UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

ZAKLJUČNA NALOGA

SPLETNA APLIKACIJA ZA DELJENE PREVOZE SOPOTNIK

KRIŠTOF FABJAN

2022
ZAKLJUČNA NALOGA
FABJAN

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Spletna aplikacija za deljene prevoze Sopotnik

(Web application for shared rides Sopotnik)

Ime in priimek: Krištof Fabjan
Študijski program: Računalništvo in informatika
Mentor: izr. prof. dr. Matjaž Kljun
Somentor: izr. prof. dr. Klen Čopič Pucihar

Koper, avgust 2022

Ključna dokumentacijska informacija

Ime in PRIIMEK: Krištof Fabjan

Naslov zaključne naloge: Spletna aplikacija za deljene prevoze Sopotnik

Kraj: Koper

Leto: 2022

Število listov: 61 Število slik: 36

Število referenc: 37

Mentor: izr. prof. dr. Matjaž Kljun

Somentor: izr. prof. dr. Klen Čopič Pucihar

Ključne besede: deljeni prevozi, spletna aplikacija, avtomatizacija prijav, LAMP, Laravel

Izvleček:
Zaključna naloga predstavlja načrtovanje in implementacijo spletne aplikacije Sopotnik, ki je namenjena izboljšavi uporabniške izkušnje rabe aplikacije za deljenje prevozov. Glavni namen aplikacije je izboljšati obstoječe rešitve na način, da je upravljanje s prevozi (prijave, odjave ...) avtomatizirano, hkrati pa je aplikacija še vedno preprosta in varna za uporabo. V nalogi je najprej predstavljen pregled lastnosti aplikacij za deljene prevoze ter potencialne rešitve. Kot osnovo za potencialni izboljšave je bila izbrana slovenska platforma "Prevoz.org". V primerjavi s to aplikacijo je prijava na prevoze s klici/sporočili nadomeščena z alternativno spletno prijavo, verodostojnost potnikov/prevoznikov pa je vidna s pomočjo ocen uporabnikov. Dodane so tudi sistemske ocene, kjer aplikacija sama spremlja delovanje uporabnika in ga nagrajuje/kaznuje glede na uporabnikova dejanja med uporabo aplikacije. Ustvarili smo tudi dodatni vmesnik za vmesne postanke, ki omogoča jasno definiranje podrobnosti o postankih. Dodatno aplikacija tudi rešuje določene pomanjkljivosti, ki so bile zaznane med analizo uporabniških vtisov na spletnih forumih. Nekaj takšnih je na primer; objava več prevozov v istem časovnem intervalu, avtomatizacija in simplifikacija vnosa podatkov ter upravljanje s prijavami na prevoz in podobno.

Key words documentation

Name and SURNAME: Krištof FABJAN

Title of final project paper: Spletna aplikacija za deljene prevoze Sopotnik"

Place: Koper

Year: 2022

Number of pages: 61 Number of figures: 36

Number of references: 37

Mentor: Assoc. Prof. Matjaž Kljun, PhD

Co-Mentor: Assoc. Prof. Klen Čopič Puchihar, PhD

Keywords: shared rides, web application, automatization, LAMP, Laravel

Abstract: The final thesis presents the design and implementation of a web application called Sopotnik, which aims to improve the user experience of a ride-sharing app. The main purpose of the application is to improve the existing solutions in a way that the management of the rides (check-in, check-out, etc.) is automated, while the application remains simple and safe to use. In the thesis we first present an overview of the features of ride sharing applications and potential upgrades. The Slovenian platform "Prevoz.org"was selected as a basis for potential improvements. Compared to this application, the calling/texting check-in is replaced by an alternative online check-in and the credibility of passengers/carriers is visible through user ratings. System ratings have also been added, where the application itself monitors users' performance and rewards/penalises them according to their actions while using the application. We have also created an additional interface for stopovers, which allows stopover details to be clearly defined. Additionally, the app also addresses certain shortcomings that were detected during the analysis of user impressions on online forums. Some of these are for example: the publication of multiple rides in one go, the automation and simplification of data entry, and the management of ride requests, etc.

Zahvala

Zahvaljujem se mentorjema izr. prof. dr. Matjažu Kljunu in izr. prof. dr. Klenu Čopič Puciharju za pomoč, nasvete in podporo pri pisanju te zaključne naloge. Posebej se zahvaljujem družini in punci, za podporo med celotnim študijem.

Kazalo vsebine

Kazalo slik

1 Uvod

Transport se skozi zgodovino aktivno spreminja. Med pisanjem te zaključne naloge, v letih 2021 in 2022, smo priča epidemiji koronavirusa in napetih geopolitičnih razmer, ki so zamajale industrijo in cene pogonskih goriv. Zaradi dragega goriva, globalnega segrevanja, slabe prometne infrastrukture in dragih javnih prevozov se veliko ljudi zateka k alternativnim možnostim transporta.

V Sloveniji bi bila vzpostavitev dobro delujočih javnih prevozov precejšen zalogaj. Vzrok za to je več dejavnikov, med glavnimi pa sta razgiban relief in razdrobljenost poselitve, kar onemogoča, da bi imela vsaka vas ali celo zaselek povezavo z javnim prevozom. Tu so še ekonomski in družbeni razlogi. V Sloveniji sega postavitev železniškega sistema v 19. stoletje, te proge pa ne omogočajo visokih hitrosti. V železniškem prometu, kljub določenim posodobitvam, še vedno prevladujejo dokaj počasni vlaki, kar naredi tovrsten način javnega prevoza tako rekoč neuporaben. Za primer, povprečen čas vožnje z vlakom med Ljubljano in Mariborom je 2 uri in 40 minut, kar je 1,5-krat več kot za pot potrebujemo z avtomobilom. Hkrati pa je tudi cena enaka porabljenemu gorivu na tej razdalji in do dvakrat višja kot na aplikaciji, katere izboljšavo bomo predstavili.

Tu so še avtobusni prevozi, ki so lahko še počasnejši. Če se veliko ljudi, sploh študentov in zaposlenih v poklicih, ki zahtevajo delo za računalnikom, odloči za prevoz z vlaki, za avtobuse ni tako. Na enaki relaciji LJ-MB, je na primer v povprečju avtobusna vozovnica še 20 % dražja od vozovnice za vlak, je pa vožnja v povprečju krajša za 15 %.

Zaradi teh razlogov se, sploh mladi, odločajo in iščejo druge možnosti prevoza. Ena najbolj priljubljenih je slovenski portal Prevoz.org, ki je namenjen deljenju prevozov z osebnimi avtomobili. Čeprav je portal Prevoz.org zelo uporaben, pa ima še vedno pomanjkljivosti in možnosti za izboljšavo, katerih načrtovanje in implementacija so opisani v tej zaključni nalogi.

1.1 Namen in cilji naloge

Namen te zaključne naloge je izboljšanje opaženih pomanjkljivosti obstoječih rešitev uporabljenih pri aplikacijah za deljenje prevozov. Cilj naloge je tako razviti spletno aplikacijo ki bo rešila opažene pomankljivosti - predvsem avtomatizacijo upravljanja s prevozi (prijave, odjave, vmesni postanki) in verifikacijo uporabnikov prek medsebojnega in sistemskega ocenjevanja uporabnikov.

1.2 Struktura naloge

V nalogi bomo predstavili izdelavo spletne aplikacije Sopotnik. Predstavitev poteka izdelave bo sestavljena iz štirih glavnih delov. Najprej bo predstavljena raziskava potreb in pregled trga, nato pa bomo začeli z načrtovanjem programa ter z implementacijo, na koncu pa bomo predstavili še testiranje aplikacije. Nalogo bomo zaključili s povzetkom.

2 Obstoječe rešitve in možne izboljšave

V tem poglavju bomo predstavili nekaj obstoječih aplikacij za deljenje prevozov. Te aplikacije bomo tudi na kratko opisali, ter predstavili njihove pomanjkljivosti in možne izboljšave.

Med predstavljenimi aplikacijami so že omenjena aplikacija Prevoz.org in aplikaciji, ki naj bi izboljšali funkcionalnosti aplikacije Prevoz.org. To ste aplikaciji NapolniMojAvto in Spletna aplikacija za vozače. Za primerjavo bomo opisali tudi BlaBlaCar, ki je najbolj znana tuja aplikacija za deljene prevoze.

Pri analizi aplikacij se bomo osredotočili predvsem na Prevoz.org, saj je naš namen izdelati aplikacijo za slovenske potrebe, tako da bomo analizirali in povzeli vtise s te aplikacije in predstavili predlagane izboljšave.

2.1 Prevoz.org

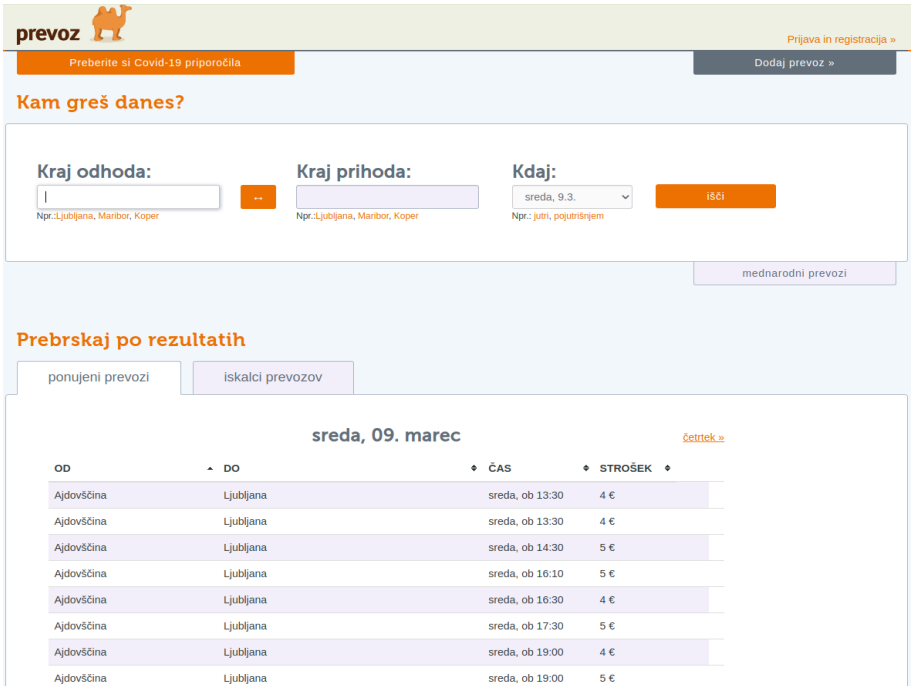
Prevoz.org je slovenska spletna platforma, ki nudi spletno in mobilno aplikacijo za deljenje prevozov. Aplikacijo so prostovoljno ustvarili študentje računalništva in družboslovja, zaradi pomanjkanja načinov za usklajevanje skupnih prevozov, predvsem za študente.

Aplikacija nudi voznikom objavo prevozov, na katere se potniki odzovejo/prijavijo tako, da pokličejo voznika na telefonsko številko, ali pa napišejo SMS sporočilo. Za potrebe verifikacije uporabnika in lažjega delovanja mobilne aplikacije je zaželjena potrditev telefonske številke, ni pa obvezna. Potniki lahko po potrebi iščejo prevoze med željenima lokacijama, ali pa objavijo povpraševanje za prevoz.

Aplikacija omogoča tudi ocenjevanje uporabnikov, ki pa je vidno šele, ko ima uporabnik več ocen.

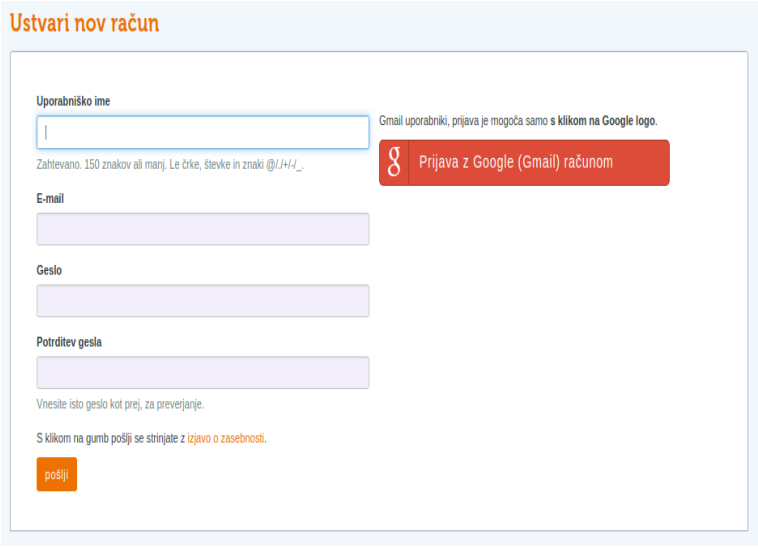
2.1.1 Uporabniška izkušnja

Ob obisku spletne aplikacije Prevoz.org hitro opazimo, da je spletna stran zasnovana preprosto in razumljivo, je pa očitno, da je grafični vmesnik že rahlo zastarel in ni tako privlačen za interakcijo, kar je vidno na Sliki 1. Na začetni strani lahko uporabnik išče prevoze z iskalnikom. Na voljo so mu tudi povezave za prijavo in objavo prevoza.



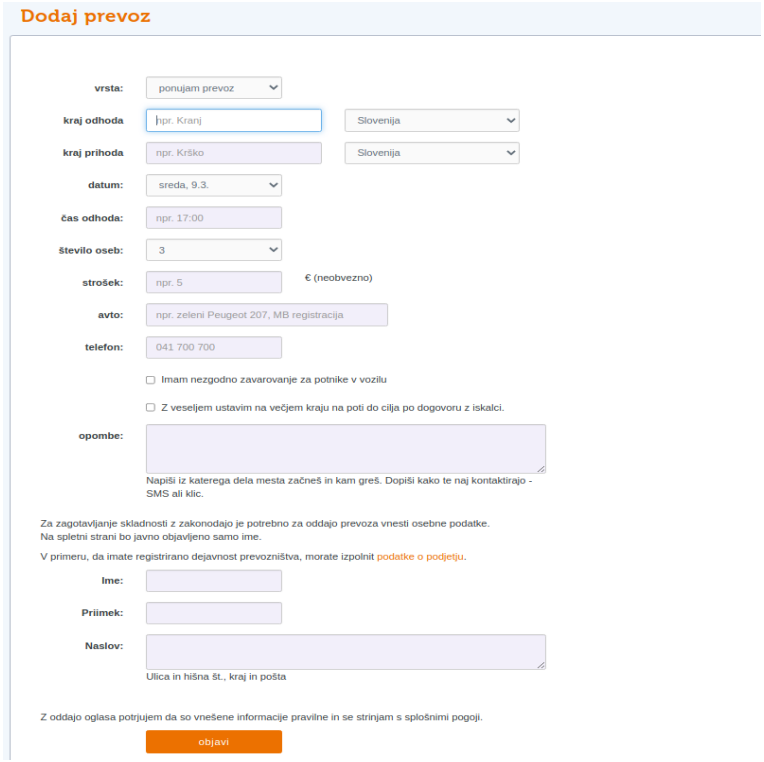
Slika 1: Začetna stran Prevoz.org.

Da lahko uporabnik objavlja prevoze, se mora predhodno registrirati in prijaviti. Registracija in prijava potekata po ustaljenem postopku (Slika 2). Aplikacija omogoča tudi hitro registracijo z Google uporabniškim računom.



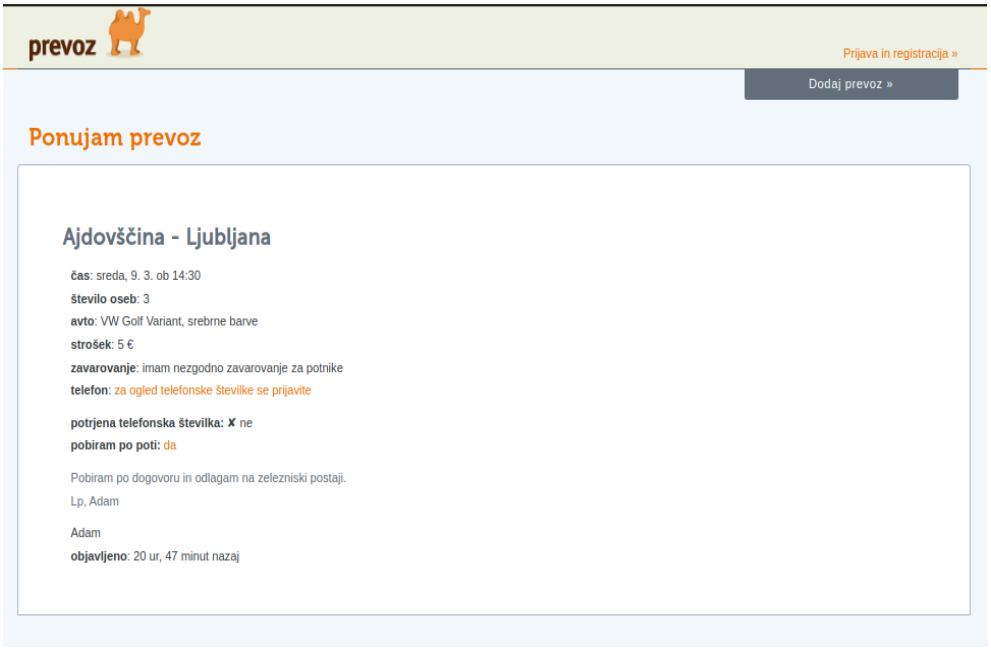
Slika 2: Registracija na Prevoz.org.

Uporabnik lahko ob prijavi vnese prevoz s pomočjo obrazca (Slika 3). Izbira lahko med objavo prevoza in objavo povpraševanja za prevoz (objavi kam želi iti in čaka, da kdo stopi v stik z njim). Uporabnik po vnosu vseh potrebnih podatkov potrdi prevoz.



Slika 3: Vnos prevoza na Prevoz.org.

Uporabnik lahko nato na seznamu prevozov izbere prevoz, ki ustreza njegovim potrebam. S klikom na prevoz se uporabniku odpre stran s podrobnostmi o prevozu, kjer lahko vidi dodatne informacije glede odhoda, voznikovega prevoznega sredstva (avtomobil, kombi ipd.) in kontaktne podatke (Slika 4).



Slika 4: Informacije o prevozu na Prevoz.org.

2.1.2 Analiza pomanjkljivosti

Aplikacija Prevoz.org je vsekakor, sploh študentom, precej olajšala potovanja po Sloveniji, kar pa še ne pomeni, da je aplikacija popolna oziroma, da ni dodatnih možnosti za izboljšave.

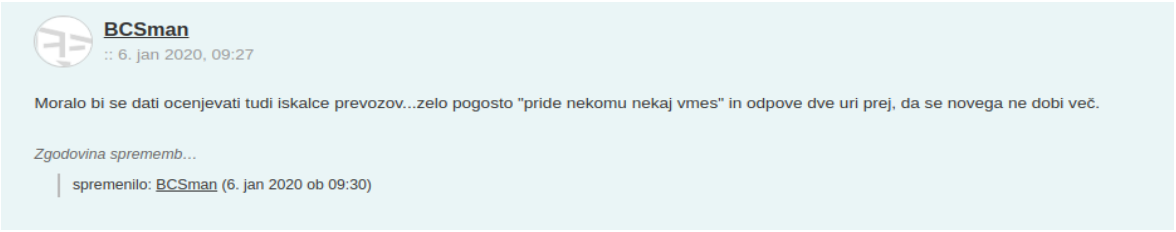
Pri zbiranju informacij o pomanjkljivostih aplikacije smo se oprli na predvsem na spletne forume. Na spletnih forumih se namreč najde precej odkritih informacij, ki jih morda v živo ni slišati/izkusiti, saj so ljudje na anonimnih forumih bolj odkriti. Nekaj pomanjkljivosti oziroma lastnih izkušenj z aplikacijo Prevoz.org bo podal tudi avtor diplomske naloge, saj je določeno obdobje tudi sam veliko uporabljal to aplikacijo.

Mnenja s spleta

Med iskanjem mnenj o aplikaciji Prevoz.org, smo na največ konstruktivnih mnenj naleteli na slovenskem forumu Slotech, od koder bomo povzeli nekaj najbolj relavantnih zapisov ([38] in [39]). Med mnenji in pripombami se največkrat pojavljajo sledeče pripombe:

- Zamujanje voznikov/potnikov in nezmožnost sistema za kaznovanje/označevanje takih voznikov (Slika 5).
- Pomanjkanje uporabnega ocenjevanja voznikov/potnikov (Slika 6).
- Pritoževanje nad zastarelim izgledom in ponavljanjem vnašanja enakih informacij za vsak prevoz. Pomanjkanje informacij o prevozu (Slika 7).

Sledi nekaj mnenj, ki odražajo najpogosteje predstavljene težave aplikacije.



Slika 5: Zamujanje in potreba po ocenjevanju [38].

Se strinjam, je problem.
Približno polovico prevozov ki jih sprejmem se odpove ali se ne zgodijo. Zaradi tega sem nekaj časa sploh nehal ponujat prevoze.
Potem so tisti, ki objavijo 16 oglasov za enako furo.
Potem je tist, ki je peljal 4 ljudi v cliotu iz kopra do celja poleti brez klime kjer dobesedno ni bilo prostora za prtljago.
Pa še en primerek zadnjič (ki je bil pač edini takrat), čakalo se ga je uro in pol s stalnimi izgovori, ko je končno prišel pa je baje bil mal zadet, vozil pa kot budala. Si ga je moja shranila v imenik samo zato, da ga slučajno ne sreča nikoli več.
Ampak to je bolj kot ne pač kultura ljudi, ki je marsikdaj na psu.

Sama stran takšna kot je, je enostavna za uporabo, in še vseeno boljše da je tako kot pa nič.

Ocenjevanje se da resiti samo v primeru, da se implementira ocenjevanje voznikov in vozacev. Ampak ne nekaj na hojladri, temvec na nivoju blablacar-a. Verjamem pa, da so za to potrebni resursi za razvijalca. V tem primeru bi se razni opisi folka ki zamuja, se ne javlja itd hitro poznali in folk ne bi dobival vec prevozov.

Slika 6: Potreba po ocenjevanju potnikov in voznikov [38].

FTad
:: 17. dec 2017, 17:42

Blinder je 17. dec 2017 ob 15:32 izjavil:

> *Tip hoce 10€ za pot ki je ponavadi 5 ali 6€. Kot da ni dovolj odda oglas iz dveh stevilk, da poskusi drugim prevoznikom uzet kako stranko. Ce ima kdo pet minut casa.. ->*

> *ideja za izboljsavo: tista zadeva ko clovek oznaci v kateri del mesta gre.. ne deluje najbolje. Skoaj noben tega ne uporablja. Treba bi spremenit tako, da ponudniki oznacijo s tagi (checkboxi) v katere dele mesta grejo. Pa naredit, da se ne da oddat oglasa za prevoz v velika mesta brez tag-ov. Ker trenutno stanje je da je kup lenob ko se jim ne ljubi pisat v oglas kam potujejo.*

Hja, v tem primeru bi bila dobra implementacija taka, da sistem lahko preveri, koliko oglasov si objavil na isti dan v dolocenem okviru casa. Ce zazna, da si za isto smer dal v casovnem razmiku 10-30 min, ti sistem onemogoci dodajanje novih oglasov.

Sicer pa, pac ne gres s tistim ponudnikom prevoza, ce je dal dvojno ceno.

Slika 7: Objava večjega števila prevozov ob istem času [39].

Osebne izkušnje

Avto je, še posebej med študijem, pogosto uporabljal storitev deljenih prevozov. Uporabljal je le aplikacijo Prevoz.org, ker je bila in je še vedno edina delujoča aplikacija za tovrstne prevoze pri nas. Aplikacijo je večinoma uporabljal, da si je poiskal prevoz od doma do Kopra, kjer sem študiral.

V splošnem je imel z aplikacijo Prevoz.org in z dejanskimi prevozi precej dobre

izkušnje. Aplikacija in takšen način prevozov je skorajda "rešilna bilka", še posebej za študente, saj je slovenski javni prevoz že pregovorno neučinkovit in drag, pri čemer je število javnih prevozov manjše ravno ob koncih tedna, ko študenti največ potujejo.

Na prevozih v nekaj letih ni imel niti ene resnično slabe izkušnje. Če bi moral izbrati, bi rekel, da so bile najslabše izkušnje, ko mu je nekdo eno uro pred prevozom odpovedal prevoz in ni dobil nadomestnega. Na srečo se mu tudi tisti dan ni nikamor mudilo. Tudi na splošno se na prevoze ni 100 odstotno zanašal. Če je imel na primer izpit na fakulteti, je šel s prevozi v Koper en dan prej, tako da je imel v rezervi še javne prevoze, če bi ga kdo "pustil na cedilu".

Še ena negativna izkušnja je bila, ko je voznika čakal že pol ure in ni bilo nobenega obvestila s z njegove strani o zamujanju. Voznik je po eni uri zamude le prišel. V opisu prevoza je imel napisano, da bodo v avtu največ trije potniki in da je možno vzeti pritljajo (manjši kovček). Ob prihodu je voznik enoprostorca imel v avtu že 6 oseb (vključno z voznikom), tako da so imeli potniki potovalke v naročju.

Kljub pozitivnem mnenju o ideji in delovanju ter implementaciji aplikacije, je tudi avtor diplomske našel veliko možnosti za izboljšavo aplikacije. Med glavnimi so:

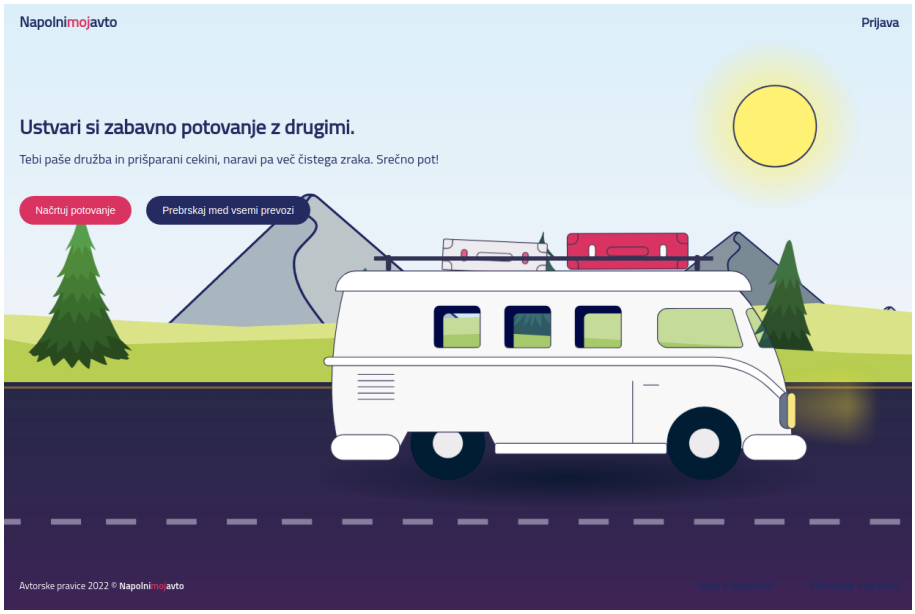
- Možnost spletne prijave na prevoze, tako da niso potrebni klici ali SMS sporočila.
- Izboljšano ocenjevanje potnikov in voznikov, saj bi se tako lahko lažje videlo s kom se vozimo. Ocenjevanje bi moralo biti mogoče po opravljenem prevozu.
- Omejitev uporabnikov, da ne morejo objavljati več prevozov ob istem času, saj je kar nekaj voznikov z namenom, da napolnijo avto in omejujejo konkurenco, objavljalo več prevozov za isto relacijo.
- Aplikacija ima zastarelo grafično podobo.
- Nepraktičnost vnosa podatkov. Uporabnik mora za vsak prevoz posebej vnašati informacije o svojem avtu in o sebi, kar je časovno potratno. Uprabniki se tega s časom tudi naveličajo, kar privede do nepopolnih opisov prevozov in zmede med potniki ter povečano potrebo po komunikaciji.
- Omogočanje urejanja prevozov, saj je trenutno edina izbira, da izbrišemo prevoz, če pride do napake pri ustvarjanju prevoza. Avtor diplomske pogreša tudi možnosti za dodajanje vmesnih postankov, da lahko voznik vidi, ali se mu izplača na določenem kraju potnika pobrati.

2.2 NapolniMojAvto

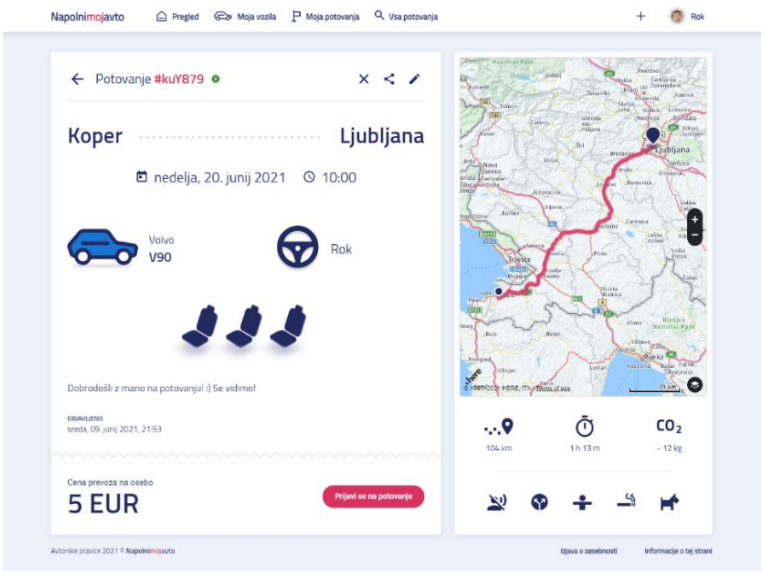
NapolniMojAvto je spletna aplikacija, ki je bila ustvarjena kot diplomsko delo, s podobnim namenom kot naša aplikacija Sopotnik. Ravno zaradi tega je bil pregled te aplikacije in diplomskega dela zelo zanimiv, saj se je lahko primerjalo naše ugotovitve in rešitve, z rešitvami aplikacije NapolniMojAvto. Tudi avtor aplikacije NapolniMojAvto se je namreč osredotočil na izboljšavo strani Prevoz.org.

Glavne ugotovitve in izbojšave avtorja so bile:

- Izboljšanje grafične podobe aplikacije. Avtor je dal največ poudarka na ta vidik, kar je tudi lepo vidno, saj je grafična podoba aplikacije zelo privlačna na pogled (Sliki 8 in 9).
- Olajšanje prijave na prevoze, saj je prijava s klici/SMS sporočili nepraktična.
- Ocenjevanje voznikov (ocenjevanja potnikov avtor ne omenja), saj bi lahko tako potniki lažje našli ustreznega voznika. Avtor to ugotovitev navede kot predlog, saj navaja, da je ta rešitev mišljena za naslednjo različico aplikacije, ki pa ni bila predstavljena v diplomskem delu.



Slika 8: Domača stran aplikacije NapolniMojAvto [36].



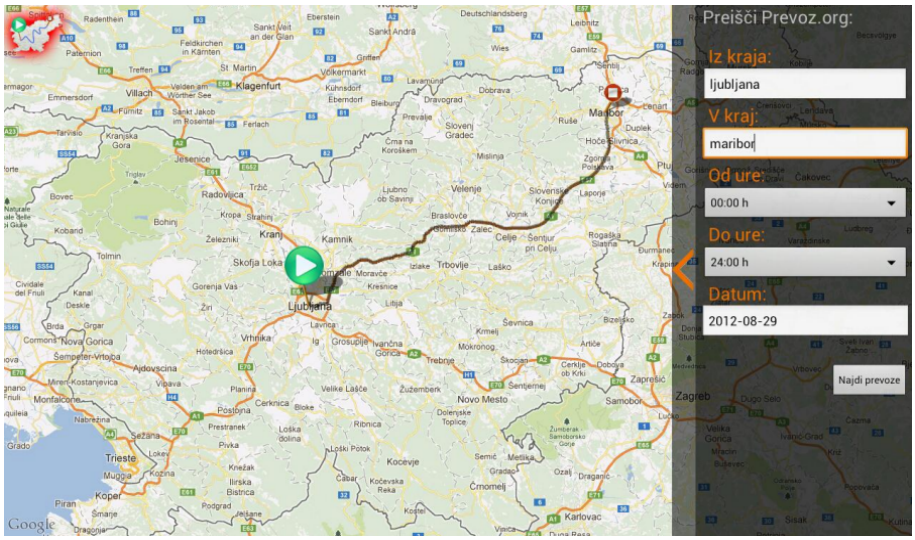
Slika 9: Vizualna podoba strani z informacijami o prevozu [36].

2.3 Spletna aplikacija za vozače

Spletna aplikacija za vozače je prav tako aplikacija, ki je bila ustvarjena za potrebe diplomskega dela. Namen aplikacije je obogatitev uporabniške izkušnje aplikacije Prevoz.org, ki je bila, sploh takrat (leta 2012) še bolj skopa kot danes.

Avtor je v omenjeni aplikaciji razvil funkcionalnosti, ki črpajo informacije iz aplikacije Prevoz.org in jih na grafično bolj privlačen način predstavijo v aplikaciji Spletna aplikacija za vozače. Glavne funkcionalnosti aplikacije so tako predstavitev začetne in končne lokacije ter izris poti med lokacijama (Slika 11) in pa tudi pregled posebnosti na cestah. Pri aplikaciji gre tako zgolj za izboljšavo vizualne izkušnje uporabe aplikacije Prevoz.org. Avtor ni implementiral dodatnih funkcionalnosti za lažje upravljanje s prevozi.

Slika 10: Seznam prevozov v Spletni aplikaciji za vozače [10].



Slika 11: Prikaz lokacije in izris poti med lokacijama [10].

2.4 BlaBlaCar

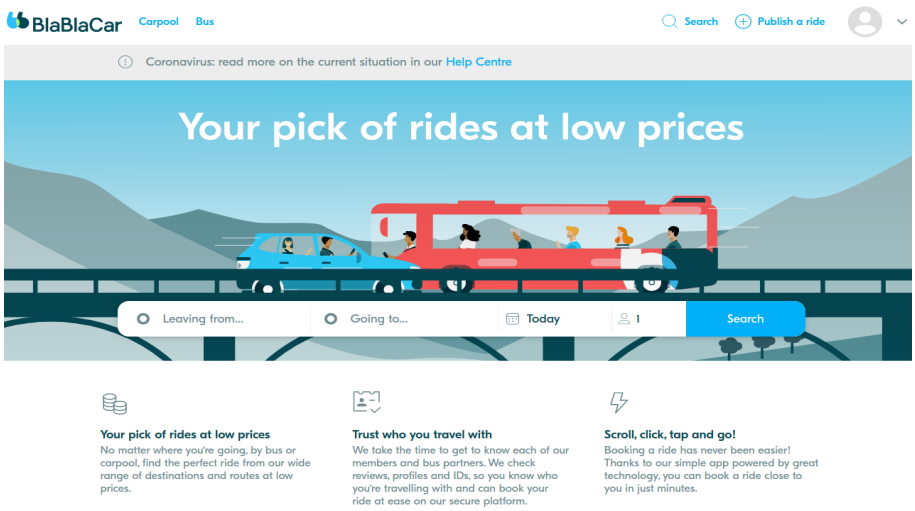
BlaBlaCar je največje svetovno omrežje za deljene prevoze s 100 milijoni uporabnikov v 22. državah, med katerimi pa še ni Slovenije. Glavne storitve aplikacije so deljeni prevozi (kjer vozniki zapolnijo svoj avto) in ponudba avtobusnih prevozov. Na ta način skrbijo za ugodnejša potovanja in zmanjšanje onesnaževanja. Po njihovih ocenah na leto prihranijo 1,6 milijona ton CO2 in omogočijo 120 milijonov prevozov na leto. [3]

Začetki projekta segajo v leto 2003. Ustanovitelj podjetja takrat ni imel osebnega avtomobila in ker so bili javni prevozi vsi zasedeni, je morala priti ponj njegova sestra. Med vožnjo je opazil, da se veliko ljudi vozi v nezasedenih avtomobilih in dbbil idejo,

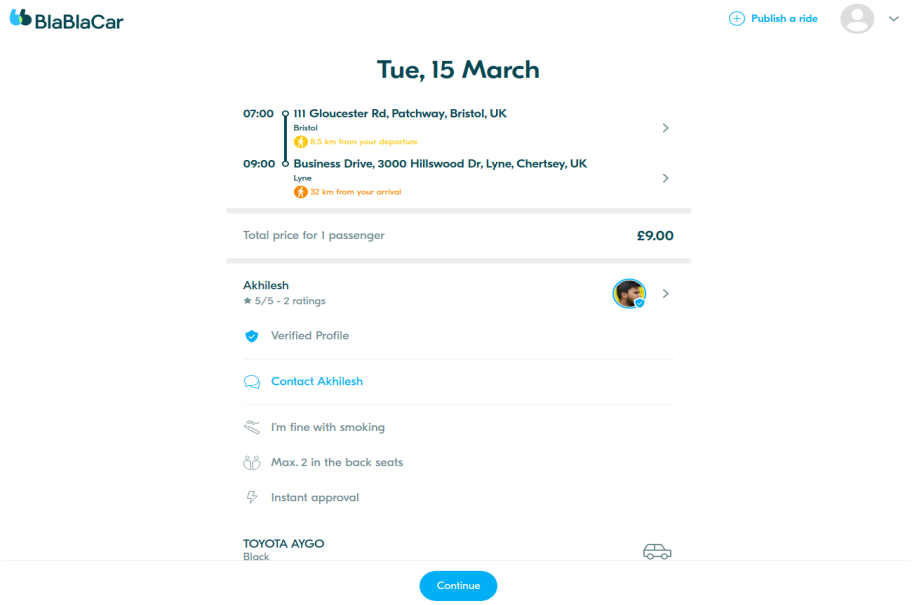
ki je nato v naslednjem desetletju prerasla v vodilno platformo za deljene prevoze na svetu [3]. Podjetje BlaBlaCar, ki si tudi lasti aplikacijo, deluje zgolj kot posrednik med vozniki in potniki. Podjetje tako nima v lasti prevoznih sredstev. Glavni vir prihodkov so provizije od vsakega prevoza, te pa se gibljejo med 18 % in 21 % na prevoz [5].

Aplikacija BlaBlaCar je precej bolj dodelana in kompleksna od recimo aplikacije Prevoz.org in tudi naše aplikacije Sopotnik. Ker za aplikacijo stoji profesionalno vodena ekipa in celotno podjetje, imajo veliko več sredstev za razvoj programske opreme. Način delovanja aplikacije je načeloma podoben aplikacijama Prevoz.org in Sopotnik. Na začetni strani je uporabniku prikazan iskalnik prevozov (Slika 12), kjer le ta lahko vnese lokaciji in dan prevoza. Uporabnik nato med seznamom prevozov izbere željeni prevoz in pregleda podrobnosti (Slika 13).

Da se uporabnik lahko prijavi na prevoz, mora prej potrditi svojo telefonsko številko, el. naslov in tudi vnesti podatke svoje kreditne/debetne kartice ali pa paypal računa. Ko se uporabnik/potnik, prijavi na vožnjo in ga voznik potrdi, se potniku odštejejo sredstva z računa v višini cene prevoza. Ta sredstva se začasno hranijo na računu BlaBlaCar. Ko se prevoz konča, prejme voznik sredstva na svoj račun. Na ta način se zavaruje tako voznike in potnike, saj če se potnik ne pojavi na prevozu brez razloga, se nakazana sredstva še vedno prenesejo vozniku in tako potnik nima možnosti za goljufijo. Poleg tega, aplikacija BlaBlaCar vsebuje tudi ocenjevanje potnikov in voznikov ter odziv na te ocene [4].



Slika 12: Domača stran aplikacije BlaBlaCar [5].



Slika 13: Podrobnosti o prevozu na BlaBlaCar [3].

2.5 Predlagane izboljšave

Navedli bomo nekaj lastnosti aplikacij za deljene prevoze, ki jih imamo namen izboljšati. Predlagane izboljšave bomo primerjali predvsem s funkcionalnostmi aplikacije Prevoz.org. Predlagane izboljšave bodo izboljšale uporabniško izkušnjo in samo delovanje sistema.

Izboljšave in dodatne funkcionalnosti bodo bolj podrobno opisane med opisom funkcijskih zahtev aplikacije Sopotnik (Poglavje 4.1).

Glavne izboljšave v primerjavi predvsem z aplikacijo Prevoz.org, so:

- Možnost clotne interakcije med uporabniki prek spletne aplikacije. Aplikacija bo nadomestila klice in SMS sporočila za usklajevanje prevozov s spletno prijavo/odjavo in sprejemanjem prevozov.
- Ocenjevanje tako potnikov kot voznikov na prevozih, ki se bo opravljalo po končanih prevozih. Ta funkcionalnost bo nadomestila ocenjevanje (samo) voznikov.
- Posodobitev vizualne podobe spletne aplikacije. Torej posodobitev grafičnega vmesnika, za bolj sodoben izgled.
- Olajšan vnos podatkov, kar pomeni, da bo lahko uporabnik v aplikaciji usrvaril in shranil svoja provozna sredstva in njihove opise. Tako mu ne bo treba pisati istih podrobnosti pri vnosu vsakega prevoza.

- Sistemsko ocenjevanje uporabnikov. Aplikacija bo zaznavala dejavnosti uporabnikov in glede na to dodeljevala sistemske ocene, ki bodo vidne registriranim uporabnikom. Na takšen način bomo lahko rešili težavo odpovedi prevozov v zadnjem trenutku, objave več prevozov ob istem času itn.

3 Študija izvedljivosti

3.1 Tehnična izvedljivost

Implementacija aplikacije je tehnično izvedljiva s številnimi odprtokodnimi in prosto dostopnimi tehnologijami, ki jih bomo natančneje opisali tudi v nadaljevanju. Ker obstaja veliko tehnologij za razvoj spletnih aplikacij jih bomo izbrali na podlagi tehnoloških rešitev, možnosti in sodobnosti, ki jih bomo preučili preko različnih forumov in po pogovoru s strokovnjaki, ki se ukvarjajo s spletnim oblikovanjem. Glede na pestrost izbire, ne pričakujemo težav pri izboru primernih rešitev za implementacijo aplikacije s funkcionalnimi in nefunkcionalnimi zahtevami, ki jih bomo definirali v poglavju 4.

Za objavo spletne aplikacije bomo potrebovali gostovanje na strežniku, za kar obstaja tudi nekaj ugodnih in brezplačnih rešitev. Ker bo aplikacijo razvijala ena sama oseba, bo morda razvoj aplikacije dolg, lahko tudi do pol leta.

3.2 Ekonomska upravičenost

Namen izdelave spletne aplikacije Sopotnik je predstavitev te kot diplomsko delo. Zaradi tega ne pričakujemo, da bi bila aplikacija objavljena v javno uporabo ali bila celo dobičkonosna. Cilj projekta je usvojiti novo znanje ter predstaviti idejo.

Za implementacijo projekta bomo uporabljali zgolj brezplačne tehnologije in rešitve. Kljub temu bodo verjetno nastali določeni manjši stroški. Stroški se bodo najverjetneje pojavili pri nakupu domene in gostovanja. Dodatni stroški bi lahko nastali tudi ob povečanem številu zahtevkov na zunanje aplikacijske vmesnike (API), ki so potrebni za delovanje aplikacije.

3.3 Pravni in etični vidik

Aplikacija bi po pravni plati delovala kot obstoječa aplikacija Prevoz.org. Ker bi bila aplikacija neprofitna, za delovanje te ne bi bila potrebna dodatna dokumentacija ali celo ustanovitev pravne osebe oziroma dejavnosti. Plačila med uporabniki se opravljajo z gotovino in so obravnavana kot medsebojna pomoč, ki ne terja davkov. Hkrati je tudi prispevek avtorja aplikacije prostovoljen. Avtor od aplikacije pridobi le dobro referenco za morebitne zaposlitve v prihodnosti. Aplikacija ni moralno ali etično sporna.

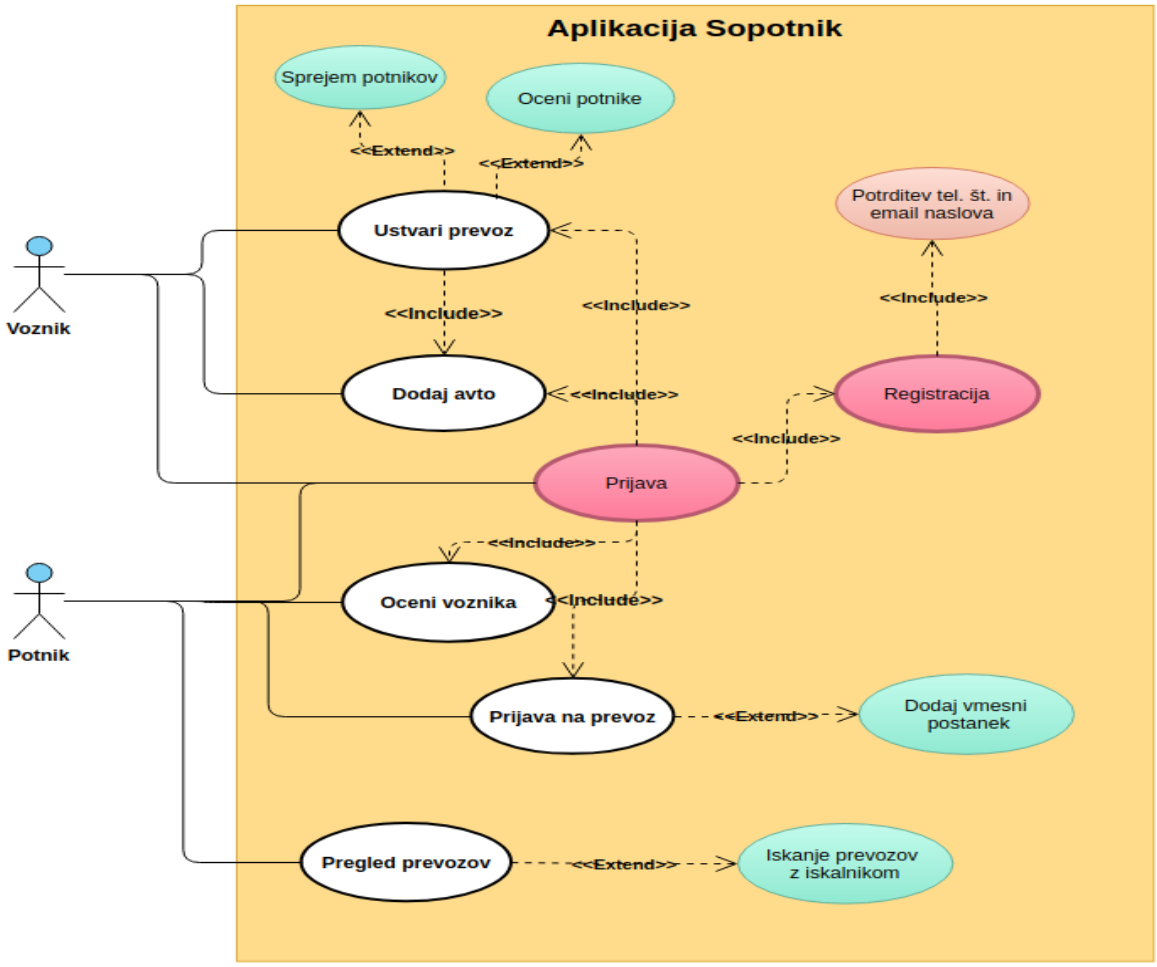
4 Analiza in definiranje zahtev

Postopke analize in definiranja zahtev nam pomaga izboljšati razumevanje celotnega projekta in pomaga pri uklajevanju želja ter zahtev stranke. Posledično tako dobimo kakovostnejši končni izdelek, ki bo zadovoljil stranko. V tem koraku razvoja programske opreme analiziramo in opišemo zahteve programskega izdelka, tako da so te točne, nedvoumne, konsistentne in realne [12, 44].

V nadaljevanju poglavja bomo predstavili vse tri vrste zahtev. Torej funkcijske, nefunkcijske in sistemske zahteve. Pri opisu funkcijskih zahtev si bomo pomagali z UML diagrami [37, 41], kjer bomo uporabili vedenjske diagrame za predstavitev interakcije uporabnika z aplikacijo. Med nefunkcijskimi in sistemskimi zahtevami pa bomo opisali sistem v kontekstu zmogljivosti, varnosti itn.

4.1 Funkcijske zahteve

Z UML diagramom primera uporabe [41] lahko učinkovito podamo zunanji pogled na sistem ter prikažemo glavne uporabniške zahteve. S tem modelom lahko preprosto prikažemo interakcijo uporabnikov s sistemom.



Slika 14: Diagram primera uporabe (angleško Use case diagram) aplikacije Sopotnik [41].

Na UML diagramu primera uporabe na Sliki 14 lahko opazimo glavne funkcijske zahteve aplikacije Sopotnik. Uporabnike aplikacije smo razdelili na dve skupini: voznike in potnike. Kot lahko vidimo, je za vse funkcionalnosti, razen za pregled prevozov potrebna prijava, ki je možna po predhodni registraciji. Za registracijo je potrebna tudi potrditev naslova elektronske pošte in telefonske številke.

Voznikom sta omogočeni dve glavni funkcionalnosti: ustvarjanje prevozov in dodajanje prevoznega sredstva (avtomobil, kombi ipd.) Voznik lahko ustvari prevoz, če ima pred tem že vnesene podatke o vsaj enem prevoznem sredstvu. Na novo ustvarjenem prevozu lahko nato sprejema potnike in jih po koncu tudi ocenjuje (Slika 14).

Potniku je ob prijavi omogočena prijava na prevoz ter ocenjevanje voznika. Potnik lahko po želji oz. prijavi na prevoz doda tudi prošnjo za vmesni postanek. Za pregled prevozov in iskanje teh s pomočjo filtra, potnik ne potrebuje prijave (Slika 14).

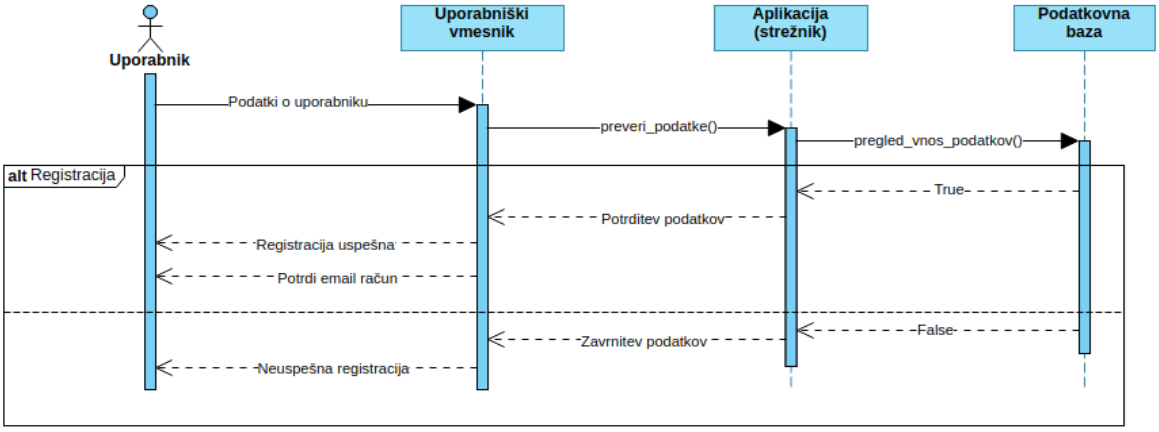
Glavne funkcijske zahteve aplikacije Sopotnik so:

- registracija v aplikacijo,

- prijava,
- objava novega prevoza,
- iskanje ustreznega prevoza,
- prijava potnika na prevoz,
- sprejem potnika na prevoz (s strani voznika),
- urejanje in izbris prevoza,
- odjava od prevoza (potnik se sam odjavi od prevoza),
- odjava potnika s prevoza (voznik odstrani potnika),
- vnos lokacije vmesnega postanka,
- ocenjevanje potnika,
- ocenjevanje voznika,
- ogled uporabniškega profila,
- vnos podatkov o prevoznem sredstvu,
- sprememba gesla,
- sprememba podatkov uporabnika,
- izbris uporabniškega profila.

S pomočjo sekvenčnega diagrama (UML sequence diagram) [42] bomo predstavili nekaj glavnih funkcijskih zahtev, ki so najbolj pomembne za delovanje aplikacije.

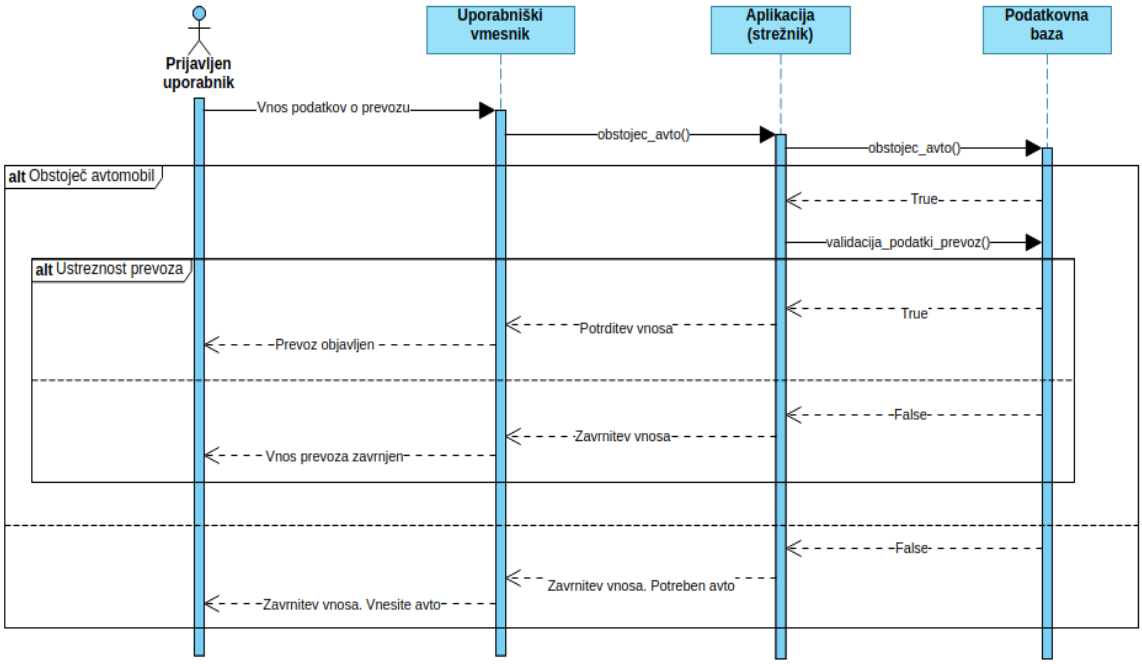
4.1.1 Registracija novega uporabnika



Slika 15: Sekvenčni diagram [42] registracije novega uporabnika.

Slika 15 prikazuje postopek registracije novega uporabnika v aplikaciji Sopotnik. Uporabnik najprej vnese osebne podatke v spletni obrazec. Ko uporabnik potrdi podatke, te prejme aplikacija, ki pregleda njihovo ustreznost. Nato aplikacija pošlje podatke podatkovni bazi, kjer se še enkrat preveri ustreznost podatkov v skladu z omejitvami podatkovne baze. Če so podatki ustrezni, se vnesejo v podatkovno bazo. Ta sporoči aplikaciji status vnosa. Aplikacija nato vrne uporabniku sporočilo. Če so bili podatki ustrezni, je registracija uspešna in uporabnik lahko potrdi svoj uporabniški račun prek elektronske pošte, če pa niso, prejme uporabnik sporočilo o napaki.

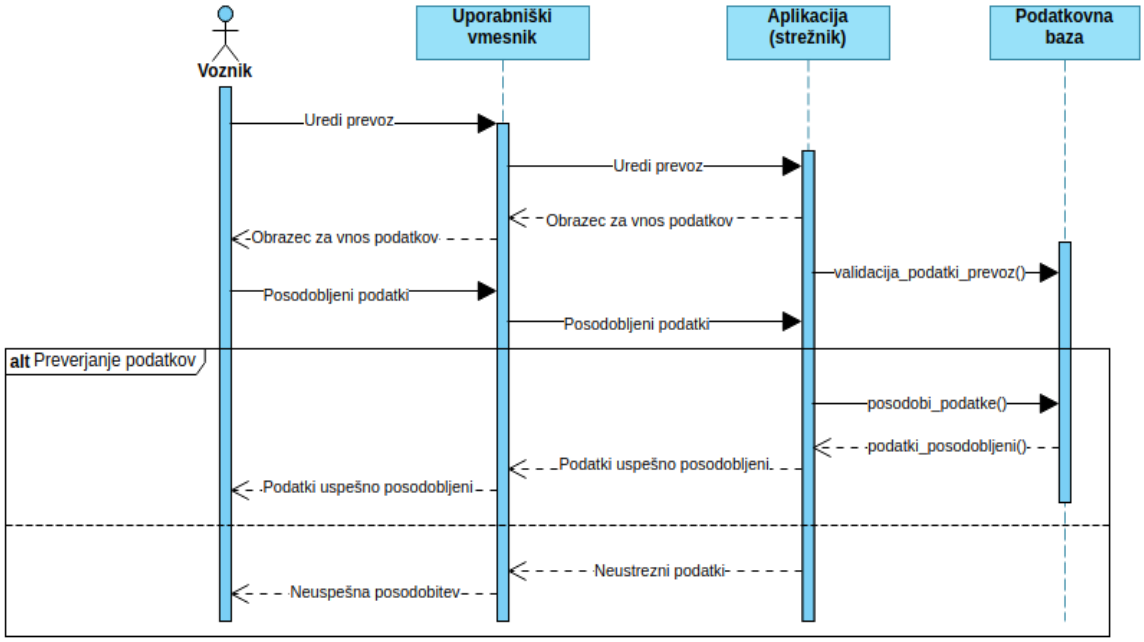
4.1.2 Objava novega prevoza



Slika 16: Sekvenčni diagram [42] vnosa novega prevoza.

Slika 16 prikazuje postopek objave novega prevoza s strani uporabnika (vozika). Uporabnik mora biti za to funkcionalnost prijavljen v aplikacijo, kar predstavlja ime "prijavljen uporabnik". Preden aplikacija ponudi uporabniku obrazec za vnos prevoza, preveri, ali ima uporabnik že vpisane podatke o vsaj enem prevoznem sredstvu. Če uporabnik še nima vpisanih podatkov o prevoznem sredstvu, ga preusmeri na stran za vnos prevoznega sredstva in prekliče ustvarjanje novega prevoza. Če pa je uporabnik že vpisal podatke o prevoznem sredstvu, mu aplikacija vrne obrazec, kamor bo uporabnik lahko vpisal podatke o novem prevozu. Uporabnik nato izpolni spletni obrazec in potrdi objavo prevoza. Aplikacija preveri določene parametre prevoza (datum, prekrivanje z drugimi prevozi) in pošlje podatke podatkovni bazi, da se shranijo. Ob ustreznem postopku voznik prejme potrditev, da je bil prevoz ustvarjen. V nasprotnem primeru aplikacija uporabniku prikaže napako in ga vrne na izpolnjevanje obrazca.

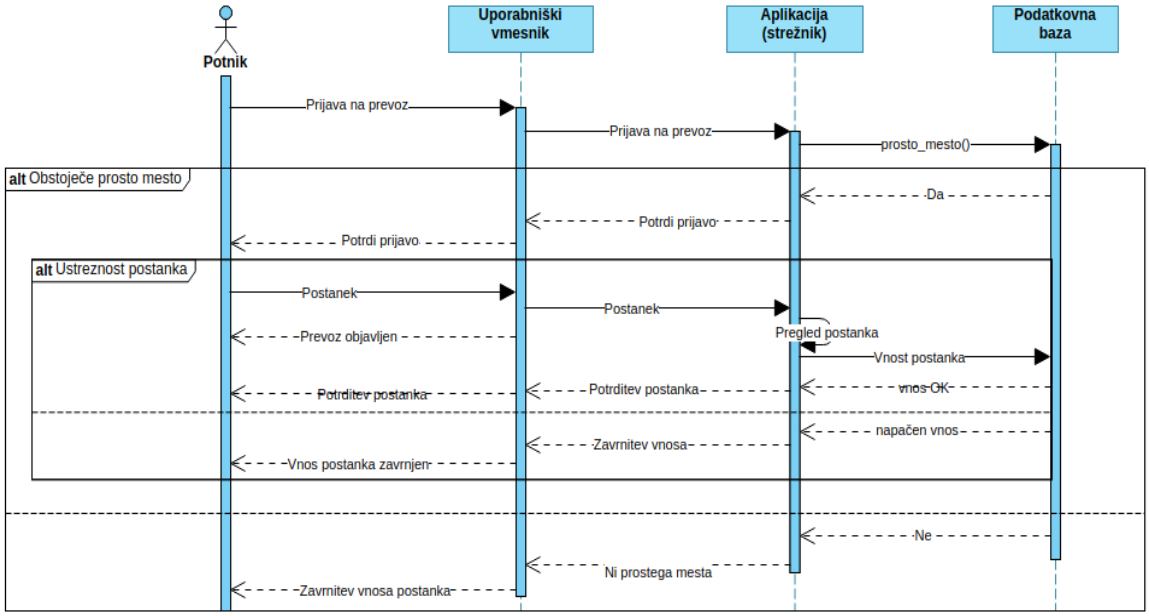
4.1.3 Urejanje podatkov prevoza



Slika 17: Sekvenčni diagram [42] urejanja prevoza.

Slika 17 prikazuje potek urejanja podatkov prevoza. Voznik lahko pred začetkom prevoza spreminja podatke o prevozu tako, da klikne na temu namenjen gumb. Nato mu aplikacija vrne obrazec za vnos posodobljenih podatkov. Voznik vnese posodobljene podatke in jih pošlje aplikaciji. Aplikacija preveri ustreznost podatkov, npr. ustreznost datuma, števila oseb, prekrivanja prevozov. Če so podatki ustrezni, jih pošlje podatkovni bazi, kjer se shranijo, če ne, pa vrne vozniku opozorilo o napaki.

4.1.4 Prijava potnika na prevoz in dodajanje vmesnega postanka



Slika 18: Sekvenčni diagram [42] prijave na prevoz in vpis postanka.

Na Sliki 18 sta prikazana postopka prijave potnika na prevoz in dodajanja postanka na prevozu. Potnik se najprej prijavi za prevoz. Aplikacija pošlje poizvedbo o prevozu v podatkovno bazo ter nato preveri, ali so na prevozu še prosta mesta. Če so na prevozu še prosta mesta, se potrdi prijava za prevoz in tako je uporabniku omogočeno dodajanje vmesnega postanka. Potnik lahko nato preko obrazca pošlje aplikaciji podatke o vmesnem postanku. Aplikacija pregleda vpisane podatke in jih nato pošlje podatkovni bazi. V primeru ustreznosti podatkov vrne podatkovna baza potrditev aplikaciji, ki nato obvesti potnika, da je bil vnos vmesnega postanka ustrezen. Če so podatki postanka napačni, aplikacija vrne potniku obvestilo o napaki.

4.2 Nefunkcijske zahteve

Nefunkcijske zahteve so zahteve in omejitve, ki predstavljajo zmožnosti in omejitve sistema [31]. V grobem jih delimo na obratovalne (poglavje 4.2.1) in evolucijske (poglavje 4.2.3) lastnosti.

4.2.1 Obratovalne lastnosti

Varnost

Varnost je eden glavnih vidikov aplikacije, saj mora uporabik za potrditev prevoza v aplikaciji vpisati precej osebnih podatkov. Za ustrezno upravljanje s podatki uporabnikov, bomo upoštevali usmeritve GDPR. Med glavnimi pravicami uporabnikov po uredbi GDPR, so med drugim pravice do dostopa, izbrisa podatkov, poprave podatkov in preklica samodejne obdelave podatkov [11].

V aplikaciji Sopotnik bomo zagotovili transparentno upravljanje z uporabniškimi podatki na naslednje načine:

- Obdelava in vnos podatkov, ki so izključno pomembni za nemoteno delovanje aplikacije.
- Možnost urejanja in brisanja osebnih podatkov s strani uporabnika.
- Kriptiranje pomembnih podatkov (gesla).
- Omejen dostop do uporabniških podatkov. Dostop do vseh podatkov profila je omogočen samo registriranim in verificiranim uporabnikom.
- Aplikacija Sopotnik ne bo delila osebnih podatkov uporabnikov s tretjimi osebami.

Uporabnost

Uporabnost aplikacije določa zmožnost, da uporabnik čim lažje upravlja z aplikacijo. To pomeni, da se lahko uporabnik nauči uporabljati aplikacijo brez težav [32]. Za aplikacijo Sopotnik načrtujemo tako zasnovo, da je uporaba aplikacije intuitivna. Določen tok aktivnosti je podoben aplikaciji Prevoz.org [35] in tudi drugim podobnim spletnim aplikacijam. Da bi uporabnik razumel določene podrobnosti glede uporabe aplikacije, bo na voljo zavihek z dodatnimi informacijami o delovanju in pravilni uporabi aplikacije.

4.2.2 Dostopnost in razpoložljivost

Dostopnost in razpoložljivost določata s kakšno lahkoto uporabnik uporablja sistem in kdaj je sistem razpoložljiv za uporabo [32]. Aplikacija Sopotnik bo morala biti neprestano dostopna, zaradi potrebe po vnosu prevozov in prijavi na prevoze v vsakem trenutku. Aplikacija bo morala omogočati več dostopov hkrati, z več različnih naprav.

4.2.3 Evolucijske lastnosti

Možnost vzdrževanja

Možnost vzdrževanja je sposobnost odprave napak v sistemu [32]. Aplikacijo Sopotnik bomo zasnovali tako, da bomo izvorno kodo razdelili v več neodvisnih modulov, kjer bo vsak implementiran na jasen način, glede na standarde. Na ta način bodo lahko izvorno kodo vzdrževali tudi drugi razvijalci, torej ne le avtor aplikacije. Potrebno bo tudi zagotoviti osebje, ki bo izvajalo vzdrževanja po predaji aplikacije v uporabo.

Razširljivost in nadgradljivost

Razširljivost in nadgradljivost je zmožnost posodabljanja sistema, da ta zadostuje potrebam storitve, ki jo podpira/izvaja [31]. Sistem je zasnovan v okolju Laravel [17, 21], ki omogoča strukturiranje kode ter lahko vzdrževanje in nadgrajevanje. Zaradi tega bomo lahko ob veliki količini zbranih podatkov dodajali tudi nove funkcionalnosti ali dodatne informativne in zanimive statistike ter potencialno kakšen priporočilni sistem.

Možnost testiranja

Možnost testiranja je eno glavnih orodij razvijalcev za nadzor in preverjanje razvoja aplikacije. Testiranje, ki ga bo izvajal razvijalec, bo potekalo sproti za vsak modul in komponento. Po prvi končni različici pa bomo dali vsako različico v testiranje tudi tretjim osebam ter tako ugotavljali primernost sistema in zmožnost sistema, da izpolnjuje vse načrtovane funkcionalnosti.

5 Načrtovanje sistema

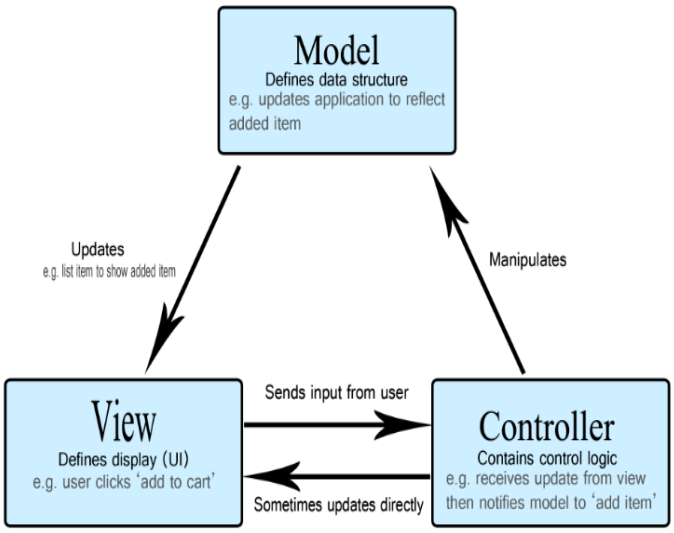
Izdelava programskega izdelka lahko hitro postanke kompleksna naloga, zato je potrebno določeno načrtovanje projekta. Bolj kot je obsežen projekt, več načrtovanja in priprav je potrebnih, da izvedba poteka po načrtu. Z ustreznim načrtovanjem procesa izdelave programskega izdelka si lahko prihranimo veliko časa pri razvoju. Hkrati nam načrtovanje tudi zmanjša možnost za napake, zniža stroške, zmanjša čas razvoja ter omogoči strukturiran in pregleden proces razvoja izdelka.

5.1 Arhitekturni vzorec sistema

Za arhitekturo sistema se bomo oprli na arhitekturni vzorec Model-Pogled-Krmilnik znan pod kratico MVC (angleško Model-View-Controller). MVC razdeli aplikacijo na 3 glavne komponente:

- Model
  - Komponenta Model izvaja poizvedbe za podatki v podatkovni bazi ter omogoča shranjevanje in urejanje podatkov ter nekaj logičnih opreacij nad podatki (Slika 19) [25].
- Pogled
  - Komponenta Pogled določa izgled in predstavitev grafičnega vmesnika aplikacije. S to komponento je omogočena interakcija človeka z aplikacijo. Uporabnik v grafični vmesnik vnaša podatke, ki se nato obdelajo v drugih dveh komponentah aplikacije. Aplikacija po drugi strani v komponenti Pogled predstavi podatke uporabniku (Slika 19) [25].
- Krmilnik
  - Krmilnik je vmesnik med komponentama Model in Pogled, ki sprocesira vso logiko in podatkovne zahteve aplikacije. Krmilnik poskrbi za urejanje podatkov s pomočjo komponente Model in predstavitev podatkov s pomočjo komponente Pogled (Slika 19). Na primer, Uporabniški Krmilnik (angleško User Controller) bo poskrbel za interakcijo s Pogledom Stranke (angleško Customer View) in poskrbel za obdelavo podatkov iz komponente Pogleda Stranke in potem te obdelane podatke poslal komponenti Uporabniški Model (angleško User Model), ki bo podatke o uporabniku shranila v podatkovno bazo (Slika 19) [25].

MVC je eden od najpogosteje uporabljenih razvojnih ogrodij za razvoj spletnih aplikacij in večjih projektov [25], saj povezano programsko logiko razdeli na tri medsebojno povezane elemente, od katerih vsah pravzame točno določeno vlogo.



Slika 19: Arhitekturni model MVC [26].

Ta arhitekturni vzorec smo izbrali zaradi nekaj preteklih izkušenj z razvojem s pomočjo te arhitekture. Prepričala nas je predvsem preprostost razvoja in pregleden način razvoja aplikacij, saj arhitektura na pregleden način loči oblikovanje (Pogled), logiko (Krmilnik) in podatke (Model). Na ta način je lahko dodajati nove funkcionalnosti in testirati obstoječe. S tem vzorcem je olajšano tudi prihodnje vzdrževanje in posodabljanje aplikacije.

5.2 Izbira tehnologij

V tem poglavju bomo opisali tehnologije, ki jih bomo uporabili pri razvoju aplikacije Sopotnik. Izbor tehnologij je opisan v poglavju 3.1. Poleg tega smo pri izbri upoštevali tudi možnost razvoja po arhitekturnem vzorcu MVC.

5.2.1 Laravel

Laravel [17] je brezplačno, odprto-kodno, PHP ogrodje za spletne aplikacije, zasnovano na arhitekturnem vzorcu Model-Pogled-Krmilnik [25]. Avtor ogrodja je Taylor Otwell, ogrodje pa temelji na ogrodju Symfony [27]. Laravel ponuja modularni paketni sistem s posebnim upravljalcem zunanjih knjižnic (angleško Composer dependancy

manager [8]), ki omogoča preprosto nalaganje zunanjih knjižnic in vmesnikov v lokalno aplikacijo. Laravel nudi različne načine dostopa do podatkovne baze, pomožne programe (angleško utilities) za predajo aplikacije v produkcijo in predprocesiranje PHP jezika za lepšo sintakso [21].

V ogrodju Laravel bomo uporabljali naslednje funkcionalnosti:

- Eloquent ORM
  - Eloquent OMR je napredna PHP implementacija za delo s podatkovno bazo. Omogoča izdelavo Modela 5.1 za vsako tabelo v podatkovni bazi ter prilaga pripadajoče metode, omejitve ter razmerja za uprabljanje objektov (Model) iz podatkovne baze [22]. S to funkcionalnostjo je tako delo s podatki iz podatkovne baze zelo pregledno in preprosto, še posebej pa je preprost pregled nad odvisnostmi med objekti.
- Query builder
  - Laravel Query builder omogoča preprost vmesnik za kreiranje in izvajanje poizvedb v podatkovno bazo. Lahko se ga uporabi za izvedbo večine operacij nad podatkovno bazo in deluje s podatkovnimi bazami, ki jih podpira Laravel (MySQL, Postgres, SQLite, and SQL Server). Ta vmesnik omogoča tudi vezavo parametrov (angleško parameter binding), ki varuje aplikacijo pred napadi kot je SQL vrinjanje (angleško SQL injecitons) [24].
- Migrations
  - Laravel migracije (angleško migrations) so sistem verzificiranja shem podatkovne baze. S tem sistemom lahko na preprost način uvajamo spremenbe v tabele v podatkovni bazi, dodajamo/brišemo tabele itn. [23].
- Jetstream paket
  - Jetstream je Laravelov paket, ki mogoča začetno izhodišče za izdelavo aplikacije, saj nudi implementacije za prijavo, registracijo, potrjevanje el. računov, dvostopenjsko avtentikacijo (angleško two-factor authentication) in nadzor nad sejami [19].

5.2.2 Bootstrap in Javascript

Za oblikovanje izgleda spletne aplikacije smo si izbrali ogrodje Bootstrap [7]. Bootstrap vsebuje HTML, CSS in JavaScript predloge/šablone za oblikovanje izgleda spletne strani. V naši aplikaciji smo za grobo oblikovanje strani uporabili Bootstrap, za

natančnejše oblikovanje pa dodatno CSS in JavaScript. Programski jezik Javascript smo uporabili predvsem za dinamične dogodke na spletni strani in za klice zunanjih aplikacijskih vmesnikov (glej poglavje 5.4).

5.2.3 PHP

PHP ("Hypertext preprocessor") [34] je programski jezik, katerega glavni namen je implementacija logike za spletne strani. V tem jeziku je spisano tudi ogrodje Laravel, ki ga uporabljamo za implementacijo aplikacije Sopotnik. PHP koda se ponavadi izvaja na strežnikih s pomočjo PHP prevajalnika (angleško interpreter). Prevedena koda je lahko različnih tipov, od HTML kode do binarnih podatkov. Prav zaradi raznolikosti jezika je ta uporabljen v veliko predlogah za spletne strani (angleško template systems), npr. Wordpress [46] in Joomla [20, 34]. Po podatkih strani W3tech je PHP še vedno z naskokom najbolj uporabljen programski jezik na strežnikih, saj kar 77,7 % spletnih strani uporablja PHP [45].

5.2.4 GIT in Github

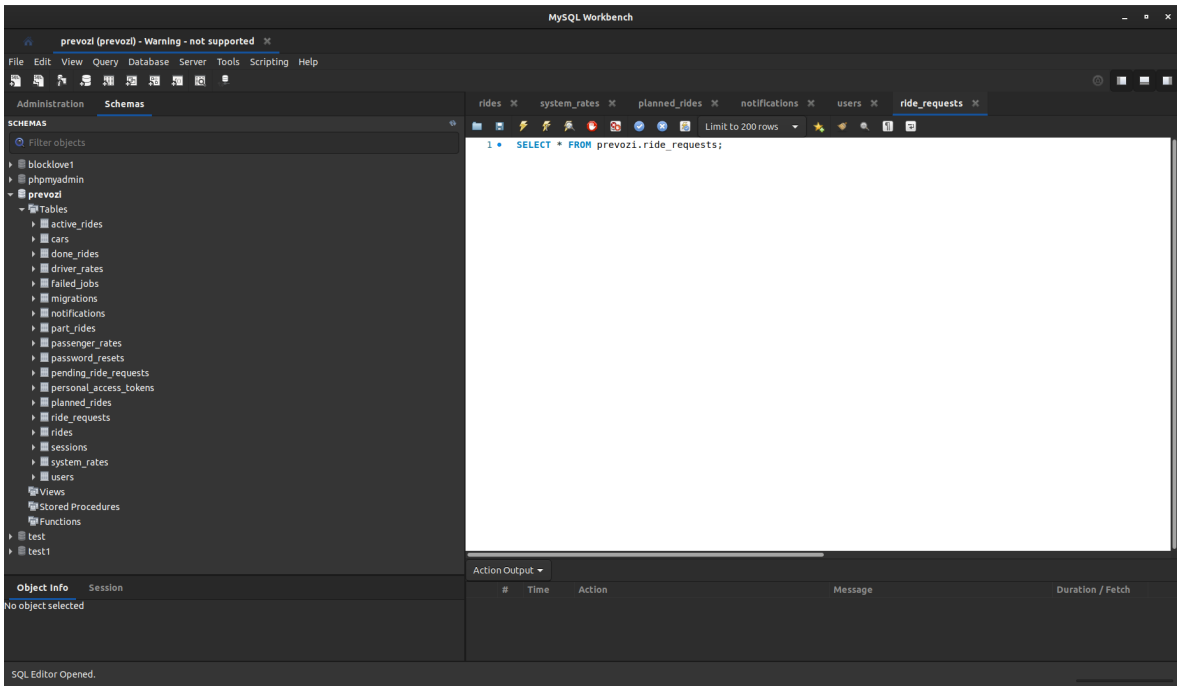
Pri izvedbi, še posebej večjih projektov, je zelo pomemben nadzor nad potekom projekta. Treba je imeti pregled nad trenutnim napredkom in različicami programa. To omogočajo sistemi za nadzor različic (angleško version control systems - VCS), med katerimi je najbolj znan program Git [13], ki ga bomo uporabili pri izdelavi aplikacije Sopotnik. Uporabili bomo GIT CLI različico [13], ki je nemenjena uporabi v ukazni vrstici (angleško command line interface ali CLI).

S programom Git smo si večinoma ustvarjali potrditve kode po vsakem končanem koraku (angleško commit), ter si izdelali potek izdelave v veje (angleško branch). Za potrebe ustvarjanja varnostne kopije kode smo kodo z lokalnega repozitorija prenesli tudi na oddaljeni (angleško remote) spletni repozitorij Github [14]. Github in Git omogočata tudi sodelovanje med razvijalci, česar sicer med razvojem tega projketa nismo potrebovali.

5.2.5 MySql in Xaamp

MySQL je odprtokodni sistem za upravljanje s podatkovnimi bazami. Je implementacija relacijske podatkovne baze, ki uporablja jezik SQL [28]. MySql deluje po načelu odjemalec-strežnik (angleško client-server), kjer je lahko strežnik kot sistem, porazdeljen na več strežnikov. MySql ponuja veliko zbirko ukazov in programskih vmesnikov za dostop do podatkovne baze. V upravljanju ga ima Oracle corporation iz Združenih držav Amerike [28]. Za lažjo uporabo programa MySql smo uporabljali grafični vme-

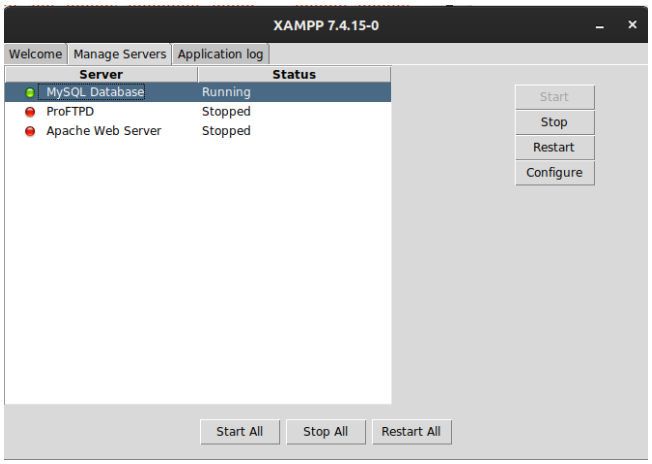
snik MySql Workbench [29] (Slika 20), ki omogoča grafičen pogled na tabele in ročno spreminjanje podatkom med lokalnim razvojem aplikacije.



Slika 20: Izgled programa MySql Workbench [29].

5.2.6 Strežnik

Za lokalni razvoj smo kot strežnik uporabili paketno rešitev spletnega strežnika Xaamp (Slika 21), ki zažene lokalni strežnik MySql [28] in spletni strežnik Apache strežnik s podporo PHP [47].



Slika 21: Izgled grafičnega vmesnika programa Xaamp z različnimi strežniki, ki jih lahko zaženemo preko vmesnika [47].

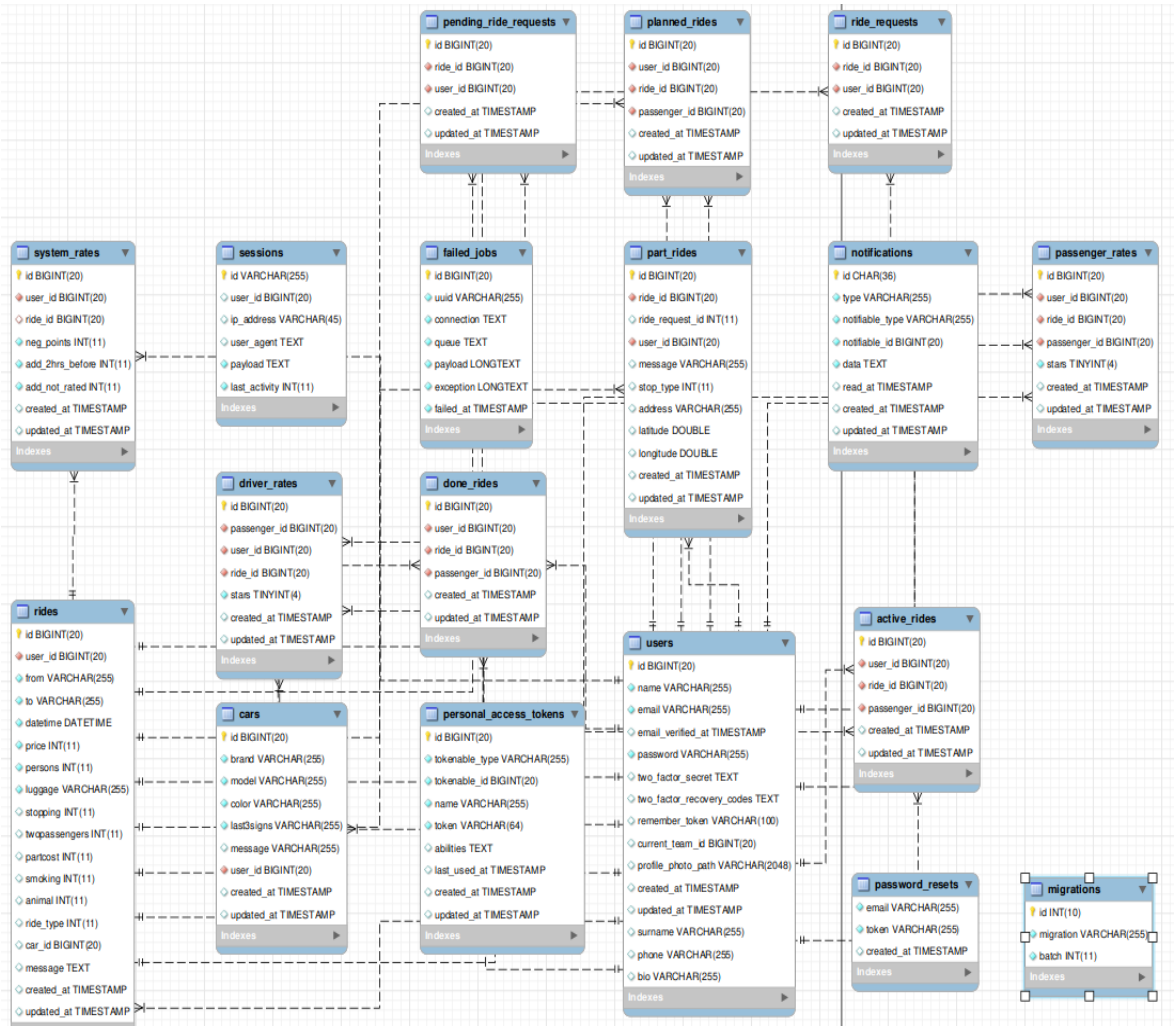
5.2.7 Visual paradigm

Spletni program Visual paradigm smo uporabili za načrtovanje sistema. Predvsem nam je koristil za načrtovanje funkcijskih zahtev ter za izdelavo sheme podatkovne baze. Visual paradigm je orodje za ustvarjanje UML [41] diagramov, generiranje programske kode, obratno inžinirstvo generiranja diagramov iz kode in podobne uporabne storitve [43].

5.3 Shema podatkovne baze

Kreiranje sheme podatkovne baze je bila ena od najbolj zahtevnih stvari pri načrtovanju sistema. Treba je bilo namreč upoštevati glavna pravila za vzpostavitev kakovostvne sheme podatkovne baze:

- Izdelava premišljenih relacij med tabelami.
- Primerno določeni dogodki pri brisanju in urejanju zapisov (angleško "on delete" in "on update" ukazi).
- Normalizacija podatkovne baze [9, 18].
- Izbira primernih atributov za opis posameznih objektov.



Slika 22: Shema (entitetno relacijski diagram) podatkovne baze aplikacije Sopotnik.

Na Sliki 22 lahko vidimo vse tabele in odvisnosti med njimi.

5.3.1 Tabele podatkovne baze

Uporabniki (tabela `users`)

V tej tabeli so glavni podatki uporabnika. To so torej ime, priimek, naslov elektronske pošte, geslo, telefonsla številka, opis osebe ter ostali podatki, pomembni za verifikacijo profila in prijavo.

Prevozi (tabela `rides`)

V tabeli prevozi so zbrani glavni podatki o prevozu. To je kraj odhoda in prihoda, datum, cena in ostale specifikacije prevoza, kot so na primer, ali je omogočeno kajenje, vmesno pobiranje ali prevoz hišnega ljubljenčka. Tabela vsebuje tuji ključ `user_id`, ki predstavlja enolični identifikator osebe, ki je vozik na prevozu in je vnesla prevoz.

Prevozna sredstva (tabela `cars`)

Tabela hrani podatke o prevoznih sredstvih uporabnikov. Kot lahko vidimo na Sliki 22, vsebuje podatke o znamki, modelu, registrski številki, barvi prevoznega sredstva ter kratek opis avta. Vsebuje tudi identifikator osebe, ki je ustvaril prevozno sredstvo (`user_id`). Preko tega identifikatorja lahko tako dostopamo do celotnega objekta uporabnika.

Prijave na prevoz

Za prijave na prevoz imamo dve tabeli, in sicer tabelo z aktivnimi prijavami na prevoz (`ride_requests`) ter tabelo z začasno umaknjenimi prošnjami (`pending_ride_requests`). V tej tabeli hranimo prošnje za prevoz, ki jih je imel uporabnik za prevoze, ki so ob istem času kot prevoz, na katerega je bil sprejet. Če se uporabnik morda odjavi, ali pa je odstranjen s prevoza, se mu ti vnosi iz `pending_ride_requests` prenesejo v tabelo `ride_requests` (Slika 22).

Potniki na prevozu

Za hranjenje podatkov o potnikih na prevozih uporabljamo tri tabele, da lahko tako spremljamo prevoze, ki se še niso začeli (`planned_rides`), prevoze, ki potekajo (`active_rides`) in kočane prevoze (`done_rides`). Tabele predvsem povezujejo tabele uporabnika (`users`) in prevozov (`rides`), saj vsebujeta tuje ključe, ki predstavljajo identifikatorja prevoza in potnika (Slika 22).

Vmesni postanki

Tabela z vmesnimi postanki (`part_rides`) (Slika 22) vsebuje podatke o vmesnih postankih, ki jih potnik vnese na prevozu ob prijavi na prevoz. Tabela vsebuje identifikatorje potnika, prevoza in prijave na prevoz ter ostale podatke o vmesnem postanku. To so naslov, koordinati, dodaten opis in informacija, ali se tam potnika pobere ali odloži (atribut `stop_type`) (Slika 22).

Ocene

Ocene voznika (tabela `driver_rates`): v tabeli ocene voznika hranimo identifikatorje voznika, potnika in prevoza ter oceno, ki jo je vozniku podal potnik (Slika 22).

Ocene potnika (tabela `passenger_rates`): tabela ocene potnika hrani identifikatorje potnika, voznika in prevoza ter oceno, ki jo je potniku podal voznik.

Sistemske ocene (tabela `system_rates`): tabela sistemske ocene hranijo podatke o točkah, ki jih aplikacija samodejno dodeli uporabiku za vsak prevoz. S temi točkami se nato dela nadaljna statistika, ki omogoča vpogled v aktivnost uporabnikov.

5.4 Zunanji aplikacijski vmesniki API

V aplikaciji smo za potrebe samoizpolnjevanja obrazcev (angleško form autocomplete) in prikaz lokacije na zemljevidu uporabili dva Googlova aplikacijska vmesnika (API). Google-ove storitve so za majhno število zahtev (requests) še brezplačne, ko pa preidejo nekaj tisoč zahtevkov, postanejo plačljive. Trenutno smo aplikacijski vmesnik omogočili z osebnimi financami, če pa bi projekt zaživel, bi bilo treba tudi to storitev knjižiti med dodatne stroške.

5.4.1 Google Places API

Aplikacijski vmesnik smo uporabili za samoizpolnjevanje krajev ob vnosu kraja odhoda ter kraja prihoda med filtriranjem prevozov in pri vnosu prevozov. Places API prejema podatke (del besede kraja) preko HTTP zahteve (angleško request) in vrne predloge za celotno besedo, ki jo dopolni [16].

5.4.2 Google Maps API

Ta aplikacijski vmesnik smo uporabili za izris lokacije na zemljevidu. Na tak način si lahko uporabnik lažje predstavlja lokacijo uporabnika. Maps API [15] deluje tako, da sprejme HTTP zahtevek, preko katerega pošljemo koordinate lokacije. Storitev nam nato izriše zemljevid z označeno lokacijo in jo vrne, da jo lahko predstavimo v svojem grafičnem vmesniku.

6 Implementacija

6.1 Priprava razvojnega okolja

Za začetek smo si morali najprej pripraviti lokalno okolje za razvoj. Namestili smo upravljalnik paketov (angleško packet manager) Composer [8], ki je nujno potreben za izdelavo novega Laravel projekta. Nato smo z ukazom `:$ composer create-project laravel/laravel sopotnik` ustvariti nov Laravel [21] projekt z imenom sopotnik. V novi projekt smo se premaknili z ukazom `:$ cd /sopotnik`.

Znotraj projekta sopotnik smo nato inicializirali nov Git [13] repozitorij z ukazom `:$git init` in izvedli prvo potrditev kode z ukazoma `:$ git add .` in `:$ git commit -m "initial commit"`. Nadalje smo se prijavili v osebni Github račun, kjer smo ustvarili nov spletni repozitorij, ki smo ga povezali s pravkar ustvarjenim lokalnim repozitorijem. To smo storili tako, da smo v ukazno vrstico vnesli ukaz :$ git remote add https://github.com/kristoffabjan/prevozi.

Po tem, ko smo si uredili razvojno okolje, smo lahko nadaljevali z izvedbo načrta. Za potrebe upravljanja s podatki uporabnikov ter za prijavo, registracijo in nadzor sej, smo namestili paket Jetstream [19] z ukazom `:$ composer require laravel/jetstream`. Ker potrebuje Jetstream tudi Javascript funkcije, smo morali namestiti še upravljalnik paketov (angleško packet manager) NPM [33]. To smo storili z ukazom `:$ sudo apt-get install npm && npm instll && npm run dev`.

Nato smo morali aplikacijo povezati še s podatkovno bazo. Najprej smo v grafičnemu vmesniku programa MySql Workbench [29] ustvarili novo shemo podatkovne baze. Potem pa smo v .env datoteki aplikacije vnesli podatke o imenu sheme baze ter uporabniško ime in geslo skrbnika baze ter IP naslov računalnika in vrata, na katerih strežnik posluša zahtevke (Slika 23.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=prevozi
DB_USERNAME=root
DB_PASSWORD=
```

Slika 23: Informacije o podatkovni bazi za povezavo aplikacije.

Za dokončno postavitev osnovnih tabel uporabnika, sej in obnovitev gesel, smo še izvedli ukaz `:$ php artisan migrate`. Ta ukaz smo uporabili tudi kasneje ob izdelavi vsake dodatne tabele [23]. Ukaz nam omogoča izvedbo kode v direktoriju migrations, kjer dodajamo nove tabele aplikacije. Nato smo morali zagnati še lokalni strežnik, ki ga

omogoča Laravel. Strežnik smo zagnali z ukazom `:$ php artisan serve`. S tem smo imeli pripravljeno okolje za implementacijo našega načrta, tako da smo lahko začeli dodajati funkcionalnosti, ki smo jih omenili v Poglavju 4.1.

6.2 Implementacija funkcionalnosti

V tem delu bomo opisali, kako smo implementirali nekatere funkcionalnosti. Opisali bomo predvsem naslednje funkcionalnosti:

- mehanizem za prijave na prevoz,
- sistemsko ocenjevanje,
- vmesni postanki,
- prekrivajoči se prevozi.

Pri opisu implementacije funkcionalnosti bomo izpustili generične mehanizme določenih funkcionalnosti, kot na primer ustvarjanje, urejanje in brisanje prevozov, prevoznih sredstev, oseb itn. Opis implementacije teh funkcionalnosti ni tako relavanten, saj izdelava vseh podobnih funkcionalnosti poteka na skoraj enak način. V splošnem se prek HTTP zahteve (angleško HTTP request) pošlje strežniku podatke, ki jih je vnesel uporabnik. Podatki se najprej pregledajo pri odjemalcu, nato pa še na strežniku. Če so podatki v korektnih formatih, se jih zapiše v podatkovno bazo, če ne, pse uporabniku vrne napako. Potek je za urejanje in brisanje skoraj identičen.

Zgoraj navedene funkcionalnost so po drugi strani bolj kompleksne. Za implementacijo je bilo potrebne več logike in odmika od ustaljenih vzorcev. V nadaljevanju bomo na kratko opisali kako smo implementirali funkcionalnosti, ter priložili izvorno kodo.

6.2.1 Mehanizem za prijave na prevoz

Prijave na prevoz v aplikaciji Sopotnik potekajo v glavnem prek spletne aplikacije, opcijsko pa tudi še prek klicev in SMS sporočil. Tukaj bomo opisali prijavo v spletni aplikaciji. Prijava poteka tako, da si uporabnik ogleda prevoze in si izbere tiste, ki mu ustrezajo. Ko se uporabnik prijavi na prevoz, se v podatkovno bazo shrani prošnja za prijavo nanj (Slika 24).

```php
//passenger and ride
public function create(User $user, Ride $ride)
{
    $ride_request = new RideRequest();
    $ride_request->ride_id = $ride->id;
    //passenger
    $ride_request->user_id = $user->id;
    $ride_request->save();

    return redirect()->route('ride', $ride);
}
```

Slika 24: Shranjevanje nove prošnje za prijavo na prevoz.

Uporabnik se lahko hkrati prijavi na toliko prevozov kot želi. Prijava uporabnika (potnika) je nato vidna vozniku na prevozu, ki ga je ustvaril (Slika 35). Ko voznik vidi prošnjo za svoj prevoz, lahko potnika sprejme (če je še dovolj prostih mest na prevozu) na prevoz. Ko je potnik sprejet, se prošnja za prevoz izbriše in ustvari se vnos v tabeli `planned_rides`, kar pomeni, da je potnik potrjen na prevoz. Potniku se, ko je potrjen na določen prevoz, vse ostale prošnje na prevoze v časovnem intervalu, ko bo potekal prevoz, na katerega je bil pravkar sprejet, začasno zbirišejo. V resnici se premaknejo v drugo tabelo `pending_ride_request`, kjer se hranijo za primer, da se potnik odjavi s prevoza, ali pa ga odstrani voznik (Slika 25). Če torej voznik odstrani potnika s prevoza, ali pa to stori potnik sam, se mu spet osvežijo prošnje za ostale prevoze, ki bodo potekali v tem časovnem oknu prevoza, s katerega je bil potnik odstranjen. Na ta način ga bo lahko na prevoz sprejel kdo drug in tako potniku ne bo potrebno treba iskati prevozov (Slika 26).

```php
//get ride_requests that overlap to this ride_request and cut them to pending_ride_request in case passenger gets
//removed from this ride by driver or himself
$overlaping_requests = $user->get_overlapping_ride_requests($ride);
if ($overlaping_requests->count() > 0) {
    foreach ($overlaping_requests as $overlaping_request) {
        //move overlaping request to pending_request table
        $pending = new PendingRideRequest();
        $pending->ride_id = $overlaping_request->ride_id;
        $pending->user_id = $overlaping_request->ride_request_user_id;
        $pending->save();

        //remove ride request that is overlapping
        $conditions = ['ride_id' => $overlaping_request->ride_id, 'user_id' => $overlaping_request->ride_request_user_id];
        $deleted_rows = RideRequest::where($conditions)->delete();
    }
}
```

Slika 25: Začasno hranjenje prošenj za prevoze v istem časovnem intervalu.

```php
//get pending_ride_requests that overlap to this ride_request and copy paste them to back ride_request in case passenger gets
//removed from this ride by driver or himself
$overlaping_pending_requests = $user->get_overlapping_pending_ride_requests($ride);
//dd($overlaping_pending_requests);
if ($overlaping_pending_requests->count() > 0) {
    //pending_ride-requests
    foreach ($overlaping_pending_requests as $overlaping_request) {
        //move overlaping request to pending_request table
        $ride_request = new RideRequest();
        $ride_request->ride_id = $overlaping_request->ride_id;
        $ride_request->user_id = $overlaping_request->user_id;
        $ride_request->save();

        //remove pending_ride_request that is overlapping
        $conditions = ['ride_id' => $overlaping_request->ride_id, 'user_id' => $overlaping_request->user_id];
        $deleted_rows = PendingRideRequest::where($conditions)->delete();
    }
}
```

Slika 26: Osvežitev prošenj za prevoze v istem časovnem intervalu kot preklican prevoz.

Ko je enkrat trenutni čas večji ali enak času začetka prevoza, se tako potniku kot prevozniku onemogoči prijavljanje/odjavljanje in urejanje prevoza. Aplikacija vsebuje mehanizem, ki vsako minuto preverja, kateri prevozi so se začeli v tej minuti in v skladu s tem načrtovane prevoze razvrsti v aktivne prevoze (Slika 27). Prav tako se podobna aktivnost zgodi, ko mehanizem zazna, da je pretekel določen čas od začetka prevoza. Ta časovni interval smo statično določili na dve uri. Torej, ko potečeta dve uri od začetka prevoza, se smatra, da je prevoz končan. Nato se lahko potnik in voznik medsebojno ocenita, prevoza pa je tudi prikazan na profilih voznika in potnika.

```php
public function handle()
{
    $now = Carbon::now('Europe/Paris');
    $end = $now->addHours(2);
    $planned_rides = DB::table('planned_rides')
        ->join('rides', 'planned_rides.ride_id', '=', 'rides.id')
        ->where([
            ['rides.datetime', '<=', $now],
            // ['rides.datetime', '>=', $end]
        ])
        ->select( 'rides.id',  DB::raw('planned_rides.id as planned_ride_id'), 'planned_rides.user_id', 'planned_rides.passenger_id')
        ->get();

    if ($planned_rides->count() > 0) {
        foreach ($planned_rides as $planned_ride) {
            //add active rides
            $active_ride = new ActiveRides();
            $active_ride->user_id = $planned_ride->user_id;
            $active_ride->ride_id = $planned_ride->id;
            $active_ride->passenger_id = $planned_ride->passenger_id;
            $active_ride->save();

            //delete planned rides
            $planned_ride_to_delete_id = $planned_ride->planned_ride_id;
            $deletedRows = PlannedRides::find($planned_ride_to_delete_id)->delete();
        }
    }else{
        $this->info("No changes");
    }

    $this->info('Successfully planned -> active');
}
```

Slika 27: Prehod prevoza v stanje aktivnega prevoza.

6.2.2 Sistemsko ocenjevanje

Funkcionalnost sistmskega ocenjevanja smo implementirali na način, da se uporabniku dodeljujejo pozitivne in negativne točke, glede na izvajanje določenih aktivnosti v aplikaciji. V začetni različici programa se sistemske ocene podajajo predvsem za odjavljenje od prevozov in prevoz oseb, ki še nimajo ocene, da se spodbudi dotok novih uporabnikov.

Pri odjavljanju potnikov s prevoza preverjamo, koliko časa je še do začetka prevoza. Če je do začetka prevoza manj kot dve uri, prikažemo vozniku opozorilo, da bo prejel -1 točko, če sedaj odjavi potnika. Če nato voznik odstrani potnika, se mu odšteje ena sistemska točka (Slika 28).

```php
//if it is less than 2 hours to ride, give driver -1 point if he removed passenger
//both in timestamp format
$now = Carbon::now('Europe/Paris')->timestamp;
$date = new DateTime($ride->datetime);
$ride_time = $date->getTimestamp();
// driver
$conditions_array = ['ride_id' => $ride->id, 'user_id' => $request->user()->id];
$sys_rate = SystemRates::where($conditions_array)->get();
//if its less than 2 hours till ride
if ( $ride_time - $now < 7200 ) {
    //if there is no sys rate for this user and ride yet, create new and assign -1 point
    if ($sys_rate->count() == 0) {
        //create sysrate for this user(driver) and give -1 point in negative points
        $new_sys_rate = new SystemRates();
        //this is driver
        $new_sys_rate->user_id = $request->user()->id;
        $new_sys_rate->ride_id = $ride->id;
        $new_sys_rate->neg_points = -1;
        $new_sys_rate->save();
    }else{
        // onnly assign -1 point
        $sys_rate[0]->neg_points = $sys_rate[0]->neg_points - 1;
        $sys_rate[0]->save();
    }
}
//end negative points
```

Slika 28: Negativna točka ob odjavi potnika s prevoza.

Ko voznik sprejme potnika na prevoz, se preveri, ali ima potnik že obstoječe ocene kot potnik. Če nima, se vozniku dodeli dodatno točko, saj je peljal uporabnika brez ocene (Slika 29).

```php
//if you accept to ride person without rates, you get one more point, you
$conditions_array_1 = ['ride_id' => $ride->id, 'user_id' => $request->user()->id];
$sys_rate_1 = SystemRates::where($conditions_array_1)->get();
//can get max +2 add not rated per ride
$passenger_rates = PassengerRate::where('passenger_id', $request->passenger_id)->get();
//if passenger is not rated yet
if ($passenger_rates->count() == 0) {
    //if driver hasnt sys_rate on this ride yet, create new one
    if ($sys_rate_1->count() == 0) {
        //create sysrate for this user(driver) and give 1 point in add_not_rated
        $new_sys_rate = new SystemRates();
        //this is driver
        $new_sys_rate->user_id = $request->user()->id;
        $new_sys_rate->ride_id = $ride->id;
        $new_sys_rate->add_not_rated = 1;
        $new_sys_rate->save();
    }else{
        if ($sys_rate_1[0]->add_not_rated < 2) {
            $sys_rate_1[0]->add_not_rated = $sys_rate_1[0]->add_not_rated + 1;
            $sys_rate_1[0]->save();
        }
    }
}
```

Slika 29: Pozitivna točka ob sprejemu potnika brez ocen.

Sistemske ocene uporabnika se nato sešteje za vse prevoze, na katerih je bil udeležen kot potnik ali voznik. Statistiko ocen se nato predstavi na profilu uporabnika in je vidna samo prijavljenim uporabnikom.

6.2.3 Vmesni postanki

S funkcionalnostjo vmesnih postankov (Poglavje 5.3.1) zagotavljamo preprost način sporočanja vozniku, kje bi želeli, da nas pobere. V aplikaciji lahko določimo lokacijo in opis postanka. Vmesni postanek se nato prikaže vozniku ob prošnji potnika za prevoz. Voznik lahko nato oceni, ali se mu splača pobrati/odložiti potnika. Glede na to potem sprejme potnika na prevoz (Slika 30).

```php
public function part_ride(Request $request, User $user, Ride $ride)
{
    $this->validate($request, [
        'address_address' => ['required'],
        'address_message' => ['required'],
        'stop_type' => ['required'],
    ]);

    //get ride request of this user on this ride
    $ride_request = RideRequest::where([
                        ['user_id', $user->id],
                        ['ride_id', $ride->id]
                    ])->get()
                    ->first();


    $request->user()->part_ride()->create([
        'ride_id' => $ride->id,
        'ride_request_id' => $ride_request->id,
        'message' => $request->address_message,
        'stop_type' => $request->stop_type,
        'address' => $request->address_address,
        'latitude' => $request->address_latitude,
        'longitude' => $request->address_longitude
    ]);

    return redirect()->route('ride', $ride);
}
```

Slika 30: Vnos novega vmesnega postanka.

6.2.4 Prekrivajoči se prevozi

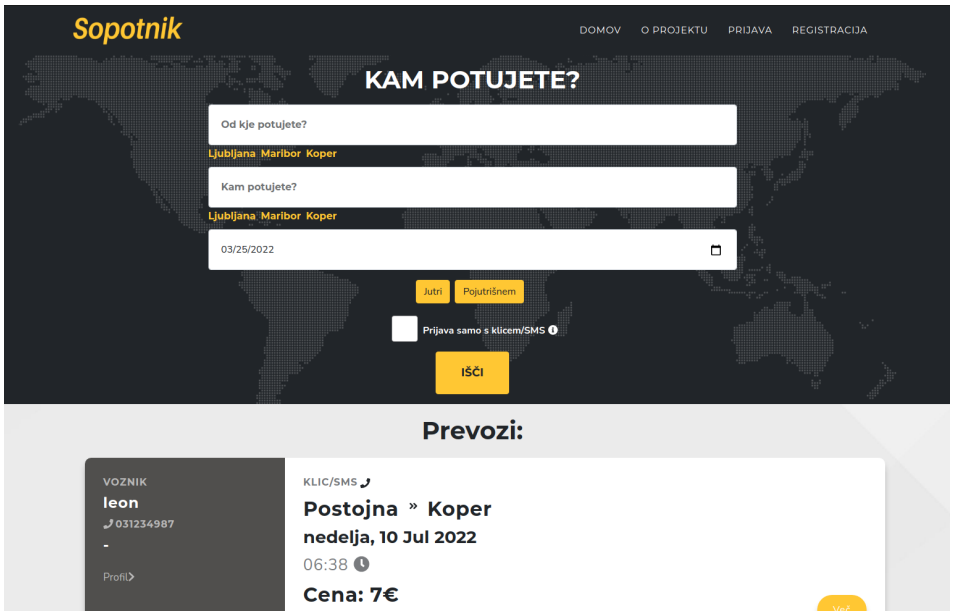
Funkcionalnost nudi večjo transparentnost, saj ne more voznik vnesti dveh prevozov v istem časovnem intervalu. Program ob vnosu novega prevoza, preveri, ali je v tem časovnem intervalu že objavljen kakšen prevoz istega uporabnika. Če je, se vnos prevoza prekine, uporbniku pa je poslano sporočilo o napaki (Slika 31).

```php
//if proposed ride is overlaping with any planned ride by user, throw error
//timestamp of rides start
$ride_time_timestamp = Carbon::parse($request->datetime)->timestamp;
//users(creating ride) rides
$rides = Ride::where('user_id', $user->id)->get();
if ($rides->count() > 0) {
    foreach ($rides as $ride) {
        //if proposed rides time is overlaping any ride time for +-1h, error
        $timestamp = Carbon::parse($ride->datetime)->timestamp;
        // || $timestamp - $ride_time_timestamp < 3600
        if ( abs($ride_time_timestamp - $timestamp) < 3600 ) {
            return redirect()->back()->withErrors(['date_error' => 'V tem času že
            imate objavljen prevoz. Sistem ne omogoča dveh prevozov ob istem času']);
        }
    }
}
```

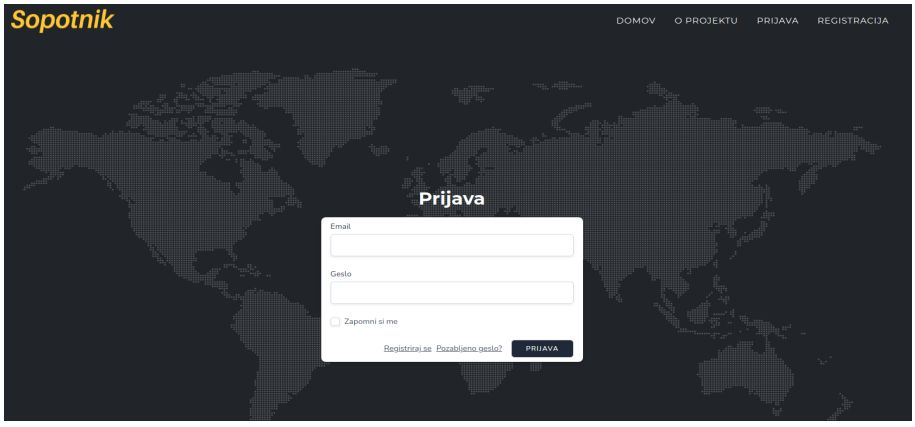
Slika 31: Onemogočanje objave dveh prevozov ob istem času.

6.3 Grafični uporabniški vmesnik

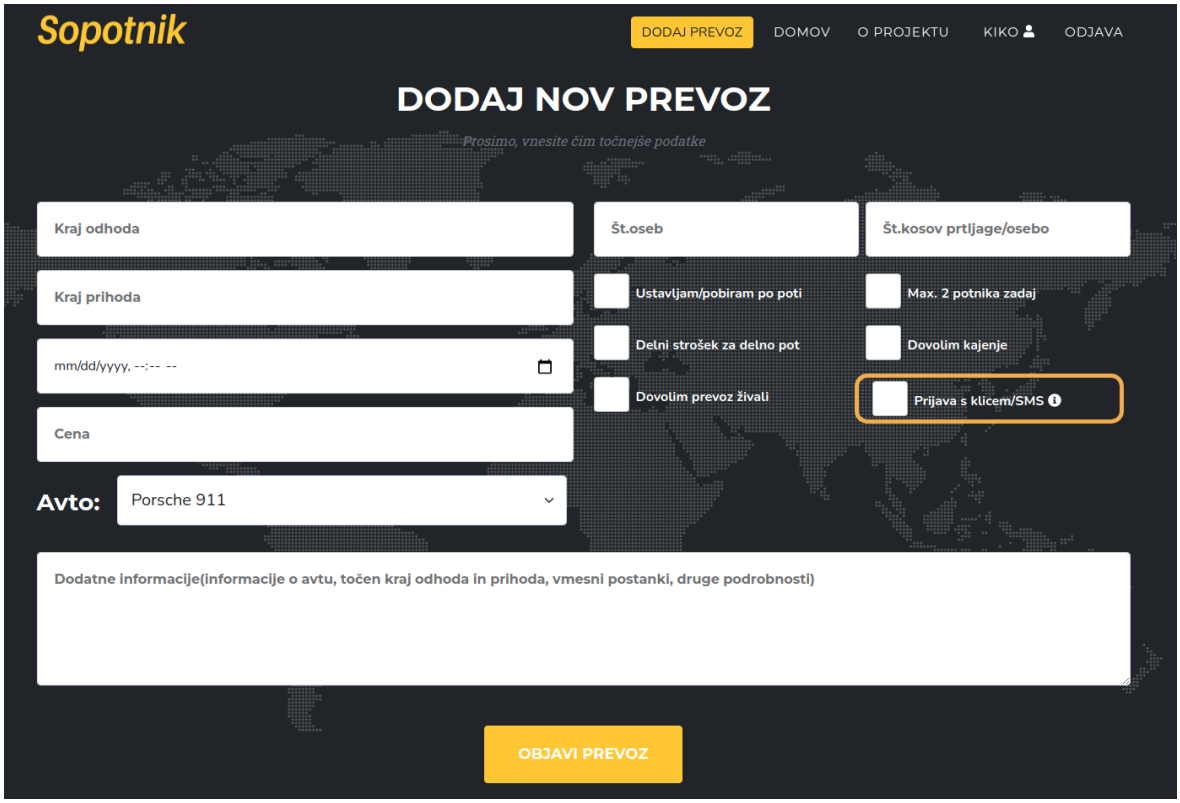
Cilj projekta je tudi izbojšanje grafične podobe. Kot smo že omenili, smo za to uporabili ogrodje Bootstrap ter HTML/CSS/Javascript (Poglavlje 5.2.2). Pri oblikovanju grafičnega vmesnika smo se osredotočili predvsem na jasnost, preglednost in preprostost uporabe. Grafični vmesnik smo prilagodili za vse velikosti zaslonov. Na naslednjih straneh bomo predstavili grafični vmesnik nekaterih glavnih strani aplikacije.



Slika 32: Domača stran aplikacije Sopotnik.

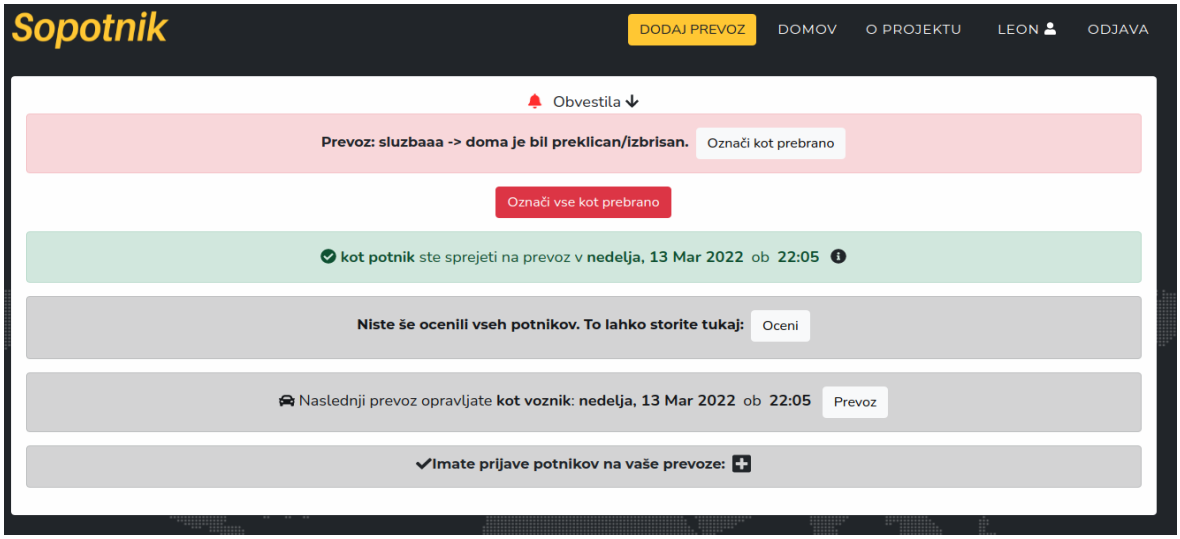


Slika 33: Stran za prijavo v aplikacijo.



Slika 34: Obrazec za vnos novega prevoza.

Slika 35: Izgled strani prevoza z možnostjo sprejemanja potnikov.



Slika 36: Izgled seznama obvestil na domači strani.

7 Testiranje

Postopek testiranja programa je ključen za uspešno objavo programa. V tem postopku namreč preverjamo skladnost rešitve z zastavljenimi specifikacijami ter funkcijskimi in nefunkcijskimi zahtevami. To storimo tako, da si določimo testne primere, ki bi razkrili kakršno koli napako v delovanju programa. Dobri testni primeri so napreč tisti, ki razkrijejo napake, ki jih razvijalec ni predvidel. Tako je test v bistvu uspešen, če najde določeno napako, ki je nismo pričakovali. Na ta način se napake opazi in odpravi v fazi razvoja, kar je neprimerno manj škodljivo, kot če se to zgodi, ko je program že v rabi.

7.1 Opis postopka tesiranja

Uporabili smo vedenjsko (blackbox) testiranje [6]. Pri tej obliki testiranja preizkuševalec ne pozna notranjega delovanja programa in izvaja le naloge, ki so omogočene uporabniku aplikacije. Preizkuševalec običajno opravlja različne testne primere, kjer preizkuša funkcijske zahteve programa, prepustnost, občutljivost na določene vhodne vrednosti itn. [6]

V našem primeru smo testiranje izvedli s pomočjo petih oseb. Testiranje smo izvedli na naslednji način. Ko smo menili, da je aplikacija končana, torej da so izpolnjene vse zahteve, smo določili testne primere, ki so jih morali kasneje preizkuševalci izvesti.

Testiranje smo izvedli v petih fazah. Ko je trenutni preizkuševalec izvedel testiranje in našel napake, smo napake odpravili in dali izboljšano različico aplikacije v testiranje naslednjemu preizkuševalcu. Ta način sicer ni najboljši in zato tudi ne najbolj uveljavljen. Na vsakem koraku testiranja je tako le en preizkuševalec, ki lahko slabo testira sistem in nas tako zavede. Za preizkuševalce smo sicer uporabili tako kolege računalničarje, kot druge kolege, ki sicer niso strokovnjaki, da smo dobili vpogled, kako intuitivna in preprosta je aplikacija za uporabo. Čeprav je vsako različico testirala le ena oseba, smo ugotovili precej napak, kar je bilo zelo dobro za odpravo težav in napredek aplikacije. Izkazala se je tudi ugotovitev iz članka [30], da testiranje z že samo enim preizkuševalcem razkrije do 20 % napak, število odkritih napak po testiranju z več kot petimi preizkuševalci pa ne razkrije več veliko novih napak [30]. Za vsako od najdenih napak smo tudi navedli kritčnost te napake s skalo od 1 do 5, kjer pomeni 1 skoraj neškodljiva napaka, 5 pa kritična napaka. Preizkuševalce smo po koncu testiranja tudi povprašali o splošnem vtisu in kako bi ocenili aplikacijo z oceno od 1 do 5, kjer je 5 najboljša ocena.

Testiranje smo izvedli v lokalnem razvojnem okolju v operacijskem sistemu Linux

(distribucija Ubuntu 20.04 LTS) v programu Google Chrome (različica 99).

7.2 Sistemsko testiranje

Testiranje sistemskih zahtev		height**Sistemska zahteva**
Željeno stanje	**Dejansko stanje**	Varna aplikacija.
Varna apliakcija. Podatki uporabnikov so zaščiteni.	Aplikacija je lahka za uporabo.	Uporaba aplikacije je lahka in intuitivna.
Uporabnost		**Dostopnost**
Aplikcija je odstopna 24/7.	Nemoten dostop do aplikacije.	Odzivno spreminjanje posavitve komponent.
	Prilagodljivost na različne ekrane	
Odzivno spreminjanje postavitve komponent.	Transakcije v podatkovno bazo uspešno izvedene.	Transakcije v podatkovno bazo uspešno izvedene.
Transakcije podatkovne baze		

Tabela 0: Tabela sistemskega testiranja[2].

7.3 Testiranje funkcijskih zahtev

7.3.1 Testni primeri

- Registracija.
- Prijava.
- Vnesi prevoz na relaciji Koper-Ljubljana, 25. 2. 2022 ob 16:30 s spletno prijavo.
- Dodaj svoje prevozno sredstvo.
- Sprejmi potnike na svoj prevoz, kjer so omogočene spletne prijave.
- Oceni potnika/voznika po vožnji.
- Prijavi se na prevoz na relaciji Pivka-Postojna, 24. 2. 2022.
- Dodaj vmesni postanek na prevozu Pivka-Postojna. Dodaj postanek v kraju Prestranek.
- Posodobi svoje podatke na profilu.

- Izbriši uporabniški profil.

7.4 Rezultati testiranja funkcijskih zahtev

Preizkuševalec 1 Napake, ki jih je odkril prvi preizkuševalec:

- Podatki se v primeru napake na ohranijo v tabeli ob napaki. **Kritičnost napake: 2**
- Možnost klika na celotno kartico prevoza, ne zgolj na premajhen gumb. **Kritičnost napake: 2**
- Zavihek z obvestili ni dovolj viden. **Kritičnost napake: 3**

Splošni vtis: 5

Preizkuševalec 2 Napake, ki jih je odkril drugi preizkuševalec, po popravljenih napakah, ki jih je zaznal Preizkuševalec 1:

- Dodatni podatki o prevoznem sredstvu bi morali biti opcijski. **Kritičnost napake: 2**
- Gumb za vnos prevoza je premalo viden. **Kritičnost napake: 3**
- Število kosov pritljage bi se moralo šteti na osebo. **Kritičnost napake: 2**
- Manjkajoče obvestilo o prejeti prošnji na svoj prevoz. **Kritičnost napake: 4**
- Opis vmesnega postanka. Ne zgolj lokacija. **Kritičnost napake: 4**

Splošni vtis: 4

Preizkuševalec 3 Napake, ki jih je odkril tretji preizkuševalec, po popravljenih napakah, ki jih je zaznal Preizkuševalec 2:

- Lokacija za delni postanek se ne osveži med prošnjami, če smo nekoga sprejeli in ga odstranimo. **Kritičnost napake: 4**
- Več gumbov z informacijami. **Kritičnost napake: 1**

Splošni vtis: 5

Preizkuševalec 4 Napake, ki jih je odkril četrti preizkuševalec, po popravljenih napakah, ki jih je zaznal Preizkuševalec 3:

- Konsistentnost uporabe jezika v samodejno generiranih napakah obrazcev. **Kritičnost napake: 1**
- Dodaj stare vrednosti v obrazec ob urejanju prevoza. **Kritičnost napake: 3**
- Omogoči izbiro vrste postanka. Ali te voznik pobere ali odloži. **Kritičnost napake: 1**

Splošni vtis: 5

Preizkuševalec 5 Napake, ki jih je odkril peti preizkuševalec, po popravljenih napakah, ki jih je zaznal Preizkuševalec 4:

- Postavitev nekaterih elementov ni po okusu preizkuševalca. Pri ocenjevanju potnikov bi raje klik na ikono, kot na celoten prevoz. **Kritičnost napake: 1**
- Ko aplikacija vrne iskane prevoze se kraji v iskalniku izbrišejo. **Kritičnost napake: 2**

Splošni vtis: 5

8 Zaključek

Razvoj aplikacije Sopotnik je bil zelo zanimiv, saj smo reševali realni problem, s katerim smo se soočili tudi sami. V tej diplomski nalogi smo opisali vse vidike razvoja, od opisa drugih sistemov, ki smo jih dopolnili in izboljšali.

Za uspešno objavo aplikacije je potrebnega še kar nekaj dela. Zamislili smo si, da aplikacija na začetku deluje le kot spletna aplikacija. Aplikacijo bi zaradi omejenih finančnih sredstev najprej oglaševali le na socialnih omrežjih in upali na čim večji doseg. Doseg bi tudi povečevali s plačljivim oglaševanjem, vendar bi imeli za to na voljo le nekaj sto evrov. Prodor aplikacije na trg bo tako verjetno potekal ogransko, saj bodo nadalnja vlaganja v aplikacijo odvisna od usepeha aplikacije. Če bi se izkazalo, da je aplikacija doživela določen uspeh, torej da jo ljudje uporabljajo, bi se, če bi se pojavila potreba, ustvarilo tudi mobilno aplikacijo. O mobilni aplikaciji smo sicer že razmišljali, saj smo želeli spletno aplikacijo zapakirati kot webview aplikacijo in jo objaviti na spletne trgovine za aplikacije. Po krajši raziskavi smo ugotovili, da se spletne trgovine za aplikacije izogibajo takšnim aplikacijam, saj želijo le prave avtohtone (angleško native) aplikacije. Zaradi tega smo sprejeli odločitev, da gremo v razvoj mobilne aplikacije po določeni začetni fazi, ki bo pokazala uspešnost aplikacije Sopotnik.

Literatura

[1] Arriva vozni redi. https://arriva.si. (Dostopano: 2022-08-03).

[2] BORIS BEIZER. Software system testing and quality assurance. 1984.

[3] Blablacar, o aplikaciji. https://blog.blablacar.com/about-us. (Dostopano: 2022-14-03).

[4] Blablacar ocenjevanje. https://support.blablacar.com/hc/en-gb/articles/360016942980-How-carpool-ratings-work. (Dostopano: 2022-14-03).

[5] Blablacar. https://en.wikipedia.org/wiki/BlaBlaCar. (Dostopano: 2022-14-03).

[6] Vedenjsko testiranje. https://www.guru99.com/black-box-testing.html. (Dostopano: 2022-25-03).

[7] Bootstrap ogrodje. https://getbootstrap.com/. (Dostopano: 2022-20-03).

[8] Composer. https://getcomposer.org/. (Dostopano: 2022-20-03).

[9] Normalizacija podatkovne baze. https://www.w3schools.in/dbms/database-normalization/. (Dostopano: 2022-19-03).

[10] NEJC GAŠPERIN. Spletna aplikacija za vozače: diplomsko delo. *Univerza v Ljubljani, Fakulteta za računalništvo in informatiko*, 2012. ((Dostopano: 2022-11-03)).

[11] Gdpr. https://eur-lex.europa.eu/legal-content/SL/TXT/?uri=CELEX%3A32016R0679. (Dostopano: 2022-18-03).

[12] Analiza in specifikacija zahtev. https://www.geeksforgeeks.org/activities-involved-in-software-requirement-analysis/. (Dostopano: 2022-16-03).

[13] Git. https://git-scm.com/. (Dostopano: 2022-20-03).

[14] Github. https://github.com/. (Dostopano: 2022-20-03).

[15] Google maps api aplikacijski vmesnik. https://developers.google.com/maps. (Dostopano: 2022-21-03).

[16] Google places api aplikacijski vmesnik. https://developers.google.com/maps/documentation/places/web-service/overview. (Dostopano: 2022-21-03).

[17] JESEE GRIFFIN. *Introduction to Laravel*. 2020.

[18] MIKE HILLYER. An introduction to database normalization. 2022.

[19] Jetstream. https://jetstream.laravel.com/2.x/introduction.html. (Dostopano: 2022-20-03).

[20] Joomla. https://www.joomla.org/. (Dostopano: 2022-20-03).

[21] Okolje laravel. https://laravel.com/. (Dostopano: 2022-19-03).

[22] Laraveleloquent orm. https://laravel.com/docs/5.0/eloquent. (Dostopano: 2022-20-03).

[23] Laravel migrations. https://laravel.com/docs/9.x/migrations. (Dostopano: 2022-20-03).

[24] Laravel query builder. https://laravel.com/docs/9.x/queries. (Dostopano: 2022-20-03).

[25] Mvc strukturni vzorec. https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.html. (Dostopano: 2022-20-03).

[26] Mvc structurer. https://developer.mozilla.org/en-US/docs/Glossary/MVC. (Dostopano: 2022-20-03).

[27] Programsko ogrodje symfony. https://symfony.com/. (Dostopano: 2022-20-03).

[28] Mysql podatkovna baze. https://sl.wikipedia.org/wiki/MySQL. (Dostopano: 2022-20-03).

[29] Mysql workbench grafični vmesnik za podatkovno bazo. https://www.mysql.com/products/workbench/. (Dostopano: 2022-21-03).

[30] Jacob Nielsen. Why you only need to test with 5 users. https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/. (Dostopano: 2022-25-03).

[31] Nefunkcijske zahteve programskega produkta. https://www.geeksforgeeks.org/non-functional-requirements-in-software-engineering/. (Dostopano: 2022-18-03).

[32] Nefunkcijske zahteve programskega produkta. https://requirementsquest.com/nonfunctional-requirement-examples/. (Dostopano: 2022-19-03).

[33] Npm(node package manager). https://www.npmjs.com/. (Dostopano: 2022-23-03).

[34] Php. https://en.wikipedia.org/wiki/PHP. (Dostopano: 2022-20-03).

[35] Prevoz.org, o aplikaciji. https://prevoz.org/faq/. (Dostopano: 2022-09-03).

[36] ROK SAMSA. Zapolni moj avto: Razvoj spletne aplikacije za skupno vožnjo neznanih potnikov. *Univerza na primorskem, Fakulteta za matematiko, naravoslovje in informacijske tehnologije*, 2021. (Dostopano: 2022-11-03).

[37] KENG SIAU. An analysis of unified modeling language (uml) graphical constructs based on bww ontology. *Journal of Database Management*, 2010.

[38] Stran prevoz.org. https://slo-tech.com/forum/t334525/299. (Dostopano: 2022-11-03).

[39] Prevoz.org - komentarji in pripombe. https://slo-tech.com/forum/t157879/49. (Dostopano: 2022-11-03).

[40] SŽ vozni redi. https://potniski.sz.si/. ((Dostopano: 2022-08-03).

[41] Uml diagrami. https://www.smartdraw.com/uml-diagram/. (Dostopano: 2022-16-03).

[42] Uml sekvenčni diagram. https://www.lucidchart.com/pages/uml-sequence-diagram. (Dostopano: 2022-16-03).

[43] Visual paradigm program. https://www.visual-paradigm.com/. (Dostopano: 2022-21-03).

[44] Specifikacija zahtev programskega produkta. https://www.visual-paradigm.com/guide/requirements-gathering/requirement-analysis-techniques/. (Dostopano: 2022-16-03).

[45] W3tech. https://w3techs.com/. (Dostopano: 2022-20-03).

[46] Wordpress. https://wordpress.com/. (Dostopano: 2022-20-03).

[47] Xaamp strežnik. https://www.apachefriends.org/index.html. (Dostopano: 2022-21-03).