Web aplikacija za rent-a-car tvrtku

Dolančić, Luka

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

Stručni studij

WEB APLIKACIJA ZA RENT-A-CAR TVRTKU

Završni rad

Luka Dolančić

Osijek, 2016

1. UVOD

Zadatak završnog rada je napraviti web sustav koji omogućava automatizaciju svakodnevnih zadataka, vođenje evidencije i upravljanje sadržajem koji je prikazan korisnicima za tvrtku čija je djelatnost iznajmljivanje automobila. Web aplikacija treba sadržavati javno sučelje koje će registriranim korisnicima omogućiti rezervacije automobila za najam u određenom periodu. Aplikacija također treba sadržavati administracijsko sučelje u kojem se omogućuje upravljanje zahtjevom za najam i upravljanje sadržajem koji se prikazuje na javnom dijelu stranice. Tijekom razvoja ovakvog sustava, pored samih funkcionalnost, vrlo je važno pažnju posvetiti i sigurnosti samog sustava, te sigurnosne propuste svesti na minimum. Završni rad je strukturiran u tri glavna poglavlja. Poglavlje "Tehnologije" detaljnije opisuje tehnologije korištene u implementaciji zadatka ovog završnog rada. Poglavlje "Funkcionalne karakteristike sustava" detaljnije opisuje funkcionalne zahtjeve prilikom izrade zadatka završnog rada i opisuje funkcionalne zahtjeve za izradu sustava iz perspektive korisnika sustava. Poglavlje "Implementacija" opisuje implementaciju funkcionalnih zahtjeva sustava kroz tehnologije opisane u poglavlju "Tehnologije". Ovo poglavlje također sadrži primjere izvornog koda korištenog pri implementaciji funkcionalnih zahtjeva.

2. TEHNOLOGIJE

Za izradu sustava za iznajmljivanje automobila potreban je niz različitih tehnologija kako bi se potpuno zadovoljili svi funkcionalni zahtjevi. Tehnologije korištene za razvoj sustava su:

- HTML 5 i CSS 3
- Bootstrap 3.3.6
- Laravel 5.2
- PHP 7.0
- MySql 5.7
- Javascript
- jQuery.

2.1. Opis tehnologija

HTML (eng. *HyperText Markup Language*) je prezentacijski jezik za izradu web stranica. Prikaz hipertekst dokumenta omogućuje web preglednik, boljerečeno zadaća web preglednika je interpretiranje HTML izvornog koda i njeogovo prikazivanje krajnjem korisniku. Temeljna zadaća HTML jezika jest uputiti web preglednik kako prikazati hipertext dokument. HTML kod se sastoji od oznaka (eng. *tags*) koje mogu služiti i za semantičko označavanje dijelova web stranice što moderni internet preglednici mogu interpretirati na neki način. Na primjer *article* oznaka u HTML5 verziji govori da je sadržaj koji se nalazi unutar tih oznaka nekakav članak, te ga određeni internet preglednici mogu na taj način interpretirati i uz njega nuditi neke dodatne mogućnosti kao što je pregled članka u punom zaslonu. Primjer HTML koda prikazan je u primjeru 2.1.1. U ovom primjeru prikazano je nekoliko vrlo bitnih elemenata HTML jezika kao što su h1 (koji semantički ima funkciju naslova najveće važnosti) element i p element (koji semantički ima funkciju paragrafa teksta).

```
<html>
<head>
    <meta charset="UTF-8">
        <link href="/css/style.css">
       <title>Naziv web stranice</title>
</head>
<body>
    <h1>Naslov paragrafa</h1>
        <p>Paragraf teksta</p>
</body>
</html>
```

Primjer 2.1.1. HTML izvorni kod s nekoliko oznaka

CSS (eng. Cascading Style Sheets) je stilski jezik koji služi za definiranje stilskih atributa hipertekst dokumenta. Svakom elementu HTML koda moguće je definirati stilske atribute. Same HTML oznake bez CSS koda imaju neke početne stilske atribute od strane internetskog preglednika. CSS kod daje mogućnost da se ti stilski atributi prepišu novim atributima koje definira programer. HTML elementima mogu se postaviti class i id atributi, preko kojih u CSS kodu možemo točno usmjeriti stilske atribute prema željenom elementu. Jedna od velikih zapreka web dizajna je nedostatak kontinuiteta u početnim stilskim atributima HTML elemenata od strane internet preglednika. Različiti internet preglednici često imaju različite početne stilske vrijednosti za različite atribute. Da bi se taj problem riješio potrebno je napisati CSS kod koji služi za resetiranje početnih vrijednosti HTML atributa kako bi se postigle iste početne vrijednosti kroz sve internet preglednike. Ta metoda se još naziva i "CSS reset". Primjer CSS koda koji može mijenjati stilske atribute HTML koda iz primjera 2.1.1 je dan u primjeru 2.1.2. U tom primjeru je h1 HTML elementu dan stilski atribut pozadinske boje.

```
h1 {
	background-color: #03fa34;
}
```

Primjer 2.1.2. CSS izvorni kod koji daje stilski atribut HTML kodu iz primjera 2.1.1

Bootstrap je platforma koja programeru pruža unaprijed određeni niz stilskih definicija za neke elemente koji se učestalo koriste u izradi web stranica. Primjeri takvih elemenata su: navigacijske trake, slike, razni gumbi, obrasci i polja u obrascima itd. Bootstrap platforma također uključuje i takozvani "css reset" o kojem je bilo riječi u tekstu prije. Glavni cilj Bootstrapa je pojednostavljenje responzivnog web dizajna. To znači da je cilj izrade web stranice omogućiti njen pregled na uređajima različitih rezolucija, počevši od širokih fullHD ekrana koji se uglavnom koriste na računalima do mobilnih uređaja i tableta. Bootstrap ovdje pruža veliku količinu klasa pomoću kojih možemo na razmjeran način podijeliti ekran po sadržaju.

Laravel je PHP platforma koja omogućuje olakšani i ubrzani razvoj web aplikacija. Programeru pruža unaprijed definiranu arhitekturu aplikacije, te određenu razinu apstrakcije iznad osnovnog PHP jezika i njegovih funkcionalnost. Više o Laravelu u potpoglavlju 2.3.

PHP je skriptni jezik napravljen za razvoj dinamičkih web stranica, a često se koristi i kao programski jezik opće namjene. PHP kod je obično procesiran od strane PHP interpretera koji je implementiran kao modul unutar web poslužitelja. Više o PHP-u i njegovoj sintaksi u potpoglavlju 2.2.

MySql je sustav upravljanja relacijskom bazom podataka i napisan je u programskim jezicima C i C++. MySql sustav nam omogućava različite načine upravljanja podacima unutar naše aplikacije. Podaci unutar MySql sustava su organizirani u tablicama, a tablice mogu biti međusobno povezane relacijama. Sustav omogućava da više korisnika imaju svoje baze na jednom računalu, a MySql vodi brigu o pravima pristupa raznim bazama podataka od strane korisnika.

Javascript je interpretirani programski jezik koji se u kontekstu web stranica najčešće koristi na strani klijenta, boljerečeno izvodi se na internet pregledniku. U zadnjih nekoliko godina javascript se također koristi i kao programski jezik koji se izvodi na strani poslužitelja (NodeJS), što programerima pruža mogućnost pisanja programskog koda u istom programskom jeziku na strani poslužitelja i klijenta. Na strani klijenta, javascript programski jezik ima pristup HTML elementima preko DOM-a (na engleskom *Document Object Model*). Javascript tako može na strani klijenta izvoditi kod i upravljati raznim elementima nakon što je HTML stranica vraćena korisniku kao rezultat HTTP zahtjeva. To omogućuje vrlo dinamičke interaktivne aplikacije na strani klijenta. Javascript nema direktan pristup bazama podataka iz sigurnosnih razloga, ali moguće je postići dohvaćanje iz baze podataka preko API-a (na engleskom *Application Programming Interface*) koji je definiran na strani poslužitelja. U primjeru 2.1.3 dan je javascript izvorni kod koji na strani poslužitelja ispisuje upozorenje (na engleskom *alert*) na ekran unutar internet preglednika. Javascript izvorni kod moguće je pisati unutar HTML dokumenta unutar *script* oznaka.

```
<script>
      alert('Ovo je upozorenje');
</script>
```

Primjer 2.1.3. Javascript izvorni kod napisan unutar HTML dokumenta unutar script oznaka

jQuery je javascript biblioteka koja pruža lakši način upravljanja s DOM (na engleskom *Document Object Model*) elementima unutar web stranica. jQuery korisniku (programeru) pruža jednostavan API za upravljanje DOM elemntima.

2.2. PHP programski jezik

PHP je interpretirani objektno orijentirani programski jezik, čija je velika prednost u izradi web stranica što se može pisati u HTML izvorni kod. Samim time PHP može u HTML kod ispisivati dinamički kreirani sadržaj, procesirati sadržaj unesen preko obrazaca, spremati sadržaj u bazu podataka kao što je MySql pomoću MySqli ekstenzije, kreirati sustav autentikacije korisnika i još

mnogo toga. PHP ima niz globalnih varijabli pomoću kojih se mogu postići razne funkcionalnosti od kojih su neke navedene u tekstu gore. U primjeru 2.2.1 prikazan je primjer načina korištenja HTML-a i obrazaca u kojem je dato polje za unos imena osobe. U HTML kodu je moguće definirati na koju PHP skriptu se šalju podaci iz obrasca, te kojom HTTP metodom se šalju podaci. Većina modernih preglednika prepoznaje samo GET i POST HTTP metode. U primjeru 2.2.2 prikazan je izvorni kod datoteke procesiranje.php na koju izvorni kod iz primjera 2.2.1 šalje podatke POST metodom. Ovaj PHP kod koristi superglobalnu $_POST varijablu kako bi primio podatke iz obrasca koji je podatke poslao POST metodom. Na kraju se podatak iz obrasca ispisuje na ekran.

```
<html>
<head>
</head>
<body>
	<form action="procesiranje.php" method="POST">
		<input type="text" name="ime">
		<input type="submit" value="posalji">
	</form>
</body>
</html>
```

Primjer 2.2.1. HTML izvorni kod s obrascem i poljem za unos imena

```
<?php
	$ime = $_POST['ime'];

	echo "Uneseno ime iz obrasca u primjeru 1.2.2.1 je: " . $ime;
?>
```

Primjer 2.2.2. PHP izvorni kod koji prima podatke iz obrasca POST metodom

Moderne PHP platforme kao što je Laravel koriste objektno orijentirane karakteristike PHP jezika da bi postigle kvalitetnu organizaciju izvornog koda, te skalabilnu arhitekturu aplikacije. Objektno orijentirani način programiranja platforme obično implicira da je kod distribuiran u klase na način da svaka klasa ima jednu i samo jednu svrhu, te obično u jednoj php datoteci se nalazi samo jedna klasa. Dobar primjer takve klase je dan u primjeru 2.2.3 gdje imamo klasu Kocka koja pruža samo razne metode za manipulacijom informacija nad objektom klase Kocka.

```
<?php

class Kocka
{
	private $stranicaA;

	public function setStranicaA($unos)
	{
		$this->stranicaA = $unos;
	}

	public function getStranicaA()
	{
		return $this->stranicaA;
	}

	public function izracunajVolumen()
	{
		$a = $this->stranicaA;

		return $a * $a * $a;
	}
}
```

Primjer 2.2.3. Klasa Kocka

Php jezik sadrži nekoliko superglobalnih varijabli koje imaju različite uloge u razvoju različitih funkcionalnost modernih web stranica. Popis svih superglobalnih varijabli unutar PHP jezika:

- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION.

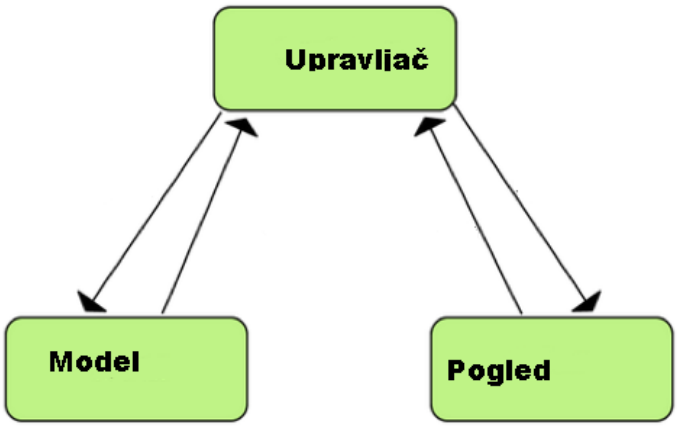
Uloga $_POST, $_GET i $_REQUEST superglobalnih varijabli je primanje podataka iz formi. $_SERVER superglobalna varijabla sadrži podatke o poslužitelju na kojem se izvodi trenutačna php skripta. Primjer tih podataka su IP adresa poslužitelja, verzija HTTP protokola preko kojeg je primljen zahtjev koji se trenutačno obrađuje, dohvaćanje naziva skripte koja se trenutačno izvodi itd. $_COOKIE i $_SESSION superglobalne varijable imaju ulogu prilikom razvoja korisničkog sustava unutar aplikacije. Njihova uloga u sustavu je da korisnik ima mogućnost ostati prijavljen u sustavu između raznih zahtjeva prema poslužitelju, tako da kad korisnik posjeti neku stranicu unutar našeg sustava korisnik ostaje prijavljen.

2.3. Laravel platforma

Laravel je MVC (na engleskom *Model-View-Controller*) platforma čija je glavna svrha korisnicima (u ovom slučaju korisnici su programeri) pružiti sučelje za programiranje koje će uvelike olakšati proces razvoja aplikacija i sustava, te korisniku pružiti već unaprijed definiranu arhitekturu aplikacije. Laravel u pozadini za većinu funkcionalnosti koristi komponente Symfony platforme, koje opet Laravelu pružaju apstraktno sučelje za upravljanje osnovnim funkcionalnostima PHP jezika i raznim apstrakcijama koje se koriste u razvoju web sustava. Jedan od primjera Symfony komponenti koje Laravel koristi je *Request* klasa preko koje možemo dohvatiti sve informacije o trenutačnom zahtjevu koji sustav obrađuje. Sama Symfony-eva *Request* klasa je apstrakcija iznad PHP-ovih superglobalnih varijabli i pruža programeru ugodnije sučelje za upravljanje funkcionalnostima. Na kraju, Laravel koristi Symfony *Request* klasu da bi iznad nje stvorio još jedan nivo apstrakcije i korisniku (programeru) pružio svoje željeno sučelje koje sadržava neke dodatne funkcionalnost koje Symfony *Request* klasa ne pruža.

Laravel koristi MVC oblikovni obrazac. U samim počecima PHP izvorni kod se većinom pisao unutar HTML koda u svojim specifičnim oznakama. To je često značilo da u istoj datoteci imamo izvorni kod HTML-a koji ima funkciju definiranja elemenata web stranice, PHP izvorni kod koji se izvodi na poslužitelju, te uz PHP su se po potrebi pisali i SQL zahtjevi da bi se sadržaj mogao dohvaćati i spremati u bazu podataka. Ovakav izvorni kod bi obično rezultirao lošom arhitekturom aplikacije, te što bi aplikacija s vremenom bila kompleksnija to bi izvorni kod bilo teže održavati. Jedna od velikih prednosti MVC oblikovnog obrasca je što rješava ovaj problem. MVC izvorni kod dijeli na tri cjeline:

- Model - dio koji je zadužen za bazu podataka
- Pogled (na engleskom *View*) - dio koji je zadužen za prezentaciju prema krajnjem korisniku
- Upravljač (na engleskom *Controller*) - dio koji koordinira informacije s modela ka pogledu i obrnuto, najčešće se koristi za povezivanje koda s dvije navedene strane



Sl. 2.3.1. MVC oblikovni obrazac

Model je dio MVC oblikovnog obrasca koji je zadužen za komunikaciju s bazom podataka. Laravel platforma koristi Eloquent ORM (na engleskom *Object Relational Mapping*). ORM je sustav koji pruža nivo apstarkcije nad bazom podataka. ORM programeru omogućava da na objektno orijentirani način pristupama bazi podataka umjesto da piše SQL upite. Jedna od zadaća ORM-a je pretvaranje metoda pozvanih nad model objektima u SQL upite. Primjer takvog korištenja Eloquent ORM-a je prikazan u primjeru 2.3.2. Eloquent ORM izvorni kod iz primjera 2.3.2 pretvara u MySql upit u primjeru 2.3.3.

```
$cars = App\Models\Cars::all();
```

Primjer 2.3.2. Primjer korištenja Eloquent ORM-a

```
"select * from `cars`"
```

Primjer 2.3.3. MySql kod u koji Eloquent ORM pretvori kod iz primjera 2.3.2

Pogled (na engleskom *View*) je zadužen za generiranje krajnjeg HTML koda koji se kao rezultat korisničkog zahtjeva vraća korisniku. Laravel kao sustav predložaka (na engleskom *templating engine*) koristi Blade. Sustav predložaka omogućava modularan razvoj samog dizajna web stranice. Za razliku od pisanja čistog HTML koda, korištenje sustava predložaka kao što je Blade ima prednost da se izvorni kod koji je isti kroz nekoliko datoteka ne mora duplicirati kao što je to slučaj kod HTML datoteka. Jedna od prednosti Bladea je da omogućava odvajanje izvornog koda koji se ponavlja na više stranica i njegovo umetanje u samo jednu Blade datoteku. Blade također pruža sintaksu za razne petlje i funkcije, te ispis varijabli koje su poslane iz upravljača. Kao sustav predložaka može se koristiti i PHP, ali Laravel koristi Blade iz razloga što je njegova sintaksa puno lakša za korištenje i održavanje. Primjer Blade sintakse dan je u primjeru 2.3.4.

```
@extends('layouts.main')
@section('content')
      @foreach($cars as $car)
            {{ $car->brand }}
      @endforeach
@endsection
```

Primjer 2.3.4. Blade sintaksa

Upravljač (na engleskom *Controller*) je kod koji najčešće povezuje modele i poglede. Njegova glavna zadaća je prepoznavanje korisničkog zahtjeva i odgovaranje na taj zahtjev. Proces odgovora na korisnički zahtjev može biti dohvaćanje određenih podataka iz baze preko ORM-a i prikaz podataka iz baze preko Blade sustava predložaka, što kao krajnji rezultat korisniku vraća HTML kod koji se prikazuje u njegovom internet pregledniku. Jedan od načina uporabe upravljača prikazan je u primjeru 2.3.5.

```php
<?php
namespace App\Http\Controller;

use App\Models\Car;

class CarController
{
	public function index()
	{
		$cars = Car::all();

		return view('car.index', ['cars' => $cars]);
	}
}
```

Primjer 2.3.5. Upravljač

U Laravel MVC platformi se koristi napredan sustav ruta (na engleskom *routing system*). Sustav ruta funkcionira tako da dohvati URL trenutačnog zahtjeva, te pokušava pronaći koja metoda je u sustavu registrirana za obradu tog zahtjeva. Najčešće je ta metoda neka metoda unutar klase nekog od upravljača. U primjeru 2.3.6 je definirana ruta za upravljač iz primjera 2.3.5. Kada korisnik u internetskom pregledniku posjeti URL "/car", sustav ruta iz primjera 2.3.6 prepoznaje taj URL jer je definiran u sustavu i samim time poziva index metodu u CarController-u.

```php
<?php

Route::get("/car", "CarController@index");
```

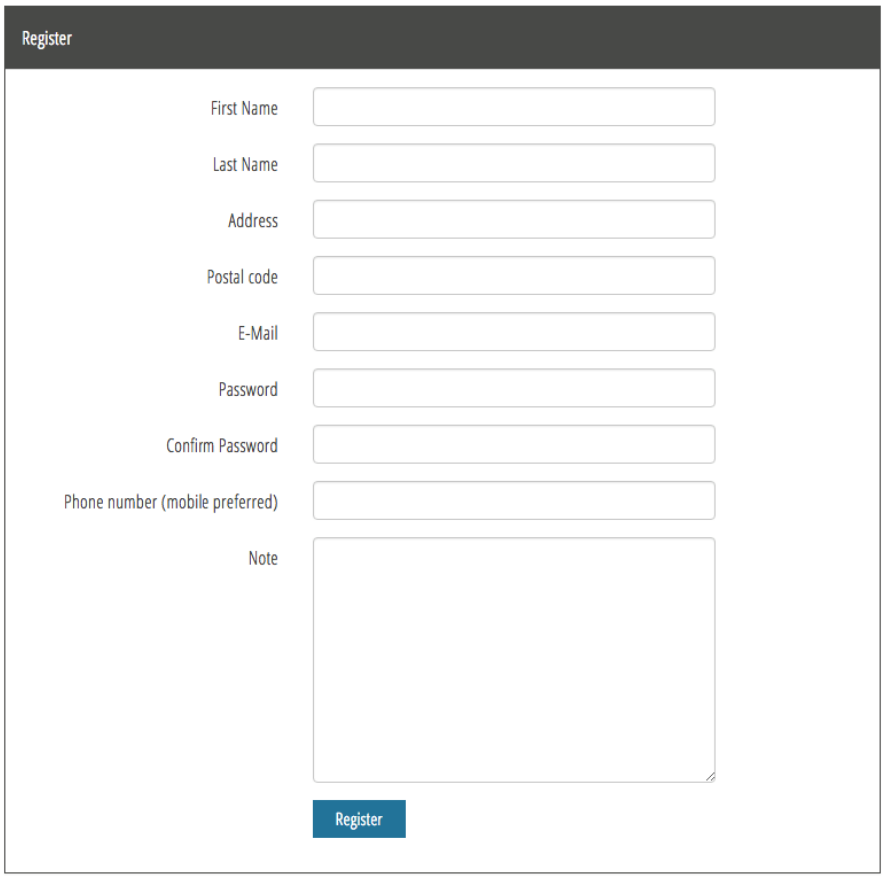
Primjer 2.3.5. Definiranje rute

3. FUNKCIONALNE KARAKTERISTIKE SUSTAVA

Sustav za iznajmljivanje automobila ima određene funkcionalne zahtjeve koje treba implementirati tijekom razvoja. U ovom poglavlju biti će navedene glavne funkcionalne karakteristike ovog sustava.

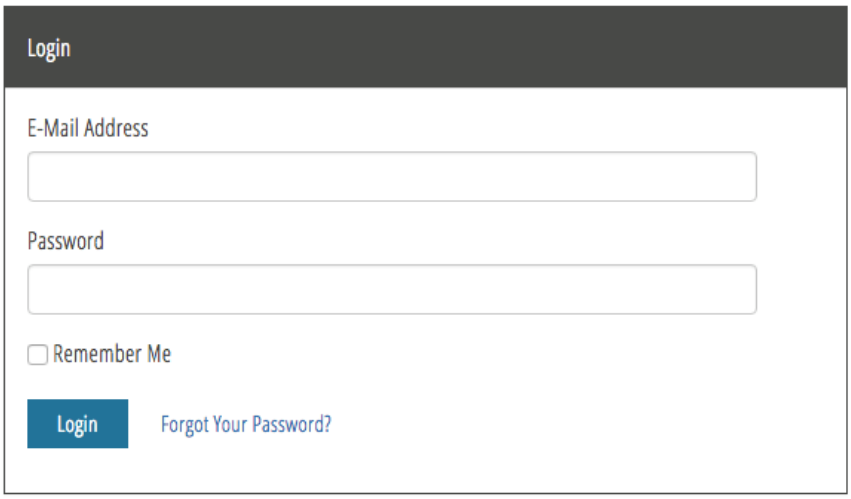
3.1. Korisnički sustav - registracija i prijava

Početne funkcionalnosti koje korisnički sustav mora imati je sustav registracije novih korisnika i sustav prijave postojećih korisnika. Na slici 3.1.1 je prikazan registracijski obrazac koji je početna točka korisničkog sustava. Nakon što korisnik unese tražene podatke, na poslužiteljovoj strani se izvodi validacija unesenih podataka, te ako podaci nisu dobro uneseni korisnika se vraća na stranicu za registraciju i naznačuje se koji podaci nisu dobro uneseni. Ovaj obrazac također ima polje za ponavljanje korisničke lozinke gdje korisnik mora dvaput unijeti istu lozinku da bi registracija bila uspješna. Korisnici koji se registriraju preko navedenog obrasca za registraciju dobivaju privilegije javnog korisnika, a ne administratora. Administratorski korisnik se registrira direktno na serveru.



Sl. 3.1.1. Obrazac za registraciju korisnika

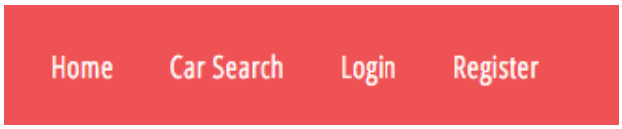
Nakon što je sustav registracije implementiran potrebno je implementirati sustav prijave korisnika. Korisnici koji su već registrirani mogu se prijaviti preko obrasca koji je prikazan na slici 3.1.2. Za ovaj obrazac za prijavu karakteristično je i to da ima zaštitu protiv napada na sustav. Zaštita je implementirana na taj način da je onemogućeno se neuspješno pokušati prijaviti u sustav više od 5 puta za redom u razdoblju od jedne minute. Ako se to dogodi, sustav blokira mogućnost prijave na taj korisnički račun na razdoblje od jedne minute. Ovo je vrlo korisno protiv *dictionary* napada na sustav, jer onemogućuje skripti koja pokreće napad da pokuša prijavu više od 5 puta u minuti. Sustav prijave također ima i mogućnost trajne prijave korisnika. Ako korisnik klikne "remember me" gumb, njegova prijava će biti postojana i nakon što korisnik zatvori prozor internetskog preglednika. Sustav prijave također ima i funkcionalnost koja omogućuje korisnicima koji su zaboravili svoju lozinku da povrate pristup sustavu.



Sl. 3.1.2. Obrazac za prijavu korisnika

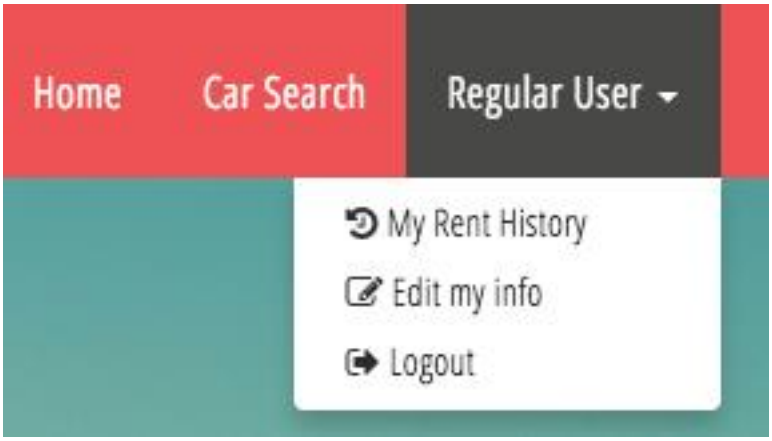
3.2. Sučelje ovisno o pravu pristupa

Korisnici u sustavu imaju dva moguća prava pristupa, tri uključujući ne registrirane korisnike. Tako smo korisnike podijelili na javne, obične registrirane i administratore. Sva tri prava pristupa imaju različita sučelja. Na slici 3.2.1 prikazane su opcije u izborniku za javne korisnike, na slici 3.2.2 prikazane su opcije izbornika za obične registrirane korisnike, a na slici 3.2.3 su prikazane opcije izbornika za administratore.



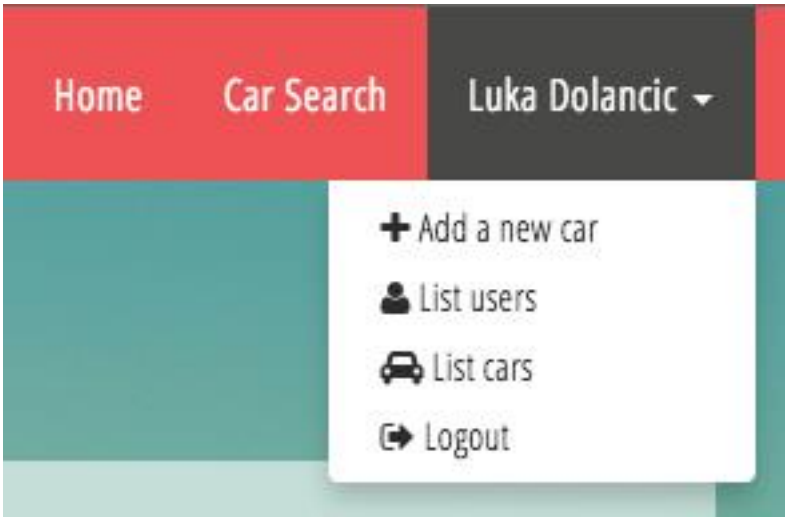
Sl. 3.2.1. opcije izbornika za javne korisnike

Na slici 3.2.2. u izborniku običnih registriranih korisnika postoje neke dodatne opcije. Oni imaju mogućnost pregleda njihove povijesti iznajmljivanja automobila kao i mogućnost promjene svojih osobnih informacija koje su unijeli prilikom registracije.



Sl. 3.2.2. opcije izbornika za obične registrirane korisnike

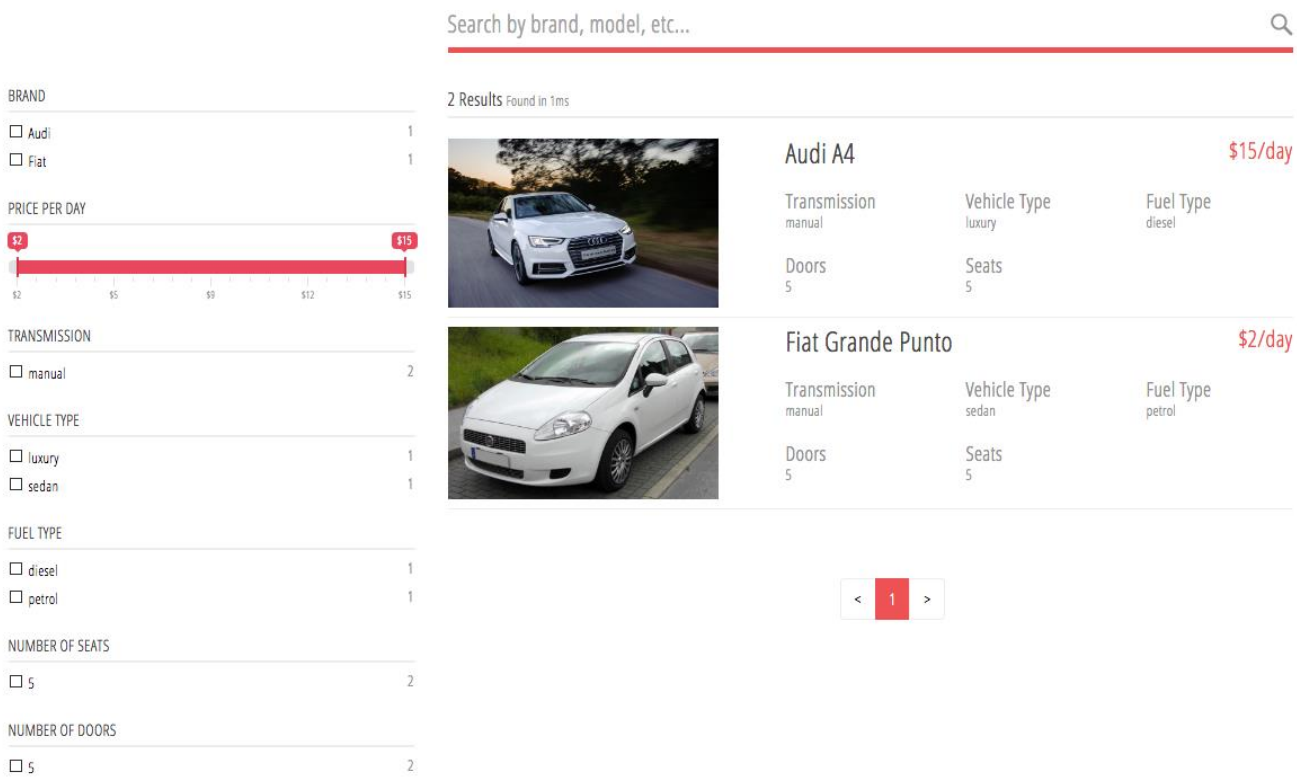
Na slici 3.2.3 prikazan je izbornik administratorskih korisnika. Ovaj tip korisnika ima neke opcije u izborniku koji drugi tipovi korisnika nemaju. Administratori imaju pristup sučelju za dodavanje novih automobila u sustav, pregled svih korisnika sustava i pregled svih automobila u sustavu.



Sl. 3.2.3. opcije izbornika za administratorske korisnike

3.3. Sustav pretrage ponude automobila

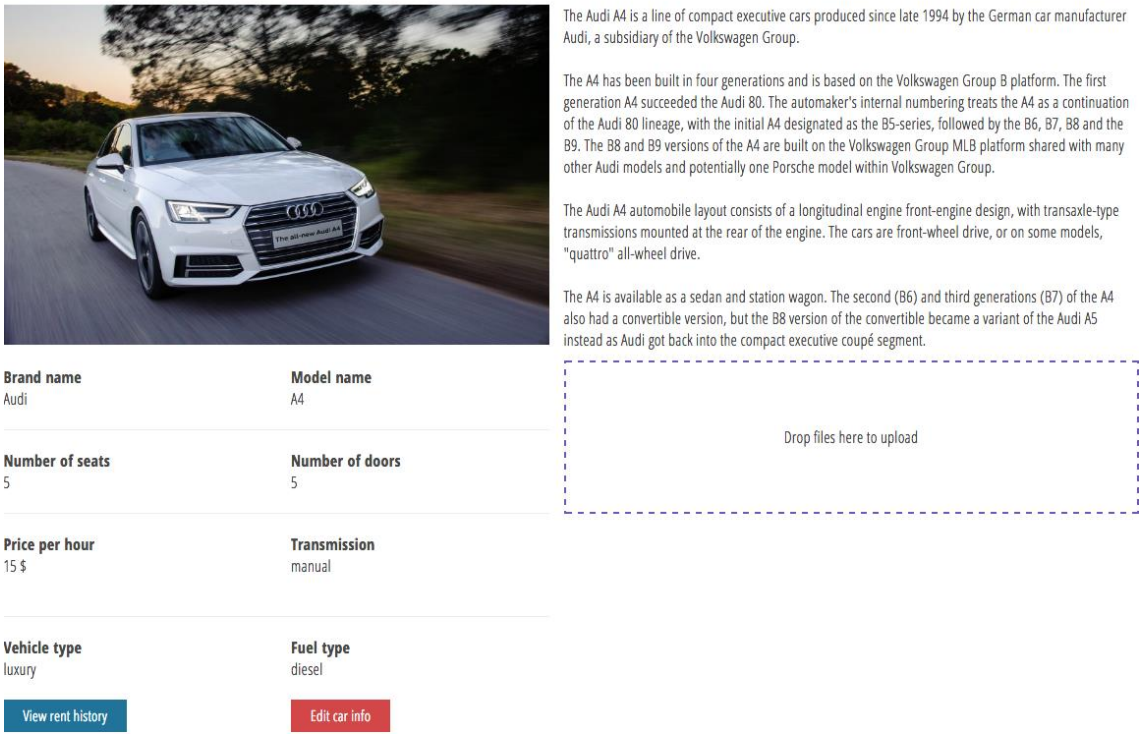
Za sustav za iznajmljivanje automobila vrlo je važno da ima dobar sustav pretrage ponude automobila koje naša tvrtka nudi. U ovom slučaju sustav pregleda i pretrage automobila implementiran je kao interaktivna tražilica sa velikim brojem mogućnosti. Postoji nekoliko načina pretrage ponude automobila koji ova tražilica nudi, a moguće je i kombinirati sve te načine. Jedan način pretrage je tekstovna pretraga. Sustav sadrži tekstovno polje u koje je moguće unijeti tražene pojmove. Mogući traženi pojam može biti bilo koji atribut automobila koji je prikazan na tražilici. Ovaj tekstovni način pretrage je također isprogramiran i tako da u određenoj granici tolerira pogreške u pisanju. Jedan od primjera toleriranja pogreške je ako korisnik želi tražiti automobil s nazivom "fiat grande punto", ali u tekstovno polje pogrešno upiše traženi pojam i napiše "fibat grende plunto", sustav prepoznaje da je to pogreška u pisanju i vraća nam kao rezultat automobil "fiat grande punto" (naravno s pretpostavkom da se automobil nalazi u bazi podataka). Sama tekstovna pretraga, kao i pretraga po filtrima, funkcionira tako da rezultate na okran prikazuje prilikom svake promjene unosa. To znači da korisnik ne mora stisnuti "pretraži" ili neki sličan gumb, nego prilikom upisa svakog slova radi se pretraga u sustavu i korisniku se u najkraćem mogućem roku rezultati prikazuju na ekranu. Ovaj sustav također ima i mogućnost dodatnog filtriranja rezultata po brendu i modelu automobila, cijeni, prijenosu, tipu vozila, tipu goriva koje automobil koristi, broju sjedala i broju vrata. Sustav pretrage također sadržava i paginaciju da bi se lakše moglo prikazati više rezultata pretrage. Sami rezultati pretrage prikazani su po redovima, gdje je za svaki rezultat prikazana slika automobila, njegov puni naziv, cijena po danu, te neke dodatne informacije kao što je broj vrata i sjedala. Samo sučelje tražilice je javno dostupno svim tipovima korisnika sustava. Sučelje tražilice je prikazano na slici 3.3.1.



Sl. 3.3.1. Sučelje tražilice

3.4. Pojedinačni pregled automobila

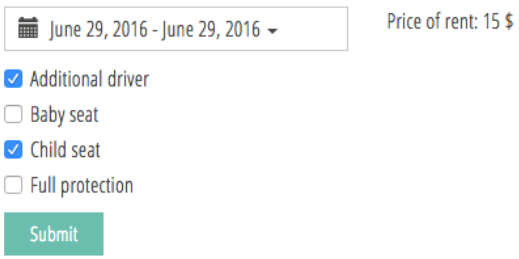
Nužno za ovakav sustav je pružiti sučelje za pojedinačni pregled automobila. Do samog sučelja se dolazi klikom na rezultat u tražilici opisanoj u prethodnom potpoglavlju. Ovo sučelje također ovisno o pravu pristupa korisnika različitim korisnicima pruža različita sučelja. Javni korisnici imaju pravo pogledati sve osnovne informacije automobila, ali nemaju pravo iznajmiti automobil, nego u svom sučelju imaju gumb koji ih navodi da se registriraju u sustav ako žele iznajmiti automobil. Obični prijavljeni korisnici u sučelju imaju gumb koji im omogućava da iznajme automobil. Administratori sustava u sučelju imaju neke dodatne mogućnosti. Jedna od tih mogućnosti je gumb koji vodi na stranicu za uređivanje informacija automobila. Druga mogućnost je pregled povijesti iznajmljivanja za navedeni automobil. Treća mogućnost je *dropzone*, područje pravokutnog oblika u koje administratorski korisnici mogu mišem dovući fotografije automobila iz svojeg datotečnog sustava. Te fotografije će se prikazivati na stranici automobila iznad glavnog opisa. Slika 3.4.1 prikazuje administratorsko sučelje pregleda stranice automobila.



Sl. 3.4.1. Sučelje za pregled automobila s administratorskim pravima

3.5. Sučelje za slanje zahtjeva za najam automobila

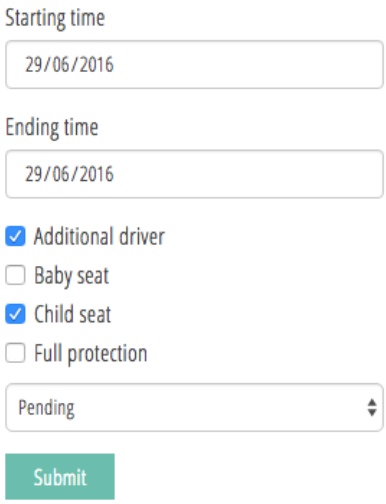
Obični registrirani korisnici imaju pristup sučelju za slanje zahtjeva za najam automobila. Do tog sučelja korisnici dolaze tako da kliknu na gumb za iznajmljivanje automobila u sučelju iz potpoglavlja 3.4. U sučelju za iznajmljivanje automobila korisnici imaju pristup interaktivnom kalendaru na kojem mogu odabrati u kojem vramenskom razdoblju žele iznajmiti automobil. Također ispod kalendara korisnici imaju ponuđene neke dodatne opcije kao što je osiguranje prilikom prometne nezgode itd. U ovom sučelju također se nalazi i kalkulator cijene najma. Cijena najma je prikazana korisniku i dinamički se mijenja kada korisnik promijeni neke od informacije u zahtjevu. Na slici 3.5.1 prikazano je sučelje za slanje zahtjeva najma.



Sl. 3.5.1. Sučelje za slanje zahtjeva najma

3.6. Sučelje za upravljanje najmom

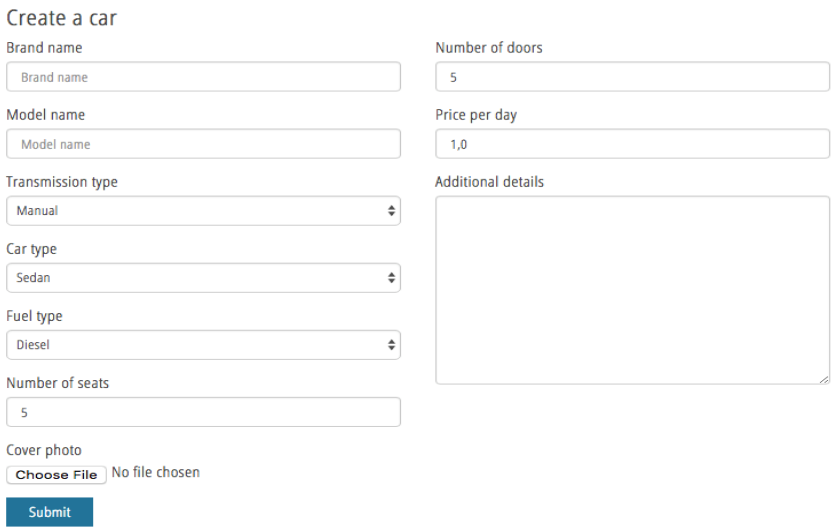
Administrator sustava ima pristup sučelju za upravljanje najmom. To sučelje omogućuje administratoru da mijenja informacije o najmu, ali najbitnija od mogućnosti koje ovo jednostavno sučelje pruža je mogućnost promjene stanja najma. Automobil se ne smatra iznajmljenim za određeni vramenski period za koji je korisnik poslao zahtjev sve dok administrator ne odobri taj zahtjev i ne promjeni status najma u "odobreno". Na slici 3.6.1 prikazan je obrazac za uređivanje zahtjeva najma s mogućnošću promjene statusa najma.



Sl. 3.6.1. Obrazac za uređivanje informacija o zahtjevu za najam

3.7. Sučelje za dodavanje automobila u sustav

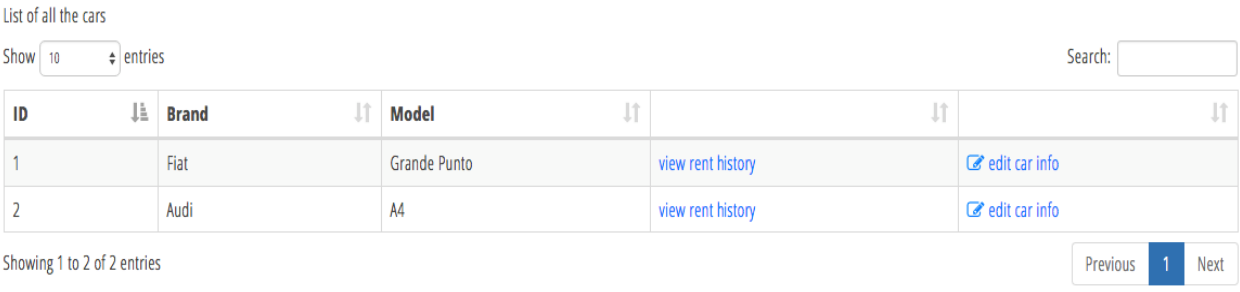
Administrator sustava također ima pristup sučelju za dodavanje novih automobila u sustav. U ovom sučelju, osim svih informacija vezanih za automobil, moguće je dodati i početnu fotografiju automobila koja se prikazuje na stranici automobila i na tražilici. Sučelje za dodavanje automobila prikazano je na slici 3.7.1.



Sl. 3.7.1. Sučelje za dodavanje novog automobila u sustav

3.8. Tablični prikaz podataka o automobilima, korisnicima i najmovima

Sustav sadrži mogućnost tabličnog prikaza podataka o automobilima, korisnicima i najmovima. Navedena mogućnost dostupna je samo administratorima sustava. U svakoj od navedenih tablica je i poveznica za uređivanje određenog tabličnog unosa, a sučelje za uređivanje se nalazi odvojeno od tabličnog prikaza. Primjer tabličnog prikaza automobila u bazi podataka sustava prikazan je na slici 3.8.1. Isto sučelje se koristi i za prikaz korisnika i najmova u sustavu, samo s različitim poljima od tablice automobila. U tabličnom prikazu automobila se također nalazi i poveznica za tablični prikaz svih najmova vezanih za taj automobil. Također, u tabličnom prikazu korisnika se nalazi poveznica za tablični pregled svih najmova pojedinog korisnika. U tabličnom prikazu najmova također za svaki najam postoji i link za pregled rute vožnje automobila za vrijeme navedenog najma.



Sl. 3.8.1. Tablični prikaz automobila u sustavu

3.9. Sustav praćenja rute automobila tijekom najma

U sustavu je definiran API koji omogućava unos geolokacijskih podataka u bazu podataka sustava. Određeni vanjski uređaj koji ima mogućnost očitavanja vlastite geolokacije i mogućnost slanja HTTP zahtjeva može podatke o lokaciji na taj način slati sustavu. Takav uređaj se može staviti u automobil i prilagoditi da u određenom vremenskom intervalu sustavu šalje geolokacijske podatke. U sustavu se nalazi sučelje u obliku karte svijeta koje za svaki pojedinačni najam automobila može na karti iscrtati rutu kojom se automobil kretao za vrijeme tog najma. Kvaliteta iscrtavanja rute na karti ovisi o intervalu slanja podataka sustavu od strane uređaja ugrađenog u automobil. Na slici 3.9.1 prikazano je sučelje na kojem se nalazi karta s iscrtanom rutom za određeni najam.

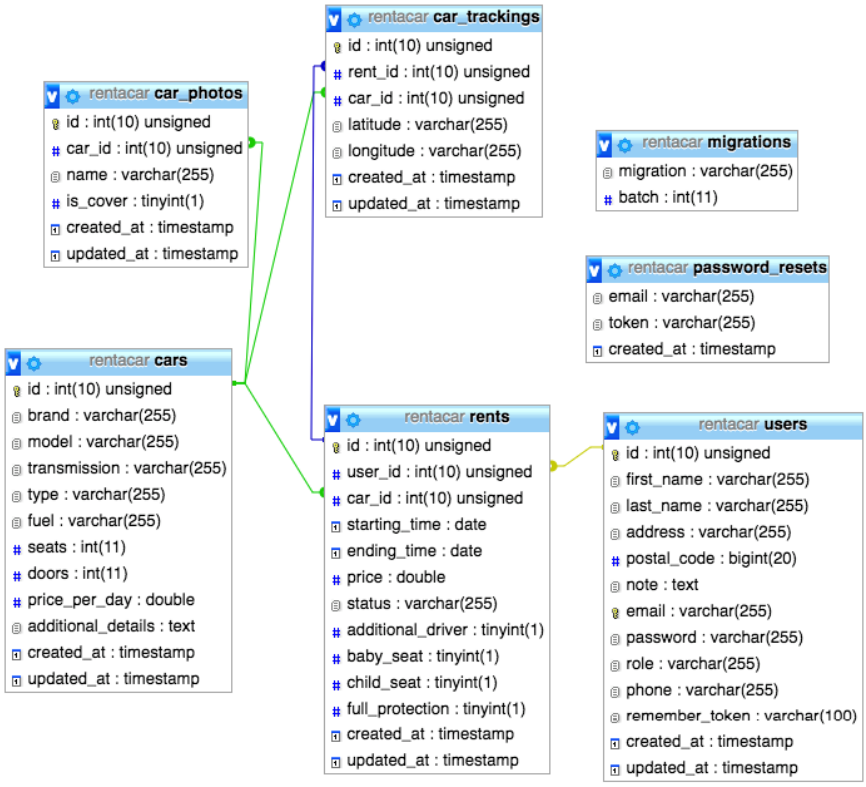
Sl. 3.9.1. Sučelje karte s iscrtanom rutom automobila za vrijeme najma

4. IMPLEMENTACIJA

U ovom poglavlju će biti više riječi o implementaciji funkcionalnosti kroz tehnologije navedene u poglavlju 2, o samoj strukturi koda, te arihtekturi aplikacije. Također će detaljnije biti opisano modeliranje baze podataka. Kompletna implementacija kroz kod bit će provedena kroz primjer sučelja za pohranu i pregled informacija o automobilu te modeliranje tablice za pohranu informacija automobila u bazi podataka.

4.1. Modeliranje baze podataka

Laravel platforma za lakše upravljanje i komunikaciju s bazom podatak koristi Eloquent ORM. Samo modeliranje baze podataka se također izvodi na objektno orijentirani način kroz migracije. Migracije omogućuju da se i sama shema baze podataka nalazi unutar SCM (na engleskom *Source Code Management*) sustava što je vrlo poželjno prilikom izrade projekta. Sama baza podataka organizirana je u nekoliko tablica čiji je dizajn prikazan na slici 4.1.1. Baza podataka sadrži i *migrations* tablicu koja ima ulogu unutar same Laravel platforme. Prilikom modeliranja baze podataka stvaraju se tablice unutar baze, kao i relacije između različitih tablica po potrebi. Na slici 4.1.1 prikazane su sve tablice koje se nalaze unutar sustava, kao i relacije između tablica.



Sl. 4.1.1. Dizajn baze podataka

Migracije unutar Laravel platforme pružaju objektno orijentirano sučelje za stvaranje entiteta i relacija u bazi podataka. U primjeru 4.1.2 prikazan je izvorni kod migracije za stvaranje tablice koja sadrži podatke o automobilu. U navedenom primjeru je prikazano stvaranje tablice *cars* pomoću Laravelovih migracija kroz objektno orijentirano sučelje. Prikazana je mogućnost Laravelovih migracija da stvaraju novu tablicu te u njoj definiraju koje polje će imati koji tip podatka.

```
<?php
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateCarsTable extends Migration
{
    public function up()
    {
        Schema::create('cars', function (Blueprint $table) {
            $table->increments('id');
            $table->string('brand');
            $table->string('model');
            $table->string('transmission');
            $table->string('type');
            $table->string('fuel');
            $table->integer('seats');
            $table->integer('doors');
            $table->double('price_per_day');
            $table->text('additional_details');
            $table->timestamps();
        });
    }

  public function down()
  {
    Schema::drop('cars');
  }
}
```

Primjer 4.1.2. Migracija za stvaranje tablice automobila

Kada se kreira ovakva migracijska klasa unutar izvornog koda projekta, potrebno je također pokrenuti i naredbu unutar laravelove aplikacije koja će ovakvu definiciju tablice iz migracije preslikati u MySql naredbe za stvaranje tablice i njenih relacija. Rezultat preslikavanja iz migracije u MySql naredbe je prikazan u primjeru 4.1.3.

```
CREATE TABLE `cars` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `brand` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `model` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `transmission` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `type` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `fuel` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `seats` int(11) NOT NULL,
  `doors` int(11) NOT NULL,
  `price_per_day` double NOT NULL,
  `additional_details` text COLLATE utf8_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;
```

Primjer 4.1.3. MySql kod koji je rezultat preslikavanja migracije

4.2. Upravljač

Upravljači u sustavu imaju ulogu povezivana izvornog koda modela s izvornim kodom pogleda. U primjeru 4.2.1 prikazan je upravljač koji se koristi prilikom obrade obrasca za unos informacija o novom automobilu u sustav. U navedenom primjeru kroz objektno orijentirano sučelje se primaju podaci poslani iz obrasca kroz $request objekt. Na temelju primljenih podataka stvara se objekt tipa automobil te se njegove informacije spremaju u bazu podataka. Iz obrasca se također pokušava procesirati i učitavanje fotografije automobila. Ako je u obrascu priložena fotografija automobila, ona se pohranjuje u datotečni sustav, te se njen naziv sprema u bazu podataka radi kasnijeg referenciranja. Nakon toga obavlja se preusmjeravanje na stranicu za pregled informacija o automobilu.

```
public function store(CarRequest $request)
{
    $car = Car::create($request->all());

    $car->processImageUpload($request, true);

    $car->pushToIndex();

    return redirect('/car/' . $car->id);
}
```

Primjer 4.2.1. Upravljač za obradu podataka prilikom unosa informacija automobila

4.3. Pogled

Pogled unutar Laravel platforme ima ulogu prezentacije podataka prema krajnjem korisniku. U primjeru 4.3.1 prikazan je pogled koji sadrži obrazac za unos podataka o novom automobilu. Radi količine izvornog koda u navedenom pogledu izostavljeni su HTML elementi koji služe samo u svrhu dizajna, a prikazani su samo elementi pogleda koji imaju funkcionalnu ulogu u obrascu. Obrazac unesene podatke šalje na rutu *car.store* koja je definirana u primjeru 4.2.1. Također forma sadržava i *csrf token* (na engleskom *cross site request forgery*) koji ima sigurnosnu ulogu u sustavu.

```
@extends('layouts.main')
@section('content')
    <form method="post" action="{{ route('car.store') }}"
enctype="multipart/form-data">
        {{ csrf_field() }}
        <input type="text" class="form-control" id="brand" name="brand"
placeholder="Brand name" required>
        <input type="text" class="form-control" id="model" name="model"
placeholder="Model name" required>
        <select name="transmission" id="transmission" class="form-control">
            <option value="manual" selected>Manual</option>
            <option value="automatic">Automatic</option>
        </select>
        <select name="type" id="type" class="form-control">
            <option value="sedan" selected>Sedan</option>
            <option value="MPV">MPV</option>
            <option value="SUV">SUV</option>
            <option value="luxury">Luxury</option>
        </select>
        <select name="fuel" id="fuel" class="form-control">
            <option value="diesel" selected>Diesel</option>
            <option value="petrol">Petrol</option>
        </select>
        <input type="number" class="form-control" id="seats" name="seats"
min="2" max="10" value="5">
        <input type="number" class="form-control" id="doors" name="doors"
min="2" max="7" value="5">
        <input type="number" class="form-control" id="price_per_day"
name="price_per_day" min="1.0" step="0.1"
               value="1.0">
        <textarea name="additional_details" cols="30" rows="10"
class="form-control"></textarea>
        <input type="file" name="photo" id="photo" class="file">
        <button type="submit" class="btn btn-primary">Submit</button>
    </form>
@stop
```

Primjer 4.3.1. Pogled koji sadrži obrazac za unos podatka o automobilu

5. ZAKLJUČAK

Tvrtke čija je djelatnost iznajmljivanje automobila imaju sve veću potrebu za računalnim sustavima koji olakšavaju upravljanje evidencijom podataka unutar tvrtke i pružanjem javnog sučelja korisnicima. Ovakav sustav ima nekoliko vrlo važnih funkcionalnih zahtjeva da bi se korisnicima mogla olakšati odluka o najmu automobila. Prije svega, korisnici ne trebaju biti registrirani da bi imali pristup podacima automobila, kao i cijenama najma za određeni automobi. Samim time se korisnicima olakšava korištenje sustava. Glavna namjena sustava je praćenje zahtjeva za najam koji registrirani korisnici mogu poslati. Registrirani korisnici su podijeljeni u dvije skupine: obični korisnici i administratori. Administratori sustava imaju mogućnost pregleda svih informacija u sustavu, a jedna od najvažnijih mogućnosti koju administratori imaju je potvrda zahtjeva za najam poslanih od strane korisnika. Administratori imaju sučelje s prikazanim informacijama o zahtjevu za najam, te na temelju toga i informacija korisnika koji je poslao zahtjev oni mogu najam odobriti, odbiti ili staviti u još neki drugi status ovisno o kontekstu situacije. Sustav je izveden pomoću raznih web tehnologija koje olakšavaju razvoj ovakvog sustava. Pored standardnih web tehnologija kao što su HTML, CSS, Javascript, PHP i MySql, u ovom sustavu se koristi i Laravel platforma. Laravel platforma je platforma izrađena na PHP programskom jeziku i koristi MVC (na engleskom *Model-View-Controller*) oblikovni obrazac. Ova platforma pri razvoju sustava pruža već unaprijed definiranu arhitekturu aplikacije, jednostavan razvoj korisničkog sustava, sustav objektno orijentirane apstrakcije nad bazom podataka, te mnoge druge funkcionalnosti koje olakšavaju razvoj ovako kompleksnog sustava.

Sam sustav je u konačnici uspješno implementiran kroz navedene tehnologije sa svim potrebnim funkcionalnostima. Sustav krajnjim korisnicima pruža brz i jednostavan način pregleda automobila i slanja zahtjeva za najam, a administratorima pruža jednostavna sučelja za nadzorom na informacijama u sustavu. Ovakav sustav se može koristiti u osnovnom scenariju tvrtke čija je djelatnost iznajmljivanje automobila.

LITERATURA

[1] Josh Lockhard, Modern PHP, prvo izdanje , O'Reilly, 2015

[2] Chuck Heintzelman, Laravel 5.1 Beauty: Creating Beautiful Web Apps in Laravel 5.1, Taschenbuch 2015

[3] Jon Duckett, HTML and CSS: Design and Build Websites, 2011

[4] David Flanagan, JavaScript: The Definitive Guide (Definitive Guides), 2011

[5] Laravel 5.2 dokumentacija, https://laravel.com/docs/5.2

[6] HTML, https://hr.wikipedia.org/wiki/HTML

[7] CSS, https://hr.wikipedia.org/wiki/CSS

[8] PHP, https://hr.wikipedia.org/wiki/PHP

SAŽETAK

Web aplikacija za rent-a-car tvrtku

Cilj ovog završnog rada je izrada sustava za iznajmljivanje automobila koji olakšava rad tvrtke koja ga koristi. Jedna od osnovnih zahtjeva je korisnicima pružiti jednostavan i intuitivan način pretraživanja i pregleda automobila koji su ponuđeni od strane tvrtke, te sučelje za slanje zahtjeva za najam. Također postoji nekoliko zahtjeva koje tijeko razvoja treba ispuniti prema administratorima samog sustava. Jedna od najvažnijih mogućnosti je dodavanje novih automobila, te potvrda zahtjeva za najam koje šalju korisnici. Također administrator sustava ima sučelja za upravljanje svim podacima unutar same baze podataka. Na kraju, u sustavu postoji i mogućnost praćenja iznajmljenog automobila gdje je omogućeno učitavanje podataka geolokacije s uređaja ugrađenog u automobil, te iscrtavanje informacija o lokaciji na karti svijeta unutar administratorskog sučelja. Izvedba ovakvog sustava nije nimalo jednostavan zadatak te zahtjeva modeliranje baze podataka i implementaciju kroz nekoliko programskih, skriptnih i stilskih jezika.

Ključne riječi: tvrtka za iznajmljivanje automobila, korisnici, zahtjev za najmom, sustav za praćenje automobila, geolokacija, baza podataka

ABSTRACT

Web application for a rent-a-car company

The goal of this paper is to develop a rent a car system that would simplify everyday tasks of a company using it. One of the main requirements is to provide users with a simple and intuitive interface for searching and viewing cars offered by the company and interface for submiting a rent request. Also, there are some requirements towards administrator users. One of the most important features is adding new cars to the system and the ability to confirm a rent request sent by a user. Administrator of the system is also provided with an interface for managing all the informations stored in a database. In the end, aplication also contains a tracking system for tracking rented cars. It is possible to send a geolocation from a device inside a car. In the end, that information is rendered on a map inside an administrator interface. Developing a system like this is not a simple task and it requires modelling a database and implementing functionalities via multiple programming, scripting and styling languages.

Key words: rent a car shop, users, rent request, car tracking system, geolocation, database

ŽIVOTOPIS

Luka Dolančić rođen je 1.2.1993. godine u Zagrebu. Tijekom 1992. godine njegova obitelj se seli u mjesto Sveti Đurađ. Tamo pohađa osnovnu školu, a 2000. godine se seli u Podravske Podgajce gdje nastavlja pohađati osnovnu školu. Nakon osnovne škole pohađa Opću gimnaziju Donji miholjac. Nakon završene srednje škole 2011. godine upisuje studij informatike na Elektrotehničkom fakultetu u Osijeku.

Potpis studenta

PRILOZI

U prilogu se nalaze materijali koji nisu neophodni za temeljito praćenje materije u završnom radu i rješenja zadatka završnog rada

P.1. CD

Sadržaj CD-a:

- primjerak završnog rada u .docx formatu
- primjerak završnog rada u pdf formatu

P.2. Web lokacija izvornog koda

Izvorni kod zadatka ovog završnog rada nalazi se na web lokaciji:
https: //github.com/ldolancic/rent-a-car