Web aplikacija za objavu oglasa

Udovičić, Dragana

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. Cloudinary 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

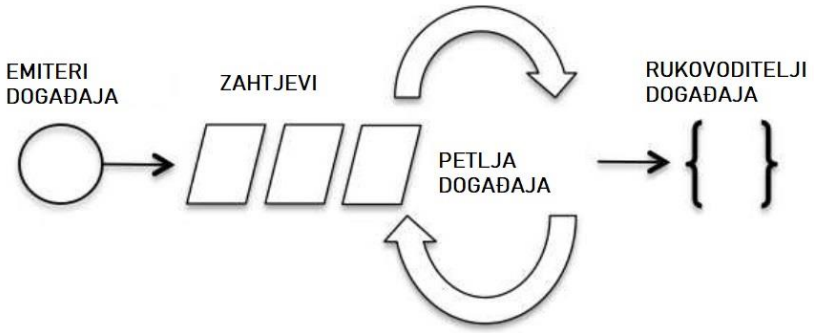
Na tržištu postoje mnogobrojne web aplikacije za prikazivanje oglasa. Većina je namijenjena širem oglašavanju proizvoda. Takve aplikacije su često nepregledne jer ne sadrže mogućnosti pretraživanja po pojedinim poljima ili sadrže mnogobrojne vrste proizvoda i time otežavaju pronalazak ciljanog sadržaja. „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 podatka 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 Managera*). 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.

Slika 3.2. *Primjer jednostavne aplikacije*

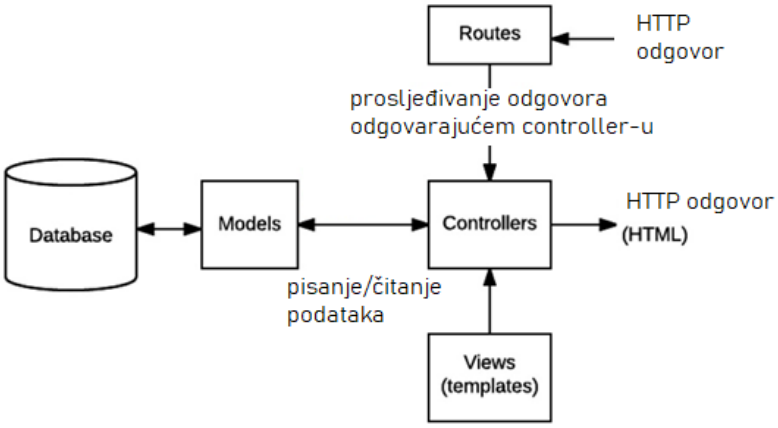
Na slici 3.2. možemo primijetiti *callback*[3] funkciju koja ima parametre *req* i *res*. *Req* predstavlja HTTP zahtjev (engl. *request*), dok *res* parametar predstavlja odgovor koji ExpressJS aplikacija šalje kada dobije HTTP zahtjev (engl. *response*). Rezultat koda iz slike 3.2. je ispisivanje željene 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,
- ap*p.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]*

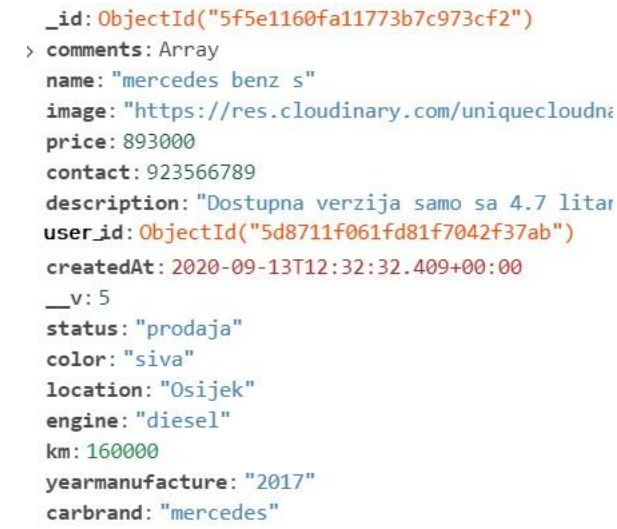
Prema slici iznad vidi se kako upravljači (engl. *controllers*) služe kako bi povezali modele (engl. *models*) i poglede (engl. *view*) te upravljali korisničkim zahtjevima. Modeli predstavljaju podatke za interakciju s bazom podataka, dok pogledi predstavljaju prikaz prethodno modeliranih podataka [8]. 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].

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.



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].

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 nedostatci 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.

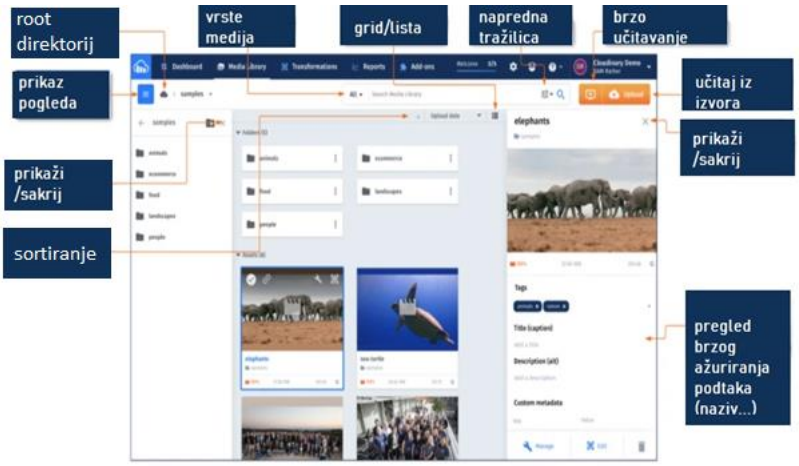
```
cloudinary.config({
  cloud_name: 'xxx',
  api_key: 'xxx',
  api_secret: 'xxx'
});
```

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 dodataka. 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 (2FA[4]), prema [15].

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

[4] 2FA (eng. *Two Facor Authentication*) provjera autetifikacije 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 predloška, udovoljava sustavu ExpressJS aplikacije. Instalacija ovog paketa u projektu radi se preko naredbe *npm install ejs* [17]. Izgled EJS predloška može se definirati uključivanjem zaglavlja (engl. *header*) i podnožja (engl. *footer*), a primjer strukture jedne ejs datoteke prikazuje slika 3.10.

```
<!DOCTYPE html>
<html>
  <head>
      <title>AutoProdaja</title>
      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.c
      <link rel="stylesheet" href="/stylesheets/app.css">
  </head>
  <body>

  </body>
</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 stanica 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 responzivnih[5] 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

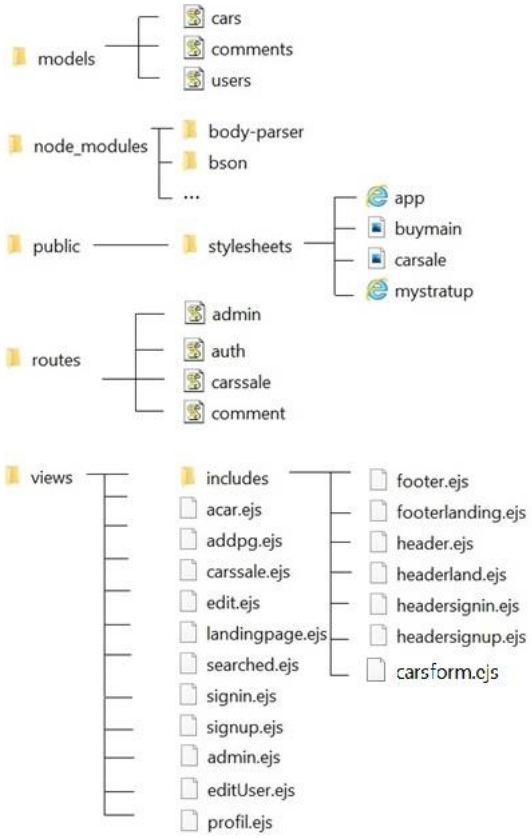
ZingChart JavaScript je komercijalni alat za prikazivanje grafova i interaktivnih karti, no dostupan je za besplatno isprobavanje, prema [20]. Instalacija ovog alata je jednostavna, može se instalirati preko *npm* naredbe ili izravno uključiti preko linka. Ovaj alat pruža različite vrste grafova i mogućnosti 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.

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].

```
<% include includes/headersignup.ejs %>

<div class="container" id="signup">
    <div class="row">
        <h1 style="text-align:center;">Registriraj se!</h1>
        <div style="width: 30%; margin:20px auto;">
            <form action="/signup" method="POST">
                        <div class="form-group">
                            <label>Ime</label>
                            <input  class = "form-control" type="text" name="firstname" placeholder="ime">
                        </div>

                        <div class="form-group">
                            <label>Prezime</label>
                            <input type="text" class="form-control"  name="lastname"  placeholder="prezime">
                        </div>

                        <div class="form-group">
                            <label>Email</label>
                            <input type="email" class="form-control" name="email" placeholder="email">
                        </div>

                        <div class="form-group">
                            <label>Korisničko ime</label>
                            <input  class = "form-control" type="text" name="username" placeholder="korisničko ime">
                        </div>

                        <div class="form-group">
                            <label>Lozinka</label>
                            <input type="password" class="form-control" name="password" id="exampleInputPassword1" placeholder="lozinka">
                        </div>

                        <div class="form-group">
                            <button class="btn btn-lg btn-default btn-block">Registriracija!</button>
                        </div>
            </form>
            <p>Prijavite se <a  href="/signin">Log In</a></p>
        </div>
    </div>
</div>

<% include includes/footer.ejs %>
```

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

```
router.get("/signout",function(req,res){
    req.flash("success","Uspješno ste odjavljeni!");
    req.logout();
    res.redirect("/");
});

router.post("/signup",function(req,res){
     users.register(new users(
         {
             firstname:req.body.firstname,
             lastname:req.body.lastname,
             email:req.body.email,
             username:req.body.username
     }),req.body.password,function(err,user){
        if(err){
            req.flash("error","Greška prilikom registarcije, molim probajte ponovno.");
            return res.render("signup");
        }
        passport.authenticate("local")(req,res, function(){
            req.flash("success","Dobrodosli "+user.firstname+" na stranicu AutoProdaja!");
            res.redirect("/carssale");
        });
    });
});
```

Slika 4.3. *Metode za registraciju korisnika*

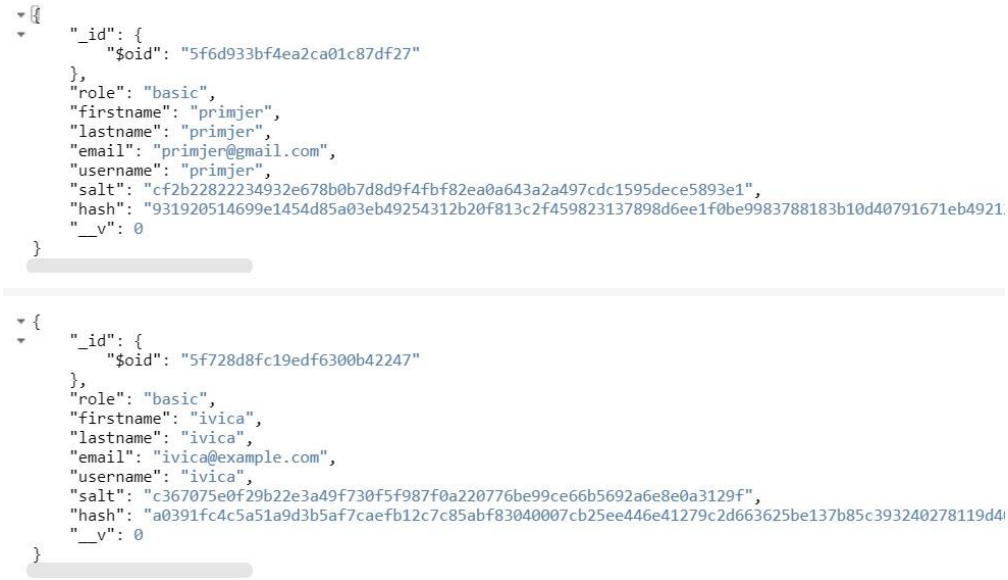
```
app.use(passport.initialize());
app.use(passport.session());
app.use(methodOverride("_method"));
app.use(flash());
app.locals.moment = require('moment');

passport.use(new LocalStrategy(users.authenticate()));
passport.serializeUser(users.serializeUser());
passport.deserializeUser(users.deserializeUser());
```

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.

```
var mongoose = require("mongoose"),
    passportLocalMongoose = require("passport-local-mongoose");

var usersSchema = new mongoose.Schema({
    firstname:String,
    lastname:String,
    email:String,
    username:String,
    password:String,
    role:{ type:String, default:"basic"},
});

usersSchema.plugin(passportLocalMongoose);
module.exports = mongoose.model("users",usersSchema);
```

Slika 4.5. *Shema users modela*



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 autentikacija, 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.

```
<% include includes/header.ejs %>
<div class="container">
    <div class="row">
        <h1 style="text-align:center;">Dodaj oglas auta!</h1>
        <div style="width: 30%; margin:20px auto;">
            <form action="/carssale" method="POST" id="add" enctype="multipart/form-data">
                <% include includes/carsform.ejs %>
                <p>Snaga motora (u kW): </p> <input class="form-control mr-sm-2" type="Number" name="enginepower" placeholder="snaga motora (u kW)"  max=1000 min=0>
                <p>Maksimalna CO2 emisija (g/km):</p> <input class="form-control mr-sm-2" type="Number" name="co2" placeholder="maksimalna CO2 emisija (g/km)"  max=1000 min=0>
                <div class="form-group">
                  <button class="btn btn-lg btn-default btn-block">Objavi!</button>
                </div>
            </form>
            <a  href="/carssale">Vrati se nazad</a>
        </div>
    </div>
</div>
<% include includes/footer.ejs %>
```

Slika 4.8. *Predložak za kreiranje nove objave*

```
router.post("/carssale",isLoggedIn,upload.single('carimage'),function(req,res){
    cloudinary.uploader.upload(req.file.path, function(result) {
            var name = req.body.carname;
            var price = req.body.carprice;
            var image = result.secure_url;
            var contact = req.body.carcontact;
            var description = req.body.descriptionform;
            var engine = req.body.engine;
            var km = req.body.km;
            var yearmanufacture = req.body.yearmanufacture;
            var carbrand = req.body.carbrand;
            var location = req.body.location;
            var color = req.body.color;
            var model = req.body.model;
            var typeofdrive= req.body.typeofdrive;
            var gas= req.body.gas;
            var airconditioning= req.body.airconditioning;
            var gearshift= req.body.gearshift;
            var numberofgears= req.body.umberofgears;
            var numberofdoors= req.body.numberofdoors;
            var numberofseats= req.body.numberofseats;
            var ecocategory= req.body.ecocategory;
            var numberofowner= req.body.numberofowner;
            var audio= req.body.audio;
            var airbags= req.body.airbags;
            var payment= req.body.payment;
            var enginepower= req.body.enginepower;
            var co2= req.body.co2;
            var owner={
                id:req.user._id,
                firstname:req.user.firstname
            }
            var newpg = {
                  name:name,
                  image:image,
                  price:price,
                  contact:contact,
                  description:description,
                  engine:engine,
                  km:km,
                  yearmanufacture:yearmanufacture,
                  carbrand:carbrand,
                  location:location,
                  color:color,
                  model:model,
                  owner:owner,
                  typeofdrive:typeofdrive,
                  gas:gas,
                  airconditioning:airconditioning,
                  gearshift:gearshift,
                  numberofgears:numberofgears,
                  numberofdoors:numberofdoors,
                  numberofseats:numberofseats,
                  ecocategory:ecocategory,
                  numberofowner:numberofowner,
                  audio:audio,
                  airbags:airbags,
                  payment:payment,
                  enginepower:enginepower,
                  co2:co2,
                }

              Cars.create(newpg, function(err, car) {
                if (err) {
                  console.log("ERROR");
                  return res.redirect('addpg');
                }
                req.flash("success","Uspješno ste dodali auto!");
                res.redirect('/carssale/');
              });
        });
});
```

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 isAuthenticate*d().*

```
function isLoggedIn(req,res,next){
    if(req.isAuthenticated()){
        return next();
    }
    req.flash("error","Morate biti prijavljeni!");
    res.redirect("/signin");
}
```

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 učitanu datoteku. Kako bi se uspostavila konekcija s oblakom, bilo je potrebno konfigurirati cloudinary parametre (slika 4.11.).

```
var storage = multer.diskStorage({
  filename: function(req, file, callback) {
    callback(null, Date.now() + file.originalname);
  }
});
var imageFilter = function (req, file, cb) {
    if (!file.originalname.match(/\.(jpg|jpeg|png|gif)$/i)) {
        return cb(new Error('Učitaj sliku!'), false);
    }
    cb(null, true);
};
var upload = multer({ storage: storage, fileFilter: imageFilter})

cloudinary.config({
  cloud_name: 'uniquecloudname',
  api_key: 'xxx',
  api_secret: 'xxx'
});
```

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).

```
var mongoose    = require("mongoose");

var carsSchema = new mongoose.Schema({
    name: String,
    image: String,
    status:  { type:String, default:"prodaja"},
    color: String,
    description:String,
    price:Number,
    location:String,
    contact:Number,
    engine:String,
    model:String,
    km:Number,
    yearmanufacture: String,
    carbrand:String,
    typeofdrive:String,
    gas:String,
    airconditioning:String,
    gearshift:String,
    numberofgears:String,
    numberofdoors:String,
    numberofseats:String,
    ecocategory:String,
    numberofowner:String,
    audio:String,
    airbags:String,
    payment:String,
    enginepower:Number,
    cod:Number,
    owner:{
       id:{ type: mongoose.Schema.Types.ObjectId,
        ref:"users"
       },
       firstname:String
    },
    createdAt: { type: Date, default: Date.now },
    comments:[{
        type: mongoose.Schema.Types.ObjectId,
        ref: "Comments"
    }]
}
);

var Cars = mongoose.model("Cars",carsSchema);
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.

```
var mongoose = require("mongoose");

var commentsSchema = new mongoose.Schema({
    text:String,
    author:{
       id:{ type: mongoose.Schema.Types.ObjectId,
        ref:"users"
       },
        firstname:String
    },
    createdAt: { type: Date, default: Date.now }

});

var Comments = mongoose.model("Comments",commentsSchema);
module.exports = Comments;
```

Slika 4.13. *Shema modela za komentare*

```
router.post("/carssale/:id/comment",ownership,function(req,res){
    var commentcomp = {
        text:req.body.text,
        author:{
            id:req.user._id,
            firstname:req.user.firstname
            }
        }
          Comments.create(commentcomp,function(err,comment){
                if(err){
                    console.log("error!");
                }
                else{
                     Cars.findOne({_id:req.params.id},function(err,car){

                       if(err){
                                console.log("error!!");
                            }
                            else{
                                 car.comments.push(comment);
                                 car.save(function(err,data){
                                        if(err){
                                            console.log("error!");
                                            }
                                        else{
                                            Cars.findById(req.params.id).populate("comments").exec(function(err,detailcar){
                                            if(err){
                                                        console.log("ERROR: ");
                                                        console.log(err);
                                                }
                                                else{
                                                        req.flash("success","Uspjesno dodan komentar!");
                                                        res.redirect("/carssale/"+detailcar._id);
                                                }
                                            });
                                        }
                                 });
                            }
                });
                }
          });
});
```

Slika 4.14. *Metoda za kreiranje komentara*

```
function ownership(req,res,next){
    if(req.isAuthenticated()){
        Cars.findById(req.params.id,function(err,car){
            if(err){
                req.flash("error","Nije pronađeno!")
                res.redirect("back");
            }
           else{
               if((car.owner.id.equals(req.user._id))  || (req.user.role='admin')) {
               next();
                }
                else{

                    req.flash("error","Nemate prava!");
                    res.redirect("back");
                }
              }
        });
    }
    else{
        req.flash("error","Prijavite se!");
        res.redirect("back");
    }
}
```

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.

```
router.get("/carssale/:id",function(req,res){
    Cars.findById(req.params.id).populate("comments").exec(function(err,detailcar){
        if(err){
            console.log("ERROR: ");
            console.log(err);
        }
        else{
                res.render("acar",{carinfo:detailcar});
        }
    });
});

```

Slika 4.16. *Primjer GET metode za dohvaćanje pojedinačnog oglasa*

```
<h4 class="pull-left">Automobili na prodaji:</h4>
        <div class="row text-right" style=" flex-wrap:wrap;
        float: left;
        width:  100%;
        height: 100%;
        object-fit: cover;
        ">
                <% pginfo.forEach(function(item){ %>
                  <% if(
                  (item.status=="prodaja")
                  ){ %>
                    <div class="col-md-3 col-sm-6">
                      <div class="thumbnail">
                          <a href="/carssale/<%= item._id %>">
                              <img src="<%= item.image %>">
                          </a>
                              <div class="caption">
                                    <h4> <%= item.name  %> </h4>
                                </div>
                                <div class="caption" style="display: none;">
                                      <h4> <%= item.description  %> </h4>
                                </div>
                    </div>
                  </div>
                   <% } %>
                  <% }); %>
      </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.

```
router.delete("/carssale/:id",ownership,function(req,res){
    Cars.findByIdAndRemove(req.params.id, function(err,car){
       if(err){
           res.redirect("/carssale")
       }
       else {
           Comments.deleteMany({_id:{$in:car.comments}}, function(err){
               if (err) {
                console.log(err);
                }
                else{
                req.flash("success","Uspješno ste izbrisali ovu objavu!");
               res.redirect("/carssale")
                }
                });

       }
    });
});
```

Slika 4.18. *DELETE metoda za brisanje objave*

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.

Slika 4.20. *Kreiranje opcije brisanja komentara*

U ovom dijelu opisane su CRUD (engl. *Create,Read,Update,Delete*) operacije nad podatcima. 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.

```
<!DOCTYPE html>
<html>
    <head>
      <title>AutoProdaja</title>
       <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css" integrity="
       <link rel="stylesheet" href="/stylesheets/app.css">
       <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
       <script src="https://cdn.zingchart.com/zingchart.min.js"></script>

        <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.

```
router.get("/search",function(req,res){
  try {
    Cars.find( {$and: [
            {price: {$lte : req.query.carprice }},
            {"$or":[
              { name: null }, { name:'' }, {name:{'$regex':req.query.carname, $options: 'i' }}
            ]},
            {"$or":[
            { color: null }, { color:'' }, {color:{'$regex':req.query.color }}
            ]},
            {"$or":[
            { location: null }, { location:'' }, {location:{'$regex':req.query.location, $options: 'i'  }}
            ]},
            {"$or":[
            { yearmanufacture: null }, { yearmanufacture:'' }, {yearmanufacture:{'$regex':req.query.yearmanufacture, $options: 'i'  }}
            ]},
            {"$or":[
            { carbrand: null }, { carbrand:'' }, {carbrand:{'$regex':req.query.carbrand }}
            ]},
            {"$or":[
            { engine: null }, { engine:'' }, {engine:{'$regex':req.query.engine }}
            ]},
            {"$or":[
            {km: {$lte : req.query.km }}
            ]},
            {"$or":[
            { model: null }, { model:'' }, {model:{'$regex':req.query.model, $options: 'i' }}
            ]},
            {"$or":[
            { typeofdrive: null }, { typeofdrive:'' }, {typeofdrive:{'$regex':req.query.typeofdrive, $options: 'i' }}
            ]},
            {"$or":[
            { gas: null }, { gas:'' }, {gas:{'$regex':req.query.gas, $options: 'i' }}
            ]},
            {"$or":[
            { airconditioning: null }, { airconditioning:'' }, {airconditioning:{'$regex':req.query.airconditioning, $options: 'i' }}
            ]},
            {"$or":[
            { gearshift: null }, { gearshift:'' }, {gearshift:{'$regex':req.query.gearshift, $options: 'i' }}
            ]},
            {"$or":[
            { numberofgears: null }, { numberofgears:'' }, {numberofgears:{'$regex':req.query.numberofgears, $options: 'i' }}
            ]},
            {"$or":[
            { numberofdoors: null }, { numberofdoors:'' }, {numberofdoors:{'$regex':req.query.numberofdoors, $options: 'i' }}
            ]},
            {"$or":[
            { numberofseats: null }, { numberofseats:'' }, {numberofseats:{'$regex':req.query.numberofseats, $options: 'i' }}
            ]},
            {"$or":[
            { ecocategory: null }, { ecocategory:'' }, {ecocategory:{'$regex':req.query.ecocategory, $options: 'i' }}
            ]},
            {"$or":[
            { numberofowner: null }, { numberofowner:'' }, {numberofowner:{'$regex':req.query.numberofowner, $options: 'i' }}
            ]},
            {"$or":[
            { audio: null }, { audio:'' }, {audio:{'$regex':req.query.audio, $options: 'i' }}
            ]},
            {"$or":[
            { airbags: null }, { airbags:'' }, {airbags:{'$regex':req.query.airbags, $options: 'i' }}
            ]},
            {"$or":[
            { payment: null }, { payment:'' }, {payment:{'$regex':req.query.payment, $options: 'i' }}
            ]},
            {"$or":[
            {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.

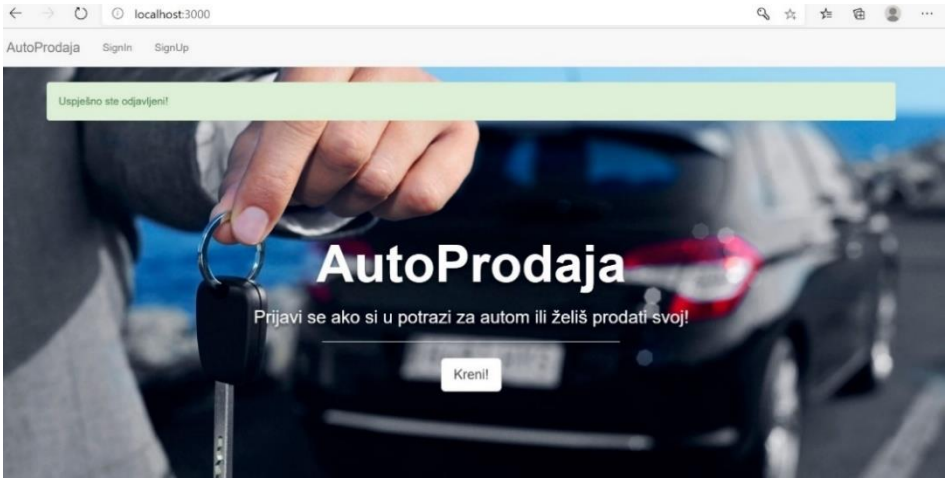
```
<% if( (curruser) ){ %>
          <div class="thumbnail"">

       <script>
      function httpGet(Url){
        var xmlHttp = new XMLHttpRequest();
        xmlHttp.open("GET", Url, false);
        xmlHttp.send(null);
        return xmlHttp.responseText;
      }

      window.onload=function(){
        var aData = JSON.parse(httpGet('http://localhost:3000/test'));

        var seriesData = {
          price: [],
          km: [],
          yearmanufacture: [],

        };

        for(var n = 0; n < aData.length; n++){
          seriesData['price'].push(aData[n]['price']);
          seriesData['km'].push(aData[n]['km']);
          seriesData['yearmanufacture'].push(aData[n]['yearmanufacture']);
        }


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

        });
      };
      </script>

     <h3 style=" margin-top: 100px;">Prikaz odnosa cijene, prijeđenih kilometara i godine proizvodnje automobila prema pojedinom oglasu</h3>
        <div id="myChart"></div>
        <h3>ID oglasa</h3>

        <ol>
          <% pginfo.forEach(item => { %>
          <li><%= `Ime auta: ${item.carbrand} , Cijena auta: ${item.price}, Prijeđeni kilometri: ${item.km},
            Godina proizvodnje: ${item.yearmanufacture}, Vlasnik oglasa: ${item.owner.firstname}, Status: ${item.status}` %></li>
          <% }); %>
        </ol>
    </div>
      </div>
      <% } %>
```

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.

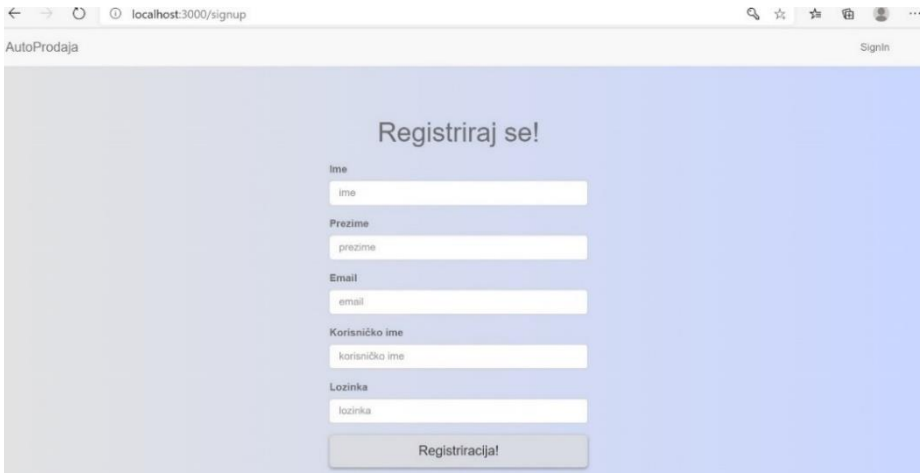
5. OPIS FUNKCIONALNOSTI I IZGLEDA WEB APLIKACIJE

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.

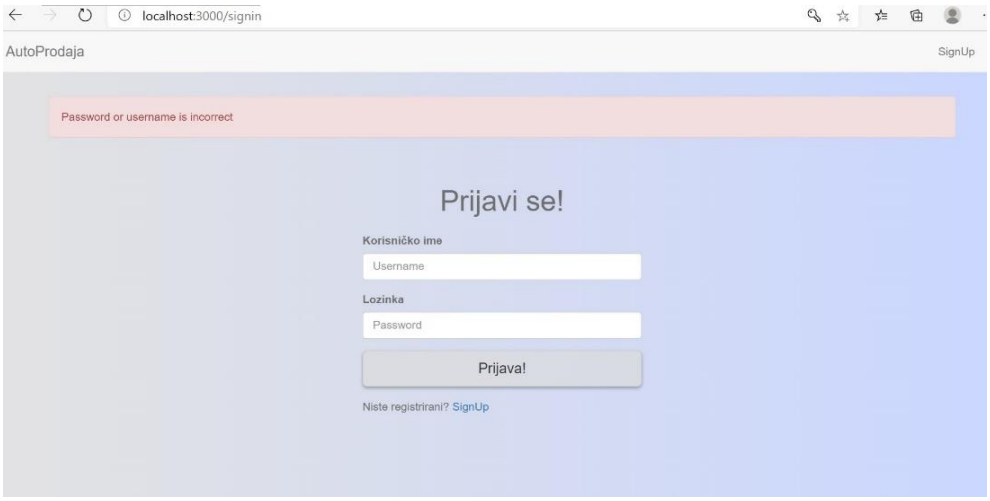


Slika 5.1. *Početna stranica web aplikacije „AutoProdaja“*

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.

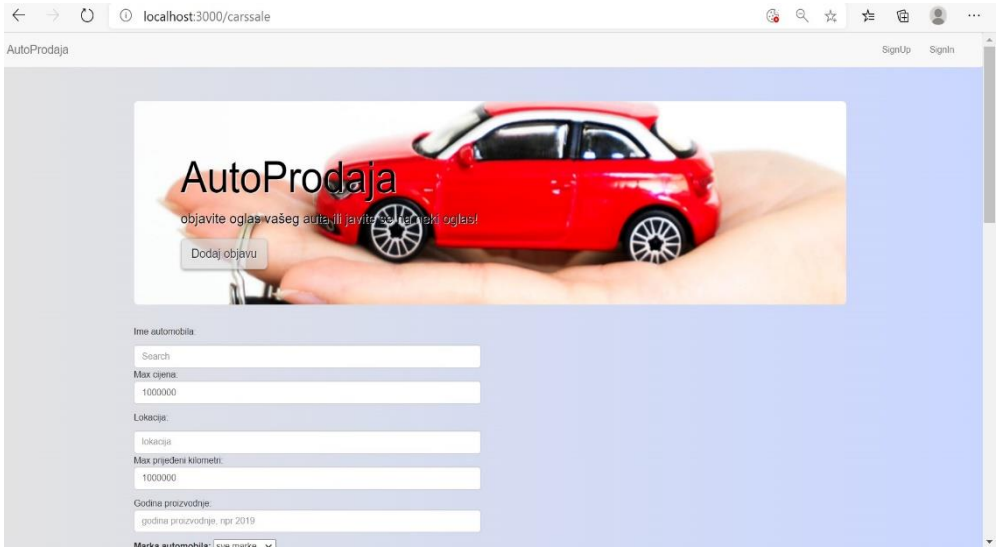


Slika 5.2. *Obrazac za registraciju korisnika*

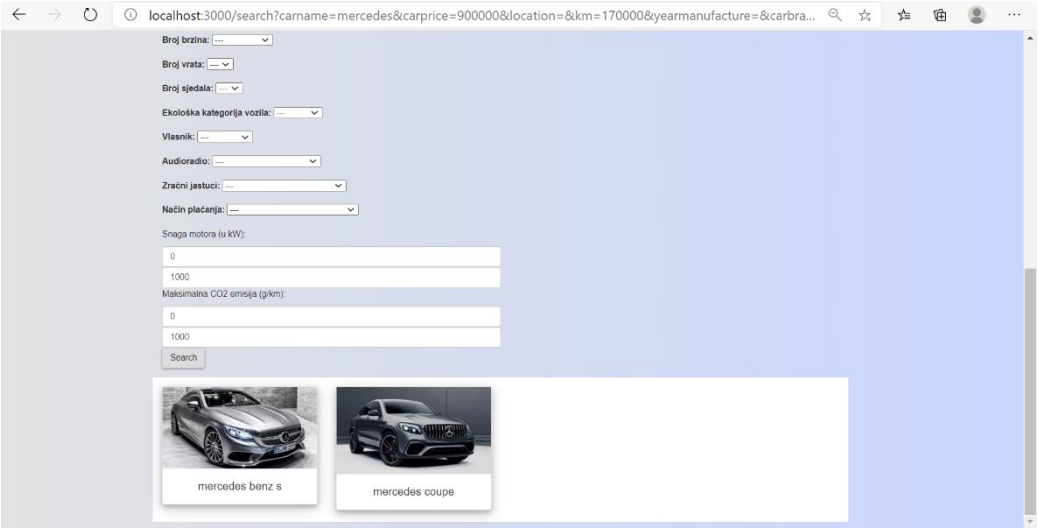


Slika 5.3. *Obrazac za prijavu korisnika*

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.

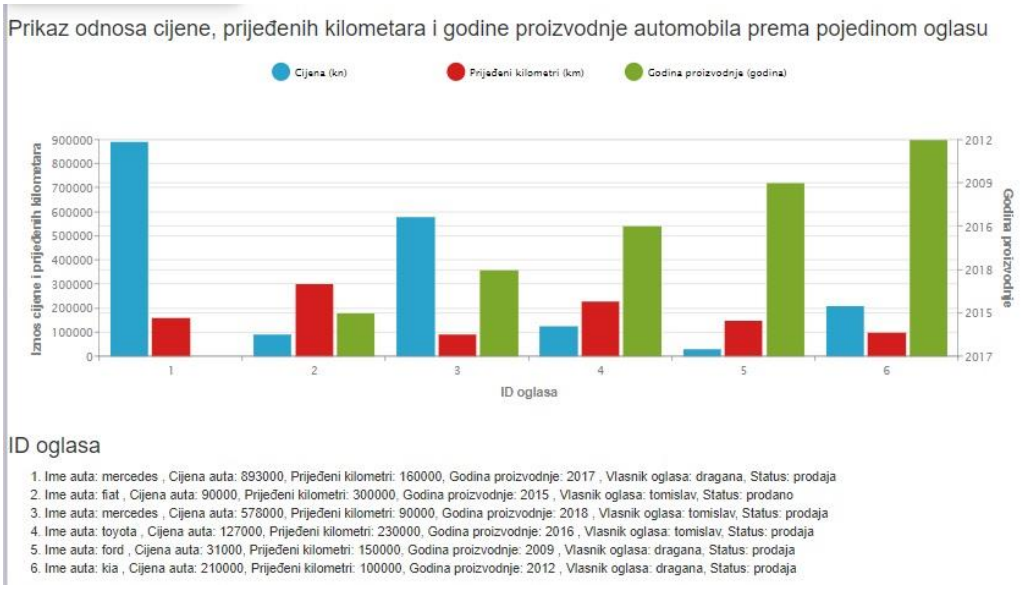


Slika 5.4. *Stranica s oglasima automobila*



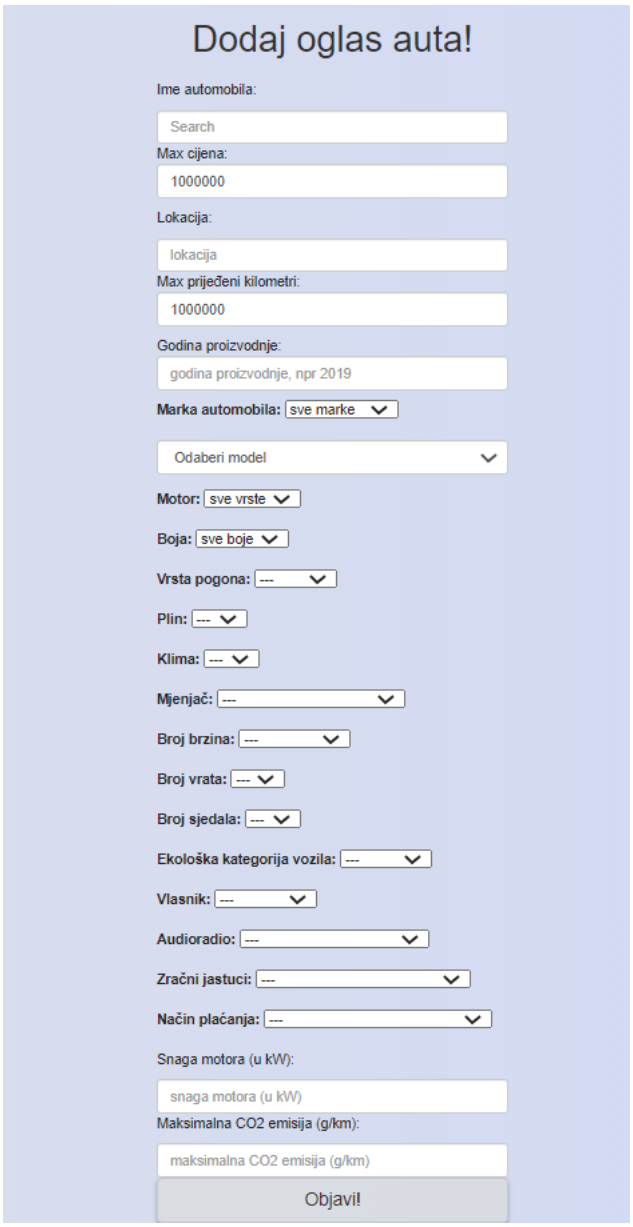
Slika 5.5. *Rezultati pretraživanja s riječi „mercedes“*

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.).

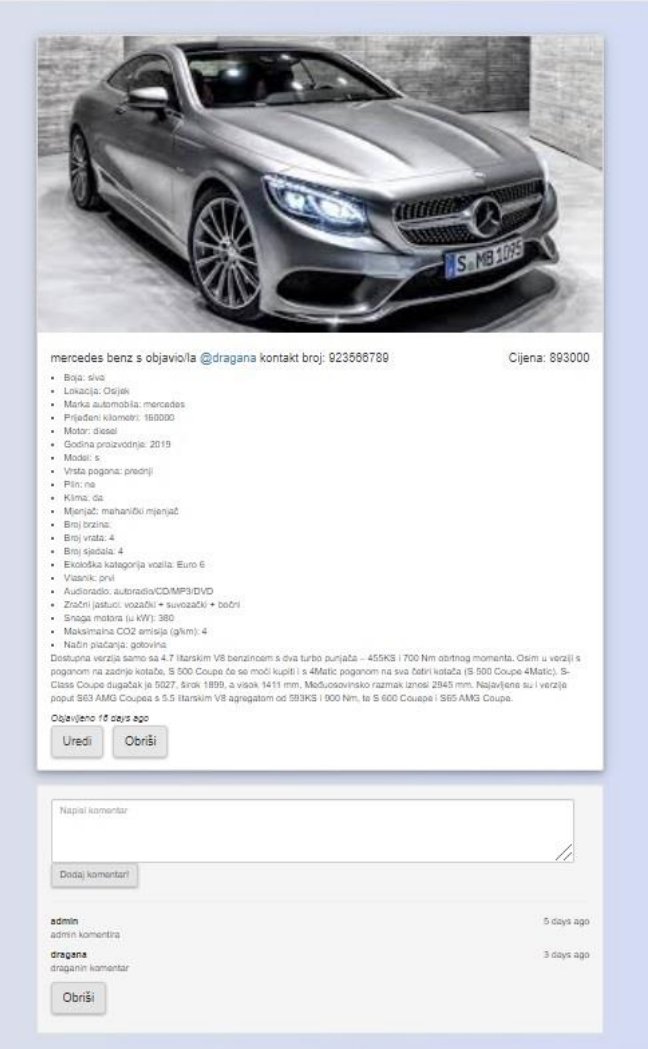


Slika 5.6. *Stranica s oglasima, grafički prikaz informacija automobila prema pojedinom oglasu*

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.).

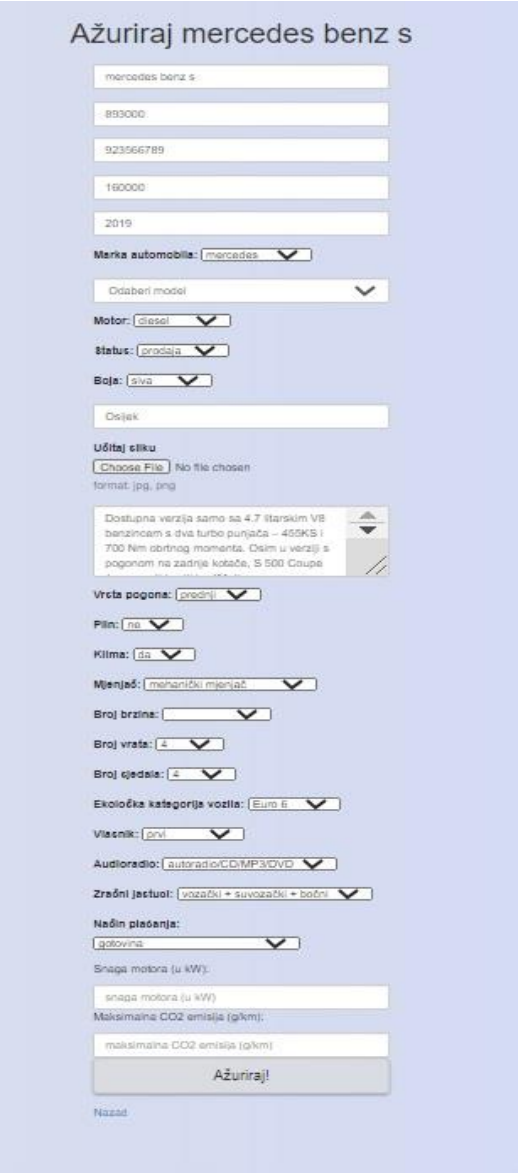


Slika 5.7. *Kreiranje novog oglasa*



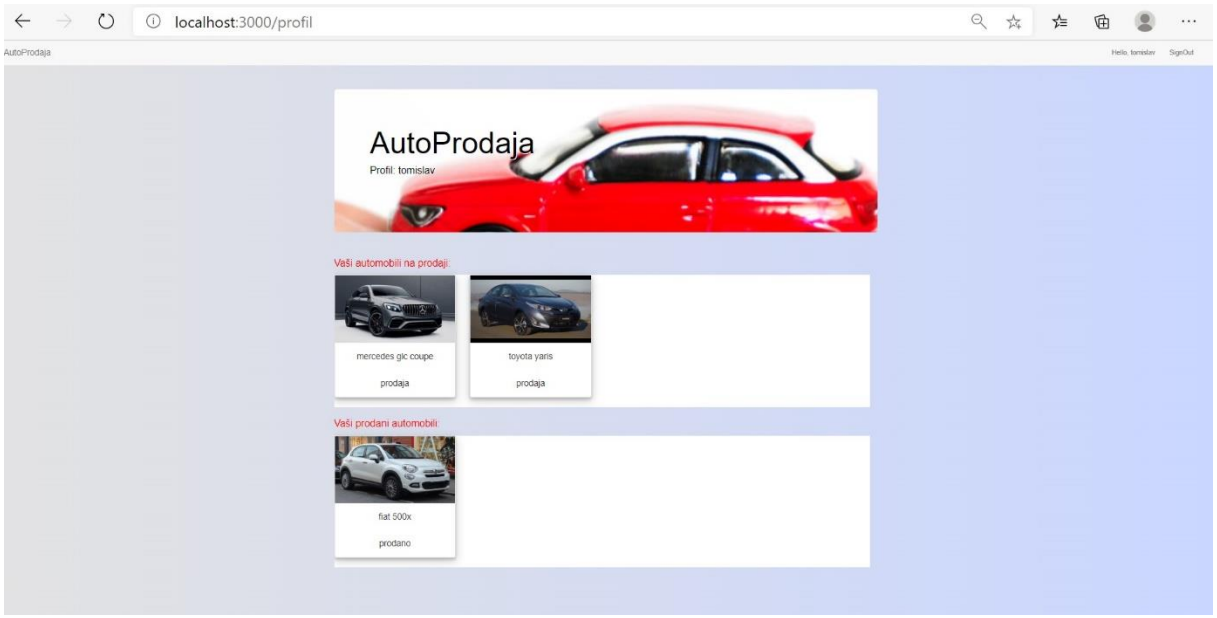
Slika 5.8. *Prikaz jednog oglasa*

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.

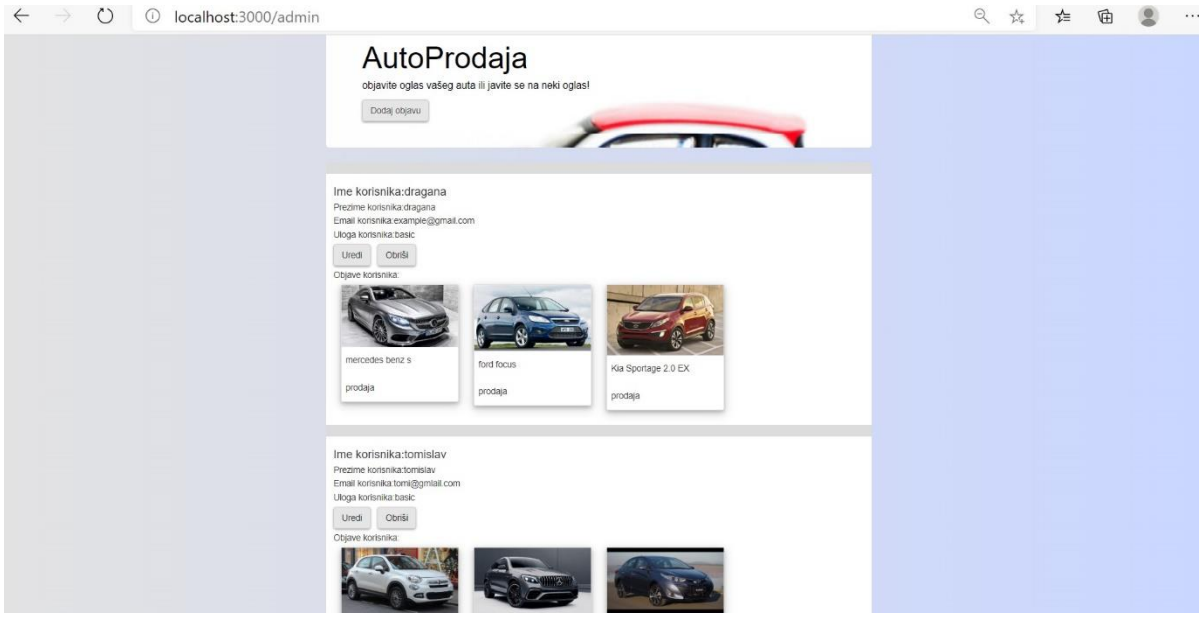


Slika 5.9. *Izgled web stranice za uređivanje objave*

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