrži tražilicu kako bi korisnicima omogućila brži pristup željenim informacijama. Osim toga, aplikacija omogućuje prijavljenim korisnicima grafički prikaz odnosa cijene, prijeđenih kilometara i godine proizvodnje automobila među svim oglasima. Prema grafičkom prikazu korisnici mogu usporediti najvažnije informacije automobila između više oglasa kako bi lakše odlučili koji oglas ih više zanima. U izradi projekta koristili su se mnogi dodatci NodeJS-a (npr. *regex*, *passport*, *zingchart*) koji na jednostavan način i uz malo koda poboljšavaju funkcionalnosti aplikacije. Ovaj projekt služi kao primjer kako na jednostavan način napraviti funkcionalnu web aplikaciju za objavu oglasa, te kao takav uvijek ga je moguće nadograditi dodatnim opcijama koje pružaju ove tehnologije.

LITERATURA

- [1] Tutorials point, Node.js Introduction, dostupno na:
https://www.tutorialspoint.com/nodejs/nodejs_quick_guide.htm (posljednji pristup: 19.09.2020.)
- [2] NodeJS, An epic battle for developer mindshare, dostupno na:
https://images.idgesg.net/images/article/2018/04/iwan_05-100755063-orig.jpg (posljednji pristup: 19.09.2020.)
- [3] T.J Holowaychuk, D.C. Wilson, Npm, Npm Software registry, Npm package-lock.json A manifestation of manifest, dostupno na: <https://docs.npmjs.com/configuring-npm/package-lock-json.html> (posljednji pristup: 19.09.2020.)
- [4] T.J Holowaychuk, D.C. Wilson, Npm, Npm Software registry, Express, dostupno na:
<https://www.npmjs.com/package/express> (posljednji pristup: 19.09.2020.)
- [5] NodeJS, Express framework, dostupno na:
https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm (posljednji pristup: 19.09.2020.)
- [6] WEBnSTUDY, Callback funkcije u JavaScriptu, objavljeno 15.09.2015., dostupno na:
<http://www.webnstudy.com/tema.php?id=js-callback> (posljednji pristup: 19.09.2020.)
- [7] L.Vivah, Osinvačica CodingCrystals.com, THE BEGINNER'S GUIDE: Understanding Node.js & Express.js fundamentals, objavljeno 28.07.2017., dostupno na:
<https://medium.com/@LindaVivah/the-beginners-guide-understanding-node-js-express-js-fundamentals-e15493462be1> (posljednji pristup: 19.09.2020.)
- [8] MDN Web Docs, MDN i MDN sur., Express Tutorial Part 4: Routers and controllers, posljednje uređivano: 18.5.2020., dostupno na: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes (posljednji pristup: 19.09.2020.)
- [9] MongoDB, MongoDB Inc., Introduction to MongoDB, dostupno na:
<https://docs.mongodb.com/manual/introduction/> (posljednji pristup: 19.09.2020.)
- [10] A. Stojanović, Tehničko veleučilište u Zagrebu, Osvrt na NoSQL baze podataka – četiri osnovne tehnologije, dostupno na: <https://hrcak.srce.hr/file/283391> (19.09.2020.)
- [11] MongoDB, MongoDB Inc., Data model design, dostupno na:
<https://docs.mongodb.com/manual/core/data-model-design/> (posljednji pristup: 19.09.2020.)
- [12] MongoDB, MongoDB Inc., JSON and BSON, dostupno na:
<https://www.mongodb.com/json-and-bson> (posljednji pristup: 19.09.2020.)
- [13] Tutorials point, MongoDB, MongoDB Inc., MongoDB overview, dostupno na:

https://www.tutorialspoint.com/mongodb/mongodb_overview.htm (posljednji pristup: 19.09.2020.)

[14] Cloudinary, NodeJS, Node.JS SDK, dostupno na:

https://cloudinary.com/documentation/node_integration (posljednji pristup: 19.09.2020.)

[15] Cloudinary, Service overview, dostupno na:

https://cloudinary.com/documentation/solution_overview (posljednji pristup: 19.09.2020.)

[16] Cloudinary, How to integrate Cloudinary in your app, dostupno na:

https://cloudinary.com/documentation/how_to_integrate_cloudinary (posljednji pristup: 19.09.2020.)

[17] EJS, dostupno na: <https://ejs.co/#about> (posljednji pristup: 19.09.2020.)

[18] S. Jaiswal(Osnivač) JavaTpoint), JavaTpoint, CSS Tutorail, dostupno na:

<https://www.javatpoint.com/what-is-css> (posljednji pristup: 20.09.2020.)

[19] S. Jaiswal(Osnivač), JavaTpoint, Bootstrap, dostupno na:

<https://www.javatpoint.com/what-is-bootstrap> (posljednji pristup: 20.09.2020.)

[20] T. Powell (Osnivač), ZingSoft Inc., California C Corporation, ZingChart, dostupno na:

<https://www.zingchart.com/docs/getting-started/your-first-javascript-chart> (posljednji pristup: 26.09.2020.)

[21] T.J Holowaychuk, D.C. Wilson, Npm, Npm Software registry, Passport, dostupno na:

<https://www.npmjs.com/package/passport> (posljednji pristup: 20.09.2020.)

[22] MDN Web Docs, MDN i MDN sur., Express tutorial part 3, zadnje uređivano: 12.8.2020.,

dostupno na: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/mongoose (posljednji pristup: 20.09.2020.)

[23] T.J Holowaychuk, D.C. Wilson, Npm, Npm Software registry, Multer, dostupno na:

<https://www.npmjs.com/package/multer> (posljednji pristup: 20.09.2020.)

[24] MongoDB, MongoDB Inc., Regex, dostupno na:

<https://docs.mongodb.com/manual/reference/operator/query/regex/> (posljednji pristup: 20.09.2020.)

SAŽETAK

Ovaj rad predstavlja jednostavni način izrade aplikacije za objavljivanje oglasa i primjene navedenih tehnologija poput NodeJS-a, ExpressJS-a i MongoDB-a. U projektu korištene tehnologije su povezane na jednostavni način te tako omogućuje funkcionalnosti poput registracije, prijave korisnika, pretraživanje oglasa, te CRUD operacije nad objektima oglasa i komentara. Osim toga omogućen je i grafički prikaz pojedinih parametara.

Ključne riječi: ExpressJS, MongoDB, NodeJS, web aplikacija

ABSTRACT

Web application for publishing advertisements

This paper presents simple way of making app for publishing advertisements and applying such technologies as NodeJS, ExpressJS and MongoDB. Technologies used in this project are connected in a simple manner and as such they are enabling functionalities as registrations and logins of users, advertisement search and CRUD operations over objects of advertisements and comments. Besides that application enables graphic representation of specific parameters.

Keywords: ExpressJS, MongoDB, NodeJS, web application

ŽIVOTOPIS

Dragana Udovičić rođena je 13. studenog 1994. u Ozimici (Žepče, BiH). Završava osnovu školu Žepče u Žepču te nakon toga opću gimnaziju KŠC Don Bosco Žepče u Žepču. 2013. godine upisuje sveučilišni preddiplomski studij elektrotehnike na Elektrotehničkom fakultetu u Osijeku. 2016. godine završila je preddiplomski studij elektrotehnike, smjer Komunikacije i informatika, nakon čega upisuje diplomski studij na kojem se opredjeljuje za blok Mrežne tehnologije. Tečno priča i razumije engleski jezik, te se trenutno bavi database developmentom.

Dragana Udovičić

PRILOZI

[1] Github repozitorij: <https://github.com/dudovicic/AutoProdaja.git>

[2] CD s elektroničkom verzijom diplomskog rada i projektnim zadatkom