

Aplikacija za evidenciju dobrovoljnih vožnji
-projektna dokumentacija-

Beograd, 2019.

SADRŽAJ

1	Kratak opis projekta	4
2	Postupak razvoja softvera	5
2.1	Plan projekta	5
2.2	Planirani rezultati za kritične tačke	6
2.3	Detaljan plan projekta	8
3	Analiza zahteva	9
4	Projektovanje sistema	10
4.1	Arhitektura sistema	10
4.2	Programski jezik i obrazloženje	11
4.3	Opis algoritama	12
4.4	Struktura programskih modula	18
4.5	Dijagram slučajeva korišćenja	28
4.6	Dijagram sekvence	30
4.7	Dijagram komponenata	31
4.8	Dijagram raspoređivanja	32
4.9	Model baze podataka	33
5	Implementacija softvera	36
6	Testiranje softvera	39
6.1	Jedinično testiranje	39
6.2	Integraciono testiranje	42
6.3	Grafik zavisnosti pronađenog broja grešaka u softveru po nedeljama	43
7	Isporuka softvera	44
	Literatura	49

1 Kratak opis projekta

Projektna tema obrađuje implementaciju aplikacije za zajednički automobilski prevoz putnika. Tema je izuzetno aktuelna i odnosi se na veoma značajnu sferu naših života, a obuhvata kako prevoz putnika, tako i prevoz robe. Život u velikom gradu često ima svoje prednosti, ali i jednu veliku manu. To su saobraćajne gužve. Često smo i sami svedoci situacija u saobraćaju gde se u jednom automobilu vozi najčešće samo vozač i eventualno jedan putnik. Ideja je da se maksimalno iskoriste resursi vožnje koja se odvija po određenoj ruti, gde se teži ka tome da se popune sva mesta u jednom automobilu ili prevoznom sredstvu. Direktne posledice ovakvog rešenja jesu smanjenje troškova vozača i putnika, kao i pozitivno dejstvo na smanjenje gužvi u saobraćaju. Takođe, na ovaj način se vrši redukciju zagađenosti koja potiče od izduvnih gasova automobila.

Imajući u vidu obrazloženu ideju, razvijena je aplikacija koja ima za cilj spajanje interesnih strana radi ostvarivanja zajedničke vožnje. Zamisao se sastoji u tome da registrovani članovi mogu da učestvuju bilo u izboru tražene vožnje ili da sami ponude vožnju drugim korisnicima. Obostrana komunikacija se odvija putem aplikacije u vidu slanja mejlova, gde se i komentari i rezervacije ostvaruju na jednostavan, brz i pouzdan način.

S obzirom na kompleksnost projektnog zadatka bilo je potrebno pažljivo planiranje svakog koraka u razvoju softvera, kako se ne bi došlo u situaciju da se vrše ispravke u nekoj o odmaklih faza dizajniranja i razvoja.

2 Postupak razvoja softvera

Za metod modelovanja softvera odabran je kaskadni model. Ovakva odluka je doneta jer se razvojni tim sastoji od samo nekoliko članova što znači da je neophodna dobra organizacija kako bi se vreme za razvoj aplikacije što bolje iskoristilo.

2.1 Plan projekta

Projekat se sastoji iz 14 faza:

1. Razgovor sa naručiocem softvera
2. Analiza zahteva
3. Sačinjavanje specifikacije
4. Sačinjavanje modela baze podataka
5. Projektovanje sistema
6. Pisanje programskog koda softverskog rešenja
7. Izrada baze podataka
8. Kodiranje serverskog dela aplikacije
9. Kodiranje klijentskog dela aplikacije
10. Testiranje softverskog rešenja
11. Implementacija softverskog rešenja
12. Verifikacija sistema
13. Testiranje sistema
14. Puštanje sistema u rad i održavanje

2.2 Planirani rezultati za kritične tačke

- **Model baze podataka:** Izrada modela nerelacione baze podataka. Model treba da sadrži sve entitete sa precizno definisanim atributima i njihovim tipom. Definisana sva ograničenja odnosno dozvoljene vrednosti koje mogu imati entiteti u okviru kolekcije (ekvivalent tabeli u relacionoj bazi podataka). Implementirani sledeći modeli (u vidu Javaskript fajlova):
 - Model korisnika (user.js): sadrži polja za čuvanje podataka o korisnicima veb aplikacije i funkcije za obradu grešaka u slučaju pokušaja prijave nepostojećeg korisnika i unosa netačne korisničke šifre (funkcija koja hešuje pristiglu šifru i upoređuje sa hešom šifre korisnika iz baze podataka)
 - Model vožnje (ride.js): sadrži polja za čuvanje podataka o vožnjama veb aplikacije kao i funkcije za obradu grešaka u slučaju pokušaja brisanja nepostojeće vožnje, kao i slučaja kada nema vožnji u bazi podataka
- **Izrađeno korisničko sučelje:** Razvijena osnovna verzija sučelja sa formama koje nude korisniku:
 - **na početnoj stani aplikacije:** mogućnost registracije, resetovanja šifre ukoliko je zaboravljena, prijavu na sistem;
 - **na početnoj strani prijavljenog korisnika:** pretragu vožnji prema nazivu, datumu, polazištu, dolazištu, zatim postavljanje zahteva za vožnju (forma sa mestima za unos polazišta, dolazišta, vremena polaska, cene, i napomene), mogućnost postavljanja ponude za vožnju sa pripadajućom formom za unos parametara (mesta za unos polazišta, dolazišta, vremena polaska, cene, i napomene), realizovanim menijem (pregled profila korisnika, korisnikovih rezervisanih vožnji, njegovih zahtevanih vožnji i njegovih ponuđenih vožnji), formu za pregled i izmenu korisnikovih podataka (ime i prezime, broj telefona, vozilo, i mobilni operater), prikaz integrisane Gugl karte, prikaz informacija o ruti (dužina u km i vreme trajanja putovanja), pripadajuće kontrole (dugmad) za zakazivanje, otkazivanje vožnje, rezervaciju mesta, slanja poruke putniku/vozaču, formu za pregled vožnje od interesa i slanje poruke vozaču/putniku;
- **Izrađena serverska strana aplikacije:** Implementirani opsluživači (hendleri) za HTTP POST, GET, PUT i DELETE zahteve. Razvijeni sledeći moduli:
 - **server.js:** Javaskript modul koji pokreće Express aplikaciju (za dalju obradu HTTP zahteva) i otvara vezu ka serveru baze podataka.
 - **index.js:** Javaskript modul koji preuzima HTTP zahtev ka serveru i ima definisane rute sa tačno definisanim funkcijama za obradu zahteva ka serveru (pozivaju se funkcije iz modula za kontrolu korisnika (UserController.js) odnosno vožnji (rideController.js)), vrši obradu grešaka i opslužuje grešku 404 (resurs nije pronađen);
 - **UserController.js:** Javaskript modul za opsluživanje svih funkcija u vezi sa

korisnikom. Implementirane su sledeće funkcije:

- funkcija za slanje verifikacionog umejla korisniku koji se registruje i upis informacije o stanju verifikovanosti u bazu podataka
- funkcija za slanje umejla za promenu šifre/resetovanje zaboravljene šifre korisnika i upis njene heš vrednosti u bazu podataka
- funkcija za proveru verifikovanosti korisnika
- funkcija koja proverava da li je isti korisnik autentifikovan (preko sesijskog kolačića)
- funkcija za odjavu korisnika sa sistema
- funkcija za pravljenje novog korisnika i njegovu prijavu na sistem
- funkcija koja kao povratnu vrednost daje spisak svih korisnika prema parametru `_id`
- funkcija koja vrši izmenu podataka korisnika u bazi podataka prema parametru `_id`
- **rideController.js:** Javaskript modul za opsluživanje svih funkcija u vezi sa vožnjama/putovanjima korisnika. Implementirane su sledeće funkcije:
 - funkcija koja vraća spisak svih vožnji sa svim pripadajućim podacima iz baze podataka
 - funkcija koja vraća vožnju iz baze podataka prema parametru `_id`
 - funkcija za izmenu vožnje u bazi podataka prema paramteru `_id`
 - funkcija za pravljenje nove vožnje i njen upis u bazu podataka
 - funkcija za brisanje vožnje iz baze podataka
- **Izrađena baza podataka:** Izrađena baza podataka prema modelima definisanim u fajlovima `user.js` i `ride.js`. Baza podataka popunjena ispitnim tj. probnim podacima kako bi se koristili prilikom daljeg razvoja veb aplikacije. Baza podataka sadrži dve kolekcije:
 - `users` – čuvaju se podaci o korisnicima veb aplikacije
 - `rides` – čuvaju se podaci o vožnjama korisnika
- **Testiranje sistema:** Napisane Javaskript test skripte za sve komponente. Testirane sve komponente aplikacije. Za pisanje skripti za testiranje iskorišćena Adapter Javaskript biblioteka koja opslužuje komunikaciju između komponenti ili proverava ispravnost rada pojedinačne komponente preko očekivanih vraćenih vrednosti.

2.3 Detaljan plan projekta

ID	Task Mode	Task Name	Duration	Start	Finish	Resource Names	Rezultat aktivnosti	May
1		Veb aplikacija Putnik	49 days	Mon 01.04.19	Wed 05.06.19			
2		Analiza zahteva	5 days	Mon 01.04.19	Fri 05.04.19			
3		Modelovanje ponašanja	2 days	Mon 01.04.19	Tue 02.04.19		Izrađen dijagram slučajeva korišćenja, dijagram komponenata i dijagram	
4		Izrada specifikacije zahteva	3 days	Wed 03.04.19	Fri 05.04.19		Izrađen dokument specifikacije zahteva	
5		Projekovanje sistema	3 days	Fri 10.05.19	Tue 14.05.19			
6		Izrada modela baze podataka	2 days	Fri 10.05.19	Mon 13.05.19		Izrađen model MongoDB baze podataka	
7		KT - Model baze podataka	0 days				Napravljen model baze podataka koji sadrži sve entitete, attribute kao i njihove tipove i ograničenja	
8		Odabir stila projektovanja	1 day	Tue 14.05.19	Tue 14.05.19		Moduli sistema organizovani prema MERN arhitekturi i OO pristupu	
9		Izrada softverskog rešenja	31 days	Tue 14.05.19	Mon 24.06.19			
10		Izrada korisničkog sučelja	11 days	Tue 14.05.19	Mon 27.05.19		Izrađen korisnički interfejs aplikacije sa svim pripadajućim elementima	
11		KT - Izrađeno korisničko sučelje	0 days				Razvijena osnovna verzija sučelja (registracija, prijava na sistem, pregled i rezervacija vožnji)	
12		Izrada serverske strane aplikacije	14 days	Tue 28.05.19	Fri 14.06.19		Napravljena serverska strana veb aplikacije sa svim pripadajućim serverskim aplikacijama	
13		KT - Izrađena serverska strana aplikacije	0 days				Realizovan server - rukovanje POST, GET, PUT i DELETE HTTP zahtevima, userController, RidesController	
14		Izrada baze podataka i njeno popunjavanje probnim podacima	2 days	Wed 19.06.19	Thu 20.06.19			
15		KT - Izrađena baza podataka	0 days				Napravljena baza podataka sa unetim probnim podacima radi korišćenja u toku razvoja aplikacije	
16		Verifikacija sistema	2 days	Fri 21.06.19	Mon 24.06.19		Verifikovan odnosno evaluiran sistem koji ispunjava specifikacijske zahteve	
17		Testiranje sistema	2 days	Tue 25.06.19	Wed 26.06.19			
18		Jedinično testiranje (metod crne kutije)	1 day	Tue 25.06.19	Tue 25.06.19		Niz funkcionalnih modula veb aplikacije	
19		Integraciono testiranje (od dna ka vrhu)	1 day	Wed 26.06.19	Wed 26.06.19		Funkcionalna, integrisana i završena veb aplikacija	
20		KT - Testiranje	0 days				Sistem testiran Javascript modulima za testiranje	
21		Izrada korisničkog uputstva	1 day	Thu 27.06.19	Thu 27.06.19		Precizno i lako razumljivo korisničko uputstvo za korišćenje veb aplikacije	
22		Izrada ugovora o održavanju i unapređivanju sistema	2 days	Fri 28.06.19	Mon 01.07.19		Definisan ugovor o održavanju i unapređivanju sistema	
23		Isporučka sistema	2 days	Mon 01.07.19	Tue 02.07.19		Isporučen i implementiran sistem spreman za realno radno okruženje	

3 Analiza zahteva

Tražena aplikacija treba da sadrži sve elemente koji su neophodni za uzajamnu interakciju između vozača i putnika, kako bi se planiranje, rezervisanje i ostvarivanje željene vožnje izvršavalo na efikasan način. Svako od prijavljenih korisnika može da rezerviše i ponudi vožnju. Postoje korisnički nalozi za članove, u kojima su smešteni svi podaci koji su neophodni i koji garantuju jedinstvene korisničke profile kako bi se komunikacija između dveju strana odvijala jednostavno i precizno. Nakon registracije, sama prijava na sistem treba da bude jednostavna i intuitivna, tako da se za kasnija prijavljivanja koriste samo imejl adresa i korisnička lozinka.

Nakon detaljne analize zahteva koji karakterišu funkcionalnost aplikacije, sačinjen je spisak koji sadrži sve module koji korisniku treba da omoguće:

- pregled i izmenu sopstvenog profila
- pretragu ponuđenih vožnji i njihovu rezervaciju
- komunikaciju sa prevoznikom slanjem poruke putem formulara u okviru aplikacije
- postojanje osnovne i napredne pretrage
 - osnovna pretraga podrazumeva unos naziva ulice i izlistavanje svih ulica koje počinju tim slovima
 - napredna pretraga koja podrazumeva mogućnost sortiranja vožnje po polazištu, odredištu, datumu i po vremenskom periodu
- mogućnost zahtevanja vožnje
- da ima pregled svojih ponuđenih i rezervisanih vožnji i informaciju o tome da li je neko zahtevao dodatne uslove putem slanja poruke u polju napomena
- zakazivanje vožnje i rezervisanje mesta za sedenje
- mogućnost otkazivanja zakazane vožnje
- da ponudi vožnju putem odgovarajućeg formulara gde treba upisati broj mesta, cenu, vreme i datum polaska, odredište i polazište
- da ponuđenu vožnju može da otkáže i izmeni

4 Projektovanje sistema

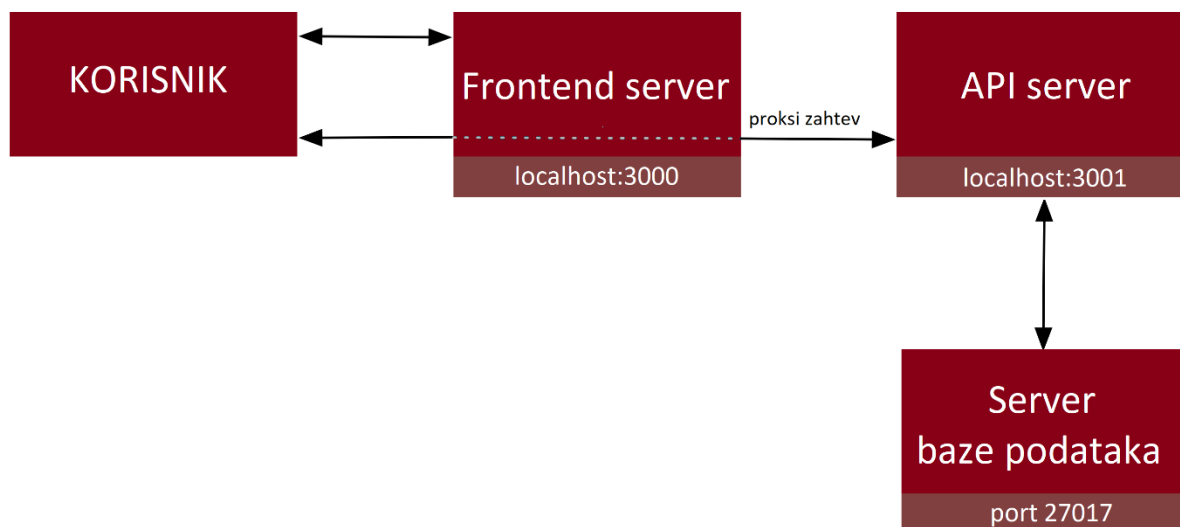
4.1 Arhitektura sistema

Sistem se sastoji od tri servera i ima troslojnu arhitekturu:

1. Server koji opslužuje klijentski deo veb aplikacije (frontend deo) koji funkcioniše na portu 3000
2. API server koji predstavlja logički deo veb aplikacije (bekend deo) i funkcioniše na portu 3001
3. Server baze podataka (mongod proces) koji funkcioniše na portu 27017

Sva tri servera su podešena da rade na localhost adresi odnosno u mreži lokalnog domena 127.0.0.1.

Kada korisnik pristupi veb aplikaciji na adresi <http://localhost:3000> njegov zahtev opslužuje frontend server (razvojni server) koji zatim sve dalje zahteve (prijava na sistem, registracija, pretraga vožnji) prosleđuje kao proksi zahtev ka API serveru na adresi <http://localhost:3001>. Na portu 3001 se primljeni zahtev obrađuje u skladu sa unutrašnjom logikom API servera (ovo je backend server). Ukoliko je neophodan pristup bazi podataka, njoj je moguć pristup samo putem API servera. Na ovaj način izvršeno je razdvajanje baze podataka od sloja logike i time je značajno povećana bezbednost sistema.



Slika 1: Prikaz osnovne arhitekture sistema

Ovakav pristup prilikom projektovanja sistema je odbaran kako bi se izbegla zabrana veb pregledača koja se tiče preuzimanja resursa sa različitih izvora (Cross-Origin Resource Sharing, skraćeno CORS). Ovakva situacija nastaje onda kada se resursi koji nisu dostupni (npr. Javaskript kod) a koji se nalaze na veb stranici, zahtevaju od drugog domena koji je izvan onog iz kog je zahtev za resursom potekao. Pošto ovakav mehanizam nije dozvoljen tj. veb pregledači ga ne dopuštaju, iskorišćena je gorepomenuta struktura koja proksi zahtevima rešava ovo ograničenje.

4.2 Programski jezik i obrazloženje

Za implementaciju softverskog dela veb aplikacije izabrane su biblioteke odnosno platforma (framework) zasnovana na Javaskript (ES6) programskom jeziku. Ovaj programski jezik je odabran zato što ima veoma veliku primenu u izradi veb aplikacija i široku programersku zajednicu koja ga aktivno koristi. Sa druge strane, postoji veliki broj biblioteka otvorenog koda napisanih u Javaskriptu koje se redovno održavaju, pri čemu vredi napomenuti da i svetski poznate softverske kompanije koje se bave razvojem veb aplikacija i sajtova aktivno koriste Javaskript. Zbog toga, Javaskript biblioteke su odlično i veoma često detaljno dokumentovane što veoma olakšava razvoj veb aplikacija i sajtova. Upravo zbog svoje popularnosti i dokazanog kvaliteta, Javaskript se često može naći kao osnova visoko specijalizovanih platformi (frameworks) pa se na taj način visoko kompleksni projekti umnogome pojednostavljaju jer postoji bazni programski jezik koji sačinjava osnov ekosistema za razvoj veb aplikacija. Za ovaj projekat odabrana je MERN (Mongodb Express React Node) softverska (bibliotečka) arhitektura. Navešćemo redom prednosti kao i osobine svake pojedine komponente arhitekture:

1. React

React je Javaskript biblioteka otvorenog koda koju održava Fejsbuk i koristi se za stvaranje pogleda (views) koji se grade (render) u HTML-u. Sa obzirom na to da React nije platforma (framework) već biblioteka, ona ne traži izričito da se koristi MVC (Model View Controller) arhitektura. Programerima je ostavljen izbor i implementacija arhitekture. Dakle React može da se koristi samo za komponentu pogleda (View) u MVC arhitekturi a ostale komponente se izvode samostalno. Ipak, u ovom projektu se koristi MVC arhitektura, s obzirom na to da je ovakva arhitektura pokazala dobre performanse u praksi.

2. Node.js

Node.js je Javaskript biblioteka i predstavlja Javaskript „izvan” pregledača. Node.js ima mogućnost (zbog postojanja unutrašnjeg mehanizma) samostalnog učitavanja više Javaskript modula bez upotrebe HTML stranice. Korišćen je za implementaciju servera i serverskih aplikacija. Koristi asinhroni model, zasnovan na događajima (event driven), bez blokiranja U/I-ja kako bi postigao paralelizam. Treba naglasiti da je u okviru ovog sloja korišćen i Webpack alat za modularizaciju koda.

3. Express

Express je rantajm okruženje koje može da pokreće Javaskript. Koristi se kako bi se pojednostavilo pisanje programskog koda za server i takođe predstavlja platformu (framework). Omogućuje programeru da definiše putanje (routes) kojima se definiše koji deo programske logike obrađuje određene HTTP zahteve.

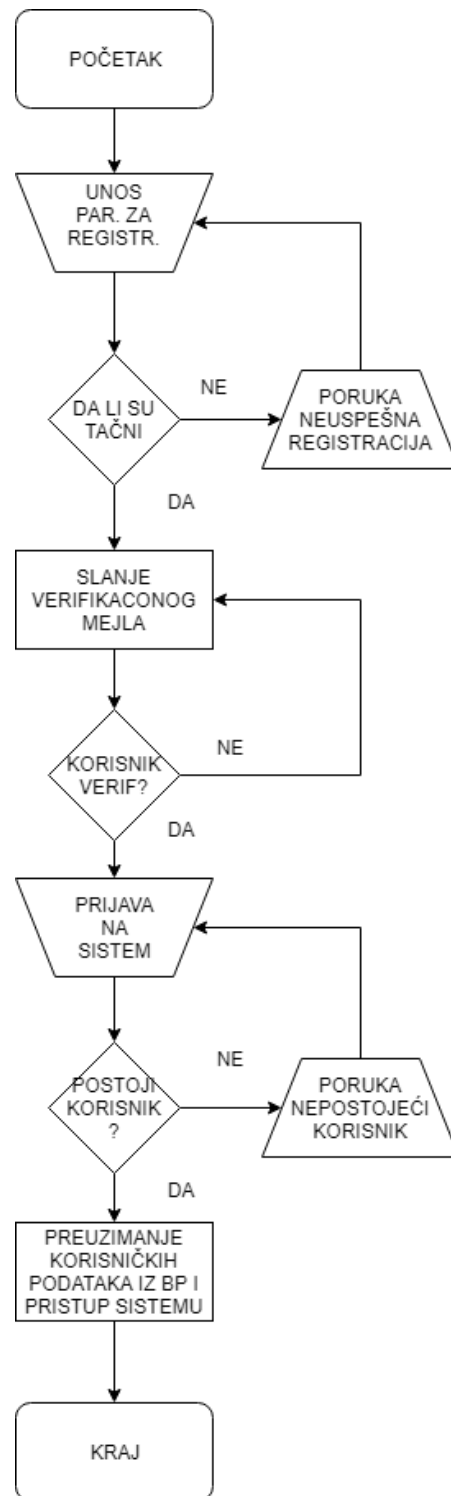
4. MongoDB

MongoDB predstavlja sistem za upravljanje nerelacionim bazama podataka. Ovakve baze podataka se odlikuju fleksibilnim šemama i korišćenju JSON zasnovanog jezika upita. NoSQL baze podataka su orijentisane ka dokumentima umesto ka relacijama i tabelama koje se sreću u klasičnim relacionim bazama podataka. Zbog toga, ovakve baze podataka mogu se veoma uspešno koristiti u veb aplikacijama koje zahtevaju veliku brzinu i veliki broj upisa odnosno čitanja iz baze podataka.

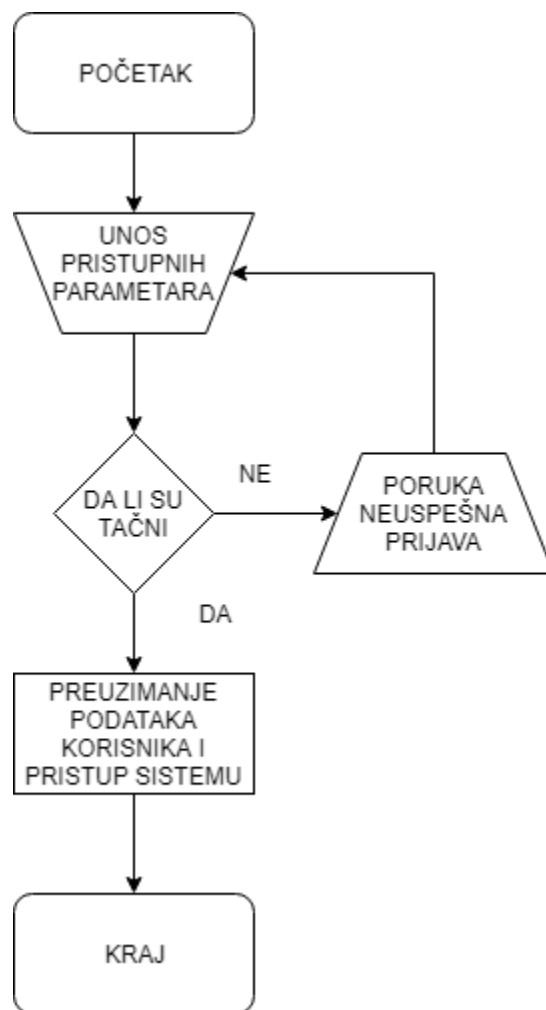
Za implementaciju klijentskog dela veb aplikacije odabrani su uobičajeni jezici (ili markap jezici) koji se tradicionalno koriste prilikom izrade veb interfejsa. Čini ih grupa koja se satoji iz HTML-a, CSS-a i Javaskripta. Međutim, ovo su samo bazni jezici, a njihovo generisanje vrši se uz pomoć React-a i JSX-a (JavaScript eXtension). JSX predstavlja komponentu tj. proširenje (extension) React-a koje programerima omogućuje da pišu Javaskript kod koji liči na HTML. Na ovaj način vrši se dinamička izmena HTML stranice direktno u klijentskom veb pregledaču bez potrebe za ponovnim uspostavljanjem veze sa serverom čime se takozvani „prazan hod” višestruko umanjuje.

4.3 Opis algoritama

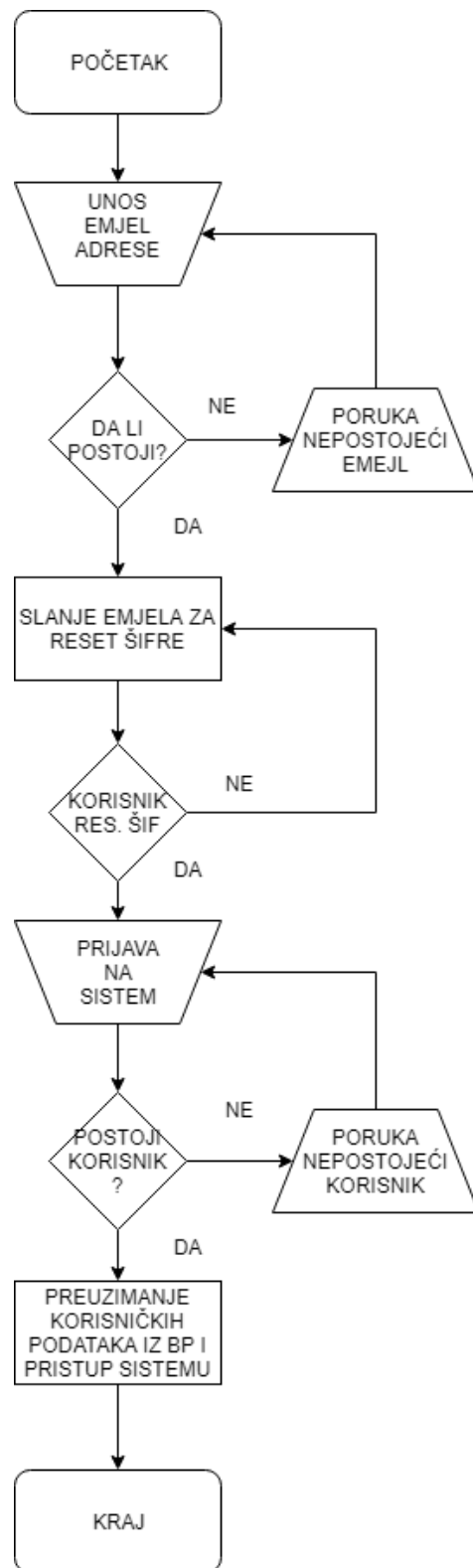
Slede blok šeme pet algoritama u njihovoj osnovnoj formi koji prikazuju algoritamsko funkcionisanje najvažnijih delova aplikacije:



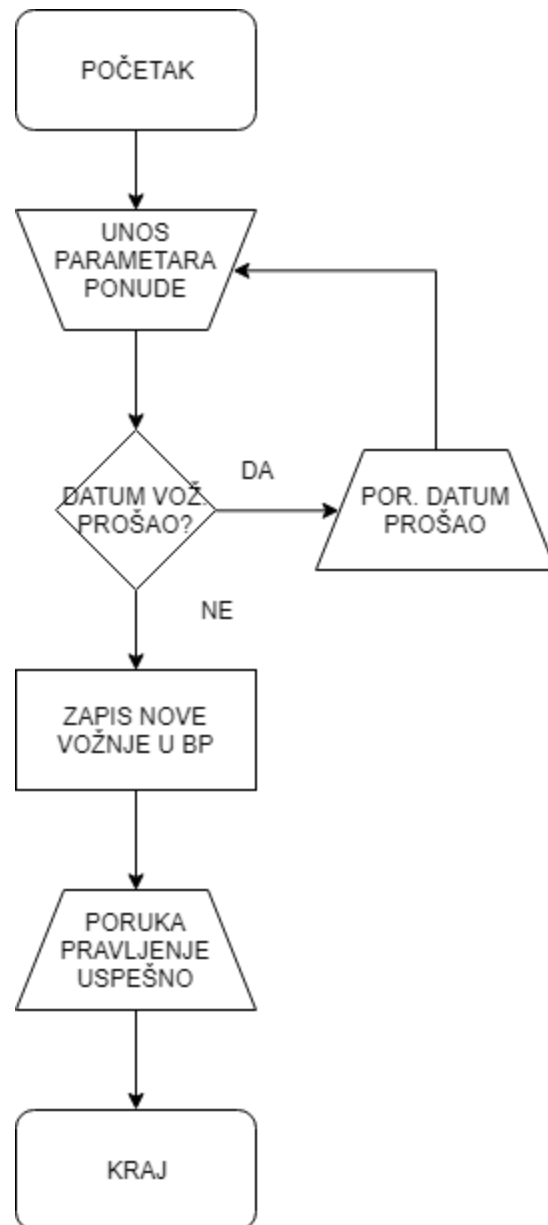
Slika 2: Registracija novog korisnika



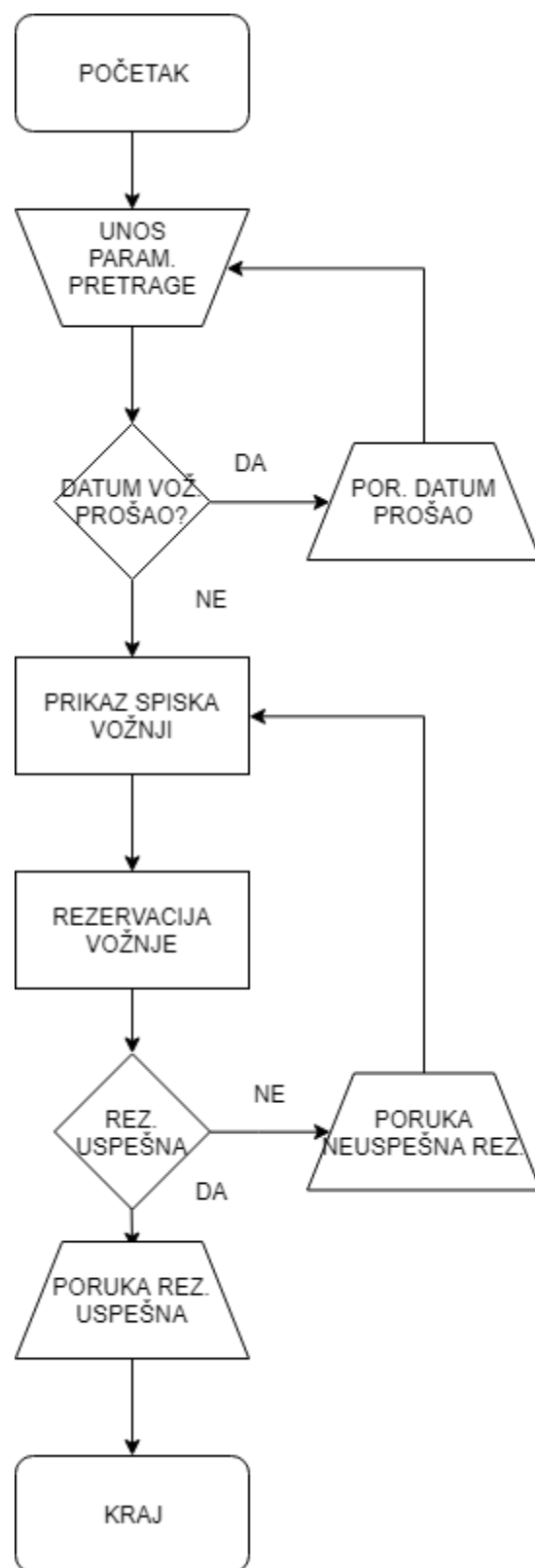
Slika 3: Prijava korisnika na sistem



Slika 4: Zaboravljena šifra



Slika 5: Ponuda nove vožnje

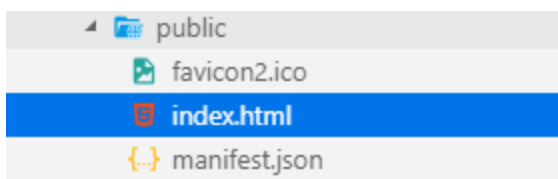


Slika 6: Traženje vožnje

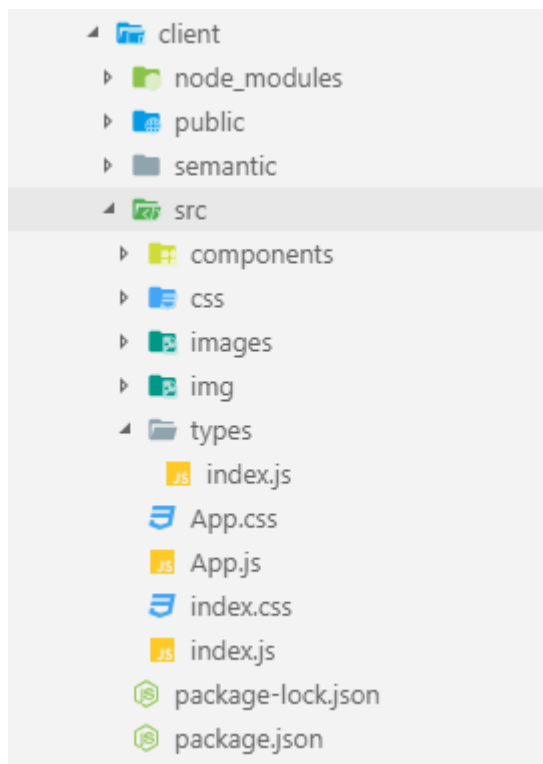
4.4 Struktura programskih modula

Veb aplikacija je strukturno podeljena na dve zasebne celine:

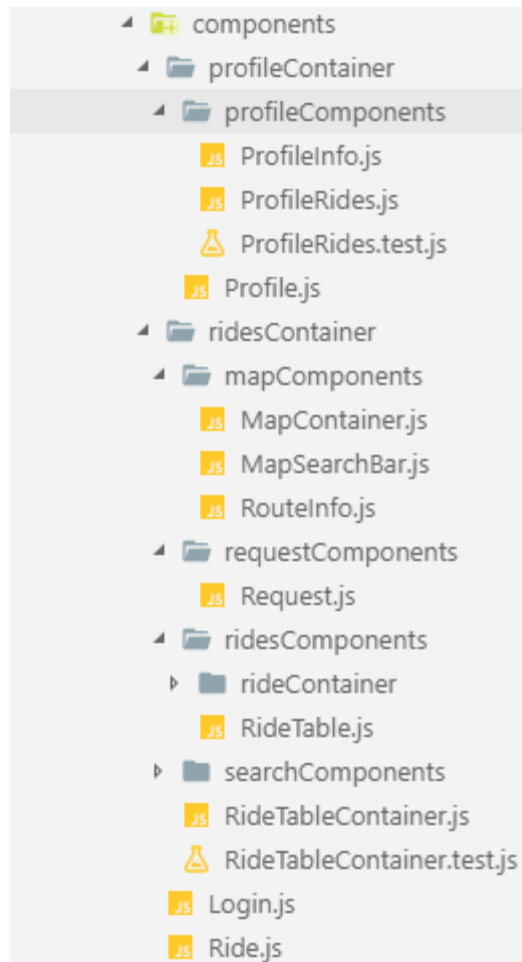
1. Klijentski deo aplikacije koji se sastoji iz 24 modula. Sledi prikaz hijerarhije modula u klijentskom delu aplikacije:



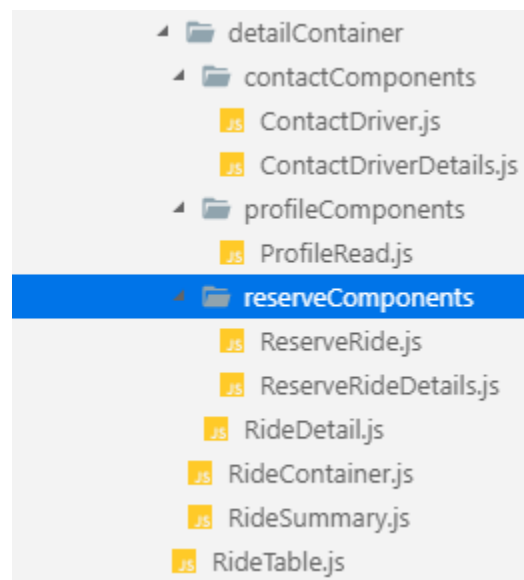
Slika 7: Sadržaj foldera Public



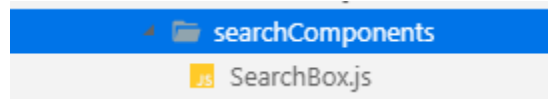
Slika 8: Sadržaj foldera src u kome su smeštene komponente aplikacije i CSS datoteke



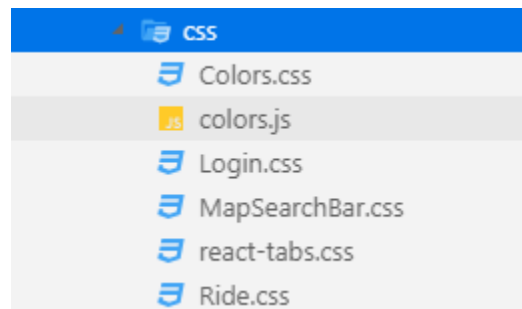
Slika 9: Hijerarhija direktorijuma components



Slika 10: Hijerarhija modula za rezervaciju reserveComponents



Slika 11: Komponenta za pretragu



Slika 12: Sadržaj direktorijuma css

Klijentski deo aplikacije je organizovan modularno gde svaki modul upravlja određenim skupom funkcionalnosti aplikacije. Sledi spisak svih komponenata, objašnjenje njihove funkcije kao i navođenje svih funkcija u okviru modula sa opisom uloge funkcija:

1. **package.json i package-lock.json:** Datoteka koja sadrži spisak svih neophodnih biblioteka koje treba da budu instalirane u razvojnom okruženju (dependencies) pre pravljenja modula aplikacije. Takođe, postoji informacija o verziji svake instalirane biblioteke.
2. **index.html:** U ovaj fajl React dinamički smešta sve HTML elemente i resurse koji će biti prikazani korisniku. Automatski se menja. Ručno se jedino mogu podesiti naslov kartice pregledača, naziv datoteke slike za ikonicu kartice pregledača, meta tagovi, dodatni veb fontovi i moduli za praćenje analitike veb aplikacije.
3. **manifest.json:** Neophodan fajl za specifikaciju metapodataka React aplikacije.
4. **Profile.js:** Komponenta profila koja:
 - a. Ispisuje informacije o korisniku koji je trenutno prijavljen na sistem
 - b. Omogućuje korisniku da izmeni informacije o svom profilu
 - c. Ispisuje trenutne ponude vožnji i zahteve za vožnjama prijavljenog korisnika
 Sadrži sledeće funkcije:
 - **handleDeleteRide:** šalje DELETE HTTP zahtev ka serveru sa `_id` parametrom vožnje za brisanje iz baze podataka i ponovo izlistava spisak vožnji bez one koja je obrisana
 - **handleCancelReservation:** šalje PUT HTTP zahtev serveru sa `_id` parametrom vožnje za koju je potrebno otkazati rezervaciju mesta i vrši izlistavanje rezervisanih vožnji bez prethodno rezervisane vožnje
 - **render():** funkcija koja dinamički prikazuje formu sa spiskom vožnji korisnika

5. **ProfileInfo.js:** Komponenta profila koja se poziva u Profile.js. Ispisuje informacije profila korisnika koji je trenutno prijavljen na sistem, omogućuje mu izmenu informacija o sebi i ispisuje ponude i zahteve za vožnjama korisnika. Sadrži sledeće funkcije:
 - a. `saveUserInfo`: koristi se za inicijalizaciju slušača asinhronih događaja (event listener) koji nastaju promenama vrednosti polja u formi. Proverava da li je uneti broj mobinog telefona u ispravnom obliku za Srbiju (ako nije, ispisuje poruku o tome korisniku) i šalje HTTP PUT zahtev serveru za smeštanje novih podataka iz forme u bazu podatka (kolekcija `user` za poslati `_id`)
 - b. `validatePhoneNumber`: funkcija koja koristi Gugl biblioteku otvorenog koda za proveravanje ispravnosti broja telefona
 - c. `render()`: funkcija koja dinamički prikazuje formu sa poljima u kojima su navedeni podaci korisničkog profila
6. **ProfileRides.js:** Komponenta profila koja ispisuje ponude i zahteve za vožnjama korisnika i nudi korisniku mogućnost da ih obriše. Poziva se u Profile.js. Sadrži sledeće funkcije:
 - a. `setCurrent`: funkcija koja beleži `_id` vožnje ukoliko korisnik klikne na neku u ponuđenoj listi
 - b. `render()`: Dinamički ispisuje listu vožnji i obnavlja listu ukoliko korisnik obriše neku od vožnji
7. **MapContainer.js:** Komponenta koja implementira API Gugl karti. Komponenta dinamički gradi (renderuje) Gugl kartu i iscrtava putanju između polazišta i odredišta. Omogućeno je i dodavanje usputnih stajališta. Karta podržava uveličavanje, smanjivanje, prevlačenje i ponovno postavljanje u centar (izabran grad Beograd). Sadrži sledeće funkcije:
 - a. `getTotalDistance`: Kao rezultat vraća ukupnu dužinu putanje za odabranu vožnju u kilometrima.
 - b. `getHours`: Kao rezultat vraća vreme putovanja izraženo u satima i minutima.
 - c. `findCenter`: Postavlja centar karte na koordinate (44.8082451, 20.4542757) kada korisnik to zatraži.
 - d. `showDirections`: Prikazuje putanju na karti između polazišta i odredišta.
 - e. `render()`: Dinamički prikazuje Gugl kartu na desnoj polovini korisničke aplikacije.
8. **MapSearchBar.js:** Komponenta koja pristupa Guglovom API-ju za automatsko dovršavanje naziva ulica prilikom pretrage polazišta odnosno dolazišta. Pritom, prikazuje u padajućem meniju predloge mesta sa geografskim koordinatama kako bi one mogle da se dalje proslede aplikaciji na obradu. Sadrži sledeće funkcije:
 - a. `handleSelect`: Unosi odabranu vrednost iz padajućeg menija i podešava vrednost adresnog polja u formi za izbor odredišta/polazišta
 - b. `geocodeByAddress`: Prosleđuje geografske koordinate polazišta/odredišta aplikaciji

- c. `render()`: Dinamički prikazuje promene u kontejneru karte u korisničkom interfejsu
9. **RouteInfo.js**: Upravlja ispisom vremena putovanja i dužine putovanja i ispisuje u pravougaonom polju ispod Gugl karte kao informaciju putniku/vozaču. Sadrži sledeću funkciju:
- a. `RouteInfo`: Vraća informacije o putanji (dužina trajanja, vreme putovanja)
10. **Request.js**: Modul koji omogućuje korisniku da zahteva/postavi novu vožnju ili da odgovori na zahtev za vožnjom drugom korisniku. Poziva se u `RideDetail.js` i `RideContainer.js`. Sadrži sledeće funkcije:
- a. `handleSave`: Funkcija nudi unos parametara nove vožnje i proverava njihovu ispravnost. Ukoliko neki od parametara ima nedozvoljenu vrednost funkcija sprečava njen unos i o tome obaveštava korisnika. Parametri su: `type`, `fulfilled`, `from`, `fromgeo`, `to`, `togeo`, `time`, `date`, `passengers`, `seats_avail`, `notes`, `price` i `user`.
 - b. `updateRide`: Šalje serveru HTTP PUT zahtev sa podacima vožnje koju treba izmeniti
 - c. `handleNewRide`: Ako je korisnik postavio novu vožnju, modul šalje serveru HTTP POST zahtev u čijem telu (body) se nalazi JSON objekat koji sadrži parametre nove vožnje za upis u bazu podataka.
 - d. `render()`: Dinamički prikazuje formu sa poljima za unos i prikaz parametara nove vožnje i prikazuje dugmad za čuvanje unosa ili otkazivanje istog.
11. **ContactDriver.js**: Komponenta koja omogućuje korisniku da kontaktira imejlom korisnika koji je postavio vožnju ili zahtev za vožnjom. Poziva se u `RideDetail.js`. Sadrži sledeće funkcije:
- a. `handleContactDriver`: Definiše telo imejla u kojem obaveštava korisnika da mu je neko poslao poruku i unosi sadržaj poruke. Poziva funkciju `sendEmail` kako bi prosledila telo poruke.
 - b. `sendEmail`: Kao parametar dobija telo imejla, podešava naslov imejla, ka kome se šalje, ko šalje poruku i pravi POST HTTP zahtev ka serveru u čijem telu se nalazi JSON objekat koji sadrži imejl. Čeka odgovor od servera, ukoliko se pojavi greška ispisuje je u komandnom prozoru razvojnog okruženja a ako sve prođe kako treba ispisuje korisniku obaveštenje o tome.
 - c. `render()`: Dinamički prikazuje formu sa poljima za rezervaciju/kontaktiranje vozača/putnika.
12. **ContactDriverDetails.js**: Komponenta zadužena za prikaz detalja vožnje i tekstualnog polja za slanje poruke korisniku. Poziva se u `ContactDriver.js` i sadrži sledeću funkciju:
- a. `render()`: Dinamički prikazuje dva tipa teksta na dugmetu za slanje poruke u zavisnosti od toga da li se poruka šalje putniku ili vozaču. Prikazuje dugmad za slanje poruke, otkazivanje slanja poruke i tekstualno polje. Takođe prikazuje

detalje vožnje i nudi mogućnost pregleda profila vozača ili korisnika koji je postavio/zahteva vožnju.

13. **ReserveRide.js**: Komponenta zadužena za rezervaciju vožnje i slanje umejla korisniku koji je postavio vožnju sa obaveštenjem o rezervaciji njegove vožnje. Poziva se u RideDetail.js. Sadrži sledeće funkcije:

- a. handleReserveRide: Kao ulazni parametar dobija napomenu (polje note) forme za registraciju tj. rezervisanje vožnje. Sastavlja telo umejla u kojem korisnika obaveštava o tome da se korisnik koji je rezervisao vožnju pridružio njegovoj vožnji i poziva funkciju za slanje umejla.
- b. sendMail: Kao parametar dobija telo umejla, podešava naslov umejla, ka kome se šalje, ko šalje poruku i pravi POST HTTP zahtev ka serveru u čijem telu se nalazi JSON objekat koji sadrži umejl. Čeka odgovor od servera, ukoliko se pojavi greška ispisuje je u komandom prozoru razvojnog okruženja a ako sve prođe kako treba ispisuje korisniku obaveštenje o tome.
- c. updateRide: Funkcija šalje HTTP PUT zahtev ka serveru sa JSON objektom u telu zahteva. JSON objekat sadrži _id parametar korisnika koji je rezervisao vožnju kako bi se pristupilo toj vožnji u bazi podataka i u polju passengers dodao _id korisnika kao indikacija sistemu koliko je još slobodnih mesta ostalo.
- d. render(): Dinamički prikazuje formu za rezervaciju vožnje kao i dugme za povratak korisnika na početnu stranu aplikacije.

14. **ReserveRideDetails.js**: Komponenta koje prikazuje detalje rezervisane vožnje. Poziva se u ReserveRide.js. Sadrži sledeću funkciju:

- a. render(): Dinamički prikazuje detalje vožnje i nudi korisniku povratak na listu rezervisanih vožnji.

15. **RideDetail.js**: Prikazuje detalje vožnje (naslov, datum, slobodna mesta, vreme polaska, cenu i napomene). Omogućuje korisniku da rezerviše ili odgovori na zahtev vožnje, kontaktira vozača i obriše ponudu/zahtev. Poziva se u RideContainer.js. Sadrži sledeće funkcije:

- a. RideDetail: Uzima kao parametar objekat props koji sadrži sva polja vožnje sa vrednostima koje su učitane iz baze podataka.
- b. return: Funkcija koja vraća HTML kod sa popunjenim vrednostima iz objekta koji je napravila funkcija RideDetail.

16. **RideContainer.js**: Modul koji povezuje rad komponentata RideDetail.js i RideSummary.js. Korisnik ga vidi kao „kontejner“ u kome se prikazuju sve informacije vezane za vožnje. Poziva se u RideTable.js.

17. **RideSummary.js**: Prikazuje sažet prikaz vožnje u listi vožnji (naslov, datum, broj slobodnih mesta). Poziva se u RideContainer.js. Sadrži sledeće funkcije:
- RideSummary: Ispisuje polazište i odredište u skraćenom tj. sažetom prikazu vožnje u listi vožnji.
 - return: Funkcija koja se poziva kada korisnik pritisne na određenu vožnju u listi vožnji. Osluškuj događaj onClick i ispisuje sažet prikaz vožnje.
18. **RideTable.js**: Za svaku vožnju stvara pravougaono polje (kontejner). Poziva se u RideTableContainer.js. Sadrži sledeću funkciju:
- RideTable: Prihvata ulazni parametar props sa detaljima vožnje i pravi objekat listRides.
19. **SearchBox.js**: Komponenta čiji je zadatak da filtrira vožnje u zavisnosti od upita koji je postavio korisnik. Može da filtrira prema reči za pretragu, vrsti vožnje (ponuda, zahtev), vremenskom periodu i da vožnje poreda prema polazištima, odredištima, ili datumu. Poziva se u RideTableContainer.js. Sadrži sledeće funkcije:
- handleBox: Opslužuje slučaj kada korisnik izabere ček polja (check box) za biranje polazišta odredišta.
 - render: Dinamički prikazuje rezultate pretrage.
20. **RideTableContainer**: Komponenta tj. kontejner (polje) koje opslužuje komponente karte i komponentu polja za pretragu vožnji. Poziva se u Ride.js. Sadrži sledeće funkcije:
- setCurrent: Kolbek funkcija koja se poziva u RideContainer.js kada korisnik klikne na ponuđenu ili zatraženu vožnju. Podešava vrednost trenutno označene vožnje.
 - render(): Dinamički prikazuje levo i desno polje korisničkog interfejsa. Levo polje sadrži spisak svih ponuđenih vožnji, polje za pretragu, dugme za pravljenje novog zahteva za vožnju i kartu. Desno polje (tj. kada se odabere kartica) sadrži spisak svih ponuđenih vožnji, dugme za pravljenje nove vožnje, polje za pretragu i kartu sa desne strane.
21. **Login.js**: Komponenta koja rukovodi odnosno opslužuje:
- Korisnikovu prijavu na sistem.
 - Korisnikovu odjavu sa sistema.
 - Registraciju novog korisnika.
 - Resetovanje korisničke pristupne šifre.
 - Promenu korisničke pristupne šifre.
- Sadrži sledeće funkcije:
- handleRegister: Kao ulazni parametar uzima objekat događaja. Događaj može biti promena vrednosti polja za unos parametara za registraciju. Nakon toga pravi objekat newUser u koji upisuje vrednosti koje su asinhrono upisane u formi za

registraciju. Nakon toga pravi HTTP POST zahtev kojim u telu POST zahteva šalje u JSON formatu parametre za upis novog korisnika u bazu podataka. Čeka odgovor od servera. U slučaju greške, ispisuje je u komandnom prozoru razvojnog okruženja a u suprotnom ispisuje poruku korisniku o tome da treba da se prijavi na imejl adresu koju je naveo u registraciji i da sa priloženim linkom koji je dostavljen na njegovu imejl adresu verifikuje nalog.

- **handleLogin:** Kao ulazni parametar uzima objekat događaja. Šalje POST HTTP zahtev sa prethodno preuzetim korisničkim imenom (imejl adresom) i hešovanom šifrom i ceo objekat šalje serveru kako bi on preko komponente `UserController.js` uporedio i proverio ispravnost parametara za prijavu na sistem.
- **resetPassword:** Šalje serveru HTTP POST zahtev sa imejlom na koji treba poslati link za resetovanje šifre korisnika za pristup sistemu. Ukoliko dođe do greške ispisuje je u komandnom prozoru razvojnog okruženja, ako ne, ispisuje korisniku obaveštenje da proveriti svoj imejl nalog radi završetka procesa resetovanja šifre naloga.
- **changePass:** Komponenta koja uzima novu šifru i ponovljenu novu šifru i pravi HTTP POST zahtev ka serveru sa pomenutim parametrima u JSON formatu u telu zahteva. Ako ne dođe do greške (server nije javio nikakvu grešku u odgovoru) vraća korisnika na stranu za prijavu na sistem.
- **render():** Funkcija za dinamički prikaz svih elemenata strane za prijavu na sistem. U zavisnosti od odabrane opcije, prikazuje formu za registraciju novog člana, prijavu postojećeg člana, resetovanje šifre i promenu šifre. Sadrži definisana polja i moguće vrednosti koja ona mogu da imaju (za imejl oblik standardne imejl adrese, za šifru ograničenje minimalne dužine od 4 karaktera, za korisničko ime ograničenje ne manje od 2 karaktera). Takođe pravi polje za prikaz pozadinskog video snimka i učitava ga.

22. **Ride.js:** Glavna komponenta zadužena za prikaz spiska vožnji. Uzima sve vožnje sa servera, i prikazuje samo one koje nisu istekle (poredi sistemski datum servera i datumsko (`date`) polje za svaku vožnju). Takođe uzima informacije o trenutno prijavljenom korisniku sistema. Sadrži sledeće funkcije:

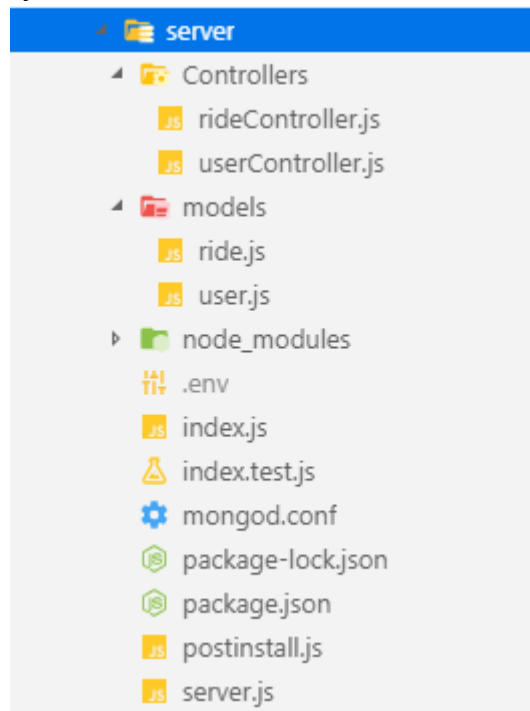
- a. **sortRides:** Raspoređuje prikaz vožnji tako što prvo prikazuje vožnje koje su ranije postavljene (pre ističu).
- b. **handleDeleteRide:** Kao ulazni parametar dobija `_id` vožnje za brisanje iz baze podataka. Pravi DELETE HTTP zahtev ka serveru i prosleđuje (u okviru linka tj. rute) `_id` vožnje za brisanje. Ako je odgovor servera potvrđan u smislu brisanja vožnje, ponovo izlistava nove vožnje bez one koja je obrisana.
- c. **handleCancelReservation:** Funkcija koja kao ulazni parametar uzima objekat `newRide` i koja šalje HTTP PUT zahtev serveru sa parametrom `newRide._id` kako bi server u bazi podataka mogao da otkáže rezervisanu vožnju.

- d. **notExpired:** Funkcija koja proverava da li je ulazni parametar ride (objekat vožnje) istekao (u smislu datuma).
- e. **handleLogout:** Funkcija koja se poziva kada korisnik želi da se odjavi sa sistema. Poziva funkciju `logoutSuccess`.
- f. **render():** Dinamički prikazuje vožnje koje su učitane sa servera. Gradi (renderuje) meni korisničkog interfejsa.

23. **App.js:** Glavna React komponenta koja rukovodi sesijskim kolačićem, rukuje odjavom sa sistema i definiše rute (linkove) vožnji.

24. **index.js:** Komponenta koja poziva ReactDOM renderer i u njega uvozi komponentu App.js

2. Serverski deo aplikacije se sastoji iz 8 komponentata. Sledi prikaz hijerarhije modula u serverskom delu aplikacije:



Slika 13: Prikaz hijerarhije serverskog dela aplikacije

Softverski deo aplikacije je organizovan modularno gde svaki modul upravlja određenim skupom funkcionalnosti serverskih aplikacija. Sledi spisak svih komponentata, objašnjenje njihovih funkcija kao i navođenje svih funkcija u okviru komponentata sa opisom uloga funkcija:

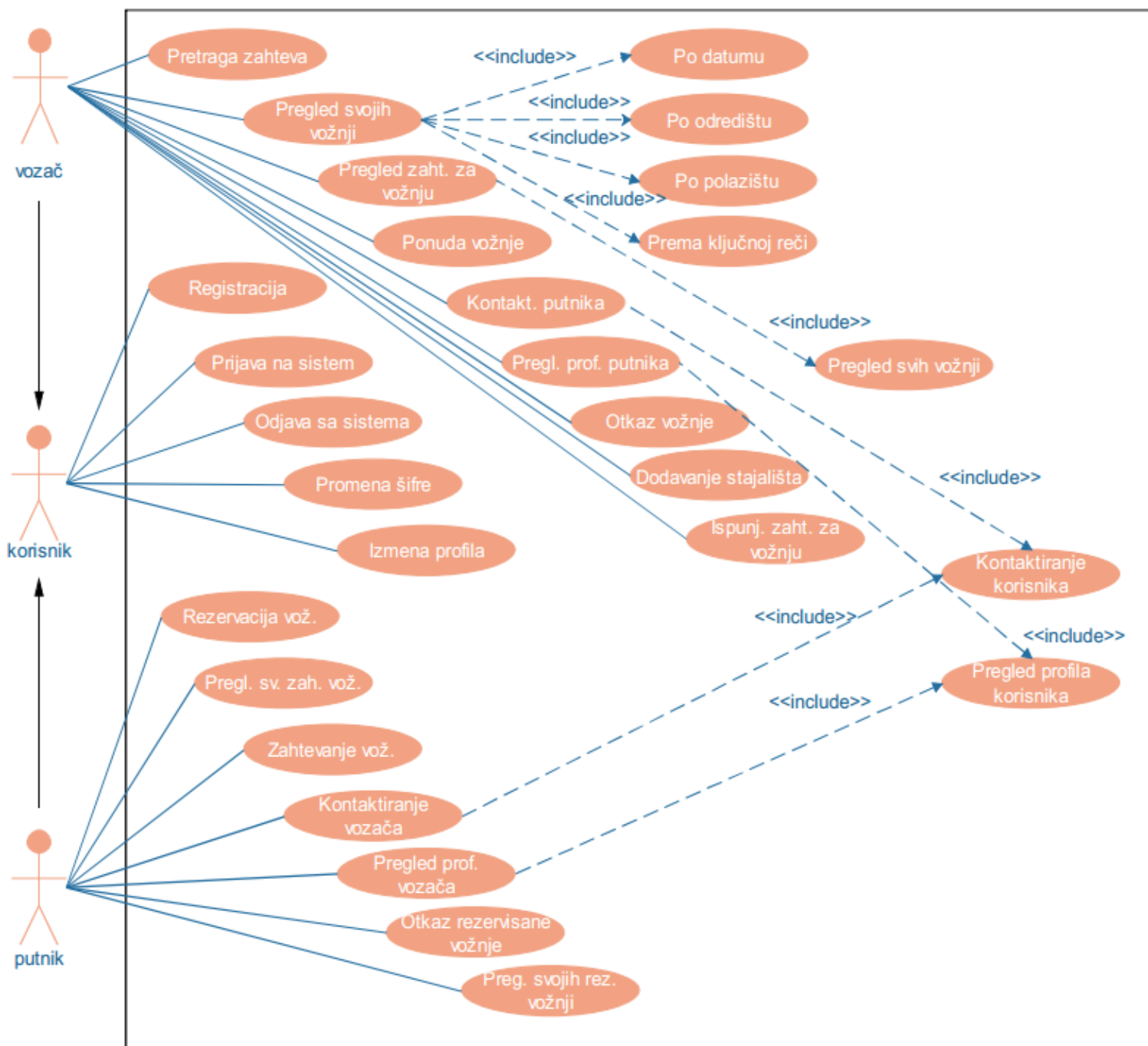
1. **package.json i package-lock.json:** Datoteka koja sadrži spisak svih neophodnih biblioteka koje treba da budu instalirane u razvojnom okruženju (dependencies) pre pravljenja modula aplikacije. Takođe, postoji informacija o verziji svake instalirane biblioteke;

2. **rideController.js:** Rukuje svim zahtevima koji se tiču vožnji. Opslužuje sledeće funkcije (ukoliko nije nastala greška): `getAllRides`, `getById`, `findByIdAndUpdate`, `createRide`, `deleteRideById`.
3. **UserController.js:** Zadužena za opsluživanje svih funkcija koje imaju veze sa korisnikom aplikacije. Sadrži sledeće funkcije:
 - a. `AccountVerificationEmail`: Zadatak ove funkcije je da korisniku pošalje verifikacioni email kako bi mogao da završi proces registracije. Šalje email i ukoliko server odgovori da je došlo do greške ispisuje je u komandnom prozoru razvojnog okruženja.
 - b. `getAllUsers`: Vraća spisak svih korisnika sistema.
 - c. `getUserById`: Vraća podatke korisnika u obliku JSON objekta za poslati parametar `_id`.
 - d. `updateUserById`: Za poslat parametar `_id` korisnika vrši izmenu vrednosti u bazi podataka.
 - e. `isAuthenticated`: Upoređuje `_id` trenutnog korisnika sa `_id` – jem korisnika iz sesijskog kolačića. Na ovaj način se sprečava da drugi neautorizovani korisnik pristupi korisničkom nalogu korisnika koji je zaboravio da se odjavi sa svog korisničkog naloga.
 - f. `createAndAuthenticateUser`: Ova funkcija proverava da li je korisnik uneo dva puta istu pristupnu šifru. Ako nije, onemogućava se dalji proces registracije. Zatim poziva funkciju `AccountVerificationEmail` za proveru verifikovnosti naloga trenutnog korisnika. Ako korisnik nije verifikovan, sistem vraća odgovarajući HTTP kod (401) ili 400 (ako nisu sva polja popunjena). Ako je registracija uspešna sistem preusmerava korisnika na njegovu početnu stranu profila.
 - g. `logout`: Briše sesijski objekat i odjavljuje korisnika sa sistema.
 - h. `verify`: Funkcija koja se izvršava kada je verifikacija korisnika uspešna. Pristupa kolekciji pod parametrom `_id` korisnika i u polje `isVerified` upisuje vrednost `true` čime se sistemu naznačava da je korisnik verifikovan. U slučaju neuspešne verifikacije (nemogućnost pristupa korisniku u bazi podataka iz bilo kog razloga) funkcija preusmerava korisnika na početnu stranu aplikacije.
 - i. `forgotPass`: Funkcija čiji je zadatak da sačini kvazislučajnu vrednost žetona za resetovanje zaboravljene korisničke šifre. Sačinjava parametar `passwordResetExpires` koji predstavlja vreme (jedan čas) za koje će žeton za promenu šifre biti validan. Nakon toga šalje email korisniku sa linkom za resetovanje šifre i žetonom za upoređivanje.
 - j. `resetPassword`: Funkcija ima zadatak da izvrši upoređivanje dveju šifara (nove šifre i nove ponovljene šifre koju korisnik unosi). Ako se ne poklapaju, šalje status 400 (šifre se ne poklapaju ili je žeton za reset šifre istekao). Ako je promena šifre uspešno izvedena, funkcija podešava parametre za slanje emaila korisniku sa obaveštenjem da je promena uspešno izvršena.

4. **ride.js**: Model kolekcije rides. Sadrži definicije polja (tipove i ograničenja) na osnovu kojih sistem pravi JSON upit kojim šalje, briše ili menja podatke vožnji u bazi podataka.
5. **user.js**: Model kolekcije users. Sadrži definicije polja (tipove i ograničenja) na osnovu kojih sistem pravi JSON upit kojim šalje, briše ili menja podatke korisnika u bazi podataka.
6. **.env**: Konfiguracioni fajl koji sadrži vrednosti konstanti koje se često pojavljuju u komponentama sistema. Kako bi se olakšala izmena tih vrednosti, njihove vrednosti se očitavaju iz ovog fajla.
7. **index.js**: Komponenta uz pomoć koje server rukuje putanjama HTTP zahteva (GET, POST, PUT i DELETE). Sadrži spisak svih funkcija koje se pozivaju kako bi se obradili klijentski zahtevi za korisnika i vožnje.
8. **server.js**: Sadrži osnovne parametre podešavanja servera (definisan tip podržanih HTTP zahteva i dozvoljena zaglavlja). Sadrži inicijalizovanu Express aplikaciju koja preuzima obradu HTTP zahteva šaljući ih na obradu index.js skripti. Takođe sadrži početna podešavanja za Express midlver (middleware) a reč je o podešavanjima u vezi sa CORS ograničenjem, kao i funkciju kojom se vrši povezivanje sa serverom baze podataka.

4.5 Dijagram slučajeva korišćenja

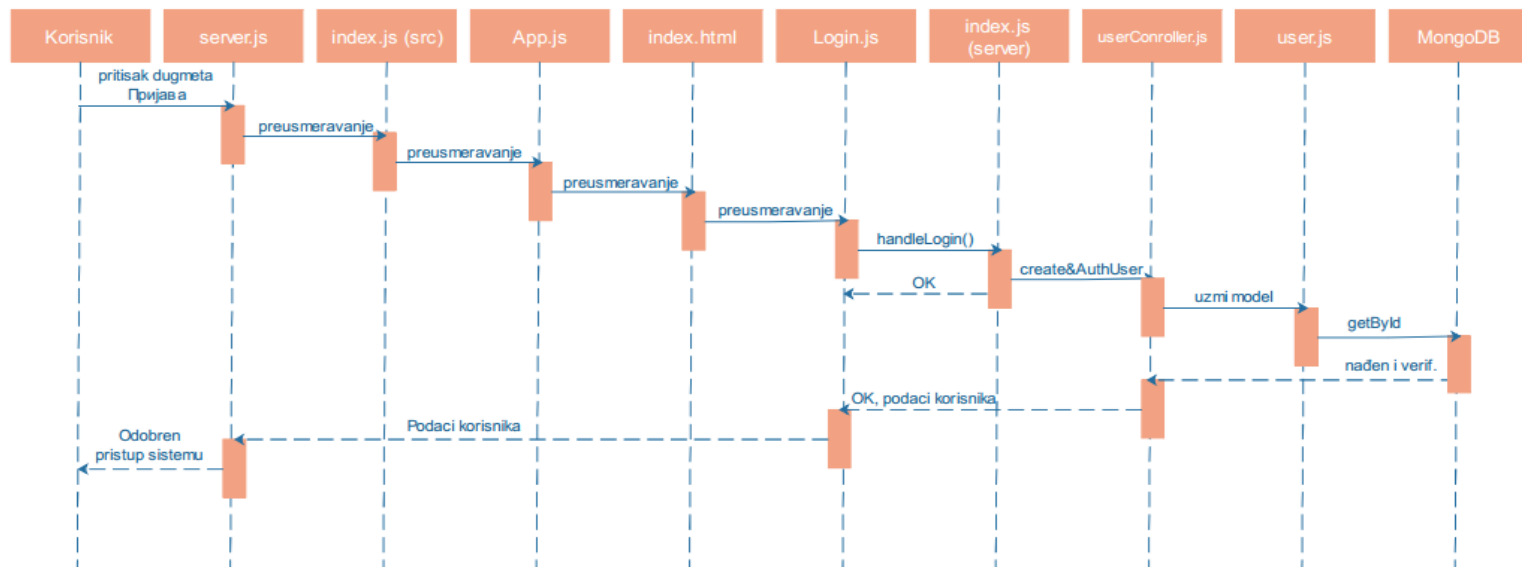
Korisnik može da se registruje na sistem, prijavi na sistem, odjavi sa sistema, promeni šifru, resetuje zaboravljenu šifru i izmeni podatke svog profila. Vozač može da učini sve ono što može i korisnik (nasleđuje korisnika) i pored toga još da ispuni zahtev putnika za vožnjom, pregleda sve zahteve za vožnju, pregleda svoje ponude vožnji, ponudi vožnju, kontaktira putnika, otkaže svoju vožnju, pregleda profil putnika, izvrši pretragu zahteva za vožnjom prema datumu, polazištu, dolazištu, i vremenu. Putnik nasleđuje ulogu korisnika i još pored toga može da izmeni profil, pregleda prfil vozača, pregleda ponuđene vožnje, rezerviše vožnju, pregleda svoje rezervisane vožnje, zahteva vožnju, kontaktira vozača i otkaže rezervaciju vožnje.



Slika 14: Dijagram slučajeva korišćenja

4.6 Dijagram sekvence

Na sledećoj slici prikazan je dijagram sekvence prijave registrovanog korisnika na sistem:

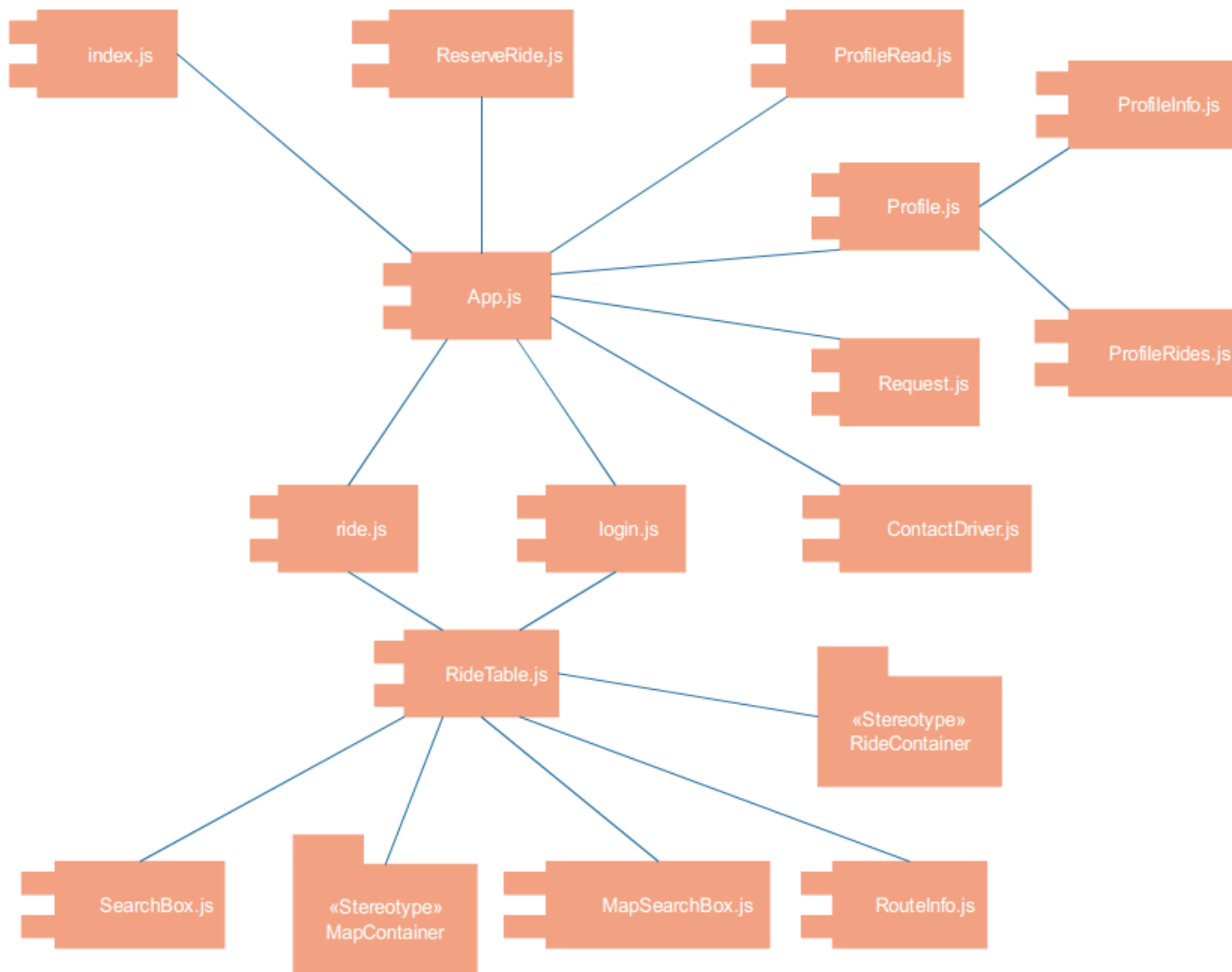


Slika 15: Prikaz dijagrama sekvence korisnika koji se prijavljuje na sistem

Korisnikov veb pregledač odlaskom na adresu veb aplikacije dobija prikaz stranice za prijavu na sistem. Nakon unosa parametara za prijavu i pritiska na dugme Пријави се, server zahtev preusmerava ka Express aplikaciji koja podatke za prijavu preusmerava ka skripti Login.js. Skripta Login.js preuzima podatke funkcijom handleLogin i šalje serveru POST HTTP zahtev sa enkapsuliranim parametrima za prijavu. Serverska komponenta zadužena za opsluživanje putanja index.js poziva funkciju createAndAuthenticateUser serverske komponente userController.js koja zatim pronalazi korisnika u bazi podataka i upoređuje parametre. Ako su ispravni, odnosno ako se heš vrednosti šifara kao i imejl adrese poklapaju (i korisnik je verifikovao svoj nalog) userController vraća podatke Login.js komponenti koja ih prosleđuje osnovnoj komponenti servera koja ponovo izgrađuje HTML (rerenderuje ga) i odobrava korisniku pristup sistemu.

4.7 Dijagram komponenata

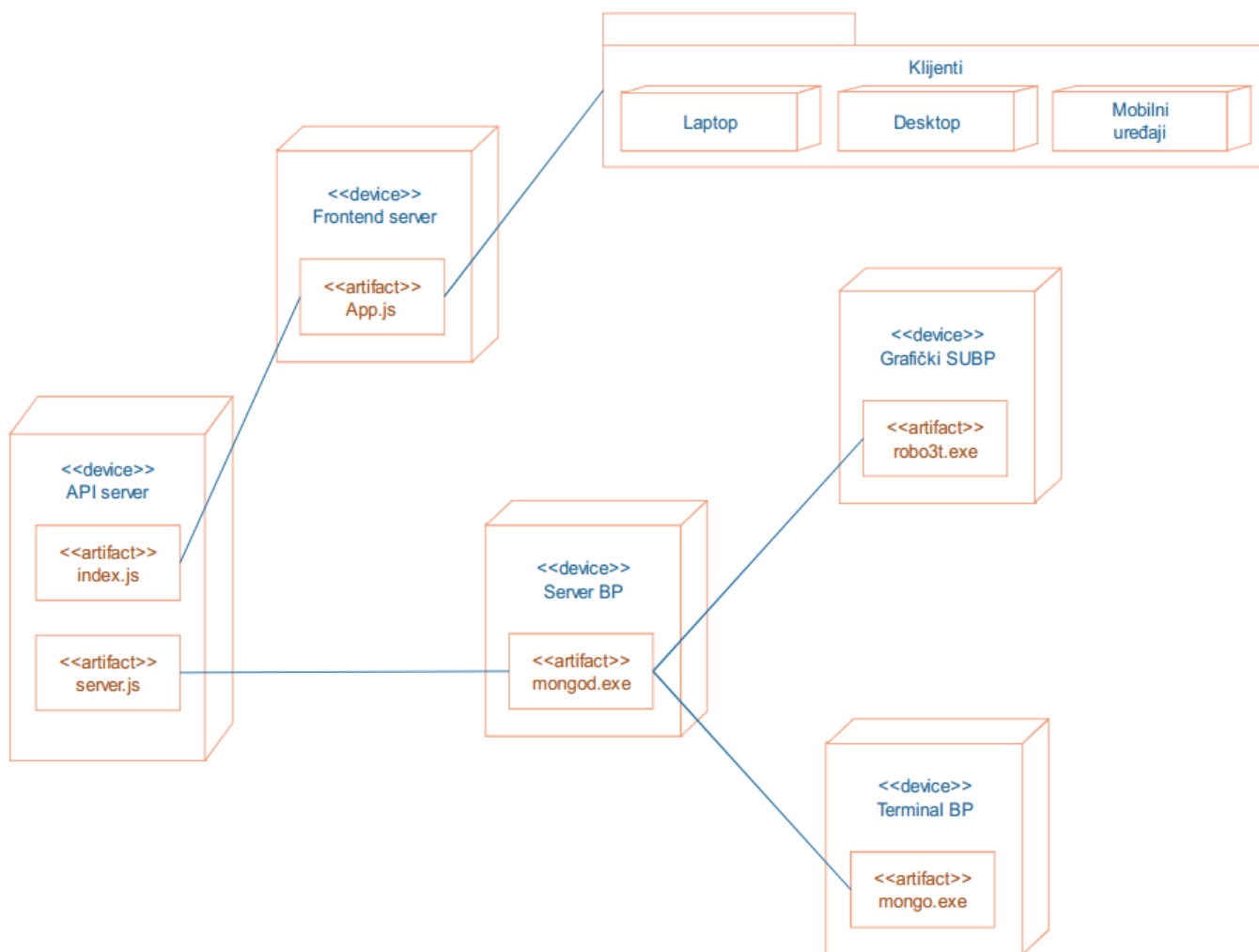
Na sledećoj slici prikazan je dijagram komponenata veb aplikacije:



Slika 16: Prikaz dijagrama komponenata veb aplikacije

4.8 Dijagram raspoređivanja

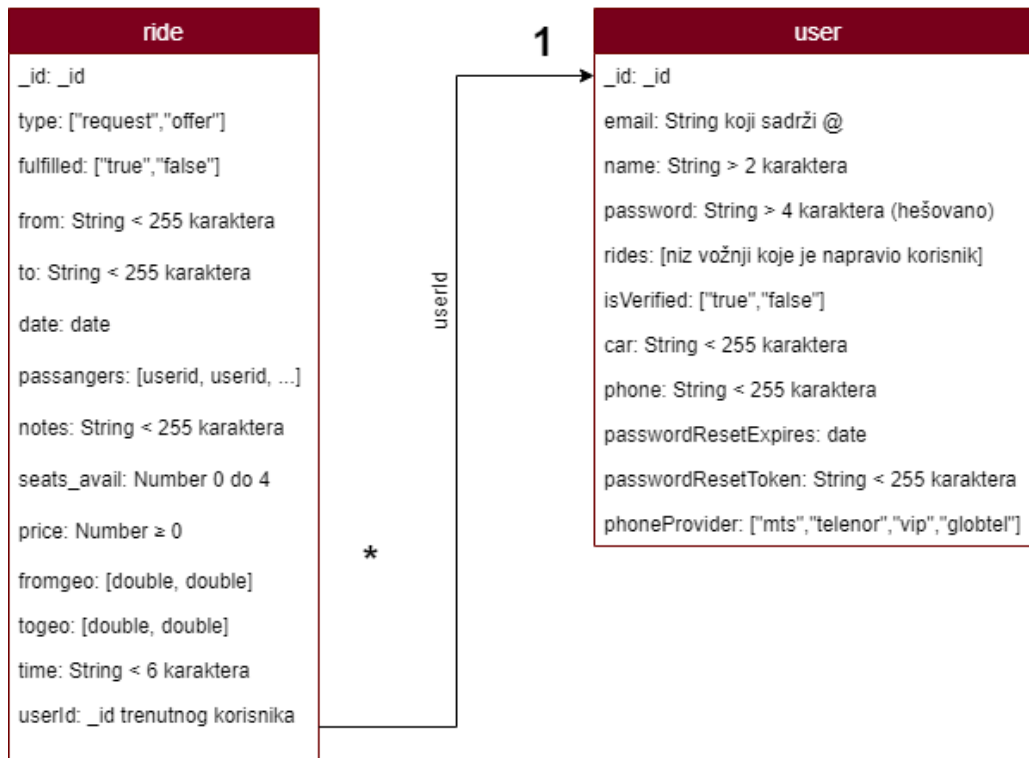
Sledi prikaz dijagrama raspoređivanja. Dijagram se sastoji od frontend servera, API servera, servera baze podatka, klijenata (laptop, desktop i mobilni uređaji) kao i dva sistema za upravljanje bazama podataka (mongo.exe – CLI i robo3t.exe - GUI):



Slika 17: Prikaz dijagrama raspoređivanja veb aplikacije

4.9 Model baze podataka

Model baze podataka aplikacije sastoji se od dve kolekcije (tabele):



Slika 18: Model baze podataka sa prikazom relacije više prema jedan

- user: U ovoj kolekciji smešteni su podaci korisnika naloga. Sastoji se od jedanaest elemenata:
 - _id: Jedinstveni identifikacioni broj korisnika u bazi podataka. Generisan je automatski od strane MongoDB sistema za upravljanje bazama podataka.
 - email: String koji čuva informaciju o imejl adresi korisnika naloga sistema. Ujedno se koristi kao korisničko ime za prijavu na sistem. Pre unosa ove vrednosti u bazu podataka i u kodu korisničkog interfejsa, kao i u kodu serverske funkcije zadužene za registraciju, nije moguće upisati imejl u bazu podataka ukoliko ne dogovara regularnom izrazu. Regularni izraz se sastoji iz provere da li se između imena i domena nalazi @ znak, kao i da li se imejl završava sa .domen delom stringa.
 - name: String koji ne može biti kraći od 2 karaktera. Služi da se u njemu sačuva ime i prezime korisnika koje će biti vidljivo ostalim korisnicima sistema. Ne koristi se kao parametar za prijavu na sistem.
 - password: String u kome se čuva hešovana vrednost korisničke pristupne šifre. Heševanje je izvršeno SHA256 algoritmom. Mora biti duža od 4 karaktera.

- rides: niz u kome se čuvaju _id vrednosti vožnji koje je korisnik napravio.
 - isVerified: Logička (boolean) vrednost koja označava da li je korisnik verifikovao svoj nalog na sistemu.
 - car: String u kome se čuva (opciona) marka ili model vozila koje korisnik poseduje;
 - phone: String u kome se čuva broj mobilnog telefona korisnika. String mora da odgovara formatu koji je uobičajen u Srbiji. Pre unosa u bazu podataka, string se šalje funkciji za proveru validnosti broja telefona i samo ako se utvrdi da je format telefona u isprvnom obliku broj telefona se kao string upisuje u bazu podataka.
 - passwordResetExpires: element tipa datuma u kome se čuva datum i vreme posle koga neće biti moguće resetovati šifru uz pomoć poslatog linka za resetovanje šifre;
 - passwordResetToken: String u kome se čuva kvazislučajno stvoren žeton (token) kojim se vrši resetovanje korisničke šifre naloga na sistemu;
 - phoneProvider: niz sa 4 elementa (nazivima mobilnih operatera u Srbiji) koje korisnik može da izabere.
- ride: U ovoj kolekciji smešteni su svi podaci vezani za vožnju. Sastoji se od četrnaest elemenata:
 - _id: Jedinstveni identifikacioni broj vožnje u bazi podataka. Generesan je automatski od strane MongoDB sistema za upravljanje bazama podataka.
 - type: String u kome se čuva informacija o vrsti vožnje. Može da ima dve moguće vrednosti, request (korisnik zahteva vožnju) ili offer (korisnik nudi vožnju)
 - fulfilled: Logička (boolean) vrednost. Označava da li je vožnja izvršena (true) ili ne (false);
 - from: String ne duži od 255 karaktera u kome se čuva naziv polazišta vožnje;
 - to: String ne duži od 255 karaktera u kome se čuva naziv dolazišta vožnje;
 - date: String od tačno 10 karaktera u kome se čuva datum vožnje;
 - time: String od tačno 5 karaktera u kome se čuva vreme početka vožnje;
 - passengers: niz _id vrednosti koji predstavljaju prijavljene putnike za vožnju;
 - notes: String ne duži od 255 karaktera u kome se čuva napomena vozača putnicima ili putnika vozačima;
 - seats_avail: Celobrojni broj u intervalu [0, 4] koji označava broj slobodnih mesta za vožnju ili broj traženih slobodnih mesta za vožnju;
 - price: Double broj u opsegu [0, 2³²] kojim se definiše cena vožnje ili tražena cena vožnje. Ukoliko je vožnja besplatna ili se traži besplatna vožnja, upsuje se 0.
 - fromgeo: uređen skup od dva double broja gde prvi predstavlja geografsku širinu a drugi geografsku dužinu polazišta. Koristi se kao informacija za ugrađenu Gugl kartu kako bi prikazala putanju putovanja.

- togeo: uređen skup od dva double broja gde prvi predstavlja geografsku širinu a drugi geografsku dužinu odredišta. Koristi se kao informacija za ugrađenu Gugi kartu kako bi prikazala putanju putovanja.
- userId: Podatak tipa _id gde je zapisan jedinstveni identifikacioni broj korisnika koji je napravio zahtev za vožnju ili ju je ponudio.

5 Implementacija softvera

Sledi primer programskog koda. Prikazan je deo koda komponente userController.js koja je zaduzena za opsluzivanje svih zahteva koji se ticu korisnika aplikacije:

```
////////////////////////////////////
const User = require('../models/user'); //napravi objekat User prema modelu user za bazu
podataka
const nodemailer = require('nodemailer'); //uvezi biblioteku nodemailer i napravi objekat
nodemailer
const async = require('async'); //uvezi biblioteku za rukovanje asinhronim događajima i napravi
objekat async
const crypto = require('crypto'); //uvezi biblioteku za hešovanje i napravi objekat crypto
//pravimo objekat transporter čija polja sadrže parametre za prijavu na SMTP server za slanje
emejllova
const transporter = nodemailer.createTransport({
  host: 'smtp.gmail.com', //adresa servera
  port: 465, //koristi se port 465 za šifrovanu vezu
  secure: true, //biramo šifrovanu vezu
  auth: {
    user: process.env.EMAIL_USERNAME, //iz .env datoteke učitavamo imejl korisničko ime
    pass: process.env.EMAIL_PASSWORD, //i pristupnu šifru
  },
});
//Pravimo objekat u kome čuvamo link za verifikaciju naloga novog korisnika
//kao i parametre za slanje verifikacionog emejla novom korisniku
const AccountVerificationEmail = (user) => {
  const baseurl = process.env.DEV_SERVER; //promenljiva u koju smeštamo prvi deo linka (putanje)
  // eslint-disable-next-line no-underscore-dangle
  const link = `${baseurl}/verify/${user._id}`; //promenljiva u koju smeštamo link za verifikaciju
  //i sastoji se od osnovne putanje i _id pramaetra korisnika koji je automatski generisao MongoDB
  //promenljiva u kojoj čuvamo parametre emejla kai njegov sadržaj koji se šalje korisniku radi
  verifikacije
  const mailOptions = {
    from: '"Путник" <singidunumrpi@gmail.com>',
    to: user.email,
    subject: 'Путник - регистрација новог налога',
    html: `Молимо вас да кликнете на следећи линк, или да га налепите у адресно поље вашег
прегледача како бисте завршили процес регистрације.
<br/> <a href=${link}>Верификуј мој налог</a> <br/> Потом се пријавите на систем са вашим
емејлом и шифром налога на Путник систему.
<br /><br /> Путник`,
  };
};
```

```

//funkcija u okviru objekta kojom šaljemo imejl sa njegovim sadržajem mailOptions i upisujemo
//eventualnu grešku u komandni prozor
transporter.sendMail(mailOptions, (err) => {
  if (err) {
    console.log(err);
  }
});
};
//Funkcija kojom stvaramo i autentifikujemo novog korisnika
exports.createAndAuthenticateUser = (request, response, next) => {
  //proveravamo da li je korisnik uneo dva puta istu šifru
  //ako nije vraćamo statusni kod 400 sa tekstom 'šifre se ne poklapaju'
  if (request.body.password && request.body.passwordConf) {
    if (request.body.password !== request.body.passwordConf) {
      const err = new Error('Šifre se ne poklapaju. ');
      err.status = 400;
      response.send({ status_text: 'šifre se ne poklapaju' });
      return next(err);
    }
  }
  //ako su dobijene vrednosti parametara novog korisnika različite od null
  if (request.body.email &&
    request.body.name &&
    request.body.password &&
    request.body.passwordConf) {
    //napravi objekat userData i smesti novopridošle podatke u njega
    const userData = {
      email: request.body.email,
      name: request.body.name,
      password: request.body.password,
    };
    //napravi nov JSON zapis prema modelu korisnika user u formatu za MongoDB BP
    User.create(userData, (error, user) => {
      if (error) {
        //ako dođe do greške vrati je sa sadržajem greške
        response.send({ status_text: error.message });
        return next(error);
      }
      //pozovi funkciju za proveru verifikovanosti naloga trenutnog korisnika
      AccountVerificationEmail(user);
      return response.send({ user, status_text: 'Korisnik nije verifikovan. Proverite imejl za
završetak verifikacije.' });
    });
  }
  //ako je korisnik uneo pogrešnu šifru ili imejl

```


6 Testiranje softvera

6.1 Jedinično testiranje

Prilikom jediničnog testiranja modula korišćeni su testovi pisani uz pomoć biblioteke Adapter. Sledi primer test skripte koja je korišćena pri jediničnom testiranju komponente Ride.js:

```
import React from 'react';
import Adapter from 'enzyme-adapter-react-16';
import { configure, shallow } from 'enzyme';
import createHistory from 'history/createMemoryHistory';
import RideTableContainer from './ridesContainer/RideTableContainer';
import Ride from './Ride';

configure({ adapter: new Adapter() });
const rides = [
  {
    id: '5ae522018ba7d20014d396e9',
    type: 'offer',
    studentID: '',
    fulfilled: false,
    from: 'Устаничка 13/1, Belgrade, Serbia',
    fromgeo: { lat: 44.7880049, lng: 20.4732467 },
    to: 'Михајла Пупина 14, Belgrade, Serbia',
    togeo: { lat: 44.7696141, lng: 20.549091 },
    time: '10:22',
    date: '2019-05-13',
    passengers: [],
    seats_avail: 4,
    notes: '',
    price: 0,
  },
  {
    id: '5ee66ca6fa6c040014ce36c8',
    type: 'request',
    studentID: '',
    fulfilled: false,
    from: 'Gundulićeva, Zemun, Belgrade, Serbia',
    fromgeo: { lat: 44.8425138, lng: 20.4069736 },
    to: 'Rade Končara 13, Belgrade, Serbia',
    togeo: { lat: 44.7899775, lng: 20.4782974 },
    time: '02:15',
```

```

    date: '2020-04-28',
    passengers: [],
    seats_avail: 2,
    notes: "",
    price: 0,
  },
  {
    id: '1aa44357pp6c040014ce36c8',
    type: 'request',
    studentID: "",
    fulfilled: false,
    from: 'Бука Караџића 4, Belgrade, Serbia',
    fromgeo: { lat: 44.8184133, lng: 20.4568197 },
    to: 'Кнеза Милоша 11, Belgrade, Serbia',
    togeo: { lat: 44.8093373, lng: 20.4647808 },
    time: '14:00',
    date: '2019-06-29',
    passengers: [],
    seats_avail: 2,
    notes: "",
    price: 0,
  },
];
describe('Ride', () => {
  test('Inicijalizuje stanje kartici za zahtevanje voznji', () => {
    const comp = shallow(<Ride
      logoutSuccess={jest.fn}
      history={createHistory()}
    />, { disableLifecycleMethods: true });
    expect(comp.state('tab')).toBe('request');
  });
  test('Kartica zahtevanih voznji prikazuje samo zahtevane voznje', () => {
    const comp = shallow(<Ride
      rides={rides}
      logoutSuccess={jest.fn}
      history={createHistory()}
    />, { disableLifecycleMethods: true });
    comp.setState({ rides, tab: 'request' });
    const rideTable = comp.find(RideTableContainer);
    expect(rideTable.exists()).toBe(true);
    expect(rideTable.at(0).prop('rides')).toHaveLength(1);
  });
  test('Kartica ponuda voznji pokazuje samo ponudjene voznje', () => {
    const comp = shallow(<Ride

```



```

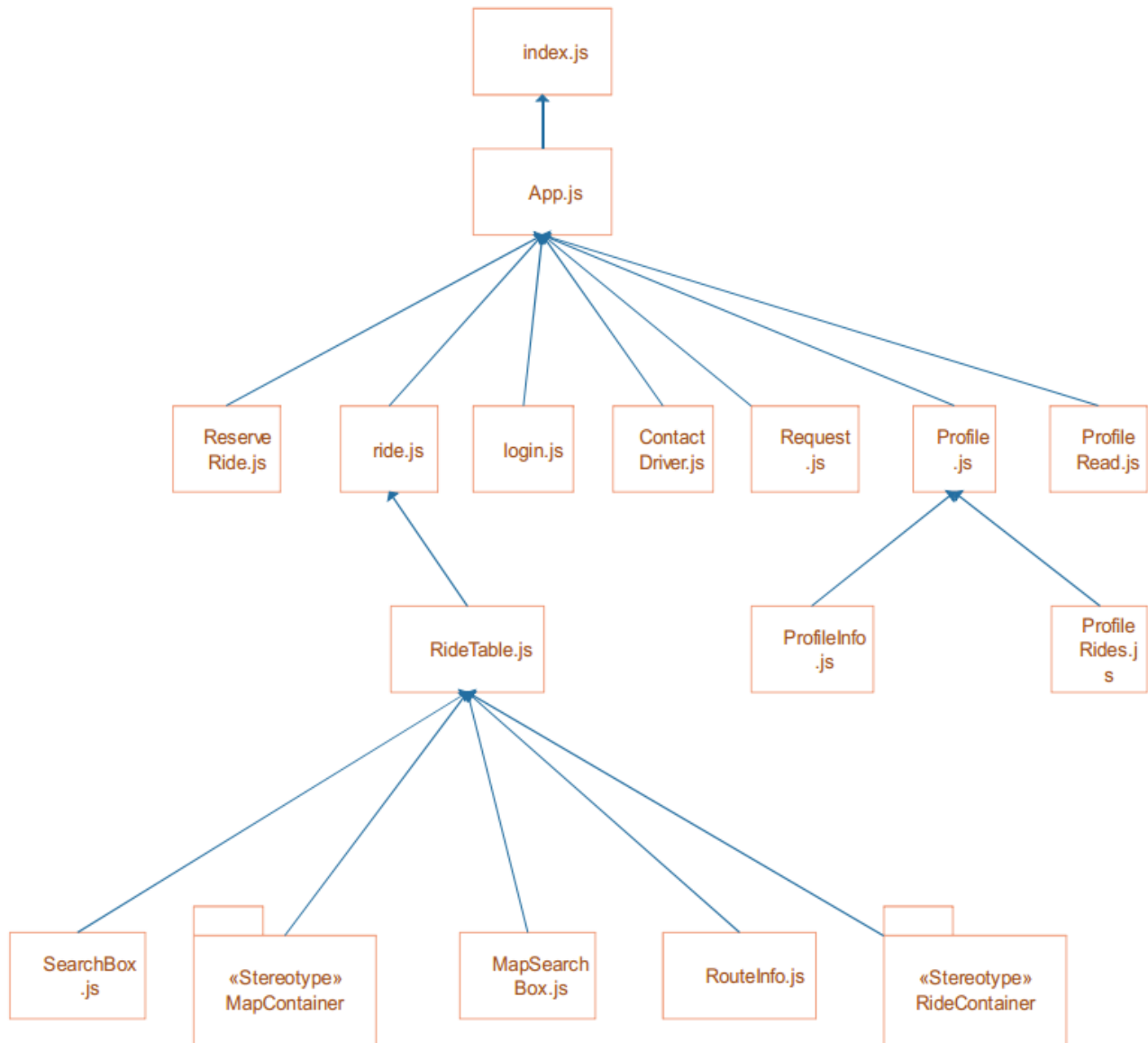
    rides={rides}
    logoutSuccess={jest.fn}
    history={createHistory()}
  />, { disableLifecycleMethods: true });
  comp.setState({ rides, tab: 'offer' });
  const rideTable = comp.find(RideTableContainer);
  expect(rideTable.exists()).toBe(true);
  expect(rideTable.at(0).prop('rides')).toHaveLength(2);
});
});

```

Listing 2: Primer skripte za testiranje kojom je testirana komponenta Ride.js

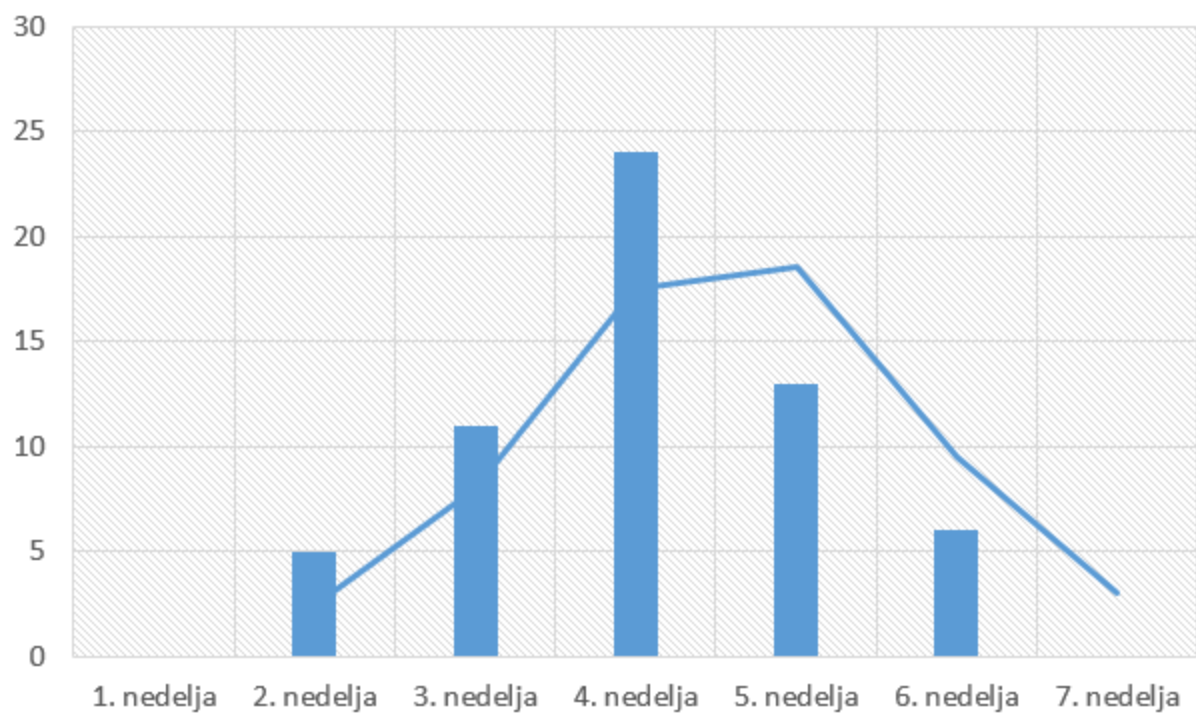
6.2 Integraciono testiranje

Za metod integracionog testiranja odabrano je integraciono testiranje od dna ka vrhu. Sledi slika hijerarhije koja pokazuje u kom redosledu su komponente testirane:



Slika 19: Prikaz hijerarhije komponenata prilikom integracionog testiranja

6.3 Grafik zavisnosti pronađenog broja grešaka u softveru po nedeljama



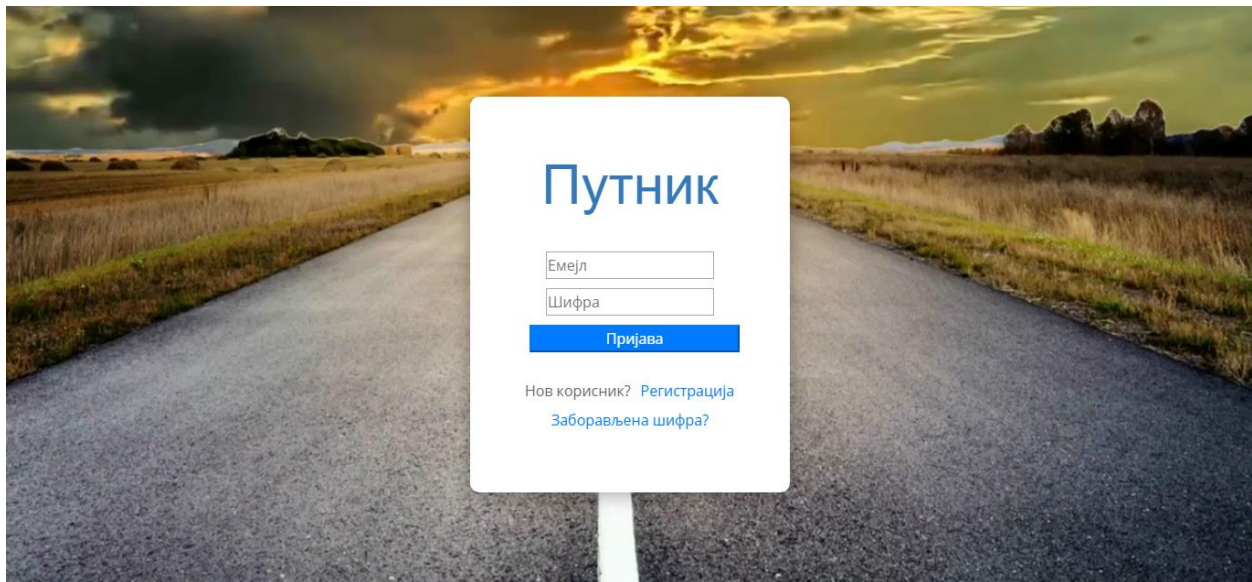
Slika 20: Grafik zavisnosti broja pronađenih grešaka po nedeljama

7 Isporuka softvera

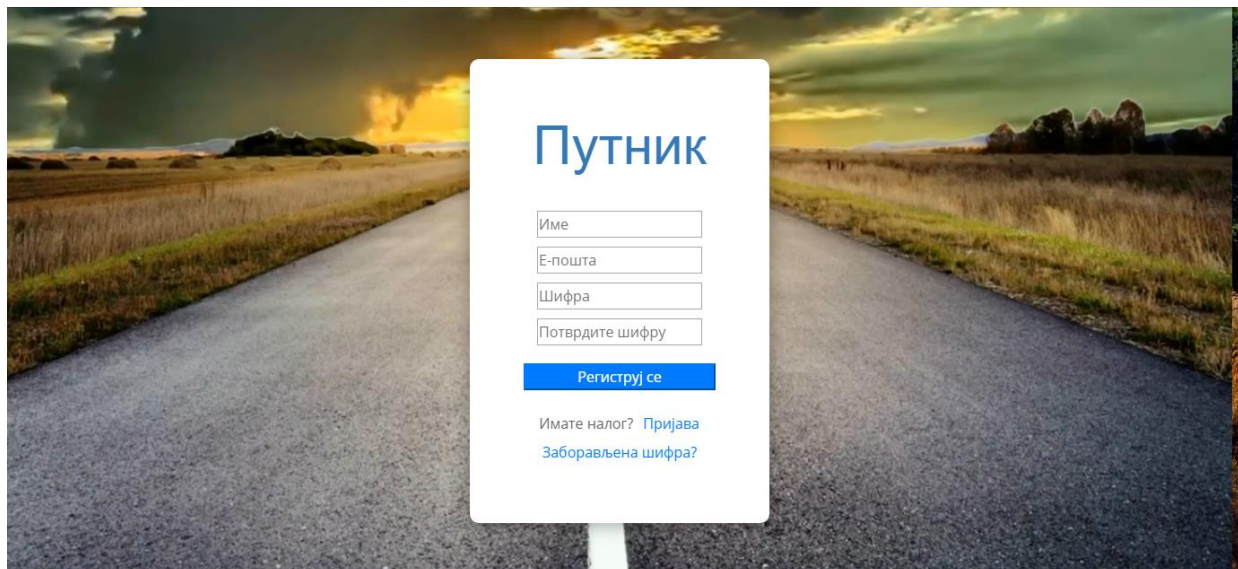
Sledi deo korisničkog uputstva za neke od najčešće korišćenih funkcionalnosti:

Registracija novog člana

Nakon učitavanja osnovnog, početnog prozora aplikacije u donjem delu ekrana postoji link na kojem piše [Регистрација](#). Potrebno je kliknuti na ovaj link i sačekati otvaranje novog prozora.



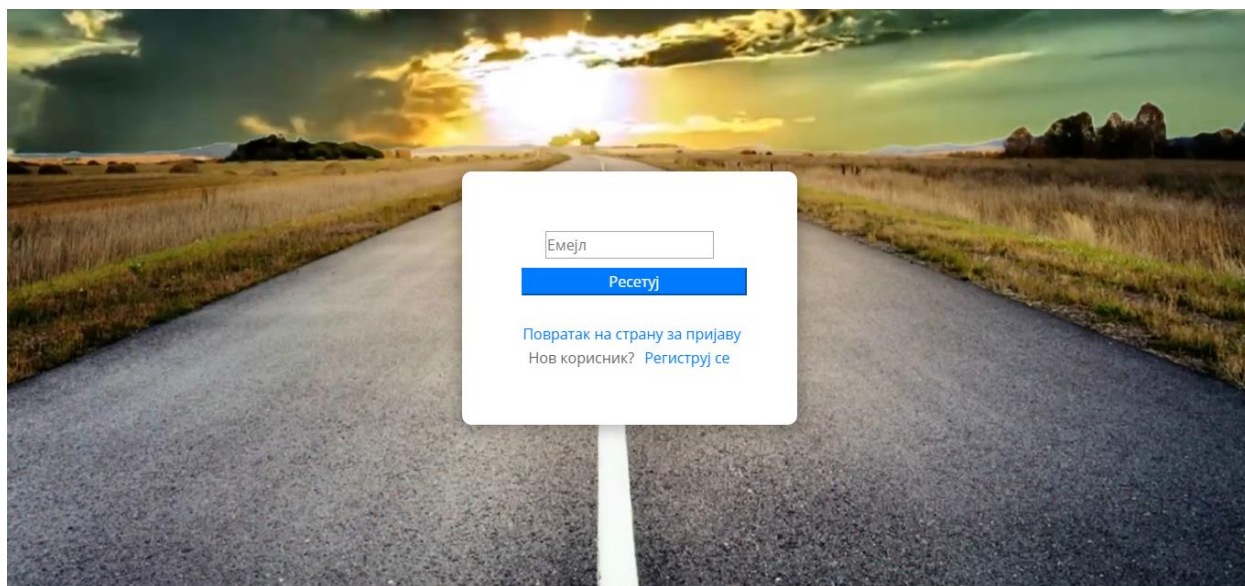
U novom prozoru se nalaze polja koja je potrebno popuniti: име, E-pošta, šifra i potvrda šifre. Ograničenja koja se odnose na ova polja figuriraju polju **Шифра** gde minimalna dužina lozinke sme da bude četiri karaktera. Pored toga u polje **Емејл** treba uneti ispravnu formu emejl adrese.



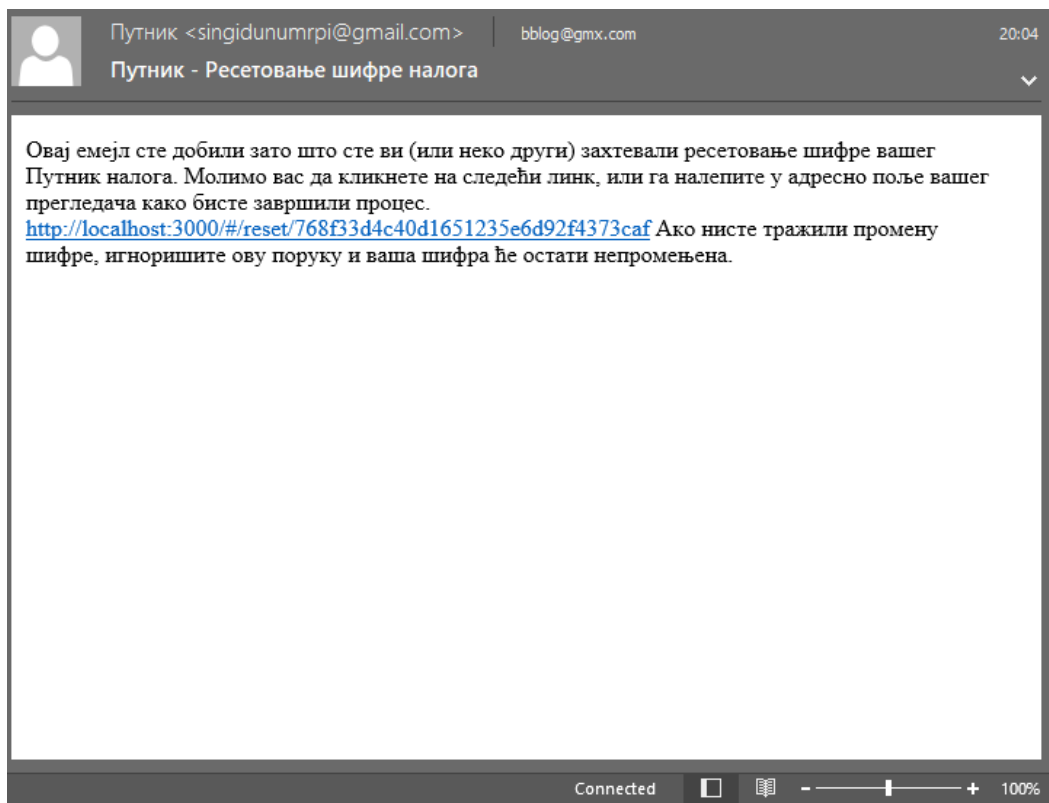
Nakon završenog pravilnog popunjavanja formulara treba kliknuti na polje **Региструј се**. Pojaviće se poruka na kojoj piše da je na imejl korisnika poslata poruka i da je potrebno kliknuti na link za verifikaciju. Nakon toga nalog je registrovan i korisnik može da počne sa korišćenjem aplikacije tako što se prijavljuje svojim mejlom i lozinkom koju je uneo prilikom registracije.

Zaboravljena šifra

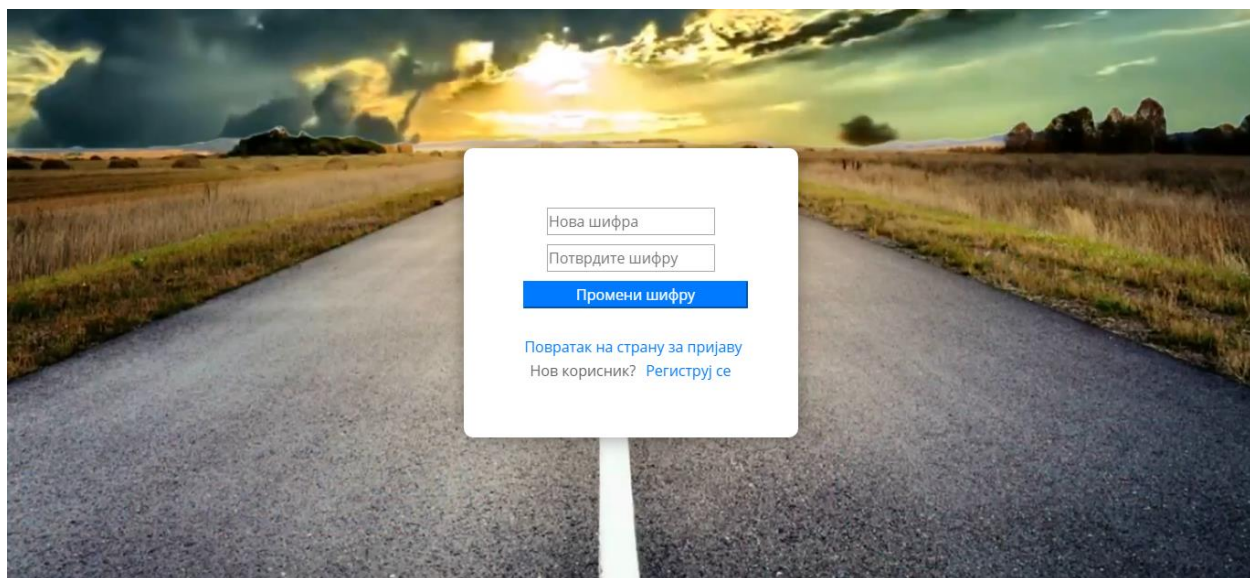
Ukoliko dođe do zaboravljanja šifre postoji način za resetovanje stare i unos nove šifre. Potrebno



je kliknuti na **Забрављена шифра?** i uneti imejl adresu koji je korišćen prilikom registracije. Nakon potvrde sa tasterom **Ресетуј**, sistem generiše poruku koju šalje na dati mejl u okviru koje se nalazi link na koji je potrebno kliknuti.



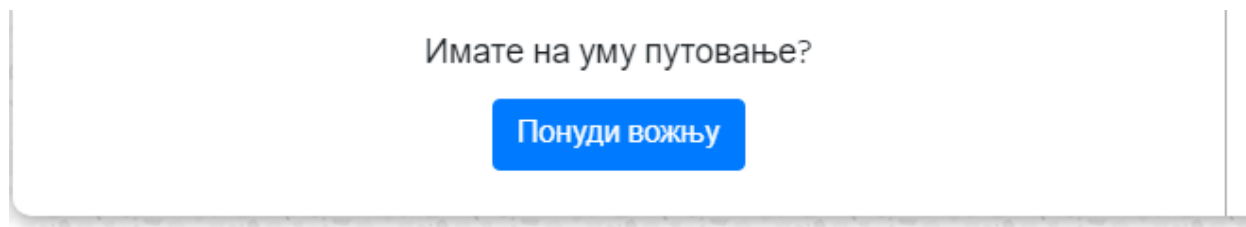
Nakon potvrde otvara se prozor u kojem se unosi nova šifra i potvrda nove šifre. Proces se završava izborom dugmeta **Промени шифру**.



U okviru korisničkog uputstva biće prikazana procedura za ponudu vožnje. Ovom procedurom registrovani korisnici ostvaruju mogućnost objavljivanja planirane vožnje. Nakon kompletiranja

svih faza, i unosa odgovarajućih podataka ponuda postave vidljiva svim korisnicima koji mogu da otpočnu sa rezervisanjem ponuđene rute.

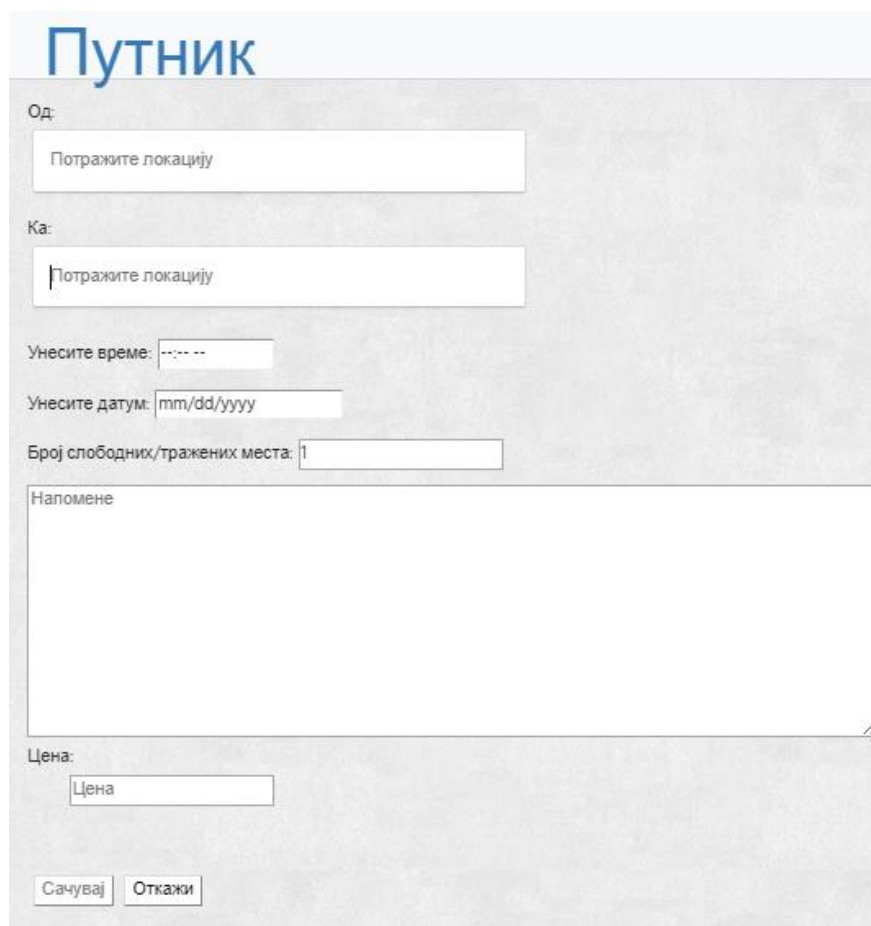
U okviru glavnog prozora za prevoznika u donjem levom uglu se nalazi dugme sa natpisom **Понуди вожњу**.



Имате на уму путовање?

Понуди вожњу

Odmah zatim se otvara prozor u kojem je potrebno uneti sve potrebne podatke kako bi vožnja koja se nudi postala validna i bila objavljena na uvid drugim korisnicima aplikacije.



Путник

Од: Потражите локацију

Ка: Потражите локацију

Унесите време: ---:--

Унесите датум: mm/dd/yyyy

Број слободних/тражених места: 1

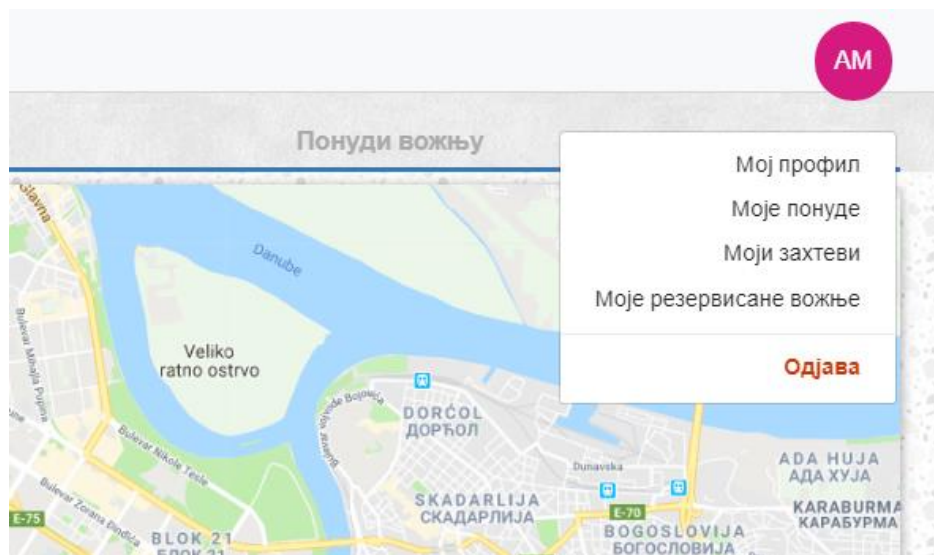
Напомене

Цена: Цена

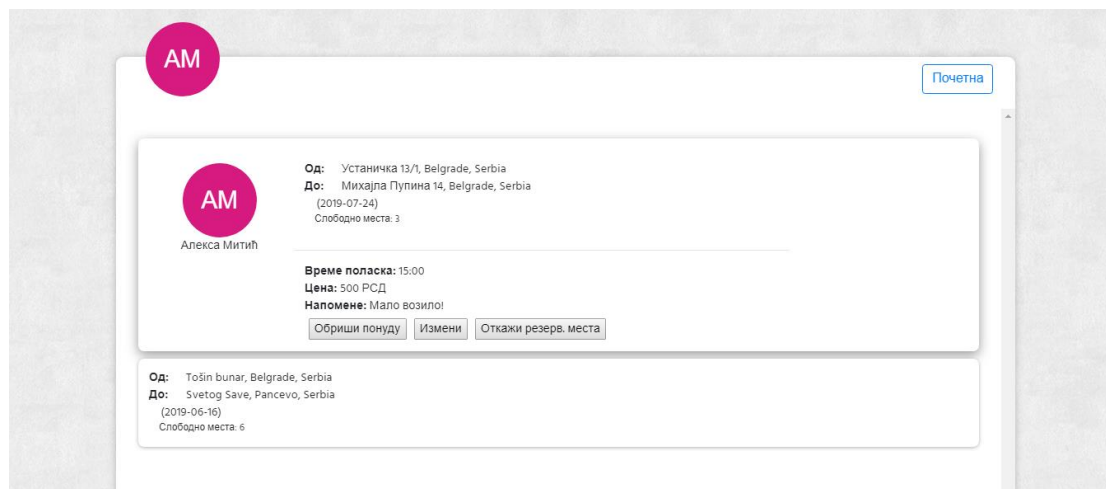
Сачувај Откажи

Unos podataka počinje sa prvim praznim poljem sa naslovom Od: gde se unosi početna lokacija tj. mesto i adresa sa koje vozač otpočinje vožnju. Sledeće polje treba popuniti sa podacima odredišta, lokacije gde se vožnja završava, koje opet čine mesto i adresa. U polje Unesite vreme

upisuje se tačno vreme kad se kreće sa polazišne lokacije. Potrebno je uneti datum planirane vožnje, gde postoji ograničenje koje sprečava korisnika da unese pogrešne podatke, sistem sprečava da se upiše datum koji je iz prošlosti. U skladu sa veličinom vozila koje koristi za planirano putovanje korisnik unosi broj slobodnih mesta koje mogu da rezervišu zainteresovani članovi. U okviru napomene mogu se navesti specifične ili karakteristične stvari koje opisuju ponuđenu vožnju. To mogu biti i neke korisne informacije koje mogu dodatno da zainteresuju korisnike kako bi se odlučili za rezervisanje vožnje. U polje cena se upisuje tražena suma za rezervaciju jednog mesta u vozilu. Klikom na dugme **Сачувај** završava se proces objavljivanja vožnje koja u tom trenutku postaje validna.



Proveru podataka objavljenje vožnje korisnik može da izvrši preko korisničkog naloga klikom na link **Моје понуде**. Gde se mogu pročitati svi elementi koji su uneti prilikom objavljivanja vožnje.



U svakom trenutku korisnik može da izmeni saržaj formulara i promeni neki od podataka izborom taster **Измени**. Takođe postoji mogućnost otkazivanja kompletne vožnje i brisanja svih objavljenih podataka klikom na taster **Обриши понуду**.

Literatura

- Tomašević, V., *Razvoj aplikativnog softvera*, Univerzitet Singidunum, Beograd, 2019.
- <https://developer.mozilla.org>, MDN veb dokumentacija, pristupljeno 09.05.2019.
- <https://docs.mongodb.com>, MongoDB dokumentacija, pristupljeno 10.05.2019.
- <https://mongoosejs.com/docs/api>, Mongoose dokumentacija, pristupljeno 10.05.2019.
- <https://reactjs.org/docs/getting-started.html>, React.js dokumentacija, pristupljeno 10.05.2019.
- <https://nodejs.org/docs>, Node.js dokumentacija, pristupljeno 10.05.2019.