Tehnologije pametnih cesta i inovativne prometne infrastrukture

Pirić, Filip

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

Sveučilišni diplomski studij

TEHNOLOGIJE PAMETNIH CESTA I INOVATIVNE PROMETNE INFRASTRUKTURE

Diplomski rad

Filip Pirić

Osijek, 2021.

Sadržaj

1. UVOD

Napretkom tehnologija napredovale su sve grane gospodarstva i industrije pa tako i prometni sustav. Prometni sustav je jedna od najbitnijih stavki za opstanak industrije i gospodarstva, a isto tako je važan i u privatnom smislu života. Kako bi se poboljšala učinkovitost, kvaliteta i sigurnost putovanja potrebno je osmisliti način kako implementirati komunikacijsko informacijsku tehnologiju u sustav prometa. Na taj način, putem inovativnih tehnologija i prometnih infrastruktura doprinijeti razvoju inteligentnih transportnih sustava za koji se smatra da će promijeniti cjelokupnu sliku današnjeg prometa.

Naime, veliku ulogu u napretku inteligentnih transportnih sustava ima internet koji omogućava vrlo brzo slanje podataka te omogućavanje tih podataka korisnicima u svakom trenutku te u stvarnom vremenu. Kako bi se povećala brzina prosljeđivanja i obrade podataka uvodi se 5G mreža, zbog koje su omogućene razne informacijsko komunikacijske tehnologije. Uz pomoć internet i 5G mreže podaci se prikupljaju u stvarnom vremenu *(engl. real time)* te se komunikacijskim tehnologijama prosljeđuju do baze podataka gdje se uz pomoć umjetne inteligencije *(engl. artificial intelligence)* i strojnog učenja obrađuju podaci te se iskorištavaju na najbolji način u svrhu poboljšanja kvalitete i brzine inteligentnih transportnih sustava.

Uz pomoć naprednih tehnologija cilj je osigurati sigurnost u prometu tako da se prometne nesreće i nezgode smanje na minimum. Također, je cilj smanjenje prometnih gužvi te reduciranje zagađivanja okoliša i povećanje sigurnosti pješaka i vozača.

1.1. Cilj rada

U radu je potrebno istražiti i opisati nove i inovativne tehnologije pametnih cesta i prometne infrastrukture koje doprinose razvoju inteligentnih transportnih sustava. Potrebno je detaljno analizirati doprinos tih tehnologija u svrhu povećanja učinkovitosti upravljanja prometom, smanjivanja prometnih gužvi i okolišnog zagađenja te poboljšanja sigurnosti pješaka i vozila. U praktičnom dijelu rada potrebno je pomoću prometnih API-ja prikazati podatke s prometnica u web aplikaciji uz mogućnost podešavanja postavki i prikaza statistike.

2. PREGLED PODRUČJA TEME

Tema ovog rada se može poistovjetiti s brojnim aplikacijama koje su dostupne korisnicima diljem svijeta. Slične aplikacije su Here, Google Maps, MapFactor, Maps.me, AMZS i brojne druge koje na bilo koji prenose informacije o prometu do korisnika. Here je navigacijska aplikacija koja sadrži sve informacije o prometu. Od najbrže rute, pa sve do informacija o gužvama na cestama, izvještavanju o prometnim nesrećama i slično. Google Maps je aplikacija koja se najbolje može poistovjetiti s aplikacijom ovoga rada. Korisnik ima na raspolaganju kartu na kojoj može označiti rutu koju želi prijeći te mu aplikacija vrati povratne informacije o zadanoj ruti, kao i kod aplikacije ovoga rada. Razlika je u tome što Google Maps ne pokazuje informacije o zadanoj ruti u prošlosti, već prikazuje samo trenutno stanje na cestama, dok ova aplikacija prikazuje vraća korisniku i informacije o događanjima na zadanoj ruti u proteklih godinu dana. Također, postoje aplikacije koje prikazuju informacije o radarima na pojedinim dionicama. Jedna takva aplikacija je AMZS. Ovo je aplikacija koja je osnovana u Sloveniji, točnije u Ljubljani. Osim što služi kao navigacijska platforma, ova aplikacija korisniku prikazuje i na kojim lokacijama se nalaze radari za snimanje brzine vožnje. Također, predstavlja najvećeg prometnog zastupnika u Sloveniji. Naime, aplikacije sličnih priroda su pomogle u napretku cijelog prometnog sustava. Korisnici imaju jasnu sliku o prometu te dionicama kojima se kreću te imaju pregled cjelokupnog prometa te stanja na cestama. Još jedna aplikacija slične prirode koja je vrlo popularna u svijetu je MapFactor Navigator. Naime, ovo je besplatna navigacijska aplikacija koja pruža korisniku mogućnost verbalne komunikacije sa samom aplikacijom. Ovo je besplatna GPS platforma ate se koristi u više od dvjesto zemalja. Projekt ovoga rada se može usporediti s navedenim aplikacijama. Osim što pruža korisniku mogućnost pregleda ruta te informacije o prometu na odabranim rutama, također pruža mogućnost mini bloga gdje korisnici mogu ostavljati svoje komentare te na taj način komunicirati s ostalim sudionicima u prometu te si na taj način mogu podijeliti vlastita iskustva. Na taj način se omogućava interakcija svih korisnika što omogućava daljnji napredak i razvoj aplikacije.

3. PAMETNE CESTE

Napredak tehnologije bilježi se svakodnevno, pa tako i utječe na svakodnevni život svih ljudi. Dvadeset i prvo stoljeće obilježila je tehnologija koja je značajno utjecala na novo takozvano moderno doba. Naime, krajem devedesetih godina (1999. godine) Japanska firma NTT DoCoMo izdala je prvi pametni telefon nakon čega su i ostali tehnološki giganti (kao što su Apple, Microsoft, Samsung itd.) krenuli sa izbacivanjem pametnih telefona. Svijet u tom trenutku postaje drugačiji te tehnološki svijet dobiva novi značaj. Primjena pametnih telefona postaje svakodnevnica te počinju ih koristiti apsolutno svi. Kako se tehnologija razvijala sve jači hardveri su uspijevali izgledati sve manje pa je tako mobitel koji je jači od nekadašnjih računala bio značajno manji od računala. Dakle, nano tehnologija dobiva na značaju te postaje sastavni dio svakog hardvera. Naime, da bi se ostvarili ciljevi koji zahtijevaju vrlo kompleksne tehnologije, potrebno je implementirati dostojan oblik tehnološko komunikacijskog kanala da bi se svi podaci mogli prenijeti do nekakvog centra podataka odnosno do nekakve baze podataka. Tu u priču upada internet gdje se više neće informacije striktno čuvati u samom hardveru telefona već će se moći sačuvati u takozvanom oblaku *(engl. cloud)*. Oblak je paradigma informatičke tehnologije koja pruža infrastrukturu za pohranu podataka putem interneta. Dakle, podaci se pohrane na "cloud" gdje su na sigurnom mjestu te su dostupni korisniku u bilo kojem trenutku. Pametni telefoni su bili samo početak svega "pametnog" što tek dolazi. Tako u tehnološku priču upadaju, među ostalim i pametne ceste. Ovaj princip gradnji cesta će samo biti još jedno tehnološko otkriće koje će obilježiti ovo stoljeće. Naime, cilj pametnih cesta je izgradnja cesta koje će biti sigurnije, intuitivnije, ekološki održivije te jeftinije. Tako su se javljale razne ideje kako osmisliti pametne ceste te kako ih iskoristiti u najboljem smislu. Jedna od ideja je "Glow in the dark" gdje će cesta svijetliti i jasno pokazivati vozaču vozne trake i kraj ceste. Također, će obavještavati vozača o vremenskim neprilikama, gužvama te općenito će prikazivati stvarnu sliku prometa na dionici kojom se vozač kreće. Razvojem ovakvih cesta povećava se ekonomska i ekološka održivost pri čemu se diže razina sigurnosti samog vozača. Pametnom cestom se može smatrati svaka cesta čija je infrastruktura povezana s naprednim komunikacijsko mrežnim tehnologijama. Drugim riječima, cesta koja komunicira sa znakovima, vozilom koji se u zadanom trenutku nalazi na pametnoj cesti te svim ostalim objektima u prometu, može se nazvati pametna cesta. Ne samo da će ovakve ceste doprinijeti sigurnosti u prometu, ekološkoj održivosti te olakšavanju vozačima već će ona biti

potpora za ostvarivanje drugih tehnoloških zahtjeva. Pa će tako olakšati uvođenje autonomne vožnje u svakodnevnicu. Naime, da bi se ostvarila autonomna razina 5[1], potrebno je osigurati sigurnu komunikaciju cesta, znakova i ostalih objekata sa samim vozilom. Kada vozilo ima sve informacije, u stvarnom vremenu, ono se može autonomno kretati bez sudjelovanja vozača prilikom vožnje.

3.1. Infrastruktura pametnih cesta

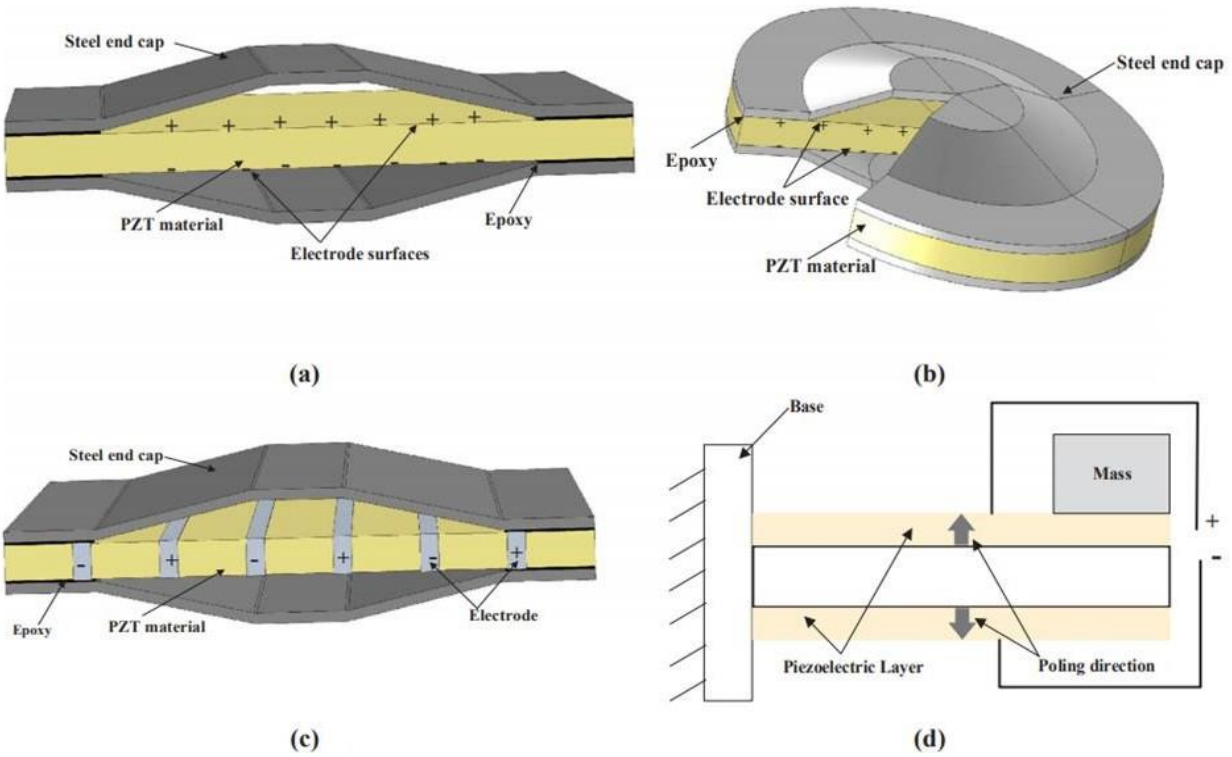
Tehnologija je kroz povijest na mnoge načine utjecala na rješavanje problema svih vrsta. U London je 1800.-tih godina sve se više ljudi počelo naseljavati u gradove [1]. Kako tada automobil još nije bio svakodnevnica kao danas, ljudi su za prijevozno sredstvo koristili konje. Gradovi su se počeli razvijati i širiti te je London svakim mjesecom brojao sve više stanovnika. Budući da su koristili konje kao prijevozno sredstvo u gradovima je bilo sve više konja. Ekvivalentno tome na cestama je bilo sve više konjskog izmeta, odnosno balege. Tako je 1894.godine u Londonu proglašena velika kriza konjske balege *(engl. The Great Horse Manure Crisis)*. Tadašnja kriza riješena je pojavom automobila. Temeljeno na navedenom primjeru današnje pametne ceste bi trebale predstavljati "auto" koji će riješiti problem svih nedostataka u dosadašnjoj infrastrukturi cesta. Naime, infrastruktura cesta mora zadovoljiti sve uvijete kako bi se pametna cesta mogla ostvariti te se smatra najbitnijom stavkom za sami početak planiranja izgradnje pametnih cesta. Prema [2], javljaju se razni problemi prometu kao što su prometne nesreće i nezgode, zagađivanje okoliša, nedostatak goriva odnosno nafte, cijena i razni drugi. Kako se kapaciteti gradova pune, promet postaje sve više zakrčen te se povećava vjerojatnost od prometnih nesreća i nezgoda. Svi navedeni problemi se mogu riješiti uvođenjem pametnih cesta. Da bi se pametna cesta mogla uvesti mora se osigurati pravilna infrastruktura cesta koja će podržavati sve napredne tehnologije te pružati podršku inteligentnim vozilima.

Kako bi se infrastruktura mogla mijenjati potrebno je krenuti od same ceste, koja je glavna stavka prometne infrastrukture. Cilj je iskoristiti ceste za prikupljanje energije.

[1] Razina autonomne vožnje gdje je omogućena vožnja bez vozača, odnosno potpuno autonomna vožnja.

Prvo rješenje koje može ostvariti ovaj cilj su solarni paneli i konektori. Energija koja bi se skupila putem solarnih panela koristila bi se za napajanje znakova, voznih traka i drugo, a višak energije koji bi ostao neiskorišten slao bi se natrag u distribucijsku mrežu. Ovakav sustav koristi foto naponske module koji bi bili postavljeni na cestu. Foto naponski moduli mogu biti monokristalni, polikristalni i amorfni. Monokristalni i polikristalni moduli imaju dug životni vijek te je njihova iskoristivost iznad 15%, dok amorfni moduli pokrivaju veću površinu, ali im je zato iskoristivost mala (do 6%). Ovakav sustav predstavlja najbolje rješenje kada je u pitanju samo prikupljanje energije. U drugu nedostaci su proklizivanje na cestama, način upravljanja, cijena i drugo.

Drugo rješenje je korištenje piezoelektričnih elemenata. Naime, cesta je građena od tvrdog materijala, odnosno betona koji je predviđen da podnese ogromne količine tereta. Kako bi se riješio problem sa samim cestama da ne budu samo podloga po kojoj se kreće, ideja je iskoristiti težinu i kretanje vozila kako bi se skupila mehanička energija te pretvorila u električnu koja bi onda napajala vozila. Ovaj efekt se može postići putem piezoelektričnih elemenata. "*Piezoelektrični efekt predstavlja promjenu električnog otpora materijala (npr. poluvodiča ili metala) pri mehaničkom naprezanju.* [3]" Piezoelektrični uređaji pod mehaničkim pritiskom stvaraju električni naboj. Na taj način se može iskoristiti težina vozila koja vrši pritisak na cesti kako bi se generirao električni naboj. Navedeni uređaji se mogu podijeliti u dvije skupine, a to su senzori i aktuatori. Također se mogu kombinirati senzori i aktuatori na način na senzori detektiraju ultrazvučni val, dok aktuatori stvaraju ultrazvučni val. Dakle, piezoelektrični uređaji bi se ugrađivali ispod površine asfalta na dubini od 5cm (Slika 2.1.) te bi na taj način bili dovoljno blizu površine kako bi se mehanički pritisak vozila mogao pretvoriti u električni naboj. Naime, postoji više mogućnosti kako bi se mogli ugraditi piezoelektrični materijali (PZT). Prva opcija je tradicionalni most gdje bi na vrhove površina PZT materijala bile postavljene elektrode u horizontalnom pravcu te bi iznad bila posebna vrsta čelika (*engl. steel end cap*) koji bi upijao pritisak i prenosio ga dalje na pzt materijale. Druga opcija je slojeviti most čija je struktura dosta slična kao kod tradicionalnog mosta jedina razlika je u tome što bi kod slojevitog mosta elektrode bile postavljene vertikalno po horizontalnoj osi PZT materijala. Još neke opcije su činela i greda na nosaču. Nedostatak PZT materijala je cijena i potrošivost materijala jer često raskopavanje ceste da bi se obnovili PZT materijali ne predstavljaju najbolje rješenje.



Slika 2.1. Pretvarači na osnovi mehaničkog stresa: a) tradicionalni most; b) činela; c) slojeviti most; d) greda na nosaču [5]

Naime, postavlja se pitanje, postoji li mogućnost napajanja pametnih cesta putem samoodržive cestovne tehnologije koja će izbjeći potrebu za obnavljanjem materijala i samih cesta. Za tu svrhu postoji rješenje putem geotermalne energije. Geotermalna energija je toplinska energija unutar Zemlje. Drugim riječima, u dubini Zemlje se nalazi geotermalni medij[2] koji probija kroz Zemlju te izbija na površinu. Taj medij se može iskoristiti u izvornom obliku ili za iskorištavanje medija za pretvorbu u neki drugi oblik energije na primjer, električna energija, toplinska i slično. Budući da je geotermalna energija neiscrpna te da su njeni izvori neograničeni, bila bi dobra podloga za napajanje pametnih cesta. Dakle, ispod površine ceste bi se postavile cijevi kroz koje bi prolazilo geotermalni ne smrzavajući medij. Pumpe, koje bi se postavile ispod površine ceste, bi uvlačile vruću tekućinu koju bi dalje iskorištavali za pretvorbu energije, a hladna voda bi se vraćala nazad u zemlju. Uz pomoć solarnih kolektora i tehnologije apsorbiranja geotermalne energije cesta bi za

[2] Fluid odnosno tekućina koja izbija iz Zemljine površine u obliku vodene pare

vrijeme vrućih dana prikupljala topline koja bi se nakupila na površini asfalta te istu tu energiju putem generatora pretvarala u električnu energiju. Ovim putem bi se znatno smanjilo isparavanje asfalta u atmosferu, čime se zadovoljavaju ekološki uvjeti. Na ovaj način bi se izbjeglo zamrzavanje cesta zbog geotermalne topline koja bi preko zime zagrijavala cestu. Financijski bi ovaj način napajanja ceste također bio jeftiniji jer bi se riješili problema čišćenja snijega, leda te ostalih neprilika koje se mogu naći na cesti. Nedostaci navedenih načina pametne cestovne infrastrukture je održavanje jer je pitanje koliko bi se često cesta morala popravljati te koliko bi popravak koštao.

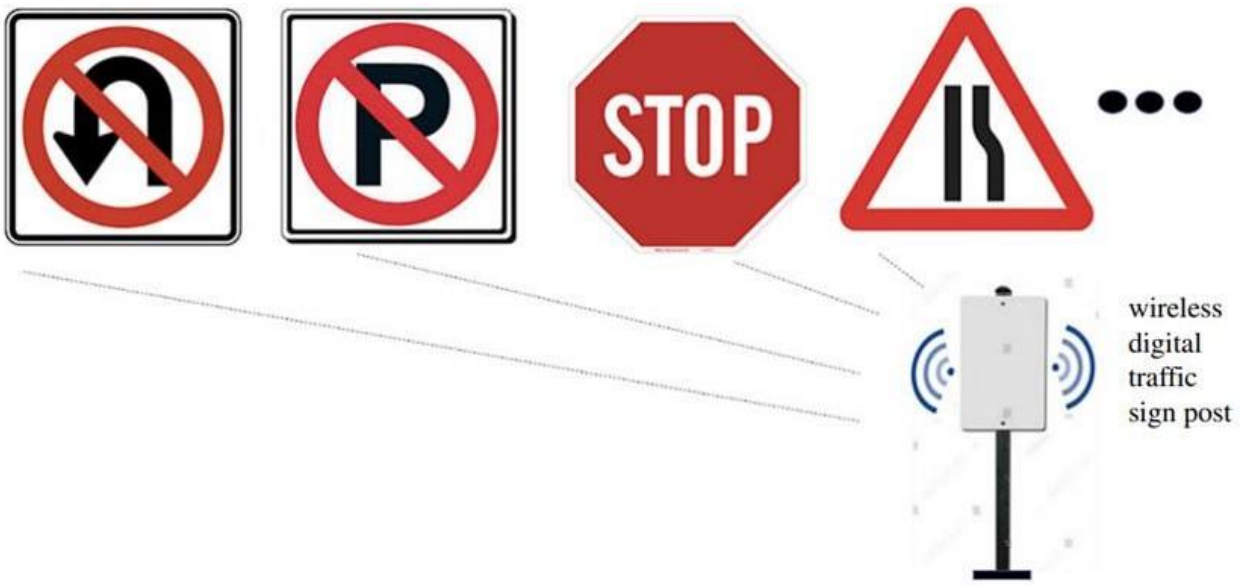
3.2. Pametni znakovi

Prometni znakovi su se uveli u promet prije stotinjak godina kada je Njemački inženjer 1885. godine predložio patent prometnih znakova [5]. Znakovi su dizajnirani na način da sam oblik znaka govori o razini opasnosti koju znak nosi. Kako bi se povećala sigurnost vožnje i kako bi autonomna vožnja bila ostvariva uvodi se ideja o pametnim prometnim znakovima. Naime, da se omogući potpuna autonomna vožnja razine pet, vozila moraju komunicirati sa svim raspoloživim objektima u prometu, a sa onima s kojima ne mogu komunicirati moraju ih prepoznati te reagirati na ispravan način kako bi se smanjila mogućnost prometnih nesreća i nezgoda. Dakle, mora se zadovoljiti asil[3] razina. Ovakvi znakovi bi imali u sebi ugrađeni sustav koji bi se sastojao od radara, lidara, senzora, mogućnost bežične komunikacije putem 5G mreže i slično. Putem radara i lidara, identificirali bi objekte koji im dolaze u susret. Prepoznali bi o kakvom objektu se radi te bi ga upozoravali o mogućim prijetnjama na tim dionicama ceste gdje se pametni znakovi nalaze. Putem 5G mreže i ostalih tehnologija, koje će biti navedene u nastavku rada, slali bi informacije do vozila. Vozilo bi primilo informacije, obradilo ih te bi sustav koji upravlja vozilom točno znao kakvu važnost ima na cesti u trenutnom vremenu. Razmatranja dizajna protokola uključuju pravovremenost, cjelovitost podataka, provjeru autentičnosti i razmjenu podataka [5]. Pametni znakovi će također upozoravati sudionike u prometu putem svijetla te zvukovnih signala. Naime, znakovi će također imati svoj spremnik energije, koji će napajati sve električne komponente znak.

[3] Asil (Automotive Safety Integrity Level) – stupanje integriteta u automobilskoj sigurnosti.

Tako će znak preko noći svijetliti te će biti uočljiv svim sudionicima u prometu. Također će svaki znak imati boju svijetla takvu da upozori vozača, koji kriterij sigurnosti predstavlja znak. Ovaj način uspostave znakova će također olakšati održavanje znakova jer ukoliko se na određenoj dionici želi promijeniti znak (zbog promijene važnosti

ceste ili zbog nekog drugog razloga), to će se moći napraviti na samome znaku jer će on biti programibilan (Slika 2.2.).



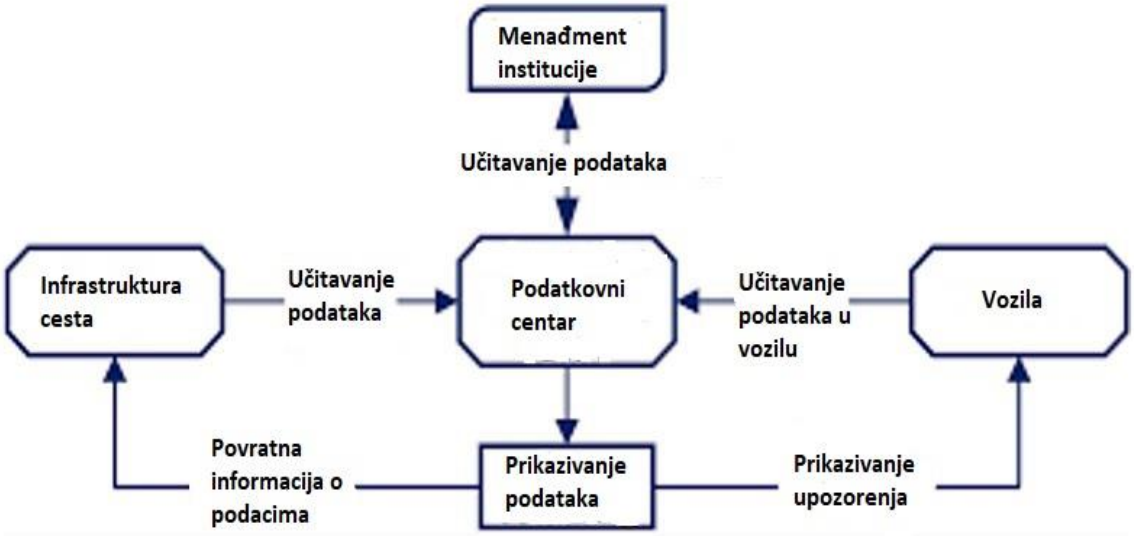
Slika 2.2. Digitalni programibilni prometni znak [5]

Takvi znakovi se nazivaju promjenjivi prometni znakovi (*engl. variable traffic signs*). Ovakvi znakovi se koriste na mjestima gdje postoji potreba za mijenjanje znakova, radi promjenjive prometne situacije. Prednost promjenjivih prometnih znakova leži u dobroj vidljivosti te u mogućnosti promjene trenutnog simbola na znaku. Ovakav znak se napaja iz kućišta koji sadrži električnu energiju potrebnu za napajanje LED lampica na samome znaku. Dakle, na znaku se prostire jako pun LED lampica na način da zauzmu punu površinu znaka. Kad trebaju zasvijetliti, zasvijetlit će samo one koje u tom trenutku formiraju određeni prikaz na znaku. Na primjer, ako se radi o znaku stop. Zasvijetlit će samo one lampice koje se protežu duž poteza koji formira znak stop na znaku. Trenutna najbolja solucija za napajanje ovih znakova leži u solarnom napajanju

istih. Ovakvi znakovi imaju veliku svjetlinu što zauzima pažnju vozaču te povećava sigurnost u prometu. Također, štednja energije ovakvim načinom napajanja je povećana, te je ekološka svijest na prihvatljivoj razini, što ujedno čini vrlo dobru podlogu da se s tim segmentom približi riješenu pametne ceste. Solarni prometni znakovi funkcioniraju na sličan način kao i u prethodnome primjeru. Postavljaju se svjetleće diode koji imaju duži vijek trajanja i manju potrošnju električne energije. Zahvaljujući solarnoj energiji i izbacivanju kablova i žica iz ovog sustava, ovakvi prometni znakovi zahtijevaju jako malo održavanja, budući da su otporni na teške vremenske uvijete kao što su jaki vjetrovi, tuča i ostalo.

3.3. Funkcije i komponente pametnih cesta

Najbitnija stavka, koju pametne ceste moraju ispuniti je prikupljanje podataka te vrlo dobra obrada istih te njihovo iskorištavanje. Ovo ujedno predstavlja i glavnu funkciju pametnih cesta. Korištenjem brojnih tehnologijskih uređaja potrebno je odrediti sliku cijele infrastrukture te njeno stanje. Na taj način se prikuplja dovoljna količina podataka koja se procesira tako da ostale tehnologije imaju koristi od tih podataka. Kako je svaki detalj reguliran nekom vrstom tehnologije, ovakva vrsta ceste mora sadržavati tehnologiju za održavanje, popravljanje, kontrolu temperature i drugo. Kako je već navedeno sva energija, koju kontrolira tehnologija je dobivena iz obnovljivih sustava energije. Najbitnija funkcija, koju pametna cesta mora zadovoljiti je informacijska povezanost. Ona povezuje infrastrukturu ceste sa vozilima na cesti putem sustava za obradu podataka. Naime, pametna cesta zajedno sa vozilom čini izvor podataka koji se šalje, bežičnim putem do centra za obradu podataka. Kako se šalje bežičnim putem, nova generacija mreže takozvana peta generacija odnosno 5G mreža, je doprinijela razvitku pametne ceste jer se podaci putem nje šalju puno većom brzinom. Centar za obradu podataka prima sve podatke te nakon njihove obrade šalje ih dalje distribucijskoj komponenti, koja je zadužena za distribuciju tih podataka. Prikaz toka podataka se može vidjeti na slici 2.3. Naime podatkovni centar je baza gdje se nalaze svi najbitniji podaci. Kako bi se izbjegao podatkovni centar, uvodi se blockchain tehnologija koja omogućuje decentralizaciju te se smatra kao jednom od najsigurnijih tehnologija jer da bi se promijenio jedan blok, moraju se promijeniti svi blokovi, što je nemoguće.



Slika 2.3. Prikaz toka podataka putem informacijske povezanosti pametne ceste [6]

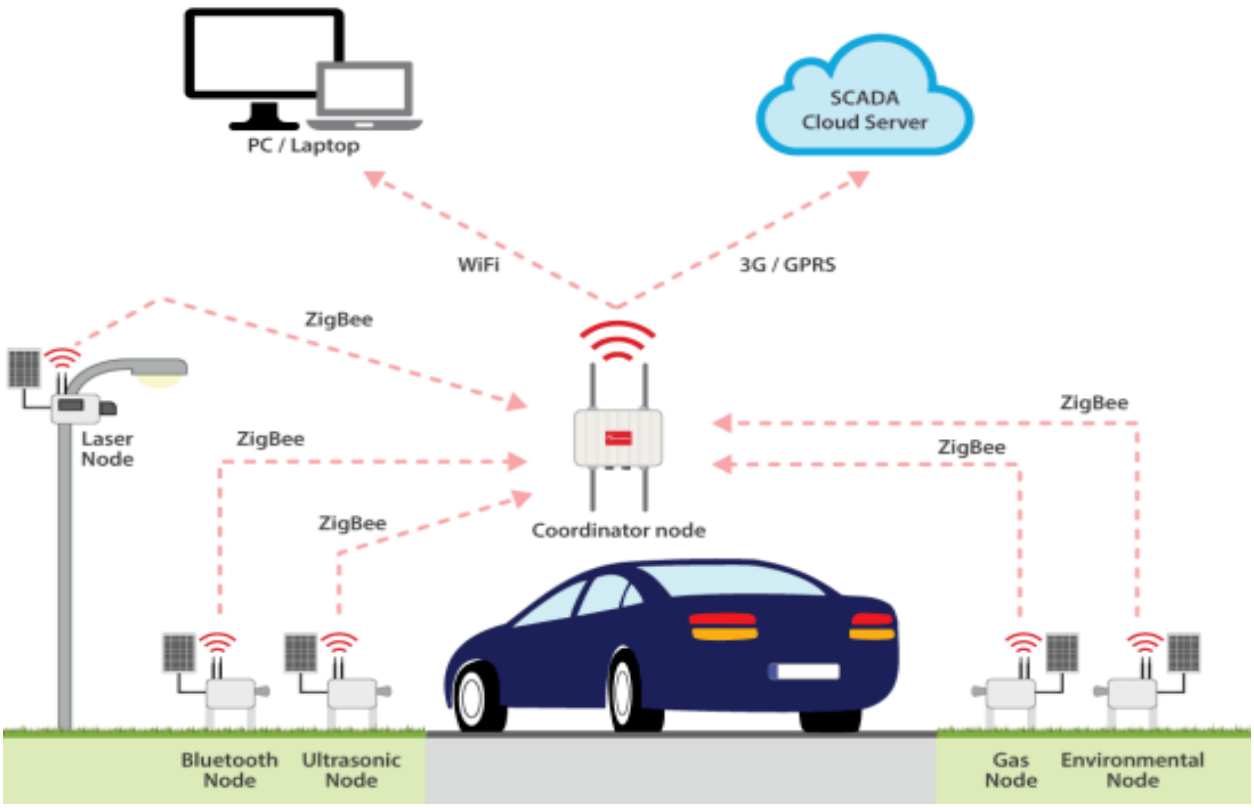
Sedamdeset posto slučajeva, prometnih nesreća je isključivo ljudski faktor, dok je u više od devedeset posto slučajeva uključen ljudski faktor. Ovaj podatak govori kako je zapravo najveća opasnost u prometu vozač. Primjenom pametne ceste se kontrolira ponašanje vozača što ujedno predstavlja i jednu od funkcija pametne ceste [6]. Pomoću tehnologija, upravljat će se cjelokupnim prometom. Ova funkcija će pridonijeti praćenju ceste u svakom trenutku. Vozilo će moći komunicirati sa cestom i sa ostalim vozilima te će reakcija prilikom nesreće biti znatno brža. Osim navedenim funkcija pametnih cesta, također će se povećati postotak prikupljanja električne energije i smanjenje zagađenja okoliša tako što će se reducirati potrošnja ispušnih plinova.

Kako bi se sve navedene funkcije obavljale, uvode se komponente pametnih cesta koje reguliraju, stvaraju i obavljaju sve funkcije. Najbitnija komponenta pametne ceste je komunikacijski sustav. Ovaj sustav je prva poveznica između ceste i vozila. Također je povezan je putem mreže s infrastrukturama ostalih cesta radi prikupljanja što veće količine podataka. Komunikacijski sustav se temelji na žičnoj i bežičnoj komunikaciji. Napretkom tehnologije bežični način preuzima glavnu ulogu jer se princip rada pametne ceste temelji upravo na bežičnom načinu slanja podataka.

Prijenos podataka predstavlja vrlo bitnu komponentu jer se pomoću podataka koji stignu do centra za obradu podataka analizira promet, lokacija, brzina i slično. Lokacija se utvrdi pomoću GPS[4] tehnologije, dok udaljenost od ostalih vozila, ruba ceste i slično, se određuje pomoću tehnologije senzora. Pod ovom komponentom, mogu se ubrojiti radar i lidar tehnologije, koje služe za detekciju objekata na sceni.

Kako bi sve tehnologije pametne ceste funkcionirale potrebno ih je spojiti, ali budući da je za sve potrebne podatke potrebno puno memorije dolazi do problema gdje skladištiti sve podatke. Za ovu svrhu potreban je jak hardver koji će zauzimati veliku količinu fizičkog prostora. U ovom slučaju do svog izražaja dolazi komponenta pametne ceste koja se zove internet stvari *(engl. Internet of Things)*. Internet stvari je tehnologija koja povezuje više različitih uređaja te razmjenu podataka između tih uređaja putem internet [7]. Ova tehnologija predstavlja vrlo bitnu komponentu jer čini cijeli sustav sigurnijim i bržim. Dakle, svi podaci koji se razmjenjuju među uređajima putem interneta, razmjenjuju se pomoću tehnologije internet stvari kao što se može vidjeti na slici 2.4. Osim navedenih komponente, sustav pametne ceste koristi i brojne napredne komponente koje su zapravo spoj više tehnologija. Primjer napredne komponente je sustav informiranja i praćenja vozača. Ova komponenta koristi veliki broj tehnologija kao što su senzori, kamere, strojno učenje i drugo. Kako je već navedeno, vozač predstavlja najslabiju kameru u vožnji te se iz tog razloga koriste tehnologije koje će cijelo vrijeme pratiti vozača i upozoravati ga na potencijalne opasnosti na cesti. Također, senzori mjere razinu stresa vozača te koncentraciju vozača na način da senzor prati jel vozač ima ruke na volanu. Kao vrlo bitnu naprednu komponentu, treba izdvojiti sigurnosno-kritični sustav vozila. Ovaj sustav upozorava vozača na opasnosti, te u visoko kritičnim situacijama može preuzeti kontrolu nad vozilom. Na taj način se smanjuje opasnost u prometu te se lakše zadovoljavaju asil razine. Također, ukoliko senzor prepozna da vozač ne drži ruke na volanu, sustav ga upozori te nakon par upozorenja sigurnosno-kritični sustav sam zaustavlja automobil na prvom mogućem stajalištu. Ovo je samo jedan primjer od bojnih. Kada se uspostavi potpuna autonomna vožnja, sigurnosno-kritični sustav će vladati automobilom u svim kritičnim situacijama te će se na taj način smanjiti broj nesreća.

[4] Engl. Global Positioning System – radionavigacijski sustav za određivanje položaja na Zemlji.



Slika 2.4. Prikaz tehnologije internet stvari [8]

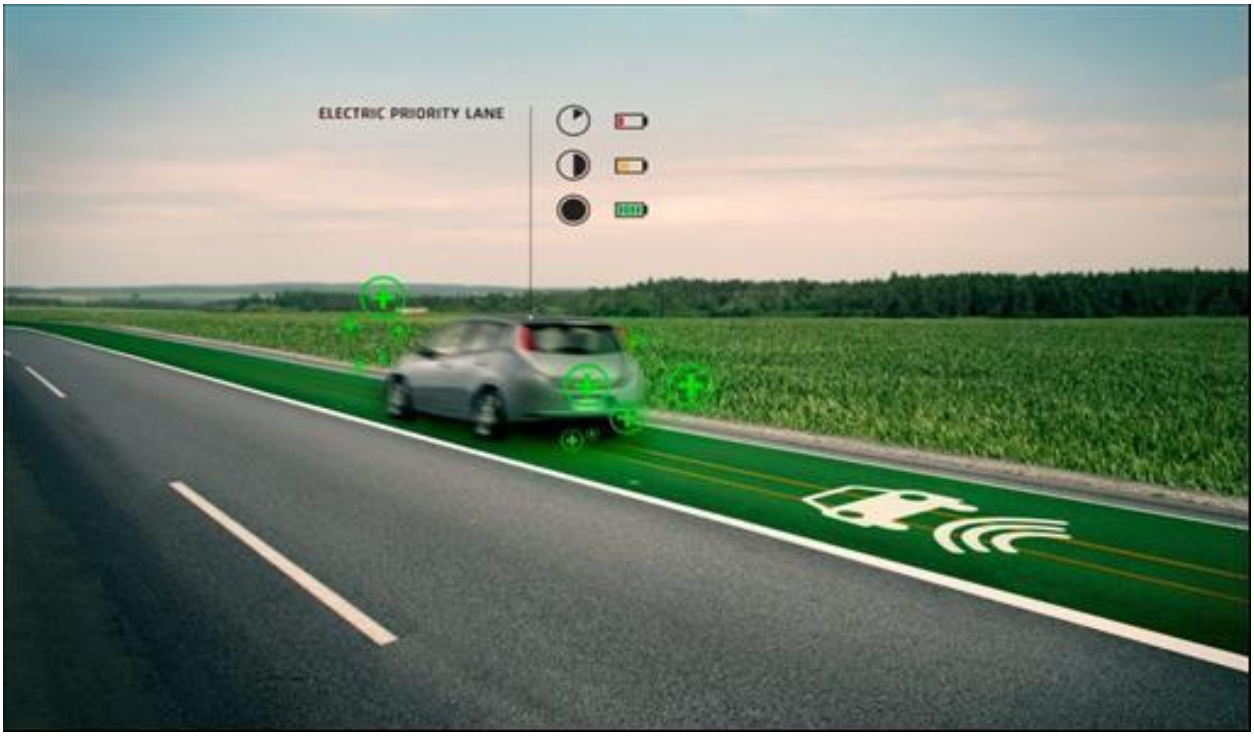
Korištenjem toliko tehnologija, cesta postaje digitalna te prikuplja ogromne količine podataka. Radi svih do sada navedenih tehnologija, omogućeno je slanje ogromne količine podataka. Na taj način se na cesti u svakom trenutku može pratiti brzina, položaj sudionika u prometu, vrijeme i drugo. Također, će policijski službenici imati uvid o podacima u prometu te će moći provoditi dodatne sigurnosne mjere na dionicama visokog rizika. Podaci koji opisuju trenutno stanje na cestama će biti otvoreni tako da svi mogu vidjeti stanje na cestama. Ovi podaci se nazivaju otvoreni podaci te će doprinijeti smanjenju gužvi na cestama. Pomoću otvorenih podataka, ostali pojedinci ili ustanove će moći unaprjeđivati svoje tehnologije te utjecati na napredak pametnih cesta. Budući da će otvoreni podaci biti dostupni i samom korisniku, korisnik će također moći utjecati na poboljšanje samog sustava ostavljajući svoj komentar o trenutnim tehnologijama.

4. TEHNOLOGIJE PAMETNIH CESTA

Današnji cestovni promet se značajno razlikuje od prije. Automobili i njihove tehnologije su drastično napredovali dok je infrastruktura ceste ostala ista. Pomoću tehnologija, ceste će se približiti automobilima u tehnološkom smislu. Naime, ceste nisu namijenjene samo za privatna vozila, već i za teška transportna vozila. Transportna vozila će imati mogućnost vaganja tereta na cesti. Na pojedinim točkama autocesta, bit će postavljene pametne vage koje će biti ugrađene u cestu, tako da vozač ne mora gubiti vrijeme na vaganje terete. Vaganje će biti provedeno putem HS-WIM[5] i ugradbenih piezoelekrtičnih uređaja na način koji je već opisan u radu. Kada je vaganje završeno, podaci se odmah šalju putem interneta do centra za obradu podataka. Za komunikaciju među uređajima zaslužena je tehnologija, internet stvari. Kada podatkovni centar obradi zadani podatak, šalje ga dalje u mrežu gdje prosljeđuje do ovlaštenih organa vlasti koje provjeravaju jel težina ispravna te vraća potvrdu istim putem nazad. Ovaj način ubrzava cijeli postupak vaganja terete što olakšava vozaču jer ne gubi više toliko vremena. Također, će biti moguće puniti vozilo dok se putuje. Prometna traka na cesti, koja je označena zelenom bojom, bit će punjač za vozila. Pomoću piezoelektričnih uređaja, kako vozilo prođe pomoću svoje težine generirat će električnu energiju koju će pametna cesta iskoristiti za punjenje vozila. Budući da autonomna vožnja sve više napreduje, pa tako i električna vozila, vozila neće zahtijevati da se stane prilikom punjenja vozila, već će to raditi na samoj cesti dok se vozilo kreće (Slika 3.1.). Sve navedene beneficije koje nudi pametna cesta moguće je koristiti samo uz upotrebu raznih tehnologija. Neke od tehnologija koje koristi pametna cesta su:

- 5G mreža
- V2X tehnologija
- VANET
- Internet stvari
- Strojno učenje
- Umjetna inteligencija
- IEEE 802.11P, Standard za komunikaciju među vozilima

[5] HS-WIM (engl. High-Speed Weigh in Motion) – vaganje tereta u pokretu



Slika 3.1. Pametna samonapajajuća prometna traka [9]

4.1. 5G Mreža

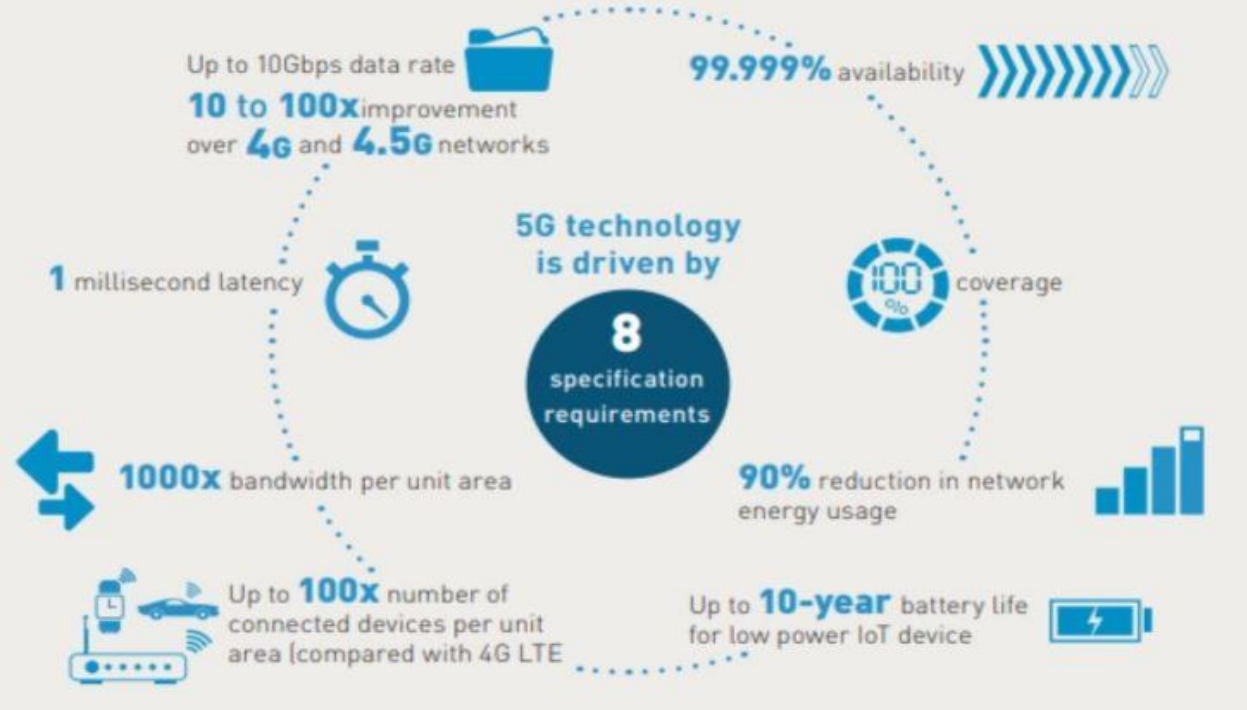
Prva generacija mobilnih mreža počela se stvarati 80-tih godina. Ta mreža je bila osnova za uspostavljanje svih ostalih generacija mreža. Na osnovu iskustava i učenja, mreže su napredovale iz generacije u generaciju, pružajući programerima bolju osnovu za dizajniranje poboljšane sigurnosti i tehnologije za signalizaciju, transport i ukupne performanse sustava. Početkom 2010. godine uvodi se 4G mreža poznata kao LTE *(engl. Long Term Evolution)*. Ova mreža se pokazala vrlo uspješnom, jer je drastično promijenila brzinu internet, što je utjecalo na razvoj brojnih drugih tehnologija. Bez 4G mreže ne bi bilo moguće imati aplikacije u automobilima jer bi brzina bila preslaba. Iako se 4G mreža pokazala jako dobrom, obzirom na njene performanse, i dalje nije bila dovoljno. Naime, zahtjevi multimedije su počele rasti zbog zahtjeva korisnika, pa tadašnja brzina prijenosa podataka nije bila dovoljna. Radi toga, dizajnirana je nova generacija mobilnih mreža, peta generacija odnosno 5G.

Peta generacija mobilnih mreža predstavlja novu generaciju mreža koja zadovoljava zahtjeve moderne mobilne telekomunikacije. U usporedbi s 4G mobilnom mrežom, 5G mreža pruža puno veću brzinu prijenosa podataka (čak do 10 puta) što predstavlja jedan od najbitnijih faktora u digitalizaciji prometnih cesta. Industrija progovara da je 5G mreža kombinacija novih rješenja koja trebaju biti standardizirana i postojećih sustava koji se temelje na 4G mreži. Brzina nije jedini fragment koji čini ovu mrežu toliko uspješnom. Osim brzine, pretpostavka je da će ova tehnologija pružiti potporu za ogromni IoT ekosustav u kojem mreže mogu zadovoljiti komunikaciju među milijardu međusobno povezanih uređaja uz optimizaciju troškova, brzina i latencija. Postoji osam specifikacija koje obilježavaju 5G mrežu [10], a to su:

- Poboljšana brzina u odnosu na 4G, od 10 do 100 puta (do 10Gbps)
- Latencija[6] od jedne milisekunde
- 1000x širina pojasa po jedinici površine
- Veći broj povezanih uređaja (do 100 puta)
- 99% veća dostupnost
- Stopostotna pokrivenost
- Smanjenje korištenja mrežne energije
- Dulji vijek baterija za IoT uređaje manje snage

Svih osam specifikacija se mogu vidjeti na slici 3.2. Osnovna ideja 5G mreže oslanja se na tehnologije radio frekvencijskih opsega koji su proširени na opsege od nekoliko GHz-a kako bi povezali podatke, aplikacije, ljude, transportne sustave pa tako i kompletne gradove. Funkcionirajući na takav način, osigurava nesmetani radi IoT-a, a i predstavlja osnovu u razvoju pametne mreže komunikacije. Važan cilj 5G standarda je pružiti interoperabilnost između mreža i uređaja, ponuditi energetski učinkovite i sigurne sustave velikog kapaciteta i značajno povećati brzinu prijenosa podataka s mnogo manje kašnjenja u vremenu odziva. Unatoč tome, peta generacija i dalje predstavlja skup ideja za visoko razvijeni sustav izvan 4G. Kao što je bio slučaj i s prethodnim generacijama.

[6] Latencija- kašnjenje



Slika 3.2. Specifikacije 5G mobilne mreže [10]

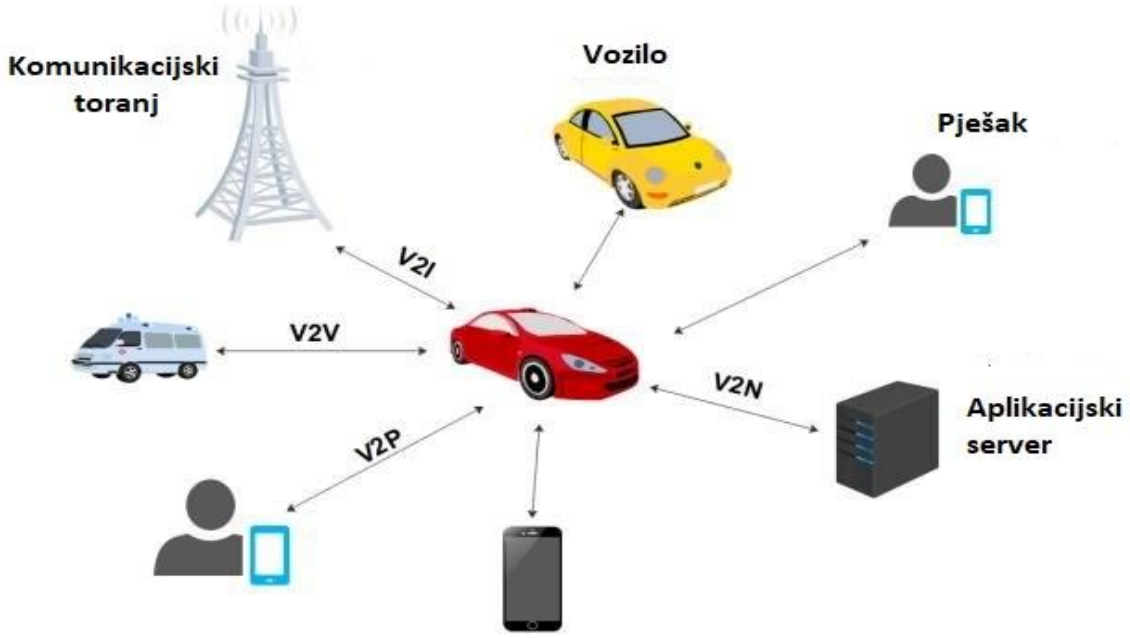
Najbitniji koncepti 5G mreže su:

- Data network name (DNN) – slična konfiguracija i princip rada kao kod APN-a *(engl. Access Point Name).*
- 5G Globally unique temporary identifier – GUTI *(engl. Globally Unique Temporary Identifier)* se koristi za zaštitu povjerljivih informacija.
- New Radio – sadrži dodatne mogućnosti u odnosu na LTE, a to su korištenje dva frekvencijska opsega, ispod 6GHz i od 24 do 100GHz.
- Subscription permanent identifier (SUPI) – svi pretplatnici unutar 5G mreže imaju jedinstveni globalni identifikator koji se zove SUPI.
- Unified Data Management (UDM) – upravlja ključevima, odgovoran je za rukovanje korisničkom identifikacijom i autorizacijom.

4.2. VANET

VANET mreža *(engl. Vehicular Ad Hoc Network)* predstavlja decentraliziranu mrežu koja je nastala od mobilne mreže MANET *(engl. Mobile Ad Hoc Network)*. Ad-hoc mreže imaju ogroman obujam što znači da pokriva veliku površinu lokalne bežične mreže *(engl. WLAN)*[7] i može imati ogroman broj korisnika odnosno vozila. Kao jedna od najvažnijih mogućnosti kod međusobnog komuniciranja vozila i vozila s infrastrukturom, jest kreiranje potpuno nove paradigme za sigurnosne aplikacije vozila. Čak i druge aplikacije koje nisu sigurnosne aplikacije, mogu uvelike poboljšati učinkovitost ceste i vozila. Također je bitno spomenuti nove zahtjeve stvorene od strane visokih brzina vozila i vrlo dinamičnih radnih okruženja. Dolaze i novi zahtjevi nužni zbog novih aplikacija životne sigurnosti, uključujući visoke stope isporuka paketa i niske latencije istih. Vrlo su velika očekivanja bila od privatnosti i sigurnosti zbog prihvaćanja kupaca i vladinog nadzora. I tada su vozila (kad je VANET počeo s komercijalizacijom) generirala i analizirala jako velike količine podataka, iako su podaci samostalni unutar jednog vozila. VANET je drastično podigao razinu svijesti vozila ili vozača. VANET komunikacija se može obaviti izravno između vozila kao „one-hop“ komunikacija ili vozila mogu ponovno prenositi poruke, tako osiguravajući „multi-hop“ komunikaciju (Slika 3.3). Kako bi se povećala robusnost komunikacije ili pokrivenost, moguće je bilo postaviti releje uz cestu. Infrastruktura uz cestu se također može koristiti za pristup internetu te se prema tome podaci i informacije o kontekstu mogu prikupljati, obrađivati i pohranjivati „negdje“, primjerice u nadolazećim Cloud infrastrukturama. Interes za automobilske mreže komunikacije je bio snažno motiviran bogatstvom aplikacije koje su se mogle omogućiti. Najveću korist najizravnijeg oblika komunikacije su imale sigurnosne aplikacije, odnosno aplikacije za sprečavanje nesreća. Kao drugo, prikupljanjem podataka o stanju prometa iz šireg područja se mogao poboljšati protok prometa, smanjiti gužve, smanjiti vrijeme putovanja, a samim time i količinu štetnih tvari koje automobil emitira tokom vožnje. Također, štetne tvari koje automobil emitira prilikom vožnje će se smanjiti povećanjem proizvodnje električnih automobila koji ne ispuštaju štetne ispušne plinove.

[7] WLAN - Wireless Local Area Network



Slika 3.3. Primjer multi-hop komunikacije [11]

Najbolje opisano i rečeno je načelo „Intelligent Transportation System World“ kongresa iz 2008: Uštedi vrijeme, spasi živote. Klase aplikacija „Sigurnost“ i „Učinkovitost“ se mogu koristiti za klasifikaciju aplikacija na temelju njihove primarne svrhe. Međutim, aspekti sigurnosti i učinkovitosti se ne mogu promatrati kao potpuno razdvojeni skup značajki. Očito sudar vozila može dovesti do gužve u prometu, poruka o nesreći se može smatrati kao sigurnosna poruka iz perspektive vozila u blizini. Istu poruku vide i udaljena vozila kao ulazni faktor za izračunavanje alternativne rute unutar primjene učinkovitosti prijevoza. Unatoč tome što je VANET konceptualno jednostavan, dizajn i implementacija je tehnički i ekonomski veoma zahtjevna. Ključni tehnički izazovi su uključivali sljedeća pitanja:

- Inherentne karakteristike radio kanala – VANET predstavlja scenarije s nepovoljnim karakteristikama za razvoj bežičnih komunikacija, tj. višestruki reflektirajući objekti koji mogu pogoršati snagu i kvalitetu primljenih signala, uz to zbog pokretljivosti okolnih predmeta i/ili samih pošiljatelja i primatelja, moraju se uzeti u obzir i efekti nestajanja, blijeđenja.

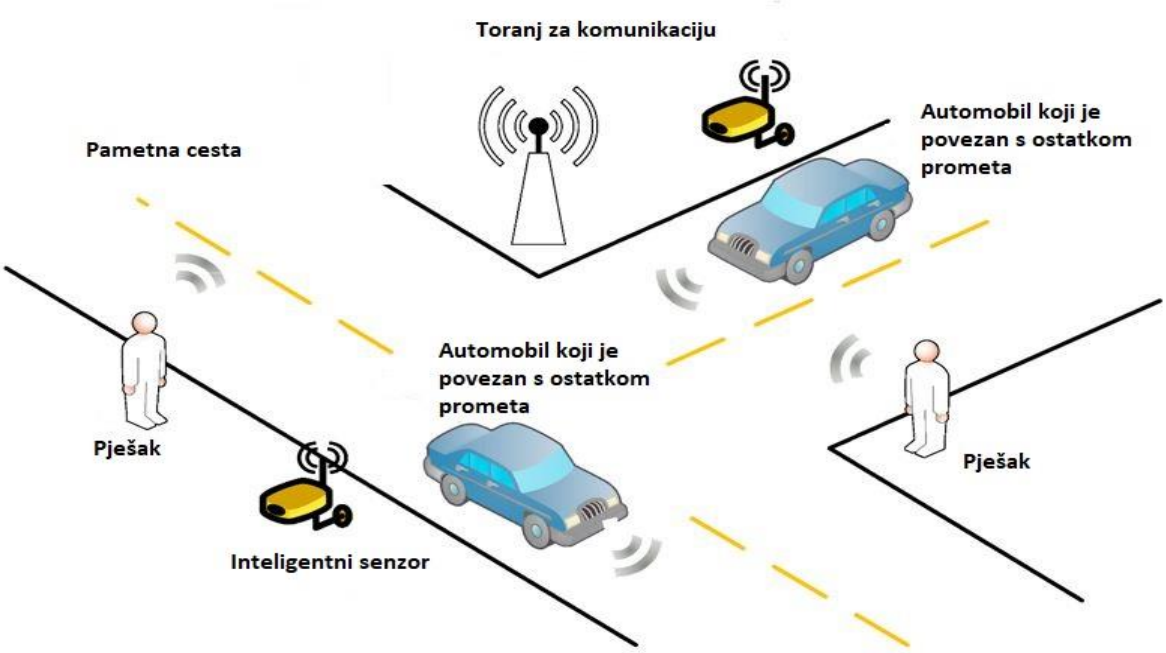
- Nedostatak internetskog centraliziranog tijela za upravljanje i koordinaciju – učinkovito korištenje dostupne propusnosti bežičnog kanala je težak zadatak u potpuno decentraliziranoj i samoorganiziranoj mreži. Nedostatak entiteta koji je sposoban sinkronizirati i upravljati prijenosnim događajima različitih čvorova može rezultati manje učinkovitom uporabom kanala i velikim brojem sudara paketa.
- Velika mobilnost, zahtjevi za skalabilnošću i široka raznolikost uvjeta okoliša – izazovi decentralizirane, samoorganizirane mreže su posebno naglašene velikom brzinom koju čvorovi u VANET-u mogu iskusiti. Njihova velika mobilnost predstavlja izazov većini iterativnih algoritama za optimizaciju usmjerenih ka boljoj upotrebi propusnosti kanala ili upotrebi unaprijed definirane rute za prosljeđivanje informacija.
- Potrebe i brige o sigurnosti i privatnosti – izazov je u balansiranju potrebe za sigurnošću i privatnošću. S jedne strane, prijemnici to žele osigurati da mogu vjerovati izvoru informacija, dok s druge strane dostupnost takvog povjerenja može biti u suprotnosti sa zahtjevima privatnosti pošiljatelja.
- Standardizacija nasuprot fleksibilnosti – bez ikakve sumnje, postoji potreba za standardiziranje komunikacije kako bi se omogućio rad VANET-u na raznim mrežama i raznim proizvođačima originalne opreme (OEM-a). Ipak, vjerojatno je da će OEM[8] proizvođači htjeti stvoriti neku varijaciju proizvoda sa svojim VANET karakteristikama što stvara određenu napetost među ciljevima. [11]

4.3. Tehnologija vozilo prema svemu

Vozilo prema svemu ili V2X predstavlja sustav komunikacije namijenjen za upotrebu u automobilskoj industriji. Podržava prijenos informacija sa pokretnog vozila na druge pokretne i nepokretne dijelove u prometu koji mogu utjecati na vozilo. Glavna svrha V2X tehnologije je unaprijediti sigurnost na cesti, povećati uštedu energije i efikasnost odvijanja prometa [12]. Uvođenjem V2X tehnologije bitno bi smanjio broj prometnih nesreća jer vozilo komunicira sa okolinom pa se na vrijeme može izbjeći opasna situacija. Kod V2X komunikacijskog

[8] OEM - Original equipment manufacturer

sustava informacija putuje od senzora sa automobila i drugih izvora kroz mrežu velike brzine i pouzdanosti što omogućuje robusnu komunikaciju s drugim automobilima, infrastrukturom kao što su semafori ili uređajem koji obavještava o privremenim radovima na cesti (Slika 3.4.). Također vozilo može komunicirati i sa pješacima preko mobilnih uređaja. Razmjenom informacija kao što su brzina vozila ili razmak od drugih sudionika u prometu, omogućava da uporabom neke tehnologije upozorimo vozača na opasnu situaciju, što smanjuje učestalost prometnih nesreća. Također moguće je povećati efikasnost protoka vozila u prometu na način da se obavještava sudionike u prometu o stanju na nadolazećem dijelu ceste, predlaganjem alternativne rute kako bi izbjegli gužve i zastoje na pojedinim dijelovima ceste kako bi brže stigli do odabranog cilja. Na parkinzima je moguće obilježiti slobodna parkirna mjesta kako bi se izbjeglo nepotrebno kruženje po parkiralištu u potrati za slobodnim parkirnim mjestom. Također postoji mogućnost automatskog plaćanja parkinga, cestarine i sličnog. Glavni cilj korištenja V2X komunikacijskih sustava je podizanje razine sigurnosti i izbjegavanje nesreća. U vozilu bi V2X sustavi mogli obavještavati vozača o vremenskim uvjetima, prometnim nesrećama, stanju na cestama, opasnim radnjama vozača blizu njih itd. Kod autonomnog vozila V2X daje dodatne informacije postojećem navigacijskom sustavu. V2X koristi bežičnu vezu kratkog dometa, koja mora biti otporna na smetnje, kako bi komunicirao sa kompatibilnim sustavima [13]. Glavni problemi u postojećim komunikacijskim standardima su nedostatak spektra, niske latencije i visoko pouzdanih prijenosa. Postojeći standard je pokazao lošu skalabilnost i nedostatak zajamčenog pružanja usluge u velikim mrežama. Dok su već brojna istraživanja pokrenuta na postojećoj LTE mreži, 5G pokazuje jako velik potencijal u V2X komunikaciji te stoga motivira daljnja istraživanja s novom mrežom. Pošto u praksi postoji jako velik broj pristupnih tehnologija i još veći broj primjena, nije moguće napraviti jedinstvenu tehnologiju koja rješava sve probleme i zahtjeve. Stoga se ne cilja na to da 5G totalno promijeni postojeću LTE arhitekturu komunikacije, već da se dodatno gradi na njoj i koristi već postojeće koncepte. Gledajući sa financijske strane, takva strategija će iskoristiti ulaganja u već postojeću infrastrukturu jer struktura već postoji te ju se samo treba nadograditi. Također kako bi se uspostavila takva infrastruktura, svaki segment prometa mora napredovati te prilagoditi se novoj infrastrukturi.

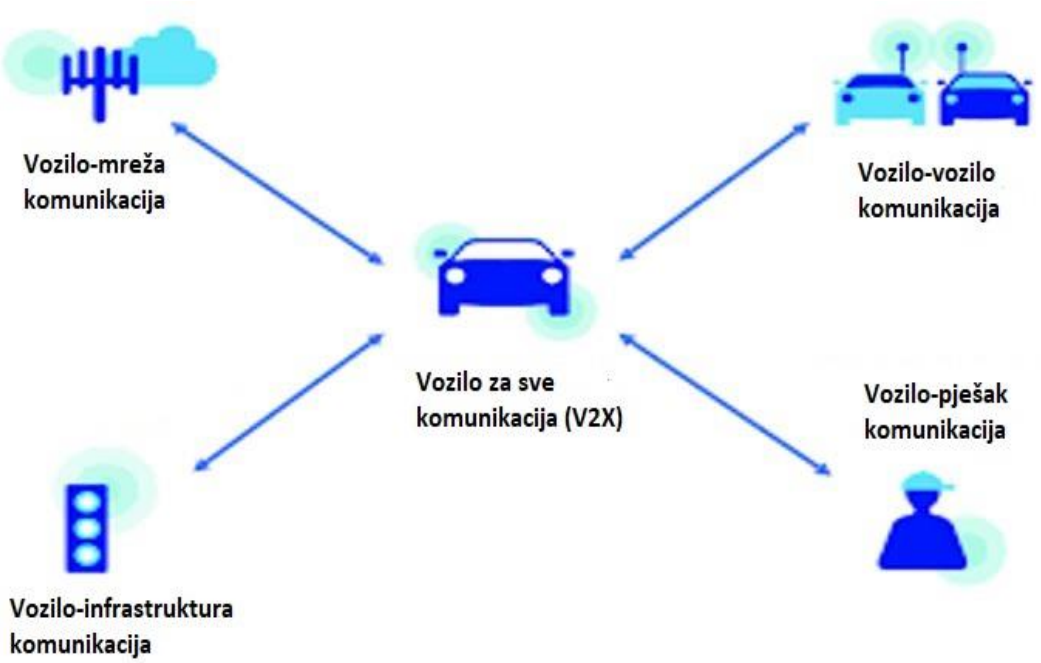


Slika 3.4. Primjer V2X tehnologije [13]

V2X sustavi su relativno novi pojam na tržištu, stoga će se sve prednosti njegove primjene pokazati tek kada se proširi njegova upotreba. Kako bi vozilo moglo putem V2X komunicirati s drugim vozilom u prometu, to vozilo također mora imati ugrađenu tehnologiju koja mu to omogućava. Također isto vrijedi i za sustave ugrađene na cestama kao što su semafori, kamere i slično. Ako samo nekoliko vozila ima ugrađenu V2X tehnologiju, ono neće imati putno koristi od toga jer nema sa kim komunicirati. Što više vozila i infrastrukture ima ugrađen V2X sustav to će se više informacija razmjenjivati između njih i benefiti njegove primjeni će se povećavati. Primjena V2X komunikacije ne staje tu, biciklisti i pješaci su također sudionici u prometu koji mogu donositi kvalitetne informacije koje mogu biti od koristi vozačima i njima samima. Očekuje se da će upotreba V2X tehnologije značajno rasti kroz sljedećih 20 godina. Delphi, Denso, Qualcomm i Continental samo su neki od poznatih proizvođača koji imaju velike planove za integraciju ovog sustava. Puno novih vozila koristi neki oblik ove tehnologije,

pogotovo se to odnosi na luksuznije modele. Ako se sustav pokaže jako koristan te nakon što mu krene opadati cijena ugradnje očekuje se njegova upotreba i u vozilima niže klase. V2X tehnologija će posebno imati utjecaj na autonomna vozila, jer oni konstantno skeniraju svoju okolinu potrazi za korisnim informacijama i opasnostima. Današnja autonomna vozila za prikupljanje podataka iz okoline najčešće koriste kamere i radare koji imaju svoja ograničenja kao što su loši vremenski uvjeti ili loše osvjetljenje. Tu može pomoći V2X komunikacijski sustav koji će vozilu dodatno pomoći u sigurnom kretanju vozila kroz promet. Trenutno postoje četiri implementacije V2X komunikacije. Ove četiri vrte mogu biti korištene sve u isto vrijeme u cilju postizanja velike razine sigurnosti i prikupljanja što više podataka sa okolnih senzora kako bi izbjegli prometne nesreće. Slika 3.5 prikazuje spomenuta četiri tipa V2X komunikacije, a to su:

1. Vozilo prema svemu (V2V)
2. Vozilo prema infrastrukturi (V2I)
3. Vozilo prema pješaku (V2P)
4. Vozilo prema mreži (V2N)



Slika 3.5. Tipovi komunikacije V2X tehnologije

4.4. Strojno učenje

Strojno učenje *(engl. Machine learning)* je praksa korištenja algoritama da parsiraju odnosno raščlane ili analiziraju podatke, uče iz njih, i onda odluče o nečemu ili predvide nešto. Bolje rečeno, to je programiranje računala na način da optimiziraju neki kriterij uspješnosti temeljem podatkovnih primjera ili prethodnog iskustva. Budući da je strojno učenje relativno nova tehnologija i vrlo napredna pitanje je kako funkcionira. Neka postoji iskustvo E, zadatak T i mjera performansi P. Učiti, znači obavljati zadatak T sve uspješnije obzirom na mjeru performansi P kako se iskustvo E povećava. U strojnom učenju, skup podataka može se podijeliti na dvije skupine, validacijski skup podataka i testni skup podataka. Generalizacijska sposobnost neke hipoteze može se izmjeriti na temelju podataka koji nisu korišteni na temelju učenja. Ovaj skup podataka se naziva validacijski skup podataka, dok ako se želi izvjestiti o pogrešci, odnosno kolika je očekivana pogreška najboljeg modela, onda se govori o testnom skupu podataka. Za svako učenje potrebni su podaci za treniranje i podaci za testiranje (Slika 3.6.). Naime, strojno učenje ima veliku ulogu u autonomnoj vožnji te u izgradnji pametnih cesta. Primjene strojnog učenja su razne, a neke od njih su:

- Detekcija objekata
- Identifikacija objekata
- Lokalizacija objekata

Slika 3.6. Princip rada strojnog učenja [14]

Postoji tri vrste strojnog učenja, nadzirano, nenadzirano i podržano. Nadzirano učenje predstavlja način strojnog učenja gdje je cilj odrediti funkcionalnu ovisnost između ulaznih veličina i izlazne veličine na temelju podatkovnih primjera. Nenadzirano učenje na raspolaganju ima samo podatke o ulaznim veličinama, a izlazna veličina ne postoji. Dok, podržano učenje omogućava agentu da samostalno otkrije optimalno ponašanje metodom pokušaja i pogrešaka, agent dobiva odgovarajuću povratnu informaciju za svaku akciju koju izvede. Nakon određenog broja pokušaja, agent bi trebao naučiti najbolji način odabira akcije u svakom koraku, što je zapravo niz akcija koji maksimizira ukupnu nagradu.

Strojno učenje je nezaobilazna tehnologija u automobilskoj industriji. Naime, bez strojnog učenja bilo bi gotovo nemoguće implementirati da automobile na slici koju snimi razdvoji pješaka od automobila. Slika koju kamera automobila snimi prolazi kroz više filtera nad kojima algoritmi obavljaju svoje zadaće te donose zaključak ovisno o slici. Primjerice, donesen je zaključak da se na slici nalaze dva automobila, jedan prometni znak i pješak. Strojno učenje se još koristi u:

- Računalnim igricama
- Znanosti (fizika, kemija, biologija, astronomija)
- Robotici (primjerice kako naučiti robota da igra šah, ili da pronađe izlaz iz labirinta)
- Telekomunikacijama, za poboljšanje mreže i kvalitete usluga
- Medicinskoj dijagnostici

Strojno učenje se može povezati sa statistikom u smislu metoda, jer imaju sličnih osobina, ali su opet različite u svom glavnom cilju. Statistika proučava populacija zaključivanja od samog uzorka, dok se strojno učenje bavi predikcijom određene akcija, odnosno omogućava stroju da pomoću prethodno stečenog iskustva i podatkovnih primjera zaključi nešto i provede odgovarajuću akciju na osnovu tog zaključka. Neki su znanstvenici primijenili ova dva područja te ih povezali u statističko učenje. Primjeri strojnog učenja su:

- Robot koji igra šah
- Stroj koji sam pronalazi izlaz iz labirinta
- Robot koji sam okreče palačinke

5. APLIKACIJA PRIKAZIVANJA PROMETNIH ZNAČAJKI

Svrha projektnog zadatka je napraviti web aplikaciju koja će prikazivati određene statistike u prometu te prikazivati specifikacije automobila. Aplikacija je napravljena u okviru rada "React" te uz dodatnu pomoć još par tehnologija. Naime, aplikacija je napravljena na responzivan način uz pomoć naprednog CSS-a *(engl. Cascading Style Sheets)*. Tehnologije pomoću kojih je napravljen projekt su React, Redux i Firebase. Više o navedenim tehnologijama u nastavku.

5.1. Općenito o Reactu

React je fleksibilna i efikasna Javascript biblioteka koja služi za izradu korisničkih sučelja. Naime, ova biblioteka doživljava sve veću popularnost i njena upotreba je sve češća. Kako je popularnost programskog jezika Javascript sve više rasla tako je došlo i do novih zahtjeva. Pa je čisti Javascript *(engl. Vanila Javascript)* postao najpopularniji jezik na svijetu. Kako ne može sve biti savršeno, tako ni ovaj jezik ne spada u tu priču. Naime, kako bi se postavio najjednostavniji tekst u obliku koda potrebno je kreirati novi element tipa paragraf ili neki drugi, koristeći funkciju "*createElement()*" tek onda se na njega može dodavati funkcionalnost. Ovo je samo jedna stvar od mnogo njih koju React biblioteka olakšava. Naime, React je uveo pojam "jsx" koji sve interakcije između Javascript-a i HTML-a obavlja za programera, kao što je kreiranje novog elementa. Kako bi se količina koda reducirala React je komponente. To su dijelovi koda koji mogu obavljati različite funkcionalnosti, uključivati u druge komponente, prosljeđivati parametar iz jedne komponente u drugu i tako dalje. Postoje dvije vrste komponenata u React-u, a to su klasne komponente i funkcionalne komponente. Klasna komponenta (Programski kod 5.1.) predstavlja komponentu čiji je kod zamotan u klasu koja nosi ime te iste komponente. Sintaksa pisanja klase u klasnoj komponenti ista je kao kod pisanja klase u nekom drugom programskom jeziku objektno orijentiranog programiranja. Na primjeru koji prikazuje programski kod 5.1. može se vidjeti primjer klasne komponente. Naime, klasna komponenta nasljeđuje komponentu koja se nalazi u mehanizmu same biblioteke React-a. Nadalje, ima funkciju render koja služi za prikazivanje napisanog koda na

web aplikaciji tako da bude čitljivo ljudskom oku. Taj kod se može uređivati putem CSS-a pa web aplikacija ima lijep izgled te omogućava korisniku lakše služenje samom aplikacijom.

```
class ShoppingList extends React.Component {
  render() {
    return (
      <div className="shopping-list">
        <h1>Shopping List for {this.props.name}</h1>
        <ul>
          <li>Instagram</li>
          <li>WhatsApp</li>
          <li>Oculus</li>
        </ul>
      </div>
    );
  }
}
```

Programski kod 5.1.. Primjer klasne komponente u React-u [15]

U funkciji return, kao što se može vidjeti na slici gore, se nalazi jsx. U ovom primjeru se nalaze elementi div, h1, ul i li. Naslov je prikazan elementom h1, ul *(engl. unordered list)*, li *(engl. ordered list)* te div element koji zamotava sve što se nalazi u funkciji *render()*, te mu se pridružuje određeno ime klase. Ključna riječ "this" referencira se na klasu te je neizbježan djelić koda bez kojeg klasne komponente jednostavno ne mogu funkcionirati. Nekad, ova riječ može biti jako zbunjujuća, pogotovo ako je netko tek krenuo ulaziti u svijet objektno orijentiranog programiranja. Kako bi se izbjegla ova riječ ove se funkcionalne komponente. Naime, ovo je samo jedan od mnoštvo primjera koji može potvrditi kako je uvođenje funkcionalnih komponenata unaprijedilo React biblioteku. Funkcionalne komponente su komponente napisane u funkcijama. Primjer funkcionalne komponente prikazuje programski kod 5.2. Uvođenjem funkcionalnih komponenata popularnost React-a je počela vrlo brzo rasti, jer je React napredovao u funkcionalnom, optimizacijskom i mehanizacijskom smislu. Funkcionalne komponente dozvoljavaju korištenje hooks-ova. Hooks je također funkcija podržana od strane React funkcionalnih komponenata.

Naime, ove funkcije pružaju jako puno prednosti, među ostalim i manje pisanja koda te bolju optimizaciju.

```
import React, { useState } from 'react';

function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

Programski kod 5.2. Primjer funkcionalne komponente u React-u [15]

Vrijednosti stanja *(engl. state)* u klasnim komponentama bi se morali navesti u konstruktoru koji pa bi se kasnije ažuriralo stanje putem funkcije *setState()* što nije uopće loš način, ali se mora pisati više koda da bi se postiglo isto ono što se sa hooks-ima postigne uz puno manje linija koda. Naime, najpoznatiji hooks je *useState()*, jedan od brojnih. Ovaj hooks se postavi tako da se napiše ime stanja te ime vrijednosti koje je zaduženo za ažuriranje *(engl. update)* tog istog state. Ta se vrijednost po konvenciji naziva sa prefiksom "*set*" te se onda doda bilo koje ime. Također, gdje god da je potrebno ažurirati stanje odnosno state samo se pozove funkcija set pa bilo koje ime. Navedeni primjer se može vidjeti na programskim kodovima 5.3 i 5.4. Kao što je već navedeno, u klasnim komponentama se mora pozvati konstruktor. Konstruktor je specijalni tip metode koji se zove prilikom inicijalizacije klase. Ova metoda inače u objektno orijentiranom programiranju prima parameter koje se u React-u navode u state-u. Kako klasa nasljeđuje vrijednosti odnosno osobine *(engl. props)* iz React komponente (extends React.Component) potrebno je pozvati funkciju *super()*. Kad se klasa kreira ona u pozadini stvori objekt koji se zove "*this*". Ključna riječ s istim imenom koja se referencira na klasu, kao što je već navedeno, je upravo taj objekt koje se kreira u pozadini. Kako bi se dodalo stanje vrijednosti odnosno *state* potrebno je napisati

"*this.state*", kako bi objekt "*this*" primio state te da bi se kasnije moglo rukovati s tim vrijednosnim stanjem.

```
class Komponenta extends React.Component {
  constructor(props){
    super(props)
    this.state = {
      name: "Pero"
    }
  }
}
```

Programski kod 5.3. Primjer navođenja state-a u klasnoj komponenti

Kao što se na navedenim slikama može vidjeti, sintaksa je puno kraća korištenjem hooks-a te se u velikim projektima gdje postoji jako puno komponenata, puno lakše snalazi u funkcionalnim komponentama nego u klasnim. Također, hooks-i cijelu ovu priču sa kreiranjem objekta u pozadini rješava za programera, pa se programer ne mora ni brinuti o tome.

```
const Komponenta = () => {
  const [name, setName] = useState("Pero")
}
```

Programski kod 5.4. Primjer navođenja state-a u klasnoj komponenti

Jako bitna stvar kod React-a su takozvani "propsi", odnosno osobine. Naime, osobine se mogu prosljeđivati iz jedne komponente u drugu što omogućava ponovno iskorištavanje iste komponente za različitu svrhu *(engl. reuse)*. Ovaj princip je zapravo smisao React-a te omogućava kraći, čišći i ljepši kod, što je jako bitno u profesionalnom programiranju. Komponenta prime osobine od

druge komponente kao parametar te ih koristi za što god je potrebno. Recimo da postoji komponenta koje se zove “glavnaKomponenta” te se u njoj poziva komponenta koja se zove “Komponenta”. Glavna komponenta ima definiranu neku vrijednost koja sadrži ime neke osobe, recimo “Pero” te proslijedi tu osobinu komponenti kao props, ali pod određenim imenom. Komponenta naziva “Komponenta” je primila tu osobinu te je može ispisati unutar svoje “return” funkcije. Ovaj primjer se može vidjeti na programskom kodu 5.5.

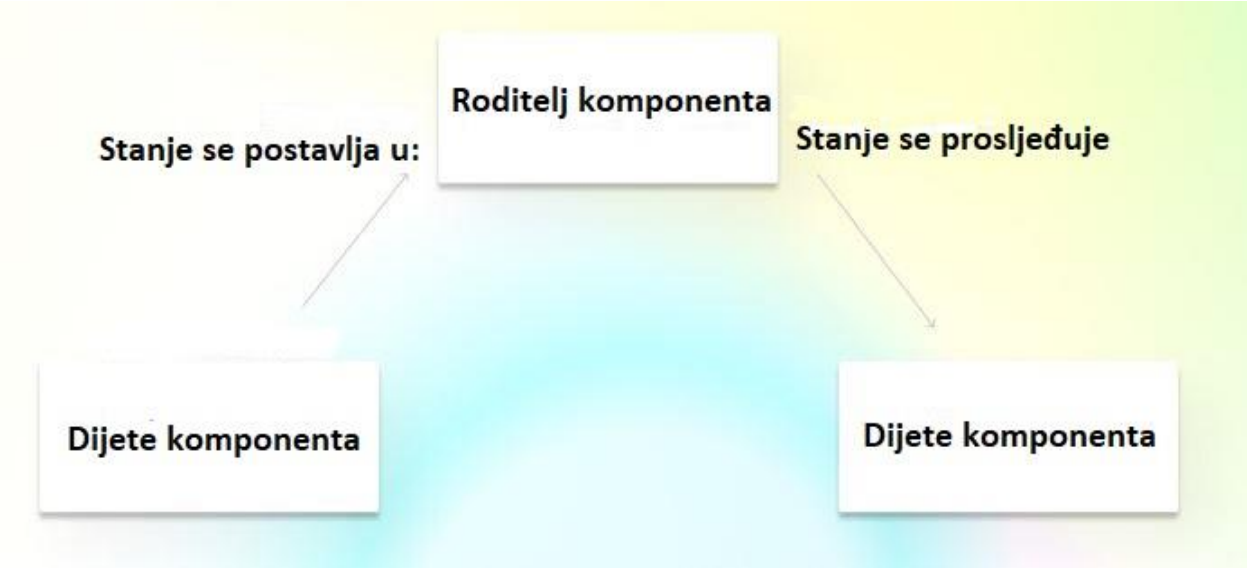
```
const glavnaKomponenta = () => {
  const ime = "Pero";
  return(
    <div>
      <Komponenta ime={ime}
    </div>
  )
}

const Komponenta = ({ime}) => {
  return (
    <div>
      <p>{ime}</p>
    </div>
  )
}
```

Programski kod 5.5. Prosljeđivanje props-a u React komponentama

Kako bi se struktura React koda održala što boljom i efikasnijom potrebno je pratiti se generalno prihvaćenih konvencija. Prvo pravilo ove biblioteke glasi “*Lifting State Up*”. Kako bi se moglo shvatiti pravili potrebno je shvatiti strukturu hijerarhije u React-u. Naime, princip React se bazira na roditelj-dijete odnosu *(engl. parent-child)*. Roditelj komponenta može prosljeđivati osobine djetetu komponenti, dok obrnuto ne može. Također, roditelj komponenta može imati više djece komponenti. Kako bi kod bio što čišći i kako se nakon puno rada na projektu programeri ne bi

izgubili u vlastitom kodu, poželjno je pratiti gore navedeno pravilo. Što znači da se state piše i prvoj roditelj komponenti te se samo prosljeđuje djeci. U tom slučaju se točno zna odakle potječe koja vrijednost i kojim se komponentama se prosljeđuje. Također, ukoliko je potrebno nešto promijeniti, to se može napraviti samo na jednom mjestu bez ikakvih problema. Primjerice ako se state piše raštrkano po više komponenti te se u jednom trenutku odluči da je u roditelj komponenti potrebno to stanje iz dijete komponente, potrebno je brisati dobar dio koda te ga pisati ponovno. Sve to se može spriječiti koristeći samo jedno pravilo (Slika 5.1.).

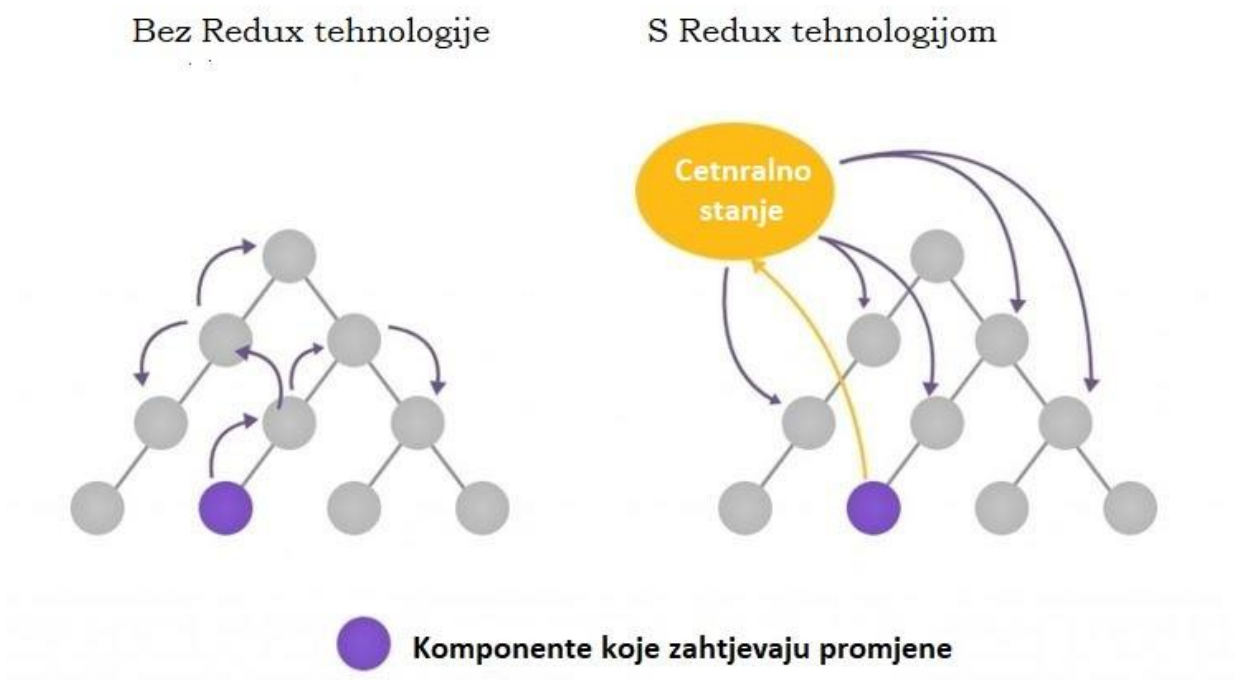
Aplikacija u ovom radu napravljena je u Reactu. Koriste se funkcionalne komponente te se komponente ponovo koriste gdje je god to moguće. Stanje vrijednosti *(engl. state)* je podignut na najveću razinu te se vrijednosti prosljeđuju u druge komponente putem osobina *(engl. props)*. CSS kod je napisan zasebnim datotekama te su pridružene svakoj klasi zasebno, što omogućava čisto napisan i dobro strukturiran kod.



Slika 5.1.. Prosljeđivanje propsa u React komponentama [16]

5.2. Redux

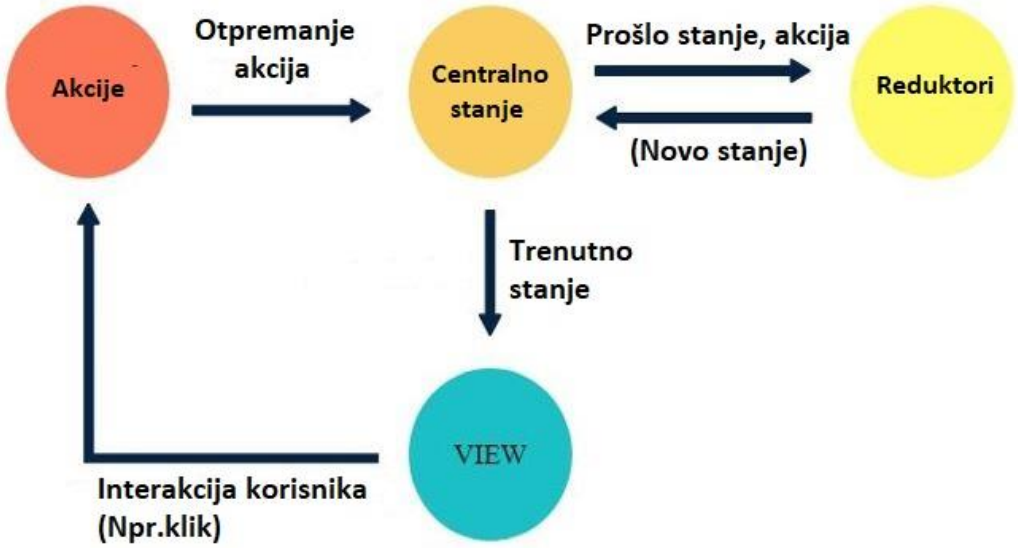
Redux je još jedna Javascript biblioteka koja služi za održavanje i upravljanje glavnog stanja *(engl.state)* aplikacije. Ova tehnologija je olakšala život frontend programerima te je sve popularnija. Redux su stvorili Dan Abramov i Andrew Clark. Naime, ova tehnologija održava stanje cijele aplikacije na jednom mjestu te omogućava da se stanje koristi u bilo kojoj komponenti bez razmišljanja o prosljeđivanju stanja i osobina komponenti. Što daje aplikaciji na učinkovitosti, poboljšava njenu optimizaciju te olakšava programerima prilikom kodiranja. U ogromnim aplikacijama gdje ima jako puno komponenata s time i koda, jako je teško se snaći u kodu i u njegovoj hijerarhiji. Također, je teško postići usklađeno ažuriranje svih stanja u aplikaciji ako su raštrkani na puno strana. Kako bi se najbolje shvatio koncept ove tehnologije najbolje ju je objasniti na primjeru. Recimo da se radi o stranici za kupovinu odjeće. Korisniku je omogućeno da pretražuje po majicama, hlačama, košuljama i slično. Svaka vrsta odjeće predstavlja jedan niz komponenata koji je zadužen za dohvaćanje informacija o određenom komadu odjeće iz baze. Kada se informacije dohvate iz baze mogu se putem Reacta predstaviti na korisničkom sučelju. Kada korisnik izabere majicu ili bilo koji drugi komad odjeće, ima mogućnost ju staviti u košaricu. Košarica je također komponenta u kojoj se sad nalazi majica koju je korisnik odabrao. Ako korisnik odabere recimo i hlače, koje se nalaze u novom nizu komponenata, onda se i hlače šalju u košaricu. Kako bi se svaki komad odjeće poslao u košaricu potrebno je jako puno prosljeđivanja kroz jako puno komponenata što utječe na brzinu i performanse aplikacije. Upravo ovdje stupa Redux na snagu. Naime, Redux omogućava da se sva odjeća koja se želi staviti u košaricu pošalje u takozvano skladište *(engl. store)*, koje predstavlja središnje stanje aplikacije. Stanja koja se nalaze u skladištu moguće je ažurirati i utjecati na njih iz bilo kojeg dijela aplikacije samo se mora određena komponenta povezati sa skladištem. Primjer kako bi prosljeđivanje izgledalo s Reduxom i bez njega može se vidjeti na slici 5.2. Dakle, stanje cijele aplikacije je postavljeno u jedno skladište. Stanja u skladištu se ne mogu mijenjati direktno, tj. imaju svojstvo nepromjenjivosti ten a taj način je sačuvan integritet stanja aplikacije. Ako se želi utjecati na stanje najbolje je prvo kopirati to stanje u novu varijablu te ga onda promijeniti. Nakon što je varijabla u kojoj se zapravo nalazi vrijednost iz stanja promijenjena, ta promijenjena vrijednost se vraća nazad u stanje koje se nalazi u skladištu. Centralno skladište se kreira metodom createStore, koja prima argumente “rootReducer”, “initialState” i konstantu za posrednički softver.



Slika 5.2.. Prosljeđivanje stanja bez Reduxa (lijeva strana slike) i s Reduxom (desna strana slike) [17]

Mehanizam Redux tehnologije može biti malo zbunjujući te u manjim aplikacijama može biti prekompleksan jer je ova tehnologija namijenjena za ogromne aplikacije i projekte. Mehanizam se sastoji od akcija, reduktora i skladišta. Naime, reduktori *(engl. reducers)* su Javascript funkcije koje definiraju kako se mijenja vrijednost stanja aplikacije. Reduktor ažurira stanje ovisno o informaciji koju je dobio od akcije. On prima dva parametra, a to su prijašnja vrijednost stanja te nova vrijednost stanja koja će se tek ažurirati. Nakon što je reduktor obavio svoju zadaću te utjecao na središnje stanje aplikacije on vraća nazad promijenjeno stanje u skladište te nakon toga stanje ide na korisničko sučelje gdje korisnik može vidjeti to isto stanje. Nadalje, akcije su obični Javascript objekti koji opisuju što se događa sa stanjem stanja, ali ne opisuju i kako se ta ista promjena događa. Svaka akcija ima tip i takozvani "*payload*"[9]. Primjer cijelog toka mehanizma Reduxa može na slici 5.3.

[9] Payload- predstavlja objekt u kojem se nalaze informacije o odgovarajućem stanju



Slika 5.3. Princip rada Reduxa [17]

Tehnologija Redux se može instalirati u projekt Reacta na način da se instalira kao npm[10] paket putem naredbe. Te se cijela aplikacija omota sa komponentom od Reduxa zvanom "Provider". Primjer navedenog se može vidjeti na slici 5.4.

```
1  <Provider store={store}>
2    <App />
3  </Provider>
```

Slika 5.4. Princip rada Reduxa

[10] Npm- upravitelj paketa za programski jezik

5.3. Firebase

Firebase je web platforma koju je Google razvio s ciljem izrade web aplikacija. Ova aplikacija pruža mnoge usluge aplikacijama. Kao što su backend usluge te usluge baze podataka. Prije svega, ovu platformu je jako lako uključiti u bilo koji projekt. Naime, na dokumentaciji Firebasea se može pronaći inicijalni kod koji se samo kopira u projekt u koji se želi dodati Firebase. Ova platforma pruža brojne usluge kao što su:

- Analtytics
- Cloud Messaging
- Authentication
- RealTime database
- Storage
- Hosting

Firebase se uključuje u projekt putem generiranog koda koji nudi sam Firebase. Potrebno je kopirati API[11] ključ, identifikacijski ključ projekta, link od baze podataka i slično. Nakon kopiranja svih bitnih podataka od Firebasea potrebno je provesti inicijalizaciju putem funkcije initializeApp. Primjer uvođenja Firebasea u projekt može se vidjeti na programskom kodu 5.6. Ovim putem se uvodi backend kod koji se može koristiti u cijeloj aplikaciji. Također pruža razne funkcije koje su već odrađene od strane platforme. Na primjer, ako se želi autentificirati korisnika, nije potrebno pisati cijelu logiku koja će provjeravati je ime korisnika ispravno uneseno ili jel korisnik ispravno autentificiran, već sve to obavlja Firebase Authentication. Također, prijave i odjave korisnika su isto pokrivene od strane funkcija koju pruže Firebase, kao što su:

- *signInWithEmailAndPassword()*
- *logInWithEmailAndPassword ()*
- l*ogout()*

[11] API – aplication programming interface

```
<script src="https://www.gstatic.com/firebasejs/5.10.0/firebase.js"></script>

<script>
  // Initialize Firebase
  var config = {
    apiKey: "AzaSyD-OnE3LRISh8ud539-s8SeTo",
    authDomain: "authenticationta1.firebaseapp.com",
    databaseURL: "https://authenticationt1.firebaseio.com",
    projectId: "authenticationtes1",
    storageBucket: "authenticatio11.appspot.com",
    messagingSenderId: "3033264672"
  };
  firebase.initializeApp(config);
```

Programski kod 5.6. Primjer implementacije Firebase platforme u projekt [18]

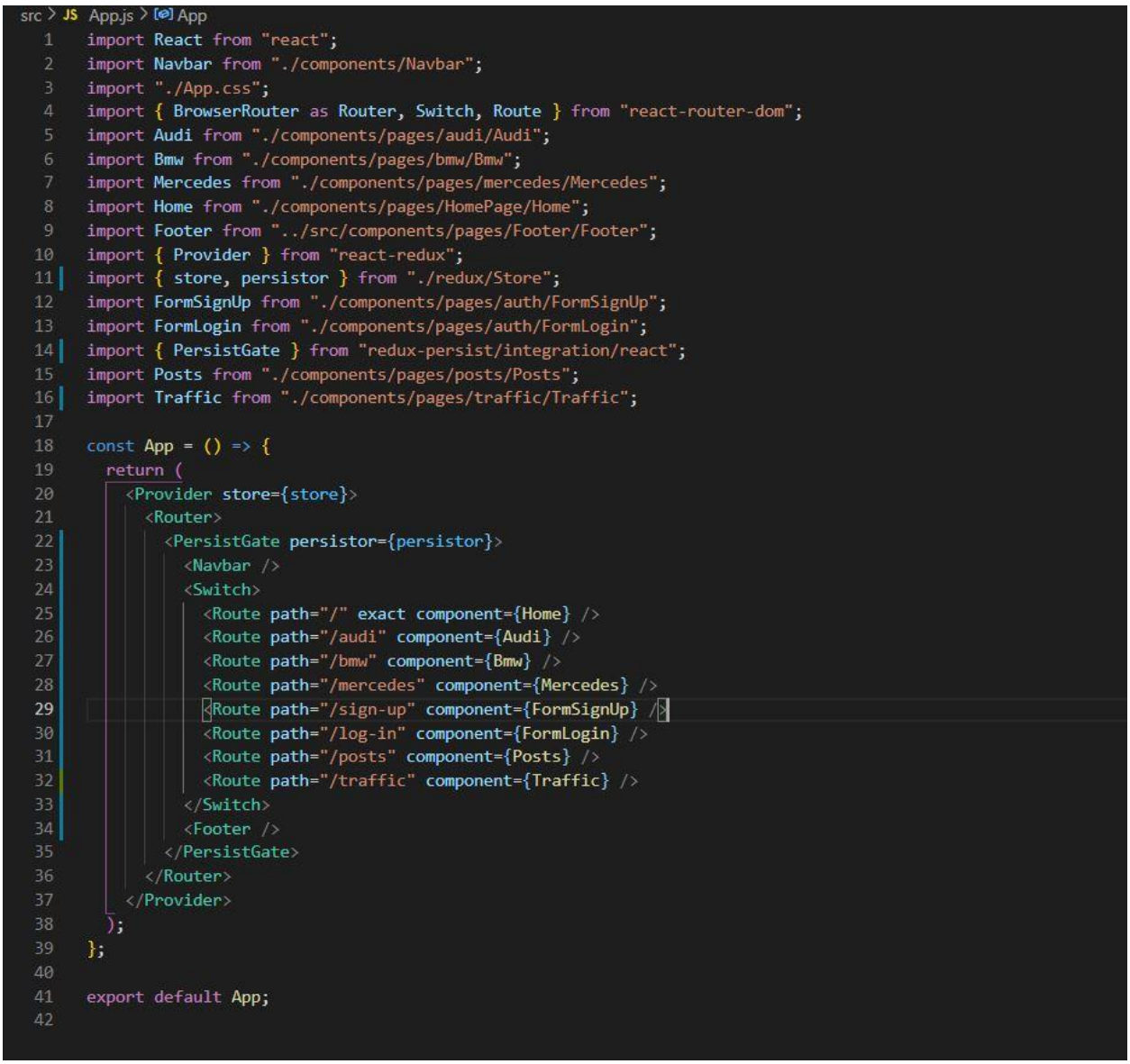
Firebase baza podataka vrlo dobro razvijena i omogućava pohranu, interakciju i sinkronizaciju podataka u stvarnom vremenu. Također moguće je vidjeti bazu u prikazu na web stranici Firebasea i utjecati na nju izravno klikom ili iz koda putem naredbi. Ova baza podataka ima svoje restrikcije i pravila. Pa tako ima točno određena pravila tko smije pristupiti bazi podataka te ako se baza dugo nije koristila Firebase onemogućava pristup bazi.

5.4. Aplikacija prikazivanja prometnih značajki

Aplikacija je napravljena u React biblioteci koristeći Redux kao skladište vrijednosti stanja i Firebase kao backend koncept sa postavljenom bazom podataka. Koncipirana je na način da početna stranica predstavlja pojedinosti o tri skupine automobila koja su odabrana nasumično. Za svaki od automobila je napravljena posebna stranica na kojoj se može naći općenito o određenoj marki automobila. Prvi gumb predstavlja običan tekst koji se pojavi na klik gumba. Ispod teksta se nalaze slike koje su napravljene u obliku animacije te se mogu mijenjati također na klik gumba, takozvani "*slide show*". Ispod animacija se nalaze gumbovi koji kad se kliknu pokažu specifikacije o pojedinim vozilima, čije se slike nalaze iznad u animaciji. Također, su napravljeni gumbovi koji kad se kliknu mijenjaju boju automobila kako bi korisnik mogao odabrati boju automobila koja

mu se najviše sviđa. Na navigacijskoj traci se može naći i stranica koja prikazuje statistike prometa prikazane u grafovima te karta na kojoj se može odabrati najbolja ruta od točke A do točke B. Navigacijska traka posjeduje i gumb za prijavu korisniku, koji ako se klikne odvede korisnika na stranicu za prijavu. Prije nego se korisnik prijavi aplikacija validira korisnika i odlučuje hoće li ga pustiti da se prijavi u sustav. Nakon prijave, aplikacija vodi korisnika na blog stranicu, gdje može pisati komentare, brisati ih, mijenjati u slučaju potrebe te ih može brisati. Opis cijele aplikacije detaljno će biti prikazan u nastavku.

Glavna datoteka aplikacije se zove "App.js". U ovoj datoteci su postavljeni svi glavni parametri koji su bitni kako bi aplikacija funkcionirala. Prva stavka ove datoteke govori o povezivanju centralnog stanja aplikacije koje pruža Redux tehnologija. Nadalje, sve rute koje se pozivaju putem cijele aplikacije se nalaze upravo u ovoj datoteci, što je postignuto putem npm pakte koji se zove "react-router-dom". Pomoću ovog paketa omogućeno je povezivanje komponenata putem ruta aplikacije. Kako se podaci korisnika ne bi izgubili nakon prelaska s jedne rute na drugu, dodana je nova značajka iz Redux paketa koja se zove "persistor". Persistor služi kako bi sve podatke unutar jedne sesije zadržao, tako da ne dođe do gubitka korisnikovih podataka usred korištenja aplikacije. Na vrh korisničkog sučelja je postavljena navigacijska traka, pomoću koje se korisnik orijentira po aplikaciji te izabire rute koje želi posjetiti. Navigacijska traka je ubačena na vrh "App.js" datoteke, a na dnu se nalazi komponenta "Footer.js" koja predstavlja podnožje aplikacije te ima određene funkcionalnosti. Izgled glavne datoteke, odnosno "App.js" datoteke se može vidjeti na programskom kodu 5.7. Ova datoteka na dnu ima takozvanu "export" značajku koja omogućava glavnoj datoteci ove funkcije da se ubaci u korijen cijele aplikacije *(engl. root)* koja će se prikazivati na web pregledniku. Naime, korijenska datoteka šalje programski kod na web preglednik u obliku „html" dokumenta. Taj dokument prima motor web preglednika koji čita kod, prevodi ga te ga prikazuje u obliku koji je čitljiv ljudskom oku. Kada se rendera stranica, stvaraju se datoteke koje prevode dio koda koji se rendera u tom trenutku, na taj način se spriječava da se memorija ne popuni. Za oslobađanje memorije se brine prikupljač smeća *(engl. Garbage collector)* koji jednostavno isprazni memoriju u trenutku kada se memorija krene puniti. Na taj način se programer koji kodira u „JS" programskom jeziku ne mora brinuti oko upravljanja s memorijom kao što bi se morao brinuti da kodira u recimo C programskom jeziku.



Programski kod 5.7. Glavna datoteka funkcije, "App.js"

Glavno stanje aplikacije, koje je ubačeno u glavnu datoteku se nalazi u "Store.js" datoteci. Ovdje je provedena inicijalizacija centralnog stanja aplikacije, inicijalizacija korijenskog reduktora te inicijalizacija posredničkog softvera koji se zove "thunk". Naime, aplikacija ima više reduktora, a svi reduktori aplikacije se zajedno pozivaju na mjestu korijenskog reduktora koji se nalazi u

centralnom skladištu glavnog stanja aplikacije. Izgled Store.js datoteke se može vidjeti na programskom kodu 5.8.

src > redux > JS Store.js > ...

```
1   import thunk from "redux-thunk";
2   import { applyMiddleware, createStore } from "redux";
3   import { composeWithDevTools } from "redux-devtools-extension";
4   import RootReducers from "./rootReducers";
5   import { persistStore } from "redux-persist";
6
7
8
9   export const store = createStore(
10    RootReducers,
11    composeWithDevTools(applyMiddleware(thunk))
12  );
13  export const persistor = persistStore(store);
14
15  export default { store, persistor };
16
```

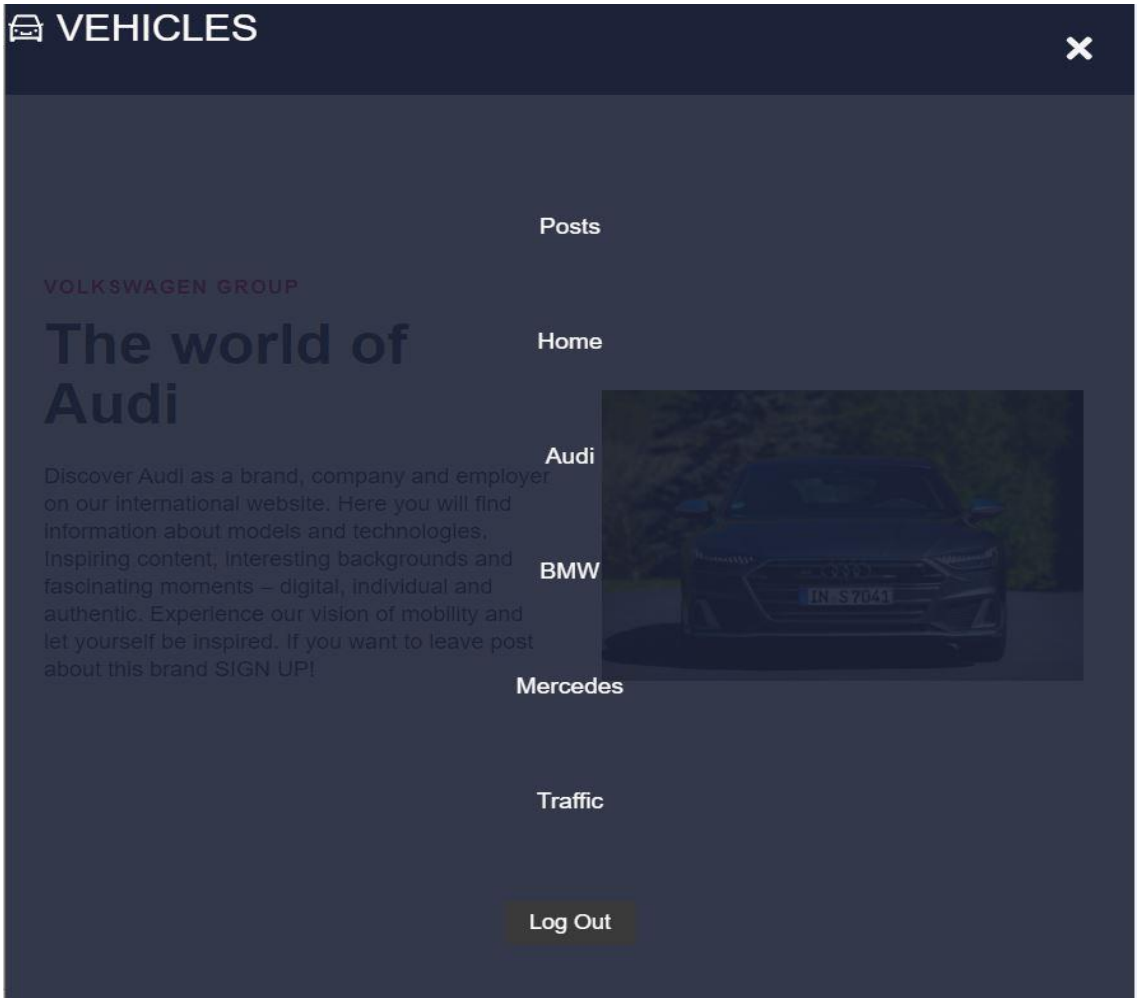
Programski kod 5.8. Skladište centralnog stanja aplikacije, "Store.js"

Navigacijska traka aplikacije se nalazi na vrhu te sadrži sve rute koje se mogu posjetiti u aplikaciji. Koja god ruta da se odabere, glavno stanje se uvijek može dohvatiti iz centralnog skladišta koje se nalazi u Store.js datoteci. Naime, kada korisnik odabere jednu rutu te klikne na nju, glavna datoteka će to prepoznati te će aktivirati odgovarajuću rutu i putem "react-router-dom" paketa će odvesti korisnika na rutu koju je odabra. Ruta se može zamisliti kao vlak koji vodi do određene komponente. Na desnoj strani navigacijske trake nalazi se ikona s natpisom "VEHICLES" koji pogađa istu rutu kao i Home. Ostale rute na navigacijskoj traci su Mercedes, Audi, Bmw, ruta za kreiranje korisničkog računa te ruta Posts koja se pokazuje samo ako je korisnik napravio račun. Mercedes, Bmw i Audi su stranice koje vode na komponente istih naziva te prikazuju podatke, slike i specifikacije o upravo tim automobilima. Navigacijska traka zauzima cijelu dužinu aplikacije te je njena pozicija fiksna i smanjuje se usporedno sa smanjivanjem rezolucije ekrana. Što znači da ako korisnik smanji rezoluciju ekrana ili koristi aplikaciju na različitim veličinama

ekrana, navigacijska traka se neće polomiti već će ostati ista neovisno na kojoj veličini ekrana se prikazuje. Izgled navigacijske trake može se vidjeti na slici 5.5.

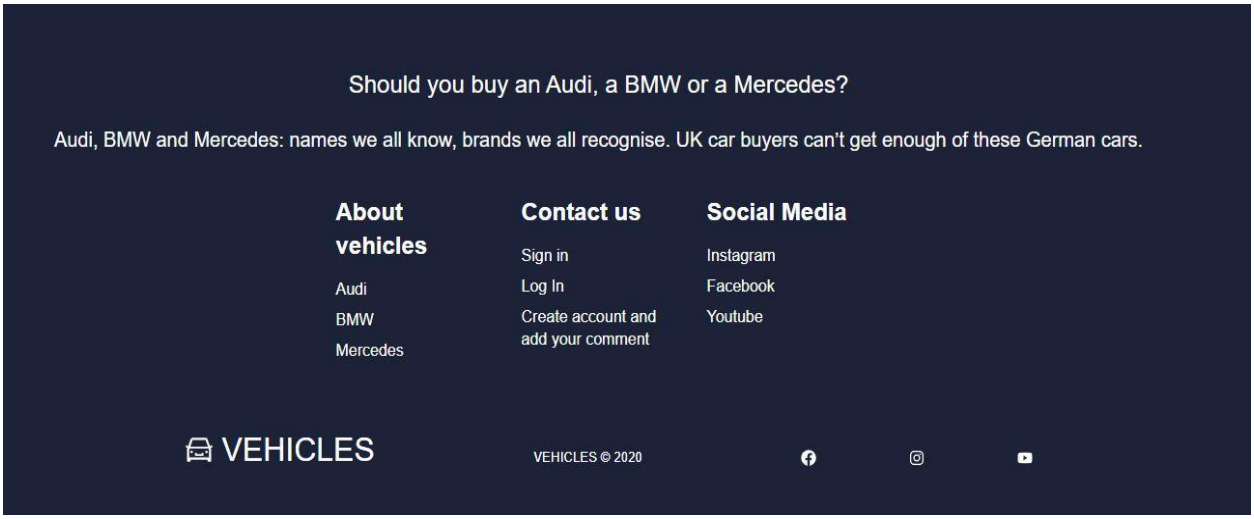
Slika 5.5. Navigacijska traka

Kao što se već moglo zaključiti, aplikacija je responzivna, što znači da će na nekom drugom prikazu također imati određeni izgled, primjerice na mobilnom prikazu što se može vidjeti na slici 5.6.



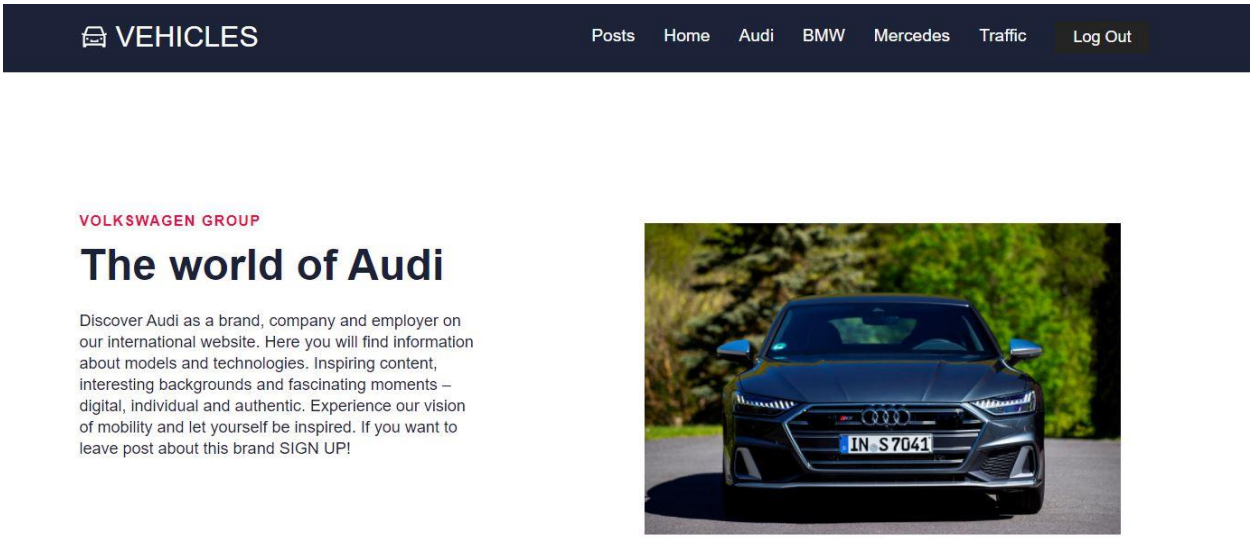
Slika 5.6. Navigacijska traka u mobilnom prikazu

Podnožje aplikacije je također fiksno te sadrži različite linkove koje korisnik može odabrati kao što su linkovi za pojedine rute aplikacije te linkove za facebook, Instagram i link za kreiranje novog računa. Izgled podnožja aplikacije *(engl. Footer)* se može vidjeti na slici 5.7. Kod za podnožje stranice se nalazi u datoteci Footer.js. Pod kod se podrazumijeva cijela logika te kodovi za dizajn podnožja stranice.



Slika 5.7. Podnožje aplikacije, ("Footer.js")

Početna stranica je vrlo jednostavna te nema puno logičkog koda u njoj. Ona se nalazi u datoteci HomePage.js. Prikazuje općeniti tekst o pojedinoj marki automobila te sliku na kojoj se nalazi sam automobil. Primjer jednog automobila se može vidjeti na slici 5.8. Podaci koji se mogu vidjeti na početnoj stranici, nalaze se u kodu aplikacije. Napravljena je jedna komponenta koja prima više podataka. Svi podaci o jednom automobilu se nalaze u zasebnoj datoteci te se ta datoteka ubacuje u komponentu koja prikazuje automobile na početnoj stranici. Na taj način je u potpunosti iskorišten potencijal Reacta jer ponovno koristimo istu komponentu za prikazivanje različitih podataka. Ovaj pojam se prema [15], naziva *reuse*. Kod za izgled početne stranice se nalazi u zasebnoj datoteci koja ima domenu css, što znači da prikazuje css kod. Kao što je već navedeno u radu, css kod služi za dizajniranje izgleda aplikacije.



Slika 5.8. Izgled početne stranice aplikacije (Izvor slike [19])

Ako korisnik odabere jedan od automobile, aplikacija će ga odvesti na rutu koja pokazuje na komponentu izabranog automobila. Za svaki automobil stranica izgleda jednako pa će se obraditi samo jedan primjer. Recimo da je korisnik izabrao Audi. Pojavit će se nova stranica koja prikazuje sve o navedenom vozilu. Na vrhu stranice se nalazi tekst koji govori općenito o automobile te ispod toga se nalazi sliko-prikaz. Naime, sliko-prikaz *(engl. slide-show)* je napravljen na sljedeći način. Ovdje je napravljen samostalni hook, koji prima parametre te mijenja tri različite slike vozila. Logika za mijenjanje slika se nalazi u datoteci ImageSlider.js, te se može vidjeti na programskom kodu 5.9. Logika je bazirana na matematičkoj x i y osi. Ovaj hook će primiti sliku kao parametar, koji se nalazi u objektu, budući da se sve u ovom programskom jeziku bazira na objekte i polja. Naime, svi objekti, a u objektima se nalaze slike, su stavljeni u polje *(engl. array)* koje se zove sliderArr. Ovo polje je potrebno mapirati putem funkcije map, što znači da će logika utjecati na svaki element u navedenom polju. Funkcije koje utječu na slike su funkcije za pomicanje u lijevo i u desno, te ako korisnik klikne da gumb za pomicanje u lijevo slika će se pomaknuti u lijevo te će druga slika doći na poziciju stare slike, što će ljudskom oku izgledati kao mijenjanje slike. Na slici je prikazan pojam item koji prima referencu na adresu u memoriji gdje se nalazi zapravo slika koju upravlja ovaj hook.

```
export const ImageSlider = ({ goLeft, goRight, x, sliderArr }) => {
  return (
    <div className="slider">
      {sliderArr.map((item, index) => {
        return (
          <div
            className="slide"
            key={index}
            style={{ transform: `translateX(${x}%)` }}
          >
            {item}
          </div>
        );
      })}
      <button id="goLeft" onClick={goLeft}>
        <i className="fas fa-chevron-left"></i>
      </button>
      <button id="goRight" onClick={goRight}>
        <i className="fas fa-chevron-right"></i>
      </button>
    </div>
  );
};

export default ImageSlider;
```

Programski kod 5.9. Logika za pomicanje slika na sučelju

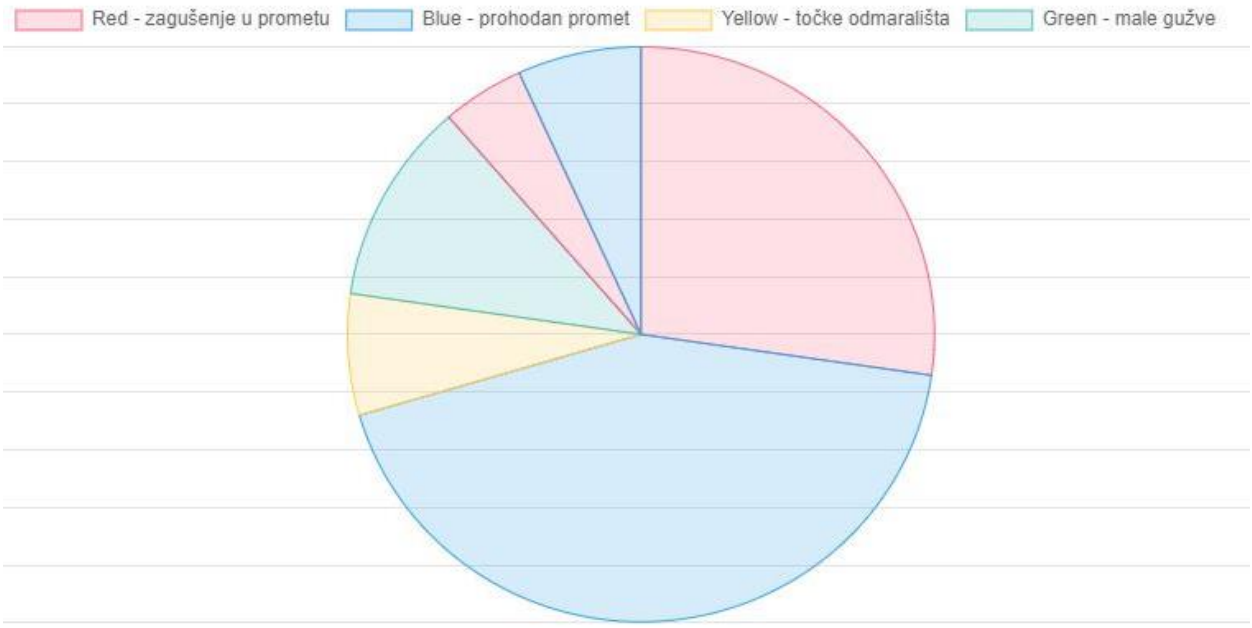
Sliko prikaz koji se prikazuje na korisničkom sučelju izvršava se na sljedeći način. Na slici se mogu vidjeti dva gumba s lijeve i desne strane. Kada se klikne na jedan od ta dva gumba slika će se pomaknuti u stranu, ovisno koji je gumb pritisnut. Kada se prijeđe preko slike s mišem, rubovi slike će se zasvijetlili kako bi pokazali korisniku točno gdje je gumb za pomicanje slika. Ovaj efekt se naziva hover efekt, koji je ja popularan u css svijetu te jako često korišten. Ispod sliko prikaza nalaze se specifikacije o pojedinom automobilu, kao što su potrošnja goriva, emisija CO2, godina proizvodnje i slično. Specifikacije će se prikazati na klik korisnika pojedinačno ovisno o odabiru automobila, što se može vidjeti na slici 5.9.

Slika 5.9. Specifikacije automobila

Logika za prikazivanje specifikacija funkcionira na isti način kao i logika za podatke na početnoj stranici. Podaci se ubacuju u komponentu koja na kaju daje prikaz kao na slici 4.4.9. Ispod specifikacija nalazi se gumb i slika pojedinog vozila. Na pritisak gumba korisnik može promijeniti boju automobila. Na paleti za boje, prisutne su dvije boje za svako vozilo te ovise o vozilu. Nema svako vozilo iste mogućnosti boja. Ovaj pristup je postignut na način da gumb mijenja stanje samo jedne komponente. Što je vrlo bitno jer gumb ne mijenja centralno stanje aplikacije, već samo jedno stanje unutar te iste komponente.

Kada korisnik, na navigacijskoj traci odabere rutu Traffic, odvest će ga na novu stranicu koja sadrži podatke o prometu te statistike. Na početku stranice nalazi se karta koja je ubačena u projekt putem Google Maps API-ja. Naime, korisnik može odabrati rute koje ga zanimaju te dobiti kao povratnu informaciju određene statistike na tim rutama. Naime, statistike su prikupljene preko API poziva koji ukazuje na podatke u stvarnom vremenu o svim detaljima u prometu. Kamere i senzori prikupljaju informacije, obrađuju ih te se zapisuju u ovom slučaju u „json“ obliku koji se hvata preko API poziva te se obrađuje i prikazuje u aplikaciji u obliku statistike. API ključ preko kojeg se hvata je jedinstven i kriptiran kako bi sigurnost bila na nivou.

Kada korisnik odabere ciljanu rutu dobit će povratnu informaciju o tome kakvo je stanje na cestama, ima li gužve te gdje može stati i odmoriti. Također, će dobiti graf koji prikazuje na danoj ruti koliko ima zagušenja u prometu, na kojim dionicama je prohodan promet, te na kojim dionicama su male gužve, kao i gdje se nalaze točke odmarališta. Graf je prikazan u ovisnosti o postocima te se može vidjeti na slici 5.10.



Slika 5.10. Graf prikaza stanja na odabranoj ruti

Glavna komponenta prikaza statistike u prometu se zove Traffic.js te je ona roditelj komponenta ostalim komponentama koje prikazuju statistike u prometu. Ona je vrlo jednostavna komponenta koja poziva svoje podređene komponente te ih prikazuje. Također se nalazi u glavnoj App.js komponenti i predstavljena je kao ruta koja vodi na korisničko sučelje za statistiku u prometu putem navigacijske trake. Izgled "Traffic.js" komponente se može vidjeti na programskom kodu 5.11. Recimo da je korisnik izabrao rutu od Osijeka do Splita. Nakon što algoritam preračuna rutu te vrati korisniku na sučelje podatak u obliku grafa navedenog iznad. Korisnika također zanima kakvo je stanje na toj ruti u proteklih par mjeseci. Naime, nije uvijek ista prohodnost na cestama

te nisu uvijek iste gužve na istim dionicama. Aplikacija preračunava sve informacije koje je dobila preko “Google” API-ja koji se može vidjeti na programskom kodu 5.10.

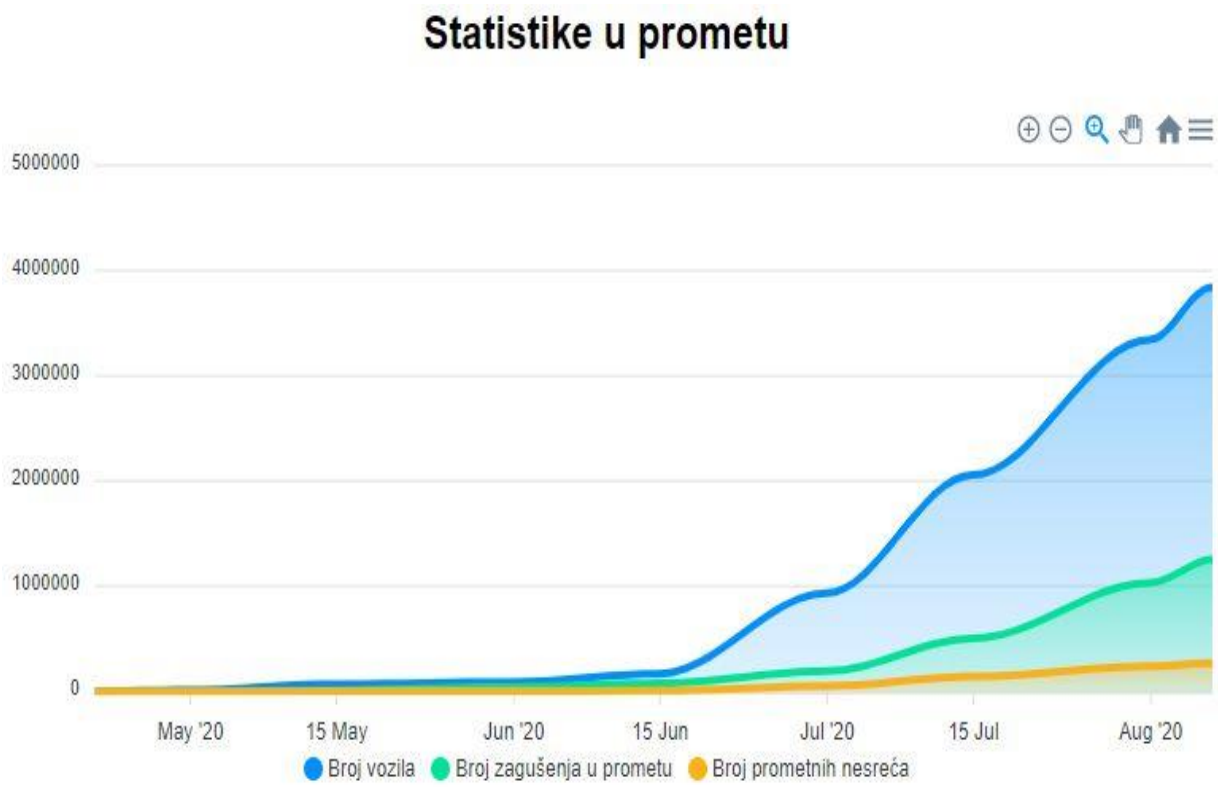
```
<MapWrapped
  googleMapURL={
      `https://maps.googleapis.com/maps/api/js?v=3.exp&libraries=geometry,
      drawing,places&key=AIzaSyAgaVOi33HerJ3T1VIHPUZRnaJw5FtqM5w
      `
  }
  loadingElement={<div style={{ height: `100%` }} />}
  containerElement={<div style={{ height: `100%` }} />}
  mapElement={<div style={{ height: `100%` }} />}
/>
```

Programski kod 5.10. Prikaz API ključa

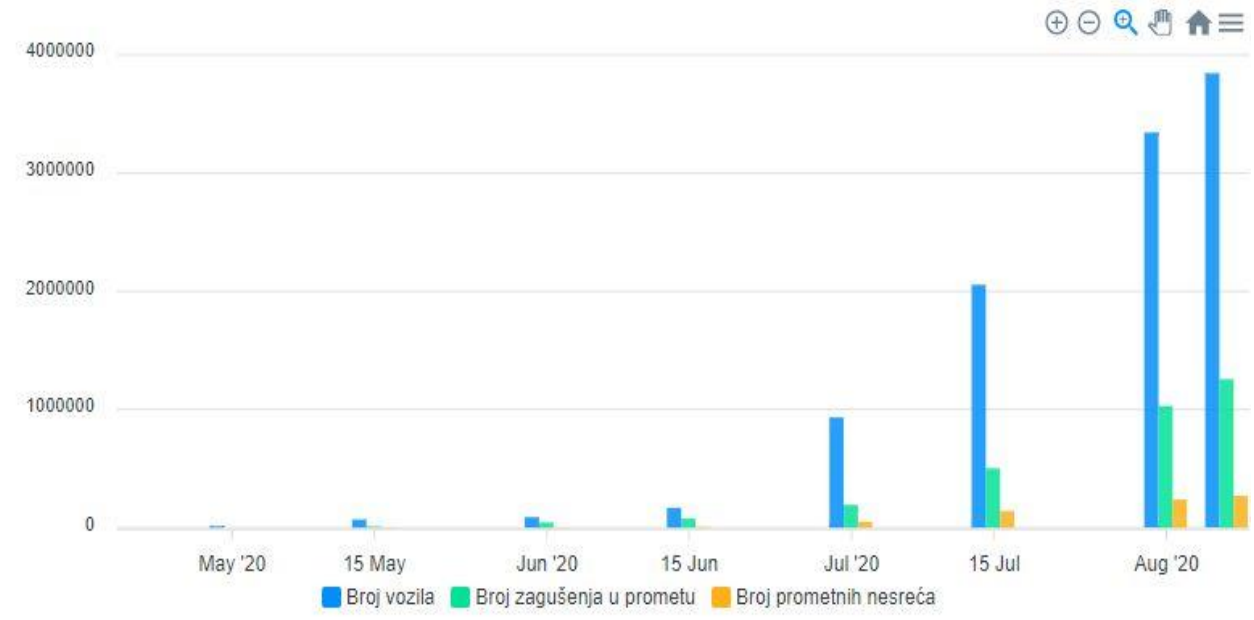
```
src > components > pages > traffic > JS Traffic.js > ...
1   import React from "react";
2
3   import Maps from "./Maps.js";
4   import Chart from "./Chart.js";
5   import Graph from "./Graph.js";
6
7   const Traffic = () => {
8     return (
9       <div>
10        <div>
11          <Maps style={{ margin: "500px 500px" }} />
12        </div>
13        <div>
14          <Chart />
15        </div>
16        <div>
17          <Graph />
18        </div>
19      </div>
20    );
21  };
22
23  export default Traffic;
24
```

Programski kod 5.11. Prikaz komponente za prikazivanje statistika odabrane rute (“Traffic.js”)

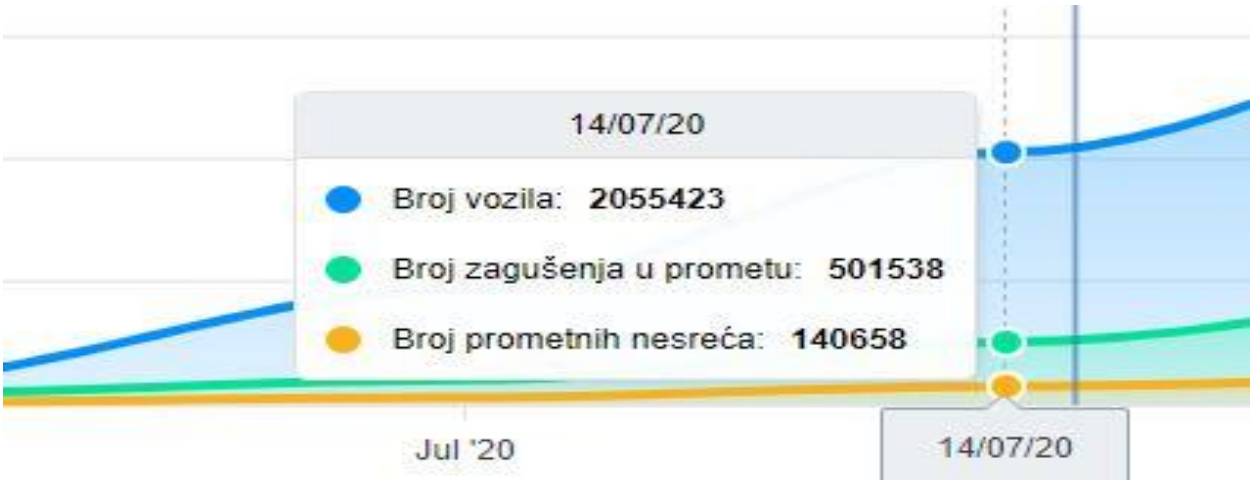
Statistike za rutu su predstavljene u vremenskom razdoblju od petog pa do osmog mjeseca. Na prikazu se točno može vidjeti kako se u vrijeme sezone povećava broj vozila u prometu. Naime, osim gužvi i zagušenja na prometnicama jako je bitan podatak i o prometnim nesrećama. Korisnika zanima kakvo je stanje na cestama te koji su mjeseci najsigurniji za putovanje. Graf koji aplikacija generira upravo to i prikazuje te se može vidjeti na slici 5.11. Na grafu se točno može vidjeti kako se broj zagušenja u prometu, broj prometnih nesreća, pa samim time i broj vozila povećavao s povećanjem mjeseca jer kad je graf krenuo rasti u tom trenutku je krenula sezona ljetovanja, pa se povećao i broj svih navedenih parametara. Navedena statistika je prikazana i na drugi način s drugim grafom što se može vidjeti na slici 5.12. Ako se postavi miš ne bilo koju točku od bilo kojeg grafa, prikazat će se mali prozor koji će prikazati brojke za točno tu točku koju je korisnik odabrao (Slika 5.13.). Plava linija na grafu prikazuje broj vozila, zelena linija prikazuje broj zagušenja u prometu te narančasta linija prikazuje broj prometnih nesreća.



Slika 5.11. Graf statistike na relaciji Osijek - Split



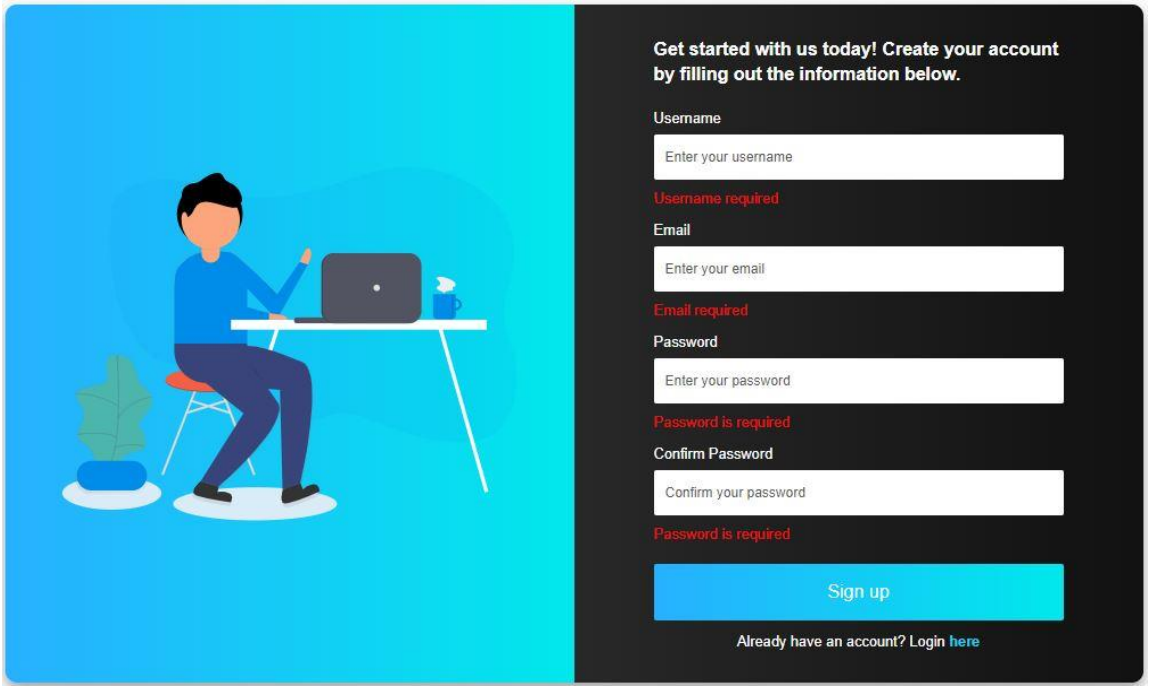
Slika 5.12. Drugi način prikaza grafa sa slike 5.11.



Slika 5.13. Prikaz statistike u točno jednoj točki

Ako korisnik ima nekakvih komentara ili prijedloga, može ih napisati u samoj aplikaciji. Naime, aplikacija ima mogućnost maloga bloga gdje korisnik može ostaviti komentar, ažurirati ga ili ga obrisati. Da bi dobio mogućnost postavljanja komentara mora napraviti korisnički račun. Korisnik

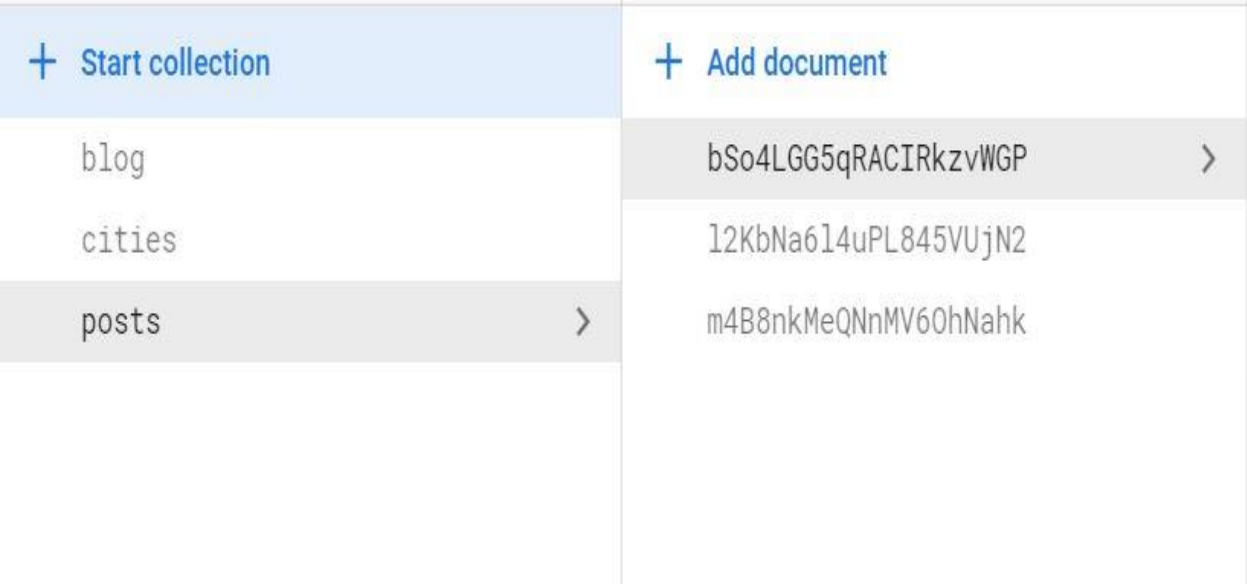
mora unijeti ime kao prvi parametar, nakon toga mora unijeti svoju e-mail adresu te šifru koju mora opet potvrditi. Izrada korisničkog računa odrađena od strane Firebase tehnologije koja se veže na Redux tehnologiju na način da se svi podaci od korisnika skupe u centralno stanje aplikacije te nakon toga da se pošalju u bazu podataka koja je podržana od strane Firebasea. U reduktorima se nalaze stanja o korisniku te stanja o greškama, jer ako dođe do greške na server ili ljudske greške da se poruka ispiše na sučelju. Akcije primaju podatak o tipu koji će obavljati, primjerice "SIGN_UP" što predstavlja tip izrade korisničkog računa. Nakon što korisnik unese valjane podatke akcija tipa "SIGN_UP" će se okinuti te će utjecati na reduktor koji će promijeniti centralno stanje aplikacije. Kada se centralno stanje aplikacije popuni sa informacijama o novome korisniku, Firebase funkcije će poslati te podatke u Firebase okolinu, obraditi ih te ih spremiti u bazu podataka. Kada korisnik ponovo posjeti stranicu, a pritom ima već izrađen račun moći će se samo prijaviti. U ovom primjeru će Firebase poslati podatke o korisniku centralnom stanju aplikacije koje će provjeriti odgovaraju li unesene informacije korisniku koji se nalaze u bazi te ako odgovaraju korisnik će se uspješno prijaviti. Ukoliko korisnik unese pogrešne podatke centralno stanje aplikacije će prepoznati grešku te vratiti poruku kojom obavještava korisnika da je upisao netočne podatke.



Slika 5.14. Prikaz validacijske poruke u slučaju greške pri izradi korisničkog računa

Prikaz poruke koju će aplikacija vratiti u slučaju pogreške prilikom unosa korisnikovih podataka nalazi se na slici 5.14. Kako bi se predstavila bolja slika o tome gdje odu podaci te kako oni izgledaju u bazi najbolje je vidjeti sam prikaz baze podataka u Firebaseu, što se može vidjeti na slici 5.15. Naime, baza podataka, prije nego što spremi podatke o korisniku, mora zaštititi njegove podatke. Iz tog razloga Firebase tehnologijama uz pomoć algoritama za šifriranja i kriptografiju pretvara šifru korisnika u poseban zapis koji ljudskom oku nije razumljiv. Kaže se da algoritmi šifriraju zaporku, te ju onda Firebase sprema u bazu podataka. Korisnikova šifra više nije u zapisu koji je korisnik upisao, već u posebnim patternima koje razumije samo kod. Prikaz korisnikovih podataka u bazi se može vidjeti na slici 5.16.

Slika 5.15. Prikaz podataka korisnika u bazi podataka, Firebase



Slika 5.16. Prikaz komentara u bazi podataka u Firebaseu

Kod za cijelu autentifikaciju korisnika se nalazi u mapi auth te sadrži datoteke “FormSignup.js”, “FormLogin.js” i “validateInfo.js” koje sadrže logiku za autentifikaciju korisnika. Također, su u Redux mehanizmu postavljene akcije i reduktori za iste ove funkcije.

Jednom kada je korisnik napravio korisnički račun, u navigacijskoj traci će se pojaviti nova kartica s nazivom Posts. Kada korisnik klikne na navedenu rutu, ona će ga odvesti na stranicu za kreiranje komentara. Također, stari napravljeni komentari bit će prikazani na ovoj ruti. Na lijevoj strani sučelja se nalazi mala ikona u obliku knjižice. Kada korisnik klikne na nju, otvorit će se modal u kojem se nalaze polja za naslov komentara, sadržaj komentara te gumb koji služi za kreiranje novog komentara. Na kartici koja prikazuje komentar nalazi se ikona kante za smeće. Ukoliko korisnik klikne na tu ikonu obrisat će komentar. Komponenta koja je zadužena za cijelu logiku bloga nalazi se u mapi posts. Redux i Firebase su također povezani sa kreiranjem komentara na isti način kao i kod autentifikacije korisnika. Firebase je povezan s projektom putem API ključa, te se konfiguracija nalazi u datoteci config.js. Naime u ovoj datoteci se nalazi mehanizam za povezivanje projekta s Firebase mehanizmom baze podataka te svih mogućnosti, funkcija i funkcionalnosti koju pruža Firebase. Konfiguracija Firebase tehnologije se može vidjeti na programskom kodu 5.12.

src > firebase > JS Config.js > ...

```
1   import * as firebase from "firebase/app";
2   import "firebase/firestore";
3   import "firebase/auth";
4   import "firebase/storage";
5   import "firebase/database";
6
7   export const firebaseConfig = {
8     apiKey: "AIzaSyAYxpRmNZirNG_EsQzTYFwUi4KjhVrIH2Q",
9     authDomain: "vehicle-application-6b264.firebaseapp.com",
10    databaseURL: "https://vehicle-application-6b264.firebaseio.com",
11    projectId: "vehicle-application-6b264",
12    storageBucket: "vehicle-application-6b264.appspot.com",
13    messagingSenderId: "422603042964",
14    appId: "1:422603042964:web:dc23e2dbe3b4f330d48481",
15    measurementId: "G-E6GXR3EWR6",
16  };
17
18  firebase.initializeApp(firebaseConfig);
19
20  const firebaseWithConfig = !firebase.apps.length
21    ? firebase.initializeApp(firebaseConfig)
22    : firebase.app();
23
24  export const FirebaseAuth = firebaseWithConfig.auth();
25  export const FirebaseDatabase = firebaseWithConfig.firestore();
26
```

Programski kod 5.12. Konfiguracija Firebasea

6. ZAKLJUČAK

S napretkom tehnologije jako puno stvari u današnjem svijetu se poboljšava i napreduje. Autonomnom vožnjom će se postići nova dimenzija tehnologije. Smanjit će se broj prometnih nesreća i nezgoda jer jedina stvar koju tehnologija ne može kontrolirati je ljudsko biće. Također, će se povećati iskorištenost vozila, transportna vozila neće kasniti te neće biti potrebno da ljudi prelaze velike kilometraže. Automobili će biti sposobni komunicirati jedni s drugima te će se moći puniti prilikom vožnje bez stajanja na benzinske postaje. Korištenje automobila će biti kao korištenje taksija, samo što će se moći posjedovati automobile. Naime, kada je osobi potrebna vožnja, vozilo će samo doći po osobu te ju odvesti gdje treba, nakon toga vozilo će ići po drugu osobu kojoj je potrebna vožnja i tako dalje. Na taj način će se u potpunosti iskoristiti automobile tijekom cijelog njegovog životnog vijeka. Također, će se emisije štetnih plinova i zagađenje okoliša smanjiti, jer će se postupno izbacivati iz svakodnevne upotrebe motori s unutarnjim izgaranjem. Iako je električni motor manje štetan od motora s unutarnjim izgaranjem, nije ni električni automobil potpuno ekološki prihvatljiv. Električni automobil mora sadržavati bateriju, koja prilikom svoje proizvodnje zagađuje okoliš.

Cilj praktičnog rada je olakšati prijevoz putem aplikacije gdje će korisnik moći vidjeti statistike na određenim dionicama. Također, ima priliku da izradi svoj vlastiti korisnički račun. Nakon što korisnik napravi svoj račun moći će ostavljati svoje komentare, brisati ih i uređivati ih ako je to potrebno. Imat će i prikaz o vozilima na način da svaka stranica predstavlja jednu marku vozila te će na svakoj stranici imati uvid u specifikacije određenog vozila.

LITERATURA

[1] Ben Johnson, “The Great Horse Manure Crisis of 1894”, dostupno na: https://www.historic-uk.com/HistoryUK/HistoryofBritain/Great-Horse-Manure-Crisis-of-1894

[2] C. K. Toh, J. A. Sanguesa, J.C. Cano, F.J. Martinez, „Advances in smart roads for future smart cities“, Proceedings of the Royal Society A, 2020.

[3] N. Bolf, “Mjerna i regulacijska tehnika”

[4] H. Wang, A. Jasim, X. Chen, „Energy harvesting technologies in roadway and bridge for different applications - A comprehensive review“, Applied energy, Vol. 212, str. od 1083 do 1094

[5] C.K. Toh, J.C. Cano, C. Fernandez-Laguia, P. Manzoni, C.T. Calafate, „Wireless digital traffic signs of the future“, IET Networks

[6] Zhao, H. & Wu, D. (2015). Definition, Function, and Framework Construction of a Smart Road. Dostupno na: https://www.researchgate.net/publication/287725459_Definition_Function_and_Framework_Construction_of_a_Smart_Road (Srpanj 2021)

[7] S. Hasanagić, Završni rad, “Tehnologije internet stvari”

[8] S. Venkitesh, Case Study: Traffic and Road Conditions Monitoring in Malaga, June 8, 2016,

[9] Dostupno na: https://menafn.com/1099754118/Smart-Roads-Market-2020-Emerging-Trends-and-Technology-Advancement-Indra-Sistemas-Cisco-Systems-Alcatel-Lucent-IBM-Siemens, (Srpanj 2021)

[10] Penttinen, J. (2019). *5G Explained*. Wiley

[11] Hartenstein, H.; Laberteaux, K. (2010). VANET: Vehicular Applications and Inter-Networking Technologies

[12] Vehicle to Everything (V2X), https://corporatefinanceinstitute.com/resources/knowledge/other/vehicle-to-everything-v2x (Srpanj 2021)

[13] J. Ahn, Y. Y. Kim, R. Y. Kim, A Novel WLAN Vehicle-To-Anything (V2X) Channel Access Scheme for IEEE 802.11p-Based Next-Generation Connected Car Networks

[14] H. Gabelica, Prediktivna analitika i strojno učenje: mogu li se strojevi učiti

[15] React official page, dostupno na: https://reactjs.org/tutorial/ (Kolovoz 2021)

[16] J. Blia, Lifting state up and passing it to another child component in React: dostupno na:

https://jawblia.medium.com/lifting-state-up-and-passing-it-to-another-child-component-in-react-15d7036f1b40 (Kolovoz 2021)

[17] ISO/IEC Celestial Systems, 2013, Redux Fundamentals: Building blocks and data flow (Rujan 2021)

[18] S. Rose, Initializing Firebase, May 6, 2019 (Rujan 2021)

[19] Službena stranica tvrtke Audi, dostupno na: https://www.audiusa.com/us/web/en.html/, (Rujan 2021)

SAŽETAK

Ovaj rad prikazuje nove tehnologije koje mogu zauvijek promijeniti današnje prometnice. Pametna cesta je zadnji korak prije nego se krene u potpunu autonomizaciju vozila nakon čega slijedi izbacivanje vozača iz prometa. Naime, vozilo će biti dovoljno pametno da obavi sve funkcije koje se od njega očekuju. Također, vozilo će se moći puniti prilikom vožnje uz pomoć ceste. Cesta će moći komunicirati s drugim cestama, prometnim znakovima i drugim vozilima koja se nalaze na cestama. Uvođenjem pametnih cesta će se iskoristiti i obnovljivi izvori energije za generiranje energije koja je potrebna vozilima da se napune. Na taj način će se povećati iskoristivost zemlje te smanjiti štetnost, što ukazuje na ekološku osviještenost. Negativna strana pametnih cesta je što će iziskivati puno novaca, u svakom slučaju više nego kod do sadašnje ceste.

Praktični dio rada je osmišljen da prikaže osobi koja se služi aplikacijom prometne statistike na određenim dionicama te da korisnik može ostaviti svoj komentar te na taj način pomoći ostalim korisnicima. Također, aplikacija ima dosta zanimljivih sadržaja koje može koristiti osoba koja se služi aplikacijom.

Ključne riječi: autonomizacija vozila, aplikacija, pametna cesta

ABSTRACT

Title: Smart road technologies and innovative transport infrastructure

This paper presents new technologies that can forever change today's roads. The smart road is last step before the vehicle become fully autonomus, followed by the removal of the driver from traffic. Furthermore, the vehicle will be smart enough to perform all the functions expected of it. Also, the vehicle will be able to be charged while driving with help of the smart road. The road will be able to communicate with other roads, traffic signs and ther vehicles on the roads. The introfuction of smart roads will also use renewable energy sources to generate the energy needed by vehicles to recharge. In this way, the usability of the land will increase and the harmfulness will decrease, which indicates environmental awareness. The downside od smart roads is that they will require a lot og money, in any case more than with the current road.

The practical part of the paper is designed to show the person who uses the application of traffic statistics on certain sections and that the user can leave a comment and thus help other users. Also, the application has a lot of interesting content that can be used by the person who uses the application.

Key words: vehicle autonomy, application, smart road

ŽIVOTOPIS

Filip Pirić, rođen u Osijeku 03.08.1996. godine. Završava osnovnu školu “Mladost” u Osijeku te 2011.godine upisuje “2. Gimnaziju” u Osijek. Godine 2015. upisuje Fakultet Elektrotehnike, Računarstva i Informacijskih tehnologija. Od 1.rujna do 1.prosinca 2019.godine radi kao praktikant u Zavodu za urbanizam i izgradnju d.o.o. Zapošljava se kao programer, 01.03.2021. u firmi “Agency04” te tamo radi do danas.

Filip Pirić