Upotreba Zend alata u izradi web aplikacije za zajedničko korištenje vozila

Ribarić, Matko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

UPOTREBA ZEND ALATA U IZRADI WEB APLIKACIJE ZA ZAJEDNIČKO KORIŠTENJE VOZILA

Diplomski rad

Matko Ribarić

Osijek, 2018. godina.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 20.09.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Matko Ribarić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D-533R, 22.09.2017.
OIB studenta:	09477069193
Mentor:	Izv. prof. dr. sc. Krešimir Nenadić
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Tomislav Keser
Član Povjerenstva:	Dr. sc. Tomislav Galba
Naslov diplomskog rada:	Upotreba Zend alata u izradi web aplikacije za zajedničko korištenje vozila
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	Opisati alat Zend kojim će se izraditi web aplikacija pomoću koje će korisnici moći dogovarati dijeljenje vozila za prijevoz na posao. Dogovaranje omogućiti samo registriranim korisnicima koji će na osnovu mjesta s kojeg kreću moći dogovarati prijevoz s drugim korisnicima. U aplikaciji omogućiti korisnicima zadavanje mjesta i vremena prijevoza.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	20.09.2018.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 03.10.2018.

Ime i prezime studenta:	Matko Ribarić
Studij:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D-533R, 22.09.2017.
Ephorus podudaranje [%]:	5%

Ovom izjavom izjavljujem da je rad pod nazivom**: Upotreba Zend alata u izradi web aplikacije za zajedničko korištenje vozila**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Krešimir Nenadić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD

Ubrzanim razvojem Internet aplikacija pojavila se potreba stvaranjem okvira (eng. framework) putem kojega bi se pojednostavilo kreiranje Internet aplikacija. Tijekom zadnjeg desetljeća klasičan način izrade Internet aplikacija temeljio se na HTML ili PHP skriptnom jeziku. Svaki dio sadržaja se posebno kodirao i postavljao na poslužitelje. Takav tip izrade Internet aplikacija se pokazao problematičnim, prvenstveno zbog razloga kompatibilnosti. Pojavom većeg broja preglednika pojavio se i problem kompatibilnosti. U vrijeme kada je Internet Explorer preuzeo vodstvo među internet preglednicima, većina aplikacija je pisana kako bi se ispravno prikazivala na njemu, dok se za ostale preglednike nije testirala funkcionalnost. Pojavom novih preglednika (Mozilla Firefox, Opera te posebice Google Chrome) te tržišne konkurentnosti postaje sve važnije da su aplikacije kompabitilne i potpuno funkcionalne na svim preglednicima.

Izuzev kompatibilnosti, veliki problem Internet aplikacija je održavanje. Nelogična arhitektura te loše struktuiran kod neki su od problema s kojima se programeri susreću prilikom ispravljanja grešaka ili dodavanja novih funkcionalnosti. Zbog toga je logičan slijed bio izrada alata koji bi pratio dosljednost korištenja programskih jezika i biblioteka (eng. library).

U ovome radu je obrađen alat koji u zadnje vrijeme postaje primarni alat sve većem broju programera. Drugo poglavlje obrađuje nastanak, povijest i značajke stavke te mogućnosti Zend programskog alata. U trećem poglavlju je prikazano osnovno modeliranje internet aplikacije kao primjer koje sve mogućnosti pruža Zend programski alat. U četvrtom poglavlju su objašnjeni glavni dijelovi arhitekture internet aplikacije te proces izrade aplikacije. Peto poglavlje se tiče testiranja aplikacije na raznim inačicama internet preglednika i hardverskih konfiguracija računala. Šesto poglavlje donosi zaključak diplomskog rada.

2. POVIJEST, ZNAČAJKE I PREDNOSTI ZEND OKVIRA

Kako se povećao broj skriptnih jezika za izradu Internet aplikacija pojavila se potreba za kreiranje programskog okvira koji bi objedinio mogućnosti različitih skriptnih jezika na jednom mjestu te olakšao izradu Internet aplikacija.

2.1. POVIJEST ZEND OKVIRA

Dvije godine nakon prve verzije PHP-a, dva izraelska progamera Zeev Suravski i Andi Gutmans su počeli razvijati osnovu za PHP inačicu 3 gdje su redizajnirali postojeći PHP rasčlanjivač (eng. parser). PHP (akronim PHP: Hypertext Preprocessor) je skriptni jezik na strani poslužitelja koji može biti implementiran u HTML ili se koristi u kombinaciji sa drugim web sistemskim predlošcima. Naziv Zend je kombinacija njihovih prezimena. Iduća iteracija PHP 4 se temeljila na prvoj inačici Zend Engine-a. Zend Engine je skriptni mehanizam koji prevodi PHP skriptni jezik. Napisan je u C programskom jeziku kao visoko optimizirani modul, koji se po prvi puta mogao koristiti izvan PHP jezika. Zend Engine pruža upravljanje memorijom i resursima, te ostale standardne servise za PHP jezik. Trenutna inačica je 3.0 koja se temelji na PHP 7 skriptnom jeziku. Ohrabreni uspjehom 1999. godine su formalno osnovali tvrtku Zend Technologies te počeli na radu Zend Engine-a iz kojega će proizaći Zend Framework (skraćeno: ZF, u daljnjem tekstu Zend okvir).

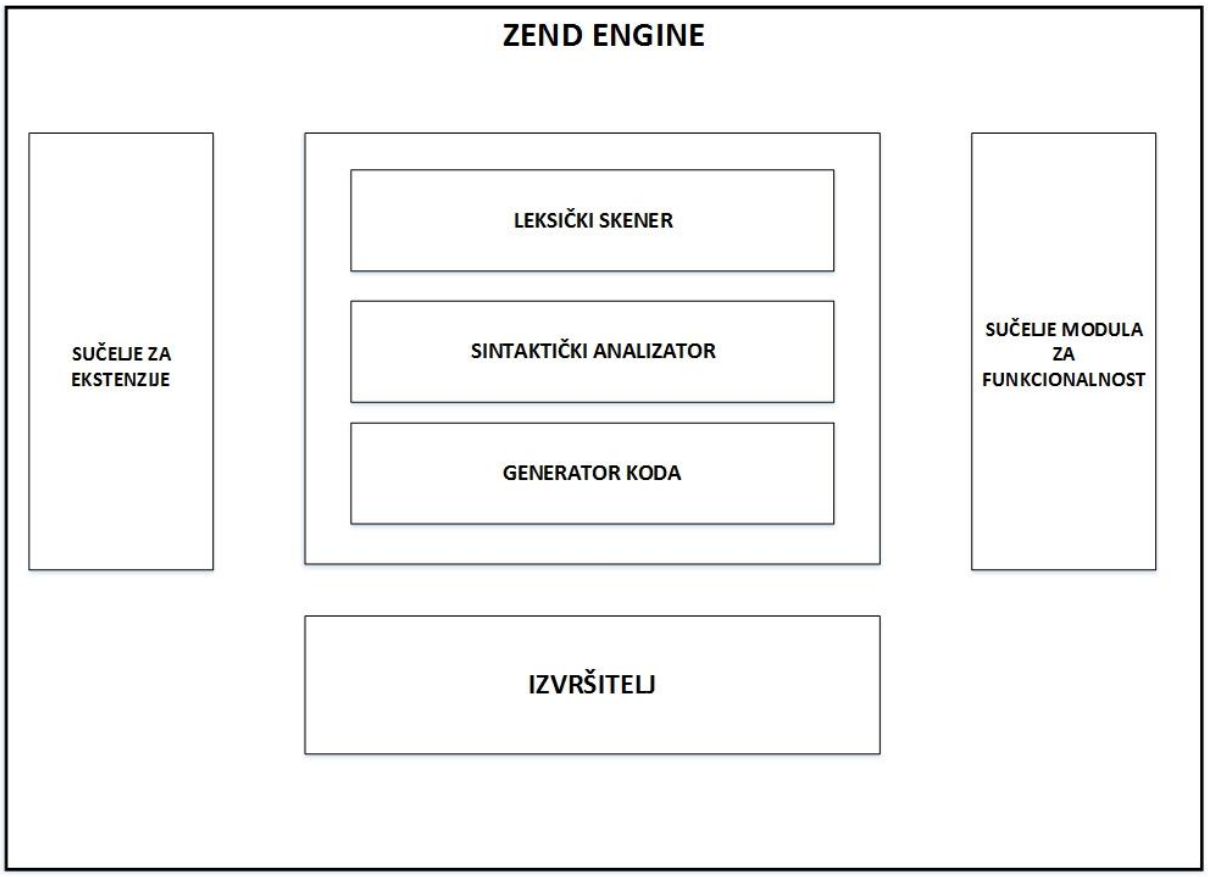
Prva inačica Zend okvira pojavila se u ožujku 2006. godine, nakon godina razvoja Zend Enginea i bila je temeljena na PHP5 skriptnom jeziku. Šestog kolovoza 2010. godine izdana je inačica 2.0 temeljena na PHP 5.3 inačici programskoj jezika.[1] Sve do rujna 2012. godine okvir je bio u razvojnoj fazi. Tek 5. rujna 2012. godine izdana je prva stabilna inačica Zend okvira u inačici 2.0

Trenutna inačica u trenutku izrade diplomskog rada je 3.0.0 izdana 3. ožujka 2016. godine

2.2. ZNAČAJKE ZEND PROGRAMSKOG OKVIRA I ZEND ENGINE-A

Osnovna arhitektura Zend Enginea se temelji na korištenju prevoditelja (eng. compiler) i sustava za vrijeme izvršenja (eng. runtime engine). PHP skripte se učitavaju u memoriju i prevode kao Zend instrukcije (eng. Zend opcodes). Nakon što se instrukcije izvrše, generirani HTML kod se šalje klijentu.

Kako bi uspješno implementirali prevoditelj Internet skripti potrebna je podijeliti posao na tri dijela. Prvi dio se sastoji od analize dolaznog koda, prevođenja u razumljiv format te samog izvršavanja koda koje obavlja prevoditelj. Drugi korak je taj da programski dio prevoditelja zadužen za funkcionalnost implementira funkcije specifične za skriptni jezik u kojemu je pisana skripta. Zadnji korak se sastoji od komunikacije prevoditelja s internet poslužiteljima. Zend engine je zadužen za prvi i dio drugog dijela odrade, dok PHP skriptni jezik je zadužen za izvršenje u cijelosti drugog i trećeg koraka.



Sl. 2.1. *Zend Engine arhitektura.*

Na slici 2.1 prikazana je osnovna arhitektura Zend Enginea. Vidljivo je kako Sučelje za ekstenzije predaje kod na analizu koji se sastoji od leksičkog skenera, sintaktitčkog analizatora i generatora koda. Nakon što se kod izgenerirao, te je prošao kroz sučelje modula za funkcionalnost, dolazi do faze izvršitelja koji predaje gotov kod internet poslužitelju.

2.3. PREDNOSTI ZEND PROGRAMSKOG OKRUŽENJA

Zend Framework je samo jedan u nizu programskih okvira koji koriste PHP jezik kao svoju bazu. Uz Zend postoji nekoliko konkurentskih programskih okvira koji također u zadnjih nekoliko godina dobivaju sve veću pozornost programera. Zend je plaćeni okvir zbog čega su većinom korisnici tvrtke koje si licence mogu priuštiti. Većina progamera koji rade sami bira besplatne inačice ovakvog alata. Prema anketi provedenoj na stranici *Sitepoint.com* Zend je tek osmi programski okvir po popularnosti. Generalno se uzima da je Zend jedan od nekoliko „velikih" programskih okvira koje danas progameri koriste za razvoj Internet aplikacija. Ostali popularni programski okviri su:

- Laravel
- Symfony
- Phalcon
- Yii
- Slim
- CakePHP
- FuelPHP
- Aura
- Codeigniter

Glavna prednost Zend programskog okruženja je programska podrška. Iz razloga što ga uglavnom koriste tvrtke, Zend pruža 0-24h programsku podršku. Još jedna velika prednost je zajednica (eng. community) programera koji uvijek imaju spremne savjete i primjere kako odraditi određeni posao koristeći Zend programsko okruženje.

S programske strane svakako treba istaknuti modularnost programskog okruženja i mogućnost korištenja ekstenzija. Uz veliku sigurnost Zend se postavio kao jedan od vodećih izbora za profesionalce. Većina ostalih programskih okruženja navedenih gore je jednostavnija i pristupačnija novim programerima, ali nemaju jednaku programsku podršku niti istu razinu stabilnosti programskih okruženja.

3. MODELIRANJE APLIKACIJE

Kako bi uspješno izradili aplikaciju potrebno je prvo kvalitetno izraditi stukturnu analizu koja uključuje analizu i modeliranje sustava. Modeliranje je proces gdje se utvrđuju elementi sustava i veza između pojedinih elemenata. Povezanost elemenatana sustava se prikazuje vizualno putem tehnika grafičkog prikaza. Modeliraju se arhitektura, logička i fizička struktura te sami procesi unutar sustava. Modeliranjem se povećava kvaliteta, stabilnost i funkcionalnosti aplikacije prema [2].

Modeliranje se grafički prikazuje korištenjem UML dijagrama. UML dijagram je dijagram temeljen na Unified Modeling Language programskog jeziku sa svhrom vizualne reprezentacije sustava sa svim aktorima, ulogama, akcijama, artifaktima i klasama. Koristi se kako bi bolje razumjeli, promijenili, održavali i dokumentirali inforacmije o sustavu. Na taj način se olakšava prepoznavanje najvažnijih slučajeva korištenja. Za izradu internet aplikacije koristit će se sljedeći dijagrami:

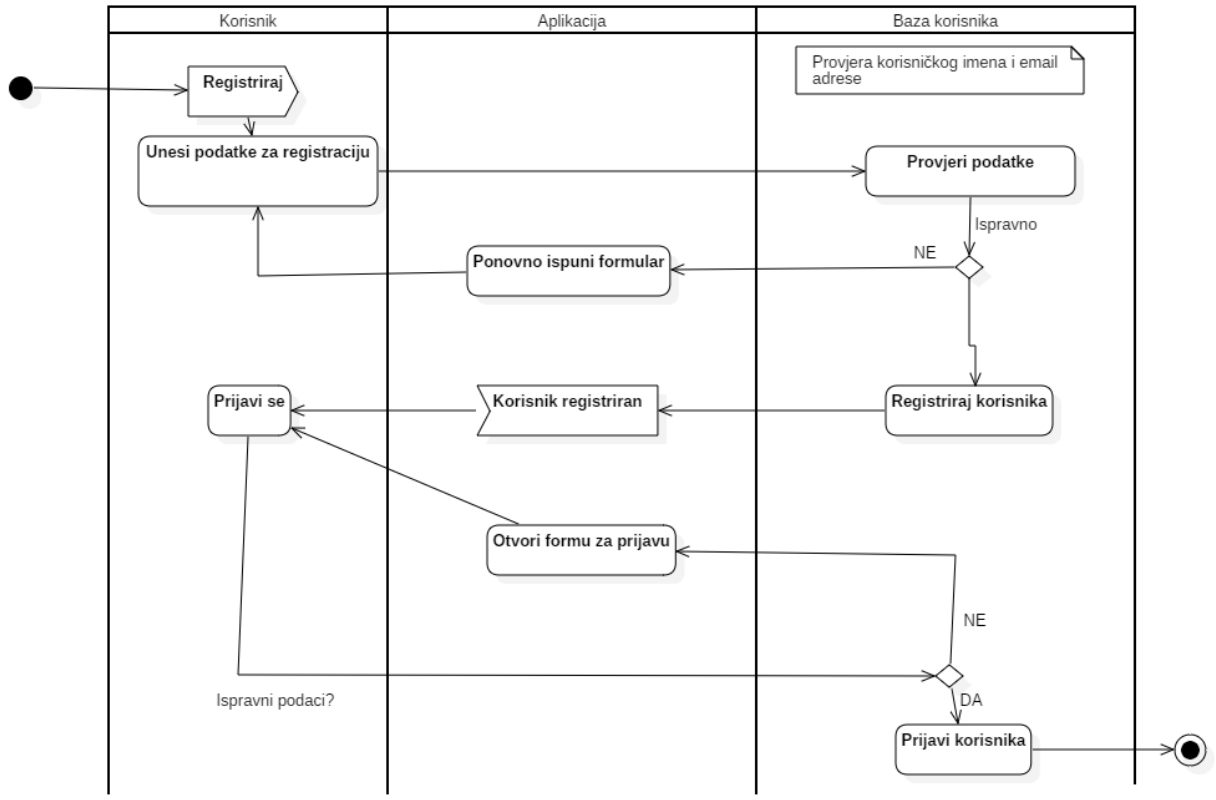
- Dijagram aktivnosti
- Dijagram raspoređivanja
- Klasni dijagram
- Dijagram slučaja korištenja

3.1. DIJAGRAM AKTIVNOSTI

Dijagram aktivnosti sastoji se od 3 glavna aktera koji su prikazani u tzv. plivajućim trakama (eng. swimlanes): korisnik, Internet aplikacija i baza korisnika. Početni događaj prikazuje se s lijeve strane početne trake i on započinje kada korisnik pokrene registraciju. Tijek kretanja aktivnosti označen je strijelicom, dok se aktivnost aktera označava elipsama. Korisnik upisuje podatke za registraciju. Podaci se šalju u bazu podataka gdje se provjerava je li korisničko ime ili adresa elektroničke pošte već upisano u bazu. Ukoliko već postoji podudarnost aplikacija ponovno nudi obrazac za registraciju i taj se postupak ponavlja dokle god postoji podudarnost podataka koje je korisnik upisao prilikom registracije s onima već unesenima u bazu podataka. Kada korisnik unese podatke koji nisu upisani u bazu podataka, završava se registracija te se unešeni podaci spremaju u bazu podataka. Nakon što je dobio elektroničku poruku o potvrdi uspješnosti registracije korisnik ima mogućnost prijave na sustav. Korisnik

upisuje korisničko ime i zaporku, te se ono ponovno provjerava s podacima upisanim u bazi podataka.

Ukoliko je neispravno korisniku se ponovno nudi obrazac za prijavu, te se proces ponavlja dokle god se korisnički podaci ne podudaraju s onima upisanima u bazi podataka. Nakon uspješne prijave u sustav cijeli proces je završen te se kraj označava sa točkom na desnoj strani dijagrama.



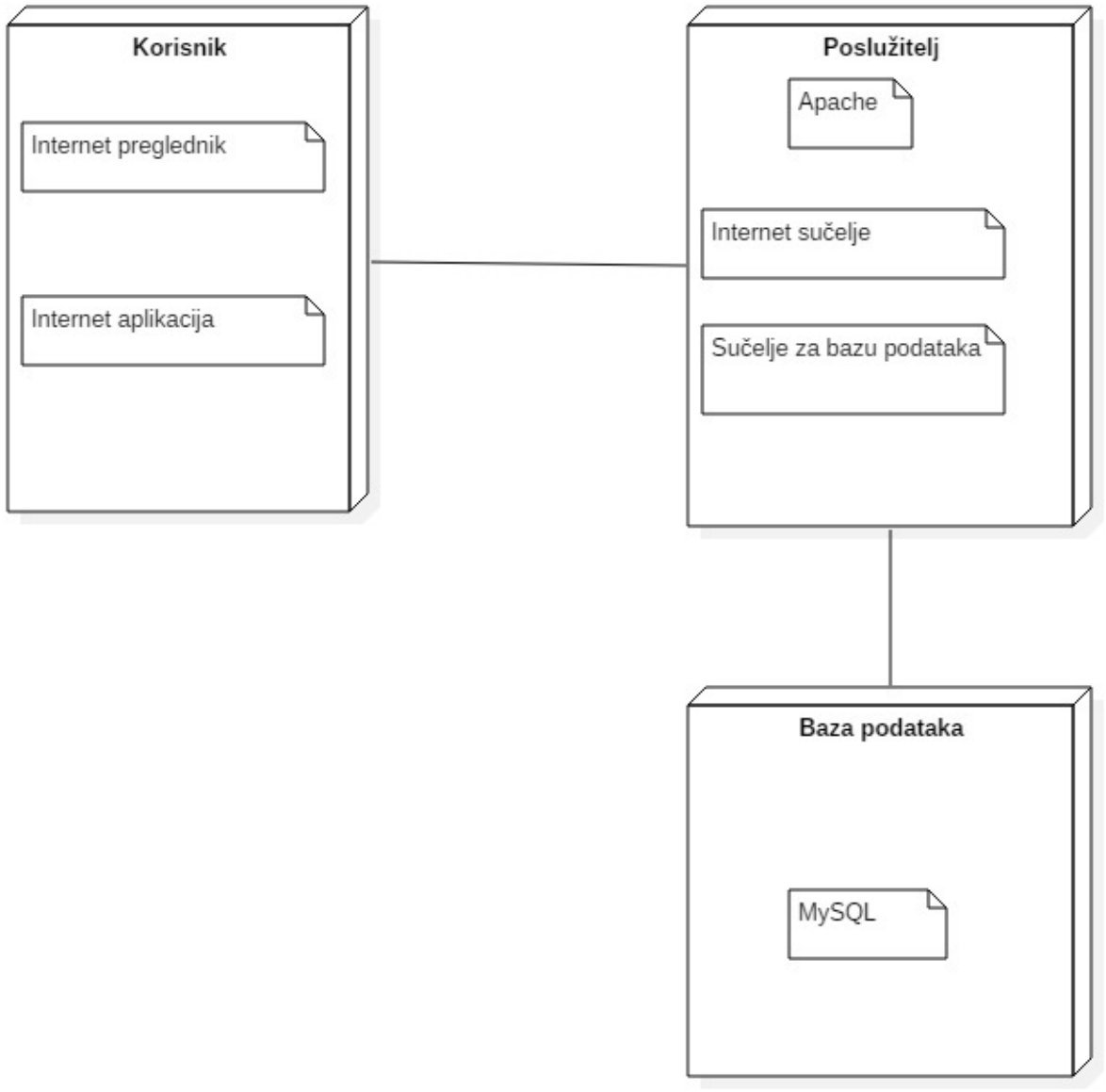
Sl. 3.1. Dijagram aktivnosti

3.2. DIJAGRAM RASPOREĐIVANJA

Dijagram raspoređivanja prikazuje arhitekturu sustava. Arhitektura sustava sastoji se od čvora korisnika koji sadržava Internet aplikaciju i preglednik koji je nužan kako bi korisnik mogao pristupiti aplikaciji. Internet aplikaciju održava progamer koji koristi PhPStorm kako bi ažurirao aplikaciju. I jedan i drugi čvor prikazuje računalo korisnika i administratora.

Čvor Internet poslužitelja, koji je ujedno i razvojna okolina aplikacije sadrži internet sučelje te sučelje baze podataka. Prema slici 3.2 čvor poslužitelja je označen kao SN (eng. *Server node*).

Za primjer demonstrirana rada ove aplikacije koristi se Apache poslužitelj. Sama aplikacija se preko poslužitelja povezuje s MySQL bazom podataka, te se koristi komunikacijska veza temeljena na TCP/IP protokolu.



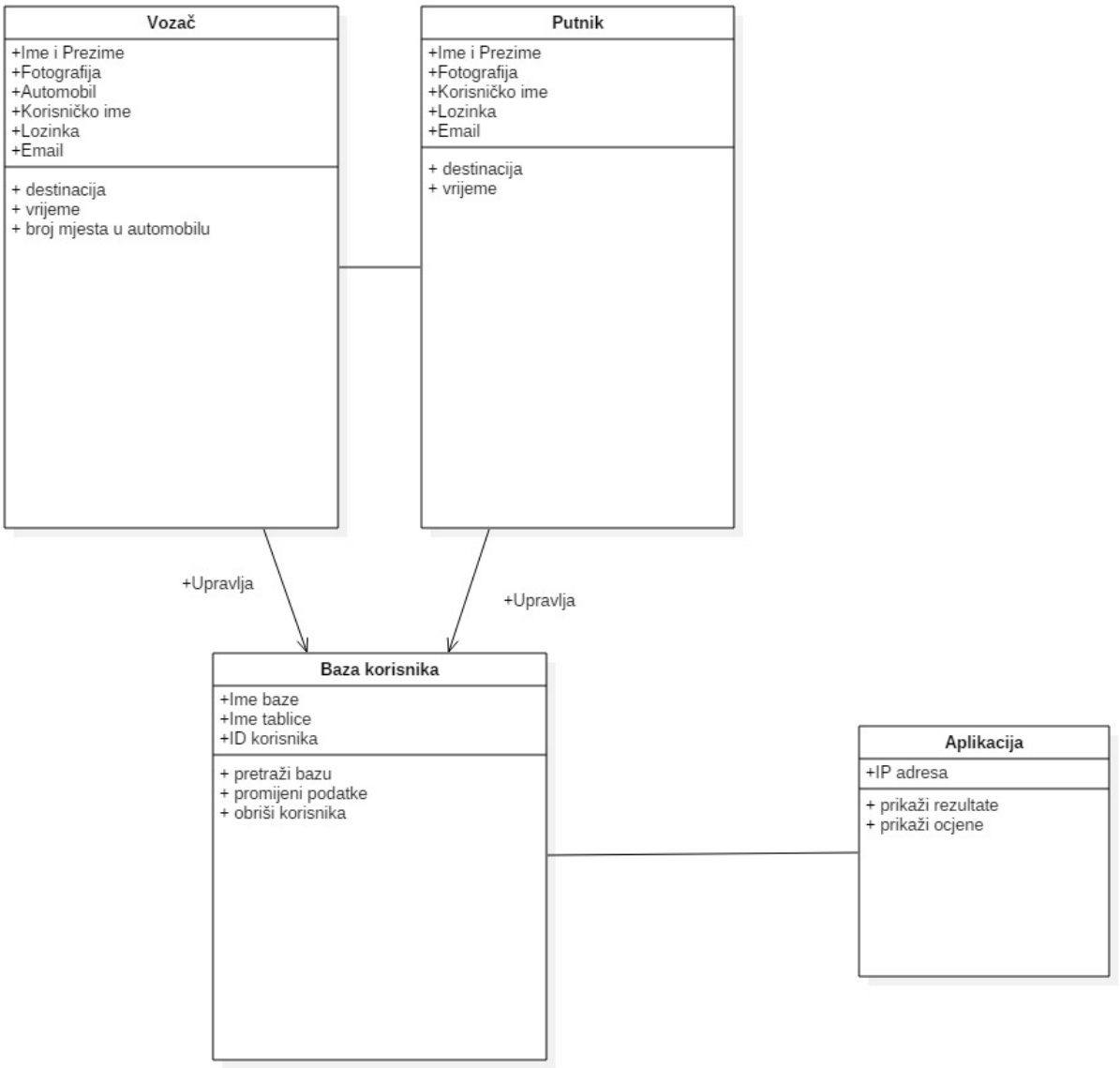
Sl. 3.2. Dijagram raspoređivanja.

3.3. KLASNI DIJAGRAM

Klasni dijagram se koristi za vizualizaciju i spajanje elemenata statičkog dijela sustava, dokumentiranje, opis funkcionalnosti te izradu izvršnog koda aplikacije. Dijagram prikazuje klase sustava, odnosno instance objekata koji sudjeluju u procesu. Opisuje klase modela, sadržava atribute, operacije i odnose između klasa koji su važni za sustav Internet aplikacije.

Za glavnog sudionika Internet aplikacije koja se izrađuje postavljen je registrirani korisnik. U ovom primjeru razliku ju se dvije klase registiranih korisnika, odnosno vozač i putnik.

Svaka klasa ima svoje specifičnosti i različite načine korištenja internet aplikacije. Unutar baze podataka se nalaze podaci o svakom korisniku, te se na osnovu tih podataka može iščitati kojoj klasi pripada registrirani korisnik.



Sl. 3.3. Klasni dijagram.

Klasni dijagram sadrži 4 glavne klase: vozač, putnik, aplikacija i baza podataka. Kao što je vidljivo na slici 3.3 klase vozač i putnik imaju iste ovlasti i mogućnosti upisa u bazu podataka, dok sama aplikacija rezultate traži unutar baze podataka i prikazuje ovisno o pojmu pretrage.

3.4. DIJAGRAM SLUČAJA KORIŠTENJA

Dijagramom slučaja korištenja modeliraju se korisnički zahtjevi i slučajevi ispitivanja sustava. Prilikom korištenja metode oblikovanja programske potpore zasnovane na slučajevima korištenja potrebno je iz njih izvesti strukturne te modele ponašanja sustava. Poželjno je osigurati konzistetnost gore navedenih modela. Svaki slučaj korištenja mora biti razumljiv programerima kako bi se pravilno mogli kreirati ostali dijagrami.

Prilikom definiranja zahtjeva naručitelj i programer, ako se radi o dvije različite osobe moraju se dogovoriti i raspraviti oko slučajeva korištenja te scenarijima u kojima se opisuje tijek određenih akcija. Slučajevi korištenja (eng. use cases) su preuzeti iz UML standarda. Temelje se na ideji mogućih scenarija, a opisuju moguće međusobne interakcije sustava. Slučaj korištenja opisuje funkcionalnost sustava kroz slučajeve razumljive korisnicima i programerima. Temeljni elementi u modelima obrazaca uporabe su: akteri, slučajevi korištenja i relacije između aktera. Detalji se još mogu dodatno prezentirati tekstom ili dodatnim dijagramima interakcije.

Naziv: Registracija korisnika

Inicijator: korisnik

Cilj: kreirati profil na internet aplikaciji

1. Aplikacija nudi obrazac za registraciju
2. Korisnik ispunjava podatke
3. Baza podataka provjerava podatke
4. Korisnik se registrira, tj. kreira profil

Dodaci:

3. Baza pronalazi korisničko ime ili e-mail u bazi

a) aplikacija zahtjeva ponovnu registraciju

b) povratak na korak 2.

3. Zaporka ne odgovara predefiniranim uvjetima

a) aplikacija zahtjeva ponovnu registraciju

b) povratak na korak 2.

Naziv: Prijava u sustav

Inicijator: Korisnik

Cilj: Prijaviti se u internet aplikaciju

1. Aplikacija nudi korisniku prijavu
2. Korisnik upisuje podatke
3. Baza podataka vrši provjeru podataka
4. Baza korisnika potvrđuje prijavu

Dodaci:

2. Baza podataka ne nalazi korisničko ime

a) Aplikacija nudi ponovnu prijavu korisnika

b) Ponavlja se korak 2.

4. Baza podataka ne može potvrditi zaporku

a) Aplikacija nudi ponovni upis zaporke

b) Ponavlja se korak 2

4. Baza podataka ne može potvrditi zaporke

a) Aplikacija nudi obrazac za promjenu zaporke

b) Korisnik upisuje e-mail s kojim se registirao

c) Dostavlja mu se poveznica za promjenu zaporke na adresu elektroničke pošte

d) Ponavlja se korak 2

Naziv: Promjena korisničkih podataka

Inicijator: Korisnik

Cilj: Promjena postojećih podataka u profilu

1. Korisnik se prijavi
2. Korisnik odabire izmjena podataka
3. Korisnik unosi promjene
4. Baza podataka ažurira podatke

Dodaci:

4. Korisnički podaci su istovjetni onima u bazi

    a) Vrati se na korak 3

Naziv: Traženje putnika

Inicijator: Korisnik

Cilj: Pronaći putnike za vožnju

1. Korisnik se prijavi u sustav
2. Korisnik upisuje traženu destinaciju
3. Aplikacija prikazuje sve destinacije koji odgovaraju pojmu pretrage
4. Korisnik odabire željenog putnika
5. Aplikaciju omogućuje kontakt prema putniku

Dodaci:

3. Nema korisnika koji odgovara pojmu pretrage

    a) Aplikacija javlja da nema nijedna destinacija koja odgovara parametrima pretrage

    b) Vrati se na korak 2

Naziv: Traženje vozača

Inicijator: Korisnik

Cilj: Pronaći vozača za vožnju

1. Korisnik se prijavi u sustav
2. Korisnik upisuje traženu destinaciju
3. Aplikacija prikazuje sve destinacije koji odgovaraju pojmu pretrage
4. Korisnik odabire željenog vozača
5. Aplikaciju omogućuje kontakt prema putniku

Dodaci:

3. Nema korisnika koji odgovara pojmu pretrage

a) Aplikacija javlja da nema nijedan korisnik koji zadovoljava uvjete

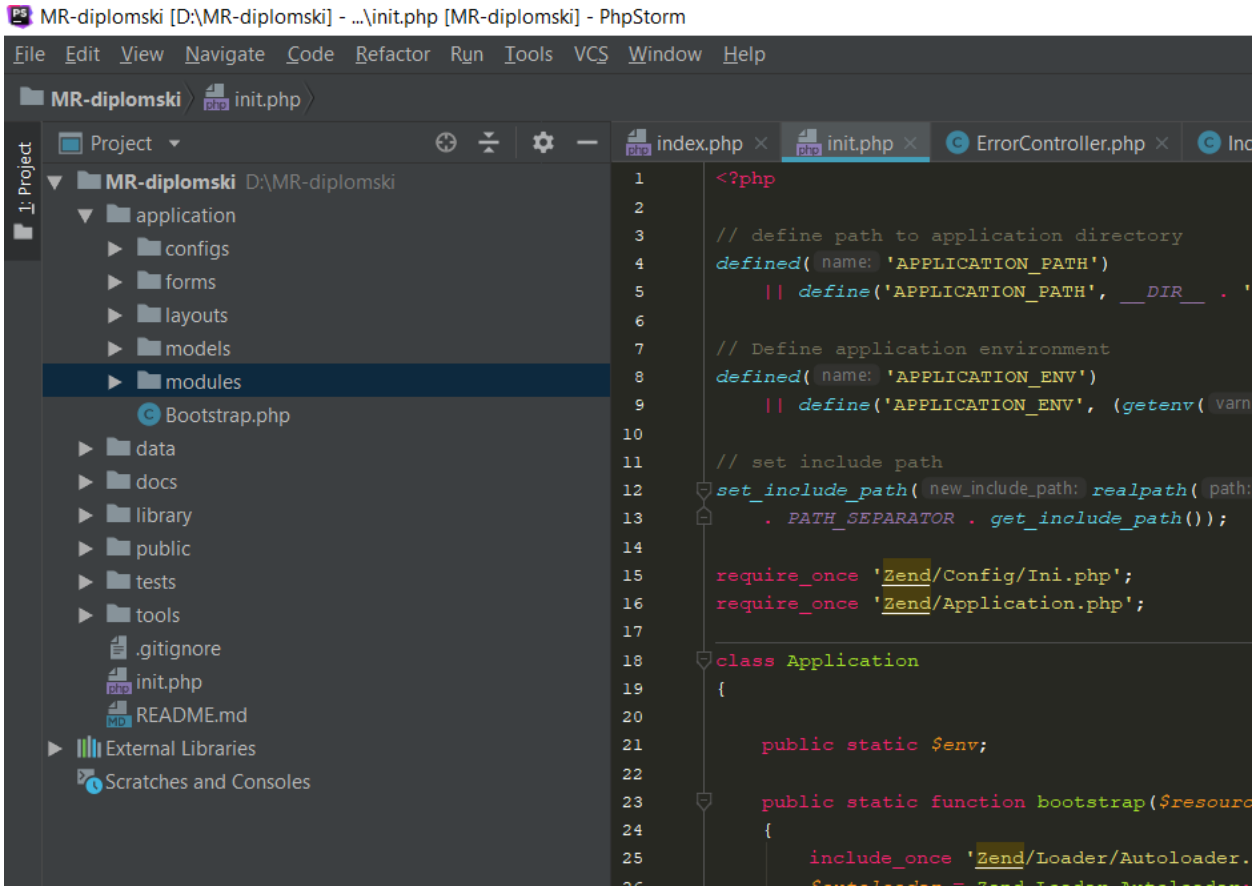
b) Vrati se na korak 2

4. IZRADA APLIKACIJE

Najbolji način prezentiranja mogućnosti Zend programskog okvira je izrada Internet aplikacije koristeći alat temeljen na Zend programskom okviru. Za temu Internet aplikacije odabrana je situacija gdje korisnici koji žele dijeliti troškove putovanja na posao mogu pronaći vozača ili suvozače s kojim bi dijelili troškove. Porastom troškova prijevoza pojavila se potreba da potpuni stranci koji idu na iste destinacije dijele troškove prijevoza. Tema je odabrana kako bi se pokazale mogućnosti Zend programskog okvira, ali postoji mogućnost daljnjeg razvoja kroz razne module i nadogradnje iz razloga što takav oblik prijevoza postaje sve privlačniji korisnicima koji traže jeftiniji prijevoz s većim komforom. Internet aplikacija će ponuditi mogućnost registracije korisnika, podešavanja vlastitih profila, te pretragu baze registriranih korisnika ovisno o terminima i relaciji na kojima je prijevoz potreban.

4.1. OPIS PROGRAMSKOG OKRUŽENJA KORIŠTENOG ZA IZRADU APLIKACIJE

Izrada aplikacije će se raditi u JetBrains PHPStorm aplikaciji. PHPStorm je komercijalno integrirano razvojno okružje stvoreno na JetBrains IntelilJ IDEA platformi. Koristi se kao editor za PHP, HTML te JavaScript. Velika prednost je mogućnost analize u stvarnom vremenu, prevencija grešaka te automatsko refaktoriranje za PHP i JavaScript programski kod. PHPStorm podržava PHP inačice od 5.3 do 7.2. Iako je PHPStorm komercijalno razvojno okružje, JetBrains pruža studentima besplatno korištenje prvih godinu dana. U trenutku izrade ovog rada zadnja dostupna inačica je 2.1 koja je izašla na tržište tijekom kolovoza 2018. godine.



Sl. 4.1. JetBrains PhpStorm sučelje.

Kako se Internet aplikacija radi lokalno, potrebno je lokalno kreirati poslužitelj i bazu podataka na kojoj će se aplikacija pokretati. Za tu svrhu je najbolje koristiti XAMPP. XAMPP je besplatno i multi platformsko rješenje za kreiranje lokalnog poslužitelja. XAMPP je akronim za:

X – multiplatformsko (eng. „X as cross platform“)

A - Apache

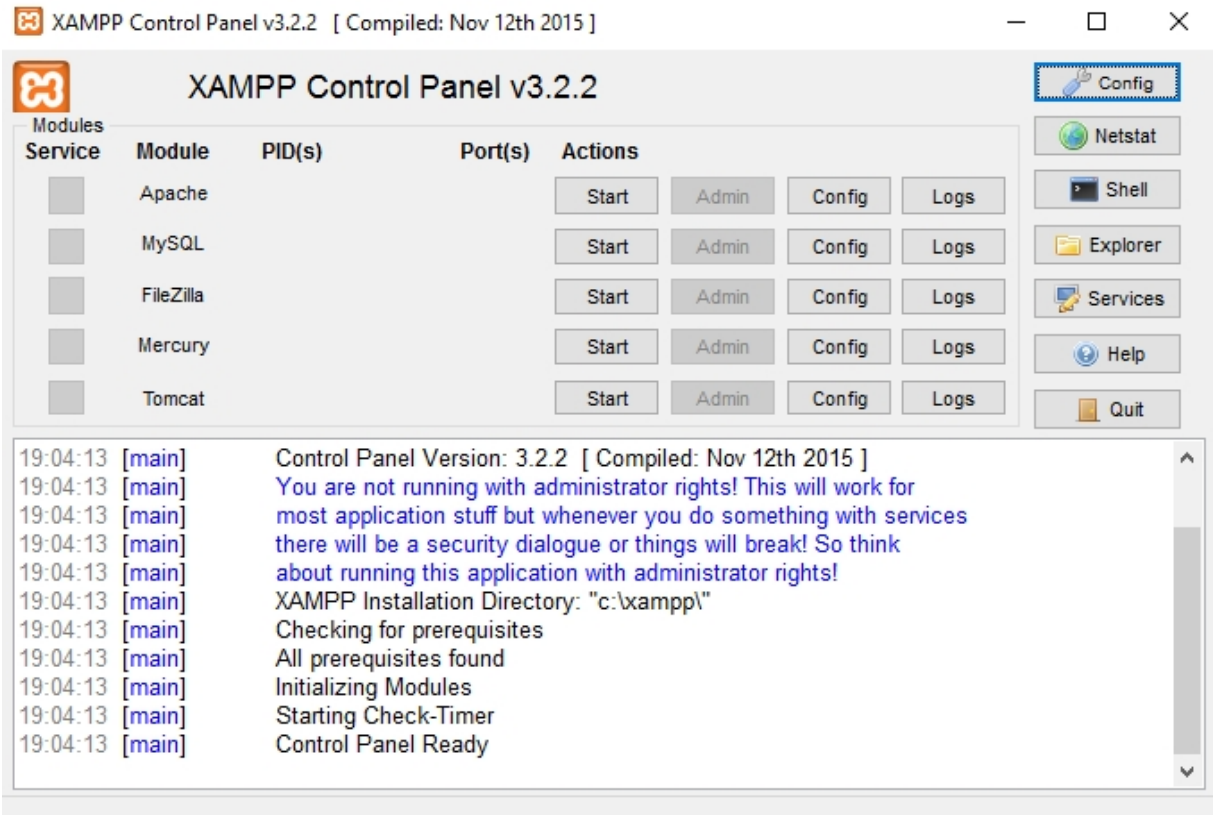
M – MariaDB (MySQL)

P - PHP

P -Perl

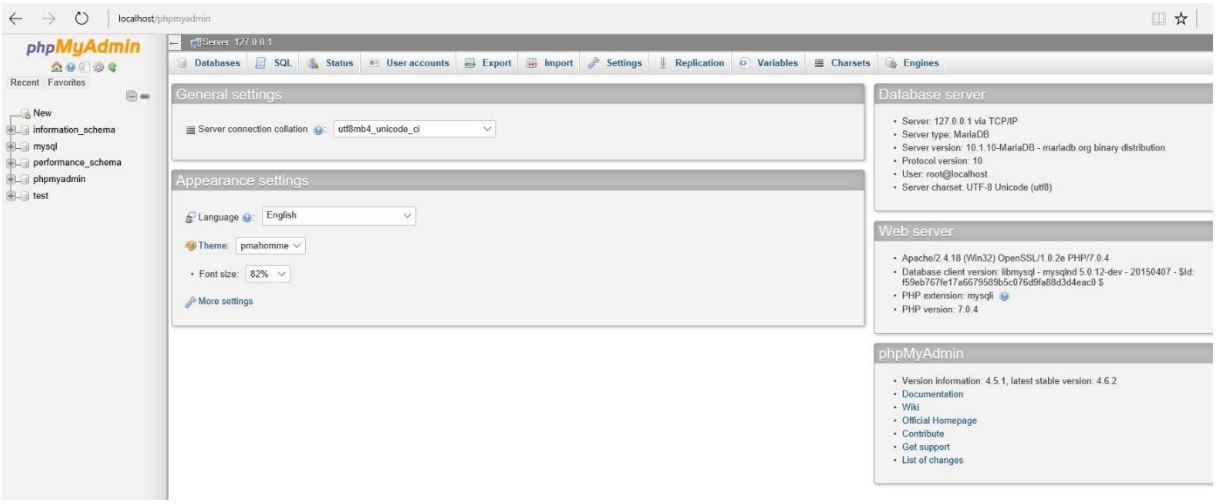
XAMPP je trenutno u inačici 7.0.6 i konstatno se razvija. XAMPP uvelike olaškava posao programerima koji se bave razvojem Internet aplikacija omogućavajući im lokalno podizanje web poslužitelja u fazi izrade aplikacije.

Tako programer može razvijati i testirati aplikaciju koristeći više tehnologija, bez da se mora brinuti podržava li poslužitelj i Internetski prostor koji je zakupio sve tehnologije koje se koriste prilikom izrade programa. Uz to bitna stavka je i visoka cijena poslužitelja koji podržava više različitih tehnologija. Na ovaj način programer ima mogućnost kreirati aplikaciju te nakon toga zakupiti Internet poslužitelj koji ima podršku za tehnologije koje je programer koristio, što uvelike smanjuje cijenu zakupa internetskog prostora.



Sl. 4.2. XAMPP kontrolna ploča.

Na slici 4.2 je vidljivo koji servisi se mogu omogućiti prilikom pokretanja XAMPP kontrolne ploče. Za izradu aplikacije koristimo module Apache i MySQL. Apache će nam omogućiti postavljanje aplikacije na virtualni lokalni poslužitelj, dok će MySQL modul osigurati bazu podataka. Baza podataka se podešava preko phpMyAdmin kontrolne ploče kojom se pristupa pomoću poveznice http://localhost/phpmyadmin/.



Sl. 4.3. phpMyAdmin kontrolna ploča.

Kako bi omogućili lokalno podizanje poslužitelja potrebno je u datoteku hosts koja se nalazi u direktoriju C://Windows/System32/drivers/etc dodati mogućnost da se internet adresi pristupa lokalno. To se radi dodavanjem adrese 127.0.0.1 te lokalnom domenom, što je u ovom slučaju diplomski.local.

```
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost

127.0.0.1               diplomski.local
```

Sl. 4.4. hosts datoteka

4.2. IZRADA GRAFIČKOG SUČELJA APLIKACIJE

Prvi dio izrade aplikacije odnosi se na izradu grafičkog sučelja. Izrada grafičkog sučelja se radi kombinacijom HTML, PHP i posebice CSS-a. Prilikom izrade grafičkog dijela potrebno je paziti da se Internet aplikacija uredno prikazuje na što većem broju internet preglednika, te da brzina učitavanja aplikacije bude dovoljno brza i optimizirana za različite načine i brzine pristupa internetu.

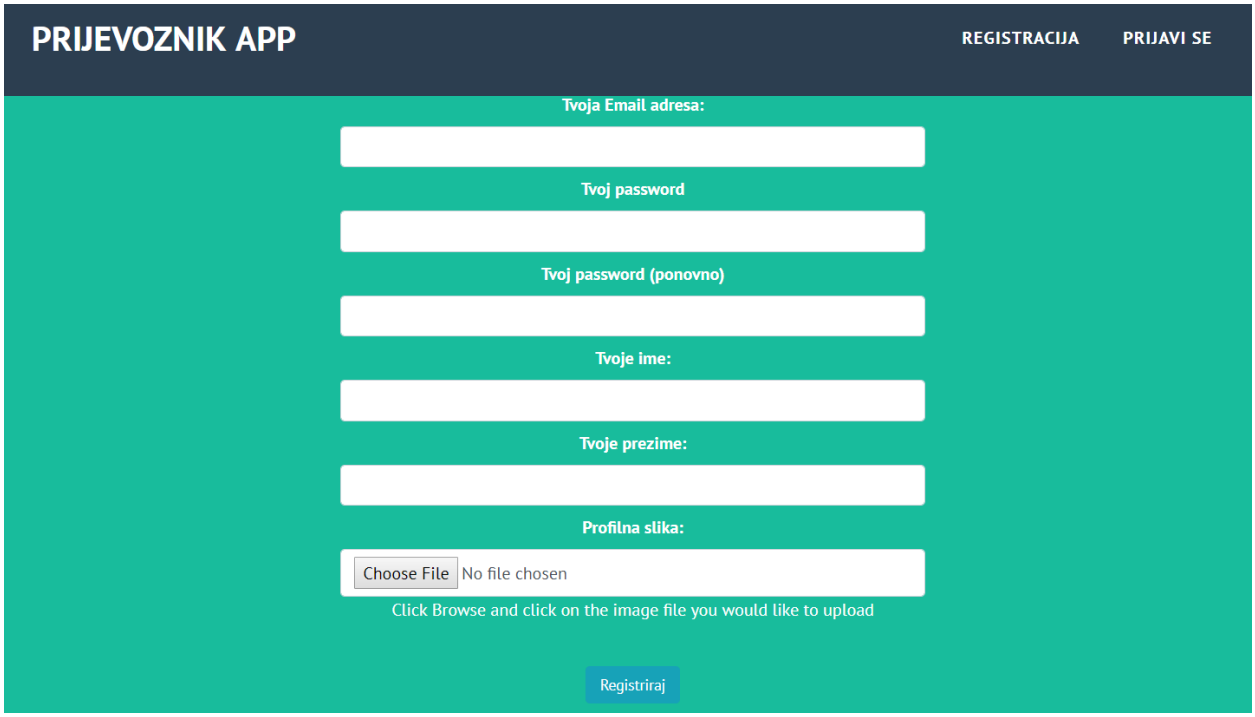
Sl. 4.5. Prikaz naslovne stranice

Naslovna stranica je vrlo jednostavna. Ističe se logo u sredini te u desnom gornjem uglu stranice poveznice za prijavu i registraciju korisnika. Za dizajn se koristi Bootstrap tema koja je dostupna za besplatno preuzimanje i editiranje sa stranice https://bootswatch.com.

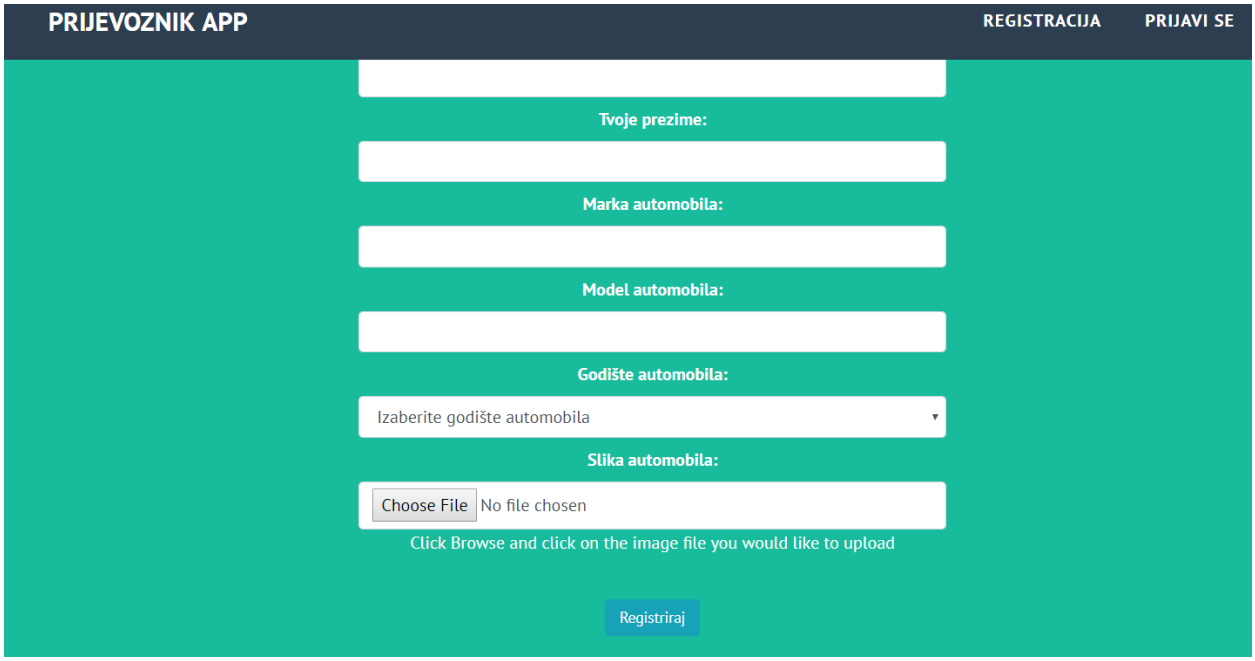
Sl. 4.6. Forma za registraciju i prijavu

Prilikom registracije nudi se mogućnost odabira tipa korisnika. Korisnici se dijele na vozača koji nudi usluge prijevoza te putnika koji traži usluge prijevoza. Korisnik s jednim korisničkim računom ne može biti istovremeno vozač i putnik.

Moguće je u sljedećim inačicama aplikacije omogućiti nadogradnju korisničkog računa iz statusa putnik u status vozač, ali u trenutačnoj inačici to nije omogućeno.



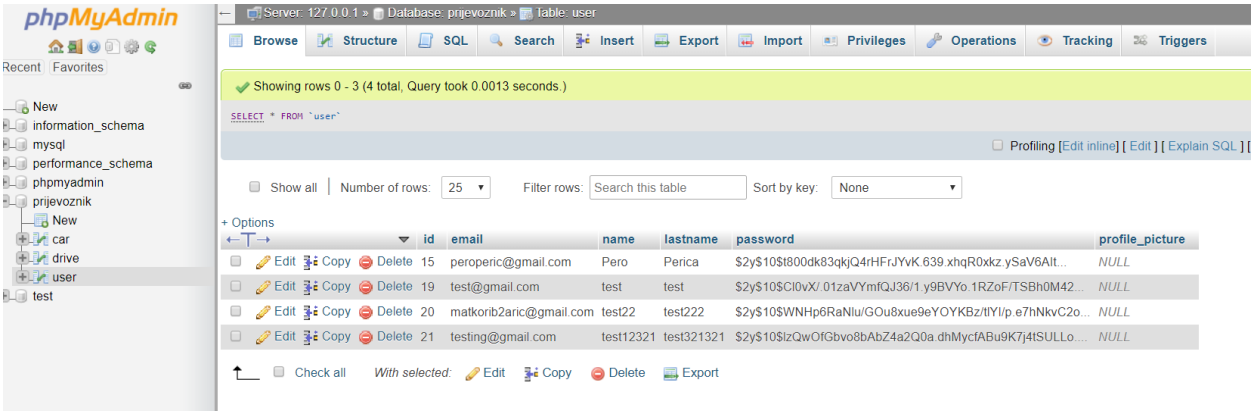
Sl. 4.7. Forma registracije za putnike



Sl. 4.8. Forma registracije za vozače

Obje forme za registraciju sadrže osnovne podatke o korisniku kao što su: ime, prezime, mail adresa i lozinka.

Razlika nastaje u dijelu gdje vozač, uz gore navedene podatke, mora unijeti i podatke o svom automobilu te priložiti fotografiju istog automobila. Forma za registraciju podržava hrvatske dijakritičke znakove. Za zaporku je postavljeno ograničenje da je minimum četiri znaka koja se moraju unijeti kako bi forma prihvatila registraciju. Verifikacija je odrađena Password Hash metodom gdje se unesena zaporka u bazi sprema kao hash zapis, te se prilikom prijave svaka iduća unesena zaporka također pretvara u hash zapis i uspoređuje sa prvobitnim unesenim kod registracije. Hash zapis je način kriptiranja teksta gdje se originalni zapis nepovratno pretvara u kriptiranu inačicu i kao takav se sprema u bazu. Zbog takvog načina nije moguće prikazati originalnu zaporku nego samo generirati novu zaporku.

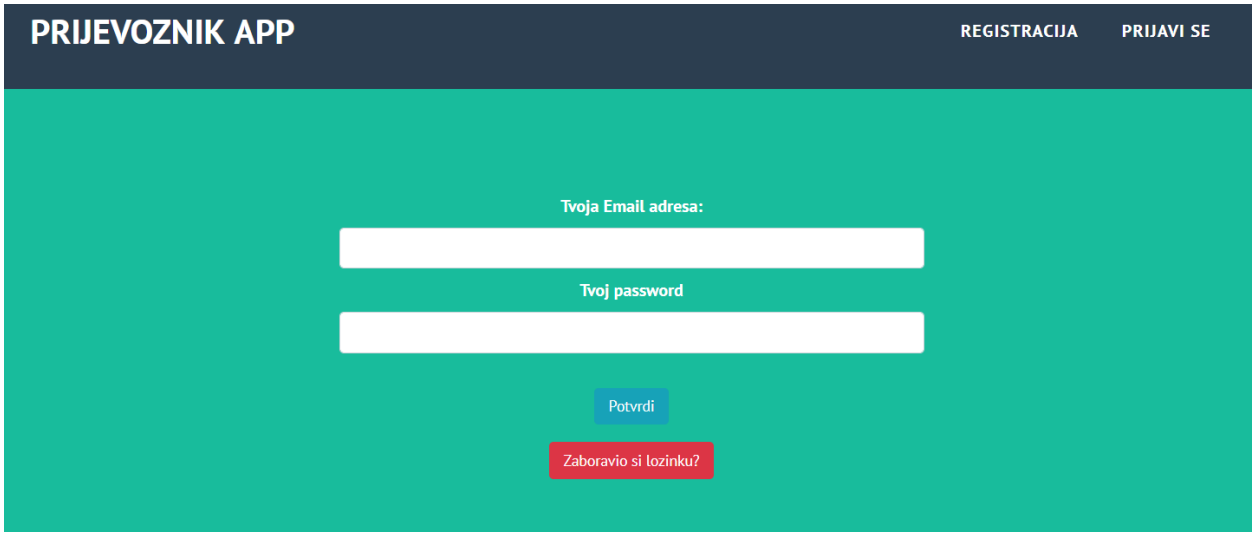


Sl. 4.9. Spremanje lozinke u bazu podataka

Prednost ove metode u odnosu na MD5 je ta što nije moguće iz hash zapisa rekonstruirati zaporku, dok za MD5 postoji način kako rekonstruirati zaporku nakon kriptiranja. Za ovaj tip servisa Password hash metoda je optimalna kada se uzme u obzir sigurnost i vrijeme potrebno za verfikaciju prijave korisnika. Prilikom odabira metode verifikacije prijave potrebno je uzeti u obzir sadržaj kojima korisnik pristupa nakon prijave, vrijeme potrebno za verifikaciju zaporke te mogućnosti poslužitelja na kojem aplikacija je postavljena. Ukoliko podignemo razinu sigurnosti dolazi do čekanja na prijavu, te povećava cijenu prostora na poslužitelju.

Nakon što se korisnik registrirao, automatski mu se otvara prozor za prijavu gdje upisuje adresu elektroničke pošte s kojom je odradio registraciju te zaporku.

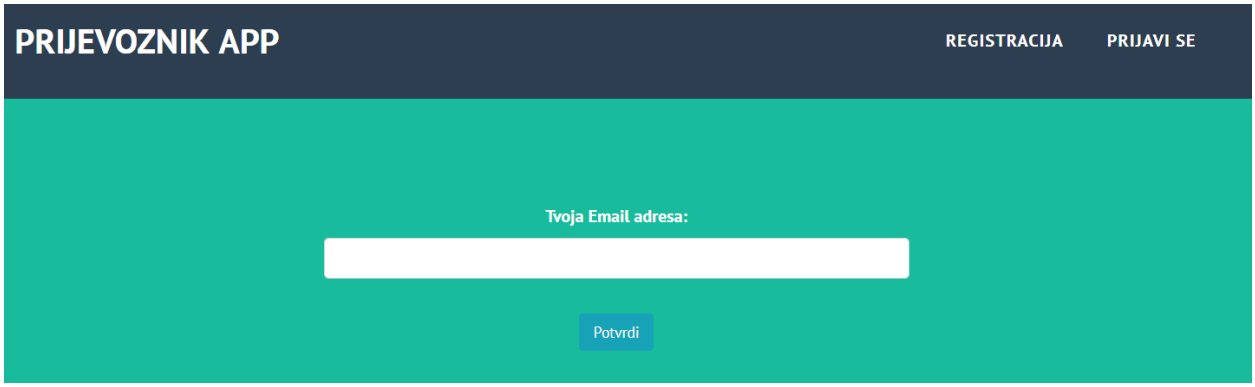
Ukoliko se oba podatka poklapaju s onima spremljenima u bazu, aplikacija daje korisniku pristup korisničkom sučelju. U suprotnome se javlja greška neispravnih podataka.



Sl. 4.10. Prijava u aplikaciju

U slučaju da je korisnik zaboravio zaporku, omogućeno mu je da preko poveznice Zaboravio si lozinku? zatraži novu. Resetiranje zaporke se odrađuje tako da korisnik na upisanu adresu elektroničke pošte dobije novu nasumično kreiranu zaporku. Poslana zaporka se automatski sprema u bazu kao nova važeća zaporka. Korisnik nakon toga ima mogućnost na poveznici promjena profila postaviti novu željenu zaporku. Ovo je jednostavniji način odrade resetiranja zaporke. Složeniji način bi se odradio tako da bi korisnik prvo dobio verifikacijsku poruku na adresu elektroničke pošte koju je unio prilikom registracije s poveznicom preko koje bi sam unio novu zaporku. Prilikom određivanja načina resetiranja zaporke uvijek se treba voditi o tome je li za tip aplikacije potreban kompleksniji sustav koji zauzima više prostora na poslužitelju i sporije se odrađuje. Za ovu aplikaciju je dovoljan način na koji se odrađuje. Preporuka je da se za u podešavanju servera s kojega se šalju zaporke korisnicima koriste adrese elektroničke pošte koje dobijemo u sklopu zakupa domene i prostora na poslužitelju. Mogu se koristiti i besplatne mail adrese kao što su gmail, yahoo i sl., ali nastaje problem s autorizacijom iz razloga što gmail,

yahoo i slični servisi imaju nekoliko razina autorizacije koji se mogu aktivirati te onemogućiti slanje nove zaporke.



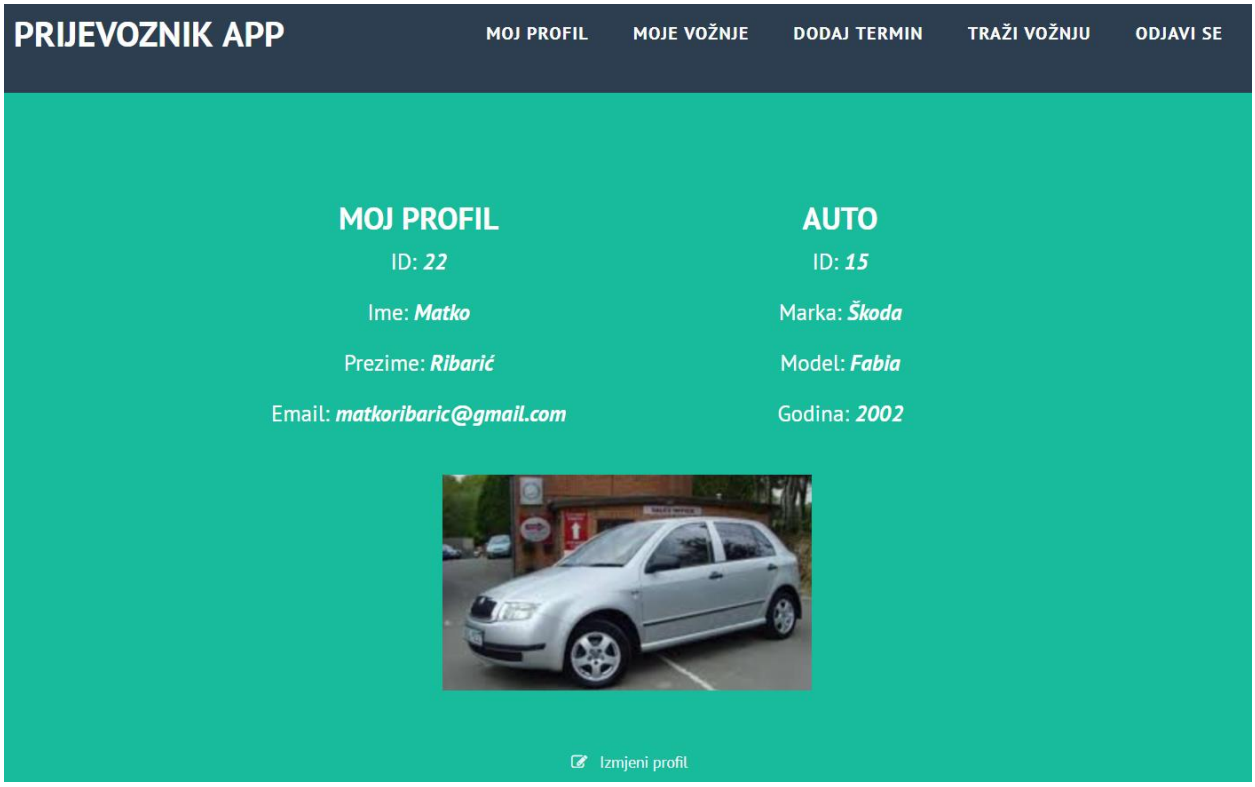
Sl. 4.11. Potvrda mail adrese za zaboravljenu zaporku

Prilikom prijave u aplikaciju korisniku se nudi mogućnosti dodavanja ili traženja vožnje ovisno o tome kakav je profil kreiran prilikom registracije. Korisnik koji se registrirao kao vozač ima mogućnosti dodavanja termina vožnje te pregledavanja termina vožnji koji su dodali putnici. Isto vrijedi za obrnuti slučaj.

Sl. 4.12. Naslovna stranica za prijavljene korisnike

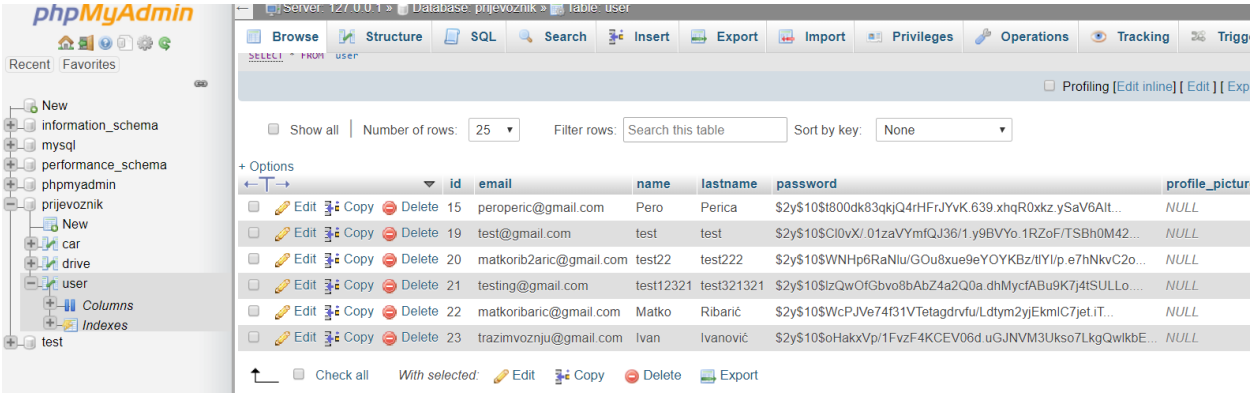
Neovisno o tipu korisnika prikazuje se ista naslovna stranica s time da poveznice Dodaj Termin i Traži Vožnju vode na različite mogućnosti ovisno o tipu prijavljenog korisnika. Ako je prijavljeni korisnik vozač poveznica Dodaj Termin omogućava dodavanje termina u kojem korisnik vozi na određenoj relaciji, dok poveznica Traži Vožnju vodi na mogućnost pretraživanja relacija i termina koji su postavili putnici koji traže vožnju. Sukladno tome, kada je prijavljeni korisnik klasificiran kao putnik navedene poveznice vode na mogućnosti dodavanja termina i relacije u kojem se traži vožnja, te pregledavanja termina i relacija na kojima je ponuđena vožnja od strane korisnika koji su klasificirani kao vozači.

Prethodno je spomenuta promjena zaporke na zahtjev korisnika koju potom može korisnik sam promijeniti na postavkama profila. Izuzev toga svaki korisnik ima mogućnost pregleda i uređivanja svojeg profila. Podaci koje može promijeniti su identični onima koji se traže prilikom registracije profila. S programske strane je promjena profila odrađena kao da se korisnik ponovno registrira, a baza samo zabilježi novo stanje podataka koje je korisnik unio.



Sl. 4.13. Prikaz „Moj profil“

Na slici 4.12 je vidljivo da svaki korisnik ima dodijeljen jedinstven identifikacijski broj (u daljnem tekstu ID). Isto je tako vidljivo da ID automobila nije identičan onome korisnika. To je iz razloga što nije nužno da je svaki korisnik vozač te posjeduje automobil. Aplikacija sama pridružuje ID automobila matičnom ID korisnika te se time omogućava brže pretraživanje baze podataka i prikaz profila. Kako bi se spriječile moguće greške u povezivanju korisnika i automobila nakon brisanja ili dodavanja, ID automobila se uvijek u bazu dodaje automatski povećavanjem ID-a.

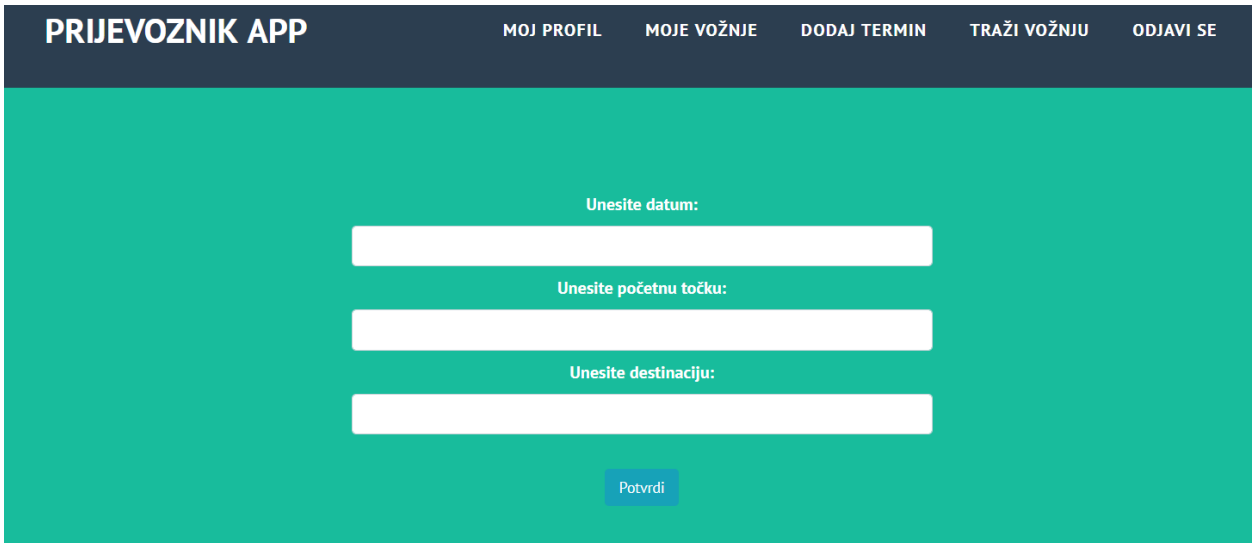


Sl. 4.14. Prikaz korisnika u bazi

Pregledom baze može se vidjeti da je registracija i dodjeljivanje ID korisnika funkcionira na prethodno naveden način. U suprotnom bi zadnji registrirani korisnik dobio ID 16, a ne 23. Na ovaj način se preventivno sprječavaju situacije u kojima se može dogoditi da se povezivanje ID korisnika i ID automobila odradi na krivi način.

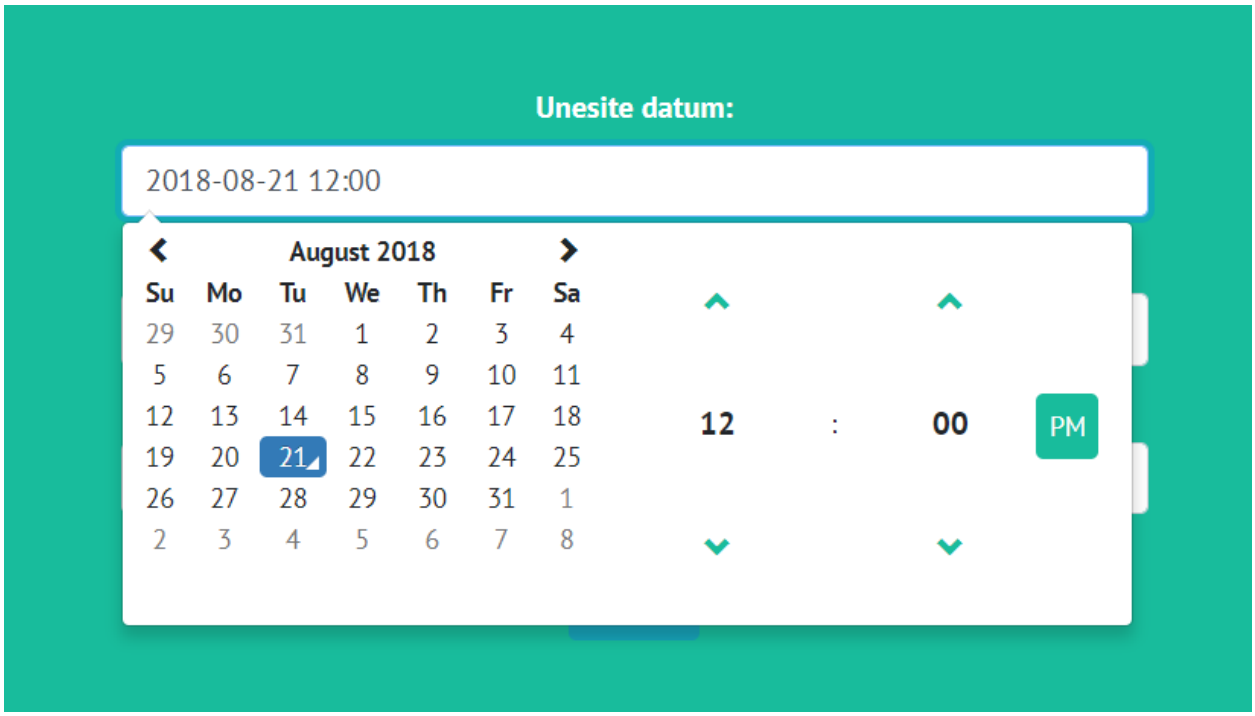
Korisnik klasificiran kao vozač postavlja termin u kojem vozi na način da klikne na poveznicu Dodaj Termin i ispuni potrebne podatke. Prvo na što treba obratiti pozornost je datum i točno vrijeme polaska. Nakon toga se unosi početna i završna destinacija.

U ovoj inačici aplikacije korisnik koji više dana vozi na istoj relaciji mora za svaki dan pojedinačno dodati termine vožnje. U idućoj inačici planira se implementacija mogućnosti da korisnik istovremeno za odabrane dane doda isti termin i relaciju vožnje. Sukladno tome putnik na isti način dodaje termin i relaciju u kojem traži vožnju ako nije već našao željeni termin pretraživanjem već dodanih termina i relacija vožnje od strane korisnika klasificiranih kao vozač.



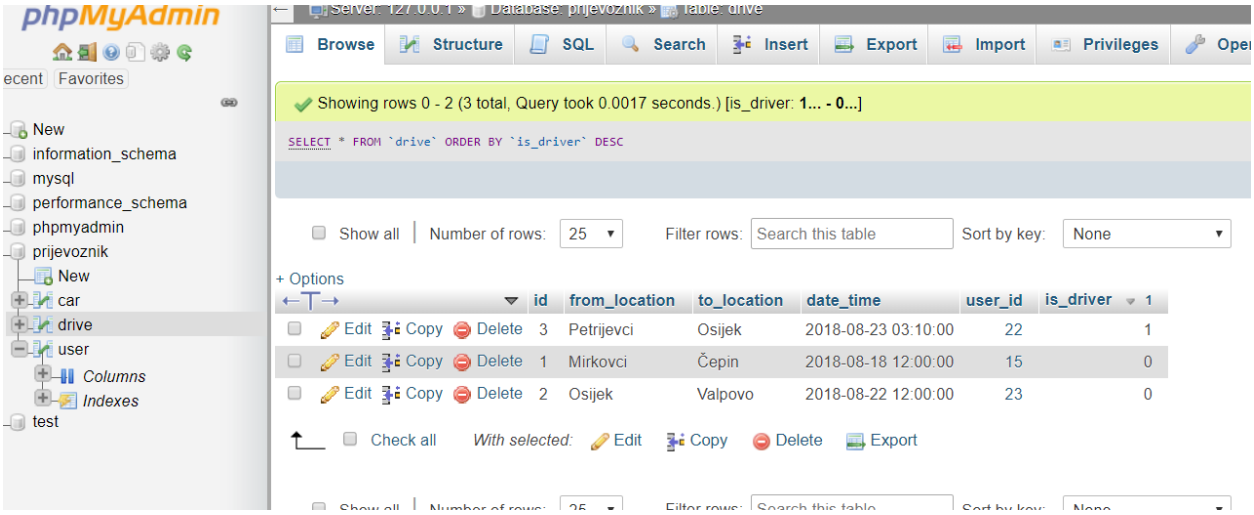
Sl. 4.15. Dodaj Termin

Kako bi se onemogućilo proizvoljno upisivanje datuma i vremena, te višestrukih pokušaja traženja odgovarajućeg formata datuma i vremena omogućeno je odabir putem padajućeg izbornika da se izbjegnu greške i ponavljanje unosa.



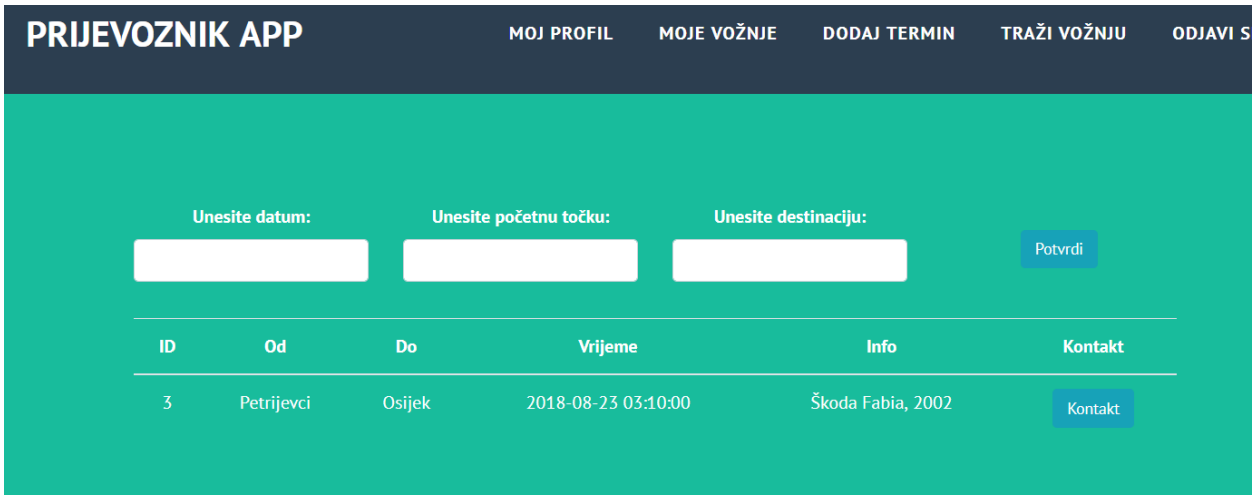
Sl. 4.16. Kalendar vožnje

Nakon što se ispune zadana polja, tipkom Potvrdi se vožnja spremi u bazu gdje joj je također kao korisnicima dodijeljen njen ID. Na slici 4.16 vidljivo je da baza svakoj vožnji dodijeli ID, te tako razlikuje vožnje koje je unio vozač u odnosu na one koje je unio putnik. To je kasnije važno kod prikaza pretrage vožnji kako se vozačima ne bi prikazivali termini ponude umjesto potražnje vožnje.



Sl. 4.17. Zapis vožnji u bazi podataka

Putniku se preko poveznice Tražim Vožnju omogućava pretragu svih postavljenih termina vožnji koje je postavio vozač. Otvaranjem poveznice putniku se nude sve ponuđene vožnje uz mogućnost dodatnog filtriranja vožnji. Za iduću inačicu aplikacije planirano je da se putniku prikažu samo za taj dan ili čak da se ne prikaže ništa dok putnik sam ne unese parametre pretraživanja. U planu je i razvoj funkcionalnosti gdje bi korisnik, neovisno je li u statusu vozač ili putnik, na polju info dobio poveznicu na javni dio profila korisnika gdje bi se vidjeli podaci koje je korisnik dao na registraciji koji se sastoje od imena, prezimena i fotografije putnika, te još dodatno fotografije, model i godište automobila vozača. Trenutno je omogućena samo informacija o modelu i godištu automobila prilikom pregleda traženih vožnji.



Sl. 4.18. Prikaz pretrage ponuđenih vožnji

Na identičan način kao što putnik traži vožnju odrađena je i pretraga traženih vožnji. Nakon što putnik ili vozač nađu termin i destinaciju koja odgovara njihovim parametrima pretrage nudi im se mogućnost slanja poruke na adresu elektroničke pošte korisnika koji je postavio traženu ili nuđenu vožnju. U ovoj fazi razvoja aplikacije kontakt se odradio na način da se samo pošalje poruka putem email protokola.

U idućim inačicama aplikacije planirano je slanje poruka unutar same aplikacije te da korisnik prilikom prijave u aplikaciju ima mogućnost čitanja i slanja poruka drugim korisnicima.

5. TESTIRANJE APLIKACIJE

Testiranje aplikacije je podijeljenjo na dva načina testiranja, mjernjem brzine učitavanja aplikacije i vizualnog prikaza korisniku. Ovo su jedni od najvažnijih kriterija korisnika za Internet aplikaciju. Testiranje se radi na dva računala, oba pokretana sa Windows 10 operativnim sustavom. Razlika je u konfiguraciji računala. Prvo računalo pokreće Intelov i5 četverojezgreni procesor sa 3,1 Ghz po jezgri, osam gigabajta radne memorije DDR3 tipa, AMD-ovu grafičku karticu HD 7770 OC s jedan gigabajt radne memorije, te čvrsti disk SSD tipa kapaciteta 120 gigabajta. Drugo računalo pokreće Intelov dvojezgreni čip brzine 1,7 Ghz, dva gigabajta radne memorije DDR2 tipa, integrirana Intelova grafika te čvrsti disk kapaciteta 250 gigabajta.

5.1. VIZUALNA USPOREDBA APLIKACIJE

Vizualno testiranje aplikacije se odvija na način da se aplikacija pokrene na nekoliko preglednika istovremeno te da se subjektivno ocjenjuje prikaz aplikacije. Internet preglednici koji su korišteni za vizualno testiranje su: Google Chrome i Microsoft Edge.

Sl. 5.1. Naslovna stranica prikazana u Microsoft Edge

Sl. 5.2. Naslovna stranica prikazan u Google Chrome

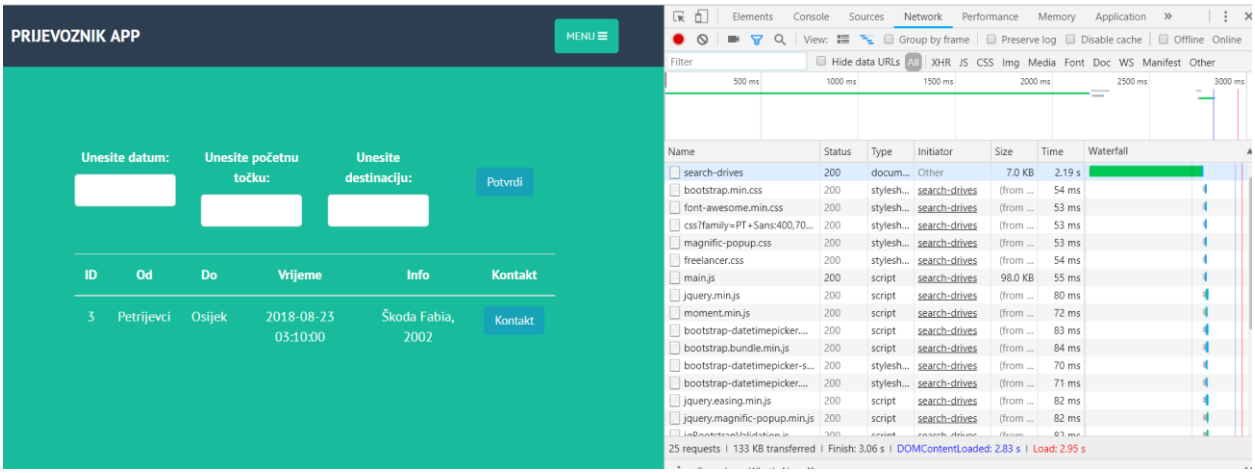
Iz slika 5.1 i 5.2 možemo zaključiti kako preglednici imaju različit način prikazivanja veličine fonta prilikom skaliranja loga. Dok Microsoft Edge zadržava veličinu fonta u gornjem izborniku, Chrome sa skaliranjem stranice smanjuje font. Aplikacija je prvenstveno rađena i optimizirana za Google Chrome preglednik, te je razvoj iste uvijek testiran u Chrome pregledniku. Uzrok ove razlike je u načinu na koji preglednici prikazuju skaliranje ili zumiranje stranice. Dizajn je rađen na principu prikazivanja postotka veličine fonta u odnosu na zadanu veličinu fonta, te se tu vidi razlika prikaza na 2 preglednika.

Prikaz boja, fontova i oblikovanja na oba preglednika funkcionira kako je i zamišljeno. Jedina primjetna razlika se nalazi u veličini fonta na naslovnoj stranici. Identičan prikaz, samo skaliran je i na uređajima koji imaju nižu rezoluciju prikaza. Primarni vizualni testovi su odrađeni na uređajima s rezolucijom prikaza 1920x1080 piksela, dok se još testiralo na rezolucijama 1280x800 i 1366x768 piksela. Veličine ekrana uređaja na kojima se testiralo su 15,6“ i 15,4“. Na oba preglednika i uređaja prikaz sadržaja je uvijek unutar okvira te nema raspadanja. Iako je font vidno različit od dizajna uvijek je svagdje jednako raspoređen. Bitno je napraviti responzivan pogled zbog toga što mobiteli i tableti preuzimaju prvenstvo korištenja aplikacija u odnosu na

računala. Mobilni prikaz je još jedan aspekt aplikacije koji bi u sljedećim inačicama trebao biti nadograđen.

5.2. MJERENJE BRZINE UČITAVANJA U RAZLIČITIM PREGLEDNICIMA

Mjerenje brzine učitavanja stranice se vršilo koristeći mogućnosti koje su ugrađene u sam preglednik. Osvježavanje stranice je rađeno nekoliko puta. Rezultati su prikazani u tablici.



Sl. 5.3. Prikaz testiranja u Google Chrome pregledniku

Stranica/Preglednik	Naslovna (s)	Prijava korisnika(s)	Traži Vožnju(s)	Dodaj Termin(s)
Chrome 1	1,45	2,31	0,82	0,81
Chrome 2	0,87	0,9	0,79	0,83
Chrome 3	0,78	1,02	0,8	0,79
Chrome 4	0,82	0,9	0,77	0,8
Chrome 5	0,84	0,88	0,81	0,81
Prosjek Chrome	**0,952**	**1,202**	**0,798**	**0,808**
Edge 1	0,99	0,95	0,73	0,87
Edge 2	0,87	0,89	0,67	0,91
Edge 3	1,1	0,91	0,45	0,77
Edge 4	1,43	1,02	0,47	0,55
Edge 5	0,93	0,93	0,61	0,74
Prosjek Edge	**1,064**	**0,94**	**0,586**	**0,768**

Tablica 5.1. Rezultati testiranja

Iz tablice 5.1 je vidljivo da se stranica u prosjeku brže otvara u Microsoft Edge pregledniku. Kada izoliramo prvo učitavanje stranice i pojedinih dijelova stranice na Google Chrome pregledniku dobiju se zanemarive razlike u vremenu učitavanja sadržaja stranice na oba preglednika. Prosjek učitavanja bilo kojeg dijela stranice je ispod jedne sekunde što je zadovoljavajuće.

6. ZAKLJUČAK

U ovom diplomskom radu napravljena je aplikacija za zajedničko korištenje vozila koristeći Zend programski okvir. Ideja za aplikaciju je nastala od potrebe jednostavnijeg pronalaska prijevoza uz relativno niske troškove putovanja. S programske strane Zend se nametnuo kao logičan izbor za izradu same aplikacije, prvenstveno zbog korištenja u tvrtkama i programske podrške.

Za modeliranje aplikacije korišteni su UML dijagrami, a najveći naglasak je bio na klasnom i sekvencionalnom dijagramu. Definirane su klase i međusobni odnosi između klasa, te slučajevi korištenja u pojedinim situacijama. Dizajn je implementiran u PHPStorm aplikaciji, dok su pojedini dijelovi odrađivani zasebno korištenjem CSS, HMTL i PHP skriptnih jezika. Aplikacija je rađena za sve preglednike, iako je prvenstveno prilagođena i velikim dijelom testirana na Google Chrome pregledniku.

Izrada aplikacije je odrađena u dvije faze. Prvi dio je bio izraditi grafičko sučelje (eng. frontend), dok se drugi dio izrade odnosio na pozadinsku podršku (eng. backend). Kako je Zend prvenstveno programski okvir za pozadinski rad na aplikacijama, taj dio izrade se radio isključivo tamo. Uz navedeno korištena je i XAMPP aplikacija, MySQL baza podataka te Apache internet poslužitelj kako bi se moglo samostalno odraditi testiranje aplikacije bez potrebe za zakupljivanjem internetskog prostora.

Aplikacija ima potencijala za razvoj, posebice na mobilnom dijelu, gdje se uz mobilni prikaz može kreirati potpuno neovisna mobilna aplikacija za korištenje na Android ili Iphone platformi. Postoji mogućnost proširenja opcija korištenja, od kreiranja veće baze podataka do povećavanja broja parametara koje se mogu koristiti prilikom pretrage.

LITERATURA

[1] Krishna Shasankar: Zend Framework 2.0 by Example Beginner's Guide, Birmingham-Mumbai, 2013

[2] Darija Ramljak: Vizualno modeliranje objektno orijentiranih sustava korištenjem uml-a, Zagreb, 2001

POPIS SLIKA I TABLICA

POPIS SLIKA

POPIS TABLICA

SAŽETAK

U ovom radu napravljen je dizajn i izrađena aplikacija za zajedničko korištenje vozila. Implementacija je napravljena u obliku internet aplikacije s pripadajućom bazom podataka. Korisničko sučelje je kreirano kako bi omogućilo registraciju korisnika, prijavu korisnika te pretragu ovisno o zadanim parametrima. Za izradu UML dijagrama korišten je StarUML program za kreriranje UML dijagrama, a za izradu aplikacije korišten je PHPStorm aplikacija. PHP, HTML i CSS su skriptni jezici koji su korišteni prilikom izrade aplikacije. Cilj dizajna i implementacije je kreiranje i pretraga baze podataka vezano za zajedničko korištenje vozila.

Ključne riječi: UML dijagram, internet aplikacija, baza podataka, HTML, PHP, CSS, Zend programski okvir

ABSTRACT

Using Zend tools in the development of web applications for sharing vehicles

This master thesis presents a design and implementation of an internet application for car sharing. Implementation of design were made in the form of web appliation with associated database. User interface was created to provide user registration, user login and search for car sharing data. To create UML diagrams in this thesis StarUML program was used. PHPStorm application was used to create web application. PHP, HMTL and CSS were programming languages used in implementation. The goal of design and implementation was creation and search database for car sharing.

Keywords: UML diagram, web application, database, HTML, PHP, CSS, Zend Framework

ŽIVOTOPIS

Matko Ribarić rođen je 28.08.1988. godine u Osijeku. Sin Ivana i Eve Ribarić s prebivalištem u Petrijevcima. Pohađao je osnovnu školu u Petrijevcima nakon čega 2003. godine upisuje Elektrotehničku i prometnu školu u Osijeku. Srednjoškolsko obrazovanje završava s vrlo dobrim uspjehom akademske godine 2006/2007. Iste godine upisuje se na Elektrotehnički fakultet u Osijeku, smjer računarstvo. 2012. godine završava smjer preddiplomskog studija te upisuje prvu godinu diplomskog studija, smjer procesno računarstvo.

Matko Ribarić _____

PRILOZI

Prilog 4.1 (stranica: korisnik)

```
?php


/**
 * Class Application_Model_User
 */
class Application_Model_User extends Zend_Db_Table_Abstract
{


  protected $_name = 'user';


  private $id;
  private $email;
  private $name;
  private $lastname;
  private $password;
  private $profile_picture;


  /**
   * @return mixed
   */
  public function getId()
  {
    return $this->id;
  }
```

```
/**
 * @return mixed
 */
public function getEmail()
{
  return $this->email;
}

/**
 * @param mixed $email
 */
public function setEmail($email)
{
  $this->email = $email;
}

/**
 * @return mixed
 */
public function getName()
{
  return $this->name;
}

/**
```

```
 * @param mixed $name
 */
public function setName($name)
{
  $this->name = $name;
}

/**
 * @return mixed
 */
public function getLastname()
{
  return $this->lastname;
}

/**
 * @param mixed $lastname
 */
public function setLastname($lastname)
{
  $this->lastname = $lastname;
}

/**
 * @return mixed
 */
```

```
public function getPassword()
{
  return $this->password;
}

/**
 * @param mixed $password
 */
public function setPassword($password)
{
  $this->password = $password;
}

/**
 * @return mixed
 */
public function getProfilePicture()
{
  return $this->profile_picture;
}

/**
 * @param mixed $profile_picture
 */
public function setProfilePicture($profile_picture)
{
```

```
  $this->profile_picture = $profile_picture;
}




/**
 * Generate and set password hash
 *
 * @param $password
 */
public function generateAndSetPasswordHash($password)
{
  $this->setPassword(password_hash($password, PASSWORD_DEFAULT));
  #$this->setPassword(md5(sha1(md5($password))));
}


/**
 * @return mixed
 * @throws Exception
 */
public function getCar()
{
  $carModel = new Application_Model_Car();


  $carDbRow = $carModel->fetchRow(array('user_id='. $this->getId()));
```

```
  if ($carDbRow) {
    return $carModel->setModelFromDbRow($carDbRow);
  }

  return null;
}




/**
 * Check password
 *
 * @param $password
 * @return bool
 */
public function checkPassword($password)
{
  return password_verify($password, $this->getPassword());
}

/**
 * Set Model From DB Row
 *
 * @param $dbRow
 * @return $this
 * @throws Exception
```

```
 */
public function setModelFromDbRow($dbRow)
{
  if (!$dbRow) {
    throw new Exception('Missing DB row');
  }

  $this->id = (isset($dbRow['id'])) ? $dbRow['id'] : null;
  $this->email = (isset($dbRow['email'])) ? $dbRow['email'] : null;
  $this->name = (isset($dbRow['name'])) ? $dbRow['name'] : null;
  $this->lastname = (isset($dbRow['lastname'])) ? $dbRow['lastname'] : null;
  $this->profile_picture = (isset($dbRow['profile_picture'])) ? $dbRow['profile_picture'] : null;

  return $this;
}

/**
 * To array
 *
 * @return array
 */
public function toArray()
{
  return array(
    'email' => $this->email,
    'password' => $this->password,
```

```
      'name' => $this->name,
      'lastname' => $this->lastname,
      'profile_picture' => $this->profile_picture
    );
  }

}
```

PRILOG 4.2 (stranica: dio koda vezan za dodavanje termina)

```
<?php

/**
 * Class Application_Model_Drive
 */
class Application_Model_Drive extends Zend_Db_Table_Abstract
{
  protected $_name = 'drive';

  private $id;
  private $from;
  private $to;
  private $dateTime;
  private $userId;
  private $is_driver = false;
```

```
/**
 * @return mixed
 */
public function getId()
{
  return $this->id;
}

/**
 * @return mixed
 */
public function getFrom()
{
  return $this->from;
}

/**
 * @param mixed $from
 */
public function setFrom($from)
{
  $this->from = $from;
}

/**
```

```
 * @return mixed
 */
public function getTo()
{
  return $this->to;
}

/**
 * @param mixed $to
 */
public function setTo($to)
{
  $this->to = $to;
}

/**
 * @return mixed
 */
public function getDateTime()
{
  return $this->dateTime;
}

/**
 * @param mixed $dateTime
 */
```

```
public function setDateTime($dateTime)
{
  $this->dateTime = $dateTime;
}

/**
 * @return mixed
 */
public function getUserId()
{
  return $this->userId;
}

/**
 * @param mixed $userId
 */
public function setUserId($userId)
{
  $this->userId = $userId;
}

/**
 * @return mixed
 */
public function getisDriver()
{
```

```
  return $this->is_driver;
}

/**
 * @param mixed $is_driver
 */
public function setIsDriver($is_driver)
{
  $this->is_driver = $is_driver;
}

/**
 * Set Model From DB Row
 *
 * @param $dbRow
 * @return $this
 * @throws Exception
 */
public function setModelFromDbRow($dbRow)
{
  if (!$dbRow) {
    throw new Exception('Missing DB row');
  }
```

```
    $this->id = (isset($dbRow['id'])) ? $dbRow['id'] : null;
    $this->from = (isset($dbRow['from_location'])) ? $dbRow['from_location'] : null;
    $this->to = (isset($dbRow['to_location'])) ? $dbRow['to_location'] : null;
    $this->dateTime = (isset($dbRow['date_time'])) ? $dbRow['date_time'] : null;
    $this->userId = (isset($dbRow['user_id'])) ? $dbRow['user_id'] : null;
    $this->is_driver = (isset($dbRow['is_driver'])) ? $dbRow['is_driver'] : null;
    return $this;
  }

  /**
   * To array
   *
   * @return array
   */
  public function toArray()
  {
    return array(
      'from_location' => $this->from,
      'to_location' => $this->to,
      'date_time' => $this->dateTime,
      'user_id' => $this->userId,
      'is_driver' => $this->is_driver
    );
  }

}
```