Razvoj RESTful web aplikacije za dijeljenje automobila

Majetić, Sara

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Preddiplomski sveučilišni studij računarstva

Završni rad

Razvoj RESTful web aplikacije za dijeljenje automobila

Rijeka, rujan 2022.

Sara Majetić
0069087990

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Preddiplomski sveučilišni studij računarstva

Završni rad

Razvoj RESTful web aplikacije za dijeljenje automobila

Mentor: doc.dr.sc. Marko Gulić

Rijeka, rujan 2022.

Sara Majetić
0069087990

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 14. srpnja 2022.

Zavod: **Zavod za računarstvo**
Predmet: **Razvoj web aplikacija**
Grana: **2.09.06 programsko inženjerstvo**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Sara Majetić (0069087990)**
Studij: Preddiplomski sveučilišni studij računarstva

Zadatak: **Razvoj RESTful web aplikacije za dijeljenje automobila / The development of RESTful web application for car sharing**

Opis zadatka:

Razviti RESTful web aplikaciju za dijeljenje automobila. Aplikacija mora podržavati izradu korisničkog računa, uslugu davanja vlastitog vozila u najam kao i najam tuđih vozila. Također, potrebno je implementirati funkcionalnost praćenja statusa održavanja vlastitih vozila (tehnički pregled, registracija i sl.). Dodatno, aplikacija mora omogućiti pristup raznim korisnim informacijama o detaljima putovanja (vremenska prognoza, cijene goriva i sl.). Za razvoj poslužiteljskog dijela web aplikacije treba koristiti Laravel radni okvir uz proizvoljno odabran sustav za upravljanje bazama podataka. Također, treba koristiti poseban paket za realizaciju autentifikacije (JWT token) koji je podržan od strane Laravela. Za razvoj klijentskog dijela aplikacije treba koristiti React JavaScript knjižicu za razvoj korisničkog sučelja uz učinkovito renderiranje aplikacije na uređajima s različitim veličinama zaslona.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Smajetić

Zadatak uručen pristupniku: 18. srpnja 2022.

Mentor:

Doc. dr. sc. Marko Gulić

Predsjednik povjerenstva za završni ispit:

Prof. dr. sc. Kristijan Lenac

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradila ovaj rad.

Rijeka, rujan 2022.

_____

Sara Majetić

Zahvala

Zahvaljujem mentoru doc. dr. sc. Marku Guliću na ukazanom povjerenju i korisnim savjetima tijekom pisanja ovoga rada. Zahvaljujem obitelji i prijateljima na podršci tijekom studiranja.

Sadržaj

Popis slika

Poglavlje 1

Uvod

1.1 O aplikaciji

Autoshare je web aplikacija namjenjena davanju osobnih vozila u najam ili iznajmljivanju tuđih vozila. Korisnik ima mogućnost izrade vlastitog korisničkog računa, nakon čega može unijeti potrebne informacije o svojim vozilima ili pretraživati vozila dostupna za najam s obzirom na lokaciju i razdoblje. Aplikacija prati i provjerava informacije o stanju održavanja svih vozila poput vođenja evidencije o odlascima na tehnički pregled ili datuma isteka registracije, kako bi u najam bila dozvoljena samo ispravna vozila. Korisnik pri završetku putovanja može ostaviti javnu recenziju koja služi kao pomoć ostalim korisnicima pri odabiru najma.

Osim same mogućnosti najma, aplikacija pruža korisne informacije o detaljima putovanja kao što su vremenska prognoza i informacije o aktualnim cijenama goriva.

1.2 Korištene tehnologije

Aplikacija je razvijena koristeći REST (eng. Representational state transfer) arhitekturu koja podrazumijeva skup ograničenja za izmjenu podataka između softverskih sustava klijent-poslužitelj.

Poslužiteljski dio aplikacije razvijen je u PHP programskom jeziku [1] uz Laravel radni okvir verzije 8 [2]. Laravel je besplatni radni okvir otvorenog koda koji je namijenjen za razvoj web aplikacija prema model-pogled-upravitelj (eng. model–view–controller, skraćeno MVC) obrascu arhitekture.

Odabrani sustav za upravljanje bazama podataka je PostgreSQL [3], koji se često naziva i Postgres. Koristi relacijski model baze podataka i podržava SQL (eng. *Structured Query Language*) standardni jezik upita, a koristi se kao primarna pohrana podataka ili skladište podataka za mnoge vrste aplikacija [4].

Klijentski dio aplikacije razvijan je Javascript programskim jezikom uz React.js knjižnicu [5]. Korisničko sučelje aplikacije izgrađeno je pomoću React Bootstrap razvojne knjižnice [6] koja pruža već gotove Bootstrap komponente za React poput navigacije, formi, kartica, polja za unos i sl.

Za realizaciju autentifikacije korišten je JWT (*JSON Web Token*) standard. JWT definira način za siguran i samostalan prijenos podataka u obliku JSON objekta (eng. *JavaScript Object Notation*), a najčešće se koristi za provjeru identiteta pri slanju zahtjeva između klijenta i servera.

Za pisanje koda korišteni su JetBrains PhpStorm [7] i WebStorm [8] integrirana razvojna okruženja. Oba alata omogućuju sprječavanje pogrešaka u hodu, dovršavanje i refaktoriranje koda, testiranje i mnoge druge funkcionalnosti koje olakšavaju pisanje i uređivanje programskog koda.

Ostale tehnologije korištene u izgradnji su *Googleov* API (eng. Application programming interface) *Google Places* [9] za dohvaćanje lokacija, *Open Weather* API [10] za dohvaćanje vremenske prognoze po lokaciji, te vPIC (eng. *Product Information Catalog and Vehicle Listing*) API [11] za dohvaćanje proizvođača i modela vozila.

Poglavlje 2

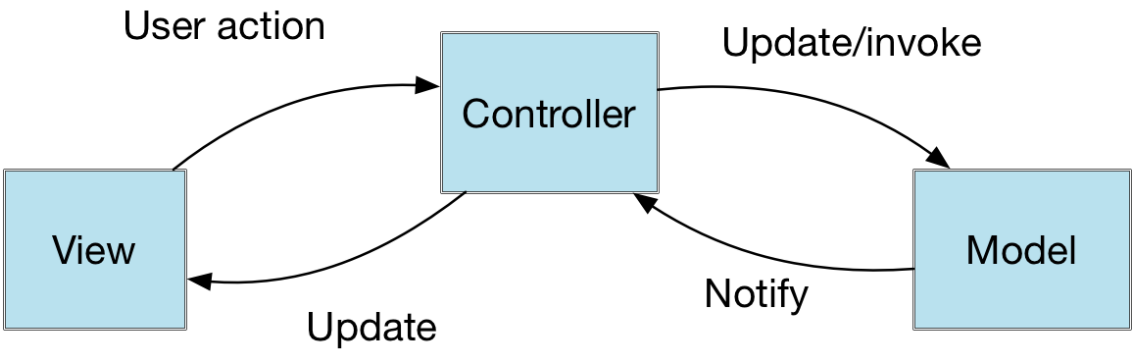
Korištene tehnologije

U ovom odjeljku biti će detaljnije opisana uporaba i značajke najvažnijih tehnologija koje su korištene za razvoj Autoshare aplikacije.

2.1 Laravel

Laravel je besplatan PHP radni okvir čiju je prvu verziju objavio u 2011. godini Taylor Otwell. Namijenjen je za razvoj web aplikacija koristeći MVC (eng. Model-view-contoller) model arhitekture [12].

Postoje različite verzije MVC modela, no sve odvajaju dijelove koda u tri komponente - model, pogled i kontroler ili upravitelj. Pogled služi za komunikaciju s korisnikom i sadrži sve vezano za korisničko sučelje. Model sadrži podatke i funkcije, te komunicira s bazom. Upravitelj upravlja korisničkim zahtjevima, ažurira pogled, pokreće neku akciju u modelu i od njega prima povratnu informaciju [13]. MVC model arhitekture rezultira "čišćim" kodom, jednostavnijim za razvoj, testiranje i održavanje. Primjer MVC arhitekture i pravila za komunikaciju između komponenti prikazan je na slici 2.1.



Slika 2.1 MVC komponente i načini interakcije [13]

Sučelje komandne linije (eng. *comand-line interface*, skraćeno CLI) koje koristi Laravel naziva se Artisan. Ono pri izgradnji aplikacije nudi niz korisnih naredbi, ali i stvaranje prilagođenih naredbi. Artisan je odlična metoda za stvaranje i upravljanje osnovnim MVC datotekama s njihovim pridruženim postavkama. U ovom projektu pretežno je korišten za upravljanje migracijama baze podataka i pokretanje poslužitelja.

Laravel uključuje alat za mapiranje odnosa objekata odnosno ORM (eng. *object relational mapper*) naziva Eloquent. Eloquent pojednostavljuje interakciju s bazom podataka time što, kao i svaki drugi ORM, omogućuje interakciju s bazom podataka na objektno-orijentiran način, čime uklanja potrebu za izravnim korištenjem SQL-a. Eloquent služi za stvaranje "model" komponente MVC strukture [14]. Svaka tablica ima odgovarajući model koji se koristi za interakciju s njom. Uz dohvaćanje zapisa, Eloquent također omogućuje umetanje, ažuriranje i brisanje podataka iz tablica [15]. U ovom projektu Eloquent je korišten primarno za stvaranje modela i pripadajućih migracija, definiranje relacija između tablica, dohvaćanje modela i izgradnja upita.

Korištenje Laravela za razvoj ovog projekta pojednostavila je opsežna dokumentacija. Značajke koje Laravel čine dobrim izborom za razvoj web aplikacija su modularnost, mogućnost testiranja, autentifikacija, usmjeravanje (eng. *routing*), upravljanje konfiguracijom, Eloquent ORM itd.

2.2 PostgreSQL

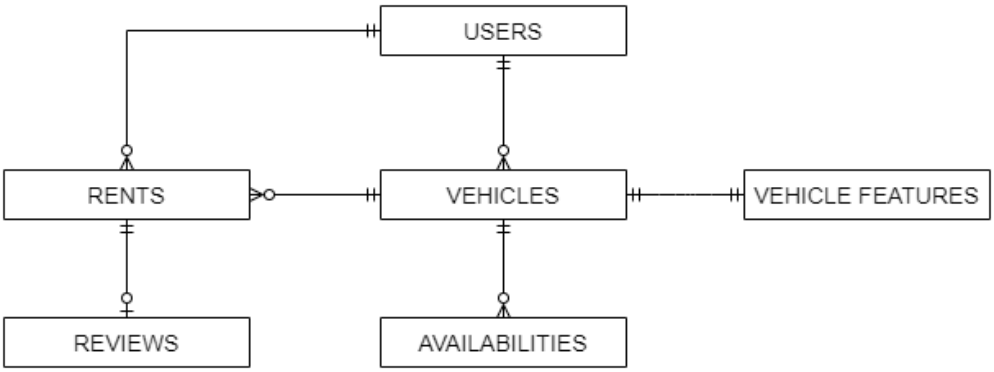
Baze podataka najčešći su način pohranjivanja podataka koji koriste *web* aplikacije, a sustavi za upravljanje bazama podataka su alati koji omogućuju jednostavno upravljanje strukturom i podacima u bazi.

PostgreSQL je široko korišten sustav za upravljanje bazama podataka otvorenog koda, što znači da se može slobodno koristiti, modificirati i distribuirati u bilo koju svrhu, bilo da je privatna, komercijalna ili akademska. Razvijen je na Sveučilištu California u Berkleyu s početkom 1986. godine, a često je referiran i pod nazivom Postgres [16]. Koristi se uz neke od najpopularnijih programskih jezika, uključujući C, C++, Python, Javu i PHP [4].

Implementiran je na temelju relacijskog modela baze podataka te podržava SQL (eng. *Structured Query Language*) standardni jezik upita. Model baze podataka definira logičku strukturu baze - određuje kako se podaci mogu pohraniti, organizirati i manipulirati, a najčešće korišteni model je upravo relacijski [17].

Relacijski model razvrstava podatke u tablice, također poznate kao relacije, s fiksnim brojem atributa i tipova podataka gdje jedan zapis ogovara jednom retku u tablici. Model također uzima u obzir vrste odnosa između tih tablica, uključujući odnose jedan-prema-jedan, jedan-prema-više i više-prema-više. Do strukture tablica u bazi dolazi se procesom normalizacije, kojim se između ostalog osigurava da u bazi nema redundantnih podataka [18]. Uobičajeno se podrazumijeva da su sve relacijske baze normalizirane, međutim u praksi se ponekad zbog povećanja performansi ne drži strogo svih pravila normalizacije.

Na slici 2.2 prikazan je ER (eng. *Entity-relationship*) dijagram *Autoshare* baze podataka iz kojega se vide relacije između entiteta.



Slika 2.2 ER dijagram baze podataka za Autoshare aplikaciju

PostgreSQL je odabran za ovaj projekt budući da je besplatan, pouzdan i jednostavan za koristiti uz Laravel radni okvir.

2.3 React.js

React.js ili React je JavaScript knjižnica otvorenog koda koja je stvorena od strane Facebooka 2013. godine, a koristi se za izgradnju korisničkih sučelja [5]. Danas je jedan od najpopularnijih okvira za razvoj brzih i efikasnih web aplikacija.

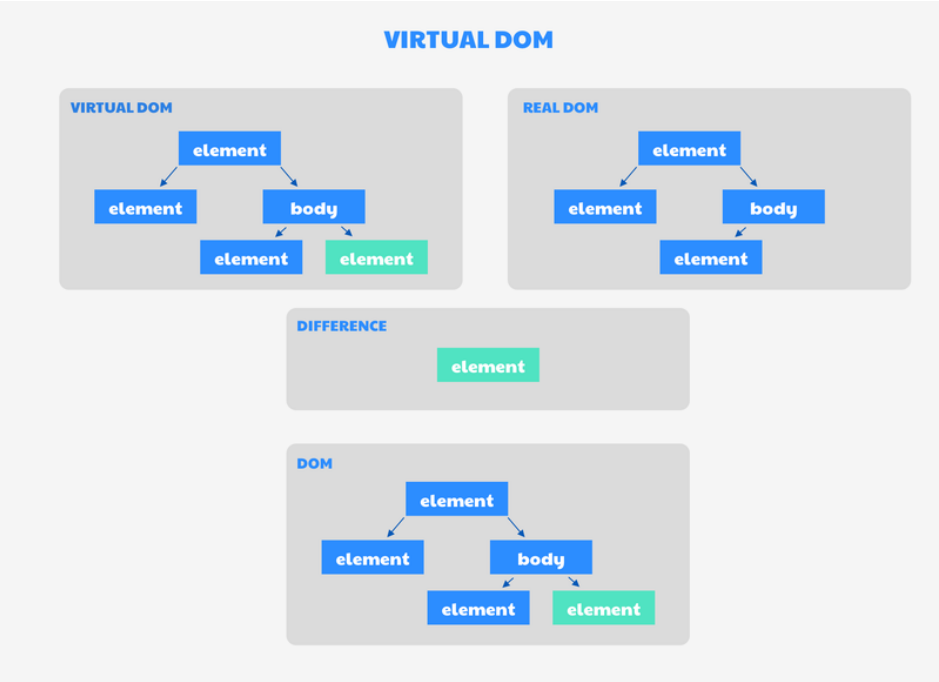
React se u ovom projektu koristi za izradu *view* komponente MVC arhitekture. Najveći doprinos Reacta pri izradi ovog projekta bila je pomoć u upravljanju stanjima koja omogućuju brzo i učinkovito ažuriranje i renderiranje komponenti, što rezultira dinamičnom i interaktivnom aplikacijom.

Pozitivna strana Reacta je to što koristi deklarativni stil programiranja, a ne imperativni. Prvi stil specificira prevoditelju (eng. *complier*) što treba učiniti, dok drugi također mora specificirati kako to učiniti, stoga programiranje s Reactom rezultira s manje koda [19].

2.3.1 React DOM

Pri učitavanju web stranice, preglednik stvara objektni model dokumenta (eng. *Document Object Model*, skraćeno DOM) koji predstavlja stranicu u obliku stabla. Omogućuje programu komunikaciju sa stranicom, promjenu strukture i sadržaja definirajući objektno orijentirani API za HTML i XML dokumente. Javascriptov poziv DOM API-ju je primjer imperativnog programiranja koji je ranije spomenuto. No za razliku od JavaScripta, React koristi koncept virtualnog DOM-a. Ideja virtualnog DOM-a je skupe DOM manipulacije svesti na minimum.

React *render()* metoda stvara DOM stablo objekata po React komponentama. Kada se promijeni stanje objekta u React aplikaciji, virtualni DOM se ažurira. Zatim se uspoređuju stvarni i virtualni model te ažuriraju samo promijenjeni objekti, umjesto ažuriranja svih objekata u stvarnom DOM-u. Ovaj proces prikazan je na slici 2.3. Ova metoda poboljšava performanse, točnije omogućuje brži prikaz promjena, posebno u usporedbi s drugim *front-end* tehnologijama koje moraju ažurirati sve objekte čak i ako se promijeni samo jedan [20].



Slika 2.3 Grafički prikaz funkcioniranja virtualnog DOM-a [21]

2.3.2 JSX

JavaScript XML (JSX) je sintaksa slična XML-u/HTML-u koja omogućuje pisanje i dodavanje HTML-a u React. Omogućuje pisanje HTML elemenata u JavaScriptu i njihovo postavljanje u DOM bez potrebe za pozivanjem React.createElement() metode. Da bi se JSX kod pretvorio u JavaScript kod razumljiv pregledniku, koristi se JavaScript prevoditelj kao što je Babel [22].

Class je često korišteni atribut u HTML-u, međutim riječ *class* rezervirana riječ u JavaScriptu, pa je u JSX-u umjesto toga korišten atribut *className* za dodavanje CSS klasa komponentama.

Izrazi se pišu unutar vitičastih zagrada , oni mogu biti React varijabla, svojstvo ili bilo koji drugi važeći JavaScript izraz. Nakon kompilacije, JSX izrazi postaju obični pozivi JavaScript funkcija i jednaki su JavaScript objektima. To znači da se JSX može koristiti unutar *if* naredbi i *for* petlji, dodijeliti varijablama, prihvatiti kao argument i vratiti iz funkcija.

React ne zahtjeva korištenje JSX-a, no koristan je kao vizualna pomoć pri radu s korisničkim sučeljem unutar JavaScript koda. Također omogućuje Reactu da prikaže korisna upozorenja i poruke o greškama.

2.3.3 Hooks

React kuke (eng *hooks*) noviji su dodatak Reactu (od verzije 16.8 nadalje). To su funkcije koje omogućuju korištenje stanja i drugih značajki Reacta bez korištenja klasa.

React nudi nekoliko ugrađenih *hook*-ova kao što su *useState*, *useContext*, *useReducer*, *useMemo* i *useEffect*. UseState i useEffect, koji se najčešće koriste, služe za kontrolu stanja i nuspojava promjene stanja.

Pravila za korištenje *hook*-ova su da se oni trebaju pozivati samo na najvišoj razini, ne unutar petlji, *if* naredbi ili ugniježdenih funkcija, te se trebaju pozivati samo iz React funkcijskih komponenti, ili iz drugih *hook*-ova [23].

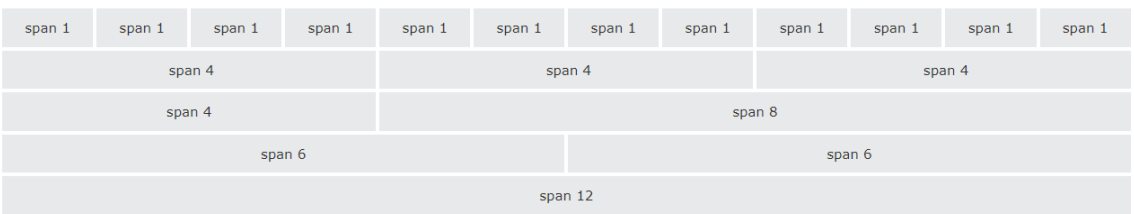
2.3.4 React komponente

Najveći dio programiranja u Reactu je stvaranje komponenata. Komponente raščlanjuju aplikaciju na manje zasebne dijelove koji se mogu koristiti i pozivati na više različitih mjesta. *Props* objekt se koristi za prijenos podataka i metoda iz roditeljske komponente u podređenu komponentu, odnosno komponentu dijete. Taj objekt je nepromjenjiv.

2.4 React Bootstrap

Uz React.js, korištena je i React-Bootstrap knjižnica. React-Bootstrap nadograđuje obični Bootstrap kao knjižnica za razvoj korisničkih sučelja. Duge CSS definicije i Bootstrap komponente sažete su u React-stilizirane komponente.

React Bootstrap poznat je po svom responzivnom dizajnu. Koristi sistem rešetke (eng. *grid system*), odnosno skup spremnika, stupaca i redaka za određivanje rasporeda i poravnavanja sadržaja stranice. Dijeli stranicu u 12 jednakih dijelova (vertikalno), kao što je vidljivo na slici 2.4. Širine elemenata mogu se definirati tako da ovise o veličini zaslona. Na primjer, na velikom zaslonu računala sadržaj bi mogao bolje izgledati organiziran u tri stupca, dok bi na manjem mobilnom zaslonu bilo bolje da su stavke sadržaja naslagane jedna na drugu [24].



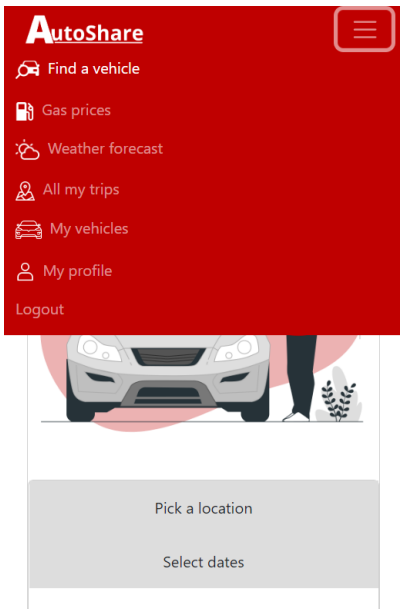
Slika 2.4 Prikaz sistema rešetke u Bootstrap-u [24]

Zadane prijelomne točke u React Bootstrapu prikazane su na slici 2.5, definirane u pikselima. One pružaju dobru osnovu za pravilno renderiranje aplikacije na ekranima različitih uređaja, također se mogu i prilagoditi.

Breakpoint	Class infix	Dimensions
X-Small	*None*	<576px
Small	`sm`	≥576px
Medium	`md`	≥768px
Large	`lg`	≥992px
Extra large	`xl`	≥1200px
Extra extra large	`xxl`	≥1400px

Slika 2.5 Šest zadanih prijelomnih točaka [25]

Responzivna navigacijska traka i kartica za pretraživanje vozila na slici 2.6 prikazuju prilagođavanje komponenti ekranu veličine mobilnog uređaja.



Slika 2.6 Navigacijska traka i kratica prilagođene ekranu mobilnog uređaja

React Bootstrap knjižnica odabrana je za izradu korisničkog sučelja ovog rada jer nudi već gotove komponente. Time pojednostavljuje i ubrzava izradu korisničkih sučelja modernog dizajna, što oslobađa više vremena za razvoj drugih funkcionalnosti.

2.5 React router

React Router je klijentska i poslužiteljska knjižnica za usmjeravanje za React. React Router radi svugdje gdje se koristi React, ali i izvan njega npr.u Node.js-u na poslužitelju [26].

Tri primarne kategorije komponenti u React Routeru su: usmjerivači, poput *<BrowserRouter>*, podudaranje ruta, kao što su *<Route>* i *<Switch>* i navigacija, poput *<Link>*, *<NavLink>* i *<Redirect>*.

React Router također dolazi uz nekoliko korisnih *hook*-ova za pristup stanju *routera* i navigaciju iz samih komponenti. U ovom projektu upotrijebljeni su *useNavigate*, *useLocation* i *useParams*.

Komponenta *<Link>* korištena je za navigaciju između stranica. Komponenta *<NavLink>* posebna je vrsta *<Link>* komponente, koja se može stilizirati kao "aktivna" kada njezino svojstvo (eng. *prop*) odgovara trenutnoj lokaciji, što je čini savršenom za izradu navigacijske trake. Na slici 2.7 prikazano je kako je komponenta *<Route>* korištena za deklaraciju ruta u aplikaciji i pridruživanje pripadajućih komponenti.

```
<Routes>
  <Route exact path='/' element={<Navigate to='/vehicles' />} />
  <Route path='/login' element={<Login setUser={setUser} />} />
  <Route path='/register' element={<Registration setUser={setUser} />} />
  <Route path='/user' element={<User setUser={setUser} />} />
  <Route path='/vehicles' element={<Vehicles />} />
  <Route path='/add' element={<AddVehicle />} />
  <Route path='/vehicle/:vehicleId' element={<EditVehicle />} />
  <Route path='/addAvailability/:vehicleId' element={<AddAvailability />} />
  <Route path='/rent' element={<Rent />} />
  <Route path='/details/:availId' element={<RentDetails />} />
  <Route path='/list' element={<ListOfAvail />} />
  <Route path='/trips' element={<History />} />
  <Route path='/review/:rentId' element={<AddReview />} />
  <Route path='/forecast' element={<Weather />} />
  <Route path='/gas' element={<GasPrices />} />
</Routes>
```

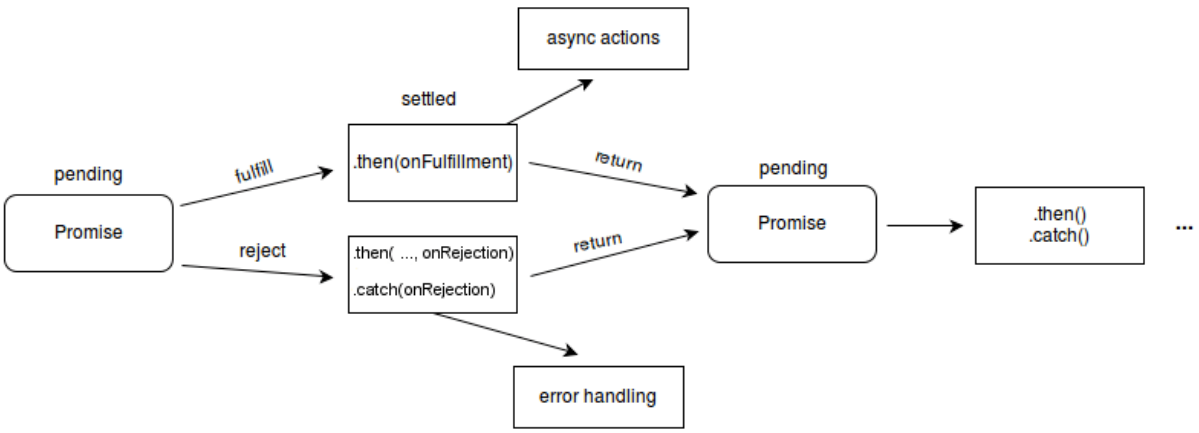
Slika 2.7 Prikaz svih ruta aplikacije

2.6 Axios

Axios je još jedna knjižnica koja se može priključiti Reactu. To je popularna knjižnica koja se uglavnom koristi za slanje asinkronih HTTP zahtjeva REST krajnjim točkama [27]. Asinkrone funkcije također se koriste za pisanje i čitanje iz datoteke ili baze podataka. Asinkrona funkcija znači da se temelji na obećanjima (eng.*promise*) i omogućuje čekanje da druga funkcija završi s izvođenjem.

Obećanja asinkronim funkcijama omogućuju vraćanje vrijednosti poput sinkronih metoda - umjesto da odmah vrate konačnu vrijednost, asinkrona funkcija daje obećanje da će vratiti vrijednost u nekom trenutku u budućnosti. Stanja i metode obećanja kod asinkronih funkcija prikazana su na slici 2.8.

Obećanje može biti u jednom od tri stanja: na čekanju (eng. *pending*), ispunjeno (eng. *fulfilled*) i odbijeno (eng. *rejected*). Obećanje je ispunjeno (eng. *settled*) ako je ili ispunjeno ili odbijeno. Nakon što obećanje postaje ispunjeno, metode Promise.prototype.then() i Promise.prototype.catch() koriste se za definiranje daljnjih akcija [28]. Često postoji potreba za izvršavanje dvije ili više asinkronih funkcija jedna za drugom, gdje svaka sljedeća operacija počinje kada prethodna operacija uspije, s rezultatom iz prethodnog koraka. To postižemo stvaranjem lanca obećanja, što je moguće jer funkcije *then()* i *then()* i same vraćaju obećanje [29].



Slika 2.8 Stanja i metode "obećanja" kod asinkronih funkcija [28]

Ključna riječ *async* prije funkcije znači da funkcija vraća obećanje, a ključna riječ

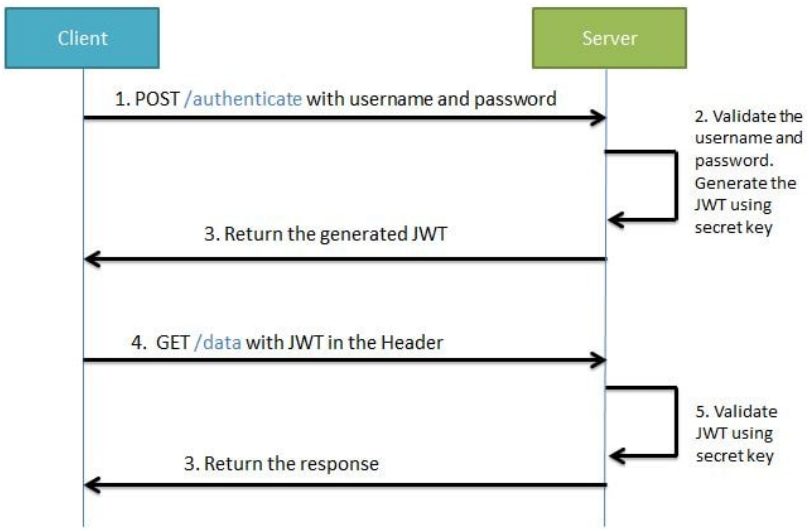
await prije funkcije tjera funkciju da čeka to obećanje i može se koristiti samo unutar asinkronih funkcija.

U ovom projektu Axios je korišten za dohvaćanje podataka iz javnih API-ja - za popis proizvođača i modela vozila, te dohvaćanje podataka koji se prikazuju na *Gas prices* i *Weather forecast* stranicama u aplikaciji. Također je upotrebljen za izradu posebne instance za autentifikaciju JWT-om, kojom se JWT token šalje u zaglavlju svakog zahtjeva prema poslužitelju.

2.7 JSON Web Token

JSON Web Token (JWT) otvoreni je standard za stvaranje pristupnih tokena temeljen na JSON-u. Oslanja se na druge standarde temeljene na JSON-u: JSON *Web* potpis i JSON *Web* enkripcija. Tokeni su dizajnirani da budu kompaktni, sigurni za URL i uobičajeno se koriste za prosljeđivanje identiteta autentificiranih korisnika između pružatelja identiteta i pružatelja usluga [30].

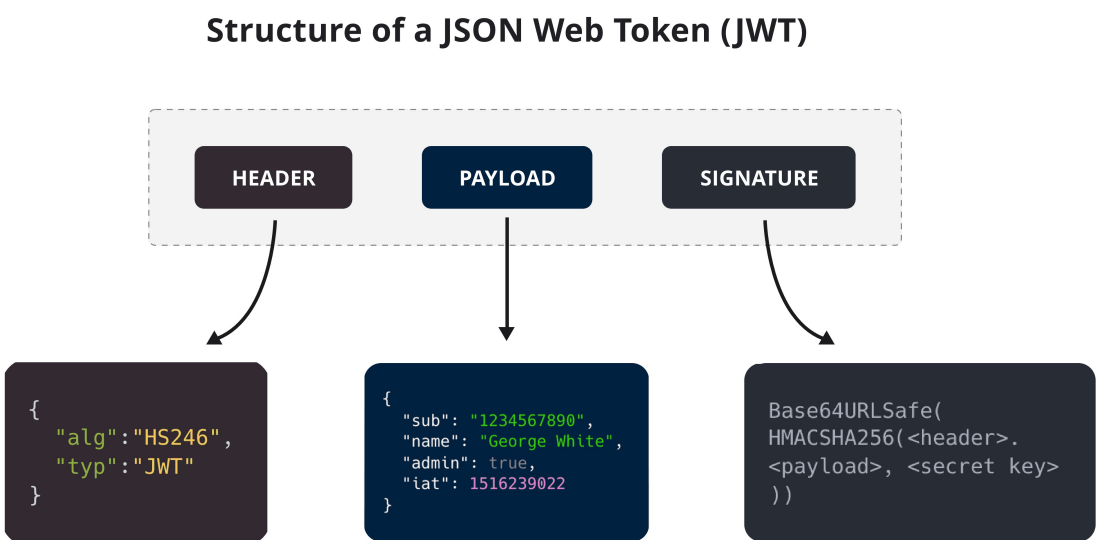
U ovom projektu JWT je korišten za autentifikaciju korisnika. Primjer procesa autentifikacije prikazan na slici 2.9 detaljnije je opisan u nastavku.



Slika 2.9 Primjer procesa autentifikacije koristeći JWT [31]

Pri prijavi ili registraciji poslužitelj za korisnika stvara novi JWT potpisan privatnim ključem koji se na strani klijenta sprema u *localstorage* preglednika. Ubuduće će klijent taj JWT koristiti za pristup resursima prosljeđivanjem JWT-a u zaglavlju svakog HTTP zahtjeva. Autentifikcija JWT-om je bez čuvanja stanja (eng. *stateless*), što znači da se podaci ne spremaju sa strane poslužitelja, već poslužitelj provjerava autentičnost tokena pomoću privatnog ključa [32].

Struktura JWT-a sastoji se od zaglavlja, sadržaja (eng. *payload*) i potpisa kao što je prikazano na slici 2.10.



Slika 2.10 Struktura JWT-a [30]

U zaglavlju, "*alg*" specificira vrstu *hashing* algoritma koji će se koristiti, a "*typ*" je vrsta tokena. U sadržaju se nalaze podaci o korisniku, a u potpisu se uzima Base64-kodirano zaglavlje i sadržaj, zajedno s tajnom vrijednošću (eng. *secret*) i kodiraju se algoritmom navedenim u zaglavlju. JWT je nakon kodiranja oblika "xxxxx.yyyyy.zzzzz", primjer na slici 2.11.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZ
SI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P
Ok6yJV_adQssw5c
```

Slika 2.11 Kodirani JWT

Za izradu ovog projekta, u Laravelu je korišten paket tymon/jwt-auth [33].

Poglavlje 3

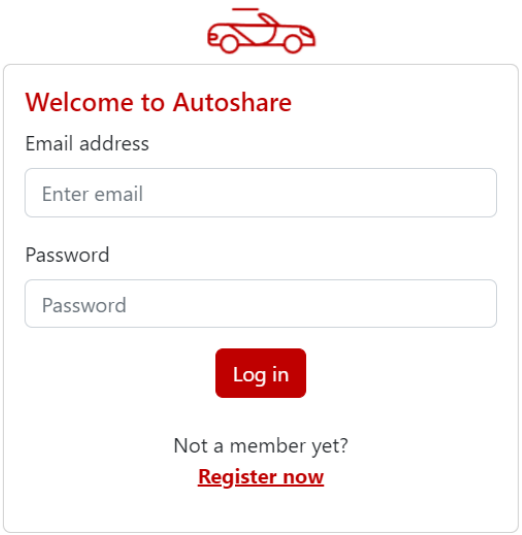
Opis aplikacije

U ovom poglavlju biti će prikazano korisničko sučelje, detaljno objašnjene sve funkcionalnosti aplikacije, te kako ih koristiti.

Korisniku je onemogućen pristup funkcionalnostima aplikacije bez prethodne prijave odnosno izrade korisničkog računa. Dok korisnik nije prijavljen u aplikaciju, putem navigacijske trake dostupne su mu jedino stranice za prijavu i registraciju (slika 3.1).

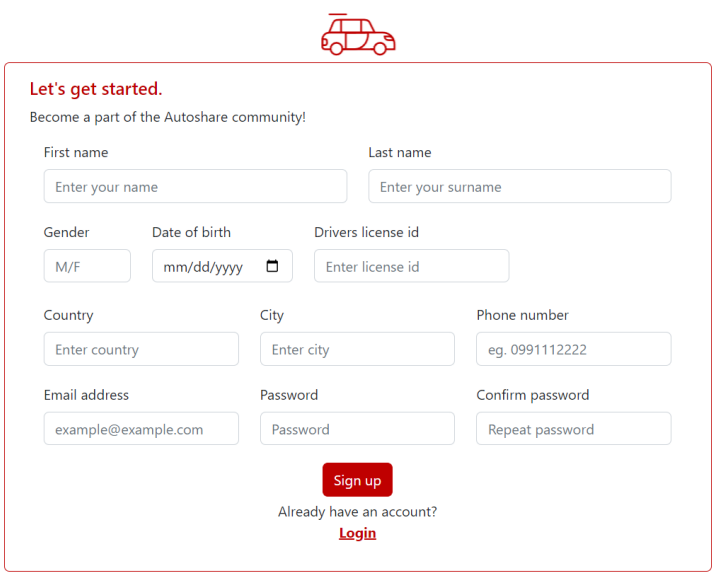
Slika 3.1 Opcije navigacijske trake za neprijavljenog koriniska

Stranica za prijavu u aplikaciju prikazana je na slici 3.2. Za uspješnu prijavu potrebno je unijeti *e-mail* adresu i lozinku.



Slika 3.2 Izgled *Login* forme za prijavu

Ukoliko korisnik prvi puta posjećuje aplikaciju, odabirom opcije *Register* u navigacijskoj traci ili na dnu *Login* prozora dolazi do forme za izradu novog korisničkog računa prikazane na slici 3.3. Za uspješnu registraciju potrebno je unijeti sva polja i pritisnuti *Sign up* gumb.

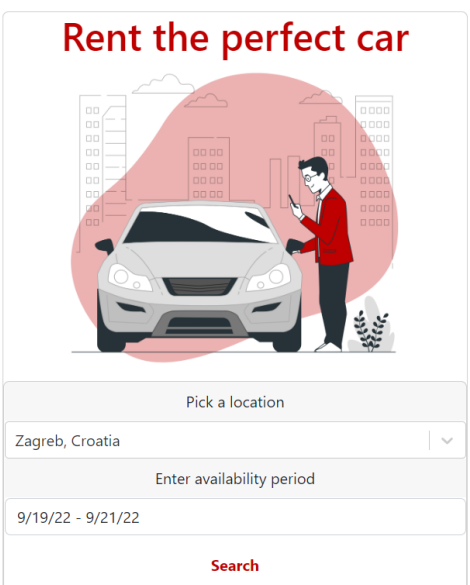


Slika 3.3 Izgled *Registration* stranice za prijavu

Pri uspješnoj prijavi ili registraciji, korisnik je preusmjeren na glavnu stranicu za pretraživanje vozila za najam, te putem navigacijske trake dobiva pristup svim funkcionalnostima aplikacije. Opcije navigacijske trake za prijavljenog korisnika su redom: *Find a vehicle*, *Rent out*, *Gas prices*, *Weather forecast*, *All my trips*, *My vehicles*, *My profile* i *Logout* (slika 3.4).

Slika 3.4 Opcije navigacijske trake za prijavljenog koriniska

Prva stanica *Find a vehicle*, koja je prikazana na slici 3.5, služi za korisnike koji su u potrazi za iznajmljivanjem tuđih vozila. Korisnik može započeti unos naziva mjesta ili grada, te među ponuđenim lokacijama odabire onu unutar koje želi pretražiti dostupna vozila. Uz to je potrebno unijeti i period u kojem se želi iznajmiti vozilo. Pritiskom na gumb *Search*, korisnik je preusmjeren na stranicu prikazanu na slici 3.6, sa svim vozilima koja odgovaraju unesenim kriterijima.



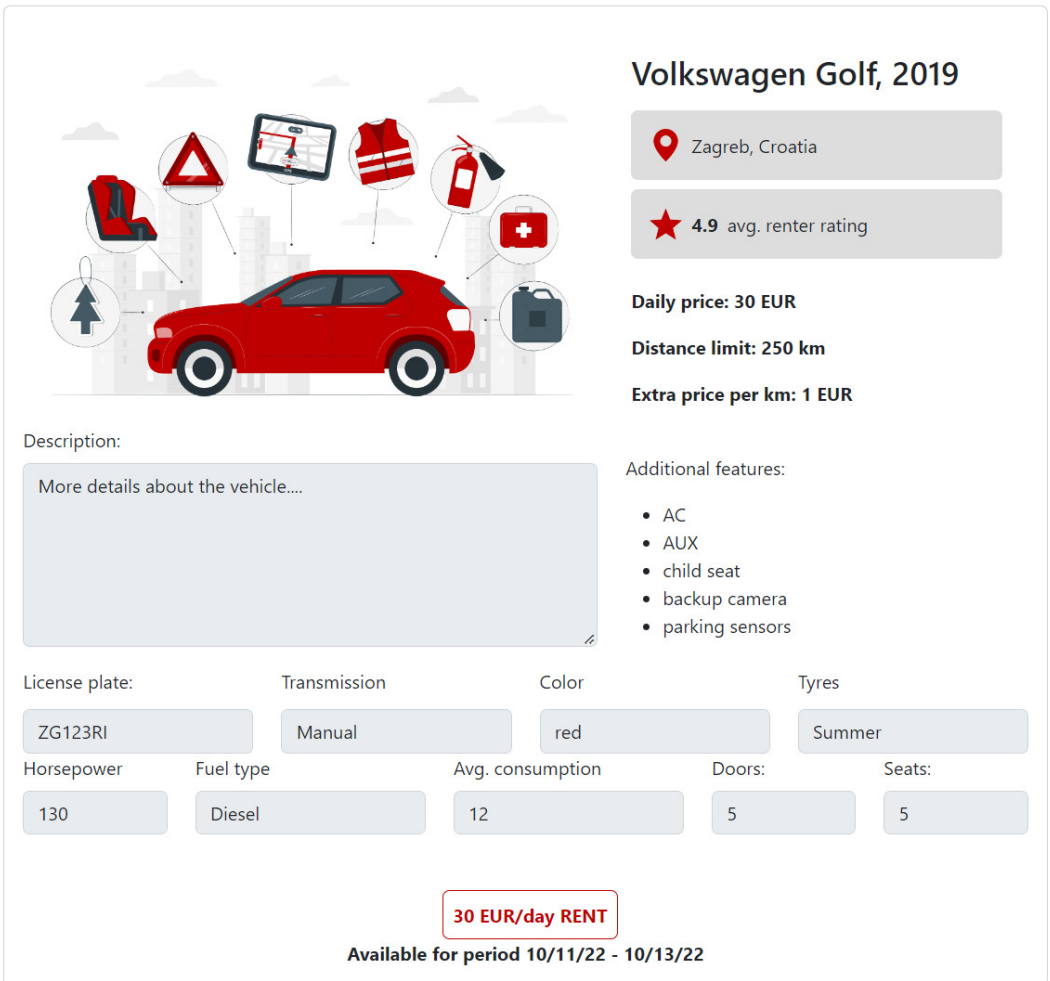
Slika 3.5 Izgled *Find a vehicle* stranice za pretraživanje dostupnih vozila

Pritiskom na gumb *Search*, korisnik je preusmjeren na stranicu prikazanu na slici 3.6, sa svim vozilima koja odgovaraju unesenim kriterijima.

Slika 3.6 Izgled stranice s rezultatima pretrage dostupnih vozila

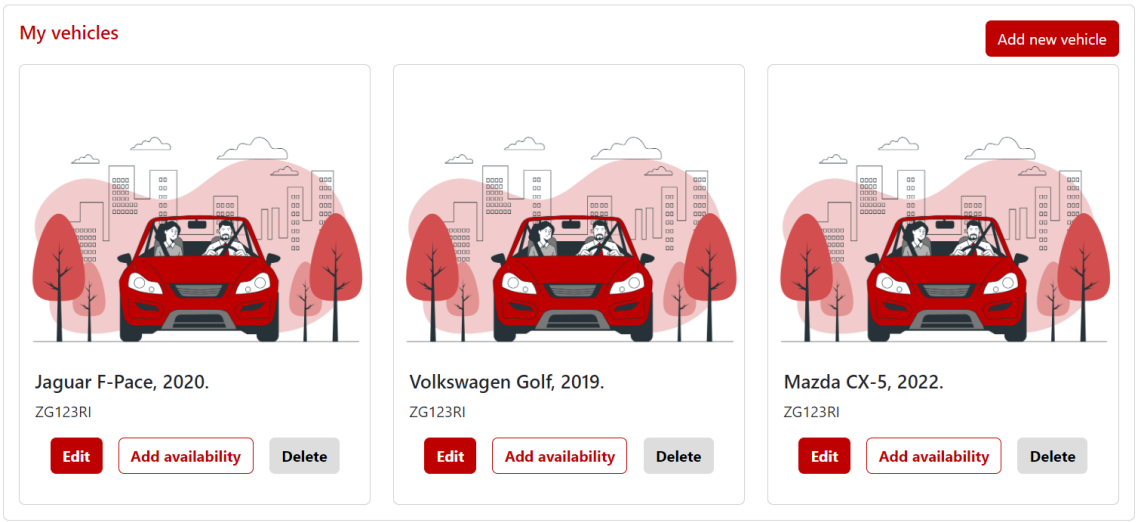
Odabirom gumba *More details*, za svako od ponuđenih vozila otvara se prikaz dodatnih detalja o najmu i specifikacijama u svrhu olakšanja odabira vozila koje korisniku najviše odgovara. To su informacije o cijeni najma, specifikacijama vozila i prosječnoj ocjeni iznajmljivača, prikazane na slici 3.7.

Na dnu te stranice nalazi se gumb za rezerviranje najma, gdje je još jednom potvrđena cijena jednog dana najma i period u kojem korisnik uzima auto na korištenje. Korisnik mora imati unesenu vozačku dozvolu kako bi smio unajmiti vozilo. Nakon što se korisnik odluči za vozilo, ono se više se neće pojavljivati u pretraživanju među vozilima dostupnima za taj period.



Slika 3.7 Izgled stranice s detaljima o najmu i vozilu

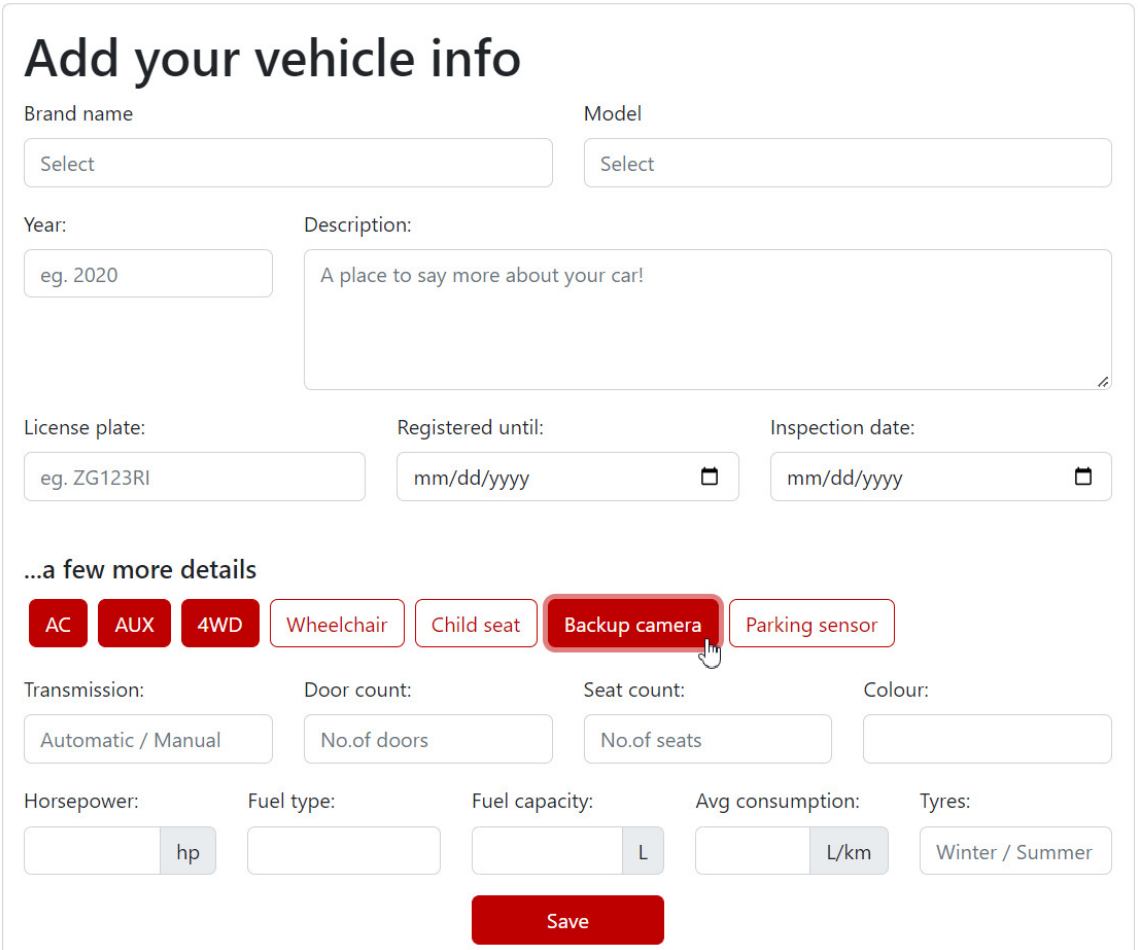
Peta opcija navigacijske trake *My vehicles*, prikazane na slici 3.8, nudi četiri funkcionalnosti, a to su dodavanje, uređivanje i brisanje osobnih vozila i dodavanje dostupnosti za pojedino vozilo. Dodavanje vozila obavezno je samo ukoliko korisnik želi dati vlastito vozilo u najam, a nije potrebno ukoliko korisnik aplikaciju koristi samo za najam tuđih vozila.



Slika 3.8 Izgled *My vehicles* stranice sa svim vozilima korisnika

Odabirom gumba *Add new* otvara se forma za unos novog vozila prikazana na slici 3.9. Odabirom gumba *Delete* vozilo se briše.

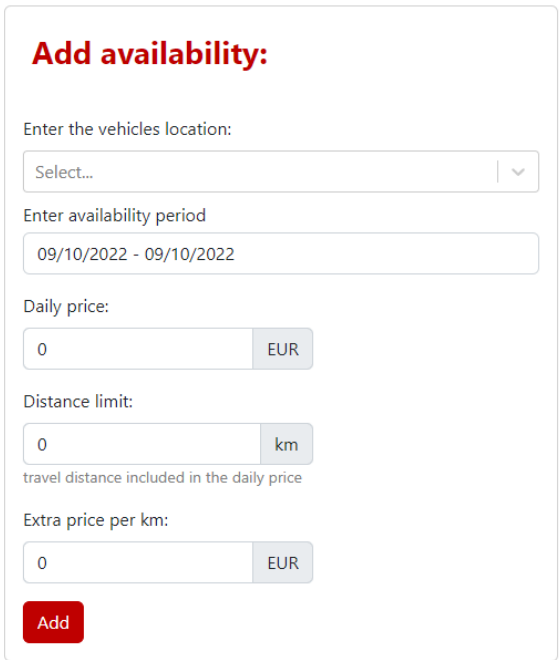
Odabirom gumba *Edit* otvara se forma za uređivanje glavnih podataka o vozilu i svojstava vozila. Izgledom je ista kao i forma za unos novog vozila. Ova opcija uvedena je budući da vozila sadrže informacije koje se kroz vrijeme mijenjaju, poput vrste guma s obzirom na godišnje doba, uvođenje ili uklanjanje dodatne opreme, novi datumi trajanja registracijske tablice i datum prolaska tehničkog pregleda.



Slika 3.9 Izgled *Add vehicles* stranice za dodavanje vozila

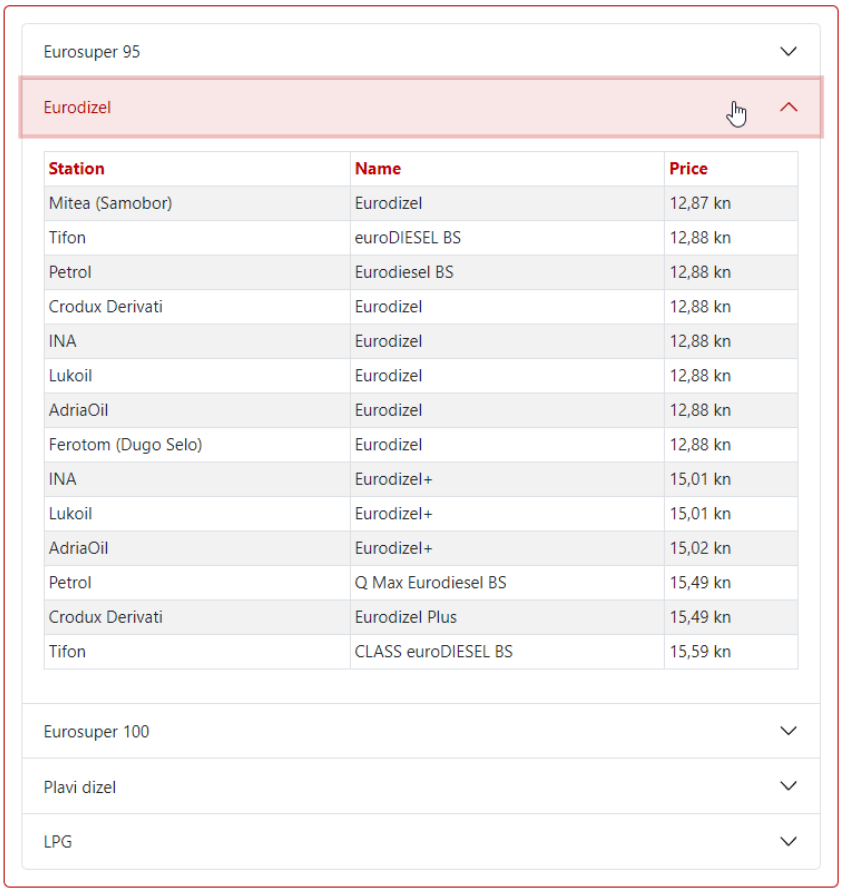
Korisnik je slobodan dati vozilo u najam uz uvjet da nije istekao datum važenja registracijske tablice i da je prošlo manje od godinu dana od zadnjeg tehničkog pregleda. U suprotnom vozilo neće biti moguće dati u najam, što je indicirano onemogućenim gumbom *Add availability*.

Prvi korak davanja vozila u najam je odabir vozila za koje se namjerava dodati dostupnost pritiskom na gumb *Add availability*, a zatim unos potrebnih informacija u novu formu prikazanu na slici 3.10. Period dostupnosti ne unosi se ručno već se otvara kalendar za unos početnog i krajnjeg datuma. Kako bi dodavanje dostupnosti bilo uspješno, potrebno je ispuniti sva polja i pritisnuti gumb *Save*, nakon čega je korisnik preusmjeren natrag na stranicu za pregled osobnih vozila.



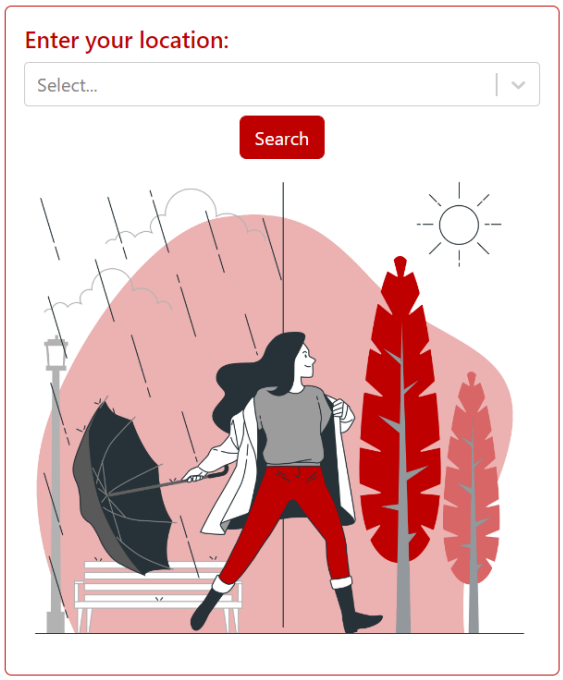
Slika 3.10 Izgled *Availability* stranice za dodavanje dostupnosti

Prve dvije opisane opcije navigacijske trake omogućuju korisnicima funkcionalnosti davanja vlastitih vozila u najam, kao i najam tuđih vozila. Sljedeće dvije opcije navigacijske trake korisniku pružaju informacije korisne za vrijeme samog putovanja. Odabirom opcije *Gas prices*, dostupne su najnovije informacije o povoljnosti cijena goriva na različitim lokacijama tj. benzinskim postajama u Hrvatskoj kao što je prikazano na slici 3.11.



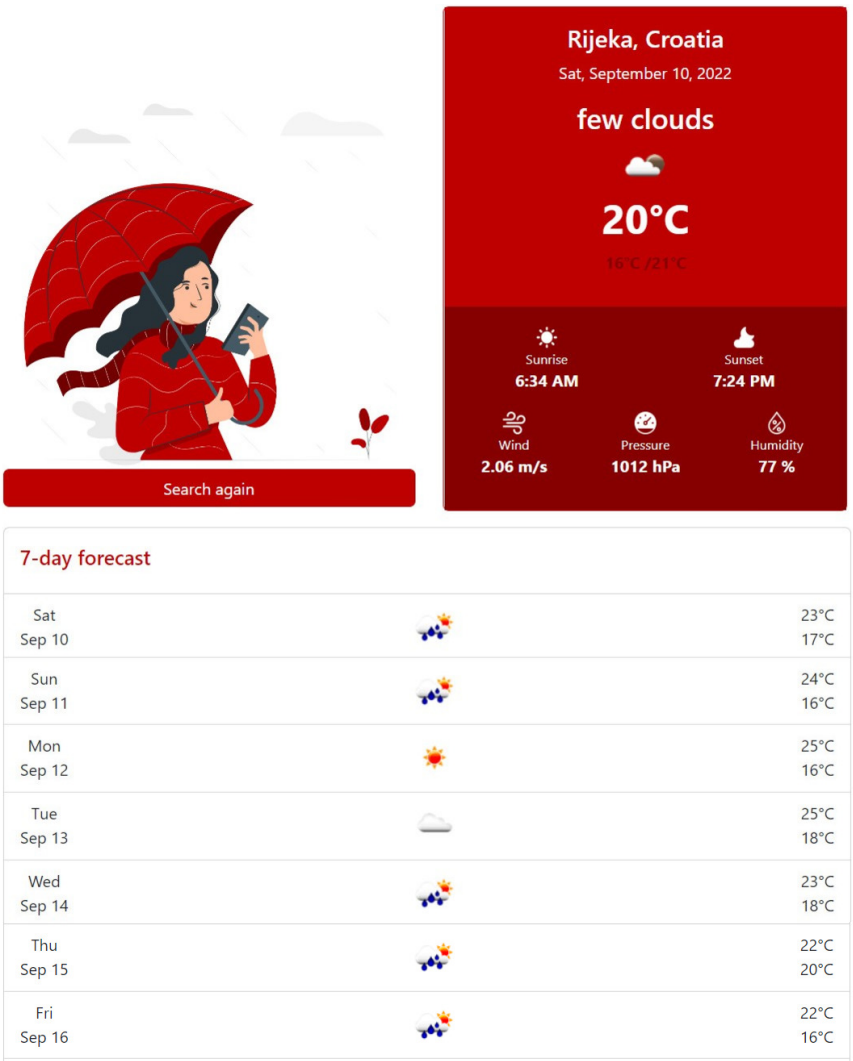
Slika 3.11 Izgled stranice *Gas prices* za uvid u cijene goriva na različitim lokacijama

Odabirom opcije *Weather forecast* nudi se unos mjesta ili grada među ponuđenim nazivima, slika 3.12.



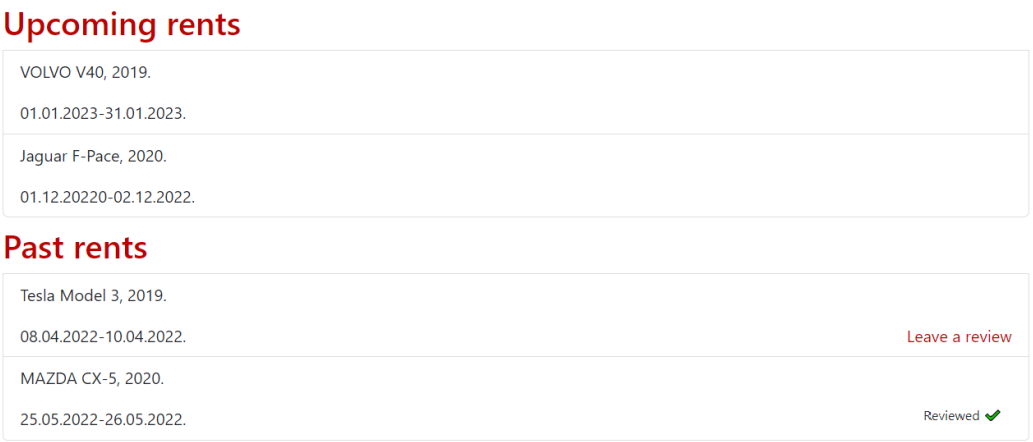
Slika 3.12 Izgled *Weather forecast* početne stranice za pretraživanje

Pritiskom na gumb *Search* učitava se stranica s vremenskom prognozom prikazana na slici 3.13. Prikazuje se dnevna vremenska prognoza koja se sastoji od trenutne temperature, slika, maksimalne i minimalne temperature, vremena izlaska i zalaska sunca, količine vlage, pritiska i jačine vjetra. Uz to, prikazuje se i sedmodnevna vremenska prognoza s datumima, predviđenom minimalnim i maksimalnim temperaturama i slikom. Pritiskom na gumb *Search again* korisnik je vraćen na početnu stranicu za odabir lokacije.



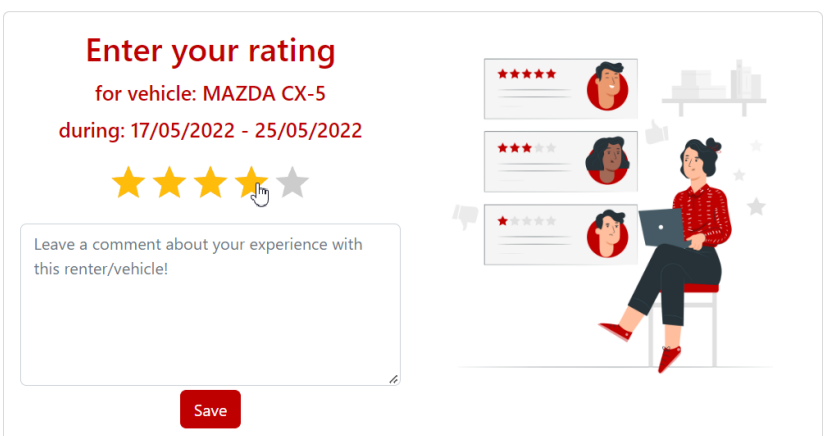
Slika 3.13 Izgled *Weather forecast* stranice nakon pretraživanja po lokaciji

Pod petom opcijom navigacijske trake *All my trips* nalazi se povijest iznajmljivanja vozila u kronološkom redoslijedu, kao i najmovi koji su tek nadolazeći. Svrha ove stranice je pregled svih najmova i mogućnost ostavljanja recenzije za završene najmove. Recenzija služi kao pomoć drugim korisnicima pri odabiru vozila i može se ostaviti jednom po najmu. Stranica *All my trips* je prikazana na slici 3.14.



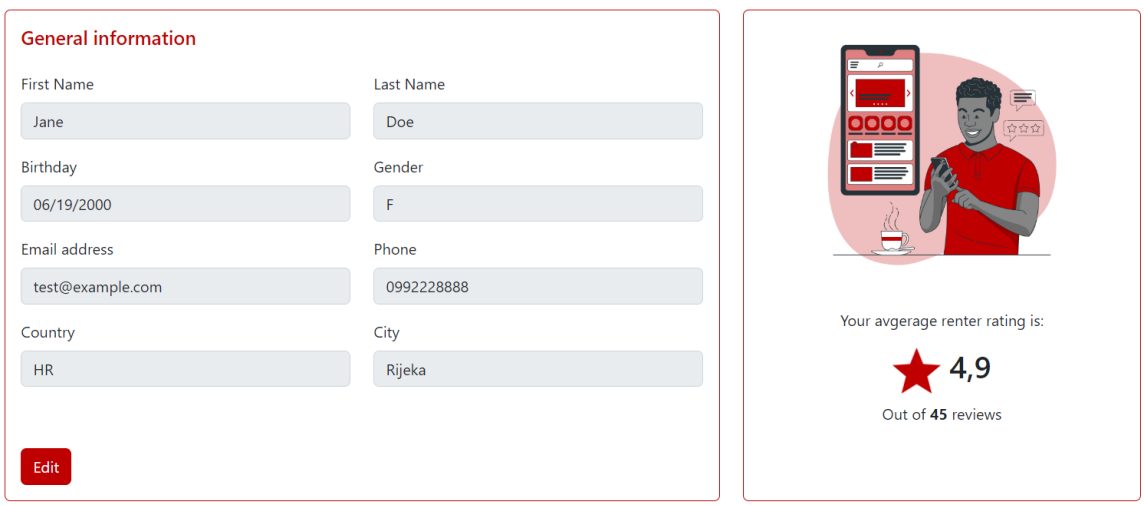
Slika 3.14 Izgled *My trips* stranice za pregled svih najmova

Odabirom opcije *Leave a review* otvara se forma za unos recenzije. Na formi se nalazi vizualni prikaz ocjene u obliku zvjezdica, gdje je potrebno odabrati ocjenu 1-5, i polje za komentar koje je predviđeno za dodatno obrazloženje odabrane ocjene. Pritiskom na gumb *Save* sprema se recenzija i korisnik je preusmjeren natrag na stranicu za pregled vlastitih najmova. Ova forma prikazana je na slici 3.15



Slika 3.15 Izgled *Add review* stranice za dodavanje recenzije

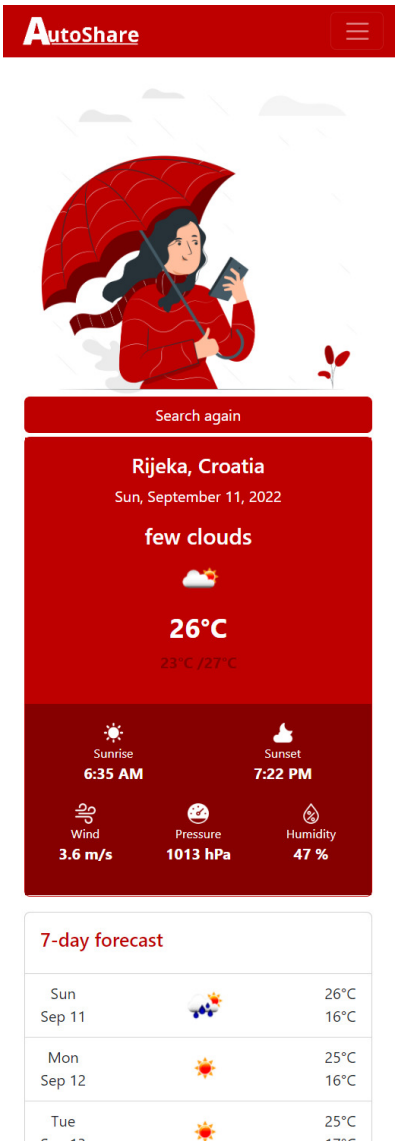
Predzadnja opcija na navigacijskoj traci *My profile* vodi korisnika na stranicu s osobnim podacima. Kartica na lijevoj strani prikazuje osobne podatke o korisniku, a pritiskom na gumb *Edit* omogućuje se uređivanje polja s podacima i gumb se pretvara u *Save* gumb za spremanje podataka. Sva polja moraju biti unesena kako bi se podaci mogli spremiti, nakon čega se polja i gumb opet vraćaju u početno stanje. Kartica na desnoj strani sadrži podatke koje korisnik ne može izmjenjivati i prikazuje podatke o prosječnoj ocjeni recenzija iznajmljivača, te ukupan broj recenzija. Obje kartice vidljive su na slici 3.16.



Slika 3.16 Izgled *My profile* stranice s osobnim podacima korisnika

Zadnja opcija navigacijske trake je *Logout*. Odabirom te opcije korisnik se odjavljuje iz svog korisničkog računa te je preusmjeren natrag na početnu *Login* stranicu za prijavu.

Aplikacija se također učinkovito renderira na zaslonima različitih veličina, pa je na slici 3.17 prikazan izgled *Weather forecast* stranice na zalsonu veličine mobilnog uređaja.



Slika 3.17 Izgled *Weather forecast* stranice na mobilnom uređaju

Ilustracije i ikone su preuzete sa *storyset* [34], i *freepik* [35] stranica.

Poglavlje 4

Opis funkcionalnosti

U ovom poglavlju biti će opisane funkcionalnosti dodavanja vozila i pretraživanja dostupnih vozila na razini koda s klijentske i poslužiteljske strane.

4.1 Dodavanje vozila

4.1.1 Unos podataka i slanje zahtjeva

Dodavanje vozila počinje pritiskom na gumb *Add vehicle* na stranici *My vehicles*. Pritiskom gumba, *Link* komponenta (slika 4.1) mijenja rutu iz */vehicles* u */add*.

```
<Link to='/add'><Button>Add new vehicle</Button></Link>
```

Slika 4.1 Link s gumbom za navigaciju do AddVehicle komponente

React Router tada renderira komponentu *AddVehicle* za dodavanje vozila, što je definirano u App.js datoteci (slika 4.2).

```
<Route exact path='/add' element={<AddVehicle />} />
```

Slika 4.2 Ruta za *AddVehicle* iz *App.js*

Komponenta *AddVehicle*, prikazana u prethodnom poglavlju na slici 3.9, sadrži formu za unos koja se sastoji od Bootstrap kartice, redaka i stupaca u kojima su raspoređeni opisi, polja za unos i potvrdni okviri (eng. *checkbox*) u obliku gumba za prebacivanje (eng. *toggle button*).

Početno stanje svih vrijednosti je prazan niz znakova, nula ili *false* (slika 4.3). Budući da su informacije o vozilu raspoređene u dvije tablice u bazi, stanje komponente također je podijeljeno u dva objekta *vehicleInfo* i *vehicleFeatures* zbog jednostavnijeg slanja na poslužitelj i spremanja u bazu.

```
const [vehicleInfo, setVehicleInfo] = useState( initialState: {
  year: '',
  description: '',
  license_plate: '',
  registered_until: '',
  inspection_date: ''
});
const [vehicleFeatures, setVehicleFeatures] = useState( initialState: {
  transmission: '',
  door_count: 0,
  seat_count: 0,
  heated_seats: false,
  ac: false,
  aux: false,
  colour: '',
  drivetrain: false,
  horsepower: '',
  fuel_capacity: 0.0,
  fuel_type: '',
  tyres: '',
  avg_consumption: 0.0,
  wheelchair: false,
  child_seat: false,
  backup_camera: false,
  parking_sensors: false
});
```

Slika 4.3 Početno stanje komponente AddVehicle

Prva dva polja za unos očekuju nazive proizvođača i modela vozila. Izvor podataka za *autocomplete* je vPIC API. Čim se komponenta prikaže, u pozadini se dohvati popis proizvođača pomoću *useEffect hook-a* koji poziva *fetchAllBrands* funkciju. Drugi *useEffect hook* za dohvaćanje modela izvršava svaki put kada se promijeni odabrani proizvođač. Oba su vidljiva na slici 4.4.

```
useEffect( effect: () => {
  async function fetchData() {
    setBrands(await fetchAllBrands());
  }

  fetchData();
}, deps: []);

useEffect( effect: () => {
  if (pickedBrand) {
    async function fetchData() {
      setModels(await fetchAllModels(pickedBrand));
    }

    fetchData();
  }
}, deps: [pickedBrand]);
```

Slika 4.4 *useEffect hook*-ovi za dohvaćanje proizvođača i modela vozila

Funkcija *fetchAllBrands* šalje GET zahtjev API-ju, iz dobivene liste objekata izvlači samo imena proizvođača, sortira ih i vraća rezultat. Funkcija *fetchAllModels* radi istu stvar, uz dodatni argument imena proizvođača kojeg treba poslati u URL-u (eng. *Uniform Resource Locator*) (slika 4.5).

```
export async function fetchAllBrands() {
  const { data } = await axios.get('https://vpic.nhtsa.dot.gov/api/vehicles/getallmakes?format=json');
  const brandNames = data.Results.map(element => element.Make_Name);
  brandNames.sort();
  return brandNames;
}

export async function fetchAllModels(brand) {
  const { data } = await axios.get(`https://vpic.nhtsa.dot.gov/api/vehicles/getmodelsformake/${brand}?format=json`);
  const modelNames = data.Results.map(element => element.Model_Name);
  modelNames.sort();
  return modelNames;
}
```

Slika 4.5 Funkcije za dohvaćanje proizvođača i modela vozila

Iz preostalih polja za unos pomoću *onChange* funkcije postavljaju se vrijednosti stanja komponente (slika 4.6).

```
function onChange(event) {
  setVehicleInfo( value: {
    ...vehicleInfo,
    [event.target.name]: event.target.value
  });
}

function onFeatureChange(event) {
  setVehicleFeatures( value: {
    ...vehicleFeatures,
    [event.target.name]: event.target.value
  });
}
```

Slika 4.6 Funkcije koje postavljaju stanje komponente

Prikaz i unos vrijednosti boolean varijabli ostvaren je korištenjem komponente *Togglebutton*. Kako bi se gumbi prikazali kao označeni ili ne označeni, u *value* svojstvu grupe moraju se nalaziti imena svih *true* vrijednosti. To se postiže filtriranjem *true* boolean vrijednosti iz vehicleFeatures stanja u listu naziva (slika 4.7).

```
for (const [name, value] of Object.entries(vehicleFeatures)) {
  if (value === true) {
    selectedToggles.push(name);
  }
}
```

Slika 4.7 Filtriranje *boolean* svojstava

Ta se lista predaje *value* svojstvu grupe (slika 4.8).

```
<ToggleButtonGroup type='checkbox' onChange={toggleFeatures} value={selectedToggles}>
  <ToggleButton variant='outline-primary' id='ac-id' value='ac' className='rounded m-1'>AC</ToggleButton>
  <ToggleButton variant='outline-primary' id='aux-id' value='aux' className='rounded m-1'>AUX</ToggleButton>
  <ToggleButton variant='outline-primary' id='drivetrain-id' value='drivetrain' className='rounded m-1'>4WD</ToggleButton>
  <ToggleButton variant='outline-primary' id='wheelchair-id' value='wheelchair' className='rounded m-1'>Wheelchair</ToggleButton>
  <ToggleButton variant='outline-primary' id='child_seat-id' value='child_seat' className='rounded m-1'>Child seat</ToggleButton>
  <ToggleButton variant='outline-primary' id='backup_camera-id' value='backup_camera' className='rounded m-1'>Backup camera</ToggleButton>
  <ToggleButton variant='outline-primary' id='parking_sensors-id' value='parking_sensors' className='rounded m-1'>Parking sensor</ToggleButton>
</ToggleButtonGroup>
```

Slika 4.8 Ispis označenih/ne označenih *toggle* gumbova

Promjenom stanja *toggle* gumbova upravlja funkcija *toggleFeatures* (slika 4.9).

```
function toggleFeatures(trueNames) {
  setVehicleFeatures( value: oldFeatures => {
    let nextFeatures = { ...oldFeatures };

    for (const name in nextFeatures) {
      if (typeof nextFeatures[name] === 'boolean') {
        nextFeatures[name] = trueNames.includes(name);
      }
    }

    return nextFeatures;
  });
}
```

Slika 4.9 Funkcija *toggleFeatures* koja upravlja promjenom stanja *true/false*

Za dodavanje vozila, po unosu svih polja potrebno je pritisnuti *Save* gumb tipa *SUBMIT*, koji pokreće slanje forme. Poziva se funkcija *onSubmit* prikazana na slici 4.10. U funkciji se pomoću *preventdefault()* sprječava da preglednik automatski pošalje formu, a zatim se šalju podaci pomoću *POST* zahtjeva na poslužitelj koristeći instancu *axios*-a konfiguriranu za autentifikaciju (slika 4.11). Tijelo zahtjeva je JSON objekt s dvije vrijednosti- informacije o vozilu *"vehicle"* i značajke vozila *"features"*.

```
function onSubmit(e) {
  e.preventDefault();

  const data = {
    vehicle: { brand: pickedBrand, model: pickedModel, ...vehicleInfo },
    features: vehicleFeatures
  };

  axios.post('/vehicles', data)
    .then(response => {
      alert('Successfully added!');
      navigate('/vehicles');
    })
    .catch(error => {
      let msg = 'Error adding:\n';
      for (let key in error.response.data) {
        msg += `${key}: ${error.response.data[key]}\n`;
      }
      alert(msg);
    });
}
```

Slika 4.10 Funkcija *onSubmit*

Ovako konfigurirana *axios* instanca (slika 4.11) služi za slanje JWT-a uz svaki zahtjev prema poslužitelju, osim kada se radi o registraciji ili prijavi u aplikaciju.

```
import axios from 'axios';

function createInstance() {
  const config = {
    baseURL: 'http://localhost:8000/api',
    headers: {
      'Content-Type': 'application/json'
    }
  };

  let instance = axios.create(config);

  instance.interceptors.request.use( onFulfilled: (config : AxiosRequestConfig ) => {
    if (config.url === '/user/login' || config.url === '/user/register') {
      return config;
    }

    const token = JSON.parse(localStorage.getItem( key: 'jwt'));
    // noinspection JSUnresolvedVariable
    config.headers.Authorization = token ? `${token.token_type} ${token.access_token}` : '';
    return config;
  });

  return instance;
}

export default createInstance();
```

Slika 4.11 *Axios* instanca konfigurirana za autentifikaciju

4.1.2 Obrada podataka na poslužitelju

Na poslužitelju se taj zahtjev prihvaća preko *endpointa* za dodavanje vozila prikazanog na slici 4.12, koji vodi na funkciju *addVehicle*.

```
Route::group([
    'middleware' => 'api',
    'prefix' => 'vehicles'
], function ($router) {
    Route::post( uri: '/', [VehicleController::class, 'addVehicle']);
    Route::get( uri: '/', [VehicleController::class, 'getUserVehicles']);
    Route::get( uri: '/{id}', [VehicleController::class, 'getVehicle']);
    Route::put( uri: '/{id}', [VehicleController::class, 'updateVehicle']);
    Route::delete( uri: '/{id}', [VehicleController::class, 'destroy']);
});
```

Slika 4.12 Krajnje točke (eng. *endpoints*) za vozila

Funkcija za dodavanje vozila nalazi se u *VehicleControlleru*. Prvo se iz primljenog objekta razdvajaju i validiraju vrijednosti o vozilu (slika 4.13) i značajke vozila. Značajke vozila validiraju se po istom principu.

```
public function addVehicle(Request $request)
{
    $validator = Validator::make($request->get( key: 'vehicle'), [
        'brand' => 'required|string|max:30',
        'model' => 'required|string|max:30',
        'year' => 'required|string|max:4',
        'description' => 'required|string|max:250',
        'license_plate' => 'required|string|max:8',
        'registered_until' => 'required|date',
        'inspection_date' => 'required|date'
    ]);
```

Slika 4.13 Provjera primljenih vrijednosti

Ako prođu potvrdu, stvara se novi zapis u *Vehicle Features*, a potom u *Vehicles*

tablici. Strani ključ *vehicle_features_id* zapisuje se iz prethodno stvorenih svojstava, a strani ključ *owner_id* zapisuje se iz dekriptiranog JWT-a (slika 4.14).

```
if ($validator->fails() || $featureValidator->fails()) {
    return response()->json(array_merge($validator->errors()->toArray(),
        $featureValidator->errors()->toArray()), status: 422);
}

$features = VehicleFeatures::create($featureValidator->validated());
$vehicle = Vehicle::create(array_merge(
    $validator->validated(),
    ['vehicle_features_id' => $features->id, 'owner_id' => auth()->user()->id]
));
```

Slika 4.14 Spremanje vozila i značajki vozila u bazu

Odgovor uspješnog stvaranja sastoji se od JSON objekta s poljima *message*, *vehicle* u kojem je objekt vozila, i HTTP (eng. *Hypertext Transfer Protocol*) statusom 201 koji označava uspješno stvaranje (slika 4.15).

```
return response()->json([
    'message' => 'Vehicle successfully registered.',
    'vehicle' => $vehicle
], status: 201);
```

Slika 4.15 Odgovor funkcije nakon uspješnog dodavanja vozila

4.1.3 Prikaz rezultata

Po primitku uspješnog odgovora s poslužitelja, na klijentu se prikazuje poruka "*Successfully added*". Zatim je pomoću *useNavigate hooka* vidljivog na slici 4.10 korisnik preusmjeren na */vehicles* stranicu za pregled vlastitih vozila na kojoj se prikazuju sva, uključujući i novo dodano vozilo.

4.2 Pretraživanje dostupnih vozila

4.2.1 Unos podataka i slanje zahtjeva

Na stranici s nazivom *Find a vehicle*, koja je početna stranica nakon prijave, nalazi se kartica s dva polja, predviđena za unos lokacije i perioda. Nakon unosa podataka i pritiska na gumb *Search*, poziva se *onSubmit* funkcija prikazana na slici 4.16. Ona iz trenutnog stanja uzima vrijednosti početnog datuma, krajnjeg datuma i identifikatora lokacije. Datumi se formatiraju tako da ne sadrže vremensku vrijednost. U obliku parova "naziv=vrijednost" funkcija prikuplja ta tri podatka u varijablu *params* koju korištenjem *useNavigate() hooka* šalje na rutu */list* putem URL parametara.

```
function onSubmit(event) {
  event.preventDefault();

  const params = {
    place_id: locationData.value.place_id,
    start_date: dates.start_date.toISOString().split( separator: 'T')[0],
    end_date: dates.end_date.toISOString().split( separator: 'T')[0]
  };

  navigate(`/list?${new URLSearchParams(params).toString()}`);
}
```

Slika 4.16 *onSubmit* funkcija forme s *Find a vehicle* stranice

/list ruta učitava *<ListOfAvail>* komponentu, koja iz URL-a izvlači vrijednosti (slika 4.17).

```
const query = useQuery();
const place_id = query.get('place_id');
const start_date = query.get('start_date');
const end_date = query.get('end_date');
```

Slika 4.17 *onSubmit* funkcija forme s *Find a vehicle* stranice

Zatim se poziva *useEffect hook* koji priprema podatke i šalje ih putem *POST* zahtjeva na poslužitelj (slika 4.18).

```
useEffect( effect: () => {
  const data = {
    location: { place_id },
    dates: { start_date, end_date }
  };

  api.post('/availabilities/search', data)
    .then(response => setSearchResults(response.data.search_results));
}, deps: [place_id, start_date, end_date]);

if (!searchResults) {
  return <h2>Loading...</h2>;
} else if (searchResults.length === 0) {
  return <h2>None!</h2>;
}
```

Slika 4.18 *useEffect* funkcija za dohvaćanje dostupnih vozila i poruke učitavanja

4.2.2 Obrada podataka na poslužitelju

Na poslužitelju se taj zahtjev prihvaća preko *endpointa* za pretraživanje dostupnih vozila prikazanog na slici 4.19, koji vodi na funkciju *search*.

```
Route::group([
    'middleware' => 'api',
    'prefix' => 'availabilities'
], function ($router) {
    Route::post( uri: '/', [AvailabilityController::class, 'addAvailability']);
    Route::get( uri: '/{id}', [AvailabilityController::class, 'get']);
    Route::post( uri: '/search', [AvailabilityController::class, 'search']);
});
```

Slika 4.19 Krajnje točke (*enpoints*) za dostupnosti vozila

Funkcija za pretraživanje dostupnih vozila nalazi se u *AvailabilityControlleru*. Prvo se, kao i u prethodnom poglavlju, iz primljenog objekta razdvajaju i validiraju vrijednosti. (slika 4.20)

```
public function search(Request $request)
{

    $placeValidator = Validator::make($request->get( key: 'location'), [
        'place_id' => 'required|string',
    ]);

    $datesValidator = Validator::make($request->get( key: 'dates'), [
        'start_date' => 'required|date',
        'end_date' => 'required|date'
    ]);
```

Slika 4.20 Validacija primljenih podataka

Ako prođu potvrdu, spremaju se u varijable kako bi se dalje mogle koristiti za pretragu. (slika 4.21)

```
$place = $placeValidator->validated();
$dates = $datesValidator->validated();

$place_id = $place['place_id'];
$start_date = Carbon::parse($dates['start_date'])->startOfDay();
$end_date = Carbon::parse($dates['end_date'])->endOfDay();
```

Slika 4.21 Pripremanje validiranih podataka

Zatim slijedi pretraživanje na temelju ta tri podatka. Za slaganje upita korišten je Eloquent *query builder* kao zamjena za pisanje SQL upita. Nakon tri *where* uvjeta, *map* funkcijom izdvajaju se i vraćaju samo potrebni podaci - *id*, proizvođač, model i godina (slika 4.22). Lista rezultata šalje se klijentu u obliku *JSON* objekta.

```
$search_results = Availability::where('place_id', '=', $place_id)
    ->where('start_date', '<=', $start_date->toDate())
    ->where('end_date', '>=', $end_date->toDate())
    ->get()->map(function ($item) {
        return [
            'availability_id' => $item->id,
            'vehicle_brand' => $item->vehicle->brand,
            'vehicle_model' => $item->vehicle->model,
            'vehicle_year' => $item->vehicle->year
        ];
    })->all();


return response()->json(['search_results' => $search_results]);
```

Slika 4.22 Eloquent *database query builder* i *return* vrijednost

4.2.3 Prikaz rezultata

Kao što je vidljivo na slici 4.18, dok klijent čeka odgovor, odnosno dok je varijabla *SearchResults* prazna, stranica će ispisati poruku "*Loading...*". Po primitku povratne vrijednosti u *useEffect()* funkciji, *SearchResults* stanje se postavlja, te se vraća poruka "*None!*" u slučaju da nema vozila koji odgovaraju pretragi, ili se vraća se prikaz svih rezultata. Ovaj prikaz vidljiv je u prethodnom poglavlju na slici 3.6.

Poglavlje 5

Zaključak

Autoshare aplikacija namjenjena je za korisnike koji bi htjeli svoja osobna vozila dati u najam u periodu u kojem ih neće koristiti, ili za ljude koji imaju želju iznajmiti neko vozilo, za putovanje ili probu vozila, po povoljnijim cijenama u odnosu na velike kompanije za najam automobila. Kako bi se osigurala sigurnost i u najam bila dozvoljena jedino ispravna vozila, korisnicima pomaže pratiti važne datume poput isteka registracijske tablice i datuma posljednjeg tehničkog pregleda.

Uz mogućnost najma, aplikacija nudi svojim korisnicima korisne informacije za vrijeme trajanja samog putovanja. To su informacije o aktualnim cijenama na različitim benzinskim postajama u Hrvatskoj, kako bi korisnik imao uvid u mjesta s najpovoljnijim cijenama. Uz to dostupan je i brzi pregled sedmodnevne vremenske prognoze za bilo koju lokaciju.

Aplikacija također ima potencijala za poboljšanje. U budućnosti, aplikacija bi se mogla poboljšati uvođenjem dodatnih funkcionalnosti kao što su dodavanje fotografija i prijava oštećenja auta za vrijeme najma radi naplaćivanja dodatnog troška za podmirivanje štete. Također, komunikacija između dvaju korisnika u obliku razgovora (eng. *chat*) bila bi praktična radi lakšeg dogovora oko točnog vremena i mjesta preuzimanja ili povratka vozila.

Aplikacija bi se mogla vizualno poboljšati korištenjem GPS-a (eng. Global Positioning System) i interaktivnih mapa za prikaz lokacija dostupnih vozila, benzinskih postaja ili stanja na cestama.

Laravel i Postgres zajedno pružaju sve potrebne funkcionalnosti poslužiteljskog dijela aplikacije.

React knjižnica prigodna je za ovakav tip aplikacije jer pomaže u upravljanju stanjima koja omogućuju brzo i učinkovito ažuriranje i renderiranje komponenti. Kombinacija JavaScripta i Bootstrapa omogućuje brzu i jednostavnu izradu dinamičnih *web* aplikacija s modernim korisničkim sučeljima koja se mogu ispravno prikazivati na uređajima različitih veličina.

Bibliografija

[1] *PHP*. adresa: `https://www.php.net/downloads.php` (pogledano 30. 7. 2022.).

[2] *Laravel*. adresa: `https://laravel.com/docs/8.x` (pogledano 30. 10. 2021.).

[3] *PostgreSQL*. adresa: `https://www.enterprisedb.com/downloads/postgres-postgresql-downloads` (pogledano 4. 1. 2022.).

[4] R. S. N. Matthew, *Beginning Databases with PostgreSQL*, 2. izdanje. Apress, 2005.

[5] *React.js*. adresa: `https://github.com/facebook/react/` (pogledano 30. 7. 2022.).

[6] *React Bootstrap*. adresa: `https://github.com/react-bootstrap/react-bootstrap` (pogledano 30. 7. 2022.).

[7] *PhpStorm*. adresa: `https://www.jetbrains.com/phpstorm/download/#section=windows` (pogledano 30. 10. 2021.).

[8] *WebStorm*. adresa: `https://www.jetbrains.com/webstorm/download/#section=windows` (pogledano 30. 10. 2021.).

[9] *Google places API*. adresa: `https://developers.google.com/maps/documentation/places/web-service/autocomplete` (pogledano 30. 7. 2022.).

[10] *Open weather API*. adresa: `https://openweathermap.org/api` (pogledano 30. 7. 2022.).

[11] *vPIC API*. adresa: `https://vpic.nhtsa.dot.gov/api/` (pogledano 30. 7. 2022.).

[12] M. Stauffer, *Laravel: Up & Running: A Framework for Building Modern PHP Apps*, 2. izdanje. O'Reilly Media, 2019.

[13] *Slika MVC arhitekture.* adresa: https://uniandes-se4ma.gitlab.io/books/chapter8/mvc-mvvm-mv-mvwhat.html (pogledano 10. 9. 2022.).

[14] B. M. Bean, *Laravel 5 Essentials.* Packt, 2015.

[15] *Eloquent.* adresa: https://laravel.com/docs/8.x/eloquent (pogledano 10. 9. 2022.).

[16] *PostgreSQL documentation.* adresa: https://www.postgresql.org/docs/12/ (pogledano 10. 9. 2022.).

[17] *Types of database models.* adresa: https://hevodata.com/learn/types-of-database-models/#:~:text=A%5C%20Database%5C%20Model%5C%20is%5C%20a,the%5C%20most%5C%20common%5C%20Database%5C%20Model (pogledano 10. 9. 2022.).

[18] *Types of database models.* adresa: https://www.lucidchart.com/pages/database-diagram/database-models (pogledano 10. 9. 2022.).

[19] M. Thakkar, "Introducing React.js", *Building React Apps with Server-Side Rendering: Use React, Redux, and Next to Build Full Server-Side Rendering Applications.* Berkeley, CA: Apress, 2020., str. 41–91, ISBN: 978-1-4842-5869-9. DOI: 10.1007/978-1-4842-5869-9_2. adresa: https://doi.org/10.1007/978-1-4842-5869-9_2.

[20] *Virtual Document Object Model (DOM).* adresa: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs (pogledano 24. 8. 2022.).

[21] *Koja je razlika između Shadow DOM -a i Virtual DOM -a?* Adresa: https://hr.quish.tv/what-is-difference-between-shadow-dom (pogledano 14. 9. 2022.).

[22] *Babel.* adresa: https://babeljs.io/docs/en/ (pogledano 14. 9. 2022.).

[23] *Hooks at a Glance.* adresa: https://reactjs.org/docs/hooks-overview.html (pogledano 14. 9. 2022.).

[24] *Bootstrap Grid System.* adresa: https://www.w3schools.com/bootstrap/bootstrap_grid_system.asp (pogledano 14. 9. 2022.).

[25] *Bootstrap available breakpoints.* adresa: https://react-bootstrap.github.io/layout/breakpoints/ (pogledano 24. 8. 2022.).

[26] *React Router*. adresa: `https://reactrouter.com/en/main/getting-started/tutorial` (pogledano 14. 9. 2022.).

[27] *Axios*. adresa: `https://www.npmjs.com/package/axios` (pogledano 10. 9. 2022.).

[28] *Promise*. adresa: `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise` (pogledano 14. 9. 2022.).

[29] *Using promises*. adresa: `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises` (pogledano 14. 9. 2022.).

[30] *What is a JWT? Understanding JSON Web Tokens*. adresa: `https://supertokens.com/blog/what-is-jwt` (pogledano 24. 8. 2022.).

[31] *What is JWT?* Adresa: `https://www.javainuse.com/spring/jwt` (pogledano 24. 8. 2022.).

[32] *JSON Web Token Structure*. adresa: `https://auth0.com/docs/secure/tokens/json-web-tokens/json-web-token-structure` (pogledano 24. 8. 2022.).

[33] *tymon/jwt-auth*. adresa: `https://github.com/tymondesigns/jwt-auth` (pogledano 30. 7. 2022.).

[34] *Ilustracije*. adresa: `https://storyset.com/` (pogledano 30. 7. 2022.).

[35] *Ikone*. adresa: `https://www.freepik.com/` (pogledano 30. 7. 2022.).

Sažetak

U ovom radu opisana je implementacija web aplikacije Autoshare. Aplikacija je namjenena za korisnike koji žele svoja osobna vozila dati u najam, ili za korisnike koji imaju želju iznajmiti neko vozilo, po povoljnijim cijenama u odnosu na velike kompanije za najam automobila. Glavne funkcionalnosti koje nudi su izrada korisničkog računa, dodavanje vozila, najam i ostavljanje recenzije. Uz to, nudi korisnicima vrijedne informacije za vrijeme trajanja samog putovanja koje uključuju cijene goriva i vremensku prognozu.

Autoshare je REST-ful aplikacija, a glavne tehnologije korištene za izradu su Laravel radni okvir te React.js knjižnica. Ostale korištene tehnologije uključuju PosgreSQL kao DBMS, Bootstrap za dizajn korisničkog sučelja, JSON Web Token za autentifikaciju. Korišteni su razni javni API-ji za dohvaćanje podataka o cijenama goriva, vremenu i proizvođačima automobila.

Ključne riječi **— Web aplikacija, Laravel, React, Bootstrap, najam vozila**

Abstract

This paper describes the implementation of a web application called Autoshare. The application is meant for users who want to rent out their personal vehicles, or users who want to rent someone else's vehicle for a price more affordable than in big car rental companies. The main functionalities it offers are creation of user accounts, adding vehicles, renting, and leaving reviews. Additionally, it provides users with helpful information during their trip, including gas prices and weather forecasts.

Autoshare is a REST-ful application and the main technologies used for development are the Laravel framework and React.js library. Other used technologies include PostgreSQL as a DBMS, Bootstrap for UI design, and JSON Web Tokens for authentication. Many public APIs were used for fetching data related to fuel prices, weather, and car manufacturers.

Keywords **— Web application, Laravel, React, Bootstrap, vehicle rental**