SVEUČILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI

Dominik Knez, Ante Šarić

Razvoj mobilne aplikacije za informiranje putnika javnog gradskog prijevoza

Zagreb, 2025.

Ovaj rad izrađen je na Fakultetu prometnih znanosti Sveučilišta u Zagrebu na Zavodu za inteligentne transportne sustave na katedri za Primjenjeno računalstvo pod vodstvom prof. dr.sc. Tončija Carića i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2024/2025.

Sadržaj

1 UVOD

U suvremenim gradskim sredinama, učinkovitost i pouzdanost javnog prijevoza sve se više oslanjaju na mogućnost pravovremenog i točnog informiranja putnika. Kvaliteta usluge javnog prijevoza više se ne mjeri isključivo tehničkim karakteristikama vozila ili infrastrukturnim kapacitetima, već i razinom dostupnih informacija koje korisnicima omogućuju planiranje putovanja u skladu s trenutačnim prometnim uvjetima. Precizni podaci o dolascima vozila, promjenama u voznom redu i mogućnostima presjedanja izravno utječu na zadovoljstvo putnika, smanjuju neizvjesnost i povećavaju spremnost građana da odaberu javni prijevoz umjesto osobnog automobila.

Razvoj digitalnih tehnologija omogućio je značajan iskorak u ovom području. Mobilne aplikacije postale su primarni kanal komunikacije između pružatelja usluga i korisnika, nudeći ne samo osnovne informacije, već i napredne funkcionalnosti poput personaliziranih obavijesti, prikaza stanja u prometu u stvarnom vremenu te mogućnosti kupnje i validacije voznih isprava. Time se javni prijevoz transformira iz statičnog sustava, temeljenog na unaprijed definiranom voznom redu, u dinamičan servis koji se prilagođava stvarnim okolnostima i potrebama korisnika.

Unatoč značajnom tehnološkom napretku, u praksi se i dalje javljaju brojna ograničenja. Informacije u stvarnom vremenu često su nedostatne, zastarjele ili neprikladno prikazane, dok korisnička sučelja nerijetko ne uspijevaju na dovoljno intuitivan način objediniti sve relevantne podatke. U takvim okolnostima osobitu važnost poprima razvoj rješenja koje će omogućiti pouzdano prikupljanje, obradu i prezentaciju informacija te njihovu jednostavnu dostupnost krajnjem korisniku. Ovaj rad posvećen je istraživanju i oblikovanju takvog rješenja, s naglaskom na unaprjeđenje kvalitete korisničkog iskustva u području javnog gradskog prijevoza.

1.1 Ciljevi, metode i koraci izrade rada

Glavni cilj ovog rada je izraditi sustav koji će unaprijediti način informiranja putnika u javnom gradskom prijevozu, pružajući precizne i ažurirane podatke kroz pregledno i funkcionalno mobilno sučelje. Sustav mora omogućiti jednostavan pristup informacijama neovisno o lokaciji korisnika, integrirati različite vrste podataka te osigurati visoku razinu

pouzdanosti i dostupnosti. Kako bi se postigli navedeni ciljevi, istraživanje i razvoj provedeni su kroz nekoliko međusobno povezanih faza:

1. Analiza postojećih rješenja i potreba korisnika – provedeno je istraživanje dostupnih aplikacija i servisa namijenjenih informiranju putnika, s ciljem prepoznavanja njihovih prednosti i nedostataka. Posebna pažnja posvećena je korisničkom iskustvu, načinu prezentacije podataka i pouzdanosti informacija.

2. Definiranje funkcionalnih i tehničkih zahtjeva – na temelju provedene analize izrađena je lista funkcionalnosti koje sustav mora sadržavati. U ovom dijelu definirani su i tehnički preduvjeti, uključujući zahtjeve vezane uz arhitekturu sustava, kompatibilnost s mobilnim uređajima te mogućnosti integracije s vanjskim izvorima podataka.

3. Dizajn arhitekture sustava – osmišljen je koncept strukture koji osigurava jasnu organizaciju sustava i međusobnu povezanost njegovih dijelova. Time je postignuta modularnost koja olakšava održavanje i stvara čvrste temelje za nadogradnje i proširenja funkcionalnosti.

4. Implementacija poslužiteljskog dijela – razvijen je mehanizam za prikupljanje i pohranu podataka, uz logiku dohvata iz vanjskih izvora, filtriranja sadržaja te pripreme za prikaz u mobilnoj aplikaciji.

5. Razvoj mobilne aplikacije – izrađeno je korisničko sučelje optimizirano za brz i intuitivan pristup informacijama. Unutar aplikacije implementirane su funkcionalnosti koje su prethodno definirane u sklopu oblikovanja zahtjeva.

6. Testiranje i optimizacija – sustav je testiran u različitim uvjetima rada radi provjere točnosti prikaza, brzine odziva i stabilnosti. Na temelju rezultata testiranja provedena su poboljšanja u svrhu unaprjeđenja performansi i korisničkog iskustva.

7. Analiza rezultata i mogućnosti daljnjeg razvoja – provedena je procjena u kojoj je mjeri sustav ostvario postavljene ciljeve te su razmotreni mogući smjerovi njegova budućeg razvoja, s naglaskom na nadogradnju i proširenje funkcionalnosti.

1.2 Struktura rada

Rad nosi naslov „Razvoj mobilne aplikacije za informiranje putnika javnog gradskog prijevoza”, a njegova se struktura sastoji od šest tematskih cjelina:

1. Uvod
2. Informiranje putnika
3. Arhitektura sustava
4. Opis, obrada i pohrana podataka
5. Funkcionalnosti i korisničko sučelje aplikacije
6. Zaključak

U drugom poglavlju razmatra se uloga informiranja putnika u suvremenom javnom prijevozu. Naglašava se važnost pravovremenog, točnog i jasno strukturiranog prijenosa informacija te se objašnjava kako je to područje usko povezano s inteligentnim transportnim sustavima. Ujedno se prikazuju postojeća rješenja u gradu Zagrebu, pri čemu se analiziraju njihove prednosti i ograničenja.

Treće poglavlje donosi prikaz arhitekture razvijenog sustava. U njemu se razrađuje koncept višeslojne arhitekture koja uključuje korisnički sloj mobilne aplikacije, aplikacijski sloj, sloj podataka te integraciju s vanjskim izvorima podataka. Naglasak je stavljen na način na koji su pojedini slojevi oblikovani i koje funkcije ostvaruju unutar sustava, kao i na način njihove međusobne komunikacije koja omogućuje usklađen rad. Uz to, poglavlje se osvrće i na razvojne tehnologije korištene u implementaciji svakog sloja.

Četvrto poglavlje bavi se obradom i pohranom podataka potrebnih za rad aplikacije. Prikazuju se unaprijed definirani podaci koji čine osnovu sustava javnog prijevoza te podaci koji nastaju u stvarnom vremenu. Opisuje se način njihova prikupljanja, organizacije i privremene pohrane te oblikovanje dodatnih struktura koje omogućuju pouzdano predviđanje dolazaka vozila i prikaz točnih informacija korisnicima.

Peto poglavlje donosi detaljan opis funkcionalnosti i korisničkog sučelja mobilne aplikacije. Prikazani su glavni zasloni i način interakcije korisnika sa sustavom, od pretraživanja stajališta

i prikaza dolazaka vozila u stvarnom vremenu, preko integriranih obavijesti i voznog reda, do osobnih postavki, profila i opcija kupnje karata.

Šesto poglavlje predstavlja zaključak rada. U njemu se sažimaju postignuti rezultati, potvrđuje ostvarenje ciljeva rada te iznose planirani smjerovi daljnjeg razvoja, uključujući objavu aplikacije, daljnja unaprjeđenja sustava te potencijal za uspostavu suradnje s davateljem usluge javnog gradskog prijevoza.

Na završetku rada nalaze se zahvale osobama koje su pridonijele njegovoj izradi, popis korištene literature te popis slika. Uz to su priloženi sažetci na hrvatskom i engleskom jeziku te kratak životopis autora.

2 INFORMIRANJE PUTNIKA

U suvremenim urbanim sredinama uloga informacija postaje jednako važna kao i sama fizička infrastruktura javnog prijevoza. Kvalitetno, točno i pravovremeno informiranje putnika predstavlja temelj funkcionalnog i atraktivnog prometnog sustava. Putnici koji raspolažu jasnim informacijama o kretanju vozila, promjenama u trasama, voznim redovima i mogućnostima presjedanja, donose informirane odluke, osjećaju veću sigurnost i zadovoljstvo, te češće odabiru javni prijevoz kao preferirani oblik mobilnosti, (1). Time se posredno utječe i na smanjenje zagušenja cestovne mreže, potrošnje energije i emisije štetnih plinova, čime informiranje postaje i instrument održive urbane mobilnosti.

Unatoč digitalnoj transformaciji javnog prijevoza, informiranje putnika u mnogim gradovima, uključujući Zagreb, još uvijek nije dostiglo razinu koja bi u potpunosti zadovoljila potrebe i očekivanja korisnika. Česti problemi uključuju nedostatak podataka u stvarnom vremenu, neintuitivna sučelja postojećih aplikacija, nepovezanost različitih izvora informacija, kao i lošu dostupnost za specifične skupine korisnika. U tom kontekstu, razvoj mobilnih aplikacija postaje jedan od ključnih koraka prema unaprjeđenju korisničkog iskustva i modernizaciji prometnog sustava.

2.1 Inteligentni transportni sustavi i funkcionalno područje informiranja putnika

U suvremenim prometnim sustavima, tehnologija igra ključnu ulogu u optimizaciji mobilnosti, povećanju sigurnosti i poboljšanju korisničkog iskustva. U tom kontekstu, inteligentni transportni sustavi (ITS) predstavljaju integraciju informacijskih i komunikacijskih tehnologija u prometnu infrastrukturu, vozila i operativne procese s ciljem stvaranja učinkovitijeg, sigurnijeg i održivijeg prometnog okruženja. ITS se promatra kao nadgradnja tradicionalnih prometnih sustava, koja omogućuje prikupljanje, obradu i distribuciju informacija u stvarnom vremenu, te pruža podršku u donošenju odluka na razini putnika, operatera i upravljačkih tijela, (2).

Prema standardnoj klasifikaciji ITS se sastoji od jedanaest funkcionalnih područja unutar kojih je definirano ukupno 32 temeljne usluge. Ta su područja međusobno povezana i obuhvaćaju sve aspekte prometnog sustava, uključujući upravljanje vozilima i prometom, nadzor okolišnih uvjeta, podršku žurnim službama, elektroničku naplatu te nacionalnu

sigurnost. Jedno od najvažnijih područja, osobito iz perspektive krajnjeg korisnika, jest informiranje putnika.

Funkcionalno područje informiranja putnika obuhvaća pružanje personaliziranih, točnih i pravovremenih informacija korisnicima, kako prije samog putovanja tako i tijekom njegova odvijanja. U kontekstu inteligentnih transportnih sustava, te informacije uključuju očekivana vremena dolaska vozila, promjene u voznim redovima, mogućnosti presjedanja, informacije o zastojima, prometnim incidentima i vremenskim uvjetima. Distribucija tih podataka može se ostvariti kroz različite komunikacijske kanale, uključujući elektroničke zaslone postavljene na stajalištima i unutar vozila, mobilne aplikacije, sustave za navigaciju i internetske platforme.

U okviru temeljnih usluga inteligentnih transportnih sustava, informiranje putnika obuhvaća različite oblike komunikacije i podrške tijekom cijelog putnog procesa. Prije samog putovanja korisnicima se omogućuje planiranje koje uzima u obzir trenutačne prometne uvjete i očekivane okolnosti. Tijekom vožnje dostupne informacije pomažu putnicima i vozačima da donose odluke u skladu s promjenama u prometu, čime se povećava fleksibilnost i učinkovitost kretanja. Osim toga, informacijski sustavi sve češće nude i personalizirane usluge koje se prilagođavaju navikama i potrebama pojedinog korisnika, dok se funkcionalnost dodatno proširuje kroz navigacijske alate koji nude prijedloge optimalnih trasa i upozorenja na moguće smetnje u prometu.

Posebno važan aspekt modernih ITS rješenja jest oslanjanje na mobilne aplikacije kao primarni medij za komunikaciju s korisnicima. S obzirom na sve veću digitalnu pismenost stanovništva i sveprisutnost pametnih telefona, informiranje putem aplikacija omogućuje ne samo veću dostupnost i fleksibilnost, već i smanjenje potrebe za fizičkom infrastrukturom na stajalištima. Osim što omogućuju korisnicima pristup personaliziranim i ažuriranim informacijama, ovakve aplikacije često uključuju dodatne funkcionalnosti poput obavijesti o nepredviđenim događajima, mogućnosti ocjenjivanja usluge i integracije s drugim modalitetima prijevoza, čime se dodatno podiže razina usluge.

Važno je istaknuti i širu društvenu dimenziju informiranja putnika. Kvalitetno informiranje ne samo da povećava zadovoljstvo korisnika, već potiče i veću uporabu javnog prijevoza, čime se pridonosi smanjenju prometnog opterećenja, energetske potrošnje i emisije štetnih plinova. Na taj način, funkcionalno područje informiranja putnika ne doprinosi samo operativnoj

učinkovitosti, već postaje instrument održive urbane mobilnosti i važan čimbenik u razvoju pametnih gradova.

2.2 Informiranje putnika u javnom prijevozu u gradu Zagrebu

Informiranje putnika u Zagrebu trenutno se ostvaruje kroz različite kanale, kako fizičke, tako i digitalne. Među fizičkim rješenjima ističu se elektronički zasloni postavljeni na pojedinim stajalištima tramvajskih i autobusnih linija, vidljivo na slici 1, koji prikazuju predviđeno vrijeme dolaska vozila. Iako korisni, ti zasloni nisu ravnomjerno raspoređeni po mreži stajališta, pa mnogi putnici, osobito u perifernim dijelovima grada, nemaju pristup takvom obliku informiranja. Dodatno, točnost prikazanih podataka često varira, osobito u situacijama poremećaja prometa, što može dovesti do nepouzdanih očekivanja.

Slika 1. Informativni zaslon na stajalištu, *(3)*

U posljednje vrijeme Zagrebački električni tramvaj (ZET), je započeo s implementacijom novih tipova zasloni unutar niskopodnih tramvaja. Na tim zaslonima, kako je vidljivo na slici 2, putnicima se prikazuje popis nekoliko sljedećih stanica uz informaciju o predviđenom vremenu dolaska do svake od njih. Posebno korisna funkcionalnost jest prikaz drugih linija koje prolaze kroz te stanice, zajedno s predviđenim vremenom njihovog dolaska. Takav pristup informiranju osobito je koristan putnicima koji planiraju presjedanje, jer omogućuje donošenje

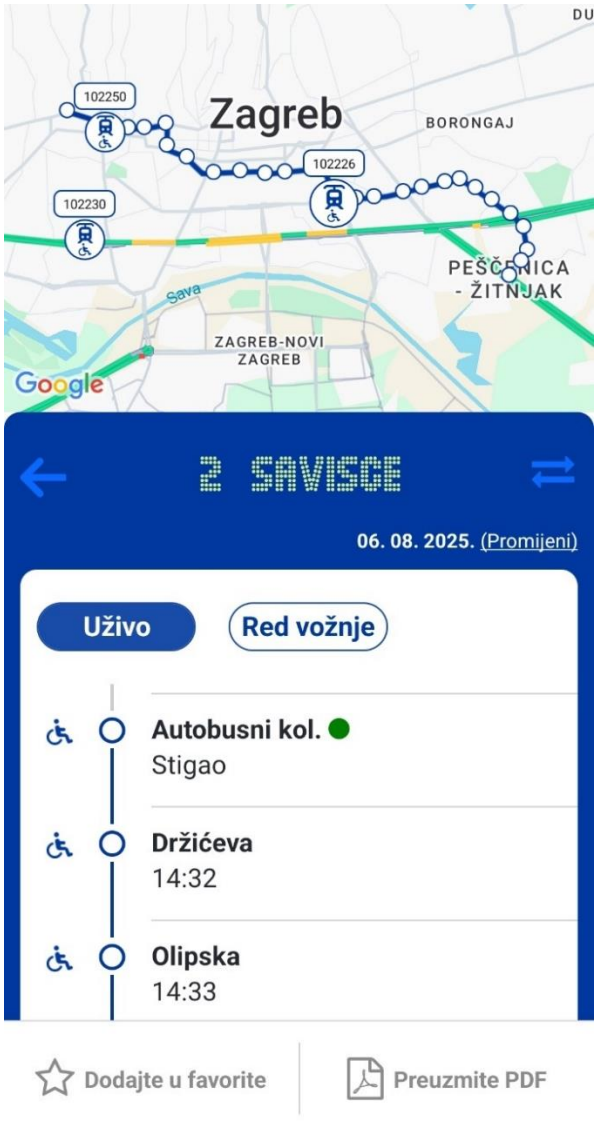
pravodobnih odluka temeljenih na preciznim podacima o dolasku linija u stvarnom vremenu. Osim toga, informacijski zaslon postavljen unutar vozila može poslužiti i kao izvor informacija za putnike koji se nalaze na stajalištu, a kojima vozilo koje upravo dolazi nije od interesa jer iščekuju dolazak druge linije. U slučajevima kada na stajalištu nije postavljen stacionarni zaslon, ili kada je njegova točnost upitna, putnik može iz vozila koje pristaje na stajalište vizualno pristupiti informacijama prikazanim na unutarnjem zaslonu, kao što su podaci o dolasku druge, željene linije. Na taj način, informacijski sustav unutar vozila nadilazi svoju primarnu funkciju i postaje nadogradnja postojećim načinima informiranja na samim stajalištima.

Međutim, važno je naglasiti da takav način informiranja, iako koristan, nije izravno namijenjen ovakvoj upotrebi. Osim što se oslanja na slučajnu dostupnost zaslona i fizičku prisutnost vozila, takav pristup podrazumijeva dodatni napor od strane korisnika koji mora aktivno tražiti informacije iz neoptimalnog izvora. U idealnom slučaju, putnici bi trebali imati jednostavan, pouzdan i centraliziran pristup svim relevantnim informacijama putem osobnog uređaja, bez obzira na to nalaze li se u vozilu ili na stajalištu. Upravo iz tog razloga mobilne aplikacije predstavljaju suvremen i učinkovit alat za unapređenje informiranja putnika.

Slika 2. Informativni zaslon unutar ZET-ovih niskopodnih tramvaja, *(4)*

U kontekstu Zagreba trenutno postoje dvije mobilne aplikacije povezane s javnim gradskim prijevozom, *„Moj ZET“* i *„ZET info“*, koje su razvijene neovisno i s različitim prioritetima, ali ih povezuje zajednički cilj pružanja informacija putnicima.

Aplikacija *Moj ZET*, (5) službeno izdana od strane ZET-a, prvenstveno je usmjerena na digitalnu kupovinu karata i njihovu validaciju putem QR koda. To je jasno vidljivo već iz njezine prezentacije na trgovini aplikacija, gdje su istaknute isključivo funkcionalnosti vezane uz plaćanje i korištenje karata, dok se mogućnosti informiranja putnika uopće ne spominju. Iako aplikacija doista sadrži određene opcije koje se odnose na informiranje, poput prikaza vozila na karti i očekivanih vremena dolaska na pojedina stajališta, te funkcionalnosti djeluju kao sekundarno dodane, bez dublje analize korisničkih potreba i stvarnog konteksta uporabe. Korisniku je, primjerice, omogućeno da odabere željenu liniju i na karti pogleda trenutačne pozicije svih aktivnih vozila na toj ruti, što je prikazano na slici 3. Iako je tehnički riječ o prikazu podataka u stvarnom vremenu, ova funkcionalnost, unatoč povezanosti sa stajalištima, izvedena je na način koji umanjuje njezinu uporabnu vrijednost u svakodnevnim situacijama.



Slika 3. Prikaz vozila na karti u aplikaciji *Moj ZET*

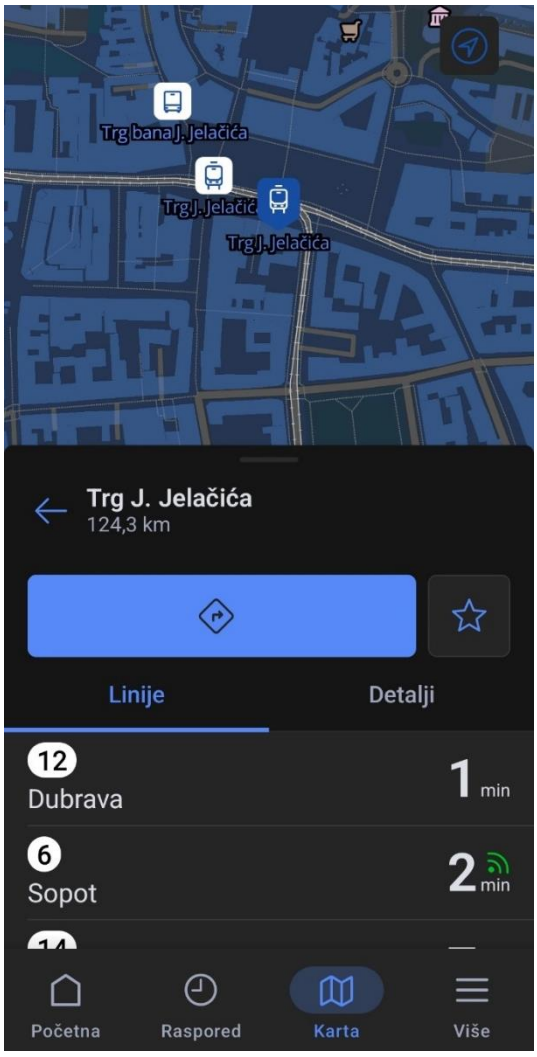
S druge strane, funkcija odabira pojedinog stajališta i prikaza dolazaka vozila, vidljivo na slici 4, u teoriji je korisna, ali se prikazani podaci temelje na unaprijed definiranim voznim redovima, a ne na stvarnoj lokaciji vozila, zbog čega često odstupaju od stvarnog stanja u prometu. Takva ograničenja smanjuju preciznost informacija i time svode funkcionalnost na razinu statičnih zaslona postavljenih na nekim od stajališta.

Slika 4. Prikaz dolazaka vozila u aplikaciji *Moj ZET*

Uz navedeno, aplikacija vizualno i funkcionalno ne ostavlja dojam modernog i intuitivnog sučelja, što dodatno utječe na ukupno korisničko iskustvo. Ovakav dojam potvrđuje i korisnička ocjena od 3.0 na *Google Play* trgovini, što je ispod prosjeka za aplikacije ove vrste i namjene.

S druge strane, aplikacija *ZET info*, (6) razvijena od strane nezavisnog autora Leona Tučeka, predstavlja korak naprijed u odnosu na *Moj ZET*, osobito u pogledu dizajna i korisničkog dojma.

U trenutku izrade ovog rada aplikacija je doživjela opsežan vizualni i funkcionalni redizajn, pri čemu su uvedene brojne animacije, moderniji izgled i bolja organizacija sadržaja. Time je ostvareno osjetno ugodnije i pristupačnije korisničko iskustvo. Glavni naglasak stavljen je na informiranje putnika, uz pregledne prikaze linija i ažurne informacije o prometu. U odnosu na *Moj ZET*, aplikacija *ZET info* pri odabiru stajališta prikazuje vremena dolaska vozila s većom točnošću, budući da se ne oslanja na unaprijed definirane podatke iz voznog reda. Unatoč toj prednosti, aplikacija ne nudi prikaz trenutne lokacije vozila na karti, kako je vidljivo na slici 5, što korisniku otežava procjenu vjerodostojnosti prikazanih vremena i smanjuje razinu povjerenja u dostupne informacije u stvarnom vremenu.



Slika 5. Prikaz dolazaka vozila u aplikaciji *ZET info*

Unatoč pozitivnim pomacima, aplikacija trenutno ima ocjenu 2.3 na *Google Play* trgovini. To je djelomično rezultat nezadovoljstva korisnika naviknutih na staro sučelje, koje su mnogi

odbacili iako nova verzija donosi objektivna poboljšanja u dizajnu i funkcionalnosti. Istovremeno, takva ocjena ukazuje i na činjenicu da funkcionalnosti još uvijek nisu u potpunosti prilagođene stvarnim potrebama korisnika.

Analiza postojećih rješenja pokazuje kako informiranje putnika još uvijek nije dovoljno razvijeno u kontekstu svakodnevnih potreba korisnika javnog prijevoza. Kroz ovu usporedbu vidljivo je da postoji prostor za poboljšanja, osobito u pogledu točnosti, dostupnosti i jasnoće prikazanih informacija. Bitno je naglasiti da su početkom akademske godine, kada je započela izrada ovog rada, obje postojeće aplikacije imale znatno skromniji opseg funkcionalnosti te su tek kasnije doživjele nadogradnje kojima su oblikovane u svoj sadašnji oblik. Razvoj ovdje predstavljene aplikacije odvijao se stoga neovisno o njima, temeljem samostalne analize korisničkih potreba i s ciljem unapređenja sustava informiranja putnika.

3 ARHITEKTURA SUSTAVA

Digitalni informacijski sustavi koji djeluju u stvarnom vremenu, osobito oni u domeni javnog prijevoza, zahtijevaju pažljivo osmišljenu i tehnički dosljednu arhitekturu. Ta arhitektura mora omogućiti kontinuirani protok podataka između više međusobno udaljenih komponenti, osigurati stabilnost i pouzdanost u radu, te biti dovoljno fleksibilna da podnese promjene i proširenja u budućnosti.

U sklopu ovog rada razvijen je sveobuhvatan sustav za informiranje putnika o dolascima vozila javnog gradskog prijevoza u Zagrebu, koji se temelji na višeslojnoj arhitekturi i modernim razvojnim tehnologijama. Sustav uključuje mobilnu aplikaciju kao korisničko sučelje, poslužiteljski servis s aplikacijskom logikom, relacijsku bazu podataka te integraciju s vanjskim izvorima podataka. Njegova arhitektura osmišljena je modularno, s jasnim razdvajanjem uloga svakog sloja, kako bi se osigurala jednostavna održivost, mogućnost nadogradnje i visoka razina performansi u radu.

U nastavku se detaljno prikazuje tehnički pregled arhitekture, uključujući temeljnu podjelu na slojeve, način međusobne komunikacije te konkretne razvojne tehnologije koje su korištene pri implementaciji.

3.1 Pregled sustava i višeslojna arhitektura

Sustav razvijen u okviru ovog rada strukturiran je prema modelu višeslojne arhitekture, koji predstavlja standardni pristup u izgradnji kompleksnijih sustava s jasno razdijeljenim funkcionalnim komponentama, (7). Takva podjela omogućuje da svaki sloj sustava bude odgovoran za specifičan skup zadataka, čime se olakšava razvoj, testiranje i održavanje, ali i omogućuje bolje razumijevanje sustava.

Na najvišoj razini, sustav se može podijeliti u četiri glavne komponente:

1. Korisnički sloj
2. Aplikacijski sloj (engl. *Application Programming Interface*, API)
3. Sloj podataka
4. Vanjski izvori podataka

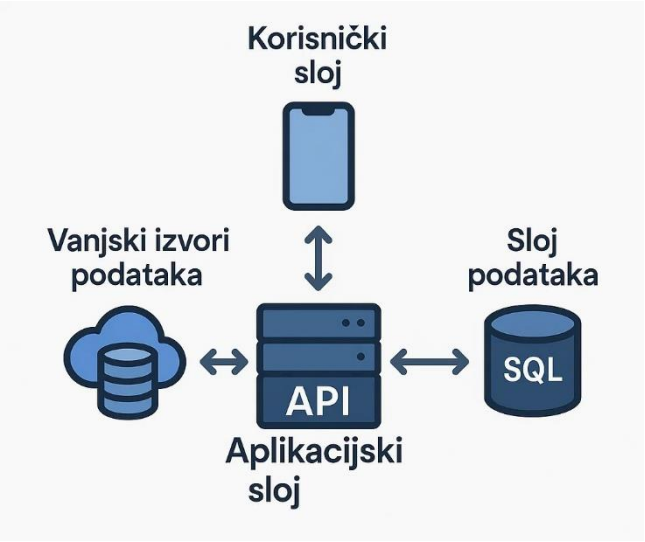
Korisnički sloj čini mobilna aplikacija koja omogućuje krajnjem korisniku interakciju sa sustavom. Ovaj sloj predstavlja vizualnu i funkcionalnu točku pristupa informacijama, poput prikaza stajališta i dolazaka vozila u stvarnom vremenu, pregleda voznih redova te obavijesti o izmjenama u prometu i novostima. Njegova je uloga prvenstveno prezentacijska jer dohvaća podatke iz dubljih slojeva sustava i prikazuje ih u obliku koji je intuitivan, pregledan i prilagođen stvarnim potrebama korisnika u prometnom kontekstu. Aplikacija se ne oslanja na lokalnu pohranu podataka (izuzev nekih korisničkih preferencija, poput opcija favoriziranja), već se u svakom trenutku dinamički povezuje s poslužiteljem kako bi dohvatila najnovije informacije.

Aplikacijski sloj predstavlja most između korisničkog sloja, sloja podataka i vanjskih izvora. Ovdje se odvija cjelokupna poslovna logika sustava. Ovaj sloj obrađuje zahtjeve koji stižu iz aplikacije, dohvaća potrebne podatke, vrši njihovo filtriranje, uparivanje i izračune, te vraća strukturirane rezultate natrag korisničkom sučelju. Osim odgovora na konkretne korisničke upite, aplikacijski sloj periodički komunicira s vanjskim izvorima kako bi prikupljao najnovije informacije o prometu, a zatim ih upisivao i sinkronizirao s bazom podataka. Time postaje operativno središte sustava koje upravlja razmjenom podataka i osigurava njihovu logičku dosljednost.

Sloj podataka čini relacijska baza u kojoj su pohranjene sve informacije relevantne za rad sustava. Struktura ovog sloja obuhvaća tablice koje sadrže podatke o stajalištima, linijama, voznim redovima, trasama i vožnjama, kao i zapise o kretanju vozila u stvarnom vremenu te povijesne podatke potrebne za izradu predikcija. Baza je strukturirana tako da omogućuje brzo izvođenje složenih upita, uz jasno definirane odnose među tablicama, što pridonosi stabilnosti, brzini pristupa i cjelokupnoj pouzdanosti sustava.

Vanjski izvori podataka odnose se na vanjske servise i otvorene podatkovne skupove iz domene javnog gradskog prijevoza. Riječ je o podacima koji obuhvaćaju planirane vozne redove, stajališta, linije i trase preuzete iz statičkih izvora, kao i stvarnovremenske informacije o trenutačnim pozicijama vozila i pripadajućim vožnjama. Sustav ih periodički dohvaća i uparuje s postojećim strukturama unutar baze podataka, čime se omogućuje pravovremeno i precizno informiranje korisnika.

Arhitektura sustava, s jasno definiranim slojevima i njihovim međusobnim odnosima, prikazana je na slici 6.



Slika 6. Arhitektura sustava

3.2 Razvojno okruženje i korištene tehnologije

Izbor razvojnih alata i tehnologija imao je ključnu ulogu u uspješnoj implementaciji ovog sustava. S obzirom na to da je riječ o sustavu koji mora funkcionirati na velikom broju mobilnih uređaja različitih proizvođača, s različitim verzijama operacijskih sustava, bilo je nužno osigurati visoku razinu kompatibilnosti, performansi i stabilnosti. Osim toga, cilj je bio razviti korisnički sloj koji je intuitivan za korištenje i omogućuje brzu interakciju s podatcima u stvarnom vremenu.

Korisnički sloj razvijen je korištenjem React Native tehnologije u okviru Expo platforme, pri čemu je kao razvojno okruženje korišten Visual Studio Code. Odabir React Native okvira nije bio slučajan, već je bio rezultat njegove mogućnosti da se pomoću jedinstvenog izvornog koda razvije rješenje za obje dominantne mobilne platforme, odnosno za Android i za iOS. Na taj se način značajno pojednostavljuje cjelokupan razvojni proces jer nije potrebno izrađivati dva odvojena korisnička sučelja u različitim programskim jezicima, već se sve funkcionalnosti mogu implementirati unutar jedinstvene kodne baze.

Za razliku od starijih pristupa razvoju mobilnih aplikacija, u kojima su se korisnička sučelja prikazivala kao web-stranice unutar preglednickog okvira, React Native omogućuje razvoj rješenja koja se ponašaju kao u potpunosti nativna. U takvim rješenjima korisničko sučelje nije prikazano pomoću web-tehnologija, već se sastoji od stvarnih komponenti operacijskog sustava, uključujući tipke, izbornike, kartice i ostale elemente s kojima korisnik izravno

komunicira. Dodatno, korisnički sloj razvijen u React Native tehnologiji ima izravan pristup hardverskim mogućnostima mobilnog uređaja, uključujući GPS modul, kameru, senzore pokreta i grafički procesor, što u konačnici rezultira znatno boljim performansama, većom glatkoćom animacija i bržim odzivom na korisničke interakcije.

Iskustva stečena tijekom ranijih pokušaja izrade ovog rješenja u alternativnim tehnologijama, poput Xamarina, ukazala su na niz ozbiljnih ograničenja. Sustavi razvijeni tim pristupom pokazivali su nedostatnu podršku za kompleksnije animacije, ograničen pristup komponentama operacijskog sustava te povećanu potrošnju memorije. Zbog tih su razloga prethodna rješenja napuštena u korist React Native tehnologije, koja se pokazala kao robusnija, fleksibilnija i tehnički naprednija opcija. Njenu primjenu dodatno podupire široka zajednica korisnika, stalna ažuriranja i bogata infrastruktura dodatnih programskih biblioteka.

Expo platforma dodatno je unaprijedila razvojni proces jer omogućuje jednostavno testiranje korisničkog sloja na stvarnim uređajima. Korištenjem službene Expo mobilne aplikacije moguće je skeniranjem QR koda učitati i pokrenuti najnoviju verziju razvijenog rješenja izravno na mobilnom uređaju, bez potrebe za povezivanjem s računalom ili korištenjem sporih emulatora. Takav pristup ubrzao je testiranje funkcionalnosti u stvarnim uvjetima rada, što je posebno važno u slučaju sustava koji se oslanja na geolokacijske usluge i obradu podatka u stvarnom vremenu.

Korisnički sloj izrađen je u programskom jeziku TypeScript, koji se temelji na JavaScriptu, ali dodaje niz značajki koje omogućuju sigurnije i stabilnije pisanje programskog koda. Tijekom razvoja, TypeScript se pokazao korisnim jer već u fazi pisanja može upozoriti na moguće greške u logici, što je posebno važno kod većih rješenja koja sadrže mnogo međusobno povezanih dijelova. Svi dijelovi korisničkog sučelja zapisani su u datotekama s nastavkom .tsx, koje omogućuju da se unutar istog dokumenta kombiniraju izgled elemenata i njihovo ponašanje. Na taj je način razvoj korisničkog sučelja bio pregledniji i brži, a istovremeno se zadržala precizna kontrola nad načinom na koji sustav reagira na korisničke radnje.

Aplikacijski sloj razvijen je korištenjem Flask okvira u programskom jeziku Python, pri čemu je kao razvojno okruženje, kao i u slučaju korisničkog sloja, korišten Visual Studio Code. Flask je minimalistički razvojni okvir koji omogućuje izgradnju poslužiteljskih rješenja jednostavnom i čitkom sintaksom. Za razliku od složenijih okvira koji nameću fiksnu strukturu i velik broj konfiguracijskih datoteka, Flask omogućuje potpunu slobodu u organizaciji koda,

što ga čini izrazito pogodnim za projekte u kojima se poslovna logika mora jasno i pregledno definirati bez suvišnih tehničkih slojeva.

Za povezivanje aplikacijskog sloja sa slojem podataka koristi se PyODBC konektor, programska biblioteka koja omogućuje stabilnu i učinkovitu komunikaciju između aplikacijskog sloja izrađenog u Pythonu i baze podataka u sustavu Microsoft SQL Server. Ovaj je pristup odabran zbog svoje pouzdanosti, jednostavne implementacije i dobre podrške za rad s relacijskim modelima podataka.

Kao temelj sustava za trajnu pohranu i strukturiranu analizu podataka, Microsoft SQL Server omogućuje siguran i brz rad s velikim količinama povezanih zapisa. Njegove ključne prednosti uključuju mogućnosti indeksiranja, upravljanja transakcijama, definiranja odnosa između tablica putem stranih ključeva, kao i primjene naprednih sigurnosnih mehanizama za kontrolu pristupa. Upravo ta razina sofisticiranosti u radu s podatcima čini SQL Server prikladnim izborom za sustave koji zahtijevaju česte upite nad velikim skupovima podataka, kao što je slučaj u ovom projektu.

4 OPIS, OBRADA I POHRANA PODATAKA

Funkcioniranje razvijene mobilne aplikacije u potpunosti ovisi o kvaliteti, dostupnosti i ažurnosti podataka o javnom gradskom prijevozu. U okviru ovog rada koriste se službeni podatci koje pruža ZET, i to u obliku GTFS (engl. *General Transit Feed Specification*) specifikacije. Riječ je o standardiziranom formatu zapisa podataka o javnom prijevozu koji omogućuje njihovu lakšu razmjenu, obradu i integraciju u različite digitalne sustave, uključujući mobilne aplikacije i web-servise, (8).

Svi korišteni podatci javno su dostupni i ustupljeni pod licencom "Otvorena dozvola – Open Licence" Republike Hrvatske, koja omogućuje slobodno korištenje, redistribuciju i obradu podataka, uz uvjet odgovarajuće atribucije izvora, (9). Ova vrsta dozvole dio je šireg okvira otvorenih podataka u Republici Hrvatskoj, kojim se promiče transparentnost, potiče inovacija i omogućuje razvoj naprednih digitalnih rješenja temeljenih na javno dostupnim informacijama.

U kontekstu razvoja pametnih gradova i digitalne transformacije javnih usluga, otvoreni podatci igraju ključnu ulogu u unaprjeđenju učinkovitosti, pristupačnosti i kvalitete usluga. Ministarstvo pravosuđa i uprave Republike Hrvatske, u suradnji s drugim tijelima državne uprave, sustavno objavljuje podatke putem Portala otvorenih podataka, što stvara temelj za razvoj brojnih aplikacija i analitičkih sustava u području prometa, zdravstva, obrazovanja i drugih sektora.

4.1 Statički podaci

Statički podatci koji čine temelj GTFS specifikacije predstavljaju skup unaprijed definiranih informacija o mreži linija, voznim redovima i stajalištima u sustavu javnog prijevoza, (10). Pri preuzimanju GTFS paketa dostupno je ukupno devet tekstualnih datoteka u CSV (engl. *Comma Separated Values*) formatu, među kojima se nalaze datoteke *agency.txt, calendar.txt, calendar_dates.txt, feed_info.txt, routes.txt, shapes.txt, stop_times.txt, stops.txt i trips.txt*. Premda se uobičajeno nazivaju statičkim podacima, većina ovih datoteka zapravo se redovito ažurira. Razlog tome su česte promjene u prometnoj organizaciji, poput izmjena trasa, reorganizacija linija ili prilagodbi voznog reda.

Statički GTFS podatci preuzeti su iz službenog izvora koji omogućuje redovito ažurirane i standardizirane informacije o javnom gradskom prijevozu u Zagrebu. Preuzimanjem s mrežne adrese https://www.zet.hr/gtfs-scheduled/latest osigurana je konzistentnost podataka i njihova usklađenost s aktualnim voznim redovima i prometnim strukturama.

Od ukupnog skupa ponuđenih datoteka, za potrebe izgradnje funkcionalnog sustava korištene su samo one koje sadržavaju ključne informacije nužne za pravilno mapiranje linija, identifikaciju stajališta i određivanje dinamike vožnje. Konkretno, analizirane su datoteke trips, routes, shapes, stop_times i stops, dok se ostale nisu koristile jer ne sadržavaju informacije bitne za rad aplikacije. Detaljna obrada svake od navedenih datoteka prikazana je u sljedećim podpoglavljima, pri čemu je poseban naglasak stavljen na način njihove međusobne povezanosti i na njihovu integraciju u relacijsku bazu podataka.

4.1.1 Datoteka *trips.txt*

Datoteka *trips.txt* predstavlja jednu od temeljnih sastavnica GTFS specifikacije jer sadržava podatke o svim predefiniranim putovanjima javnog prijevoza, pri čemu svaki redak odgovara jednom putovanju. Primjer sadržaja ove datoteke prikazan je na slici 7.

```
route_id,service_id,trip_id,trip_headsign,trip_short_name,direction_id,block_id,shape_id
268,"0_20","0_20_26820_268_10003","V. Gorica",,0,26820,"268_18"
268,"0_20","0_20_26820_268_10004","Gl.kolodvor",,1,26820,"268_9"
268,"0_20","0_20_26820_268_10009","V. Gorica",,0,26820,"268_18"
268,"0_20","0_20_26820_268_10010","Gl.kolodvor",,1,26820,"268_9"
268,"0_20","0_20_26820_268_10015","V. Gorica",,0,26820,"268_18"
268,"0_20","0_20_26820_268_10016","Gl.kolodvor",,1,26820,"268_9"
212,"0_20","0_20_21220_212_10001","Sesvete",,0,21220,"212_5"
268,"0_20","0_20_26820_268_10021","V. Gorica",,0,26820,"268_18"
311,"0_20","0_20_31102_311_10005","Cerov. vrh",,0,31102,"311_14"
14,"0_20","0_20_1401_14_10002","Zapruđe",,1,1401,"14_25"
271,"0_20","0_20_27103_271_10009","Glavnica D.",,0,27103,"271_9"
7,"0_20","0_20_701_7_12966","Savski most",,1,701,"7_68"
```

Slika 7. Isječak sadržaja datoteke *trips.txt*

Najvažniji atribut unutar ove tablice je *trip_id*, koji jednoznačno identificira svaku pojedinačnu vožnju. U trenutku kada vozilo započinje novu turu, bez obzira radi li se o početku prometovanja tijekom dana ili o nastavku vožnje nakon promjene smjera na terminalu, ono se referencira putem jednog od unaprijed definiranih *trip_id* identifikatora. Ove vrijednosti planira i održava ZET, a njihov broj varira ovisno o broju predviđenih vožnji u određenom

vremenskom razdoblju. Budući da se *trip_id* koristi kao ključ za povezivanje s brojnim drugim GTFS datotekama, njegovo razumijevanje presudno je za ostvarenje funkcionalne integracije podataka unutar sustava.

Atribut *route_id* povezuje svaki pojedini *trip* s pripadajućom linijom javnog prijevoza. U praksi, tramvajske linije obično su označene jednoznamenkastim ili dvoznamenkastim brojevima, dok autobusne linije nose troznamenkaste oznake. *Trip_headsign* označava krajnje odredište vožnje, odnosno naziv stajališta prema kojem vozilo prometuje, čime korisniku omogućuje razumijevanje smjera kretanja unutar iste linije.

Posebnu ulogu ima atribut *shape_id*, koji povezuje pojedino putovanje s njegovom geografskom putanjom, što će biti dodatno razjašnjeno u daljnjoj razradi GTFS datoteka. Preostali atributi, poput *direction_id*, *service_id*, *trip_short_name* i *block_id*, u kontekstu ove aplikacije nisu bili od značaja te nisu uključeni u daljnju analizu.

Struktura SQL tablice koja odgovara sadržaju datoteke *trips.txt* prikazana je na slici 8 i koristi se za pohranu pripadajućih podataka.

```
CREATE TABLE trips (
    route_id int NOT NULL,
    service_id nvarchar(50) NOT NULL,
    trip_id nvarchar(100) NOT NULL,
    trip_headsign nvarchar(100) NOT NULL,
    trip_short_name nvarchar(50),
    direction_id int NOT NULL,
    block_id int NOT NULL,
    shape_id nvarchar(50) NOT NULL
);
```

Slika 8. Izrada tablice *trips* u SQL-u

4.1.2 Datoteka *routes.txt*

Datoteka *routes.txt* sadržava informacije o svim definiranim linijama javnog prijevoza, a primjer njezina sadržaja prikazan je na slici 9. Svaka linija, odnosno ruta, identificirana je atributom *route_id*, koji se može izravno povezati s istoimenim atributom u datoteci *trips.txt*, čime se uspostavlja veza između pojedinog putovanja i njegove pripadajuće linije.

```
route_id,agency_id,route_short_name,route_long_name,route_desc,route_type,route_url,route_color,route_text_color
2,0,"2","Črnomerec - Savišće",,0,,"ffffff","000000"
3,0,"3","Ljubljanica -Savišće",,0,,"ffffff","000000"
4,0,"4","Savski most - Dubec",,0,,"ffffff","000000"
5,0,"5","Prečko-Park Maksimir",,0,,"ffffff","000000"
6,0,"6","Črnomerec - Sopot",,0,,"ffffff","000000"
7,0,"7","Savski most-Dubrava",,0,,"ffffff","000000"
8,0,"8","Mihaljevac - Zapruđe",,0,,"ffffff","000000"
9,0,"9","Ljubljanica-Borongaj",,0,,"ffffff","000000"
11,0,"11","Črnomerec - Dubec",,0,,"ffffff","000000"
12,0,"12","Ljubljanica -Dubrava",,0,,"ffffff","000000"
13,0,"13","Žitnjak - Kvater.trg",,0,,"ffffff","000000"
14,0,"14","Mihaljevac-Zapruđe",,0,,"ffffff","000000"
17,0,"17","Prečko - Borongaj",,0,,"ffffff","000000"
```

Slika 9. Isječak sadržaja datoteke *routes.txt*

Atribut *route_short_name* predstavlja broj linije vidljiv krajnjem korisniku, primjerice „2”, „6” ili „11”, dok *route_long_name* sadržava prošireni opis linije, u formatu “početno stajalište – završno stajalište”. Ostali atributi, poput *agency_id*, *route_type*, *route_url*, *route_color* i *route_text_color*, nalaze se u strukturi datoteke, ali u okviru ovog rada nisu korišteni jer nisu bili potrebni za realizaciju funkcionalnosti aplikacije.

Struktura SQL tablice u koju se pohranjuju podatci iz datoteke *routes.txt* prikazana je na slici 10.

```sql
CREATE TABLE routes (
    route_id int NOT NULL,
    agency_id int,
    route_short_name nvarchar(50),
    route_long_name nvarchar(255),
    route_desc nvarchar(255),
    route_type int,
    route_url nvarchar(255),
    route_color nvarchar(10),
    route_text_color nvarchar(10)
);
```

Slika 10. Izrada tablice *routes* u SQL-u

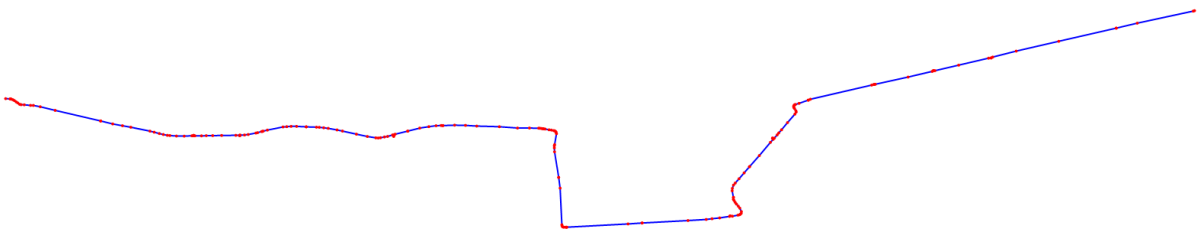
4.1.3 Datoteka *shapes.txt*

Sadržaj datoteke *shapes.txt*, prikazan na slici 11, opisuje geografsku putanju kojom se vozilo kreće unutar mreže javnog gradskog prijevoza.

```
shape_id,shape_pt_lat,shape_pt_lon,shape_pt_sequence,shape_dist_traveled
"2_8",45.789396,16.034915,10001,
"2_8",45.789938,16.035449,10002,
"2_8",45.790728,16.036258,10003,
"2_8",45.790727,16.036405,10004,
"2_8",45.790727,16.036405,20001,
"2_8",45.790971,16.036516,20002,
"2_8",45.792495,16.038063,20003,
"2_8",45.792573,16.038153,20004,
"2_8",45.792573,16.038153,30001,
```

Slika 11. Isječak sadržaja datoteke *shapes.txt*

Svaki *shape_id*, koji je prethodno naveden u datoteci *trips.txt*, ovdje se proširuje na niz konkretnih točaka koje definiraju trasu putovanja. Za svaki identifikator rute unutar ove datoteke zapisano je više redaka, a svaki redak sadržava par geografskih koordinata (*shape_pt_lat*, *shape_pt_lon*) te pripadajući redoslijedni broj (*shape_pt_sequence*). Kada se svi zapisi za određeni *shape_id* poslože prema redoslijedu, dobiva se detaljna trajektorija kojom se vozilo kreće tijekom putovanja, kao što je prikazano na slici 12.



Slika 12. Primjer konstruirane putanje vozila temeljem *shape_id* 6_12

Svi podatci iz ove datoteke pohranjuju se u odgovarajuću SQL tablicu, čija je struktura prikazana na slici 13.

```sql
CREATE TABLE shapes (
    shape_id nvarchar(50) NOT NULL,
    shape_pt_lat float NOT NULL,
    shape_pt_lon float NOT NULL,
    shape_pt_sequence int NOT NULL,
    shape_dist_traveled float
);
```

Slika 13. Izrada tablice *shapes* u SQL-u

4.1.4 Datoteka *stop_times.txt*

Datoteka *stop_times.txt*, čiji je primjer prikazan na slici 14, ključna je za definiranje voznog reda pojedinih putovanja unutar sustava javnog prijevoza.

```
trip_id,arrival_time,departure_time,stop_id,stop_sequence,stop_headsign,pickup_type,drop_off_type,shape_dist_traveled
"0_20_26820_268_10003","00:30:00","00:30:00","110_82",1,"V. Gorica",,,
"0_20_26820_268_10003","00:32:30","00:32:30","238_24",2,"V. Gorica",,,
"0_20_26820_268_10003","00:33:40","00:33:40","450_24",3,"V. Gorica",,,
"0_20_26820_268_10003","00:35:20","00:35:20","1085_24",4,"V. Gorica",,,
"0_20_26820_268_10003","00:36:20","00:36:20","277_24",5,"V. Gorica",,,
"0_20_26820_268_10003","00:37:20","00:37:20","1233_21",6,"V. Gorica",,,
"0_20_26820_268_10003","00:38:30","00:38:30","565_24",7,"V. Gorica",,,
"0_20_26820_268_10003","00:39:40","00:39:40","566_24",8,"V. Gorica",,,
"0_20_26820_268_10003","00:40:50","00:40:50","1087_24",9,"V. Gorica",,,
"0_20_26820_268_10003","00:41:30","00:41:30","1188_24",10,"V. Gorica",,,
"0_20_26820_268_10003","00:44:00","00:44:00","1189_24",11,"V. Gorica",,,
"0_20_26820_268_10003","00:45:00","00:45:00","1190_21",12,"V. Gorica",,,
"0_20_26820_268_10003","00:46:00","00:46:00","1191_21",13,"V. Gorica",,,
"0_20_26820_268_10003","00:46:40","00:46:40","1192_21",14,"V. Gorica",,,
"0_20_26820_268_10003","00:47:20","00:47:20","1843_21",15,"V. Gorica",,,
```

Slika 14. Isječak sadržaja datoteke *stop_times.txt*

Za svaki *trip_id*, ova datoteka precizno određuje plan kretanja vozila, uključujući popis stajališta (*stop_id*) kroz koja vozilo prolazi, njihov redoslijed (*stop_sequence*) te predviđena vremena dolaska (*arrival_time*) i odlaska (*departure_time*) sa svake pojedine stanice. Drugim riječima, za svako planirano putovanje unutar sustava, ovdje je zadan njegov potpuni vremenski i prostorni raspored.

Atribut *stop_id* koristi se za povezivanje s fizičkom lokacijom stajališta, čiji su detalji opisani u sljedećem podpoglavlju o datoteci *stops.txt*. Osim navedenih vremenskih i sekvencijalnih informacija, prisutan je i atribut *stop_headsign*, koji dodatno označava smjer kretanja vozila prema krajnjem odredištu.

Struktura SQL tablice u koju se pohranjuju podaci iz ove datoteke prikazana je na slici 15.

```sql
CREATE TABLE stop_times (
    trip_id nvarchar(50),
    arrival_time nvarchar(8),
    departure_time nvarchar(8),
    stop_id nvarchar(50),
    stop_sequence int,
    stop_headsign nvarchar(100),
    pickup_type int,
    drop_off_type int,
    shape_dist_traveled float
);
```

Slika 15. Izrada tablice *stop_times* u SQL-u

4.1.5 Datoteka *stops.txt*

Datoteka *stops.txt* sadržava sve relevantne podatke o stajalištima unutar sustava javnog gradskog prijevoza. Primjer njezina sadržaja prikazan je na slici 16.

```
stop_id,stop_code,stop_name,stop_desc,stop_lat,stop_lon,zone_id,stop_url,location_type,parent_station
"98_1","1","Črnomerec",,45.815002,15.934932,,,0,98
"98_2","2","Črnomerec",,45.815398,15.933924,,,0,98
"98_11","11","Črnomerec",,45.815127,15.934536,,,0,98
"98",,"Črnomerec",,45.815175,15.934464,,,1,
"99_23","23","Črnomerec",,45.815318,15.933231,,,0,99
"99_26","26","Črnomerec",,45.814933,15.933877,,,0,99
"99_31","31","Črnomerec",,45.816087,15.933219,,,0,99
"99_32","32","Črnomerec",,45.816059,15.933327,,,0,99
"99_36","36","Črnomerec",,45.814943,15.934140,,,0,99
"99_42","42","Črnomerec",,45.815739,15.933910,,,0,99
"99_52","52","Črnomerec",,45.815800,15.934131,,,0,99
```

Slika 16. Isječak sadržaja datoteke *stops.txt*

Svako stajalište definirano je atributom *stop_id*, dok *stop_name* označava stvarni naziv stanice, primjerice „Črnomerec”. Osim naziva, ključni su atributi *stop_lat* i *stop_lon*, koji precizno određuju geografsku lokaciju svakog stajališta. Atribut *parent_station* omogućuje grupiranje više fizičkih lokacija u jedinstvenu logičku cjelinu, što je uobičajeno u situacijama kada se stajališta nalaze na suprotnim stranama ceste ili su razdvojena prema smjerovima kretanja. Na taj se način uspostavlja hijerarhijska struktura između glavnih stanica i njihovih pripadajućih podstanica, čime se olakšava organizacija prikaza i obrada podataka unutar aplikacije.

Struktura SQL tablice u koju se pohranjuju podatci iz datoteke *stops.txt* prikazana je na slici 17.

```
CREATE TABLE stops (
    stop_id varchar(255) NOT NULL,
    stop_code nvarchar(255),
    stop_name nvarchar(255) NOT NULL,
    stop_desc nvarchar(255),
    stop_lat float,
    stop_lon float,
    zone_id varchar(255),
    stop_url nvarchar(255),
    location_type int,
    parent_station nvarchar(255)
);
```

Slika 17. Izrada tablice *stops* u SQL-u

4.1.6 Relacije između tablica u GTFS strukturi

Iako su GTFS datoteke formalno odvojene i svaka sadržava podatke o različitim elementima sustava javnog prijevoza, njihova je međusobna povezanost izrazito snažna. Ova se povezanost ostvaruje kroz zajedničke identifikatore, odnosno atribute koji se pojavljuju u više tablica i služe kao ključ za međusobno povezivanje podataka. Ključni među njima su *trip_id*, *route_id*, *shape_id* i *stop_id*.

U praksi, ova se povezanost ostvaruje kroz SQL upite koji koriste operacije spajanja tablica (*JOIN*) radi integracije više vrsta podataka u jednu logičku cjelinu. Primjerice, povezivanje može započeti od jednog zapisa iz tablice *trips*, identificiranog putem atributa *trip_id*. Na temelju tog zapisa moguće je dohvatiti pripadajući *route_id*, koji vodi prema tablici *routes*, gdje se nalaze osnovne informacije o liniji, poput njezinog broja i smjera kretanja.

Zatim se, pomoću atributa *shape_id*, isti zapis iz *trips* tablice može povezati s tablicom *shapes*, čime se dobiva konkretna geografska putanja koju vozilo prati tijekom putovanja. Nadalje, atribut *trip_id* omogućuje dohvat vremenskog rasporeda iz tablice *stop_times*, koji za svako stajalište definira planirano vrijeme dolaska i odlaska. Putem atributa *stop_id* iz te se tablice ostvaruje veza s tablicom *stops*, koja sadržava detaljne informacije o svakom stajalištu, uključujući naziv i geografsku lokaciju.

Zahvaljujući toj višestrukoj povezanosti, svako je putovanje moguće u potpunosti rekonstruirati kako u prostornom, tako i u vremenskom smislu. Sustav tako može prikazati liniju kojom vozilo prometuje, njezinu točnu trasu na karti, popis stajališta uz pripadajuće vremenske podatke, kao i sve druge informacije relevantne za korisnički prikaz u aplikaciji.

4.1.7 Učitavanje i pohrana statičkih podataka

Kako bi se podaci iz GTFS datoteka uspješno integrirali u bazu podataka sustava, razvijena je Python skripta koja automatizira proces učitavanja i pohrane svih statičkih podataka. Skripta koristi biblioteku *pyodbc* za uspostavu veze s bazom podataka Microsoft SQL Server, dok se za obradu datoteka koristi standardna *csv* biblioteka.

Prije svakog novog učitavanja podataka, skripta najprije briše sve prethodne zapise iz postojećih SQL tablica. Time se osigurava da baza sadrži isključivo ažurne i konzistentne

podatke, bez dupliciranja ili zadržavanja zastarjelih zapisa. Ovaj pristup posebno je važan s obzirom na činjenicu da se GTFS datoteke povremeno ažuriraju, osobito u slučaju promjena u voznom redu, trasama ili organizaciji linija.

Svaka datoteka iz GTFS skupa učitava se zasebno, i to sljedećim redoslijedom: *routes.txt*, *shapes.txt*, *stops.txt*, *trips.txt* i *stop_times.txt*. Za svaku od njih definirana je zasebna funkcija unutar skripte koja otvara pripadajuću datoteku, preskače zaglavlje te iterativno čita svaki redak. Pročitani podaci potom se formatiraju i umeću u pripadajuću SQL tablicu koristeći SQL *INSERT* naredbe s parametrima.

Tijekom obrade, skripta vodi računa o mogućim praznim poljima u datotekama. Takva se polja eksplicitno pretvaraju u *NULL* vrijednosti u skladu s definiranim tipovima podataka u bazi. Također, podaci se prema potrebi pretvaraju u odgovarajuće numeričke ili tekstualne tipove, čime se osigurava usklađenost s očekivanom strukturom baze podataka.

Nakon što se svi zapisi umetnu, poziva se metoda za potvrdu transakcije (*commit*) te se veza prema bazi zatvara. Time se završava proces učitavanja i pohrane podataka, koji se može pokrenuti ručno ili uključiti u periodički raspored izvođenja kako bi sustav uvijek raspolagao ažurnim statičkim podacima o prometnoj mreži.

4.2 Dinamički podaci

Uz statičke podatke koji čine temelj strukture prometne mreže, za ostvarenje potpune funkcionalnosti razvijene aplikacije bilo je nužno obraditi i integrirati i podatke u stvarnom vremenu. Riječ je o dinamičkim podatcima koje ZET pruža putem protokola GTFS-RT (engl. *General Transit Feed Specification – Realtime*), (11) a koji omogućuju kontinuiranu aktualizaciju podataka o trenutačnom prometnom stanju na temelju kretanja vozila u stvarnom vremenu. Ovi se podatci učestalo ažuriraju te predstavljaju ključnu komponentu za prikaz trenutačne lokacije vozila, praćenje njegove putanje i procjenu dolaska na pojedina stajališta.

4.2.1 Dekodiranje GTFS-RT zapisa

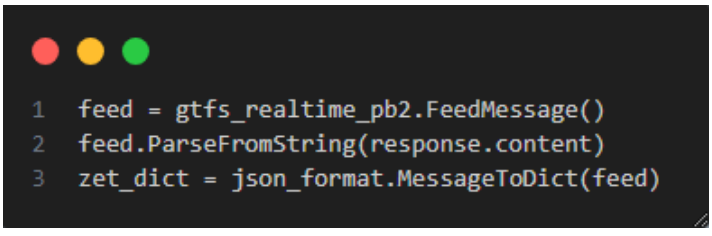
Dinamički podatci u stvarnom vremenu preuzimaju se iz službenog izvora koji omogućuje kontinuirano ažurirane informacije o prometnom stanju u zagrebačkom javnom gradskom

prijevozu. Preuzimanjem s mrežne adrese https://zet.hr/gtfs-rt-protobuf osigurava se pravovremenost i aktualnost podataka, koji se isporučuju u binarnom formatu temeljenom na tehnologiji *Protocol Buffers* (*protobuf*), (12) razvijenoj u sklopu tehnološke infrastrukture tvrtke *Google*. Riječ je o visoko optimiziranom obliku serijalizacije podataka, koji omogućuje učinkovit prijenos strukturiranih informacija uz minimalnu veličinu zapisa i visoku brzinu dekodiranja. Ipak, takav format nije neposredno čitljiv niti prikladan za izravnu analizu, što predstavlja izazov pri njegovoj uporabi u konvencionalnim programskim okruženjima.

S obzirom na to da dekodirani sadržaj mora postati pristupačan standardnim algoritamskim operacijama, prvi korak u obradi dinamičkih podataka uključuje njihovo dekodiranje. U tu se svrhu koristi specijalizirana biblioteka *gtfs-realtime-bindings*, koja je usklađena s pripadajućom *.proto* datotekom. Ta datoteka definira strukturu i semantiku svih mogućih entiteta unutar GTFS-RT poruke. Njezina je funkcija ključna jer omogućuje da se binarni sadržaj interpretira na način koji je usklađen sa standardom GTFS-RT.

Dekodiranje se u konkretnom rješenju provodi učitavanjem binarnog sadržaja u objekt klase *FeedMessage*, koji je unaprijed generiran na temelju *.proto* definicije. Nakon toga se poziva metoda *ParseFromString*, kojom se sadržaj u binarnom obliku dekomprimira i prevodi u konkretne podatke.

Kako bi se osiguralo daljnje jednostavno upravljanje dobivenim podatcima unutar okruženja programskog jezika *Python*, oni se dodatno prevode u strukturirani rječnički oblik (engl. *Dictionary*) nalik JSON formatu, što je prikazano na slici 18, korištenjem funkcije *MessageToDict*. Rezultat ovog procesa je strukturirani objekt pod nazivom *zet_dict*, koji predstavlja čitljiv i logički organiziran prikaz podataka o kretanju vozila u stvarnom vremenu. Takav format omogućuje programski pristup svakom podatkovnom elementu pojedinačno, čime se osigurava fleksibilnost u analizi i daljnjoj obradi podataka.



Slika 18. Pretvorba GTFS-RT binarnog sadržaja u čitljiv JSON rječnik

4.2.2 Struktura podataka i razvoj tablice *TripUpdates*

Nakon što su GTFS-RT podatci uspješno dekodirani i prevedeni u strukturirani rječnički oblik, dobiven je programski čitljiv prikaz trenutačnih stanja svih aktivnih vozila u prometu, vidljivo na slici 19. Takav oblik podataka omogućuje detaljan uvid u kretanje vozila, ali i preciznu segmentaciju pojedinih informacija relevantnih za daljnju analizu i prikaz u mobilnoj aplikaciji.


```
artDate': '20250723', 'scheduleRelationship': 'SCHEDULED', 'routeId': '134'}, 'stopTimeUpdate': [{'stopSequence': 5, 'arr
EDULED', 'routeId': '14'}, 'stopTimeUpdate': [{'stopSequence': 15, 'arrival': {'delay': -200, 'time': '1753268686'}, 'dep
{'id': 'XDZKTZS07Q', 'tripUpdate': {'trip': {'tripId': '0_12_23801_238_10055', 'startDate': '20250723', 'scheduleRelation
p': 'SCHEDULED', 'routeId': '5'}, 'stopTimeUpdate': [{'stopSequence': 1, 'departure': {'delay': 0}, 'stopId': '197_2', 's
', 'vehicle': {'id': '458'}}}, {'id': 'XDZKTZS06S', 'tripUpdate': {'trip': {'tripId': '0_12_23701_237_10027', 'startDate'
07A', 'tripUpdate': {'trip': {'tripId': '0_12_701_7_11530', 'startDate': '20250723', 'scheduleRelationship': 'SCHEDULED',
rture': {'delay': 206, 'time': '1753268896'}, 'stopId': '1892_23', 'scheduleRelationship': 'SCHEDULED'}], 'timestamp': '1
': '117_3', 'scheduleRelationship': 'SCHEDULED'}], 'timestamp': '1753268835'}}, {'id': 'X3ZKTZS05C_102291', 'vehicle': {'
'X7ZKTZS0DK', 'tripUpdate': {'trip': {'tripId': '0_12_12302_123_10113', 'startDate': '20250723', 'scheduleRelationship':
'scheduleRelationship': 'SCHEDULED', 'routeId': '147'}, 'stopTimeUpdate': [{'stopSequence': 1, 'departure': {'delay': 0},
nce': 13, 'arrival': {'delay': -116, 'time': '1753268820'}, 'departure': {'delay': -116, 'time': '1753268820'}, 'stopId':
24', 'scheduleRelationship': 'SCHEDULED'}], 'timestamp': '1753268837'}}, {'id': 'XEZKTZS05F_393', 'vehicle': {'trip': {'t
', 'tripUpdate': {'trip': {'tripId': '0_12_11602_116_10049', 'startDate': '20250723', 'scheduleRelationship': 'SCHEDULED'
002_310_10095', 'startDate': '20250723', 'scheduleRelationship': 'SCHEDULED', 'routeId': '310'}, 'stopTimeUpdate': [{'sto
imestamp': '1753268806'}}, {'id': 'XCZKTZS07N', 'tripUpdate': {'trip': {'tripId': '0_12_22601_226_10045', 'startDate': '2
'tripUpdate': {'trip': {'tripId': '0_12_17205_172_10320', 'startDate': '20250723', 'scheduleRelationship': 'SCHEDULED', '
28_10433', 'startDate': '20250723', 'scheduleRelationship': 'SCHEDULED', 'routeId': '228'}, 'position': {'latitude': 45.8
hip': 'SCHEDULED', 'routeId': '3'}, 'position': {'latitude': 45.80386, 'longitude': 16.019716}, 'timestamp': '1753268834'
tude': 45.812607, 'longitude': 15.895808}, 'timestamp': '1753268838', 'vehicle': {'id': '48'}}}, {'id': 'X9ZKTZS0JH', 'tr
'SCHEDULED'}], 'timestamp': '1753268822'}}, {'id': 'X3ZKTZS0DI_102403', 'vehicle': {'trip': {'tripId': '0_12_801_8_10444'
_12_15001_150_10073', 'startDate': '20250723', 'scheduleRelationship': 'SCHEDULED', 'routeId': '150'}, 'position': {'lati
topSequence': 18, 'arrival': {'delay': -211, 'time': '1753268667'}, 'departure': {'delay': -211, 'time': '1753268667'}, '
'0_12_703_7_11289', 'startDate': '20250723', 'scheduleRelationship': 'SCHEDULED', 'routeId': '7'}, 'stopTimeUpdate': [{'s
tionship': 'SCHEDULED', 'routeId': '4'}, 'stopTimeUpdate': [{'stopSequence': 1, 'departure': {'delay': 0}, 'stopId': '272
ion': {'latitude': 45.76684, 'longitude': 16.000109}, 'timestamp': '1753268840', 'vehicle': {'id': '409'}}}, {'id': 'XFZK
ZKTZS0MW', 'tripUpdate': {'trip': {'tripId': '0_12_10601_106_10114', 'startDate': '20250723', 'scheduleRelationship': 'SC
'263'}, 'position': {'latitude': 45.91415, 'longitude': 16.12407}, 'timestamp': '1753268827', 'vehicle': {'id': '807'}}},
12_1707_17_11179', 'startDate': '20250723', 'scheduleRelationship': 'SCHEDULED', 'routeId': '17'}, 'stopTimeUpdate': [{'s
```


Slika 19. Izvorni izgled GTFS-RT zapisa u dekodiranom rječniku

Podatci dobiveni putem GTFS-RT protokola strukturirani su oko dvaju temeljnih entiteta: *tripUpdate* i *vehicle*. Iako funkcionalno različiti, ovi entiteti zajedno omogućuju cjelovit prikaz prometnog stanja svakog vozila u realnom vremenu. Entitet *tripUpdate* sadržava informacije o planiranom tijeku vožnje, uključujući identifikator putovanja (*tripId*), oznaku linije (*routeId*), redoslijed stajališta (*stopSequence*), identifikator sljedeće postaje (*stopId*) te moguće vremenske odmake poput kašnjenja (*delay*). Nasuprot tomu, entitet *vehicle* definira stvarni prostorni položaj vozila putem geografskih koordinata (*latitude*, *longitude*), uz ponovno navođenje pripadajućeg *tripId*-a i *routeId*-a. Iako unutar GTFS-RT zapisa postoje i dodatni atributi, upravo su ovi navedeni ključni za potrebe analize i vizualizacije unutar sustava.

Važno je naglasiti da se ovi entiteti ne prenose u zajedničkom zapisu, već dolaze kao zasebni JSON objekti s odvojenim identifikatorima (*id*), čime njihovo međusobno povezivanje zahtijeva dodatnu logiku. Na prvi pogled može se učiniti da zapisi pripadaju nepovezanim

jedinicama, no pomnijim uvidom vidljivo je da oba entiteta sadrže isti *tripId*, što omogućuje njihovu semantičku integraciju. Kao ilustracija takve situacije, na slici 20 prikazani su odvojeni zapisi tipa *tripUpdate* i *vehicle* koji se odnose na isto vozilo koje prometuje na liniji 134.


```
{
  "id": "X8ZKTZS0KW",
  "tripUpdate": {
    "trip": {
      "tripId": "0_12_13401_134_10312",
      "startDate": "20250723",
      "routeId": "134"
    },
    "stopTimeUpdate": [{
      "stopSequence": 1,
      "stopId": "611_27",
      "departure": { "delay": 0 }
    }],
    "timestamp": "1753268837"
  }
}

{
  "id": "X8ZKTZS0KW_576",
  "vehicle": {
    "trip": {
      "tripId": "0_12_13401_134_10312",
      "startDate": "20250723",
      "routeId": "134"
    },
    "position": {
      "latitude": 45.806293,
      "longitude": 15.923709
    },
    "timestamp": "1753268837",
    "vehicle": {
      "id": "576"
    }
  }
}
```


Slika 20. GTFS-RT skupovi podataka tripUpdate i vehicle

Premda oba zapisa referenciraju istu vožnju, razlikuju se u formalnoj strukturi svojeg *id* atributa. Dok *tripUpdate* koristi osnovni identifikator (*X8ZKTZS0KW*), zapis tipa *vehicle* uključuje dodatni sufiks (*X8ZKTZS0KW_576*). Ova razlika u strukturi nije bila očita u početnoj fazi razvoja, što je dovelo do problema u prikazu pozicije vozila. Iako su podatci o geografskim koordinatama bili dostupni, nisu bili povezani s pripadajućim *tripUpdate* zapisima zbog pogrešne pretpostavke da oba entiteta dijele isti *id*.

Kasnijom analizom uočeno je da se sufiks unutar *vehicle.id* koristi za razlikovanje poruka vezanih uz istu vožnju. Rješenje se temeljilo na primjeni jednostavne normalizacije identifikatora, pri čemu se vrijednost atributa *id* razdvaja na temelju znaka donje crte, a za

potrebe povezivanja uzima se samo prvi dio tako razdvojenog identifikatora, što je vidljivo na slici 21. Na taj su način entiteti uspješno povezani u zajedničku logičku cjelinu, čime je omogućena njihova integracija u daljnji proces obrade.

```python
if trip_update:
    data["id"] = original_id
elif vehicle_info:
    data["id"] = original_id.split('_')[0]

id_vozila = data.get("id")
```

Slika 21. Normalizacija identifikatora vozila

Na temelju tako povezanih informacija razvijena je tablica *TripUpdates*, koja predstavlja dinamički sloj baze podataka za pohranu svih relevantnih informacija o trenutačnom stanju vozila. U tu se tablicu pohranjuju podaci iz oba entiteta, *tripUpdate* i *vehicle*, čime se uspostavlja jedinstveni zapis za svako aktivno vozilo u stvarnom vremenu. Tablica je definirana SQL strukturom koja je prikazana na slici 22.

```sql
CREATE TABLE TripUpdates (
    id nvarchar(255) NOT NULL,
    tripId nvarchar(255),
    routeId nvarchar(255),
    stopId nvarchar(255),
    stopSequence int,
    departureDelay int,
    arrivalDelay int,
    latitude float,
    longitude float
);
```

Slika 22. Izrada tablice *TripUpdates* u SQL-u

Uz osnovne atribute preuzete iz GTFS-RT zapisa, tablica je proširena s tri dodatna stupca, kako je vidljivo na slici 23, čija je svrha unaprijediti podatkovni model i omogućiti dublju analizu kretanja vozila. Prvi od njih, *lastUpdated*, bilježi točan datum i vrijeme kada je određeni zapis zadnji put umetnut ili ažuriran.

Drugi dodatni atribut je *routeDirection*, koji specificira naziv krajnje postaje, odnosno smjer vožnje, kao što je primjerice „Črnomerec". Budući da ta informacija nije sadržana u GTFS-RT

zapisu, dohvaća se iz statičke GTFS tablice *trips* povezivanjem prema *tripId*-u. U toj se tablici nalazi atribut *trip_headsign*, koji služi kao izvor podatka o smjeru prometovanja.

Treći dodatni atribut, *vehicleDescription*, koristi se za klasifikaciju tipa vozila na temelju numeričke vrijednosti atributa *routeId*. Ako je ta vrijednost manja od 100, smatra se da je riječ o tramvajskoj liniji, dok se u suprotnom vozilo svrstava u kategoriju autobusa. Ova je klasifikacija implementirana radi omogućavanja vizualnog razlikovanja linija i stajališta prema tipu vozila, što se potom reflektira i u korisničkom sučelju mobilne aplikacije.

```
ALTER TABLE TripUpdates
ADD lastUpdated datetime NOT NULL DEFAULT GETDATE(),
    routeDirection nvarchar(100),
    vehicleDescription nvarchar(255);
```

Slika 23. Proširenje tablice *TripUpdates* dodatnim atributima

4.2.3 Periodičko dohvaćanje podataka

Budući da se dinamički GTFS-RT podatci emitiraju u vrlo kratkim vremenskim razmacima, u pravilu svakih pet do deset sekundi, bilo je potrebno osmisliti mehanizam koji omogućuje učinkovito dohvaćanje isključivo ažuriranih podataka. Vrijeme između pojedinih osvježavanja nije strogo definirano i ovisi o internom stanju sustava na strani pružatelja podatka (ZET). Zbog toga bi učestalo slanje zahtjeva bez prethodne provjere moglo rezultirati višestrukim preuzimanjem istog sadržaja, čime bi se nepotrebno opteretili mrežni kapaciteti i procesorski resursi bez stvarnog dobitka u novim informacijama.

Kako bi se takva situacija izbjegla, implementiran je sustav za provjeru ažuriranosti sadržaja temeljen na standardnim HTTP zaglavljima *ETag* i *Last-Modified*, što je prikazano na slici 24. Riječ je o zaglavljima koja predstavljaju jedinstvene oznake verzije sadržaja, a koje poslužitelj automatski generira i uključuje u svaki svoj odgovor, (13). Prilikom slanja novog zahtjeva prema GTFS-RT izvoru, backend aplikacija u zaglavlje uključuje zadnje poznate vrijednosti *ETag* ili *Last-Modified*, koje je prethodno pohranila. Ako se sadržaj na strani poslužitelja nije promijenio, vraća se statusni kod *304 Not Modified*, čime se signalizira da nema potrebe za ponovnim preuzimanjem podataka. U takvom slučaju program preskače daljnju obradu jer već raspolaže ažurnom verzijom sadržaja.

```python
global last_etag, last_modified
try:
    headers = {}
    if last_etag:
        headers["If-None-Match"] = last_etag
    elif last_modified:
        headers["If-Modified-Since"] = last_modified

    response = requests.get(feed_url, headers=headers, timeout=(2, 3))

    # Ako server kaže da se sadržaj nije promijenio
    if response.status_code == 304:
        print("[INFO] GTFS feed nije se promijenio (ETag / Last-Modified). Preskačem obradu.")
        return None

    response.raise_for_status()

    # Spremi nove zaglavlja ako postoje
    if "ETag" in response.headers:
        last_etag = response.headers["ETag"]
    if "Last-Modified" in response.headers:
        last_modified = response.headers["Last-Modified"]


except Exception as e:
    print(f"Greška pri preuzimanju GTFS feeda: {e}")
    return None
```

Slika 24. Provjera ažuriranosti GTFS-RT podataka pomoću HTTP zaglavlja

Opisani mehanizam omogućuje selektivnu obradu isključivo onih podataka koji su se zaista promijenili od posljednjeg zahtjeva. Na taj se način postiže višestruka optimizacija jer se smanjuje količina prenesenih podataka, ubrzava rad aplikacije i rasterećuju se mrežni i procesorski resursi, što je osobito važno s obzirom na visoku frekvenciju emitiranja GTFS-RT zapisa.

Kako bi proces dohvaćanja podataka bio u potpunosti automatiziran, razvijena je periodička funkcija pod nazivom *fetch_periodically*, prikazana na slici 25, koja svakih tri sekunde pokreće dohvat podataka iz GTFS izvora. Unutar te funkcije poziva se metoda *fetch_feed_data*, u kojoj se provodi prethodno opisana logika provjere ažuriranosti na temelju HTTP zaglavlja, čime se osigurava da se preuzimaju isključivo novi ili izmijenjeni zapisi.

```python
def fetch_periodically():
    """Periodički dohvaća podatke iz GTFS feeda i čisti stare podatke."""
    counter = 0
    while True:

        feed_dict = fetch_feed_data()
        if feed_dict:
            process_feed_data(feed_dict)

        if counter % 20 == 0 or counter == 0:  # svakih 20 ciklusa (20 * 3s = ~60s)
            delete_old_data()

        counter += 1

        time.sleep(3)
```

Slika 25. Funkcija *fetch_periodically*

Uz dohvat novih podataka, funkcija *fetch_periodically* uključuje i dodatni mehanizam za održavanje baze podataka. Svakih dvadeset ciklusa, odnosno otprilike svakih 60 sekundi, poziva se funkcija *delete_old_data* čija je zadaća automatsko uklanjanje zapisa o vozilima koja više nisu aktivna. Na taj se način sprječava nepotrebno zadržavanje zastarjelih podataka u tablici *TripUpdates*.

Funkcija *delete_old_data*, vidljiva na slici 26, briše sve retke u kojima vrijednost atributa *lastUpdated* nije promijenjena u posljednjih pet minuta. Taj atribut bilježi trenutak posljednje izmjene svakog zapisa, odnosno vrijeme kada su podatci o određenom vozilu zadnji put ažurirani. Kada vozilo završi svoju vožnju, najčešće dolaskom na završno stajalište, prestaje emitirati nove podatke i navedeni se atribut više ne ažurira. Sustav to prepoznaje kao indikator neaktivnosti te automatski uklanja takve zapise, čime se baza održava ažurnom i resursno učinkovitim.

```python
def delete_old_data():
    """Deletes records older than 5 minutes from the TripUpdates table."""
    try:
        with pyodbc.connect(connection_string) as conn:
            with conn.cursor() as cursor:
                delete_query = """
                    DELETE FROM TripUpdates
                    WHERE lastUpdated < DATEADD(MINUTE, -5, GETDATE())
                """
                cursor.execute(delete_query)
                conn.commit()
                print("[SQL] Stari podaci izbrisani.")
    except Exception as e:
        print(f"Greška pri brisanju starih podataka: {e}")
```

Slika 26. Funkcija *delete_old_data*

4.2.4 Implementacija privremene memorijske strukture

Unatoč kontinuiranom ažuriranju i naizgled stabilnoj infrastrukturi za prijenos podataka u stvarnom vremenu, tijekom razvoja sustava uočeni su značajni nedostaci u preciznosti pojedinih atributa, osobito atributa *stop_id*, koji bi trebao označavati točno stajalište prema kojem se vozilo kreće. Iako je svaki *trip_id* u GTFS specifikaciji unaprijed povezan s precizno definiranim redoslijedom stajališta i predviđenim vremenima dolaska i odlaska (navedenima u tablici *stop_times*), stvarna dinamika prometa često odstupa od tog plana. Takva odstupanja uzrokuju logičke nedosljednosti jer se stvarni položaj vozila na terenu ne podudara s očekivanim redoslijedom kretanja definiranog u statičkom voznom redu. Naime, zapisi unutar *tripUpdate* entiteta temelje se na planiranom redoslijedu stajališta iz tablice *stop_times*, ne uzimajući pritom u obzir trenutačne GPS koordinate koje realno odražavaju poziciju vozila. U praksi to znači da sustav za određeno vozilo može navoditi *stop_id* stajališta koje je ono već napustilo, iako se na temelju stvarne lokacije vozila, zajedno s informacijom o ruti i smjeru, može pouzdano zaključiti koje je stajalište zapravo sljedeće.

Kako bi se taj problem otklonio i omogućilo precizno određivanje sljedećeg stajališta za svako vozilo, razvijen je mehanizam koji povezuje podatke u stvarnom vremenu s pripadajućim statičkim tablicama GTFS strukture. Ključni podatak u tom postupku je *trip_id*, koji omogućuje identifikaciju konkretnog putovanja. Na temelju tog identifikatora iz tablice *trips* dohvaća se

pripadajući *shape_id*, nakon čega se s pomoću te vrijednosti pristupa tablici *shapes* i dohvaćaju svi zapisi čiji *shape_id* odgovara odabranome. Svaki zapis u tablici predstavlja pojedinačnu točku trase, a njihovim povezivanjem u zadanom redoslijedu oblikuje se neprekinuta geografska putanja koja definira trajektoriju kretanja vozila.

Paralelno s postupkom konstruiranja geografske putanje rute, iz tablice *stop_times* dohvaća se redoslijedni popis svih stajališta (*stop_id*) koje vozilo posjećuje tijekom svojeg putovanja, pri čemu se kao ključ koristi *trip_id*. Na temelju tako dobivenih *stop_id* vrijednosti pristupa se tablici *stops*, iz koje se za svako stajalište preuzimaju njegove geografske koordinate. Na taj se način svako stajalište može prostorno smjestiti i povezati s prethodno definiranom putanjom kretanja. Svako stajalište se mapira na onaj segment trase koji mu je prostorno najbliži, a udaljenost između stajališta i segmenata računa se korištenjem Haversineove formule. Time se za svaku stanicu određuje konkretan segment trase na kojem se ona nalazi, čime se ostvaruje veza između statički definiranih stajališta i stvarne geografske putanje vozila.

U trenutku kada sustav zaprimi trenutačnu lokaciju vozila (atributi *latitude* i *longitude*) putem real-time feeda, ta se pozicija mapira na najbliži segment trase. Dobiveni rezultat upisuje se u novo dodani atribut *shapeSegment_currentPosition* unutar tablice *TripUpdates*.

Usporedbom tog segmenta s onima pridruženima stajalištima, sustav zatim može jednostavno odrediti koje je sljedeće stajalište na ruti. Traži se prva stanica čiji je segment veći od onoga na kojem se trenutno nalazi vozilo. Na primjer, ako je vozilo na segmentu 9, a stanice su raspoređene na segmentima 2, 5, 8, 11 i 14, sustav, prema toj logici, zaključuje da je sljedeće stajalište ono na segmentu 11. Ti se podatci pohranjuju u dva dodatna atributa tablice *TripUpdates*, pri čemu *next_stop_id* označava identifikator sljedeće stanice, a *shapeSegment_nextStopId* bilježi pripadajući segment.

S obzirom na to da sustav svakih nekoliko sekundi zaprima ažurirane podatke za stotine vozila, neprestano ponovno dohvaćanje i obrada statičkih tablica za svaki zapis predstavljali bi ozbiljno računalno opterećenje. Kako bi se to izbjeglo, razvijen je mehanizam privremene memorije (*trip cache*) koji zadržava ključne informacije o vozilu jednom kada postanu dostupne. Time se omogućuje višekratna uporaba već obrađenih podataka i smanjuje potreba za pristupom bazi prilikom svakog ažuriranja.

Kada sustav zaprimi novi zapis, neovisno o tome je li riječ o *tripUpdate* ili *vehicle* entitetu, najprije se provodi normalizacija identifikatora vozila uklanjanjem sufiksa nakon znaka donje

crte. Time se osigurava dosljedna referenca između oba tipa zapisa. Potom se provjerava postoji li u memoriji već spremljen zapis za to vozilo. Ako ne postoji, memorijska se struktura inicijalizira korištenjem *trip_id*-a, prema već opisanoj logici mapiranja trase i stajališta.

Jednom kada je memorijska struktura inicijalizirana, sustav više ne mora dohvaćati podatke iz statičkih GTFS tablica, već ih može preuzimati iz privremene memorije. Prilikom zaprimanja novog *vehicle* zapisa moguće je odmah odrediti trenutačni položaj vozila na ruti i sljedeće stajalište, bez dodatnih izračuna. U tablicu *TripUpdates* tada se ažuriraju tri podatka: *shapeSegment_currentPosition*, *next_stop_id* i *shapeSegment_nextStopId*. U slučaju da je zaprimljeni zapis tipa *tripUpdate*, moguće je inicijalizirati memorijsku strukturu, ali nije moguće izvesti geolokacijsku obradu jer nedostaju trenutačne koordinate. U tom slučaju, pozicija vozila i određivanje sljedećeg stajališta odgađaju se do primitka odgovarajućeg *vehicle* zapisa.

Učinkovitost ovog mehanizma omogućuje obradu podataka za stotine vozila svakih nekoliko sekundi bez zastoja i suvišnih upita prema bazi. Iako se u ovoj fazi korištenja memorijske strukture možda ne čini presudnim znati koje je stajalište sljedeće, u nastavku rada bit će prikazano da upravo ta informacija čini temelj brojnih funkcionalnosti unutar mobilne aplikacije.

4.2.5 Proširenje statičke tablice *stops*

Tijekom razvoja aplikacije uočeno je kako podaci sadržani u statičkoj GTFS tablici *stops*, premda strukturno informativni, nisu bili dostatni za implementaciju svih planiranih funkcionalnosti. Naime, jedan od ključnih zahtjeva bio je omogućiti korisniku uvid u linije javnog prijevoza koje prometuju kroz pojedino stajalište. Iako je *stops* sadržavala osnovne atribute poput *stop_name*, *stop_lat*, *stop_lon* i *parent_station*, nije uključivala informacije o linijama koje zaista opslužuju točno određenu lokaciju. Kako bi se nadoknadio taj nedostatak, tablica je proširena s dvama dodatnim atributima: *vehicleNumber* i *vehicleDescription*, što je prikazano na slici 27.

```
ALTER TABLE stops
ADD vehicleNumber nvarchar(255),
    vehicleDescription nvarchar(255);
```

Slika 27. Proširenje tablice *stops* dodatnim atributima

Atribut *vehicleNumber* definira popis linija koje prometuju kroz stajalište. Za glavne stanice taj se popis sastoji isključivo od identifikatora linija, dok se za podstanice bilježe i smjerovi prometovanja, čime se omogućuje precizna identifikacija vozila prema odredištu. Dodatno, *vehicleDescription* predstavlja klasifikacijski atribut koji određuje vrstu vozila koje opslužuje pojedino stajalište ovisno o linijama koje kroz njega prolaze, pri čemu se razlikuju tramvajska i autobusna stajališta.

U početnoj fazi razvoja planirano je da se ovi atributi automatski popunjavaju tijekom svakog ažuriranja podataka u stvarnom vremenu. Budući da svaki zapis o vozilu uključuje i vrijednost *stop_id*, smatralo se izvedivim povezati zapis s odgovarajućim stajalištem iz tablice *stops* i unutar atributa *vehicleNumber* upisati vrijednost *route_id*, ukoliko ona ondje još nije bila zabilježena. Međutim, praksa je pokazala da je ovakav pristup nepouzdan. Atribut *stop_id* u zapisima u stvarnom vremenu često nije precizno odgovarao stvarnoj lokaciji stajališta kroz koje je vozilo u tom trenutku prolazilo. Ipak, ta netočnost nije predstavljala značajnu prepreku za ažuriranje *vehicleNumber*, jer nije bilo nužno da se *stop_id* u potpunosti podudara s lokacijom vozila u trenutku zapisa. Bilo je dovoljno da se određena linija u bilo kojem dijelu svojeg planiranog putovanja referencira na to stajalište, budući da su vrijednosti *stop_id* određene unaprijed, temeljem sekvenci iz tablice *stop_times*.

Pravi se problem javlja na završnim dionicama trase. Kada vozilo dosegne krajnju stanicu, prestaje odašiljati podatke, a s obzirom na to da vrijednost *stop_id* često zaostaje za stvarnim položajem vozila, posljednja stajališta na ruti nisu bila obuhvaćena. Kao posljedica toga, navedena stajališta nisu uključena u proces ažuriranja te atribut *vehicleNumber* za ta stajališta ostaje nepopunjen, iako ih je vozilo u stvarnosti posjetilo.

Ova je nedosljednost otklonjena implementacijom mehanizma za određivanje *next_stop_id* na temelju trenutačne lokacije vozila, čija je logika prethodno objašnjena. Ovaj pristup omogućuje preciznije ažuriranje tablice *stops*, neovisno o pouzdanosti *stop_id* zapisa u real-time podacima. Prilikom svake obrade poruke provjerava se postoji li unutar atributa *vehicleNumber* za dani *next_stop_id* već zabilježeni *route_id*, a u slučaju podstanica i pripadajući *routeDirection*. Podaci se pohranjuju u obliku sortirane liste brojeva linija, odvojenih zarezima (na primjer *"2,6,11"*), dok se kod podstanica koristi oblik *route_id–routeDirection* (primjerice *"11–Dubec"*) radi kasnije obrade.

Treba naglasiti da ažuriranje *vehicleNumber* atributa ne mora biti kontinuirano. Budući da vozila redovito prometuju u kružnim trasama, empirijski je utvrđeno da vremenski okvir od jednog do dva sata omogućuje da svako stajalište bude obuhvaćeno barem jednom pojavom svake linije. Stoga je mehanizam popunjavanja aktiviran samo tijekom inicijalne faze rada sustava.

Po završetku inicijalnog ažuriranja, sustav dodatno određuje vrijednost atributa *vehicleDescription*, koji omogućuje klasifikaciju stajališta prema tipu vozila. Kriterij klasifikacije temelji se na obrascu prema kojem se linije s brojevima manjima od sto odnose na tramvaje, dok se ostale svrstavaju među autobusne. Takva kategorizacija omogućila je implementaciju dodatnih funkcionalnosti u korisničkom sučelju, poput diferencijacije ikona na karti ili filtriranja stajališta prema vrsti prijevoza.

4.2.6 Izrada tablice *TripHistory* za potrebe predikcije vremena dolaska vozila

Uz tablicu *TripUpdates*, koja se kontinuirano ažurira s ciljem bilježenja zapisa trenutačnog stanja vozila u prometu, razvijena je i pomoćna tablica *TripHistory*. Njezina je struktura prikazana na slici 28 i u potpunosti odgovara strukturi tablice *TripUpdates*, no za razliku od nje, u *TripHistory* se podaci ne ažuriraju, već se trajno pohranjuju u obliku novih zapisa. Time se formira povijesna baza podataka koja može poslužiti za različite oblike naknadne analize, pri čemu je njezina glavna svrha izračun predviđenog vremena dolaska vozila na stajališta.

```
CREATE TABLE TripHistory (
    id nvarchar(255) NOT NULL,
    tripId nvarchar(255),
    routeId nvarchar(255),
    stopId nvarchar(255),
    stopSequence int,
    departureDelay int,
    arrivalDelay int,
    latitude float,
    longitude float,
    lastUpdated datetime NOT NULL DEFAULT GETDATE(),
    routeDirection nvarchar(100),
    vehicleDescription nvarchar(255),
    nextStopId nvarchar(255),
    shapeSegment_nextStopId int,
    shapeSegment_currentPosition int
);
```

Slika 28. Izrada tablice *TripHistory* u SQL-u

Svaki zapis unutar tablice *TripHistory* odgovara jednom stanju vozila u određenom trenutku na točno određenom segmentu njegove rute. Međutim, uvjet za unos zapisa jest njegova jedinstvenost u pogledu kombinacije vrijednosti *trip_id* i *shapeSegment_currentPosition*. Prilikom obrade podataka u stvarnom vremenu sustav provjerava postoji li u tablici već zapis s istim identifikatorom vožnje i istim segmentom na ruti. Ako takav zapis ne postoji, kreira se novi unos, dok se u suprotnom podatak ne pohranjuje kako bi se izbjeglo nepotrebno gomilanje duplikata.

Opisani pristup omogućuje kontrolirano i podatkovno efikasno prikupljanje reprezentativnih primjera koji su relevantni za kasniju analizu. Budući da je za pouzdanu predikciju vremena dolaska dovoljno imati jedan reprezentativni primjer kretanja vozila za svaki segment unutar svake pojedine rute, višestruko bilježenje istih stanja nije potrebno niti poželjno. Na taj se način ostvaruje optimalna ravnoteža između količine pohranjenih podataka i njihove analitičke vrijednosti u kontekstu algoritama predikcije.

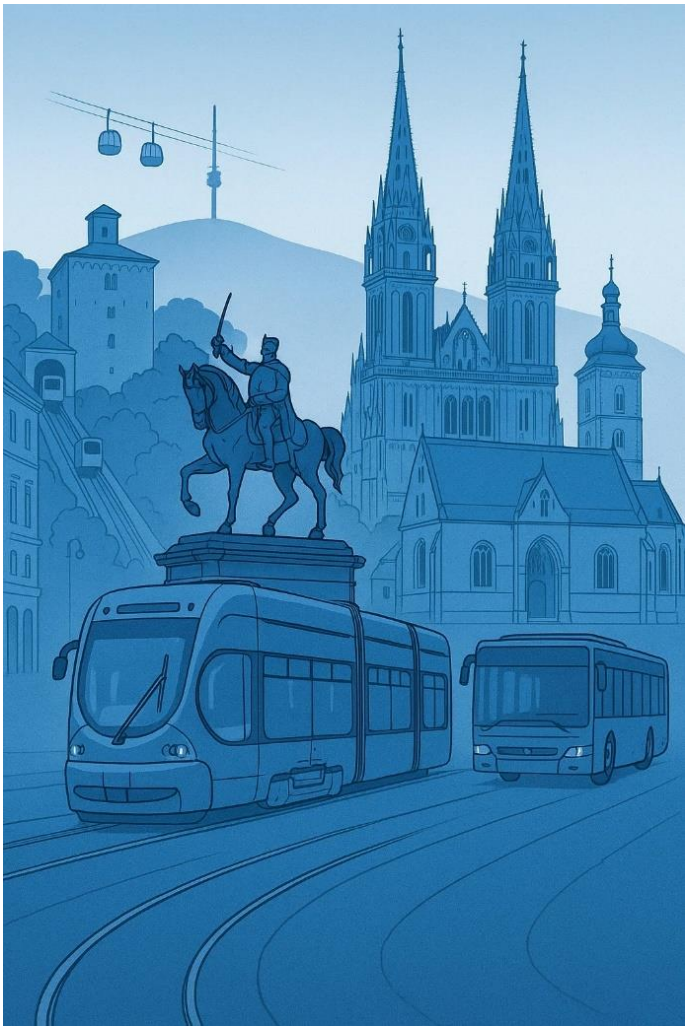
5 FUNKCIONALNOSTI I KORISNIČKO SUČELJE APLIKACIJE

Mobilna aplikacija predstavlja krajnji sloj interakcije između korisnika i cjelokupnog informacijskog sustava, a njezin dizajn i funkcionalnosti ključni su za dostupnost i razumljivost prometnih podataka u stvarnom vremenu. Nakon prikaza arhitekture i pozadinskih mehanizama koji omogućuju dohvat i obradu informacija, ovo se poglavlje usredotočuje na korisničko iskustvo, vizualni identitet i funkcionalne mogućnosti same aplikacije.

Poseban naglasak tijekom razvoja aplikacije stavljen je na oblikovanje korisničkog sučelja koje je istovremeno pregledno, funkcionalno i jednostavno za korištenje. Sučelje slijedi principe minimalističkog dizajna, pri čemu su izbjegnuti svi suvišni elementi, a u prvi plan stavljene su funkcije koje su korisniku doista potrebne. Cilj je bio postići što prirodnije i intuitivnije korisničko iskustvo, kako bi se aplikacija mogla koristiti bez potrebe za dodatnim objašnjenjima ili prethodnim tehničkim znanjem.

Vizualni identitet aplikacije osmišljen je s namjerom da na jasan način odražava povezanost s Gradom Zagrebom i njegovim javnim gradskim prijevozom. U tu je svrhu izrađen vlastiti logotip (slika 29), kao i uvodni zaslon (slika 30) koji se prikazuje prilikom pokretanja aplikacije, dok se njezin sadržaj učitava. Oba vizualna elementa izrađena su uz pomoć umjetne inteligencije, (14) a njihova se estetika temelji na simbolima zagrebačkog javnog prijevoza, uključujući prepoznatljive tramvaje i autobuse koji prometuju pod okriljem ZET-a, kao i dodatna vozila poput uspinjače i žičare. Osim toga, u grafički identitet uključeni su i motivi karakteristični za Grad Zagreb, među kojima se ističu skulptura bana Josipa Jelačića, crkva svetog Marka, Zagrebačka katedrala, kula Lotrščak te vrh Sljemena s prepoznatljivim telekomunikacijskim tornjem. Takav dizajn nije bio usmjeren samo na estetiku, već i na stvaranje vizualne prepoznatljivosti i isticanje lokalnog identiteta u kontekstu javnog gradskog prijevoza.

Slika 29. Logotip aplikacije, *(14)*



Slika 30. Uvodni zaslon aplikacije, *(14)*

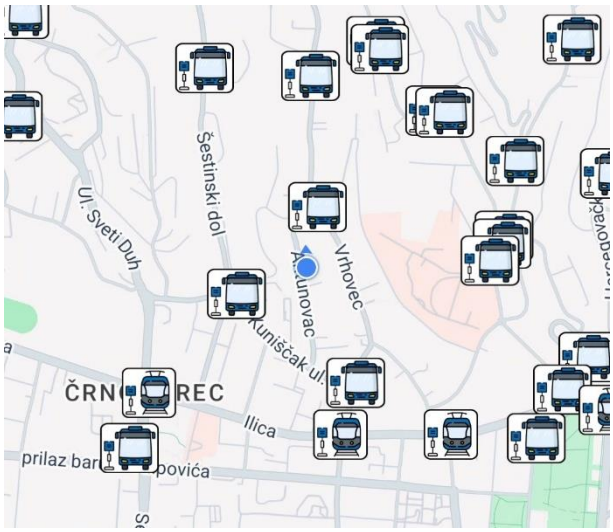
Aplikacija sadrži pet glavnih zaslona, a to su početni zaslon, prikaz voznog reda, kupnja karata, korisnički profil i pregled obavijesti. Kretanje između tih zaslona omogućeno je pomoću donje navigacijske trake, prikazane na slici 31, koja je stalno prisutna i fiksno smještena u donjem dijelu zaslona. Svaka stavka unutar trake označena je odgovarajućom ikonom i nazivom zaslona, čime se korisniku omogućuje brzo i intuitivno snalaženje unutar aplikacije. Ikone korištene u navigacijskoj traci preuzete su s platforme *Freepik*, kao i sve ostale ikone korištene unutar aplikacije. Sve su ikone dodatno prilagođene tako da odgovaraju vizualnom identitetu aplikacije, pri čemu je posebna pažnja posvećena dosljednosti u stilu i korištenju iste obitelji ikona, kako bi se postigla što veća estetska ujednačenost i profesionalan vizualni dojam.

Slika 31. Navigacijska traka

5.1 Početni zaslon

Početni zaslon predstavlja središnje mjesto aplikacije, odnosno prostor na kojem se odvija najveći dio interakcije između korisnika i prometnih podataka. Prilikom svakog pokretanja aplikacije, korisnik se automatski usmjerava upravo na ovaj zaslon. Vizualnu osnovu čini interaktivna karta, izrađena uz pomoć Google Maps servisa, na kojoj se prikazuju sve relevantne informacije o stajalištima i prometnoj situaciji u stvarnom vremenu.

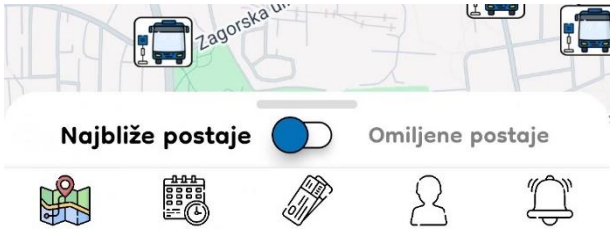
Odmah po pokretanju, aplikacija traži dopuštenje za pristup GPS lokaciji korisnika. Ta se informacija koristi kako bi se karta automatski centrirala na korisnikovu trenutačnu lokaciju, uz blago povećanje prikaza radi boljeg pregleda neposredne okoline. Istovremeno aplikacija šalje zahtjev prema API-u (*/get_all_stops*), kako bi se dohvatili svi zapisi stajališta iz baze podataka, konkretno iz tablice *stops*. Nakon što se podaci učitaju, API ih vraća aplikaciji, koja zatim stajališta prikazuje na karti u obliku ikona. Prikazuju se isključivo glavna stajališta kako bi se očuvala preglednost i izbjeglo pretrpavanje karte informacijama, što je vidljivo na slici 32.



Slika 32. Inicijalni prikaz karte

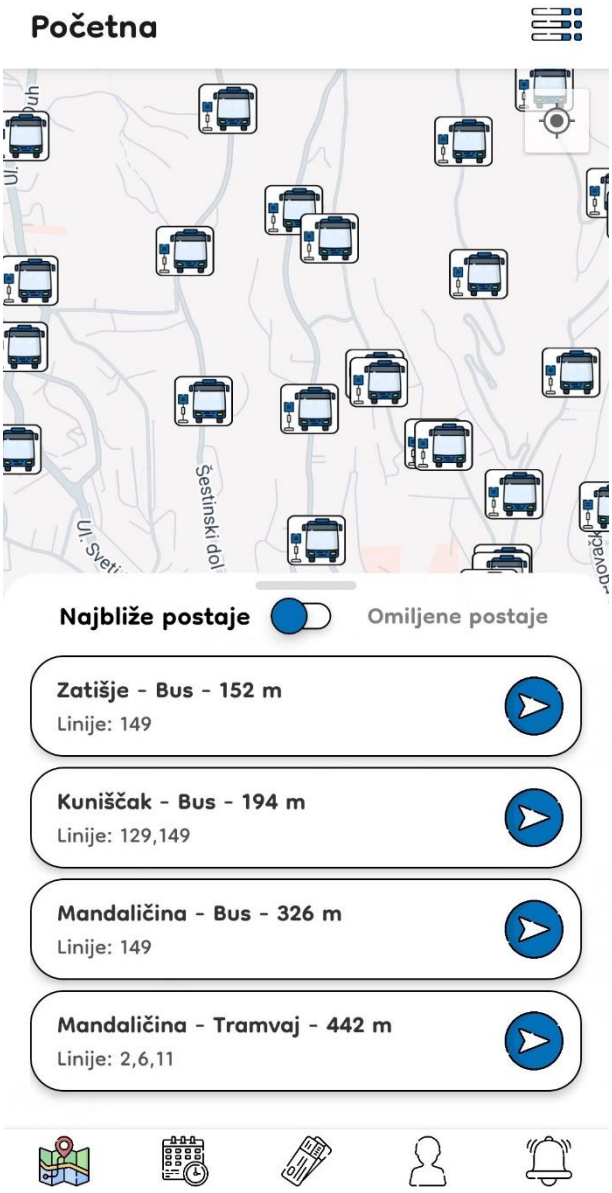
Ikone stajališta u aplikaciji vizualno su diferencirane prema vrsti vozila koje ih opslužuje, odnosno razlikuju se za tramvajska i autobusna stajališta. Ova je podjela omogućena uvođenjem atributa *vehicleDescription* unutar tablice *stops*, koji se popunjava na temelju linija koje prolaze kroz svako pojedino stajalište. Bez takve klasifikacije ne bi bilo moguće korisniku izravno na karti prikazati kojoj vrsti javnog prijevoza određeno stajalište pripada.

U donjem dijelu zaslona nalazi se klizni izbornik, prikazan na slici 33, koji je u početnom stanju spušten i smješten neposredno iznad donje navigacijske trake. Riječ je o središnjem elementu početnog zaslona putem kojeg korisnik pristupa većini funkcionalnosti unutar aplikacije. Klizni se izbornik dinamički mijenja ovisno o korisničkim radnjama i odabranim opcijama, što će biti detaljnije prikazano u nastavku razrade početnog zaslona. U svojem početnom prikazu, izbornik omogućuje pregled najbližih i omiljenih stajališta, između kojih se korisnik može jednostavno prebacivati pomoću interaktivnog prekidača. Taj je prekidač izrađen uz pomoć animacijske biblioteke i omogućuje prirodnu promjenu prikaza bilo dodirom, bilo povlačenjem prsta po zaslonu.



Slika 33. Izgled spuštenog kliznog izbornika

Kada se izbornik otvori, u prikazu „*Najbliže postaje*” korisniku se prikazuju četiri stajališta koja su najbliža njegovoj trenutačnoj lokaciji, što je vidljivo na slici 34. Ta se udaljenost računa na strani aplikacije, na temelju koordinata dohvaćenih putem API-a, a stanice se prikazuju prema rastućem redoslijedu udaljenosti. Ako korisnik pređe na prikaz „*Omiljene postaje*”, prikazuju se stajališta koja je prethodno označio kao favorite. Prilikom prvog korištenja aplikacije taj će popis biti prazan, ali se automatski popunjava kako korisnik dodaje stajališta u svoje favorite tijekom uporabe.



Slika 34. Prikaz cijelog početnog zaslona s otvorenim kliznim izbornikom

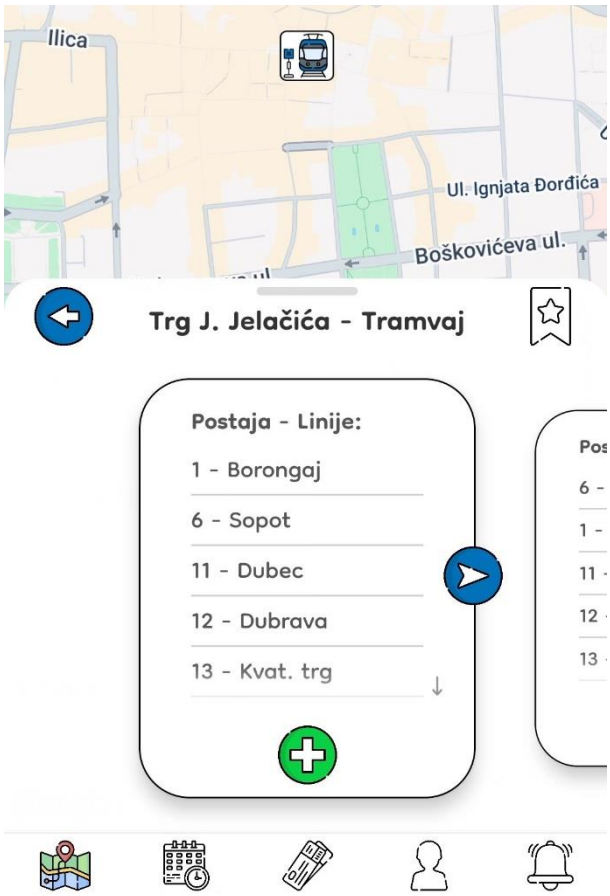
5.1.1 Odabir željene postaje

Odabir stanice unutar aplikacije može se ostvariti na dva načina, ovisno o tome želi li korisnik odabrati stanicu izravno na karti ili iz pregleda dostupnih stajališta unutar kliznog izbornika. U slučaju odabira putem karte, korisnik klikne na ikonu stanice prikazanu na karti, nakon čega se prikaz automatski centrira na odabranu lokaciju. Klizni se izbornik pritom smanjuje i prikazuje osnovne informacije o toj stanici, uključujući njezin naziv, vrstu vozila (tramvaj ili autobus) te popis linija koje ju opslužuju, što je prikazano na slici 35. S lijeve strane prikazanih informacija nalazi se zelena ikona koja omogućuje potvrdu odabira, dok se s desne strane nalazi crvena ikona za poništavanje, čijim se pritiskom korisnik vraća na prethodni prikaz.

Slika 35. Izbornik za potvrdu odabrane postaje

Druga mogućnost odabira postaje temelji se na korištenju kliznog izbornika, unutar kojeg korisnik može odabrati stanicu iz kartica prikazanih u kategorijama najbližih ili omiljenih postaja. Svaka kartica sadrži osnovne podatke o stanici, uključujući njezin naziv, udaljenost od korisnikove trenutačne lokacije te linije koje kroz nju prometuju. S desne strane kartice smještena je dodatna ikona koja korisniku signalizira da se klikom na nju odabire pripadajuća stanica.

Nakon što korisnik odabere stanicu, prikaz na karti se mijenja tako da nestaju sve ostale stanice, a ostaje prikazana isključivo ona koju je korisnik odabrao. Karta se automatski centrira i uvećava na tu lokaciju, čime se osigurava jasan fokus na odabrano stajalište. Klizni izbornik se pritom otvara (ako već nije bio otvoren), a njegov se sadržaj ažurira kako bi prikazivao sve relevantne informacije vezane uz tu stanicu, što je vidljivo na slici 36.



Slika 36. Prikaz zaslona pri odabiru željene stanice

U gornjem dijelu izbornika prikazuje se zaglavlje koje obuhvaća naziv stanice uz oznaku vrste vozila, ikonu za povratak na prethodni prikaz te ikonu za dodavanje stanice među omiljene. Kada korisnik označi stanicu kao omiljenu, ikona se ispunjava plavom bojom, čime se vizualno naglašava aktivacija te funkcije. Sustav za pohranu favorita temelji se na lokalnoj memoriji uređaja, putem mehanizma *AsyncStorage*, što omogućuje spremanje omiljenih stajališta bez potrebe za prijavom ili stvaranjem korisničkog računa. Time aplikacija ostaje jednostavna za korištenje, a funkcionalnost dostupna svim korisnicima bez dodatnih preduvjeta. Stanica označena kao omiljena automatski se dodaje u prikaz „*Omiljene postaje*“ unutar kliznog izbornika, prikazano na slici 37, gdje ostaje dostupna za buduće korištenje.

Slika 37. Korištenje opcije favoriziranja stanica

U središnjem dijelu izbornika prikazuju se sve podstanice koje pripadaju odabranoj glavnoj stanici, a predstavljaju fizička stajališta raspoređena prema smjerovima prometovanja. Njihov prikaz temelji se na strukturi podataka koja ih povezuje s glavnom stanicom putem atributa *parent_station*. Za svaku podstanicu prikazane su linije koje kroz nju prometuju, zajedno s pripadajućim smjerovima vožnje, a ti se podatci dohvaćaju iz atributa *vehicleNumber* pridruženog stanici. Navigacija između više podstanica omogućena je horizontalnim pomicanjem unutar izbornika ili klikom na strelice smještene s lijeve i desne strane prikaza, ovisno o tome na kojoj se kartici podstanice korisnik trenutačno nalazi. Kada korisnik želi pregledati vozila koja dolaze prema odabranoj podstanici, dovoljno je da odabere zelenu ikonu s oznakom plus, čime se pokreće funkcionalnost prikaza i praćenja najbližih vozila u stvarnom vremenu.

5.1.2 Prikaz najbližih vozila za odabranu postaju

Jedna od ključnih funkcionalnosti razvijene mobilne aplikacije jest mogućnost prikaza vozila koja su u danom trenutku najbliža odabranoj podstanici i koja se kreću prema njoj. Ova značajka korisniku omogućuje da u stvarnom vremenu prati približavanje vozila svih linija koje prometuju kroz promatranu stanicu, što korisniku omogućuje precizniji uvid u prometnu situaciju i dolazak željenog vozila.

Nakon što korisnik aktivira ovu funkcionalnost, aplikacija šalje zahtjev prema API-u (*/get_closest_vehicles*), s identifikatorom odabrane podstanice. Na temelju te vrijednosti API pristupa tablici *stops*, iz koje dohvaća pripadajući zapis o stanici i pristupa atributu *vehicleNumber*. Taj atribut sadrži popis svih ruta koje prometuju kroz zadano stajalište, pri čemu su linije zapisane u obliku tekstualnog niza koji kombinira broj linije i smjer

prometovanja. Ovaj se niz programski razdvaja, a za svaku pojedinu rutu identificira se broj linije i njezin pripadajući smjer.

Za svaku od tako dobivenih ruta pokreće se optimizirani upit nad tablicom *TripUpdates*, pri čemu se filtriraju isključivo ona vozila koja prometuju na konkretnoj liniji i u određenom smjeru, te koja u zapisima posjeduju poznate geografske koordinate. Na taj način formira se podskup aktivnih vozila koja se, u teoriji, kreću prema zadanoj stanici.

Međutim, činjenica da se određeno vozilo nalazi na ruti koja prolazi kroz stanicu ne znači nužno da se kreće prema toj stanici. Stoga je bilo potrebno razviti logiku koja razlikuje vozila koja su stanicu već prošla od onih koja joj se još približavaju. Ta se odluka temelji na usporedbi trenutačnog segmenta trase na kojem se vozilo nalazi s pripadajućim segmentom same stanice. Trenutačni segment vozila prethodno je izračunat i pohranjen u atributu *shapeSegment_currentPosition* unutar tablice *TripUpdates*. S druge strane, segment stanice određuje se na temelju informacija pohranjenih u privremenoj memorijskoj strukturi specifičnoj za svako putovanje. Budući da je svako vozilo jednoznačno određeno putem identifikatora *trip_id*, dovoljno je kombinirati taj identifikator sa *stop_id* stanice kako bi se iz memorije dohvatila vrijednost segmenta na kojem se stanica nalazi. Ta je veza prethodno ustanovljena mapiranjem svake stanice na najbliži segment geografske putanje pridružene danom *trip_id*-u. Nakon što su oba segmenta poznata, dovoljno ih je usporediti. Ako je segment vozila veći od segmenta stanice, to znači da je vozilo već prošlo stanicu i takvo vozilo se odbacuje. U suprotnom, vozilo je još uvijek na putu prema stanici i uzima se u obzir.

U slučaju da za pojedinu rutu postoji više aktivnih vozila koja još nisu prošla stanicu, odabire se ono koje se nalazi najbliže stajalištu, što se određuje usporedbom vrijednosti trenutačnih segmenata. Na taj se način za svaku liniju identificira jedno jedino vozilo koje je najbliže promatranoj podstanici.

Osim osnovnih informacija kao što su identifikator putovanja, broj linije, naziv smjera i geografske koordinate vozila, aplikacija istovremeno dohvaća i dodatne podatke iz privremene memorije, među kojima se nalaze cijela trasa kretanja vozila (*full_route*), već prijeđeni dio puta (*to_vehicle_route*), dionica koja još slijedi (*from_vehicle_route*), segment od trenutne pozicije vozila do odabrane stanice (*to_stop_route*) te popis svih stajališta koja slijede. Ovi se podatci strukturiraju u obliku JSON objekta koji se potom prosljeđuje aplikaciji.

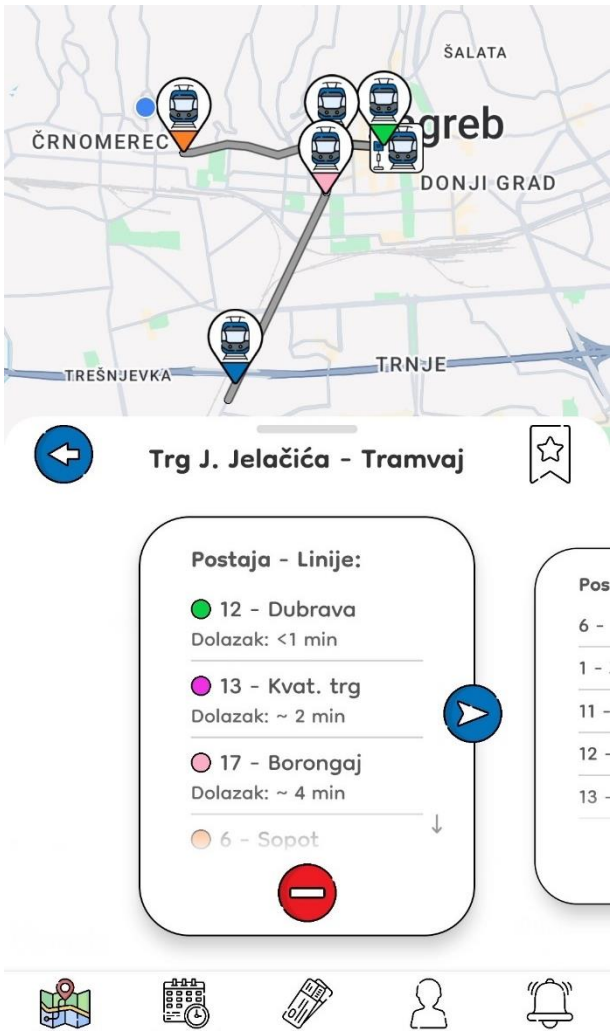
Nakon primitka podataka, korisničko sučelje se prilagođava novom načinu prikaza, što je vidljivo na slici 38. Na karti se pojavljuju ikone vozila koje se kreću prema stanici, pri čemu je tip vozila (autobus ili tramvaj) vizualno diferenciran na temelju prethodno definiranog atributa *vehicleDescription*. Boja ikone određuje se prema pripadajućoj liniji, čime se postiže dodatna preglednost i vizualna koherentnost prikaza. Uz same ikone vozila, na karti se prikazuju i njihove pripadajuće trajektorije, odnosno segmenti geografskih putanja od trenutačnih pozicija vozila do odabrane stanice prema kojoj se vozila kreću. Te su putanje prikazane sivom linijom, čime se na karti jasno ističu rute kojima će se vozila kretati, a korisniku se time omogućuje lakše prostorno anticipiranje njihova dolaska na odabrano stajalište.

Sve dok je neka podstanica aktivno odabrana, aplikacija kontinuirano šalje zahtjeve prema API-u (*/get_closest_vehicles*) u intervalima od pet sekundi, čime se omogućuje prikaz kretanja vozila u gotovo stvarnom vremenu. Paralelno s kartografskim prikazom, ažurira se i sadržaj kliznog izbornika. Ikona uz podstanicu, koja je na početku zelena i sadrži simbol plusa, nakon aktivacije praćenja mijenja se u crvenu ikonu s minus znakom, čime se korisniku intuitivno daje do znanja da, ako to želi, u bilo kojem trenutku može prekinuti prikaz vozila za tu stanicu.

Važno je istaknuti da aplikacija omogućuje praćenje isključivo jedne podstanice u svakom trenutku. Ako korisnik aktivira prikaz vozila za neku drugu podstanicu, prethodno se praćenje automatski prekida, a novi se podatci učitavaju u skladu s novim odabirom. Ovakva arhitektonska odluka omogućuje održavanje jednostavnog i preglednog korisničkog sučelja te minimizira mogućnost zabune u interpretaciji prikazanih podataka.

Dodatno, kako bi se korisniku olakšalo praćenje dolaska vozila, redoslijed ruta u kliznom izborniku prilagođava se dinamici prometa. Prije aktivacije prikaza vozila, rute su prikazane prema rastućem broju linije, no nakon što je prikaz uključen, redoslijed se automatski mijenja tako da se najprije prikazuju one rute čije će vozilo prvo stići na stanicu. Ova funkcionalnost temelji se na procjeni vremena dolaska vozila, pri čemu se koriste povijesni podatci pohranjeni u zasebnoj tablici *TripHistory*. Mehanizam predikcije bit će detaljnije opisan u sljedećem podpoglavlju.

Kako bi se dodatno povećala povezanost vizualnog i tekstualnog prikaza, uz svaku rutu u kliznom izborniku prikazuje se mala kružna ikona u boji koja odgovara boji ikone vozila na karti. Time se postiže jasna vizualna poveznica između prikaza ruta u izborniku i njihovih odgovarajućih oznaka na karti.



Slika 38. Prikaz najbližih vozila za odabranu postaju

5.1.3 Predikcija vremena dolaska vozila na stanicu

Precizno predviđanje vremena dolaska vozila na stajalište jedno je od najvažnijih očekivanja korisnika u aplikaciji koja se oslanja na podatke u stvarnom vremenu. Kako bi se ostvario taj cilj, unutar sustava je razvijen vlastiti model predikcije dolazaka, temeljen na analizi povijesnih podataka prikupljenih u tablici *TripHistory*.

Nakon što su prethodno određena najbliža vozila za svaku od ruta koje prometuju kroz odabranu podstanicu, unutar istog zahtjeva prema API-u (*/get_closest_vehicles*) poziva se dodatna funkcija pod nazivom *predicted_arrival_times*, prikazana na slici 39. Njoj se prosljeđuju podatci svih prethodno određenih najbližih vozila zajedno s identifikatorom stanice prema kojoj se radi predikcija.

Za potrebe tog izračuna koriste se dva ključna parametra. Prvi je trenutni segment rute na kojem se vozilo trenutačno nalazi, a koji je pohranjen u atribut *shapeSegment_currentPosition* unutar tablice *TripUpdates*. Drugi parametar odnosi se na segment rute na kojem se nalazi odabrana stanica, odnosno pozicija stanice u okviru konkretne putanje kojom se vozilo kreće. Budući da svako vozilo prati svoju vlastitu geografsku putanju definiranu atributom *shape_id*, ista stanica ne mora biti smještena na istom segmentu kod različitih vozila. Dvije vožnje koje prometuju kroz isto stajalište, ali pripadaju različitim rutama ili smjerovima, mogu imati različite trase, pa tako i različite segmente na kojima se ista stanica nalazi. Upravo zbog toga segment stanice treba promatrati u kontekstu konkretnog vozila za koje se radi predikcija. Kako bi se takav podatak dohvatilo, koristi se privremena memorijska struktura specifična za svako pojedino vozilo. Budući da je svaka vožnja jednoznačno određena identifikatorom *trip_id*, kombinacijom tog identifikatora i *stop_id* stanice moguće je iz memorije izdvojiti informaciju o tome na kojem se segmentu putanje ta stanica nalazi.

Kada su poznate vrijednosti oba segmenta, funkcija pristupa tablici *TripHistory*, u kojoj su trajno pohranjeni povijesni zapisi kretanja vozila. Prvi korak u obradi uključuje filtriranje zapisa koji imaju isti *route_id* i *routeDirection* kao i promatrano vozilo, čime se osigurava da se promatraju isključivo vožnje koje su se u prošlosti kretale jednakom rutom i u istom smjeru. Među tako filtriranim zapisima traže se vozila koja unutar vlastite povijesti posjeduju dva zapisa koja ispunjavaju točno određeni uvjet. Prvi zapis mora sadržavati vrijednost atributa *shapeSegment_currentPosition* jednaku trenutačnom segmentu vozila za koje se računa predikcija, dok drugi zapis mora imati vrijednost segmenta koja odgovara onoj stanice prema kojoj se predviđa dolazak. Oba zapisa moraju pripadati istom vozilu, odnosno imati isti *trip_id*, čime se osigurava da se izračun odnosi na jedno konkretno povijesno kretanje, a ne na proizvoljne i nepovezane vožnje.

Za svaki takav pronađeni par zapisa računa se razlika između njihovih vremenskih oznaka pohranjenih u atributu *lastUpdated*. Dobivena razlika predstavlja stvarno vrijeme koje je tom vozilu u prošlosti bilo potrebno da prijeđe istu dionicu koju trenutno vozilo tek treba prijeći. Međutim, u stvarnim prometnim uvjetima često dolazi do oscilacija u kretanju vozila, što može biti uzrokovano različitim čimbenicima kao što su zastoji u prometu, tehničke pogreške u sustavu ili neuobičajene prometne okolnosti. Kako bi se smanjio utjecaj takvih iznimki i osigurala stabilnost predikcije, dodatno se provodi obrada podataka. U tom se postupku primjenjuje filtriranje na temelju percentila, pri čemu se u analizu uključuju isključivo one

vrijednosti trajanja koje se nalaze između 20. i 80. percentila cjelokupnog skupa rezultata. Na taj se način iz predviđanja uklanjaju ekstremno niske i ekstremno visoke vrijednosti, koje bi zbog svog odstupanja mogle negativno utjecati na pouzdanost konačnog rezultata

Nakon toga izračunava se srednja vrijednost preostalih vremenskih razlika, koja se interpretira kao procijenjeno vrijeme dolaska vozila na ciljanu stanicu. Ta vrijednost, izražena u sekundama, integrira se u strukturirani JSON objekt koji se vraća aplikaciji zajedno s ostalim podatcima o najbližim vozilima. Naposljetku, ova se informacija koristi unutar kliznog izbornika za prikaz predviđenih vremena dolaska, kao i za sortiranje ruta prema očekivanom redoslijedu dolazaka.

Budući da se zahtjev prema API-u šalje svakih pet sekundi sve dok je neka podstanica aktivno odabrana, funkcija za predikciju vremena također se poziva jednakim ritmom. Na taj se način osigurava kontinuirano ažuriranje prikazanih informacija i održava visoka razina preciznosti, zahvaljujući dinamičkom povezivanju trenutačne pozicije vozila s povijesnim obrascima kretanja prethodnih vožnji na istoj liniji i u istom smjeru.

```python
def predicted_arrival_times(closest_vehicles, child_stop_id):
    try:
        with pyodbc.connect(connection_string) as conn:
            with conn.cursor() as cursor:
                for route_id, vehicle_data in closest_vehicles.items():
                    trip_id = vehicle_data.get("tripId")
                    route_direction = vehicle_data.get("routeDirection")
                    route_id_int = int(vehicle_data.get("routeId", 0))
                    current_segment = vehicle_data.get("shapeSegment_currentPosition")
                    stop_segment = trip_cache.get(trip_id, {}).get("stop_segments", {}).get(child_stop_id)
                    if not trip_id or not route_direction:
                        vehicle_data["predicted_arrival_seconds"] = None
                        continue
                    if current_segment in [None, 0] or stop_segment in [None, 0]:
                        vehicle_data["predicted_arrival_seconds"] = None
                        continue
                    if current_segment >= stop_segment:
                        vehicle_data["predicted_arrival_seconds"] = None
                        continue
                    cursor.execute("""
                    WITH Differences AS (
                        SELECT
                            DATEDIFF(SECOND, seg_current.lastUpdated, seg_stop.lastUpdated) AS arrival_diff
                        FROM TripHistory AS seg_current
                        JOIN TripHistory AS seg_stop
                        ON seg_current.tripId = seg_stop.tripId
                        WHERE seg_current.routeId = ?
                        AND seg_current.routeDirection = ?
                        AND seg_current.shapeSegment_currentPosition = ?
                        AND seg_stop.shapeSegment_currentPosition = ?
                        AND seg_current.lastUpdated < seg_stop.lastUpdated
                    ),
                    Ranked AS (
                        SELECT
                            arrival_diff,
                            ROW_NUMBER() OVER (ORDER BY arrival_diff) AS rn,
                            COUNT(*) OVER () AS total_count
                        FROM Differences
                    )
                    SELECT AVG(arrival_diff) AS avg_arrival_seconds
                    FROM Ranked
                    WHERE rn BETWEEN total_count * 0.2 AND total_count * 0.8;
                    """, (route_id_int, route_direction, current_segment, stop_segment))
                    row = cursor.fetchone()
                    if row and row[0] is not None:
                        vehicle_data["predicted_arrival_seconds"] = int(row[0])
                    else:
                        vehicle_data["predicted_arrival_seconds"] = None
    except Exception as e:
        print(f"[Predikcija] Greška: {e}")
```

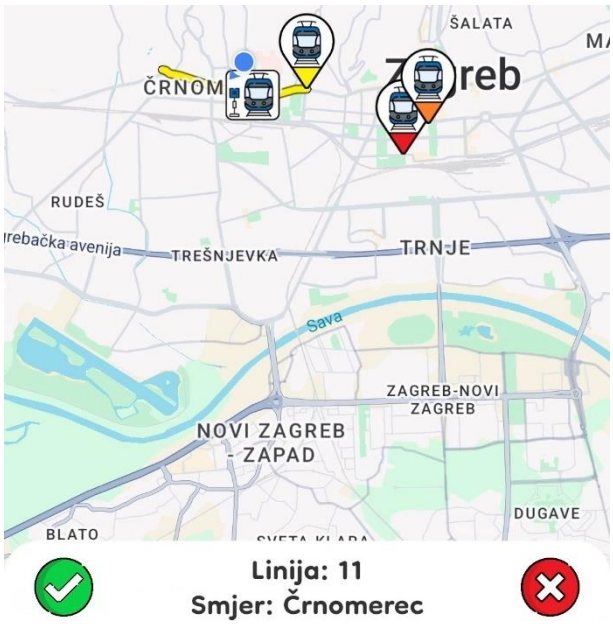
Slika 39. Funkcija *predicted_arrival_times*

5.1.4 Odabir željenog vozila

Dodatna funkcionalnost unutar početnog zaslona aplikacije odnosi se na mogućnost individualnog praćenja vozila nakon što korisnik odabere njegovu ikonu na karti. U trenutku odabira aplikacija reagira promjenom vizualnog prikaza, kombinirajući prilagodbu karte s

aktivacijom dodatnih informacija u korisničkom sučelju, što je vidljivo na slici 40. Karta se automatski centrira na poziciju odabranog vozila, a razina uvećanja prilagođava se tako da osigura optimalnu preglednost. Istovremeno, sve dotad prikazane sivo obojene trajektorije ostalih vozila uklanjaju se iz prikaza kako bi se korisnički fokus usmjerio isključivo na vozilo koje je odabrao. Umjesto uklonjenih trajektorija, na karti se prikazuje cijela preostala dionica puta odabranog vozila, počevši od njegove aktualne pozicije pa sve do završne stanice, pri čemu je trasa prikazana u boji koja odgovara boji ikone vozila. Ovim prikazom korisniku se jasno vizualizira cijela preostala dionica puta za odabrano vozilo.

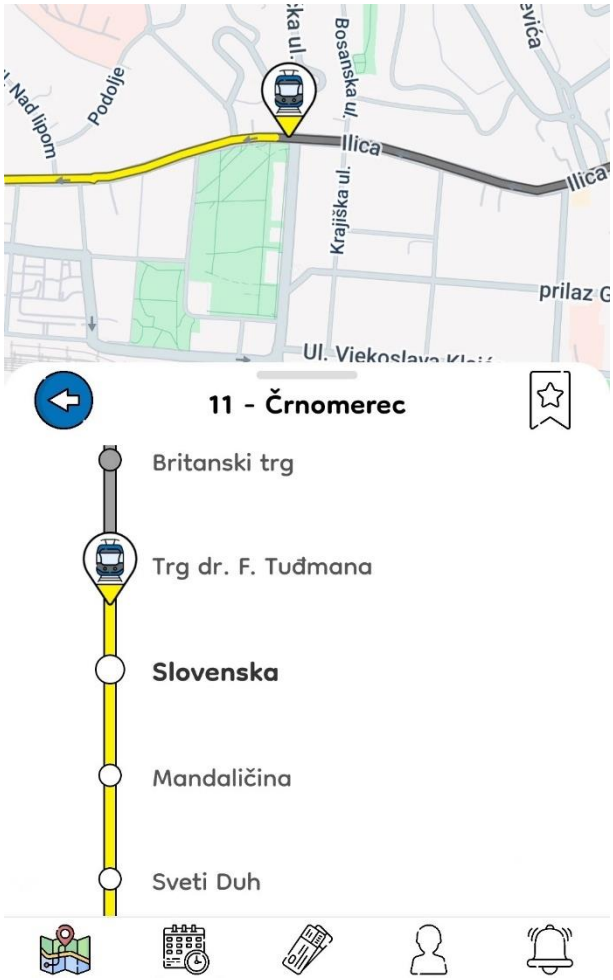
Paralelno s promjenama na karti, dolazi i do ažuriranja kliznog izbornika u donjem dijelu zaslona. Na sličan način kao pri odabiru stanice, izbornik se smanjuje i prikazuje osnovne informacije o vozilu, uključujući broj linije i naziv krajnje stanice, odnosno smjer kretanja. Unutar ovog prikaza korisniku su ponuđene dvije ikone, pri čemu se zelena ikona nalazi s lijeve strane i služi za potvrdu odabrane stanice, dok se crvena ikona nalazi s desne strane i omogućuje poništavanje odabira te time povratak na prikaz odabrane podstanice.



Slika 40. Izbornik za potvrdu odabranog vozila

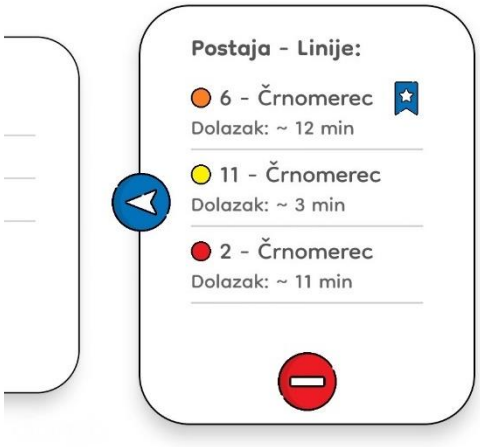
Potvrdom odabira vozila unutar kliznog izbornika, izbornik se proširuje i njegov se sadržaj ažurira, što je prikazano na slici 41. Istovremeno se na karti mijenja prikaz tako da se uklanjaju svi prethodno prikazani elementi, uključujući ikone ostalih vozila i prethodno odabrane stanice. Umjesto njih, na karti se prikazuje proširena trajektorija odabranog vozila duž cijele njegove

rute. Dio puta koji je vozilo već prošlo prikazan je sivom bojom, dok je ostatak trase koji vozilo tek treba prijeći istaknut bojom koja odgovara boji njegove ikone. Na taj se način osigurava maksimalna preglednost i potpuni fokus na trenutno odabrano vozilo i njegovu rutu.



Slika 41. Prikaz zaslona pri odabiru željenog vozila

U ažuriranom prikazu kliznog izbornika, gornji dio sadrži zaglavlje koje obuhvaća ikonu za povratak na prikaz podstanica, središnji prikaz broja linije i njezina smjera te ikonu za dodavanje linije u popis omiljenih. Klikom na ikonu favorita odabrana linija pohranjuje se u lokalnu memoriju uređaja, a vizualna promjena boje ikone korisniku signalizira da je linija uspješno označena kao favorit. U svakom sljedećem prikazu podstanice, ako ta linija prometuje kroz nju, automatski se prikazuje na vrhu popisa linija, neovisno o predviđenom vremenu njezina dolaska, što je vidljivo na slici 42. U slučaju da postoji više favoriziranih linija, one se međusobno dodatno sortiraju prema očekivanom vremenu dolaska, dok se tek nakon njih prikazuju linije koje nisu označene kao omiljene.



Slika 42. Korištenje opcije favoriziranja linija

Povratno, središnji dio prikaza odabranog vozila unutar kliznog izbornika donosi intuitivnu i vizualno jasnu reprezentaciju cijele trase kretanja tog vozila. Prikazana je vertikalna traka koja simbolički prikazuje trasu od početne do završne stanice, a duž nje su kružno označena sva stajališta na ruti. Svaki kružni indikator odgovara jednoj stanici, a s njegove desne strane ispisan je naziv te stanice.

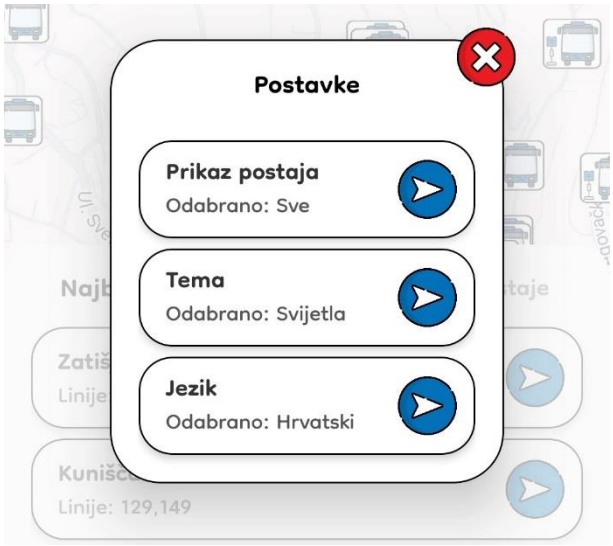
Trenutačna pozicija vozila označena je ikonom vozila koja se smješta uz odgovarajući segment trake. Trasa kojom je vozilo već prošlo prikazana je sivom bojom, dok je ostatak trase koji vozilo tek treba prijeći istaknut u boji koja odgovara boji njegove ikone. Na taj način korisnik može lako razlikovati već prijeđeni dio puta od onoga koji slijedi, pri čemu mu natpisi uz stanice omogućuju lakšu orijentaciju i razumijevanje trenutne pozicije vozila na ruti. Pozicija vozila na traci, kao i sama traka, ažuriraju se u stvarnom vremenu, sinkronizirano s njegovim stvarnim kretanjem na karti.

5.1.5 Dodatne funkcionalnosti i globalne postavke

Iako je početni zaslon zamišljen kao glavno mjesto interakcije između korisnika i informacija o javnom prijevozu, tijekom razvoja aplikacije poseban je naglasak stavljen na jasnoću, intuitivnost i funkcionalnu cjelovitost. Umjesto da se sučelje optereti velikim brojem opcija, fokus je bio na kvaliteti implementiranih značajki i njihovoj stvarnoj korisnosti u svakodnevnoj upotrebi. Svaka funkcija unutar ovog zaslona pažljivo je osmišljena kako bi

korisniku omogućila brz pristup relevantnim informacijama, jednostavno donošenje odluka i pregledno praćenje stvarnog stanja u prometnoj mreži.

Osim osnovnih funkcionalnosti povezanih s prikazom stanica i vozila, početni zaslon izravno je povezan i s globalnim postavkama aplikacije. Tim se postavkama može pristupiti s bilo kojeg zaslona putem fiksno pozicionirane ikone smještene u gornjem desnom kutu korisničkog sučelja. Klikom na tu ikonu otvara se dijaloški prozor, prikazan na slici 43, koji sadržava tri osnovne postavke koje zajedno predstavljaju suvremeni pristup personalizaciji korisničkog iskustva u mobilnim aplikacijama.



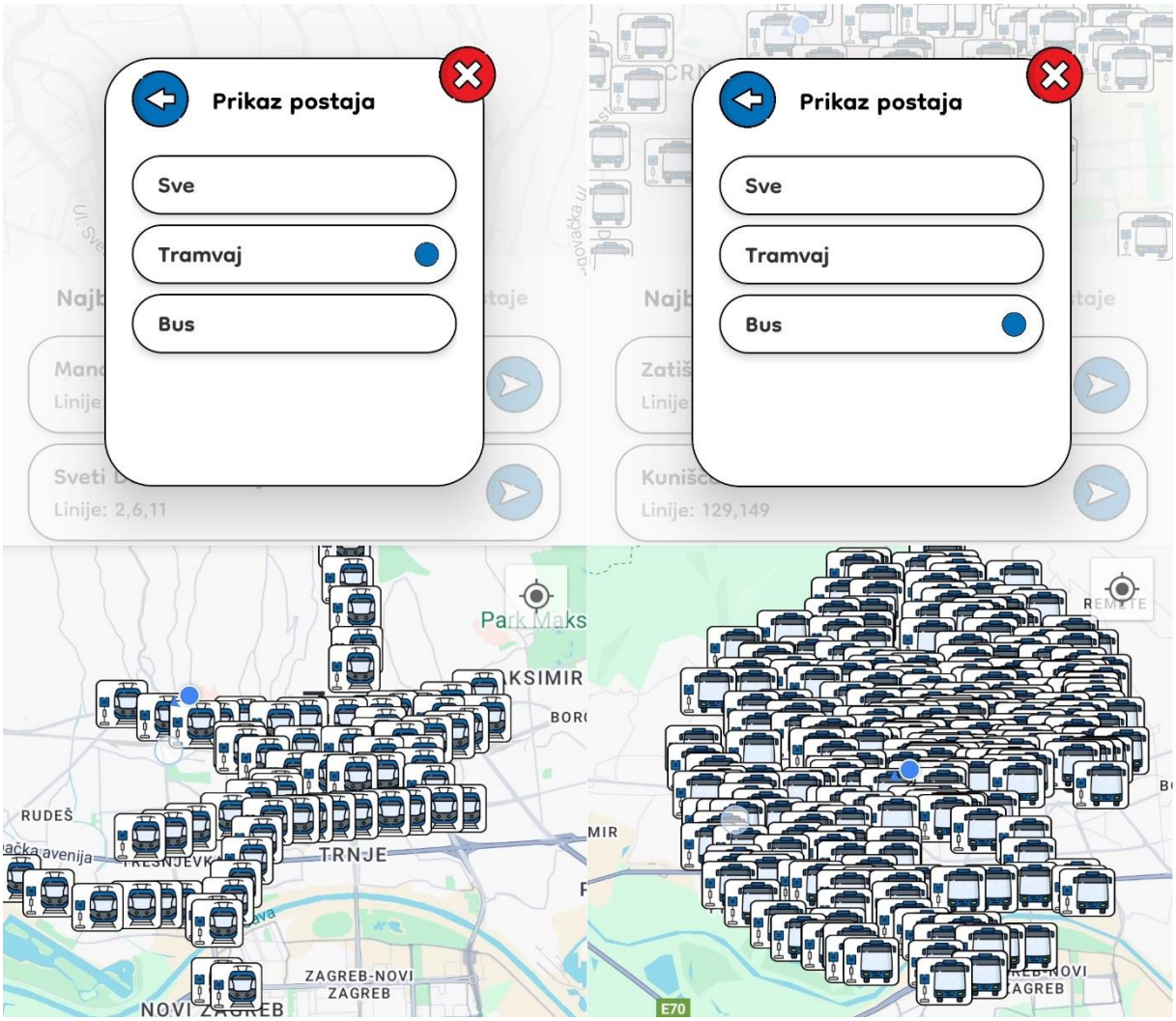
Slika 43. Otvoreni dijaloški prozor „Postavke”

Prva od dostupnih postavki odnosi se na odabir teme prikaza. Korisnik može birati između svijetlog i tamnog načina rada, čime se sučelje može prilagoditi osobnim preferencijama i različitim uvjetima osvjetljenja. Osim estetske vrijednosti, tamni način rada pridonosi manjoj potrošnji energije na uređajima s OLED zaslonima, a istodobno omogućuje ugodnije korištenje u noćnim uvjetima ili u prostorima sa slabijim osvjetljenjem. Takva prilagodljivost vizualnog identiteta aplikacije pridonosi većem stupnju pristupačnosti, osobito korisnicima s osjetljivijim vidom.

Druga se postavka odnosi na odabir jezika korisničkog sučelja. Aplikacija podržava prikaz sadržaja na hrvatskom i engleskom jeziku, čime se u znatnoj mjeri proširuje njezina dostupnost i funkcionalna primjenjivost. Ova jezična prilagodba omogućuje učinkovito korištenje

aplikacije ne samo domaćim korisnicima, već i stranim posjetiteljima, osobito turistima, kojima ona predstavlja relevantan i pouzdan alat za orijentaciju u mreži javnog gradskog prijevoza.

Treća opcija izravno utječe na prikaz prometne infrastrukture na karti, što je vidljivo na slici 44. Korisnik može odabrati želi li da se prikazuju sva stajališta, samo tramvajska ili samo autobusna. Ova se postavka ne odražava isključivo na vizualne oznake na samoj karti, već i na sve povezane prikaze unutar aplikacije, uključujući prikaz najbližih i omiljenih postaja. Na taj se način prilagodba prikaza odražava na svim razinama korištenja aplikacije, omogućujući korisniku jednostavno snalaženje i preglednost u skladu s njegovim stvarnim interesima i navikama u svakodnevnom korištenju javnog prijevoza.



Slika 44. Prikaz filtriranja tramvajskih i autobusnih stajališta na karti.

Način na koji su globalne postavke integrirane u aplikaciju pokazuje pažljivo osmišljenu arhitekturu, u kojoj se postavke ne odnose samo na izgled sučelja, nego i na sadržaj koji se prikazuje te način na koji korisnik s tim sadržajem komunicira. Time se dodatno potvrđuje orijentiranost aplikacije prema krajnjem korisniku, s ciljem da se složenost sustava javnog gradskog prijevoza prikaže na jednostavan, pristupačan i interaktivan način, prilagođen potrebama pojedinca.

5.2 Zaslon „Obavijesti”

Drugi po važnosti zaslon unutar aplikacije jest zaslon „Obavijesti”, čija je osnovna funkcija pružiti korisnicima ažurne informacije o izmjenama u prometu te novostima vezanim uz rad i organizaciju ZET-a. U kontekstu javnog gradskog prijevoza, transparentno i pravovremeno obavještavanje korisnika o privremenim obustavama linija, preusmjeravanjima, tehničkim poteškoćama ili novim inicijativama predstavlja važan aspekt kvalitete usluge. Upravo zato ovaj zaslon integrira izravni pristup službenim informacijama koje ZET redovito objavljuje na svojim web-stranicama.

5.2.1 Dohvat i obrada RSS podataka

Za potrebe implementacije zaslona „Obavijesti”, unutar aplikacije razvijen je sustav za dohvat i prikaz aktualnih informacija koje se odnose na novosti i privremene izmjene u javnom prijevozu. Kao temelj za realizaciju ove funkcionalnosti koristi se tehnologija RSS (engl. *Really Simple Syndication*), (15). Riječ je o standardiziranom formatu za distribuciju ažurnih sadržaja temeljenom na XML strukturi, koji omogućuje da aplikacije i web-servisi periodički preuzimaju najnovije informacije s odabranog izvora. Takav pristup omogućuje automatizirano osvježavanje sadržaja bez potrebe za ručnim unosom ili primjenom složenijih metoda, poput automatskog analiziranja HTML strukture web stranica radi izdvajanja relevantnih informacija (engl. *Web scraping*).

ZET nudi dvije javno dostupne RSS adrese koje služe kao izvor podataka. Prva se odnosi na aktualne novosti vezane uz organizaciju i rad poduzeća (https://www.zet.hr/rss_novosti.aspx), dok druga sadržava informacije o privremenim izmjenama u prometu (https://www.zet.hr/rss_izmjene.aspx).

Pri svakom pokretanju aplikacije automatski se šalju dva zahtjeva prema API-u, i to za dohvat aktualnih novosti (*/get_zet_news*) te prometnih izmjena (*/get_zet_changes*). Nakon primitka zahtjeva, prikazanih na slici 45, API pristupa pripadajućim RSS izvorima i dohvaća sadržaj u obliku XML dokumenta. Dobiveni sadržaj dekodira se pomoću biblioteke *xml.etree.ElementTree*, kojom se omogućuje strukturirana obrada XML elemenata. Iz svakog zapisa unutar XML-a izdvajaju se četiri ključne informacije koje obuhvaćaju naslov obavijesti, opis, poveznicu na puni sadržaj te datum objave. Tako obrađeni podatci prevode se u strukturirani JSON format i šalju natrag aplikaciji, spremni za prikaz unutar korisničkog sučelja.

Prednost ovakvog pristupa očituje se u njegovoj jednostavnosti i učinkovitosti. Budući da aplikacija ne pohranjuje podatke o novostima i izmjenama u vlastitoj bazi, izbjegnuta je potreba za održavanjem dodatnih tablica, dok se istovremeno osigurava pristup uvijek ažurnim informacijama izravno s izvora. Zahvaljujući strukturiranosti RSS formata, obrada podataka odvija se brzo i bez značajnog opterećenja na strani API-a. Na taj je način funkcionalnost prikaza novosti i izmjena ostvarena bez narušavanja performansi sustava te uz zadržavanje minimalne složenosti implementacije.

```python
@app.route("/get_zet_changes", methods=["GET"])
def get_zet_changes():
    try:
        rss_url = "https://www.zet.hr/rss_promet.aspx"
        response = requests.get(rss_url)
        response.raise_for_status()

        root = ET.fromstring(response.content)
        items = root.find('channel').findall('item')
        changes_list = []

        for item in items:
            changes = {
                "title": item.findtext("title"),
                "description": item.findtext("description"),
                "link": item.findtext("link"),
                "pubDate": item.findtext("pubDate")
            }
            changes_list.append(changes)

        return jsonify(changes_list)

    except Exception as e:
        print(f"Greška kod dohvaćanja RSS-a: {e}")
        return jsonify([]), 500

@app.route("/get_zet_news", methods=["GET"])
def get_zet_news():
    try:
        rss_url = "https://www.zet.hr/rss_novosti.aspx"
        response = requests.get(rss_url)
        response.raise_for_status()

        root = ET.fromstring(response.content)
        items = root.find('channel').findall('item')
        news_list = []

        for item in items:
            news = {
                "title": item.findtext("title"),
                "description": item.findtext("description"),
                "link": item.findtext("link"),
                "pubDate": item.findtext("pubDate")
            }
            news_list.append(news)

        return jsonify(news_list)

    except Exception as e:
        print(f"Greška kod dohvaćanja RSS-a: {e}")
        return jsonify([]), 500
```

Slika 45. Funkcije *get_zet_news* i *get_zet_changes*

5.2.2 Prikaz sadržaja

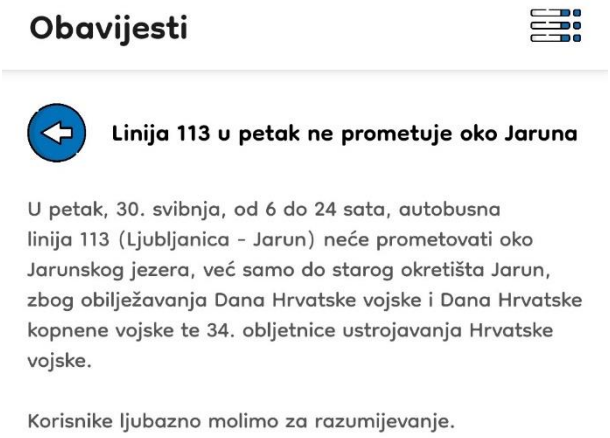
Kada se korisnik nalazi na zaslonu „Obavijesti”, vidljivom na slici 46, u gornjem dijelu prikazuje se interaktivni prekidač dizajnom identičan onome korištenom u početnom zaslonu, unutar kliznog izbornika. Pomoću ovog prekidača korisnik može odabrati želi li pregledavati izmjene u prometu ili novosti, pri čemu se aktivna kategorija vizualno ističe u odnosu na neaktivnu.

Svaki zapis dohvaćen iz RSS izvora prikazuje se u obliku kartice koja sadrži osnovne informacije. U gornjem dijelu kartice istaknut je naslov obavijesti, dok se neposredno ispod njega nalazi datum objave, čime se korisniku omogućuje procjena vremenske relevantnosti prikazanih informacija. Na samom rubu kartice, u njezinu donjem desnom kutu, nalazi se ikona u obliku zelenog simbola plusa, koja vizualno upućuje na mogućnost otvaranja proširenog prikaza s dodatnim informacijama.

Slika 46. Prikaz zaslona „Obavijesti”

Klikom na karticu otvara se prošireni prikaz obavijesti, vidljiv na slici 47, koji prikazuje puni opisni tekst preuzet iz polja *description* unutar RSS zapisa. U gornjem dijelu proširenog prikaza nalazi se zaglavlje s ikonom za povratak na prethodni prikaz te središnje postavljenim naslovom obavijesti.

Ovaj prikaz koncipiran je tako da zadrži preglednost čak i u slučaju duljih tekstova, čime se korisniku omogućuje brz i jednostavan pristup informacijama bez potrebe za napuštanjem aplikacije ili otvaranjem vanjskih poveznica.



Slika 47. Prošireni prikaz obavijesti

5.3 Zaslon „Vozni red”

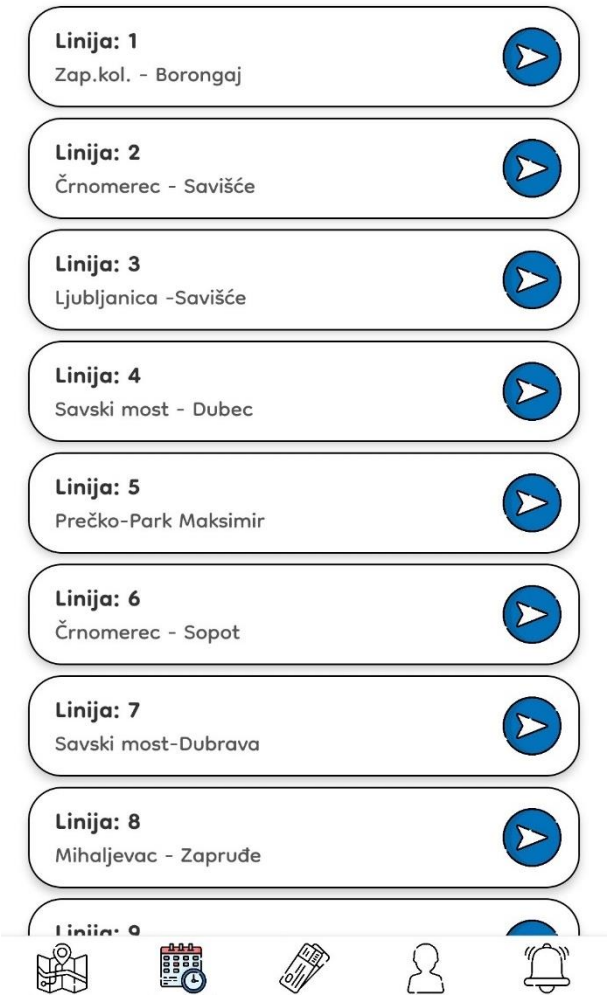
Treći temeljni zaslon aplikacije, nazvan „Vozni red”, omogućuje pregled službenih voznog redova za sve aktivne linije javnog gradskog prijevoza u Zagrebu. Iako su procjene dolaska vozila u stvarnom vremenu iznimno korisne u svakodnevnom prometnom kontekstu, i dalje postoji značajan broj korisnika, osobito onih koji putovanje započinju na početnim stajalištima linija, kojima je ključna informacija o službenom rasporedu polazaka. To se ponajprije odnosi na autobusne linije s fiksnim vremenskim polascima, čiji su rasporedi na početnim postajama često dosljedni i pouzdani.

5.3.1 Dohvat i prikaz svih linija

Po pokretanju aplikacije šalje se zahtjev prema API-u (/*getRoutes*) radi dohvaćanja svih dostupnih linija iz baze podataka, točnije iz tablice *routes*. Iz navedene tablice dohvaćaju se

atributi *route_id*, kao jedinstveni identifikator linije izravno jednak broju linije, te *route_long_name*, koji sadržava opis rute u formatu „polazište – odredište”. Nakon što API obradi zahtjev, podatke strukturira u JSON formatu i prosljeđuje ih aplikaciji.

Po primitku podataka, aplikacija generira listu svih dostupnih linija, pri čemu se za svaku liniju prikazuje broj linije te smjer kretanja, odnosno početna i završna stanica. Svaka linija prikazana je u obliku interaktivne kartice koja omogućuje daljnju navigaciju prema pripadajućem voznom redu, što je vidljivo na slici 48.



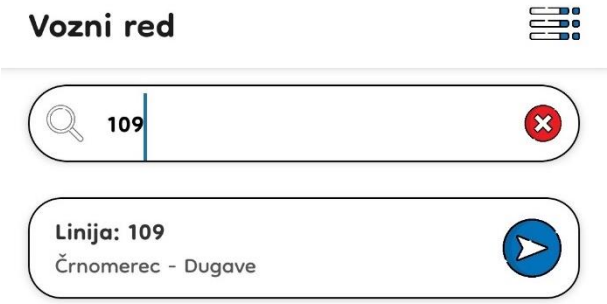
Slika 48. Prikaz kartica unutar zaslona „Vozni red”

5.3.2 Filtriranje linija

S obzirom na velik broj linija javnog prijevoza u Zagrebu, unutar zaslona „Vozni red” implementirana je funkcionalnost pretraživanja koja korisnicima omogućuje brzo i učinkovito

filtriranje rezultata. Na vrhu zaslona nalazi se tražilica (engl. *Search bar*), prikazana na slici 49, koja omogućuje unos ključnih pojmova prema kojima se prikazane linije filtriraju. Korisnik može unijeti broj linije, naziv polazišnog ili završnog stajališta, a aplikacija će u stvarnom vremenu prikazati isključivo one linije čiji atributi odgovaraju unesenim pojmovima.

Pretraživanje se izvodi lokalno, što znači da se ne šalju dodatni zahtjevi prema API-u. Umjesto toga, aplikacija koristi već dohvaćene podatke u JSON formatu te nad njima primjenjuje funkciju *filter*, kojom se iterativno provjerava sadrži li naziv rute unesene ključne riječi. Kako bi se omogućilo pretraživanje neovisno o veličini slova, sve se tekstualne vrijednosti prethodno pretvaraju u mala slova pomoću metode *toLowerCase*. Ovakav pristup omogućuje trenutačno osvježavanje prikaza rezultata bez dodatnog opterećenja API-a te osigurava brz odaziv korisničkog sučelja.



Slika 49. Filtriranje linija korištenjem tražilice

5.3.3 Dohvat i prikaz PDF-a voznog reda

Ključna funkcionalnost zaslona „Vozni red" odnosi se na mogućnost prikaza službenih PDF dokumenata s voznim redovima za sve aktivne linije javnog gradskog prijevoza. Odabirom željene linije unutar sučelja aplikacije pokreće se zahtjev prema API-u (*proxy-pdf*), s ciljem dohvaćanja PDF dokumenta sa službenih mrežnih stranica ZET-a. Dokumenti su organizirani prema predvidivom obrascu URL adresa, pri čemu se struktura URL-a mijenja ovisno o broju linije. Na temelju vrijednosti atributa odabrane linije (*route_id*) konstruira se odgovarajući URL, kako je prikazano na slici 50, koji upućuje na pripadajuću datoteku i omogućuje njezin prikaz unutar aplikacije.

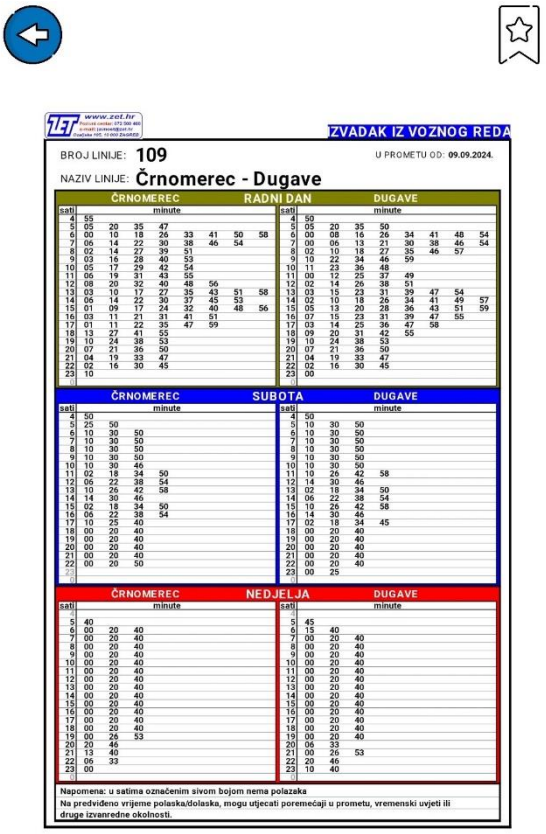
```
const getZetPdfUrl = (route_id: number): string => {
  if (route_id >= 1 && route_id <= 34) {
    return `https://www.zet.hr/UserDocsImages/Tramvajske%20linije%20-%20vozni%20red/${route_id}.pdf`;
  } else {
    return `https://www.zet.hr/UserDocsImages/Autobusne%20linije%20-%20rasporedi/${route_id}.pdf`;
  }
};
```

Slika 50. Konstruiranje URL-a koji vodi do odgovarajuće datoteke

Na strani API-a implementiran je mehanizam za dohvat i privremenu pohranu PDF sadržaja, pri čemu API preuzima dokument s generirane URL adrese te ga sprema u lokalnu mapu. Prikaz PDF dokumenata u mobilnoj aplikaciji ne odvija se putem izravnog učitavanja datoteke, već posredstvom prilagođenog pregledničkog mehanizma integriranog unutar samog API-a. Ovaj mehanizam temelji se na rješenju otvorenog koda *PDF.js*, razvijenom u okviru Mozilla zajednice, koji omogućuje prikaz PDF sadržaja unutar HTML stranice, bez potrebe za dodatnim softverskim alatima na korisničkom uređaju, (16).

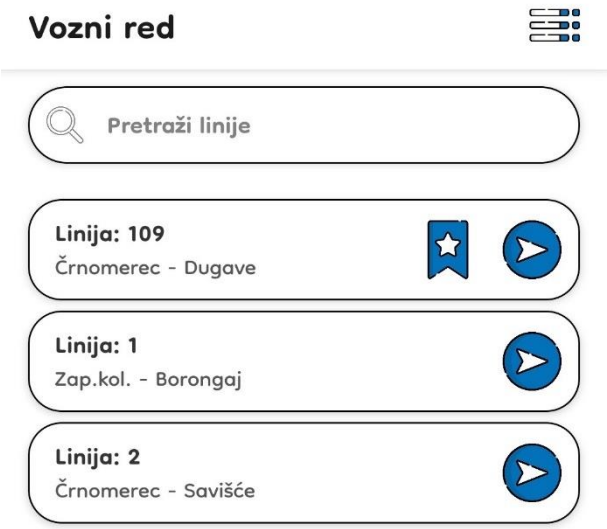
PDF.js komponenta i HTML sučelje *viewer.html*, kao osnovni elementi preglednika, smješteni su lokalno u strukturi API-a. Prilikom korisničkog zahtjeva za prikazom dokumenta, aplikaciji se ne vraća preuzeta PDF datoteka, već se otvara ugrađena komponenta *WebView*, koja unutar aplikacije funkcionira kao interni preglednik. *WebView* tada učitava stranicu *viewer.html* s API-a, pri čemu se putem URL adrese prosljeđuje putanja do prethodno pohranjenog PDF dokumenta.

Nakon što *WebView* komponenta učita stranicu *viewer.html*, korisniku se unutar aplikacije otvara zaseban prikaz koji služi kao preglednik PDF voznog reda, što je vidljivo na slici 51. Upravo se tu ostvaruje prikaz same datoteke uz dodatne interaktivne elemente korisničkog sučelja, koji upotpunjuju funkcionalnost i poboljšavaju pristupačnost prikazanih informacija.



Slika 51. Prikaz PDF preglednika

U gornjem dijelu prikaza nalazi se zaglavlje s dvjema ikonama. Lijeva ikona služi za povratak na prethodni prikaz liste linija, dok desna omogućuje korisniku da označi trenutno prikazanu liniju kao omiljenu. Ako korisnik označi liniju kao omiljenu, ona se prilikom povratka na glavni prikaz zaslona „Vozni red" prikazuje na vrhu popisa, što je vidljivo na slici 52. U slučaju više favoriziranih linija, one se prikazuju grupirano te su dodatno sortirane po rastućem identifikatoru *route_id*. Time se omogućuje brži pristup voznim redovima linija koje korisnik najčešće koristi, bez potrebe za ponovnim pretraživanjem.



Slika 52. Korištenje opcije favoriziranja linije voznog reda

Ugrađeni PDF preglednik podržava sve standardne funkcionalnosti koje se očekuju od prikaza dokumenata. Omogućeni su slobodno pomicanje po dokumentu te povećavanje i smanjivanje prikaza. Sučelje preglednika pritom je jednostavno i nenametljivo, usmjereno isključivo na sadržaj voznog reda i osnovne kontrole. Dodavanjem ove funkcionalnosti osigurano je integrirano i neprekinuto korisničko iskustvo, pri čemu se dokument prikazuje unutar aplikacije bez potrebe za napuštanjem sučelja ili korištenjem vanjskih preglednika i dodatnih aplikacija.

5.4 Zasloni „Profil” i „Karte”

Zasloni „Profil” i „Karte” integrirani su u korisničko sučelje mobilne aplikacije kao konceptne cjeline namijenjene demonstraciji mogućih smjerova razvoja digitalnih rješenja za javni gradski prijevoz. Njihova je implementacija usmjerena isključivo na ilustraciju potencijalnih funkcionalnosti, bez stvarne povezanosti sa sustavom naplate ili obradom osobnih podataka korisnika.

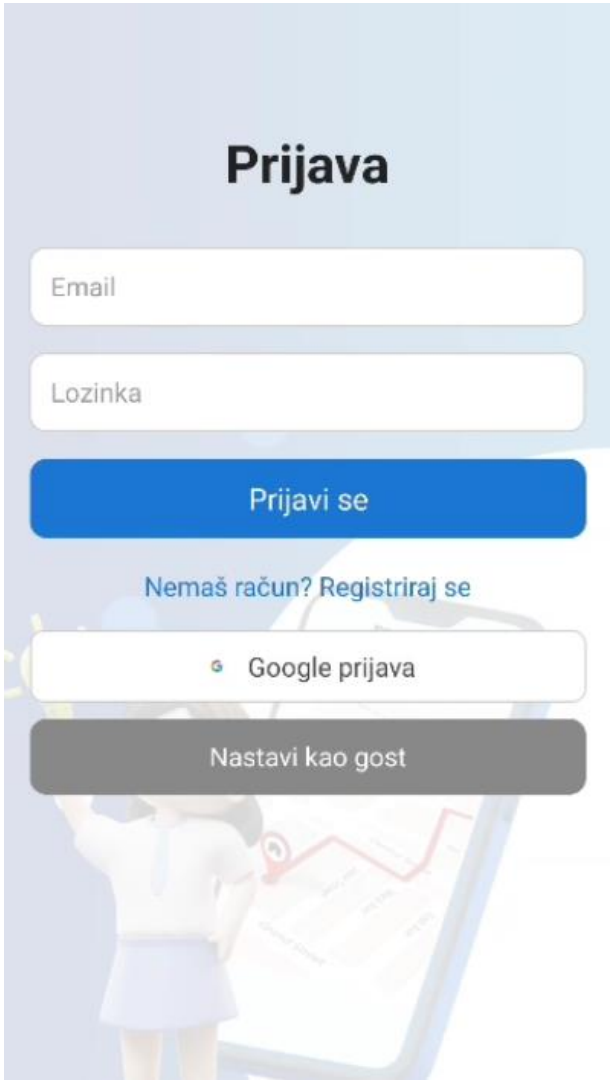
Razvoj ovih zaslona temelji se na ideji prikaza budućih mogućnosti poput prijave korisnika te digitalne kupnje, prikaza i verifikacije karata unutar mobilne aplikacije, uz naglasak na jednostavnost korištenja i jasnoću vizualne prezentacije. Posebno se ističe potencijal njihove primjene u kontekstu suradnje s davateljem usluge poput ZET-a, gdje bi ovakva rješenja mogla

poslužiti kao temelj za digitalnu modernizaciju postojećeg sustava. Funkcionalnosti svakog od zaslona detaljnije su predstavljene u nastavku.

5.4.1 Zaslon „Profil”

U okviru mobilne aplikacije zamišljen je koncept za upravljanje korisničkim identitetom, koji se u trenutačnoj verziji realizira kroz zaslon „Profil”. Iako ne obavlja stvarnu autentifikaciju ni pristup vanjskim sustavima, njegova je svrha ilustrirati mogući smjer razvoja prema personaliziranom korisničkom iskustvu. Njegova se funkcija prvenstveno očituje u omogućavanju prijave i registracije korisnika. U tom bi kontekstu korisnički profil predstavljao središnje mjesto za upravljanje osobnim podacima, preferencijama i povezivanjem s vanjskim sustavima. Time bi se u punoj implementaciji stvorio preduvjet za personalizirani prikaz sadržaja na drugim dijelovima aplikacije, primjerice za kupnju voznih karata ili pregled pretplatnih karata unutar zaslona „Karte”.

Kako bi se omogućio pristup personaliziranim funkcionalnostima, prilikom prvog pokretanja aplikacije korisniku se prikazuje uvodni zaslon s četiri ponuđene opcije: prijava pomoću adrese e-pošte i lozinke, izrada novog korisničkog računa, autentifikacija putem Google računa ili nastavak korištenja aplikacije bez prijave, kao gost. Ovakav fleksibilan pristup, prikazan na slici 53, omogućuje korištenje aplikacije i bez stvaranja računa, ali istovremeno otvara prostor za buduću integraciju s korisničkim sustavima davatelja javnog prijevoza, čime bi se omogućile naprednije funkcionalnosti vezane uz individualni status korisnika.



Slika 53. Prikaz zaslona za prijavu u aplikaciju

Prilikom pokušaja prijave, aplikacija šalje zahtjev prema API-u (*/login*), u kojem se nalaze adresa e-pošte i lozinka unesene u korisničko sučelje. Nakon primitka zahtjeva, API pristupa bazi podataka i provjerava postoji li korisnik čiji podaci odgovaraju unesenima. Ako je prijava uspješna, aplikacija zaprima identifikator korisničkog računa, dok se u slučaju neuspjeha prikazuje poruka o netočnim podacima. Ova funkcija, prikazana na slici 54, predstavlja osnovni mehanizam autentifikacije i potvrđuje mogućnost povezivanja aplikacije s korisničkim računima.

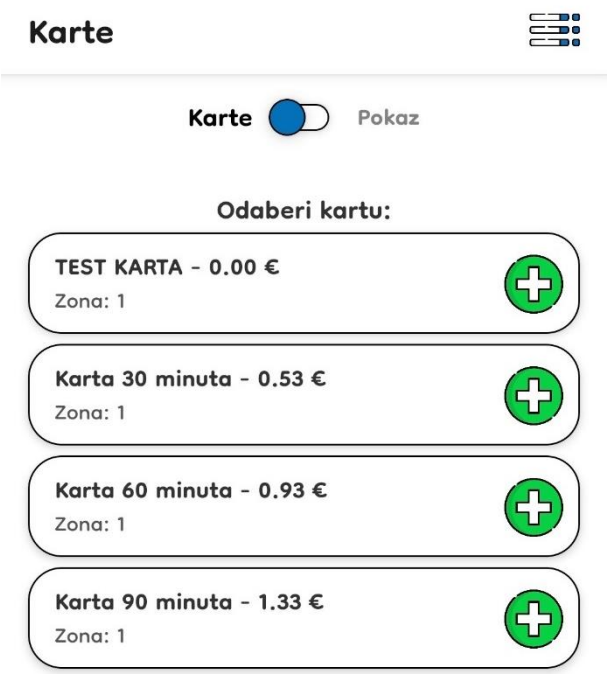
```python
@app.route('/login', methods=['POST'])
def login():
    data = request.get_json()
    email = data.get('email')
    password = data.get('password')

    try:
        with pyodbc.connect(connection_string) as conn:
            cursor = conn.cursor()
            cursor.execute(
                "SELECT id FROM Users WHERE email = ? AND password = ?",
                (email, password)
            )
            row = cursor.fetchone()
            if row:
                return jsonify({'status': 'success', 'user_id': row[0]})
            else:
                return jsonify({'status': 'error', 'message': 'Neispravni podaci'}), 401
    except Exception as e:
        return jsonify({'status': 'error', 'message': str(e)}), 500
```

Slika 54. Funkcija *login*

5.4.2 Zaslon „Karte”

Ovaj dio korisničkog sučelja strukturiran je u dva glavna prikaza između kojih se korisnik može jednostavno prebacivati pomoću interaktivnog prekidača. Navedeni prekidač omogućuje odabir između prikaza „Karte” i „Pokaz”, s ciljem postizanja maksimalne preglednosti i jasnoće navigacije unutar aplikacije.

U prikazu „Karte” korisniku su dostupne interaktivne kartice koje reprezentiraju različite vrste voznih karata, što je vidljivo na slici 55. Svaka kartica sadrži informaciju o trajanju i cijeni karte, dok istaknuta zelena ikona plusa vizualno označava mogućnost odabira i pokretanja procesa kupnje.



Slika 55. Inicijalni prikaz zaslona „Karte”

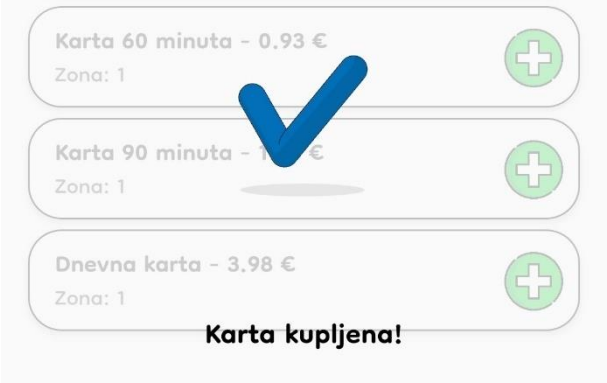
Odabirom željene karte, njezina se kartica vizualno ističe ispunjavanjem plavom bojom, prikazano na slici 56, čime se korisniku jasno daje do znanja koja je karta trenutačno odabrana. Istodobno se ispod kartica dinamički prikazuje dodatni interaktivni okvir koji funkcionira kao gumb s jasno istaknutim tekstom „Kupi kartu” te prikazom pripadajućeg iznosa cijene karte.

Slika 56. Odabir željene karte

Nakon što korisnik odabere gumb „Kupi kartu”, isti se okvir proširuje te se njegov sadržaj ažurira prikazom poruke kojom se korisnika upućuje da potvrdi želi li doista kupiti odabranu kartu, što je vidljivo na slici 57. U proširenom prikazu pojavljuju se dva dodatna gumba, pri čemu plavi gumb nosi oznaku „Potvrdi”, a crveni gumb oznaku „Odustani”, čime se korisniku omogućuje jednostavno donošenje konačne odluke.

Slika 57. Proširenje okvira radi dodatne potvrde

Potvrdom kupnje pokreće se kratka animacija koja vizualno potvrđuje uspješnu transakciju, kako je prikazano na slici 58.



Slika 58. Animacija potvrde kupnje karte

Po završetku animacije potvrde plaćanja, njezina se središnja ikona u obliku oznake potvrde zamjenjuje ikonom QR koda. Ta se ikona postupno smanjuje i animirano prelazi prema novo dodanom informativnom okviru smještenom u donjem dijelu zaslona, kako je prikazano na slici 59. Na taj se način korisniku vizualno daje do znanja da je njegova digitalna karta dostupna unutar tog okvira.

Sam okvir prikazuje osnovne podatke o kupljenoj karti, uključujući njezinu vrstu, preostalo trajanje valjanosti i pripadajuću ikonu QR koda. Neposredno ispod nalazi se traka napretka (engl. *Progress bar*) koja korisniku na intuitivan način prikazuje koliko je vremena preostalo do isteka valjanosti karte. Traka se postupno smanjuje i pritom mijenja boju, počevši od zelene, zatim prelazeći u narančastu i naposljetku u crvenu.

Za vrijeme trajanja aktivne karte, sve opcije za kupnju karata prikazuju se u neaktivnom (posivljenom) obliku, čime se korisniku jasno daje do znanja da istovremeno nije moguće kupiti dodatnu kartu.

Slika 59. Prikaz zaslona „Karte" za vrijeme aktivne kupljene karte

Dodirom na okvir QR kod se otvara u punoj veličini, prikazano na slici 60, što omogućuje njegovu jasnu vidljivost i jednostavno očitavanje tijekom kontrole karata.

Slika 60. Prikaz QR koda kupljene karte

Prilikom isteka valjanosti kupljene karte, opcije za kupnju ponovno postaju dostupne, a okvir u kojem se prikazivala aktivna karta zamjenjuje se novim prikazom koji sadrži povijest svih prethodno kupljenih karata, što je vidljivo na slici 61, omogućujući korisniku uvid u vlastite transakcije.

Slika 61. Prikaz okvira „Povijest kupljenih karata”

Odabirom okvira s poviješću prikazuju se kartice koje sadrže pregled svih prethodno kupljenih karata, zajedno s pripadajućim informacijama kao što su vrsta karte, cijena i vrijeme kupnje, vidljivo na slici 62. U zaglavlju prikaza smještena je dodatna ikona koja korisniku omogućuje povratak na osnovni prikaz „Karte”.

Slika 62. Prošireni prikaz povijesti kupljenih karata

U drugom prikazu, označenom kao „Pokaz", korisniku se prikazuje QR kod digitalnog pokaza, koji je zamišljen kao virtualna inačica stvarnog pretplatnog dokumenta, što je vidljivo na slici 63. U slučaju integracije s davateljem usluge, QR kod bi bio generiran iz službene baze podataka te bi posjedovao pravnu valjanost za korištenje u javnom prijevozu, što trenutačno nije slučaj.

Slika 63. Prikaz zaslona „Pokaz"

6 ZAKLJUČAK

U okviru ovog rada razvijen je sveobuhvatan informacijski sustav koji korisnicima javnog gradskog prijevoza omogućuje pouzdan, precizan i jednostavan pristup informacijama o prometu u stvarnom vremenu. Sustav se temelji na višeslojnoj arhitekturi u sklopu koje se statički i dinamički podaci prikupljaju, obrađuju i sinkroniziraju putem poslužiteljskog servisa, a zatim prikazuju unutar mobilne aplikacije. Aplikacija obuhvaća čitav niz funkcionalnosti, počevši od prikaza dolazaka vozila i njihove trenutne pozicije na karti, preko personaliziranih postavki i predikcije vremena dolaska, pa sve do podrške za kupnju i validaciju karata, čime se zaokružuje suvremeno digitalno korisničko iskustvo, usmjereno na jednostavnost i intuitivnost.

Postignuti rezultati potvrđuju kako je primjenom dostupnih otvorenih podataka i suvremenih tehnologija moguće značajno unaprijediti kvalitetu informiranja putnika u javnom prijevozu. Razvijeni sustav pruža visoku razinu pouzdanosti u prikazu informacija u stvarnom vremenu, uz korisničko sučelje oblikovano u skladu sa svakodnevnim potrebama putnika. Time ovaj projekt ne predstavlja samo tehničko rješenje, već i konkretan doprinos digitalnoj transformaciji javnog prijevoza i unaprjeđenju urbane mobilnosti.

Cjelokupan rad aplikacije prikazan je u videozapisu dostupnom na platformi YouTube, na sljedećoj poveznici: https://youtu.be/crGpP_DdGYM. Video demonstracija pruža uvid u način korištenja aplikacije, izgled i strukturu sučelja, prikaz informacija u stvarnom vremenu te sve ključne funkcionalne komponente sustava.

Sljedeća faza razvoja usmjerena je na završnu pripremu aplikacije za objavljivanje na Google Play i Apple App Store platformama, čime će ona postati široko dostupna krajnjim korisnicima. Planira se razvoj mehanizma za automatsko prepoznavanje i učitavanje novih verzija statičkih GTFS podataka odmah po njihovoj objavi, što bi dodatno doprinijelo automatizaciji i dugoročnoj održivosti sustava. Također se planira nadogradnja modula za predviđanje dolazaka vozila, pri čemu će se dosadašnji pristup temeljen na prosječnim vrijednostima vremena dolaska zamijeniti naprednijim modelima utemeljenima na strojnom učenju, čime bi se mogla postići značajno veća točnost prognoza. Poseban naglasak bit će stavljen na daljnju optimizaciju korisničkog sučelja, identificiranje i otklanjanje mogućih grešaka u radu te unaprjeđenje performansi i skalabilnosti aplikacije.

U konačnici, ostvareni rezultati otvaraju mogućnost buduće suradnje s davateljem usluge javnog gradskog prijevoza, s ciljem integracije razvijenog rješenja u službene kanale informiranja, čime bi se dodatno povećala njegova dostupnost, relevantnost i korisnička vrijednost, a time i ukupna učinkovitost sustava javnog prijevoza.

7 ZAHVALE

Zahvaljujemo se našem mentoru, prof. dr. sc. Tončiju Carić, na strpljenju i vođenju tijekom izrade ovog rada.

Posebno se zahvaljujemo dr. sc. Tomislavu Erdeliću na korisnim savjetima, vrijednom iskustvu i kontinuiranoj podršci. Njegova prisutnost i angažman tijekom cijelog istraživanja bili su ključni za uspješan završetak rada.

Zahvaljujemo se i doc. dr. sc. Miroslavu Vujiću na poticajnom pristupu i podršci tijekom nastave. Upravo je kroz njegov kolegij predložena ideja da osmislimo vlastiti projekt, što je u konačnici potaknulo razvoj ovog rada.

Također, zahvaljujemo se Nikoli Mardešiću, mag. ing. traff., na korisnoj podršci tijekom razvoja projekta.

Zahvaljujemo se Zavodu za inteligentne transportne sustave na njihovoj podršci i pristupu resursima.

LITERATURA

1. Monzon A, Hernandez S, Cascajo R. Benefits of real-time passenger information: Helping people shift toward more sustainable transport modes. Transport and Telecommunication. 2013.

2. Bošnjak I. Inteligentni transportni sustavi - ITS 1 Zagreb: Fakultet prometnih znanosti; 2006.

3. Klindić I. Fotografije Zagreba. [Online]. [cited 2025 Kolovoz. Available from: https://fotografijezagreba.hr/.

4. Jutarnji list. ZET krenuo u modernizaciju. [Online].; 2022 [cited 2025 Kolovoz. Available from: https://www.jutarnji.hr/vijesti/hrvatska/zet-krenuo-u-modernizaciju-novi-uredaj-zasad-samo-u-jednom-a-do-kraja-godine-u-pedeset-tramvaja-15230090.

5. Zagrebački električni tramvaj. Moj ZET. [Online]. [cited 2025 Kolovoz. Available from: https://moj.zet.hr/.

6. Tuček L. ZET info. [Online]. [cited 2025 Kolovoz. Available from: https://zet-info.com/.

7. Tu Z. Research on the Application of Layered Architecture in Computer Software Development. Journal of Computing and Electronic Information Management. 2023.

8. Wong J. Leveraging the General Transit Feed. Transportation Research Record: Journal of the Transportation Research Board. 2013.

9. Portal otvorenih podataka. Otvoreni podaci. [Online]. [cited 2025 Kolovoz. Available from: https://data.gov.hr/o-portalu-otvorenih-podataka-i-sto-su-otvoreni-pod.

10. General Transit Feed Specification. General Transit Feed Specification Reference. [Online]. [cited 2025 Kolovoz. Available from: https://gtfs.org/documentation/schedule/reference/.

11. General Transit Feed Specification. GTFS Realtime Reference. [Online]. [cited 2025 Kolovoz. Available from: https://gtfs.org/documentation/realtime/reference/.

12. Google Developers. Protocol Buffers. [Online]. [cited 2025 Kolovoz. Available from: https://protobuf.dev/.

13. Fielding RT, Reschke JF. Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests. Internet Engineering Task Force. 2014.

14. OpenAI. Sora. [Online]. [cited 2025 Kolovoz. Available from: https://sora.chatgpt.com/explore.

15. RSS Advisory Board. RSS 2.0 Specification. [Online]. [cited 2025 Kolovoz. Available from: https://www.rssboard.org/rss-specification.

16. GitHub. Mozilla. PDF.js – A general-purpose, web standards-based platform for parsing and rendering PDFs. [Online]. [cited 2025 Kolovoz. Available from: https://github.com/mozilla/pdf.js.

SAŽETAK

Dominik Knez, Ante Šarić

Razvoj mobilne aplikacije za informiranje putnika javnog gradskog prijevoza

U suvremenim urbanim sredinama javni prijevoz predstavlja jedan od ključnih elemenata održive mobilnosti, a njegova kvaliteta sve se više povezuje s dostupnošću točnih i ažurnih informacija. Putnicima je pritom jednako važna pouzdanost informacija kao i fizička infrastruktura ili tehnička obilježja vozila. Kada raspolažu preciznim podacima o dolascima, presjedanjima i mogućim odstupanjima od voznog reda, korisnici mogu donositi informirane odluke i s većim povjerenjem birati javni prijevoz. U radu se polazi od analize postojećih rješenja u gradu Zagrebu, pri čemu je utvrđeno da dostupne aplikacije i informacijski sustavi ne zadovoljavaju u potpunosti stvarne potrebe korisnika, ponajprije zbog nedostatne točnosti podataka u stvarnom vremenu i neintuitivnih korisničkih sučelja.

Kao odgovor na uočene izazove razvijen je sustav koji povezuje statičke i dinamičke podatke javnog prijevoza te ih objedinjuje u jedinstvenu i preglednu cjelinu. Sustav je oblikovan kroz višeslojnu arhitekturu koja omogućuje pouzdanu obradu i prijenos informacija prema korisnicima. Na toj osnovi izrađena je mobilna aplikacija koja putnicima nudi prikaz dolazaka vozila u stvarnom vremenu, pregled linija i stajališta, vizualizaciju kretanja vozila na karti, obavijesti o izmjenama u prometu te mogućnost digitalne kupnje i validacije karata. Posebna je pažnja posvećena predviđanju vremena dolaska vozila na stajališta, čime se postiže viša razina točnosti prikazanih informacija i jača povjerenje korisnika u sustav.

Rezultati rada pokazuju da kombinacija strukturirane obrade podataka i suvremenog korisničkog sučelja može značajno unaprijediti iskustvo putnika i doprinijeti većoj privlačnosti javnog prijevoza. Time se potvrđuje važnost digitalnih tehnologija u razvoju inteligentnih transportnih sustava te njihova ključna uloga u oblikovanju održivih i pametnih gradova.

Ključne riječi: Javni gradski prijevoz; Informiranje putnika; Mobilna aplikacija; Podaci u stvarnom vremenu; Predikcija dolazaka vozila

SUMMARY

Dominik Knez, Ante Šarić

Development of a mobile application for passenger information in urban public transport

In modern urban environments, public transport represents one of the key elements of sustainable mobility, and its quality is increasingly associated with the availability of accurate and up-to-date information. For passengers, the reliability of information is equally important as the physical infrastructure or the technical characteristics of vehicles. When provided with precise data on arrivals, transfers, and possible deviations from the timetable, users can make informed decisions and choose public transport with greater confidence. This paper begins with an analysis of existing solutions in the city of Zagreb, revealing that currently available applications and information systems do not fully meet the actual needs of users, primarily due to insufficient real-time accuracy and non-intuitive user interfaces.

As a response to these challenges, a system was developed that integrates static and dynamic public transport data into a unified and accessible whole. The system is designed through a multi-layer architecture that ensures reliable processing and transfer of information to passengers. On this basis, a mobile application was created, offering real-time vehicle arrivals, an overview of lines and stops, visualization of vehicle movement on the map, notifications on service changes, as well as digital ticket purchase and validation. Special attention was devoted to predicting vehicle arrival times at stops, thereby ensuring greater accuracy of the displayed information and strengthening user trust in the system.

The results demonstrate that the combination of structured data processing and a modern user interface can significantly improve the passenger experience and contribute to the greater attractiveness of public transport. This confirms the importance of digital technologies in the development of intelligent transport systems and highlights their crucial role in shaping sustainable and smart cities.

Keywords: Urban public transport; Passenger information; Mobile application; Real-time data; Vehicle arrival prediction

ŽIVOTOPIS AUTORA

Dominik Knez rođen je 29. travnja 2000. godine u Zagrebu, Hrvatska. Godine 2019. završio je svoje srednjoškolsko obrazovanje u Gornjogradskoj gimnaziji u Zagrebu, opći smjer. Godine 2023. završio je preddiplomski studij Inteligentni transportni sustavi i logistika na Fakultetu prometnih znanosti Sveučilišta u Zagrebu, smjer Inteligentni transportni sustavi. Trenutno je na diplomskom studiju na istom smjeru na istom fakultetu.

Ante Šarić rođen je 26. siječnja 2000. godine u Šibeniku, Hrvatska. Godine 2018. završio je svoje srednjoškolsko obrazovanje u Tehničkoj školi Zadar, smjer Zrakoplovni tehničar za instrumente, radio i elektroopremu. Godine 2023. završio je preddiplomski studij Inteligentni transportni sustavi i logistika na Fakultetu prometnih znanosti Sveučilišta u Zagrebu, smjer Inteligentni transportni sustavi. Trenutno je na diplomskom studiju na istom smjeru na istom fakultetu.

POPIS SLIKA