SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 729

MOBILNA APLIKACIJA ZA PRAĆENJE SUBJEKTIVNOG DOŽIVLJAJA UDOBNOSTI VOŽNJE

Marko Dodik

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 729

MOBILNA APLIKACIJA ZA PRAĆENJE SUBJEKTIVNOG DOŽIVLJAJA UDOBNOSTI VOŽNJE

Marko Dodik

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 11. ožujka 2022.

ZAVRŠNI ZADATAK br. 729

Pristupnik:	**Marko Dodik (0036517897)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	doc. dr. sc. Jurica Babić

Zadatak: **Mobilna aplikacija za praćenje subjektivnog doživljaja udobnosti vožnje**

Opis zadatka:

Razvoj tehnologije, promjene u potrebama tržišta i ekološki zahtjevi potiču različite sektore, uključujući i transportni sektor, na stalni napredak. Istraživanjem i analizom podataka u transportnom sektoru moguće je doći do novih saznanja te unaprijediti mobilnost korisnika. Kako bi se podaci mogli analizirati potrebno ih je prethodno prikupiti na odgovarajući način. Vaš zadatak je oblikovati, implementirati i dokumentirati programsko rješenje za praćenje subjektivnog doživljaja udobnosti vožnje. Navedeni podaci poslužit će za označavanje kontekstno obogaćenog skupa automobilskih podataka koji se istovremeno prikupljaju tijekom vožnji. Programsko rješenje treba izvesti kao aplikaciju za pokretne uređaje namijenjenu putniku koji tijekom vožnje bilježi predefinirana opažanja u stvarnom vremenu (npr. naglo kočenje, naglo ubrzanje, gust promet, neugodnost). Rješenje mora moći prikupiti podatke od strane putnika te ih pohraniti na udaljeni sustav za pohranu podataka. Eksperimentalnu evaluaciju rješenja potrebno je obaviti na uzorku od barem 10 vožnji. Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Rok za predaju rada: 10. lipnja 2022.

Zahvaljujem se mentorskom timu na pomoći i savjetima koje su mi pružili u izradi ovog rada. Želim se također zahvaliti obitelji i prijateljima koji su pomagali i pružali podršku kroz studij.

SADRŽAJ

1. Uvod

Jedan od velikih problema današnjice jest emisija stakleničkih plinova kojoj u velikoj mjeri pridonosi i sam razvoj industrije transporta (1). Od početka industrijske revolucije, izgaranje fosilnih goriva doprinijelo je povećanju ugljikovog dioksida u atmosferi od 40%, slijedno tome i povećanju efekta staklenika, to jest dodatnog zagrijavanja Zemljine površine i donjih slojeva Zemljine atmosfere selektivnim propuštanjem toplinskog zračenja (2)(3). Jedan od prvih koji je primijetio učinak staklenika bio je Joseph Fourier, još 1820. godine kada je ujedno i počelo izučavanje tog efekta na našu planetu. Uviđanjem sve većeg problema koji efekt staklenika stvara, ljudi se suočavaju s njim potpisivanjem protokola iz Kyota. Protokol su do 2020. godine potpisale 192 države s ciljem smanjivanja ugljikovog dioksida i drugih stakleničkih plinova (4).

U industriji transporta, uz sve veći razvoj tehnologije, prelazi se na električna vozila te vozila s manjom potrošnjom, više se ulaže u dizajn i udobnost u vozilu. Uvode se strože sankcije za vozila s većom koncentracijom ispušnih plinova te sankcije za starija vozila. Dodatno, prikupljanjem ključnih podataka s vozila i okoline moguće je bolje razumjeti koji faktori utječu na emisiju stakleničkih plinova, potrošnju goriva te sigurnost i putnikov doživljaj vožnje. Automobilska industrija sve se više približava svojem vrhuncu u performansama, stoga vozač želi za sebe i za ostale putnike brzu, ali i sigurnu vožnju.

Izazov koji se danas javlja su preferencije pojedinih putnika u vozilu. Na primjer, neki putnici preferiraju sporiju vožnju, dok drugi radije odabiru vožnju s više ubrzanja. Kao rješenje za prikupljanje podataka o preferencijama putnika, u ovom se radu predlaže aplikacija za pokretne uređaje. Aplikacija je namijenjena putniku koji tijekom vožnje bilježi unaprijed definirane događaje u stvarnom vremenu. Točnije, putnik će uslijed specifičnog događaja tijekom vožnje odmah isti zabilježiti putem aplikacije. Podaci zabilježeni u vožnji spremaju se na udaljenu bazu podataka, kako bi im se kasnije moglo pristupiti za daljnju analizu. Aplikacija ima jednostavno sučelje čime se olakšava korištenje.

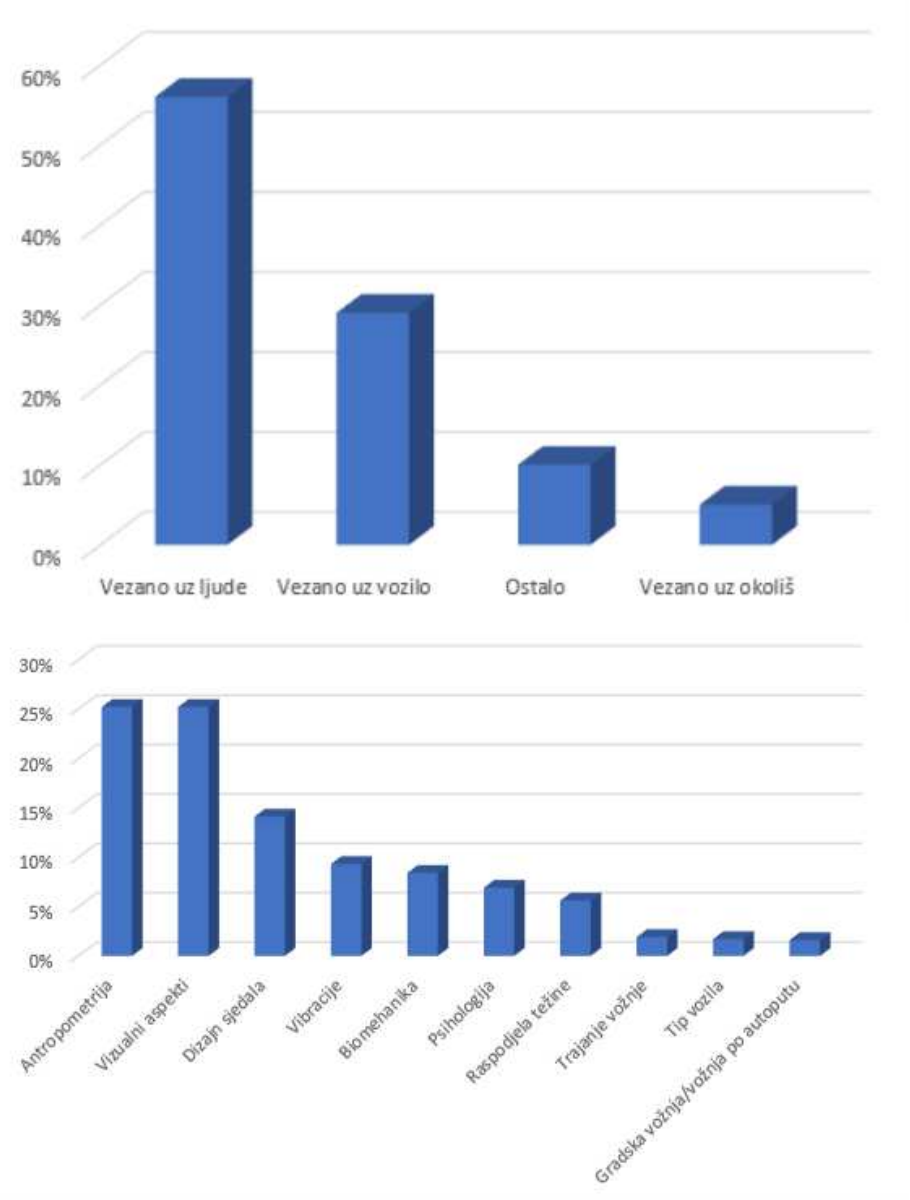
2. Pregled postojećih rješenja

U ovom se poglavlju daje pregled triju povezanih rješenja sa sličnim problemima s kojima se suočavamo u razvijanju programske potpore.

Aris i suradnici (5), u svom radu objašnjavaju kako i na koji način prikupljaju podatke iz automobila preko OBD-II (eng. *On-board diagnostics*). Podaci koje prikupljaju sadrže općenite informacije o automobilu, brzini i broj okretaja s vremenskom oznakom, lokaciji automobila, log datoteke s podacima ukoliko dođe do nesreće te podsjetnike za održavanje automobila. Podaci se ažuriraju svakih 5 sekundi. Kao rezultat istraživanja dostupan je veliki skup podataka koji se može dalje analizirati.

Rimpas i suradnici (6), proučavaju korisnost potrošnje automobila s obzirom na vrstu automobilske vožnje. Korištenjem OBD-II prikupljaju informacije o automobilu, njegovoj brzini, broju okretaja te geolokaciji. Prikupljaju i podatke o težini svakog vozila koje sudjeluje u istraživanju. Podaci su korišteni za proučavanje efikasnosti potrošnje za različiti broj okretaja i različite brzine automobila. Zaključak istraživanja jest da su automobili najučinkovitiji u rasponu broja okretaja između 2200-2500 okretaja u minuti te bez dodatnog pritiskanja papučice gasa. U tim uvjetima automobil gotovo da i ne koristi dodatan benzin. Kada je u pitanju brzina, vožnja automobilom na autocesti se pokazala najučinkovitijom. U gradskim vožnjama se to prolongira kroz smanjenu učinkovitost zbog učestalog stajanja i ponovnog kretanja. Naposljetku, veliku ulogu ima i sam vozač te način vožnje. Ako vozač ima oscilacije u vožnji, odnosno česte izmjene gasa te kočenja, potrošnja goriva se povećava, a učinkovitost smanjuje.

Rajhans i suradnici (7), proučavaju i definiraju faktore koji čine vožnju udobnijom te sigurnijom. Navedenim istraživanjem su sumirali postojeće znanje iz dostupne literature, tj. knjiga i objavljenih istraživačkih radova. Rezultat istraživanja prikazan je dijagramom na Slici 2.1.



Slika 2.1: Prikaz faktora udobnosti prilikom vožnje

3. Model i arhitektura sustava

3.1. Specifikacija aplikacije

Životni ciklus programske potpore možemo promatrati kroz četiri veće aktivnosti:

- **specifikacija programske potpore**,
- **oblikovanje i implementacija**,
- **validacija i verifikacija**,
- **evaluacija programske potpore**.

U prvoj aktivnosti, specifikaciji programske potpore, treba odrediti funkcionalnosti te ograničenja sustava u ovisnosti o zahtjevima. Postupak kojim će se to odrediti jest proces inženjerstva zahtjeva (eng. *requirements engineering*). Zadatak procesa jest određivanje funkcionalnosti te ograničenja na sustav temeljem analize zahtjeva.

Aplikacija koju je potrebno stvoriti mora biti intuitivna, laka za korištenje, ali je uz to od iznimne važnosti i sam dizajn aplikacije koji će klijenti prvo primijetiti. Kao unikatni identifikator vožnje, koristit će se e-mail korisnika stoga je potrebno implementirati funkcionalnost unosa e-mail adrese. Implementacija pristupa upitnicima koji su predviđeni za nove putnike i vozače. Potrebno je implementirati konekciju s bazom podataka te kreiranje objekta koji sadrži: vrijeme kreiranja, unikatni identifikator vožnje i poruku, koji će se potom proslijediti bazi podataka. Također, nakon same vožnje potrebno je omogućiti pristup predviđenim upitnicima.

3.2. Funkcionalni zahtjevi i dijagrami obrazaca uporabe

3.2.1. Funkcionalni zahtjevi

Funkcionalni zahtjevi aplikacije opisuju koje bi funkcije sustav trebao obavljati i kako će se on ponašati. Implementiraju se još u fazi projektiranja sustava.

Akteri sustava su:

- **Vozač**
- **Putnik**
- **Stvoritelj sustava**

Vozač ima mogućnosti:

- unijeti svoju e-mail adresu
- ispuniti upitnik o informacijama o vozaču
- ispuniti upitnik o doživljaju vožnje

Putnik ima mogućnosti:

- unijeti svoju e-mail adresu
- ispuniti upitnik o informacijama o putniku
- ispuniti upitnik o doživljaju vožnje

Stvoritelj sustava ima:

- punu kontrolu nad korisnikom
- punu kontrolu nad bazom podataka

Baza podataka ima mogućnosti napraviti:

- spremanje podataka o vožnjama

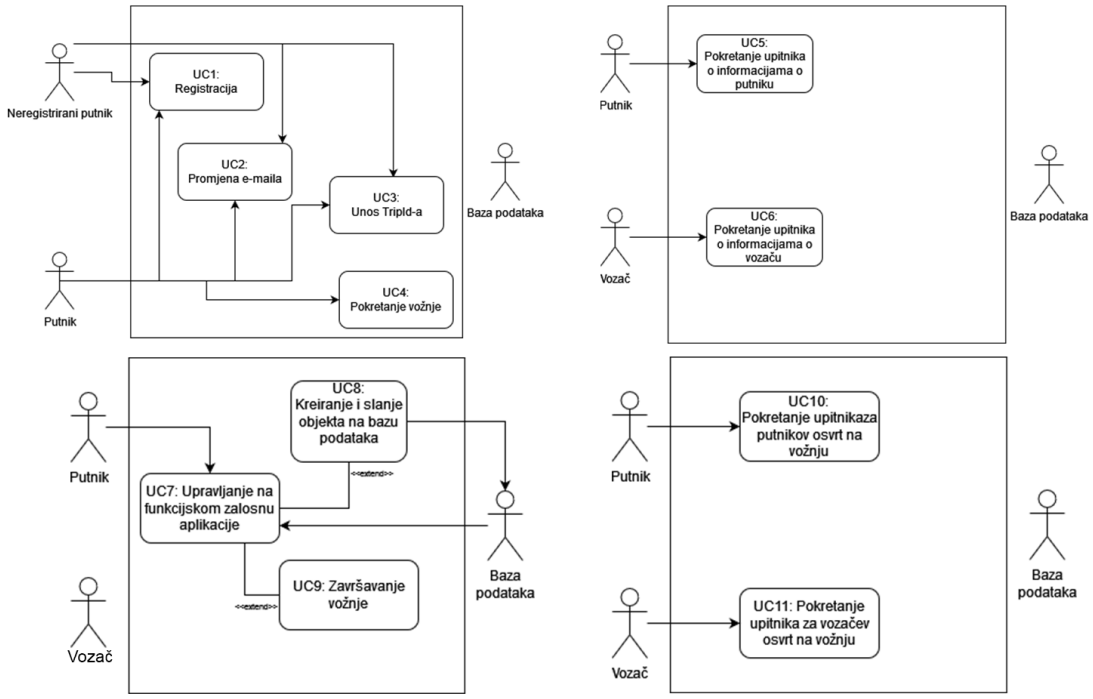
3.2.2. Dijagrami obrazaca uporabe

Na slici 3.1 prikazujemo četiri UML dijegrama sa slučajevima upotrebe (eng *use cases*). Svaki dijagram se sastoji od glumaca (eng. *actors*), sustava (eng. *system*) i ciljeva (eng. *goal*).

U prvom UML dijagramu glumci su: neregistrirani putnik, putnik te baza podataka. Neregistrirani korisnik ima mogućnosti se registrirati unosom e-mail adrese(UC1), može tu e-mail adresu promjeniti(UC2) te unijeti *TripId*(UC3). Unosom e-mail adrese on postaje registriranim putnik, u daljnjem tekstu putnik. Putnik također može se ponovno registrirati(UC1), promjeniti e-mail(UC2), unijeti *TripId*(UC3), ali može i započeti vožnju(UC4).

Drugi UML dijagram prikazuje glumce putnika i vozača. Putnik može pokrenuti upitnik "Informacijama o putniku"(UC5), a vozač upitnik "Informacijame o vozaču"(UC6).

Treći UML dijagram prikazuje putnika kada pokrene vožnju. U tom trenutku putnik upravlja funkcijskim zaslonom aplikacije(UC7), tamo se kreiraju i šalju objekti na bazu podataka(UC8) koja sprema podatke. Putnik na kraju vožnje može završiti samu vožnju(UC9).



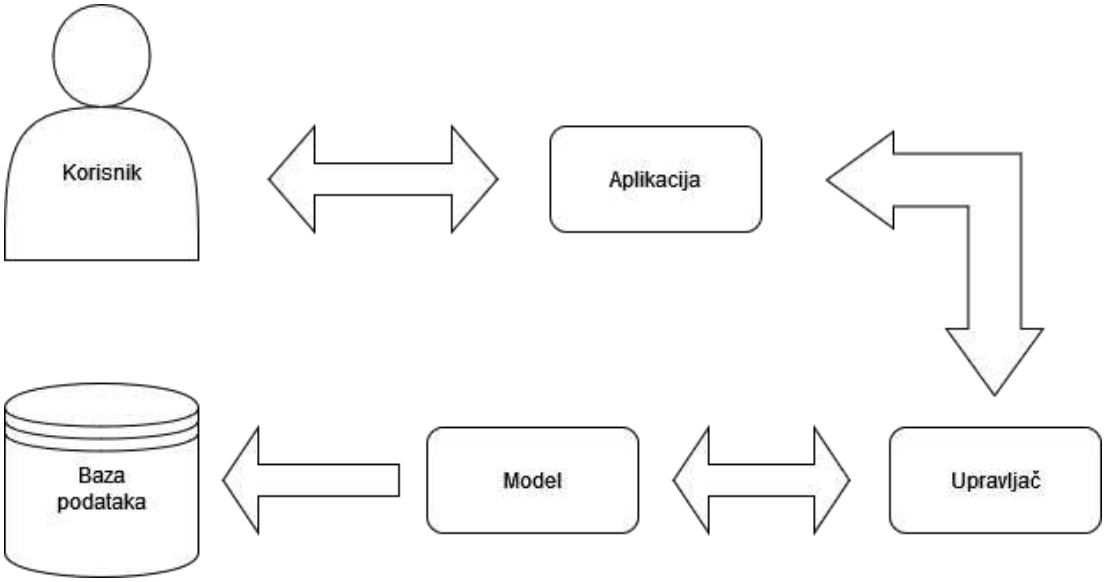
Slika 3.1: UML dijagram obrazaca uporabe

Posljednji, četvrti, UML dijagram prikazuje putnika koji može pokrenuti upitnik "Putnikov osvrt na vožnju"(UC10) te vozača koji može pokrenuti upitnik "Vozačev osvrt na vožnju"(UC11).

3.3. Arhitektura sustava

Obrazac softverske arhitekture koji je korišten za izradu aplikacije jest *Model-View-Controller (MVC)*. Koristi se za odvajanje aplikacije u tri cjeline: *Model* jest logika i svi podaci aplikacije, također sadrži sve veze između korisnika i baze podataka. *View* jest prikaz aplikacije korisniku. *Controller* izvršava zahtjeve korisnika.

Na slici 3.2 vidi se da korisnik preko aplikacije (*view*), koja izvršava zahtjeve preko upravljača (*controller*), uspostavlja komunikaciju s modelom koji ima direktnu poveznicu s bazom podataka.



Slika 3.2: Obrazac softverske arhitekture

4. Implementacija programskog rješenja

4.1. Korištene tehnologije

Flutter jest radni okvir za razvoj programske potpore otvorenog koda kojega je razvila tvrtka Google. Koristi se za izradu *cross-platform* aplikacija (8). Podržani operacijski sustavi su Linux, Android, Windows, Google Fuchsia, Mac te iOS. Objavljen je 2017. godine. Tri velike prednosti Fluttera nad drugim radnim okvirima su: brzina, produktivnost te fleksibilnost. Ono što čini Flutter bržim od ostalih radnih okvira jest samostalno kompajliranje koda u ARM ili Intel strojni kod ili JavaScript. Produktivnost se prikazuje upravo kod kreiranja aplikacije gdje se koristi dinamičko prevođenje (eng. *just-in-time compile*) koje prikazuje sve promjene u aplikaciji bez ponovnog pokretanja iste. Ponovnim pokretanjem aplikacije (eng. *Hot Reload*) prikazuje se simulacija aplikacije na uređaju s promjenama napravljenima u kodu. Fleksibilnost proizlazi upravo iz velikog broja podržanih operacijskih sustava gdje se vrlo lako može upravljati dizajnom i funkcionalnostima, neovisno o samom operacijskom sustavu. Flutter aplikacije pisane su u Dart programskom jeziku koji je osmišljen za izradu klijentske strane aplikacije(9).

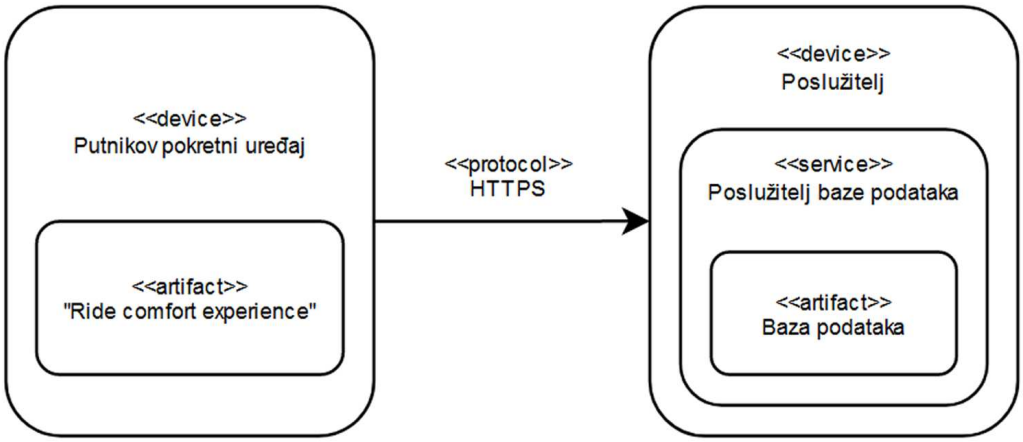
MongoDB jest *cross-platform* program za baze podataka koji se koristi za pohranu, dohvaćanje i upravljanje polustrukturiranim podacima(10). Tipovi polustrukturiranih podataka su JSON i XML oblik. Prednost ovakvog NoSQL tipa baze podataka jest što su programeri slobodniji u spremanju podataka u samu bazu jer se tipovi podataka ne moraju poklapati, veze između entiteta su jednostavnije te su sami modeli podataka pojednostavljeni zbog relacijskih modela. Ovakav tip baze podataka je najbolje koristiti: kod baza koje imaju velika opterećenja s unosom podataka, baza s nestabilnim shemama, kada podaci koji se spremaju u bazu sadrže lokaciju (eng. *location-based data*), kod baza koje zauzimaju mnogo prostora, baza gdje je potrebna velika dostupnost u nestabilnom okruženju te baza kod kojih ne postoji administrator.

Latex jest programski jezik za pisanje strukturiranih tekstova. LaTeX je napisao

Amerikanac Leslie Lamport ranih 1980-ih godina (11). Prednost nad ostalim uređivačima teksta je veliki broj dodatnih paketa za klase dokumenata, posebno za oblikovanje te podršku za jezike. Najveća mana jest upravo kompleksnost za nove korisnike u odnosu na druge uređivače tekstova.

4.2. Arhitektura implementiranog sustava

Arhitektura implementiranog sustava prikazuje se preko UML dijagrama razmještaja (slika 4.1). Dva uređaja na slici su Pokretni uređaj koji koristi putnik te Poslužitelj. Na putnikovom pokretnom uređaju je instalirana aplikacija "Ride comfort experience" koja komunicira zajedno s uređajem, preko HTTPS protokola s Poslužiteljem. Poslužitelj kroz zahtjeve zapisane u *header* dijelu protokola HTTPS preusmjerava na poslužitelja baze podataka, a poslužitelj baze podataka do baze podataka.



Slika 4.1: UML dijagram razmještaja

5. Rezultat i diskusija

5.1. Dizajn aplikacije

5.1.1. Paleta boja

Osnovna paleta boja korištena u aplikaciji jest prikazana na slici 5.1 s heksadekadskim kodom svake od boja. Za dvije temeljne boje u aplikaciji izabrane su tamno siva te narančasta. Narančasta boja je aktivna i energična boja. Izaziva povjerenje, radost i zadovoljstvo. Tople boje (raspon boja između crvene i žute, a uključuju narančastu, ružičastu, smeđu i bordo), predstavljaju toplinu i kretanje. U dizajnu kada se postavi pored hladnih boja(sve nijanse sive), topla boja će se isticati i dominirati.

Slika 5.1: Paleta boja

5.1.2. Vizualni identitet

Vizualni identitet aplikacije (slike 5.2) izabran i izrađen je tako da prikazuje dva glavna motiva koji se kriju iza njenog nastanka

- **vožnju** - prikazanu kroz ikonu automobila
- **udobnost** - prikazanu kroz ikonu srca

Slika 5.2: Vizualni identitet aplikacije

5.1.3. Naziv aplikacije

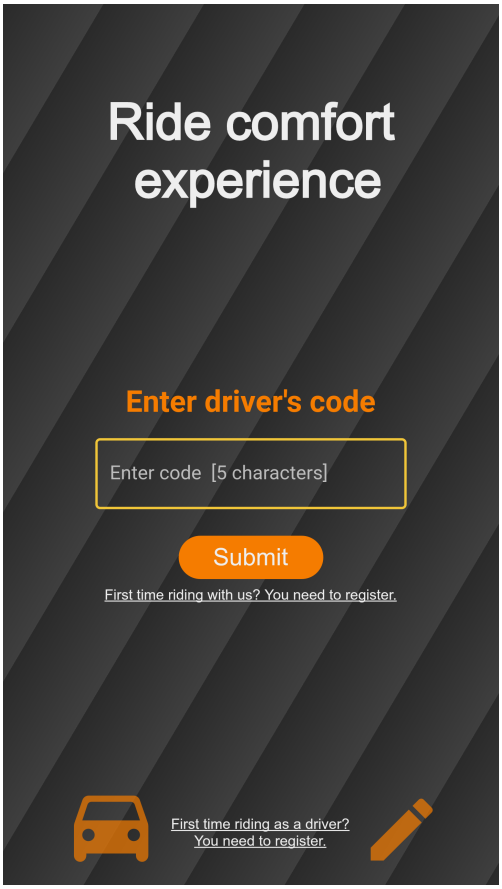
Naziv ove aplikacije je "Ride comfort experience". Odabran je kako bi korisnik iz naziva intuitivno mogao zaključiti čemu služi aplikacija.

5.2. Zasloni u aplikaciji

5.2.1. Početni zaslon

Aplikacija je kreirana za prikupljanje podataka o vožnji u stvarnom vremenu. Podaci se spremaju na udaljenu bazu podataka protokolom HTTPS. Putnik i vozač su predviđeni korisnici aplikacije. Putnik nakon unošenja jedinstvenog identifikatora vožnje (*TripId*), pokreće vožnju te može početi s prikupljanjem podataka. Prikupljeni podaci krucijalni su za daljnju analizu.

Pokretanjem aplikacije korisniku se prikazuje početni zaslon (slika 5.3). Na zaslonu korisnik ima jedno polje u koje treba unijeti *TripId* kojega mu dodjeljuje vozač iz "Automotive Data Collector" aplikacije. TripId jest jedinstveni identifikator svake vožnje.

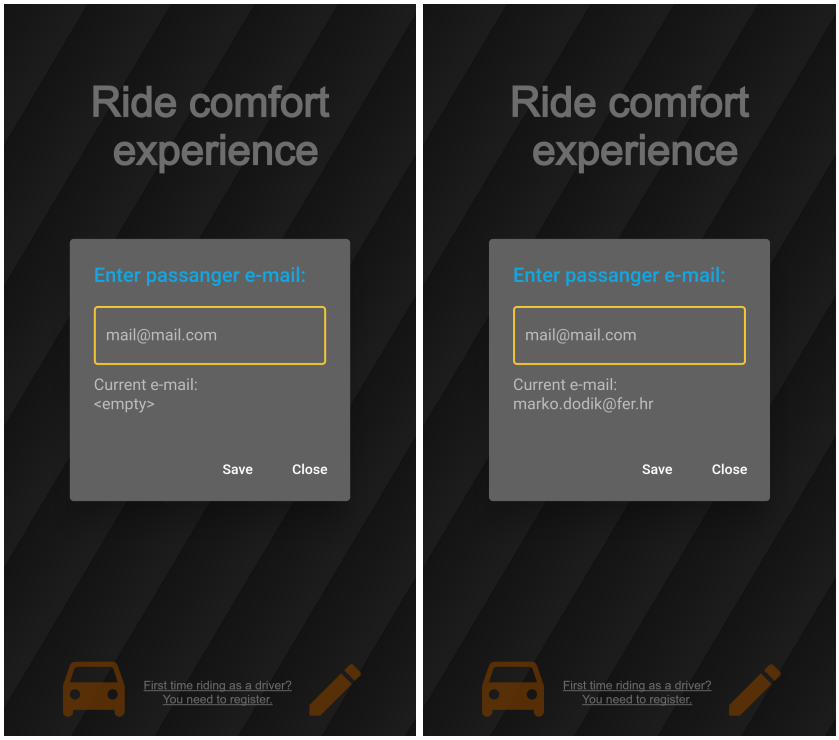


Slika 5.3: Početni zaslon aplikacije

Na početnom zaslonu korisnik ima pristup na dvije poveznice. Prvi link koji se nalazi ispod *Submit* gumba korisnika će odvesti na Googleov upitnik kojega **putnik** mora ispuniti prije svoje prve vožnje kao putnik u motornom vozilu. Drugi link nalazi

se na dnu zaslona pored ikone automobila te vodi na upitnik kojeg **vozač** treba ispuniti prije svoje prve vožnje u ulozi vozača.

Zbog lakšeg i spretnijeg korištenja aplikacije implementirana je funkcionalnost spremanja putnikovog računa elektroničke pošte *e-mail address*. Klinkom na ikonu olovke otvara se privremeni zaslon za promjenu računa. Na slici 5.4 prikazana su stanja kada putnik još nije unio svoj mail (slika lijevo) te kada putnik već ima spremljen mail (slika desno). Prednost ove implementacije jest automatsko ispunjavanje upitnika za putnika.

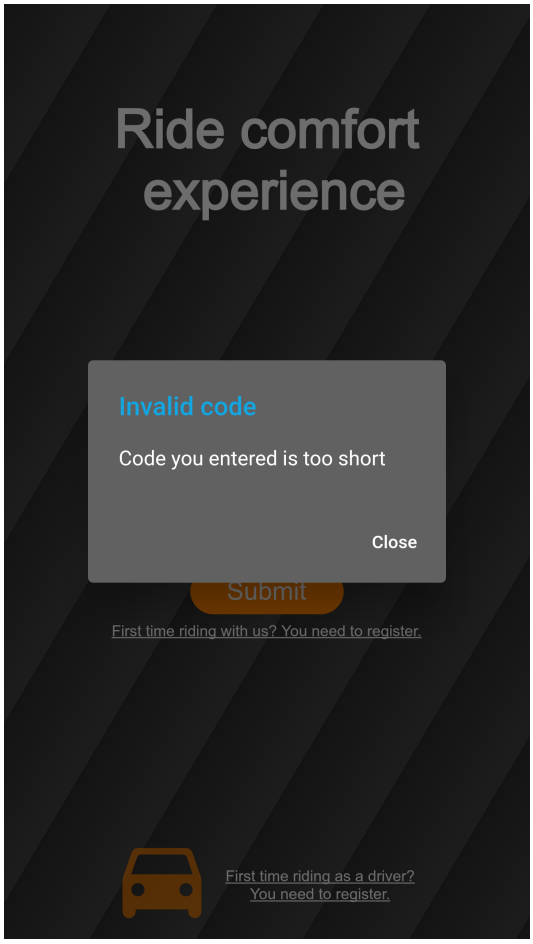


Slika 5.4: Privremeni zaslon za promjenu računa elektroničke pošte

Unese li putnik kod koji nije duljine točno pet znakova, to jest ukoliko putnik upiše manje od pet znakova (broj znakova koje korisnik može unijeti u polje je ograničen), dobit će obavijest u obliku vidljivom na slici 5.5, gdje ga aplikacija upozorava da je unio kod koji je prekratak te će korisnik nakon pritiska na *Close* gumb moći ponoviti postupak.

Ukoliko putnik unese kod koji je zadovoljio sve uvjete, ali je pogrešan, moći će se na svom pametnom uređaju vratiti na prethodni zaslon pritiskom na tipku *back*.

Unese li putnik ispravan kod, aplikacija će ga preusmjeriti na Funkcijski zaslon.



Slika 5.5: Krivi format *TripId*-a

5.2.2. Funkcijski zaslon

Na funkcijskom zaslonu (slika 5.6) korisniku se prikazuje *TripId* te devet gumba. Od devet gumba, njih osam namijenjeno je prikupljanju podataka (funkcijski gumbi) u stvarnom vremenu tijekom vožnje.

Pomno izabrani događaji su najčešći problemi i strahovi putnika prema istraživanju HAK-a (Hrvatski autoklub)(12) (13). Oni su:

- Naglo ubrzanje (engl. *Fast acceleration*)
- Naglo kočenje (engl. *Sudden braking*)
- Ne korištenje žmigavaca (engl. *Not using blinkers*)
- Prelaženje na crveno svijetlo (engl. *Crossing red light*)
- Ulaz u zavoj pri velikoj brzini (engl. *Turn at high speed*)
- Konstantno mijenjanje vozne trake (engl. *Car slalom*)

– Ne propuštanje pješaka na pješačkom prijelazu (engl. *Not letting pedestrian cross*)

– Vožnja s malim razmakom iza auta ispred (engl. *Close to car ahead*)

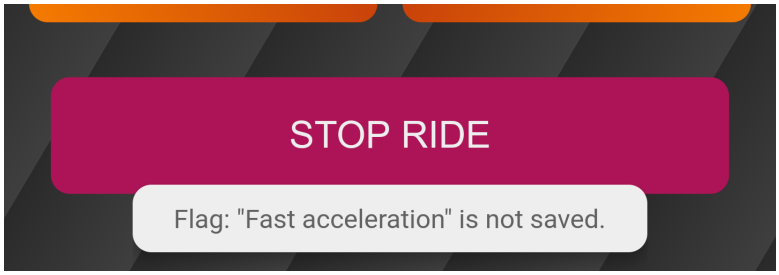
Zadnji gumb završava vožnju i preusmjerava korisnika na Zaslon s obrascima.

Slika 5.6: Funkcijski zaslon

Klikom na funkcijski gumb u prvom koraku kreira se *Flag* objekt. U sljedećem koraku se aplikacija spaja na bazu podataka te, ako je konekcija uspješna, aplikacija pokušava poslati objekt na samu bazu podataka. Ispituje se uspješnost transakcije objekta. Ako je uspješno poslan i dodan u predefiniranu kolekciju na bazi, aplikacija ispisuje kratku obavijest na zaslonu kao što je prikazano na slici 5.7.

U slučaju kada se podatak ne spremi u bazu podataka, aplikacija će kao povratnu vrijednost vratiti obavijest kako objekt nije spremljen. Primjer možemo vidjeti na slici 5.8.

Slika 5.7: Uspješna transakcija objekta na bazu podataka



Slika 5.8: Neuspješna transakcija objekta na bazu podataka

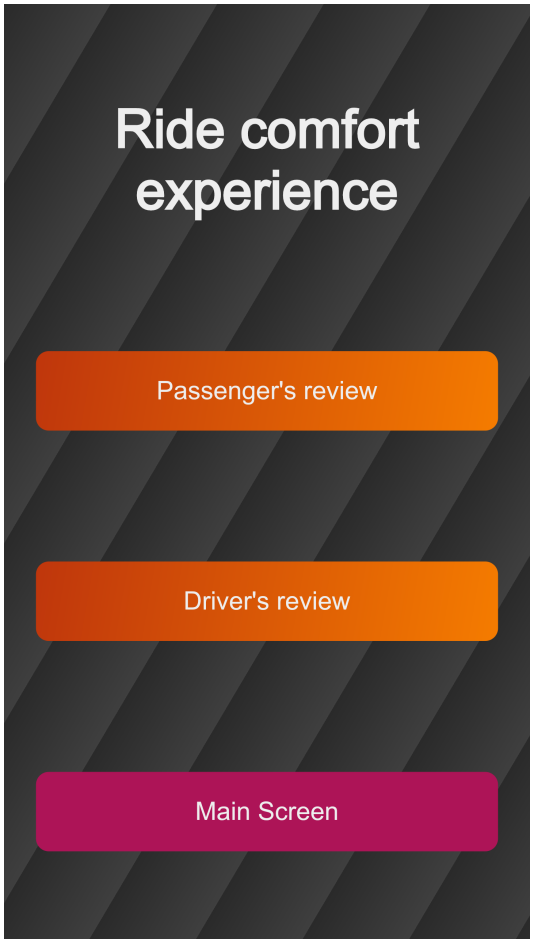
5.2.3. Zaslon s obrascima

Na zaslonu s obrascima nalaze se tri gumba kao što prikazuje slika 5.9.

Prvi gumb, *"Passanger's review"*, preusmjerava korisnika na Googleov upitnik kojeg je putnik obvezan ispuniti nakon svake vožnje kako bi dao dodatni osvrt na upravo završenu vožnju.

Drugi gumb, *"Driver's review"*, korisnika odvodi na Googleov upitnik koji je namijenjen vozaču. Vozač je također obvezan ispuniti ovaj upitnik nakon svake vožnje gdje se ostavlja dodatni osvrt na vožnju kako bi mogli što bolje opisati vožnju.

Zadnji, treći gumb korisnika preusmjerava na početni zaslon gdje može pokrenuti novu vožnju.



Slika 5.9: Zaslon s obrascima

5.3. Baza podataka i bilježenje događaja

5.3.1. Spajanje na bazu podataka

Spajanje na *MongoDB* izvedeno je u *MongoDatabase* klasi kroz asinkronu metodu *connect()*, kao što je prikazano na slici 5.10.

Kreira se potreban URI koji se sastoji od obaveznog dijela (*scheme*) **mongodb://** koji govori aplikaciji da se spaja na Mongo bazu podataka.

Sljedeća su dva parametra korisničko ime te šifra odvojeni operatorom ":", koji nisu obavezni ukoliko autentifikacijski dio na bazi podataka nije postavljen.

Autentifikacijski dio URI-a odvojen je operatorom "@" iza kojega slijedi *IP* adresa same baze te *port*, odvojeni operatorom ":".

Sljedeći dio URI-a jest *path* koji označava ime kolekcije na koju se spajamo.

Zadnji dio URI-a je *query* u kojem aplikacija postavlja bazi podataka da se želi prijaviti kao *admin*, to jest u zaglavlje HTTP upita stavlja za *authSource* vrijednost

"*admin*".

```
static connect() async {
  db = await Db.create("mongodb://" +
      "username" +
      ":" +
      "password" +
      "@" +
      "XXX.XXX.XXX.XXX" + //ip
      ":" +
      "XXXX" + //port
      "/marko_dodik" + //collection name
      "?authSource=admin"); //query
  await db.open();
  userCollection = db.collection("passenger_data");
}
```

Slika 5.10: Metoda za spajanje na bazu podataka

5.3.2. Kreiranje objekta Flag

Na slici 5.11 prikazana je klasa objekta *Flag* koji se kreira i šalje na bazu podataka.

```
class Flag {
  final String message;
  final String dateTime;
  final String rideCode;

  Flag(this.message, this.dateTime, this.rideCode);

  Flag.fromJson(Map<String, dynamic> json)
      : message = json['message'],
        dateTime = json['dateTime'],
        rideCode = json['rideCode'];

  Map<String, dynamic> toJson() =>
      {'message': message, 'dateTime': dateTime, 'rideCode': rideCode};
}
```

Slika 5.11: Klasa objekta i metode

Klasa sadrži 3 podatka, *message*, *dateTime* te *rideCode*. Svi podaci su objekti *String* klase. Objekt *message* sadrži jednu od poruka funkcijskih gumbiju, *dateTime* sadrži datum i vrijeme u trenutku kreiranja objekta te *rideCode* predstavlja *TripId*

U klasi su definirana dva kontruktora za kreiranje Flag objekta, te jedna metoda. Metoda *toJson()* pretvara objekt *Flag* u *String* objekt *JSON* formata.

5.3.3. Slanje objekta

Metoda koja osigurava spremanje objekta u bazu podataka jest insert*(Map<String, dynamic> flag)*. Metoda prima jedan parametar instance Flag, koji će potom spremiti u bazu. Prikazano na slici 5.12

```
static insert(Map<String, dynamic> flag) async {
  await userCollection.insert(flag);
  return true;
}
```

Slika 5.12: Metoda za spremanje objekta u bazu podataka

Nad *userCollection* objektom, koji je kolekcija na bazi podataka u koju aplikacija sprema *Flag* objekte te je definiran i inicijaliziran pri uspostavljanju konekcije s bazom u klasi *MongoDatabase*, vrši se metoda insert() koja u kolekciju dodaje objekt. Doda li metoda uspješno objekt u kolekciju, kao povratnu informaciju vratit će *bool* objekt *true*.

Ukoliko je proces neuspješan, metoda baca iznimku koja se obrađuje te vraća korisniku obavijest da objekt nije spremljen u bazu.

5.3.4. Zatvaranje konekcije

Metoda *closeConnection()* koristi se za sigurno zatvaranje konekcije s bazom podataka te ju se preporučuje uvijek koristiti zbog optimizacije baze i vrlo lake kontrole iznimaka (slike 5.13).

```
static closeConnection() {
  db.close();
}
```

Slika 5.13: Metoda za zatvaranje koneckije s bazom podataka

Nad objektom db, koji je upravo objekt baze podataka koji je kreiran pri uspostavljanju konekcije s bazom, poziva se metoda *close()* te ako je konekcija nije uspješno zatvorena,

vratit će se iznimka.

5.4. Prikupljeni podaci

Slika 5.14 prikazuje vožnje u kojima su se skupljali podaci unutar aplikacije za istraživanje o preferencama putnika o udobnosti vožnje.

Slika 5.14: Vožnje u kojima su se skupljali podaci

Filterom u MongoDB *{rideCode : "5844f"}* možemo odabrati vožnju koja je imala *tripId*="5844f". Slika 5.15 je prikaz podataka prikupljenih tijekom te odabrane vožnje. Iz njih se može zaključiti da je vozač vozio dinamičnijom vožnjom koja nije u cijelosti

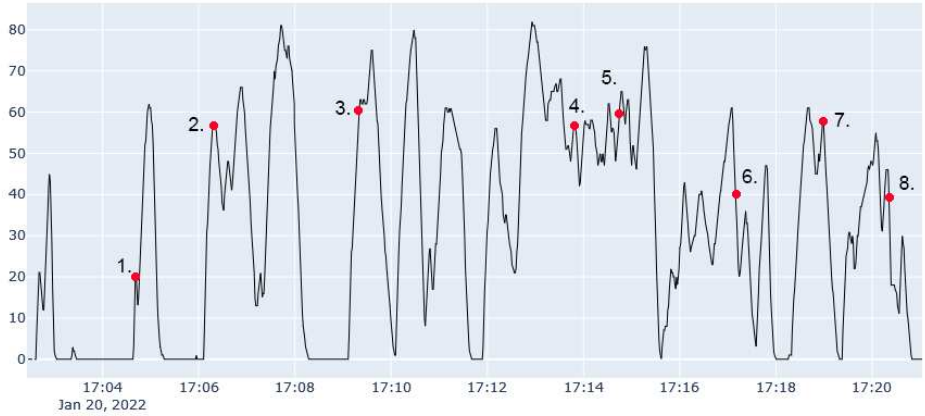
odgovarala putniku. Dodatnim rješavanjem upitnika u aplikaciji, obogaćujemo podatke s kojima baratamo.

Slika 5.15: *Flagovi* u vožnji "5844f"

Svi prikupljeni podaci kasnije će biti ključni za daljnju analizu koju prati rad "Eco-efficient driving pattern evaluation for sustainable road transport based on contextually enriched automotive data".

Provedeno je istraživanje koristeći razvijenu aplikaciju na 10 vožnji. U sljedećem tekstu nisu opisane vožnje koje su slične drugim vožnjama. Na slici 5.16 se vidi prikaz prve vožnje na karti. Vožnja se odvijala iz kvartovskog naselja Dubrava rutom Avenija Dubrava, Ulicom Dragutina Mandla, Branimirovom ulicom, Heinzelovom ulicom te preko Vukovarske ulice sve do Sveučilišta u Zagrebu Fakulteta elektrotehnike i računarstva. Graf prikaza prve vožnje 5.17 pokazuje brzinu kretanja vozila u ovisnosti o vremenskom intervalu u kojem se vožnja događala.Na grafu je također označeno osam točaka. Svaka od tih točaka predstavlja jedan od događaja kojeg je kreirao putnik te koji se spremio u bazu podataka. Na slici 5.18 prikazani su pohranjeni događaji u bazi. Prva točka označava događaj kada vozač vozi na premaloj udaljenosti iza vozila, a to možemo vidjeti i kroz usporavanje vozača u sljedećem trenutku. Drugi, treći, četvrti, peti i sedmi zabilježeni događaji su "slalomi" automobila, to jest često mijenjanje prometne trake. Na slici 5.17 se to može primijetiti kroz oscilacije brzine u manjem intervalu oko samih zabilježenih događaja. Šesti i osmi zabilježeni događaj predstavlja izostanak paljenje pokazivača smjera. U tim trenutcima vozilo se nalazi u zavoju, a to je dodatno vidljivo i kroz pad brzine nešto prije te poslije samog zabilježenog događaja.

Slika 5.16: Prikaz 1. vožnje na karti



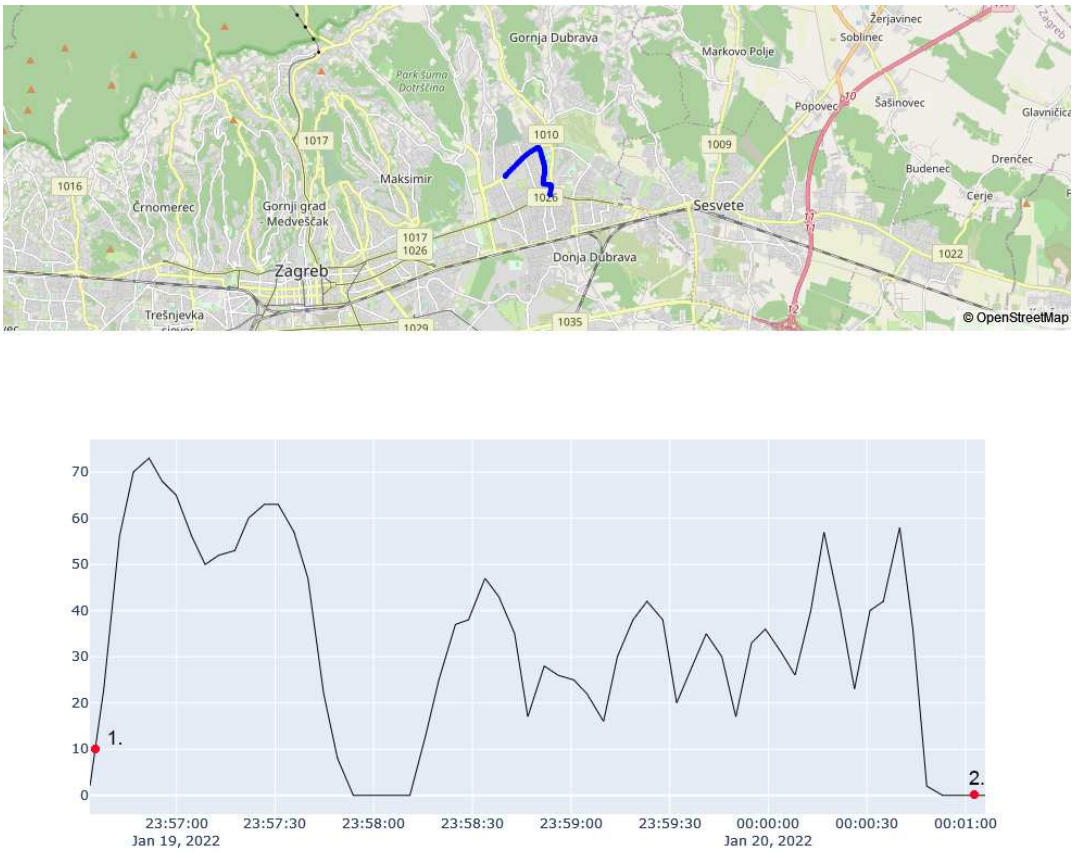
Slika 5.17: Prikaz grafa 1. vožnje

passenger_data

	_id ObjectId	message String	dateTime String	rideCode String
1	ObjectId('61e996247d2f4d6ef91…	"Close to car ahead"	"2022-01-20 18:04:35.624269"	"3b1d5"
2	ObjectId('61e996987d2f4d6ef91…	"Car slalom"	"2022-01-20 18:06:32.224081"	"3b1d5"
3	ObjectId('61e9974f7d2f4d6ef91…	"Car slalom"	"2022-01-20 18:09:35.466964"	"3b1d5"
4	ObjectId('61e998547d2f4d6ef91…	"Car slalom"	"2022-01-20 18:13:56.471523"	"3b1d5"
5	ObjectId('61e9988f7d2f4d6ef91…	"Car slalom"	"2022-01-20 18:14:55.279445"	"3b1d5"
6	ObjectId('61e999237d2f4d6ef91…	"Not using blinkers"	"2022-01-20 18:17:22.718554"	"3b1d5"
7	ObjectId('61e999857d2f4d6ef91…	"Car slalom"	"2022-01-20 18:19:01.148430"	"3b1d5"
8	ObjectId('61e999d27d2f4d6ef91…	"Not using blinkers"	"2022-01-20 18:20:16.142876"	"3b1d5"

Slika 5.18: Prikaz podatak skupljenih u 1. vožnji

Druga vožnja, slika 5.19, *tripId* "efcfa", jest vožnja kroz kvartovsko naselje Dubrava. Zabilježena su dva događaja, oba kateogrije "Not using blinkers". Koristeći vremenske podatke korištene za kreiranje rute vožnje te vremenske podatke iz objekta kojega je kreirao putnik, uočava se da vozač nije upalio pokazivače smjera kada je izlazio iz parkinga te na kraju vožnje kod parkiranja vozila.

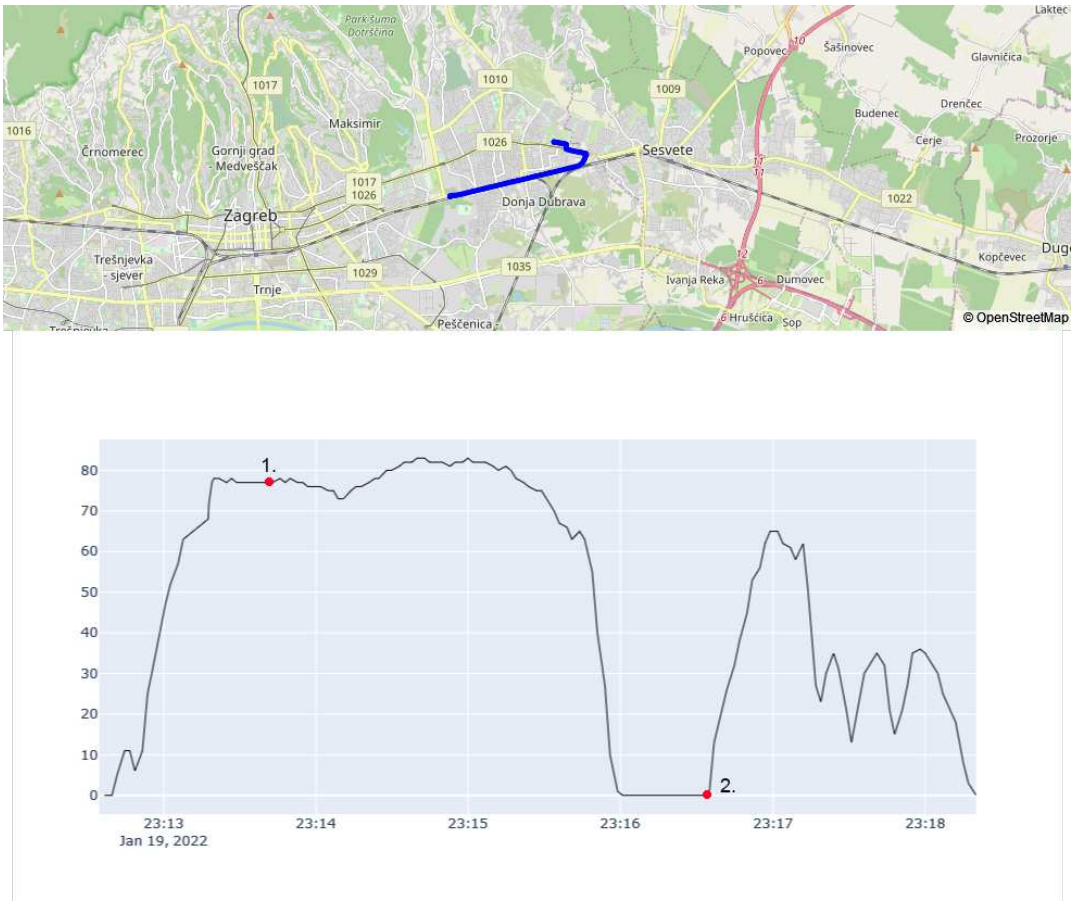


Slika 5.19: Prikaz 2. vožnje s grafom

Treća vožnja, prikaza na slici 5.20 s *tripId*-em "a23ba", ima dva zabilježena događaja. Oba događaja pripadaju kategoriji "Not using blinkers" te po promjenama brzina i samom kartom rute kojom je vozilo prometovalo, zaključuje se da vozač nije palio pokazivače smjera kod skretanja u desno na dva križanja.

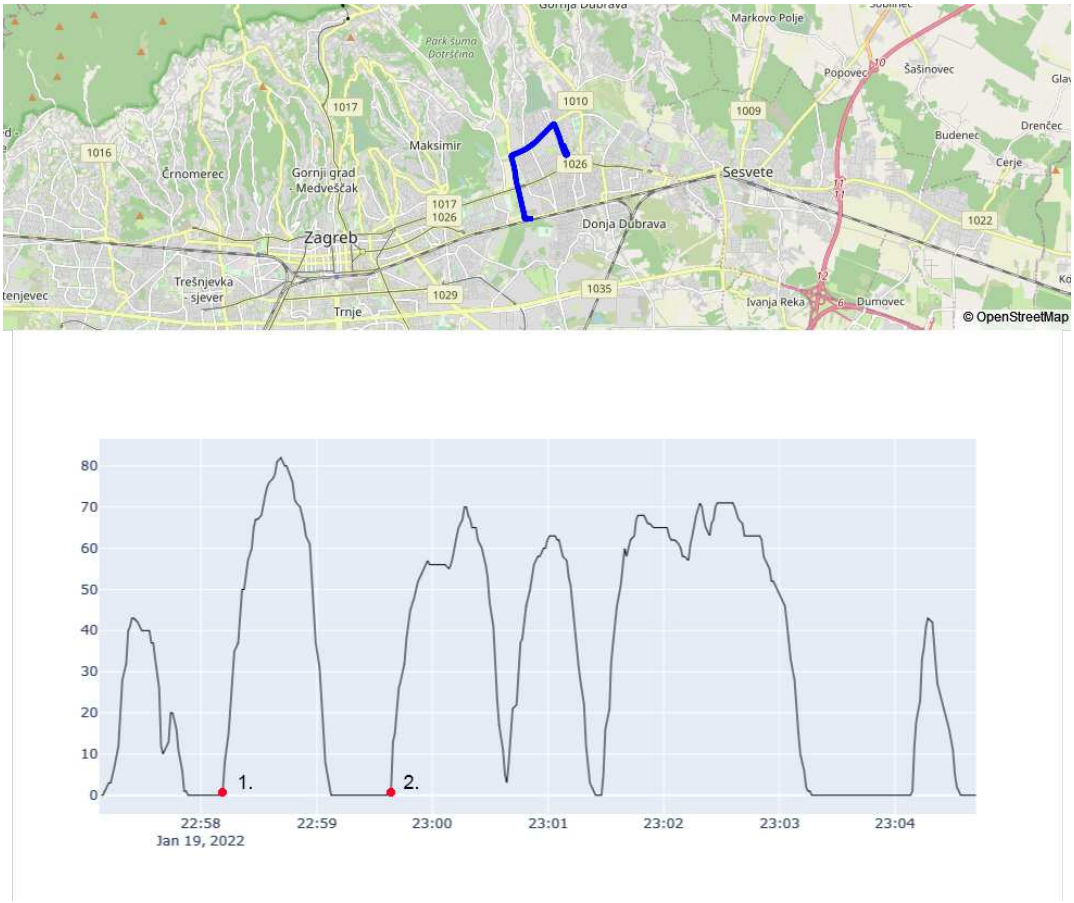
Slika 5.20: Prikaz 3. vožnje s grafom

Četvrta vožnja prikazana je na slici 5.21 te su naznačena dva događaja. Vožnja ima *tripId* "0374f". Prvi događaj označen na slici pripada "Close to car ahead" što rezultira stalnom brzinom automobila u kratkom intervalu prije te poslije zabilježenog događaja. Drugi događaj, "Not using blinkers", zabilježen je u trenutku kada je vozilo kretalo te pretpostavljamo da se vozilo nalazilo na križanju i nije se kretalo prije samog događaja.



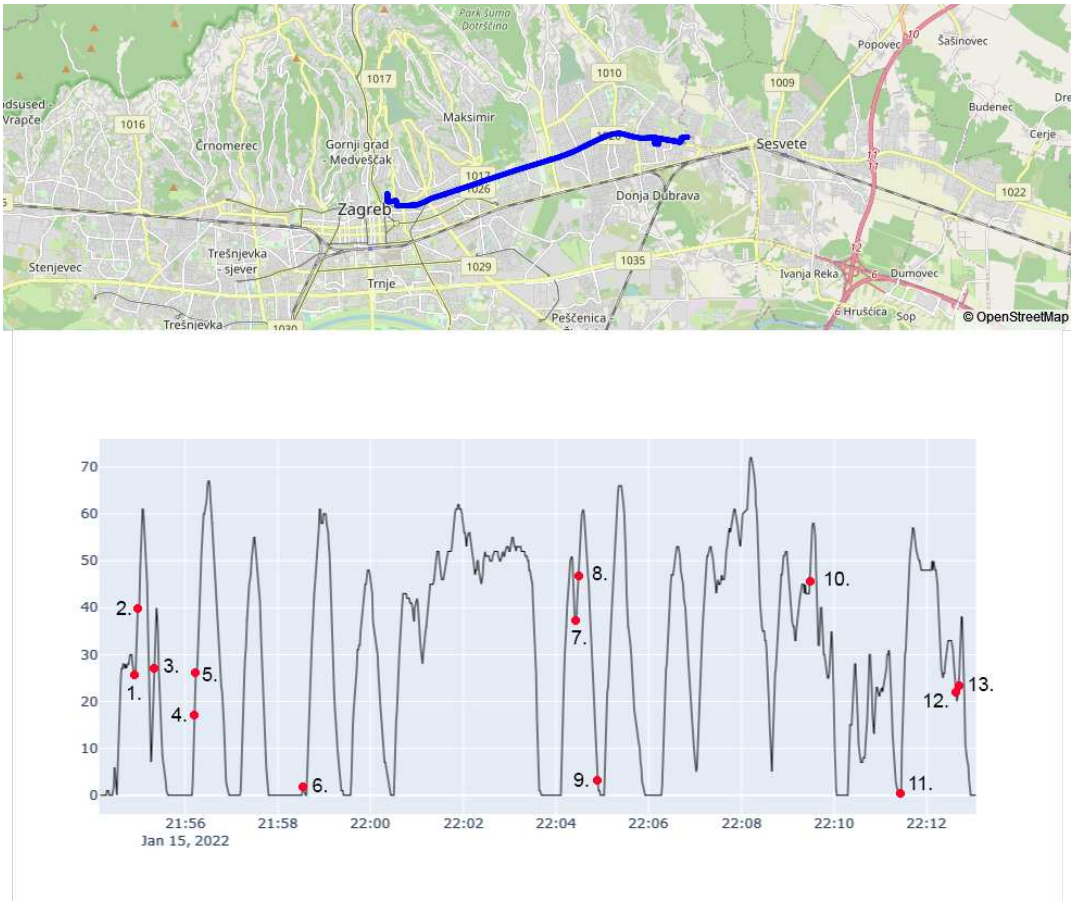
Slika 5.21: Prikaz 4. vožnje s grafom

Kod pete vožnje s *tripId*-em "e2411" kreirana su dva događaja. Oba događaja su kategorije *"Not using blinkers"*.



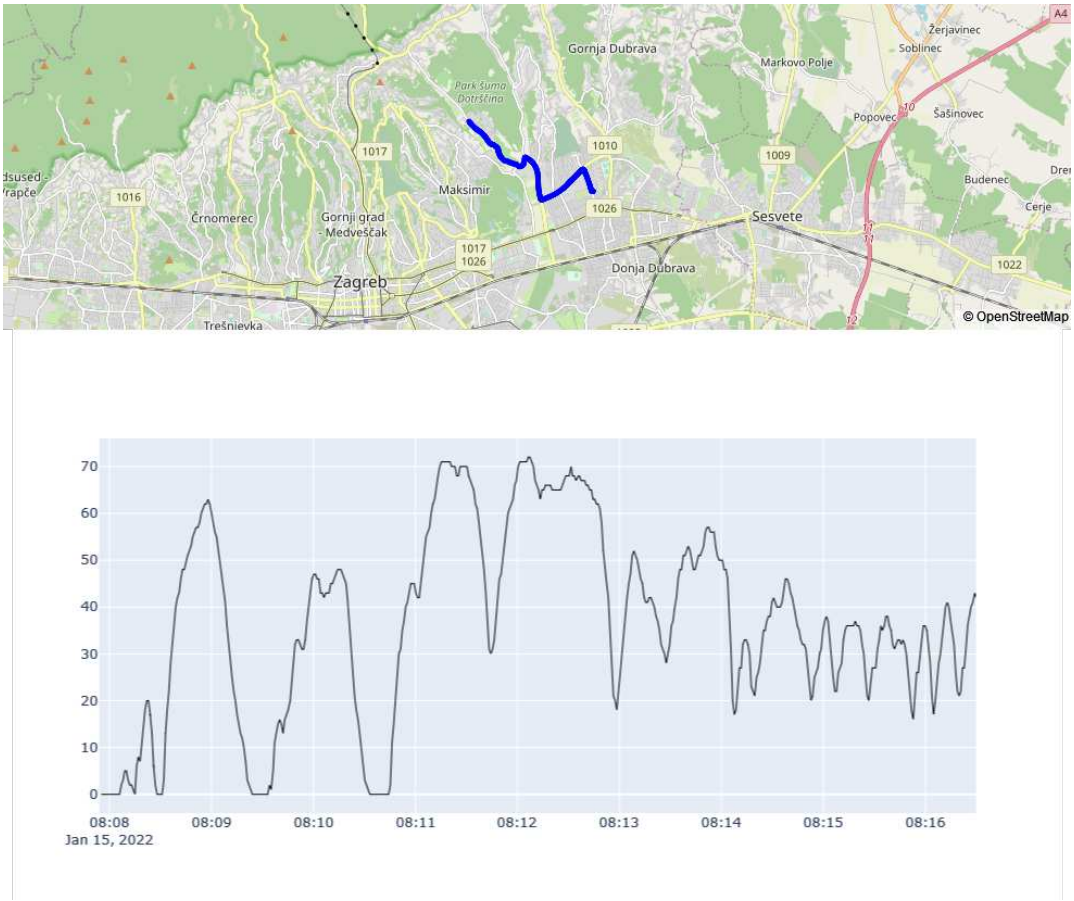
Slika 5.22: Prikaz 5. vožnje s grafom

Vožnja pod brojem šest, prikazana na slici 5.23, počela je pored športsko rekreativnog centra Šalata, a završava na okretištu Dubec. U vožnji s *tripId*-em "2b7f2" zabilježeno je trinaest događaja. Prvi i peti događaj su iz kategorije "Turn at high speed", drugi, treći, četvrti, osmi, deseti te trinaesti su iz kateogrije "Fast acceleration". Šesti i deveti događaj pripadaju kategoriji "Close to car ahead"". Sedmi događaj zabilježava vozačevo učestalo mjenjanje vozne trake, to jest "Car slalom". Jedanaesti i dvanaest i događaji pokazuju da vozač nije upalio pokazivače smjera, to jest zabilježen je događaj iz kategorije "Not using blinkers".



Slika 5.23: Prikaz 6. vožnje s grafom

Sedma vožnja krenula je iz Ulice Josipa Torbarine, a završava pored šetališta Jazbina. Na grafu nema označenog niti jednog događaja. Taj podatak označava da putnik nije imao niti jednu primjedbu na vožnju te mu je vožnja u potpunosti odgovarala.

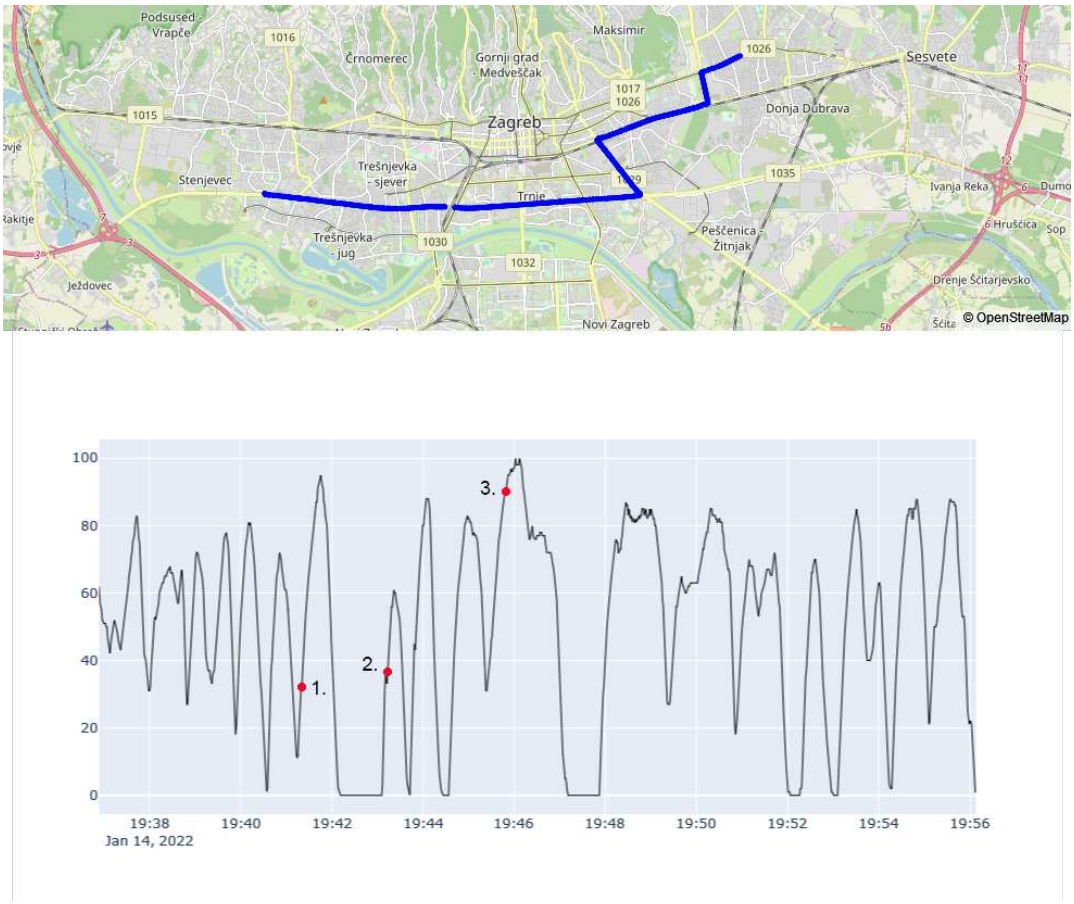


Slika 5.24: Prikaz 7. vožnje s grafom

Vožnji pod brojem osam, prikazana na slici 5.25, pridružen jest *tripId* "99632". Vožnja je trajala dvadeset i sedam minuta, ali ima zabilježen samo jedan događaj. Događaj je iz kategorije "Turn at high speed" te pokazuje da je putniku vožnja odgovarala.

Slika 5.25: Prikaz 8. vožnje s grafom

Posljednja vožnja u prikazana ovdje jest deveta. Njoj je pridružen *tripId* "5844f". Vožnja je započela na Trgu Ivana Kukuljevića u kvartovskom naselju Špansko, a završava u Malekovoj Ulici u kvartovskom naselju Dubrava. U vožnji su označena tri događaja. Prvi zabilježen događaj jest "Close to car ahead", drugi "Fast acceleration", te treći "Not using blinkers".



Slika 5.26: Prikaz 9. vožnje s grafom

6. Zaključak

Konačni rezultat ovog završnog rada jest mobilna aplikacija koja služi za prikupljanje podataka o subjektivnom doživljaju udobnosti vožnje. Mobilna aplikacija je osmišljena da bude jednostavna, lako upotrebljiva svima te pouzdana u prikupljanju i spremanju podataka. Aplikacija prikuplja i generira podatke u stvarnom vremenu te ih šalje i sprema na udaljenu bazu podataka. Objekt podatka koji se sprema na bazu podataka sadrži objekt *dateTime* preko kojega će se vrlo lako moći upariti podatke koji se šalju preko OBD-a na drugu bazu te će se dobiti šira slika o vožnjama. Upravo zato će se ti podaci koristiti za nadogradnju postojećeg sustava koji koristi OBD te kontekstno obogaćuje podatke.

U aplikaciju su implementirani i upitnici koji prikupljaju dodatne informacije o vozaču i putniku prije prve vožnje. Također, uključeni su i upitnici u kojima se prikupljaju subjektivni podaci o svakoj vožnji. Razvijeno programsko rješenje je i eksperimentalno evaluirano. Tijekom vožnji prikupljali su se objektivni podaci s vozila te podaci o subjektivnom doživljaju udobnosti vožnje. Točnije, putnik je koristeći razvijenu aplikaciju bilježio predefinirane događaje tijekom vožnje. Prikupljani podaci su pohranjeni te su međusobno uspoređeni i analizirani.

Sljedeći korak u nadogradnji aplikacije jest lokalno spremanje svih podataka te slanje na bazu tek na kraju svake vožnje. Takav pristup bi otvorio nove mogućnosti rada aplikacije te bi aplikacija radila i bez internetske veze, to jest konekcije na bazu. Kolekciju podataka bi automatski mogli proširiti s podacima putnika i vozača. Dodatni korak bi bio implementacija automatskog kreiranja karte s rutom putovanja na kraju svake vožnje uz pomoć čitanja podataka iz baze podataka gdje se spremaju podaci s OBD uređaja.

LITERATURA

[1] S. Waldman, "Atmospheric carbon dioxide hits record levels," 03 2017. poveznica: https://www.scientificamerican.com/article/atmospheric-carbon-dioxide-hits-record-levels/, pristupljeno 14. siječnja 2022.

[2] NASA, "Graphic: The relentless rise of carbon dioxide," poveznica: https://www.scientificamerican.com/article/atmospheric-carbon-dioxide-hits-record-levels/, pristupljeno 14. siječnja 2022.

[3] NASA, "Climate change: How do we know?," poveznica: https://climate.nasa.gov/evidence/, pristupljeno 14. siječnja 2022.

[4] UNFCCC, "Kyoto protocol," poveznica: https://unfccc.int/kyoto_protocol, pristupljeno 14. siječnja 2022.

[5] I. Aris, M. F. Zakaria, S. Bashi, and R. Sidek, "Development of obd-ii driver information system," 03 2007.

[6] D. Rimpas, A. Papadakis, and M. Samarakou, "Obd-ii sensor diagnostics for monitoring vehicle operation and consumption," *Energy Reports*, vol. 6, 10 2019.

[7] N. Rajhans, R. Rajhans, and D. Ahuja, "Analyzing factors affecting driving comfort and their interrelationship in a passenger car using ahp," 12 2007.

[8] "Flutter." poveznica: https://flutter.dev, pristupljeno 14. siječnja 2022.

[9] "Dart." poveznica: https://dart.dev/, pristupljeno 14. siječnja 2022.

[10] "Mongodb." poveznica: https://www.mongodb.com/, pristupljeno 14. siječnja 2022.

[11] “Latex project,” poveznica: https://www.latex-project.org/, pristupljeno 14. siječnja 2022.

[12] HAK, “Osnovni rezultati istraŽivanja navika vozaČa,” poveznica: https://www.hak.hr/datoteka/1200/rezultati-hak-ovog-istrazivanja-o-navikama-vozaca.pdf, pristupljeno 14. siječnja 2022.

[13] HAK, “Rezultati istraživanja navika vozača,” poveznica: https://www.hak.hr/datoteka/1201/nenad-zuber_zona-istrazivanje-prezentacija.pdf, pristupljeno 14. siječnja 2022.

[14] “Mongodb github.” poveznica: https://github.com/mongodb/mongo, pristupljeno 14. siječnja 2022.

[15] “Database types.” poveznica: https://www.javatpoint.com/types-of-databases, pristupljeno 14. siječnja 2022.

[16] “Mongodb dart package.” poveznica: https://pub.dev/packages/mongo_dart, pristupljeno 14. siječnja 2022.

[17] “Mongodb dart package.” poveznica: https://www.hak.hr/, pristupljeno 14. siječnja 2022.

[18] FER, “Fer programsko inženjerstvo,” poveznica: https://www.fer.unizg.hr/predmet/proinz, pristupljeno 14. siječnja 2022.

[19] F. Martinelli, F. Mercaldo, V. Nardone, A. Orlando, and A. Santone, “Cluster analysis for driver aggressiveness identification,” poveznica: https://pdfs.semanticscholar.org/1491/1eddf9495dbc60f3cb1382adbf5d0bf08b2f.pdf, pristupljeno 14. siječnja 2022.

[20] A. af Wåhlberg, “Short-term effects of training in economical driving: Passenger comfort and driver acceleration behavior,” *International Journal of Industrial Ergonomics*, vol. 36, no. 2, pp. 151–163, 2006.

[21] J.-H. Hong, B. Margines, and A. Dey, “A smartphone-based sensing platform to model aggressive driving behaviors,” *Conference on Human Factors in Computing Systems - Proceedings*, 04 2014.

[22] A. Puchalski and I. Komorska, “Driving style analysis and driver classification using obd data of a hybrid electric vehicle,” *Transport Problems*, vol. 15, pp. 83–94, 12 2020.

[23] B. Barabino, M. Coni, A. Olivo, G. Pungillo, and N. Rassu, “Standing passenger comfort: A new scale for evaluating the real-time driving style of bus transit services,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 12, pp. 4665–4678, 2019.

[24] M. Loman, B. Šarkan, and T. Skrúcaný, “Comparison of fuel consumption of a passenger car depending on the driving style of the driver,” *Transportation Research Procedia*, vol. 55, pp. 458–465, 2021. 14th International scientific conference on sustainable, modern and safe transport.

Mobilna aplikacija za praćenje subjektivnog doživljaja udobnosti vožnje

Sažetak

Tema ovog rada odnosi se na prikupljanje i spremanje podataka u stvarnom vremenu o zapažanjima putnika i vozača tijekom vožnje putem aplikacije za pokretne uređaje. Svrha prikupljanja podataka jest istraživanje preferencija vožnje kod različitih osoba te proučavanja mogućih načina prilagodbe vožnje s ciljem povećanja zadovoljstva uslugom prijevoza. Prilikom prvog pristupa aplikaciji, putnik i vozač ispunjavaju upitnik u svrhu utvrđivanja preferencija o vožnji. Potom, na kraju provedene vožnje, putnik i vozač ispunjavaju drugi upitnik o samom doživljaju provedene vožnje. Središnji dio aplikacije je praćenje subjektivnog doživljaja udobnosti vožnje putnika. Točnije, putnik ima priliku prijaviti predefinirane događaje (npr. naglo ubrzanje, naglo kočenje, neprilagođeno skretanje) koje zamjećuje tijekom vožnje. Prijavljeni događaji se u stvarnom vremenu šalju i pohranjuju u bazu podataka kako bi bili dostupni za daljnja istraživanja.

Ključne riječi: transportni sektor, vožnja, udobnost, aplikacija za pokretne uređaje, Flutter

Mobile application for tracking the subjective experience of driving comfort

Abstract

The topic of this work is collecting and storing data created by driver and passenger on a trip in real time using mobile application. The goal of this paper is to help in research about driving preferences for different people and using collected data in order to increase passanger satisfaction with the transport service. After first time accessing application, passanger and driver need to fill a form about their driving preferences. Then, at the place of the complited ride, the passanger and the driver shall fill another form about the experience of the trip itself. The main part of application is tracking passanger's comfort expirience. More specifically, the passenger has the possibility to report predefined events (e.g. rapid braking, rapid acceleration, uncontrolled steering) that he/she is experiencing during the ride. Reported incidents shall be logged in real time and stored in the database as they would be available for further analysis.

Keywords: transport sector, driving, comfort, mobile application, Flutter