Izrada Web aplikacije za prikaz podataka dobivenih iz GPS trajektorija praćenih vozila

Knežević, Aleksandar

SVEUČILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI

Aleksandar Knežević

Izrada web aplikacije za prikaz podataka dobivenih iz GPS trajektorija praćenih vozila

DIPLOMSKI RAD

Zagreb 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI
POVJERENSTVO ZA DIPLOMSKI ISPIT

Zagreb, 31. ožujka 2023.

Zavod: **Zavod za inteligentne transportne sustave**
Predmet: **Napredne baze podataka**

DIPLOMSKI ZADATAK br. 7185

Pristupnik: **Aleksandar Knežević (0035200741)**
Studij: Inteligentni transportni sustavi i logistika
Smjer: Inteligentni transportni sustavi

Zadatak: **Izrada Web aplikacije za prikaz podataka dobivenih iz GPS trajektorija praćenih vozila**

Opis zadatka:

Cilj rada je izraditi interaktivnu web kartu za prikaz povijesnih podataka dobivenih iz GPS trajektorija praćenih vozila. Potrebno je kreirati bazu podataka u koju će se spremiti povijesne GPS trajektorije, te izraditi interaktivnu web aplikaciju s različitim opcijama dohvata i prikaza podataka iz baze podataka.

Mentor:

Tomislav Erdelić

dr. sc. Tomislav Erdelić

Predsjednik povjerenstva za diplomski ispit:

SVEUČILIŠTE U ZAGREBU

FAKULTET PROMETNIH ZNANOSTI

DIPLOMSKI RAD

Izrada web aplikacije za prikaz podataka dobivenih iz GPS trajektorija praćenih vozila

Development of a Web Application for Displaying Data Obtained from GPS Trajectories of Tracked Vehicles

Mentor: dr. sc. Tomislav Erdelić

Student: Aleksandar Knežević
JMBAG: 0035200741

Zagreb 2023.

Izrada web aplikacije za prikaz podataka dobivenih iz GPS trajektorija praćenih vozila

Sažetak:

Programska rješenja za GPS praćenje vozila mogu pomoći poslovanju pri povećanju produktivnosti i efikasnosti uz mogućnosti pronalaženja boljih ekonomskih rješenja poslovanja te smanjenja troškova operacija. Poznavajući stvarno stanje prometnog procesa svoje flote vozila kao što su praćenje vozila i trajanje procesa uveliko može olakšati planiranje naših troškova i optimizacije poslovanja.

U ovom radu obrađeni su podaci prikupljeni iz GPS trajektorija praćenih vozila te je dizajnirana i implementirana relacijska baza podataka u koju su spremljeni navedeni podaci. Analizirane su mogućnosti ASP.NET platforme unutar Microsoft Visual Studia s ciljem izrade web sučelja za prikaz i uređivanje podataka dobivenih iz GPS trajektorija praćenih vozila. Web sučelje povezano je s relacijskom bazom podataka.

Kao rezultat ovog rada izrađeni su aplikacija koja sama obrađuje i unosi podatke u bazu podataka te web sučelje s interaktivnom kartom povezanom s dotičnom bazom podataka.

Ključne riječi: SQL; Visual Studio; C#; ASP.NET; Delphi; OSM karta

Development of a Web application for displaying data obtained from GPS trajectories of tracked vehicles

Abstract:

Software solutions for GPS vehicle tracking can help businesses increase productivity and efficiency with the possibility of finding better economic business solutions and reducing operating costs. Knowing the real state of the traffic process of our fleet of vehicles, such as vehicle tracking and process duration, can greatly facilitate cost planning and business optimization.

In this paper, the collected data obtained from the GPS trajectories of tracked vehicles were processed, and a relational database was designed and implemented in which the above data was stored. The capabilities of the ASP.NET platform within Microsoft Visual Studio were analyzed with the aim of creating a web interface for displaying and editing data obtained from GPS trajectories of tracked vehicles. The web interface is connected to a relational database.

As a result of this work, an application that itself processes and enters data into the database and a seb interface with an interactive map connected to the respective database were created.

Key Words: SQL; Visual Studio; C#; ASP.NET; Delphi; OSM map

Sadržaj

1. Uvod

Povećanjem količine prometa i potrebe za transportom pojavljuje se potreba za unaprjeđenjem prometne infrastrukture i razvoja prometnih tehnologija. Fizička ograničenost te ekonomska neopravdanost konstantne izgradnje novih prometnica s ciljem zadovoljenja nove prometne potražnje doveli su do razvitka nove prometne grane zvane Inteligentni transportni sustavi (ITS).

ITS se može opisati raznim definicijama, a u suštini ITS je spoj informacijskih i komunikacijskih tehnologija i aplikacija u sklopu područja prometa. ITS-u najviše doprinose informacijske i telekomunikacijske tehnologije te telematika.

Telematika je spoj informacijske i komunikacijske tehnologije za slanje i primanje podataka. Telematika nam pomaže u planiranju rute, lociranju flote vozila, nadzoru vozača, nadzoru rada itd. Razvojem telematike omogućeno je uvođenje kompliciranijih ITS rješenja. Primjerice razvojem kamera koje imaju mogućnost otkrivanja umora u očima vozača razvilo se ITS rješenje koje određuje prekid vožnje i time povećava sigurnost u prometu.

Jedan od telematičkih alata je Globalni sustav pozicioniranja (eng. Global Positioning System, GPS) koji se može ugraditi unutar vozila kako bi olakšalo praćenje vozila. GPS je satelitski radionavigacijski sustav razvijen i u vlasništvu Sjedinjenih Američkih Država.

Razvojem tehnologije i računalnih sustava došlo je do potrebe za uređivanjem i spremanjem velike količine podataka iz čega su nastale baze podataka. Baza podataka je organizirana skupina podataka uglavnom elektronički pohranjenih u računalnom sustavu. Bazom podataka uglavnom upravlja sustav za upravljanje bazom podataka (eng. Database Management System, DBMS)

Napretkom tehnologije, ITS-a, telematike i GPS-a razvijaju se programska rješenja za GPS praćenje vozila. Postoje razne mogućnosti i funkcionalnosti GPS-a u praćenju flote vozila: planiranje rute, upravljanje gorivom, optimizacije prijevoznog procesa, optimizacije logističkog procesa, kontrola vozačevog vremena i rada, optimizacije samih prometnica kroz upravljanje prometnog toka i sl.

Aplikacija izrađena u ovom radu kombinacija je više tehnoloških rješenja: bazi podataka, GPS trajektorija praćenih vozila i digitalne karte. Aplikacija može ispuniti više mogućnosti i funkcionalnosti ovisno u svrsi za koju bi se koristila.

1.1. Ciljevi rada

Cilj ovog diplomskog rada je prikupljene podatke GPS trajektorija praćenih vozila povezati i prikazati na karti u web sučelju. Prikupljene podatke potrebno je obraditi te ih unijeti u dizajniranu i implementiranu bazu podataka. Potrebno je analizirati funkcionalnosti i mogućnosti ASP.NET platforme unutar razvojnog okruženja Microsoft Visual Studia s ciljem izrade web sučelja za prikaz podataka dobivenih iz GPS trajektorija praćenih vozila. U sklopu rada navedeno web sučelje potrebno je povezati s bazom podataka. Svrha ovog rada je izraditi web aplikaciju za prikaz podataka dobivenih iz GPS trajektorija praćenih vozila.

Ciljevi rada:

- opis, pohrana i obrada podataka
- priprema podataka za prikaz na web sučelju
- izrada grafičkog web sučelja
- implementacija veze između baze podataka i grafičkog web sučelja

1.2. Struktura rada

U uvodnom poglavlju opisani su ciljevi i svrha diplomskog rada koji će dovesti do krajnjih rezultata. U drugom poglavlju opisani su prikupljeni podaci, korištena karta, njihov značaj te njihovo korištenje. U trećem poglavlju opisana je SQL baza podataka te način na koji će podaci biti pohranjeni u bazu podataka. U četvrtom poglavlju opisano je razvojno okruženje u kojem će se izraditi grafički dio web sučelja. U navedenom poglavlju analiziraju se i određene dodatne mogućnosti programskog paketa Visual Studio. U istom poglavlju opisan je i način na koji se baza podataka spaja s razvojnim okruženjem i način rada aplikacije. U petom poglavlju prikazana je analiza rada web aplikacije. Zadnje poglavlje predstavlja zaključak rada u kojem se daje osvrt na odrađeno i moguća poboljšanja sustava.

2. GPS podaci trajektorija vozila

U sklopu ovog diplomskog rada korišteni su podaci iz datoteke GPS_podaci. Unutar datoteke GPS_podaci nalaze se podaci o trajektorijama vozila na području Zagreba. Poligon razmatranog područja prikazan je na slici 1, a korištene koordinate su prikazane u tablici 1.

Slika 1: Prikaz razmatranog poligona

Tablica 1: Koordinate poligona

Koordinata	Geografska širina	Geografska dužina
1	15.831590	45.745499
2	15.831590	45.858586
3	16.161497	45.858586
4	16.161497	45.745499

Zabilježene su rute vozila koji su prošle unutar razmatranog poligona. Jedan redak datoteke odgovara jednoj ruti koja naravno ima varijabilan broj GPS zapisa ovisno o duljini rute. Jedan GPS zapis sastoji se od točno 9 podataka koji se vremenski ponavljaju kroz rutu kako se vozilo kreće po cestovnoj mreži. Format i primjer jednog zapisa je sljedeći.

1389367308;16.160558760166168;45.793641131601149;16.160628497600555;45.793769237134569;42;290;-391666;107

UTC;MATCH_X;MATCH_Y;ORG_X;ORG_Y;ORG_S;ORG_H;LINK_ID;MATCH_D

Zapisi unutar jedne rute su vremenski sortirani, a atributi pojedinog zapisa odvojeni su znakom ";", a na kraju rute (niza zapisa) dodan je znak za novi redak. Opis atributa dan je u tablici 2.

Tablica 2: Opis atributa

Atribut	Primjer	Opis
UTC	1389367308	UTC vrijeme u sekundama
MATCH_X	15.952734500169754	Pridružena geografska dužina (°)
MATCH_Y	45.780397919459887	Pridružena geografska dužina (°)
ORG_X	15.952774733304977	Originalna geografska dužina (°)
ORG_Y	45.7804016606603244	Originalna geografska širina (°)
ORG_S	50	Originalna brzina kretanja (km/h)
ORG_H	348	Originalni geografski smjer kretanja u odnosu na sjever (°) (360°=0°)
LINK	-214696	ID pridruženog linka digitalnoj karti
MATCH_D	110	Pridružena prijeđena cestovna udaljenost od prethodnog zapisa (m)

Podaci korišteni u diplomskom radu prikupljeni su u sklopu SORDITO projekta te su ustupljeni od strane Fakulteta prometnih znanosti. Korišteni podaci trajektorija vozila prikupljeni su periodu od kolovoza 2009. do listopada 2014. godine preko 4908 vozila. Digitalna karta korištena u sklopu praćenja vozila je ustupljena od strane Mireo d.d. Vozila su bila opremljena GNSS (eng. Global Navigation Satellite System) uređajima za praćenje. GNSS uređaji su bili ugrađeni u teretna vozila, taksi vozila, osobna vozila, vozila za prijevoz robe i sl. Količina prikupljenih podataka iznosila je 6.55 milijardi zapisa, spremljenih u 320 GB računalnog prostora, [1].

Veličina datoteke GPS_podaci iznosi 3.2 GB. Konzolnom naredbom putem Command Prompta dobio se broj linija unutar datoteke GPS_podaci. Naredba **type GPS_podaci.txt | find /v /c ""** prikazala je 518047 linija u datoteci. Svaka pojedinačna linija sadrži informaciju o jednoj ruti. U sklopu ovog rada u bazu podataka, iz datoteke GPS_podaci, učitano je 80986 linija što predstavlja 4 330 144 zapisa.

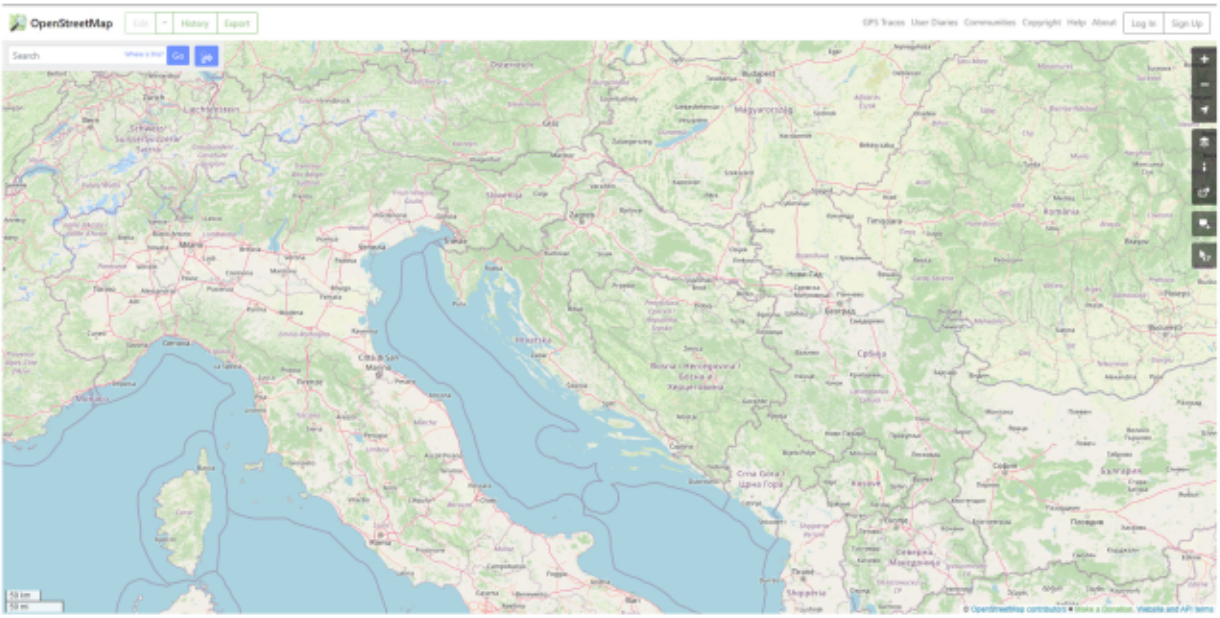
2.1. OpenStreetMap karta

U svrhu ovog rada koristit će se OpenStreetMap karta. OpenStreetMap (OSM) je velika besplatna baza podataka geografskih podataka otvorenog pristupa. OSM je izgradila zajednica kartografa koji doprinose i održavaju podatke o cestama, stazama, kafićima, željezničkim postajama i još mnogo toga, diljem svijeta, [2].

OSM se gradi na znanju kartografa u lokalnoj zajednici. Suradnici koriste snimke iz zraka, GPS uređaje i terenske karte niske tehnologije kako bi potvrdili da je OSM točan i ažuran, [2].

OSM je baza podataka otvorenog pristupa: slobodni ste ju koristiti u bilo koju svrhu sve dok referencirate OpenStreetMap i njegove suradnike. Ako su podaci mijenjani ili nadograđivani na određene načine, rezultati se mogu distribuirati samo pod tom licencom, [2].

Kada korisnik napravi promjene pomoću softvera za uređivanje kao što je Java OpenStreetMap editor (JOSM) ili iD (Javascript OSM editor), softver komunicira sa središnjim OpenStreetMap poslužiteljem i obavještava ga o promjenama. Na tom poslužitelju nalazi se ogromna baza podataka koja sadrži sve podatke o lokaciji i atribute o svakom pojedinom zemljopisnom obilježju u cijelom OpenStreetMapu, [3]. U nastavku na slici 2 je prikazan izgled OSM karte.



Slika 2: Prikaz OSM karte

Za rad na OSM karti preko aplikacija potrebno je preuzeti OSM kartu ili dijelove OSM karte. Preuzimanje OSM karte moguće je s različitih web stranica. Standardni oblik podataka pri preuzimanju je u XML (eng. Extensive Markup Language) formatu. Jedna takva stranica je Planet OSM na kojoj se mogu preuzeti podaci za kartu cijelog planeta (Zemlje) i/ili izvadci određenih dijelova s karte. Veličina karte planeta je 130 GB-a, a veličina dijelova preuzetih s karte ovise o veličini područja koje se želi preuzeti. Veličina podataka takvih preuzetih područja su uglavnom u rasponu od 100 MB – 1GB. Primjerice veličina podataka za Hrvatsku je oko 150MB-a, [4].

3. Relacijska baza podataka

Relacijska baza podataka je virtualni prostor u koji se spremaju prikupljeni podaci iz stvarnog svijeta u formatu koji omogućuje da se upravlja prikupljenim podacima u programskom okruženju. Programi baza podataka koriste neku od verzija SQL (eng. Structured Query Language) jezika. Iako postoji više verzija SQL jezika, sve verzije, podupiru česte naredbe kao štu su SELECT, UPDATE, DELETE, INSERT i sl. U ovom poglavlju bit će opisano razvojno okruženje SQL Server Management Studio (SSMS) te dizajn relacijske baze podataka.

3.1. Organizacija baze podataka

U ovom potpoglavlju prikazana je količina učitanih podataka, dizajn baze podataka te pregled podataka unutar same baze.

Pri izradi diplomskog rada korišteno je Microsoftovo SQL Server Management Studio (SSMS) razvojno okruženje za izradu relacijske baza podataka. SSMS je razvijen od strane Microsofta te je to integrirana okolina te koristi Microsoftovu vrstu bazi podataka (MSSQL). SMSS omogućuje pristup, podešavanje, administriranje i upravljanje svim komponentama SQL Servera i SQL baze podataka. SMSS nudi jednostavan skup alata (široka skupina grafičkih alata i brojne skripte) kojima se može razviti željena baza podataka, [5]. U sklopu rada podaci trajektorija vozila su spremljeni u tablicu **croutes** u bazi podataka. Prikaz tablice se vidi na slici 3.

Column Name	Data Type	Allow Nulls
ID_route	int	☐
UTC	int	☑
MATCH_X	float	☑
MATCH_Y	float	☑
ORG_X	float	☑
ORG_Y	float	☑
ORG_S	int	☑
ORG_H	int	☑
LINK_ID	int	☑
MATCH_DIST	float	☑

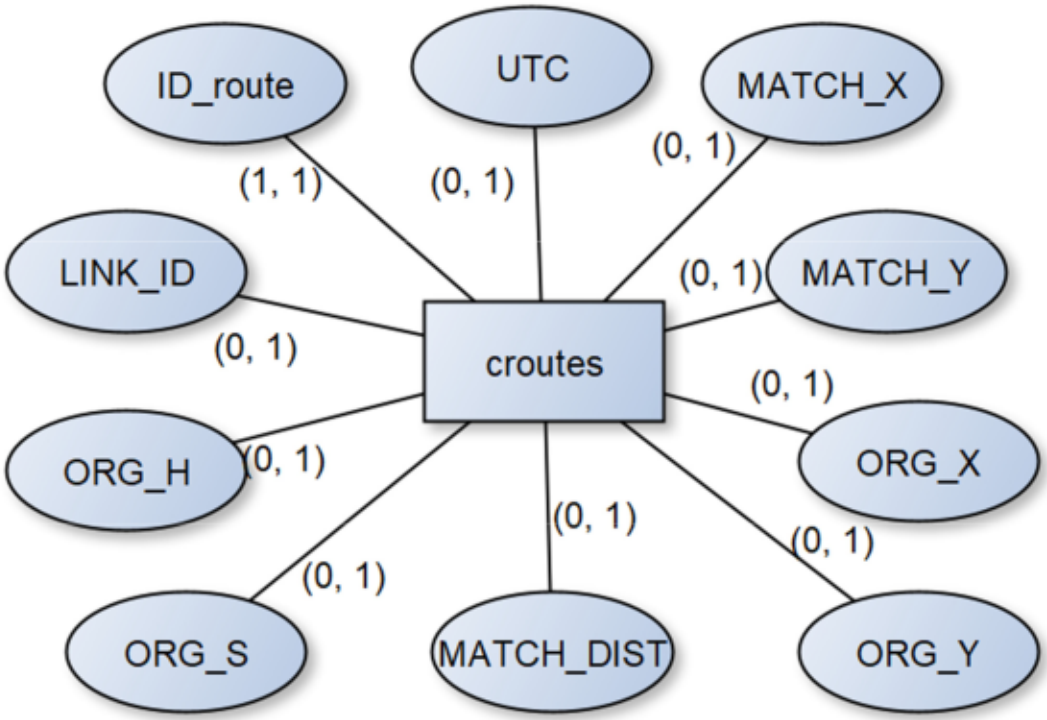
Slika 3: Tablica croutes

U tablici se nalaze sljedeći stupci sa sljedećim tipovima podataka i svojstvima:

- ID_route; cjelobrojna vrijednost (eng. integer, int); mora sadržavati vrijednost (NOT NULL)
- UTC; int; ne mora sadržavati vrijednost (NULL)
- MATCH_X; realni broj (eng. float); NULL
- MATCH_Y; float; NULL
- ORG_X; float; NULL
- ORG_Y; float; NULL
- ORG_S; int; NULL
- ORG_H; int; NULL
- LINK_ID; int; NULL
- MATCH_DIST; float; NULL

Tip podatka int se koristi za predstavljanje cjelobrojnih vrijednosti dok se tip podatka float koristi za brojeve realnih vrijednosti. U bazi nisu korišteni strani i primarni ključevi jer nije bilo potrebe za njima.

Shema ove baze podataka prikazana je dijagramom entiteta na slici 4.



Slika 4: Dijagram entiteta tablice croutes

U bazi je uneseno 80986 ruta što se vidi sa slike 5. Ukupan broj ruta se dobije SQL upitom s dodatnom uputom distinct koja dozvoljava da se svaki zapis ID_route pojavi samo jedanput. S dodatnim zapisom order by desc se postiglo da je zadnja zabilježena ruta na vrhu tablice.

```
SELECT distinct ID_route
  FROM [Baza_podataka_diplomski].[dbo].[croutes] order by id_route desc
```

Results | Messages

ID_route
80986
80985
80984
80983

Slika 5: Prikaz broja ruta unutar baze podataka SQL upitom

Ukupan broj zapisa je 4 330 144 zapisa što se vidi iz slike. Ukupan broj zapisa se dobije SQL upitom preko naredbe count koja broji pojavljivanja zapisa za bilo koje polje u tablici.

```
SELECT count(ID_ROUTE)
  FROM [Baza_podataka_diplomski].[dbo].[croutes]
```

100 %

Results | Messages

	(No column name)
1	4330144

Slika 6: Prikaz broja zapisa unutar baze podataka SQL upitom

Baza je napravljena tako da uz svaki zapis od 9 atributa dodan je ID rute. Takvim načinom zapisivanja rezultirao je skupljanjem svih zapisa jedne rute pod istim ID-em rute što se vidi iz slike 7. SQL upitom je izdvojen ograničen broj podataka iz tablice za prikaz.

```
SELECT TOP 1000 [ID_route]
      ,[UTC]
      ,[MATCH_X]
      ,[MATCH_Y]
      ,[ORG_X]
      ,[ORG_Y]
      ,[ORG_S]
      ,[ORG_H]
      ,[LINK_ID]
      ,[MATCH_DIST]
  FROM [Baza_podataka_diplomski].[dbo].[croutes]
```

Results | Messages

ID_route	UTC	MATCH_X	MATCH_Y	ORG_X	ORG_Y	ORG_S	ORG_H	LINK_ID	MATCH_DIST
1	1389367308	16.1605587601662	45.7936411316011	16.1606284976006	45.7937692371346	42	290	-391666	107
1	1389367317	16.1592458188534	45.7939908496613	16.1593115329742	45.7941124089009	45	290	-391666	109
1	1389367325	16.1579771339893	45.7943284096647	16.1580482125282	45.7944611889395	49	290	-391666	105
1	1389367333	16.1566092073917	45.7946921493186	16.1566869914532	45.7948352133649	52	290	-391666	113
1	1389367340	16.1553499102592	45.7950194194719	16.1554303765297	45.7951708981493	54	290	-391665	104
1	1389367347	16.1540651321411	45.7953532330482	16.1541429162025	45.7955009706313	54	290	-391665	106
1	1389367354	16.1527656018734	45.7956711488809	16.1528420448303	45.7958273009933	55	288	-391664	107
1	1389367361	16.1514472961426	45.7959881278613	16.151519715786	45.7961377338334	55	288	-391664	108
1	1389367368	16.1501222848892	45.7962929496049	16.1501920223236	45.7964481649441	54	286	-391664	108
1	1389367375	16.148826777935	45.796579069111	16.1488991975784	45.7967389587834	54	286	-391664	106
1	1389367381	16.1475178599358	45.7968698622671	16.1475875973701	45.7970241409782	53	286	-391664	106
1	1389367388	16.1462223529816	45.7971774842435	16.1462880671024	45.7972990365321	54	286	-391663	106
1	1389367395	16.1449858546257	45.797505674814	16.1450287699699	45.7975842158605	54	286	391669	103
1	1389367402	16.1436849832535	45.797853498608	16.1437050998211	45.7978908988867	55	290	391669	108
1	1389367411	16.1421655118465	45.7981929052172	16.1421748995781	45.7982303052681	57	282	391670	124
1	1389367418	16.1407493054867	45.7983799052205	16.1407546699047	45.7984032801768	55	280	391670	112
1	1389367426	16.1392553150654	45.7986136543421	16.1392794549465	45.7986716239725	53	286	391670	119
1	1389367433	16.138012111187	45.7989250066491	16.138017475605	45.7989371615086	52	286	391640	103
1	1389367440	16.1367420852184	45.7992672116822	16.136746108532	45.7992737565668	56	294	391641	106
1	1389367447	16.1355337500572	45.7997851901659	16.1355324089527	45.7997851901659	57	306	391644	110
1	1389367454	16.1344675719738	45.8004789371979	16.1344622075558	45.8004770672714	59	318	391644	113
1	1389367461	16.1336266994476	45.8013260074406	16.133595854044	45.8013147880514	59	328	391644	115
1	1389367468	16.1329977214336	45.8022684280726	16.1329963803291	45.8022684280726	59	340	391644	116
1	1389367475	16.1326718330383	45.8032856261036	16.1326718330383	45.8032865610197	60	350	391644	116
1	1389367482	16.1326691508293	45.8043243082366	16.1326463520527	45.804326178034	57	4	391644	116
1	1389367489	16.1328582465649	45.8052900502178	16.1328636109829	45.8052900502178	55	8	391645	109
1	1389367496	16.1331385374069	45.8062314689286	16.1331318318844	45.8062333386619	56	10	391645	107
1	1389367502	16.133424192667	45.8071990475486	16.133418828249	45.807199982399	57	10	391645	110
1	1389367510	16.1337567865849	45.8083348792659	16.1337795853615	45.8083292702773	59	14	-391681	130
1	1389367517	16.1342141032219	45.8093370094996	16.1342369019985	45.8093314006119	60	18	-391681	117
1	1389367523	16.1346915364265	45.8101895538625	16.1347009241581	45.8101858146613	60	22	-391681	102
1	1389367531	16.1352802813053	45.8111869768103	16.1352990567684	45.8111804333252	62	24	-391681	120
1	1389367537	16.1357992887497	45.812047905779	16.1358287930489	45.8120385580875	62	22	-402536	104
1	1389367543	16.1363169550896	45.8129060171756	16.1363370716572	45.8128994738926	61	20	-402536	103
1	1389367549	16.1367809772491	45.8137603763895	16.1368064582348	45.8137538332068	61	18	-402536	102
1	1389367555	16.1371886730194	45.8146240697559	16.1371994018555	45.8146212655784	59	16	-402535	101
1	1389367562	16.1375387012959	45.8156232827061	16.1375842988491	45.8156158050331	58	12	-402534	114
1	1389367569	16.137835085392	45.8166168695734	16.1378726363182	45.8166112614188	57	10	-402534	113

Slika 7: Prikaz načina zapisivanja ruta unutar tablice u bazi podataka

3.2. Delphi

Pri izradi rada korišteno je Delphi programsko okruženje pri pripremi podataka za relacijsku bazu podataka. Delphi je programski jezik opće namjene i softverski proizvod koji koristi Delphi dijalekt programskog jezika „Object Pascal" i pruža integrirano razvojno okruženje (Integrated Development Enviroment, IDE) za brzi razvoj aplikacija za desktop, mobilne, web i konzolne softvere. Delphi trenutno razvija i održava Embarcadero Technologies. Delphijevi kompajleri generiraju izvorni kod za Microsoft Windows, macOS, iOS, Android i Linux (x64).

Delphi uključuje uređivač koda, vizualni dizajner, integrirani debugger, komponentu za kontrolu izvornog koda i podršku za dodatke trećih strana. Uređivač koda ima Code Insight (dovršavanje koda), Error Insight (provjera pogrešaka u stvarnom vremenu) i refactoring. Dizajner vizualnih obrazaca ima opciju korištenja biblioteke vizualnih komponenti (Visual Component Library, VCL) za razvoj na Windows platformi ili okvir FireMonkey (FMX) za razvoj na više platformi. Podrška za bazu podataka ključna je značajka i pruža je FireDAC (komponente za pristup bazi podataka). Delphi je poznat po brzoj brzini kompilacije, izvornom kodu i produktivnosti programera.

Delphi je izvorno razvio Borland kao alat za brzi razvoj aplikacija za Windows kao nasljednika Turbo Pascala. Delphi je postojećem jeziku dodao potpuno objektno orijentirano programiranje, a jezik je narastao da podržava generičke, anonimne metode, zatvaranja i izvornu podršku za Component Object Model (COM), [6], [7], [8], [9].

3.3 Delphi aplikacija za upis u bazu podataka

Delphi je RAD (eng. Rapid Application Development) alat razvijen programskim jezikom pascal. Delphi aplikacija se koristio za izradu pomoćnog programa za unos ruta iz tekstualne datoteke u MSSQL bazu podataka.

Razvojni alat omogućuje brzo kreiranje aplikacija korištenjem vizualnih i nevizualnih komponenata. Na slici 8 je forma za učitavanje podataka na kojoj su vidljive nevizualne komponente koje služe za pristup bazi podataka (ADOConnection1, ADOTable1, ADOQurey1i Datasource1).

Slika 8: Delphi forma za učitavanje podataka u bazu

U tekstualnoj datoteci svaka linija podataka sadrži informacije o jednoj ruti. Svaka prijelomna točka rute je opisana s 9 atributa. Unos podataka je organiziran tako da se broj pročitane linije iz tekstualne datoteke u bazu upisuje kao identifikacijski broj rute, a svaka prijelomna točka je novi zapis u bazi podataka s istim identifikacijskim brojem.

Na slici 9 je prikazan dio koda koji parametrizira nevizualnu komponentu **ADOConnection1** za pristup MSSQL bazi podataka. Varijabla **conString** sadržava sve parametre potrebne za otvaranje veze. Varijabla **conString** sastavljena je od fiksnog dijela teksta i dijelova teksta preuzetih kroz polja za unos. Polje za unos **edit4** prima informaciju o korisniku baze podataka „User ID“. Polje za unos **edit5** prima informaciju o korisničkoj zaporci za pristup bazi podataka. Polje za unos **edit2** prima informaciju o nazivu baze podataka. Polje za unos **edit1** prima informaciju o poslužitelju baze podataka. Naredba **ADOConnection1.Active := False** zatvara prethodnu vezu. Naredba **ADOConnection1.ConnectionString := conString** predaje komponenti parametre za novu vezu. Naredba **ADOConnection1.Active := True** otvara novu vezu.

```
conString := 'Provider=SQLNCLI11.1;Integrated Security="";Persist Security Info=False;User ID='+trim(edit4.Text)+';Password ='
+trim(edit5.Text)+';Initial Catalog='+trim(edit2.text)+';Data Source='
+trim(edit1.text)+';Use Procedure for Prepare=1;Auto Translate=True;Packet Size=4096;Workstation ID=192.168.5.155;'+
'Initial File Name="";Use Encryption for Data=False;Trust Server Certificate=False;Application Intent=READWRITE;';
AdoConnection1.Connected := False;
ADOConnection1.ConnectionString := conString;
AdoConnection1.Connected := True;
```

Slika 9: Parametrizacija veze prema bazi podataka u sklopu Delphi aplikacije

Program provjerava da li postoji baza podataka u koju želimo unijeti podatke te ako ne postoji program je napravi što se vidi sa slike 10.

Varijabla **createstring** sadržava parametre za nevizualnu komponentu **ADOQuery1** i sastavljena je od fiksnog teksta i dijela informacije koja je preuzeta s polja za unos kroz vizualnu komponentu **edit2**. Naredba **ADOQuery1. Active := False** zatvara vezu komponente s bazom podataka. Naredba **ADOQuery1.SQL.Clear** briše sadržaj prethodnog SQL upita. Naredba **ADOQuery1.SQL.Add(createstring)** dodaje novi sadržaj SQL upita. Naredba **ADOQuery1.ExecSQL** izvršava SQL upit. Ako baza podataka već postoji prikazat će se poruka da baza već postoji, a ako ne postoji napravit će se baza podataka. Nakon toga u varijablu createstring upisuju se naredbe za korištenje baze podataka. Izvođenje SQL upita je identično kao i kod SQL upita za izradu baze podataka.

```
createstring := 'CREATE DATABASE '+trim(edit2.text)+'';
try
ADOQuery1.Active := False;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add(createstring);
ADOQuery1.ExecSQL;
except
showmessage('baza '+trim(edit2.text)+' već postoji' );
end;
createstring := ' USE '+trim(edit2.text)+' ';
ADOQuery1.Active := False;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add(createstring);
ADOQuery1.ExecSQL;
```

Slika 10: Pravljenje baze podataka u sklopu Delphi aplikacije

Sa slike 11 može se vidjeti da program provjerava da li postoji tablica u koju se želi upisati podatke te:

- ako postoji, obriše tablicu i ponovno je napravi,
- ako ne postoji, napravi novu tablicu.

Varijabla **createstring** sadržava parametre za nevizualnu komponentu **ADOQuery1** i sastavljena je od fiksnog teksta i dijela informacije koja je preuzeta s polja za unos kroz vizualnu komponentu **edit3**. Sadržaj varijable je SQL upit za brisanje tablice preuzete iz **edit3** polja. U slučaju da SQL upit završi s neuspjelim brisanjem tablice, prikazuje se poruka da tablica ne postoji. Nadalje se zatvara i otvara veza s bazom podataka da bi se osvježila informacija o postojanju tablice u bazi podataka. Nakon toga **createstring** varijabla se popunjava sadržajem potrebnim za izradu nove tablice koja će imati naziv preuzet iz **edit3** polja i u varijabli opisanu strukturu tablice. Struktura tablice je sljedeća:

- ID_route; cjelobrojna vrijednost (int); mora sadržavati vrijednost (NOT NULL)
- UTC; int; ne mora sadržavati vrijednost (NULL)
- MATCH_X; realni broj (float); NULL
- MATCH_Y; float; NULL
- ORG_X; float; NULL
- ORG_Y; float; NULL
- ORG_S; int; NULL
- ORG_H; int; NULL
- LINK_ID; int; NULL
- MATCH_DIST; float; NULL
- ON [PRIMARY]; dodjeljuje tablicu u shemu PRIMARY (shemu dbo)

Izvođenje SQL upita je identično kao i kod SQL upita za izradu baze podataka i izrade tablice.

```
try
createstring :=' DROP TABLE dbo.'+trim(edit3.text)+' ';
ADOQuery1.Active := False;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add(createstring);
ADOQuery1.ExecSQL;
ADOQuery1.Close;
except
showmessage('tabela '+trim(edit3.text)+' ne postoji' );
end;
AdoConnection1.Connected := False;
ADOConnection1.ConnectionString := conString;
AdoConnection1.Connected := True;
createstring :=' CREATE TABLE dbo.'+trim(edit3.text)+'( '+
' [ID_route] [int] NOT NULL,'+
' [UTC] [int] NULL,'+
' [MATCH_X] [float] NULL,'+
' [MATCH_Y] [float] NULL,'+
' [ORG_X] [float] NULL,'+
' [ORG_Y] [float] NULL,'+
' [ORG_S] [int] NULL,'+
' [ORG_H] [int] NULL,'+
' [LINK_ID] [int] NULL,'+
' [MATCH_DIST] [float] NULL '+
') ON [PRIMARY]';
ADOQuery1.Active := False;
ADOQuery1.SQL.Clear;
ADOQuery1.SQL.Add(createstring);
ADOQuery1.ExecSQL;
```

Slika 11: Pravljenje tablice unutar baze podataka u sklopu Delphi aplikacije

Nakon toga upisuje podatke na sljedeći način kao na slici 12. Nevizualna komponenta **ADOTable1** preuzima ime tablice iz polja **edit3**. Izvođenjem naredbe **ADOTable1.Active := true** tablica postaje aktivna i raspoloživa za sljedeće operacije. Cjelobrojna varijabla **I** se popunjava s vrijednošću 0 te će se nadalje koristiti kao brojač. Naredba **AssignFile** pridružuje tekstualnu datoteku u lokalnom direktoriju s nazivom preuzetog iz **edit6** polja. Naredba **Reset(F)** postavlja pokazivač na početak datoteke. Petlja **while not EOF(F)** obavlja operacije do završetka tekstualne datoteke. Naredba **Readln(F, slog)** čita jednu liniju iz datoteke u string varijablu **slog**. Petlja **while pos(';',slog)>0** se izvodi dokle varijabla „slog“ ima sadržaj. Petlja for **K := 1 to 9** dijeli varijablu slog na 9 dijelova koji su međusobno razdvojeni znakom „;“ pri čemu se varijabla **slog** svaki put smanjuje za obrađeni dio. Brojač **I** služi za označavanje ruta. Brojač **K** služi za označavanje sadržaja dijela zapisa rute. Svaki pročitani dio stringa se pretvara u potreban tip podatka za upis u tablicu u bazi podataka. Naredba **stringreplace** služi za prilagodbu stringa kako bi se string mogao pretvoriti u realni broj. Budući da svaki dio rute ima 9 atributa petlja puni varijable odgovarajućim sadržajem koji se nakon toga upisuje u tablicu pri čemu je sadržaj brojača **I** jednak identifikaciji rute. Naredba **ADOTable1.append** dodaje prazan zapis u tablicu. Naredba

ADOTable1.FieldByName('naziv polja').value := varijabla upisuje podatak u odgovarajuće polje tablice. Naredba **ADOTable1.Post** završava proces upisa u tablicu.

```
ADOTable1.TableName := trim(edit3.text);
AdoTable1.Active := true;
    I := 0;

    AssignFile(F,locdir+trim(edit6.Text));
    Reset(F);
    while not EOF(F) do begin
      I:= I+1;
      Readln(F,slog);
            while pos(';',slog)>0 do begin
        t_ID_ROUTE := I;
        for K := 1 to 9 do begin
          dio := copy(slog,1,pos(';',slog)-1);

          dio := stringreplace(dio,'.',',',[rfreplaceall]);
          slog := copy(slog,pos(';',slog)+1,length(slog));
          if length(dio) > 0 then begin
            if K=1 then t_UTC := strtoint(dio);
            if K=2 then t_MATCH_X := strtofloat(dio);
            if K=3 then t_MATCH_Y := strtofloat(dio);
            if K=4 then t_ORG_X := strtofloat(dio);
            if K=5 then t_ORG_Y := strtofloat(dio);
            if K=6 then t_ORG_S := strtoint(dio);
            if K=7 then t_ORG_H := strtoint(dio);
            if K=8 then t_LINK_ID := strtoint(dio);
            if K=9 then t_MATCH_DIST := strtofloat(dio);
          end;
        end;
        ADOTable1.append;
        ADOTable1.FieldByName('ID_ROUTE').value := t_ID_ROUTE;
        ADOTable1.FieldByName('UTC').value :=t_UTC;
        ADOTable1.FieldByName('MATCH_X').asFloat :=t_MATCH_X;
        ADOTable1.FieldByName('MATCH_Y').asFloat :=t_MATCH_Y;
        ADOTable1.FieldByName('ORG_X').asFloat :=t_ORG_X;
        ADOTable1.FieldByName('ORG_Y').asFloat :=t_ORG_Y;
        ADOTable1.FieldByName('ORG_S').value :=t_ORG_S;
        ADOTable1.FieldByName('ORG_H').value :=t_ORG_H;
        ADOTable1.FieldByName('LINK_ID').value :=t_LINK_ID;
        ADOTable1.FieldByName('MATCH_DIST').asFloat :=t_MATCH_DIST;
        ADOTable1.Post;
        end;
    end;
```

Slika 12: Upis podataka unutar baze podataka u sklopu Delphi aplikacije

4. Izrada web aplikacije sustava

U ovom poglavlju bit će opisani arhitektura sustava, Visual Studio razvojno okruženje, ASP.NET platforma te NuGet biblioteke. Na kraju će biti opisano korištenje i rad web aplikacije u sustavu.

4.1. Microsoft Visual Studio

Pri izradi ovog rada korištena je ASP.NET platforma u Visual Studiju. U ovom poglavlju će se analizirati i opisati neke mogućnosti Visual Studia koje su potrebne za izradu aplikacije. Visual Studio je integrirana razvojna okolina koja se može koristiti za uređivanje, debug i izradu kôda pri izradi aplikacija. ASP.NET platforma uključuje programski jezik C#, HTML jezik i CSS, [10].

Microsoft Visual Studio integrirano je Microsoftovo razvojno okruženje. Koristi se za razvoj računalnih programa, web aplikacija, web stranica, web usluga i mobilnih aplikacija. Visual Studio koristi Microsoftove platforme za razvoj softvera kao što su Windows API, Windows Forms, Windows Presentation Foundation, Windows Store i Microsoft Silverlight. Može proizvesti i izvorni i upravljani kôd. Visual Studio uključuje uređivač kôda koji podržava IntelliSense (komponenta za dovršavanje kôda), kao i refaktoriranje kôda. Integrirani program za ispravljanje pogrešaka radi i kao program za uklanjanje pogrešaka na razini izvora i kao stroj za uklanjanje pogrešaka. Ostali ugrađeni alati uključuju profiler kôda, dizajner za izgradnju GUI aplikacija, web dizajner, dizajner klasa i dizajner sheme baze podataka. Prihvaća dodatke koji proširuju funkcionalnost na gotovo svakoj razini - uključujući dodavanje podrške za sustave za kontrolu izvora (poput Subverzije i Gita) i dodavanje novih skupova alata poput urednika i vizualnih dizajnera za jezike specifične za domenu ili skupove alata za druge aspekte razvoja softvera (poput Azure DevOps klijenta: Team Explorer), [10].

Visual Studio podržava 36 različitih programskih jezika i omogućuje da uređivač kôda i program za ispravljanje pogrešaka podržavaju (u različitom stupnju) gotovo bilo koji programski jezik, pod uvjetom da postoji usluga specifična za taj jezik. Ugrađeni jezici uključuju C, C++, C++ /CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML i CSS. Podrška za druge jezike kao što su Python, Ruby, Node.js i M, između ostalog, dostupna je putem dodataka, [11].

4.2. ASP.NET

.NET platforma je razvojna platforma koja se sastoji od alata, programskih jezika, te biblioteka za izradu različitih tipova aplikacija. Dodatni okviri kao što su ASP.NET proširuju .NET s mogućnostima za izradu specifičnih tipova aplikacija. ASP.NET je mrežni okvir otvorenog kôda napravljen od strane Microsofta namijenjen za izradu web sučelja, [12].

C# je široko korišten programski jezik koji se koristi za izradu web usluga i web aplikacija kao što su ASP.NET aplikacije. U ovom diplomskom radu C# je korišten pri izradi funkcionalnosti sustava. HTML (eng. HyperText Markup Language) je standardni jezik za označavanje u dokumentima namijenjenim za web pretraživače. HTLM dokumenti pružaju strukturirani izgled web stranice u web browseru. Elementi HTLM-a su osnovni dijelovi svake HTML web stranice. CSS (eng. Cascading Style Sheets) opisuje kako će se HTML elementi prikazati krajnjem korisniku. CSS je stilski jezik temeljen na pravilima. Pravila se definiraju određivanjem grupa stilova koji će se primjenjivati na određenom HTML elementu ili na grupi HTML elemenata. U sustavu se koristi C# za funkcionalnost sustava, HTML i CSS za izgled i strukturu sustava.

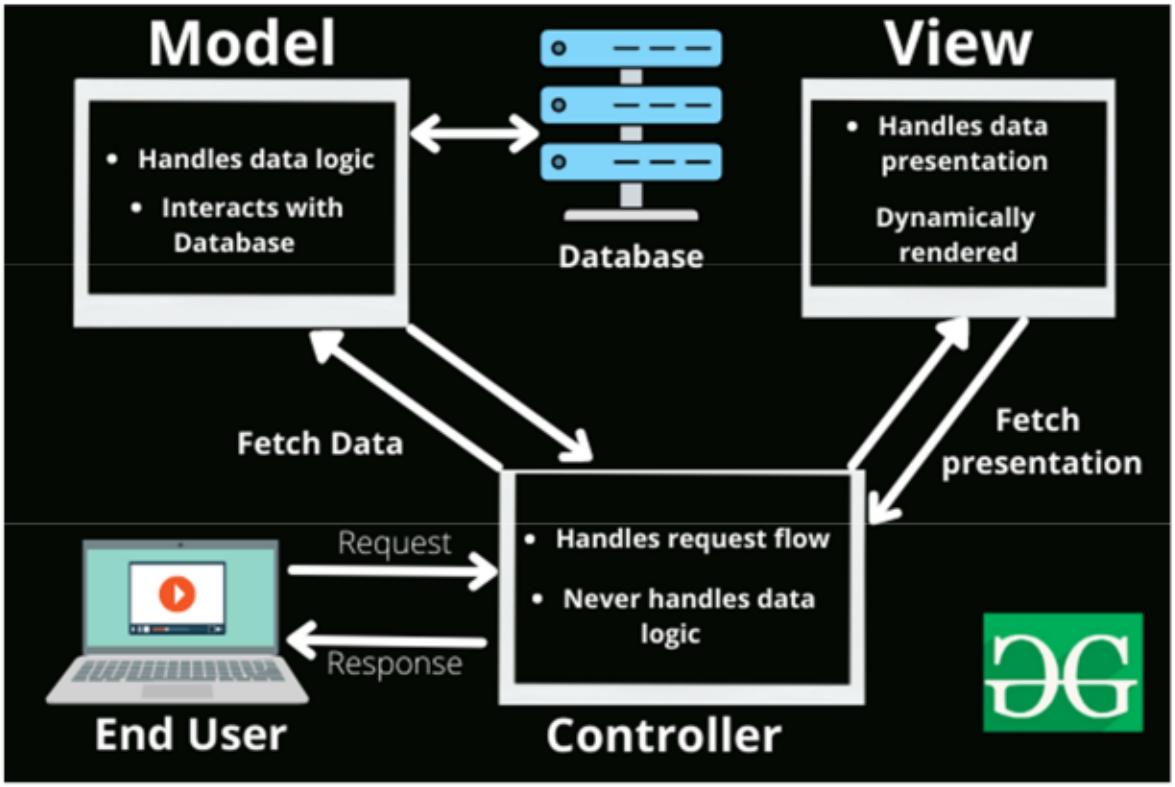
4.3. NuGet

NuGet je alat preko kojeg razvojni inženjeri (developeri) mogu stvarati, dijeliti i koristiti kôd koji im je potreban pri izradi aplikacije. Developeri koji naprave kôd koji žele podijeliti opišu njegovu funkcionalnost te ga pakiraju za dijeljenje u NuGet paketima. NuGet sustav sam rješava neposredne probleme između developera koji šalje paket i developera koji želi koristiti paket u svom kôdu. Nakon preuzimanja NuGet paketa i nakon instalacije paketa u Visual Studio okruženju, mogu se pozvati traženi paket i komponente koje su došle s paketom unutar kôda. Korišteni NuGet paketi su sljedeći:

- EntityFramework
- jQuery
- Microsoft.AspNet.Mvc
- Newtonsoft.Json

4.4. Arhitektura sustava

Web aplikaciju u ovom diplomskom radu rađena je unutar okruženja Model-View-Controller (MVC). je arhitektonsko i dizajnersko okruženje koje dijeli aplikaciju na tri glavne logičke komponente Model, View i Controller. Svaka komponenta MVC-a izgrađena je za rukovanje specifičnim elementima razvoja aplikacije. MVC arhitektura Izolira poslovnu logiku i prezentacijsku logiku jedno od drugog. MVC se koristio za desktop grafička korisnička sučelja (eng. Graphical User Interface, GUI), a danas je jedan od najčešće korištenih industrijski standardiziranih okvira za web razvoj za stvaranje skalabilnih i proširivih projekata. Također se koristi za izradu mobilnih aplikacija, [13]. Prikaz MVC arhitekture može se vidjeti iz slike 13.



Slika 13: Prikaz MVC arhitekture

Komponenta Model predstavlja cijelu logiku podataka s kojom korisnik radi. To može predstavljati podatke koji se prenose između komponenti View i Controller ili bilo koje druge podatke povezane s poslovnom logikom. Komponenta Model dohvaća podatke iz baze podataka odgovarajući na zahtjev kontrolera jer kontroler ne može sam komunicirati s bazom podataka. Model je u interakciji s bazom podataka i vraća potrebne podatke kontroleru.

Komponenta View koristi se za svu logiku korisničkog sučelja aplikacije. View služi za izradu korisničkog sučelja za korisnika. Prikazi se stvaraju pomoću podataka koji su prikupljeni preko komponente modela, ali te podatke ne uzima izravno, već ih prima preko komponente kontrolera.

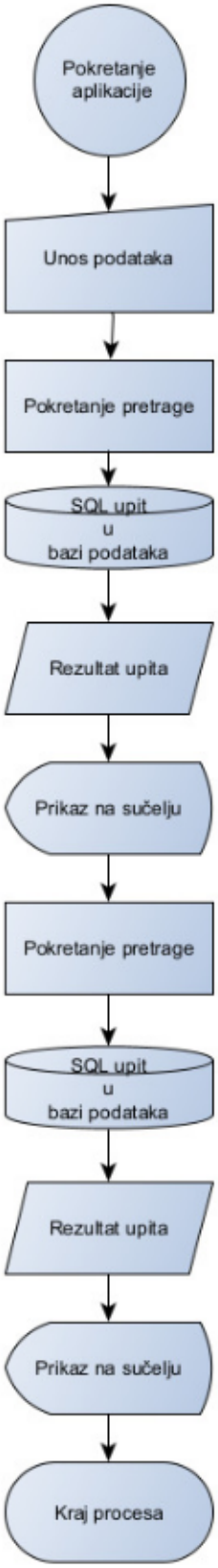
Kontroler je komponenta koja omogućuje međusobno povezivanje komponenti View i Model tako da djeluje kao posrednik. Komponenta Kontroler se ne mora brinuti o rukovanju logikom podataka, ona samo govori modelu što treba učiniti. Obrađuje svu poslovnu logiku i dolazne zahtjeve te manipulira podacima pomoću komponente modela i komunicira s prikazom kako bi se prikazalo konačno sučelje, [13].

4.5. Web aplikacija za prikaz podataka dobivenih iz GPS trajektorija praćenih vozila

U ovom potpoglavlju opisan je način rada i korištenja web aplikacije za prikaz podataka dobivenih iz GPS trajektorija praćenih vozila. Cilj ove aplikacije je prikazati rute te informacije o prikazanim rutama koje su dobivene iz GPS trajektorija praćenih vozila. U aplikaciji se bira prostor na karti unutar kojega se želi učitati rute. Zatim se bira ruta kroz padajući izbornik koja je prolazila kroz taj prostor te se ruta može učitati. Te na kraju se mogu zatražiti informacije koje opisuju tu odabranu rutu. Opis aplikacije će se provesti kroz dva dijela. Opisa frontend dijela aplikacija i opisa backend dijela aplikacije. Dio aplikacije koji korisnik vidi i koristi zove se frontend, a dio aplikacije koji se odvija u pozadini, kod poslužitelja se naziva backend. Prikaz sučelja aplikacije može se vidjeti iz slike 14.

Slika 14: Prikaz sučelja aplikacije

Prikaz toka aplikacije nalazi se na slici 15 u nastavku.



Pokretanje web aplikacije iz preglednika pozivom prema poslužitelju web aplikacije.

Unos podataka za izdvajanje ruta u određenom području na karti je omogućen upisom koordinata krajnjih točaka četverokuta ili iscrtavanjem četverokuta na karti korištenjem miša.

Pritiskom na tipku Load routes from selected segment aplikacija poziva metode koje izrađuju SQL upit s odgovarajućim parametrima.

Izvođenjem SQL upita u bazi podataka se dobije skup rezultata koji zadovoljavaju postavljene uvjete.

Rezultatom upita se popunjava padajući izbornik s identifikacijskim brojevima ruta koje su zadovoljavale postavljene uvjete.

Na sučelju se prikazuje broj ruta koje zadovoljavaju postavljene uvjete i vrijeme izvođenja operacija.

Pritiskom na tipku Load route aplikacija poziva metodu koja izrađuje SQL upit za prikupljanje podataka o izabranoj ruti.

Izvođenjem SQL upita u bazi podataka dobije se skup rezultata koji opisuje izabranu rutu.

Rezultatom upita predaje metodi koja iscrtava odabranu rutu.

Na sučelju se grafički prikazuje odabrana ruta. Duplim klikom miša na sučelju se prikažu informacije o odabranoj ruti.

Kraj.

Slika 15: Prikaz toka aplikacije

4.5.1. Backend sustav aplikacije

U ovom potpoglavlju bit će opisan backend dio sustava aplikacije. Aplikacija pristupa bazi podataka kroz globalnu varijablu. U aplikaciji je pridružena klasa global u kojoj je definirana veza na bazu podataka. U stringu DBCon su spremljeni parametri potrebni za povezivanje na bazu podataka koji će se dalje koristiti kroz aplikaciju. Skup parametra za uspostavljanje veze s bazom podataka se još zove connection string.

Connection string je skup znakova koji se koristi za povezivanje Visual Studio okruženja s bazom podataka. U connection stringu se nalaze informacije o lokaciji baze (192.168.5.155\SQLEXPRESS), nazivu baze podataka kojoj se pristupa (Initial Catalog=Baza_podataka_diplomski) te korisnički podaci za spajanje na nju (korisničko identifikacija i zaporka). Metoda **DBCon** služi za predaju i preuzimanje sadržaja varijable **_DBCon**. Klasa global i connection string mogu se vidjeti u nastavku.

```
    public class global
    {
        static string _DBCon = @"Data Source=192.168.5.155\SQLEXPRESS;Initial
Catalog=Baza_podataka_diplomski;User ID=Aleks;Password=andar";

        public static string DBCon
        {
            get
            {
                return _DBCon;
            }
            set
            {
                _DBCon = value;
            }
        }
```

Kako bi aplikacija mogla pristupiti bazi podataka aplikaciji se dodaju biblioteke koje sadrže potrebne funkcije. Uključivanje biblioteka može se vidjeti u nastavku.

```
using System.Data;
using System.Data.SqlClient;
```

Nadalje u aplikaciji se podacima iz baze podataka pristupa na sljedeći način.

```
parametri veze:

string connectionString = global.DBCon;

pravljenje instance veze:

using (SqlConnection sqlCon = new SqlConnection(connectionString))

otvaranje veze:
```

```
sqlCon.Open();
```

U nastavku se definiraju potrebne metode i varijable. U kontroleru MapController definirane su sljedeće klase.

Klasa WebResponseClick koja sadrži podatke koje ćemo vratiti u web preglednik prikazana je u nastavku. U klasi su deklarirani metoda WebResponseClick i public (javne) varijable koje će se naknadno koristiti za razmjenu podataka između frontend i backend komponenata aplikacije.

```
public class WebResponseClick
{
    public WebResponseClick()
    {
    }
    public Route route;
    public bool succ;
    public string errorDesc;
    public WebResponseClick wres;
    public string stringroutes;
    public string[] spl;
}
```

Klasa GPSA koja je definirana za spremanje jednog GPS podatka nalazi se u nastavku. Klasom je definirana struktura podataka koji opisuju jedan link trajektorije.

```
public class GPSA
{
    public int idRoute;
    public int utc;
    public double matchX;
    public double matchY;
    public double orgX;
    public double orgY;
    public int orgS;
    public int orgH;
    public int linkID;
    public double matchDist;

}
```

Klasa Route za spremanje jedne rute se nalazi u nastavku. U klasi je definiran model punjenja podataka o linkovima koji čine jednu rutu. To jest u klasi je ruta određena kao lista linkova.

```
public class Route
{
    public int id;
    public List<GPSA> podaci;
```

```
        public Route(int routeID)
        {
            podaci = new List<GPSA>();
            id = routeID;
        }

    }
```

Klasa **MapController** u kojoj smo definirali metode koje se pozivaju na web stranici se nalazi u nastavku. U klasi MapController su definirane metode **getRoute** i **getRouteIDs**.

```
public class MapController : Controller
```

U nastavku su definirane dvije metode:

- metoda koja dohvaća točno jednu rutu koja se zove **getRoute**

Metoda getRoute na početku otvara vezu s bazom podataka, klasom **WebResponseClick** i klasom **Route**. SQL upitom **"SELECT * FROM dbo.croutes where ID_route=@routeID"** se iz baze podataka izdvajaju podaci o ruti čiji je identifikacijski broj metodi predan parametrom **routeID.** Naredbom **command.Parameters.AddWithValue("@routeID",routeID)** se SQL upitu predaje vrijednost parametra **routeID**. Korištenjem komponente **SqlDataReader** se čitaju vrijednosti dobivene SQL upitom i upisuju se u novu instancu klase **GPSA.** Rezultat upita se predaje putem instance **wres WebResponseClick** klase (**return Content(JsonConvert .SerializeObject(wres), "application/json")**; .

```
public IActionResult getRoute(string routeID)
        {
            //dohvat podataka iz baze
            Console.WriteLine(routeID);
            string connectionString = global.DBCon;
            DataTable dtbl = new DataTable();
            int currentRouteInd = 0;
            Route currentRoute = new Route(currentRouteInd);
            WebResponseClick wres = new WebResponseClick();
            using (SqlConnection sqlCon = new SqlConnection(connectionString))
            {
                sqlCon.Open();
                SqlCommand command = new SqlCommand("SELECT * FROM dbo.croutes where
ID_route=@routeID", sqlCon);
                command.Parameters.AddWithValue("@routeID",routeID);
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        GPSA gps = new GPSA();
                        gps.utc = Convert.ToInt32(reader["UTC"]);
                        gps.matchX = Convert.ToDouble(reader["MATCH_X"]);
                        gps.matchY = Convert.ToDouble(reader["MATCH_Y"]);
                        gps.orgX = Convert.ToDouble(reader["ORG_X"]);
                        gps.orgY = Convert.ToDouble(reader["ORG_Y"]);
                        gps.orgS = Convert.ToInt32(reader["ORG_S"]);
```

```
                        gps.orgH = Convert.ToInt32(reader["ORG_H"]);
                        gps.linkID = Convert.ToInt32(reader["LINK_ID"]);
                        gps.matchDist = Convert.ToDouble(reader["MATCH_DIST"]);
                        currentRoute.podaci.Add(gps);
//                          Console.WriteLine(String.Format("{0}", reader["UTC"]));
                    }
                }

                sqlCon.Close();
                //                sqlDa.Fill(dtbl);
                wres.succ = true;
                wres.route = currentRoute;
            }


            return Content(JsonConvert.SerializeObject(wres), "application/json");

        }
```

- metoda koja dohvaća sve identifikatore ruta unutar određenih geografskih širina i dužina označenih na karti koja se zove **getRouteIDs.**

Metoda **getRouteIDs** preuzima parametre dviju točaka koje definiraju četverokut u kojem se traže rute. Ti parametri predstavljaju geografsku širinu i dužinu točaka. Na početku metoda ispituje odnos među točkama kako bi kasnije mogla napraviti pretragu unutar raspona od manjeg prema većem. Nakon toga metoda **getRouteIDs** otvara vezu s bazom podataka, klasom **WebResponseClick** i klasom **Route**. SQL upitom **("SELECT distinct ID_route FROM dbo.croutes where ((MATCH_X between @glx AND @ddx) AND (MATCH_Y between @gly AND @ddy)) order by ID_route")** se iz baze podataka izdvajaju identifikacijski brojevi ruta koje se nalaze unutar traženog raspona. Naredbom **command.AddWithValue()** se popunjava SQL upit parametrima koji su predati funkciji (**glx**, **gly**, **ddx**, **ddy**). Korištenjem komponente **SqlDataReader** se čitaju vrijednosti dobivene SQL upitom te se svaki **routeID** podatak dodaje u string **selRoutes1** koristeći ; kao delimiter. Nakon što su podaci pročitani puni se polje **spl** podacima iz stringa **selRoutes1** koji su razdvojeni ; delimiterom. Rezultat upita se predaje putem instance **wres WebResponseClick** klase (**return Content(JsonConvert .SerializeObject(wres), "application/json");** .

```
    public IActionResult getRouteIDs(double glx, double gly, double ddx, double
ddy)
        {
            //dohvat ruta koje zadovoljavaju kriterij iz baze

            string selRoutes1 = "";
            double manjix;
            double manjiy;
            double vecix;
            double veciy;
```

```
            if (glx > ddx)
            {
                manjix = ddx;
                vecix = glx;
            }
            else {
                manjix = glx;
                vecix = ddx;
            };
            if (gly > ddy)
            {
                manjiy = ddy;
                veciy = gly;
            }
            else
            {
                manjiy = gly;
                veciy = ddy;
            };
            string connectionString = global.DBCon;
            DataTable dtbl = new DataTable();
            int currentRouteInd = 0;
            Route currentRoute = new Route(currentRouteInd);
            WebResponseClick wres = new WebResponseClick();
            using (SqlConnection sqlCon = new SqlConnection(connectionString))
            {
                sqlCon.Open();
                SqlCommand command = new SqlCommand("SELECT distinct  ID_route FROM
dbo.croutes" +
                    " where ((MATCH_X between  @glx AND  @ddx) AND (MATCH_Y between
@gly AND  @ddy))" +
                    " order by ID_route", sqlCon);
                command.Parameters.AddWithValue("@glx", manjix);
                command.Parameters.AddWithValue("@gly", manjiy);
                command.Parameters.AddWithValue("@ddx", vecix);
                command.Parameters.AddWithValue("@ddy", veciy);

                using (SqlDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                    selRoutes1 = selRoutes1 + ";" +
Convert.ToString(reader["ID_route"]);


                    }
                }

                sqlCon.Close();
                Console.WriteLine(selRoutes1);
                wres.succ = true;
                string[] spl = selRoutes1.Split(';');
                wres.spl = spl;

            }


            return Content(JsonConvert.SerializeObject(wres), "application/json");}
```

4.5.2. Frontend sustav aplikacije

U ovom potpoglavlju biti će opisan frontend dio sustava aplikacije. Sučelje aplikacije je napravljeno korištenjem prezentacijskog jezika za pisanje web stranica (eng. HyperText Markup Language, HTML) i stilskog jezika CSS (eng. Cascade Style Sheets) te JavaScript funkcija.

Sučelje je vizualno dizajnirano korištenjem div-ova (pravokutnih elemenata koji definiraju poziciju dijelova HTML stranice). Svojstva div-ova su opisana unutar site.css datoteke.

U ovom poglavlju definirati će se sadržaj prikaza i rada aplikacije koji vidimo pri korištenja web preglednika. Za opis svojstava elemenata sučelja korišteni su sljedeći CSS atributi:

- border, određuje debljinu i boju linije okvira;
- position, određuje način smještaja unutar sučelja, to jest da li je pozicija elementa apsolutno ili relativna u odnosu na ostale elemente sučelja;
- left, udaljenost od lijevog ruba sučelja;
- right, udaljenost od desnog ruba sučelja;
- top, udaljenost od vrha sučelja;
- bottom, udaljenost od dna sučelja;
- height, visina elementa sučelja;
- background, pozadinska boja elementa;
- z-index, redoslijed pojavljivanja preklapajućih elemenata na ekranu;

Svakom elementu se mogu pridružiti stilski atributi direktno kod deklariranja elementa na sučelju.

Div u koji se smješta mapa.

```
<div id="divMap"></div>
```

Svojstva div-a **divMap** opisana su u **site.css** datoteci.

```
divMap {
    border: 2px solid #cccccc;
    position: absolute;
    left: 20%;
    right: 0%;
    top: 55px;
    bottom: 0%;
    height: 100%;
    background: #cccccc;
    z-index: 5;
}
```

Div za izbor rute koja se prikazuje se nalazi u nastavku. Unutar diva se nalaze div padajućeg izbornika **divSelectRoute** i tipka za prikaz koja nas vodi na metodu **loadSelectedRoute**.

```
 <div id="divOptions" style="background-color: transparent; font-family: serif;
font-weight: bold; color: dodgerblue;">
    <div class="form-inline" id="divSelectRoute" style="width: 100%; margin: 2px;
padding: 3px; height:30px">
    </div>
    @* Gumb koji kada kliknemo nas vodi na metodu loadSelectedRoute*@
    <button class="btn btn-primary" type="button"  + onclick="loadSelectedRoute()">
       Load route
    </button>
 </div>
```

Svojstva div-a **divOptions** opisana su u site.css datoteci.

```
#divOptions {
    border: 2px solid #cccccc;
    position: absolute;
    left: 0%;
    right: 80%;
    top: 50px;
    bottom: 0%;
    background: white;
    z-index: 6;
}
```

Div za prikaz koordinata odabranog četverokuta s mape prikazan je u nastavku. Div sadrži polja za unos krajnjih točaka četverokuta unutar kojega se želi izdvojiti rute. Polja su popraćena s elementima label koji opisuju namjenu polja. U div-u se također nalaze opcije korištene pri crtanju četverokuta te tipka za pokretanje metode **getDio().** Metoda **getDio** puni label elemente informacijama o podacima dobivenim unutar metode.

```
 <div  id="divDiomape" style=" margin: 2px; padding: 3px;background-color:
transparent; font-family: serif; font-weight: normal;">

    <label id="tlglx">Upper left X:</label><br>
    <input type="text" id="glx" name="glx"><br>
    <label id="tlgly">Upper left Y:</label><br>
    <input type="text" id="gly" name="gly"><br>
    <label id="tlddx">Lower right X:</label><br>
    <input type="text" id="ddx" name="ddx"><br>
    <label id="tlddy">Lower right Y:</label><br>
    <input type="text" id="ddy" name="ddy"><br><br>
    <form hidden ="HIDDEN" class="form-inline">
        <label>Geometry type  </label>
        <select id="type">
            <option value="Point">Point</option>
            <option value="LineString">LineString</option>
            <option value="Polygon">Polygon</option>
            <option value="Circle">Circle</option>
            <option value="Square">Square</option>
            <option value="Box">Box</option>
            <option value="None">None</option>
        </select>
```

```
    </form>
    <button class="btn btn-primary" type="button" onclick="getDio()">Load routes
from selected segment</button>
    <br><br><label id="nrsa">Number of rutes in selected segment: </label><label
id="nrsan" name="nrsan"> 0</label><br>
    <label id="ttake">Time taken: </label><label id="ttaken" name="ttaken">
0</label><br>
</div>
```

Svojstva div-a divDiomape opisana su u site.css datoteci.

```
#divDiomape {
    border: 2px solid #cccccc;
    position: absolute;
    left: 0px;
    right: 80%;
    top: 150px;
    bottom: 0%;
    background: white;
    z-index: 8;
}
```

Div za prikaz loga fakulteta opisan je u nastavku. Unutar div-a se nalazi jedan slikovni element.

```
 <div id="divLogo" style=" margin: 2px; padding: 0px;background-color: transparent;
font-family: serif; font-weight: normal;">

    <img src=".\fpz_icon.png" alt="FPZ" width="340" height="340">

</div>
```

Svojstva div-a divLogo opisana su u site.css datoteci.

```
#divLogo {
    border: 2px solid #cccccc;
    position: absolute;
    left: 0px;
    right: 80%;
    top: 550px;
    bottom: 0%;
    background: white;

    z-index: 7;
}
```

Funkcija stranice definirana je JavaScript kodom. U JavaScript kodu sve varijable koje su definirane izvan funkcija su public varijable, a sve varijable koje se nalaze unutar funkcija su lokalne varijable te njihov sadržaj nije vidljiv izvan funkcije. Definiranje Java script globalnih varijabli I parametara mape je u nastavku.

```
<script type="text/javascript">
    var oroute;
    var udaljenost =0;
    var pocetak = 0;
    var lonLat ;
    var coords ;
```

```
var prvak = 'True';
//
```

Za prikaz karte je korištena ol (OpenLayer) biblioteka:

<script src="~/js/ol.js"></script>

Svojstva za prikaz mape su u nastavku. U svojstvima su definirane instance slojeva izvora, rastera i vektora koji se naknadno koriste u prikazu na mapi.

```
var raster = new ol.layer.Tile({
    source: new ol.source.OSM()
});

var source = new ol.source.Vector({
    wrapX: false
});

var vector = new ol.layer.Vector({
    source: source,
    style: new ol.style.Style({
        fill: new ol.style.Fill({
            color: 'rgba(255, 255, 255, 0.2)'
        }),
        stroke: new ol.style.Stroke({
            color: '#ffcc33',
            width: 2
        }),
        image: new ol.style.Circle({
            radius: 7,
            fill: new ol.style.Fill({
                color: '#ffcc33'
            })
        })
    })
});
```

Učitavanje mape na web pregledniku napravljeno je na sljedeći način:

```
var map = new ol.Map({
    interactions: ol.interaction.defaults({ doubleClickZoom: false }),
```

Ucitavanje OSM karte i stavljanje u div element divMap

```
    target: 'divMap',
    layers: [ new ol.layer.Tile({
            source: new ol.source.OSM()
                   }), vector
    ],
```

Centar karte je postavljen na 15.981179, 45.805281 (otprilike Zagreb)

```
        view: new ol.View({
            center: ol.proj.fromLonLat([15.981179, 45.805281]),
            zoom: 12
        })
    });
```

Funkcija koja na svaki klik mape sprema koordinate u varijablu prikazana je u nastavku. Svaki put kada se mišem klikne na mapu funkcija preuzima koordinate točke klika. Izbor četverokuta se vrši s dva odvojena klika i zbog toga funkcija razlikuje prvi i drugi klik. Funkcija ujedno pročitane koordinate upisuje polja za unos **glx**, **gly**, **ddx** i **ddy**.

```
    map.on('click', function (event) {
        var point = map.getCoordinateFromPixel(event.pixel);
        lonLat = ol.proj.toLonLat(point);
        if (prvak=='True'){
            prvak = 'False';
            var sglx = document.getElementById("glx");
            var sgly = document.getElementById("gly");
            sglx.value = lonLat[0];
            sgly.value = lonLat[1];}
            else if (prvak =='False') {
            prvak = 'True';
            var sddx = document.getElementById("ddx");
            var sddy = document.getElementById("ddy");
            sddx.value = lonLat[0];
            sddy.value = lonLat[1];
        }

    });
```

Funkcija koja na dupli klik miša prikazuje odabrane informacije o prikazanoj ruti prikazana je u nastavku. Funkcija prikazuje ukupnu udaljenost koja je prijeđena u ruti te vrijeme početka rute. Podaci za prikaz udaljenosti i vremena početka rute preuzimaju se iz public varijabli **udaljenost** i **pocetak**. U funkciji se vrijeme početka rute prilagođava iz UTC standarda prema hrvatskom standardu pisanja datuma i vremena. Podaci se prikazuju preko **window.alert** funkcije.

```
    map.on('dblclick', function (event) {
         var ukupudaljenost = document.getElementById("daljina");
        ukupudaljenost.value = udaljenost;

        var datump = document.getElementById("date");
        var d = new Date(0);
        d.setUTCSeconds(pocetak);
        var year = d.getFullYear();
        var month = d.getMonth() + 1;
        var day = d.getDate();
        var hours = d.getHours();
        var minutes = d.getMinutes();
        var seconds = d.getSeconds();
        datump.value = day + "-" + month + "-" + year + " " + hours + ":" + minutes
+ ":" + seconds;
        var poruka;
        poruka = "vrijeme početka vožnje: " + day + "-" + month + "-" + year + " " +
hours + ":" +
        minutes + ":" + seconds + "\nprađena udaljenost: " + udaljenost + "m";
        window.alert(poruka);

    });
```

Funkcija koja predaje parametre za selekciju ruta iz baze koje prolaze kroz selektirano područje na karti prikazana je u nastavku. Funkcija iz HTML elemenata **glx**, **gly**, **ddx** i **ddy** preuzima informacije o koordinatama područja unutar kojega se traže rute. Funkcija priprema i predaje parametre za izvođenje backend funkcije **getRouteIDs**. Po uspješnom izvršavanju funkcije poziva se funkcija **kojerute**. U suprotnom prikazuje grešku.

```
function getDio(){

        var sglx = document.getElementById("glx");
        var sgly = document.getElementById("gly");
        var sddx = document.getElementById("ddx");
        var sddy = document.getElementById("ddy");
//        alert(sglx.value);
        $.ajax({
            type: "POST",
            url: "@Url.Action("getRouteIDs")",
            data: { "glx": sglx.value, "gly": sgly.value, "ddx": sddx.value, "ddy":
sddy.value },
            success: kojerute,//Funkcija koja se poziva ako je poziv bio uspješan
            failure: function (response) {
                //Ispiši grešku u malom prozorčiću
                alert(response.error);
            }
        });

        }
```

Funkcija koja padajući izbornik popunjava identifikacijskim brojevima odabranih ruta prikazana je u nastavku. Funkcija preuzima rezultat izvođenja funkcije **getRouteIDs** te popunjava HTML element **selectRoute** identifikacijskim brojevima ruta. Ujedno HTML element **nrsan** popunjava brojem izdvojenih ruta unutar označenog područja.

```
function kojerute(response) {
        var nrsad = document.getElementById("nrsan");

        var odgovor = response.spl;
        nrsad.innerHTML = odgovor.length-1;
        divSelectRoute = document.getElementById("divSelectRoute");
        divSelectRoute.innerHTML = "";
        str = '<div class="form-inline"> <label for= "selectRouteDivLine" class=
"labelLeft"> Select route:</label> <select name="selectRoute" id="selectRoute">';
        for (var i = 1; i < odgovor.length; i++) {
            var addStr = '<option value="' + odgovor[i] + '">' + odgovor[i] +
'</option>';
            str += addStr;
        }
        str += ' </select> </div>';

        divSelectRoute.innerHTML = str;
    }
```

Funkcija za iscrtavanje četverokuta za odabir ruta na karti prikazana je u nastavku. Funkcija nakon prvog klika miša počinje, praćenjem kretanja miša, iscrtavati četverokut do drugog klika miša. Nakon dva klika miša dopušta ponovno iscrtavanje četverokuta.

```
var draw; // global so we can remove it later
function addInteraction() {

            maxPoints = 2;
            geometryFunction = function (coordinates, geometry) {
                if (!geometry) {
                    geometry = new ol.geom.Polygon(null);
                }
                var start = coordinates[0];
                var end = coordinates[1];
                geometry.setCoordinates([
                    [start, [start[0], end[1]], end, [end[0], start[1]], start]
                ]);
                return geometry;
            };

        draw = new ol.interaction.Draw({
            source: source,
            type: 'Circle',
            geometryFunction: geometryFunction,
            maxPoints: maxPoints
        });
        draw.on('drawend', function (e) {

            //                mydd();
            coords = e.feature.getGeometry().getCoordinates();
            coords1 = e.feature.getGeometry().getExtent();
            var lonLats = ol.proj.toLonLat(coords[0]);
            var lonLats1 = ol.proj.transform(coords1, 'EPSG:3857', 'EPSG:4326')



        })

        map.addInteraction(draw);

}


//
/**
* Handle change event.
*/
type.onchange = function () {
    map.removeInteraction(draw);
    addInteraction();
};

addInteraction();
```

Funkcija za učitavanje rute prikazana je u nastavku. Na početku izvođenja funkcija briše već postojeće vektore kojima je iscrtana prethodna ruta. Funkcija dohvaća odabranu vrijednost iz padajućeg izbornika pod nazivom **selectRoute**. Funkcija poziva backend funkciju **getRoute** te po uspješnom izvođenju funkcije **getRoute**, pokreće izvođenje funkcije **succCallGetRoute**. Ako ne uspije javlja grešku.

```
    function loadSelectedRoute(){
         if (activePlotedRoute != undefined) {
            vectorSource.removeFeature(activePlotedRoute);
            vectorSource.removeFeature(activeIconStart);
            vectorSource.removeFeature(activeIconEnd);
            activePlotedRoute = undefined;
            activeIconStart=undefined;
            activeIconEnd=undefined;
        }
        //Dohvati odabranu vrijednost iz padajućeg izbornika pod nazivom selectRoute
        var selectedRouteID = $('#selectRoute').val();
        //Pozovi funkciju u backendu za dohvat podataka
        $.ajax({
            type: "POST",
            url: "@Url.Action("getRoute")",
            data: { "routeID": selectedRouteID },
            success: succCallGetRoute,//Funkcija koja se poziva ako je poziv bio
uspješan
            failure: function (response) {
                //Ispiši grešku u malom prozorčiću
                alert(response.error);
            }
        });
    }
```

Funkcija za iscrtavanje rute na karti prikazana je u nastavku. Funkcija preuzima podatke o ruti preko funkcije **getRoute**. Podaci su obliku višedimenzionalnog polja. **[x, 9]** pri čemu je **x** broj linkova rute, a **9** predstavlja atribute koji pripadaju svakom linku. Za iscrtavanje rute funkcija čita podatke **matchX** i **matchY**. U varijablu udaljenost se zbrajaju udaljenosti linkova. U varijablu pocetak se upisuje **UTC** vrijeme početka prvog linka radi naknadnog prikaza. Funkcija također provodi sve radnje potrebne za crtanje rute na OSM kartu.

```
function succCallGetRoute(response) {
    console.log(response);
    if (response.succ == false) {
        //Ispiši grešku u malom prozorčiću
        alert(response.errorDesc);
    }
    else {
        var route = response.route;

        //Dohvati točke u ruti
        var points = [];
        udaljenost =0;
        pocetak=0;
        for(var i=0;i<route.podaci.length;i++){
            var GPSpodatak=route.podaci[i];
            points.push([GPSpodatak.matchX, GPSpodatak.matchY]);
            // zbraja udaljenosti
            udaljenost = udaljenost + GPSpodatak.matchDist;
        }
        // vrijeme početka rute
        var GPSpodatak = route.podaci[0];
        pocetak = GPSpodatak.utc;
        //Polilinije za link
        var polyline = new ol.geom.LineString(points);
        //Pretvori nazad iz 4326 u 3857 koji korisiti OSM
        polyline.transform('EPSG:4326', 'EPSG:3857');

        //Napravi novu OSM ikonu/znacajku
        var iconFeature = new ol.Feature({
            geometry: polyline,
        });
        //Postavi stil
        iconFeature.setStyle(new ol.style.Style({
            stroke: new ol.style.Stroke({
                color: 'rgba(0,0,255,0.7)',
                width: 3
            })
        }));
        //Dodaj je u vektorski sloj
        vectorSource.addFeature(iconFeature);
        //Postavi da je trenutno iscrtan bas ta ikona rute,
        // tako da je kasnije mozes lako pobrisati
        activePlotedRoute=iconFeature;

        //Početni pin
        var start=points[0];
        activeIconStart= new ol.Feature({
        geometry: new ol.geom.Point(ol.proj.fromLonLat([start[0], start[1]]))
```

```
        });
        //Postavi stil
        activeIconStart.setStyle(new ol.style.Style({
                image: new ol.style.Icon({
                    anchor: [0.5, 46],
                    anchorXUnits: 'fraction',
                    anchorYUnits: 'pixels',
                    src: '@Url.Content("~/orangePin32x32.png")'
                })
        }));
        //Dodaj je u vektorski sloj
        vectorSource.addFeature(activeIconStart);

        ////Završni pin
        var end=points[points.length-1];
        activeIconEnd = new ol.Feature({
        geometry: new ol.geom.Point(ol.proj.fromLonLat([end[0], end[1]]))
        });
        //Postavi stil
        activeIconEnd.setStyle(
            new ol.style.Style({
                image: new ol.style.Icon({
                    anchor: [0.5, 46],
                    anchorXUnits: 'fraction',
                    anchorYUnits: 'pixels',
                    src: '@Url.Content("~/purplePin32x32.png")'
                })
        }));
        //Dodaj je u vektorski sloj
        vectorSource.addFeature(activeIconEnd);
    }
}
```

5. Analiza rezultata

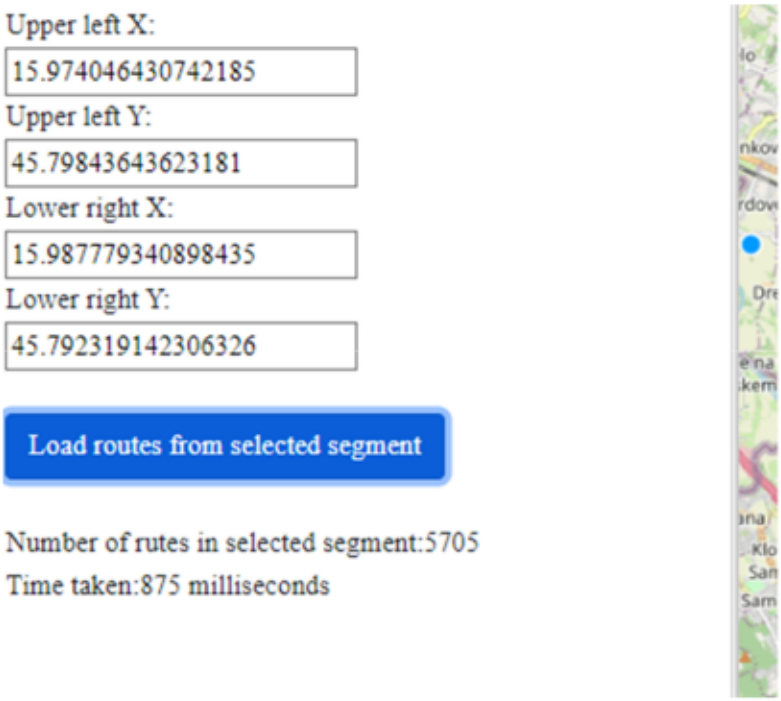
Da bi dobili dobar pregled rezultata rada i općenito uspješnosti aplikacije provesti će se više iteracija testova rada aplikacije. Testovi će sadržavati više različitih odabira ruta na karti, pregleda izbornika ruta, prikaza ruta i informacija o prikazanoj ruti. Redoslijed izvođenja operacija analiza je sljedeći:

- odabir područja u kojem se traže rute
- odabir učitavanja ruta unutar odabranog područja
- odabir učitavanja pojedine rute iz padajućeg izbornika
- učitavanje odabrane rute na karti
- prikaz informacija o odabranoj ruti na karti

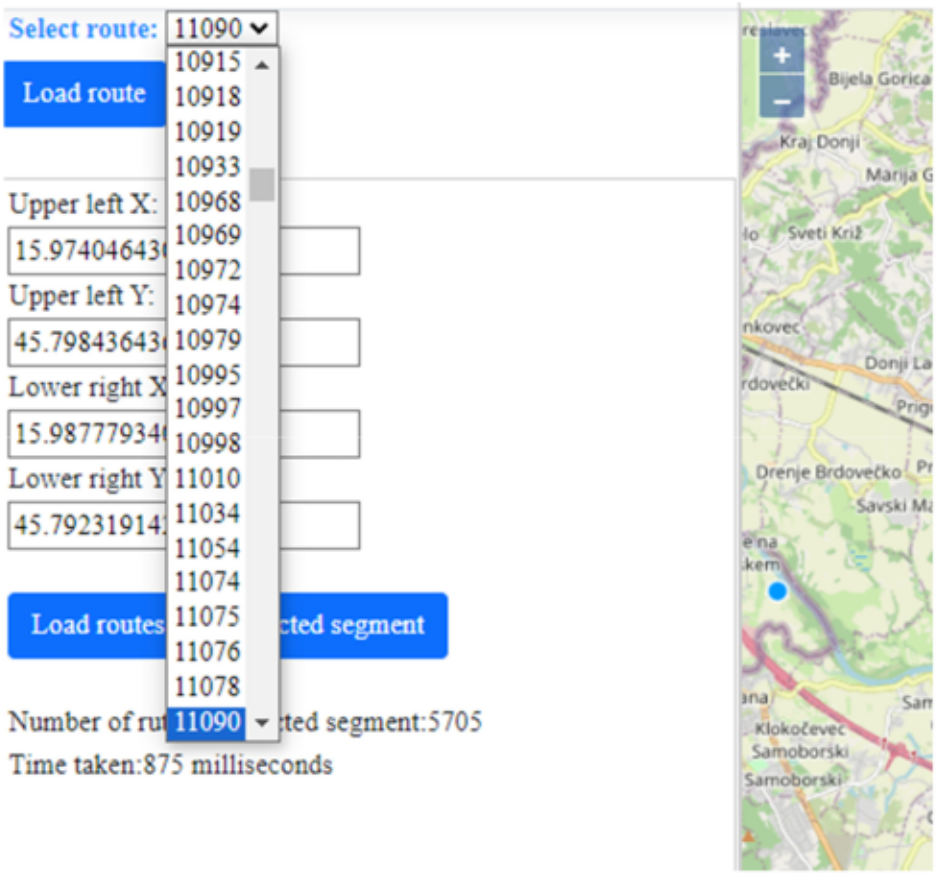
5.1. Analiza 1

Za prvo testiranje odabire se područje nad naseljem Trnje. Proces prve analize može se vidjeti na slikama 16, 17, 18, 19 i 20.

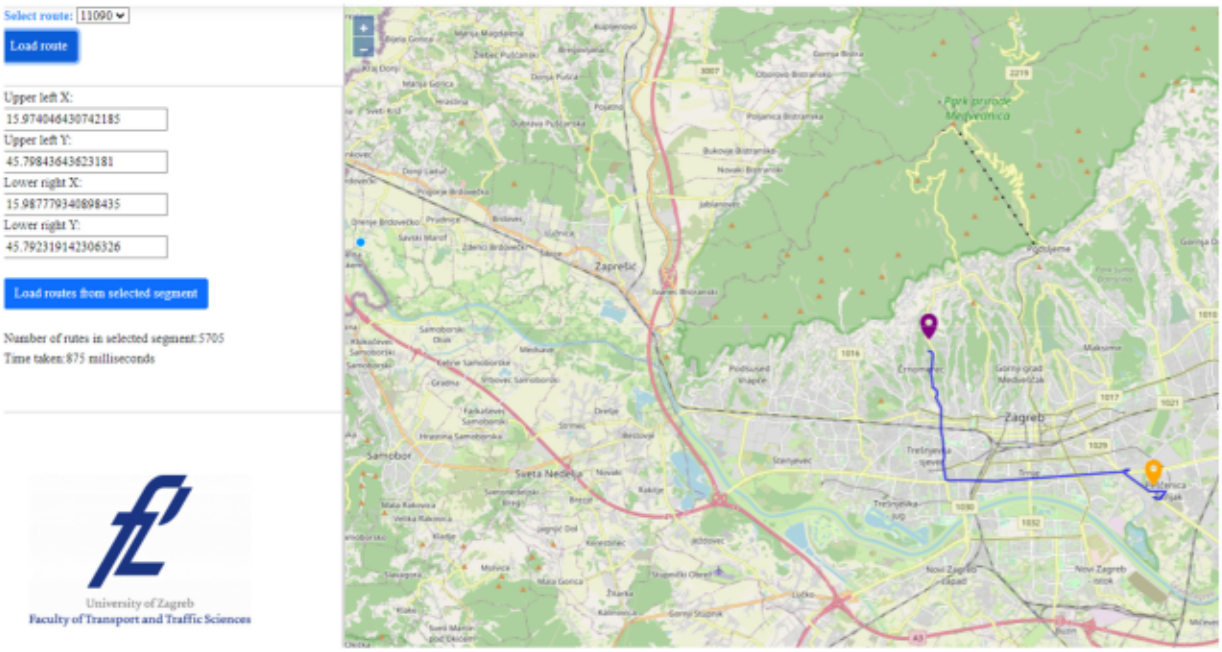
Slika 16: Odabir ruta u području (Analiza 1)



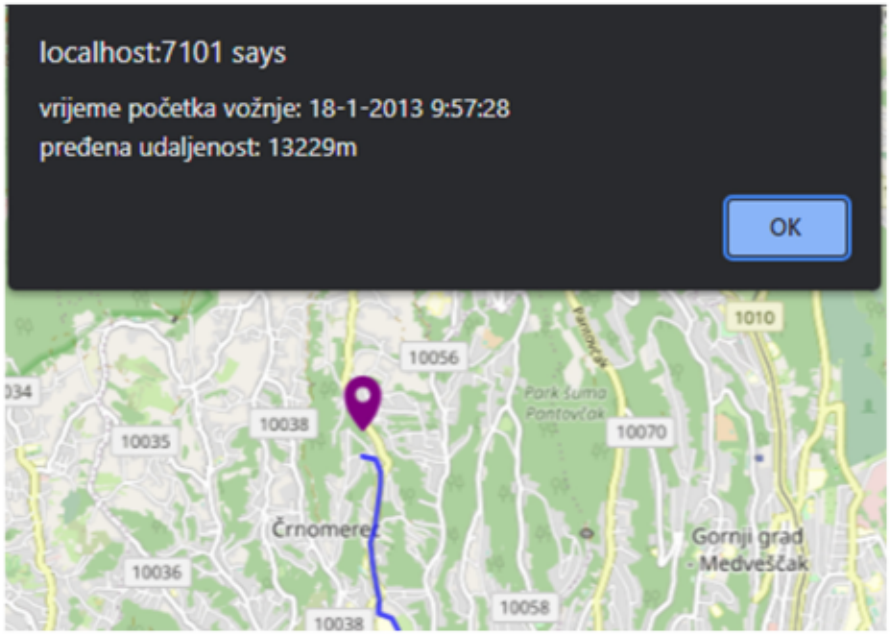
Slika 17: Učitavanje ruta iz pravokutnika u izbornik (Analiza 1)



Slika 18: Biranje rute iz izbornika (Analiza 1)



Slika 19: Crtanje odabrane rute (Analiza 1)



Slika 20: Prikaz informacija o ruti (Analiza 1)

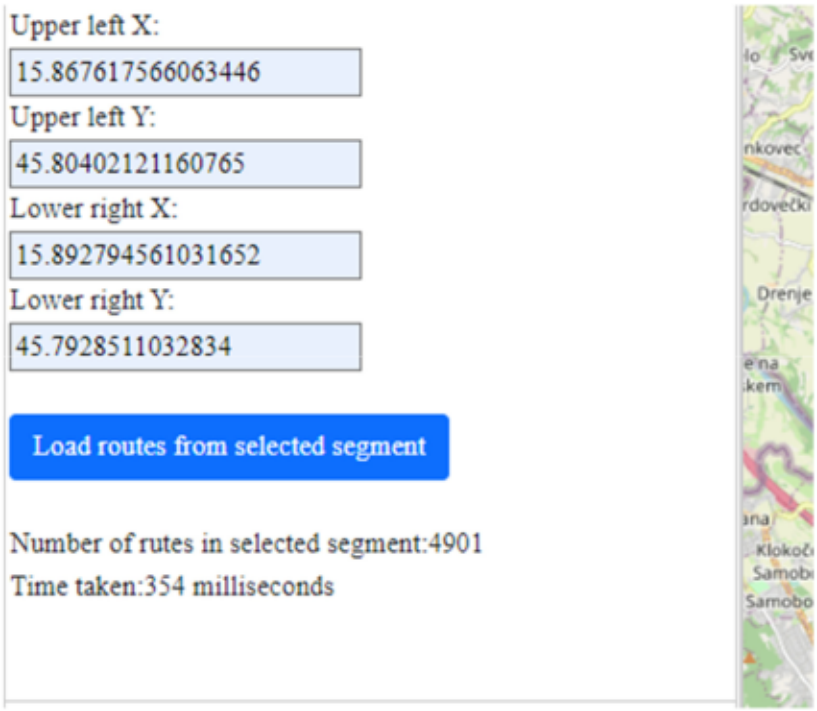
U prvom testiranju odabrano je područje omeđeno koordinatama X: 15.974 Y: 45.798 i X: 15.987 Y: 45.792 te su se učitale rute u tom području. U odabiru je učitano 5705 ruta te je trajanje učitavanja ruta iznosilo 875 milisekundi. Zatim smo odabrali rutu broj 11090 i prikazali je na karti. Dalje smo duplim klikom na karti dobili informacije o ruti koji nam pokazuju:

- prijeđenu udaljenost rute: 13229m
- datum početka vožnje : 18.1.2013.
- vrijeme početka vožnje: 9:57:28

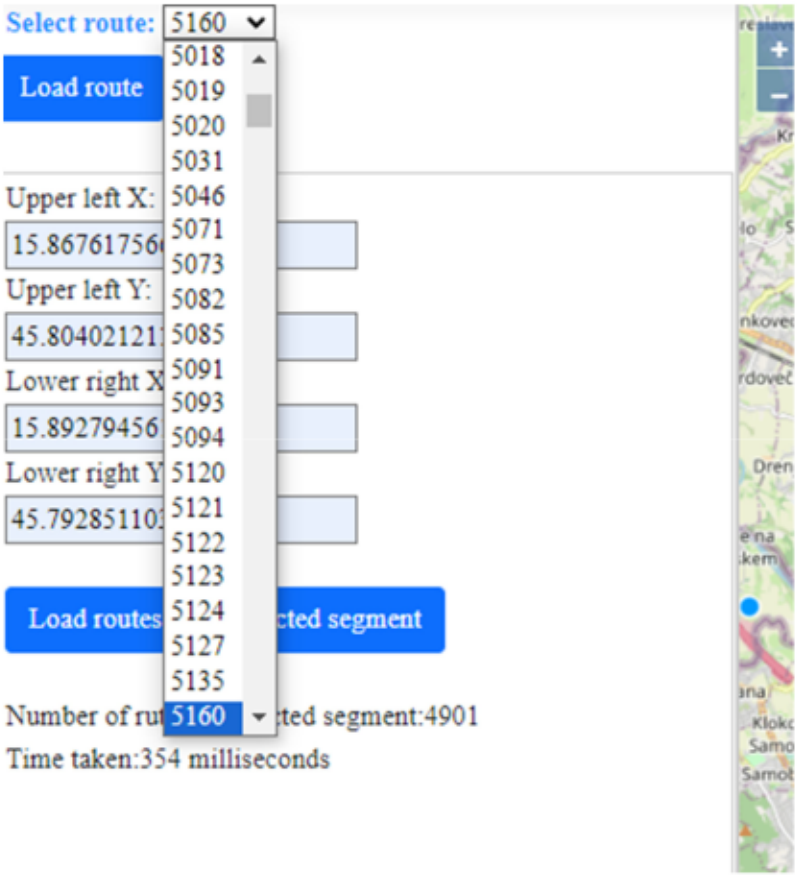
5.2. Analiza 2

Za drugo testiranje odabire se područje nad naseljem Stenjevec. Proces druge analize može se vidjeti na slikama 21, 22, 23, 24, 25.

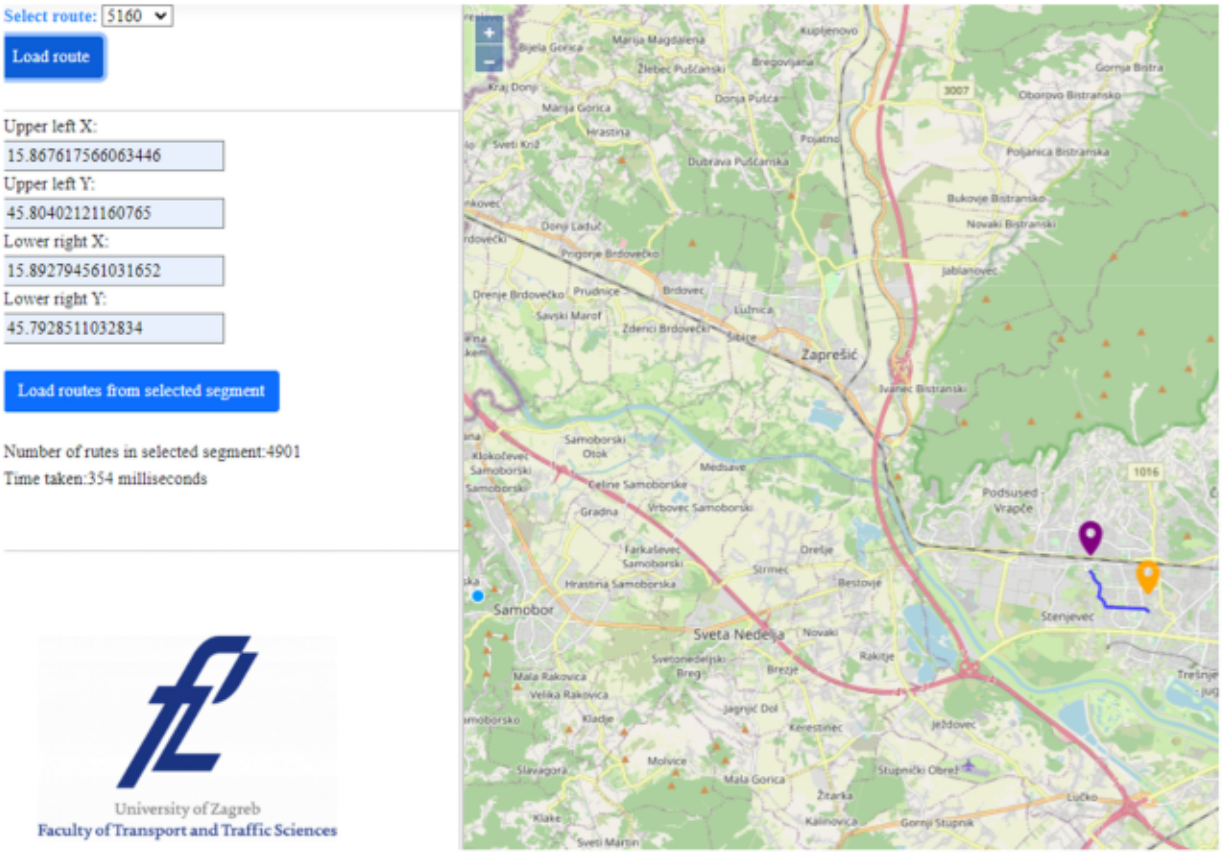
Slika 21: Odabir ruta u području (Analiza 2)



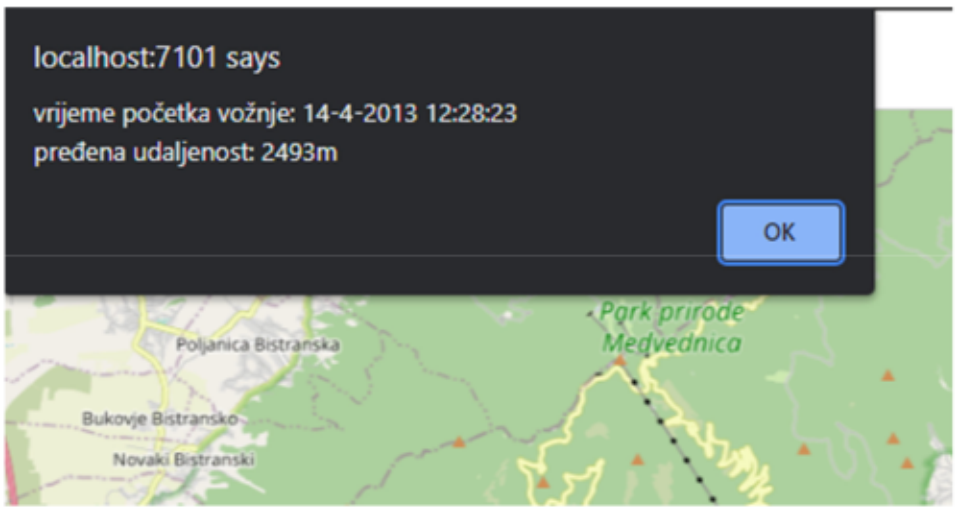
Slika 22: Učitavanje ruta iz pravokutnika u izbornik (Analiza 2)



Slika 23: Biranje rute iz izbornika (Analiza 2)



Slika 24: Crtanje odabrane rute (Analiza 2)



Slika 25: Prikaz informacija o ruti (Analiza 2)

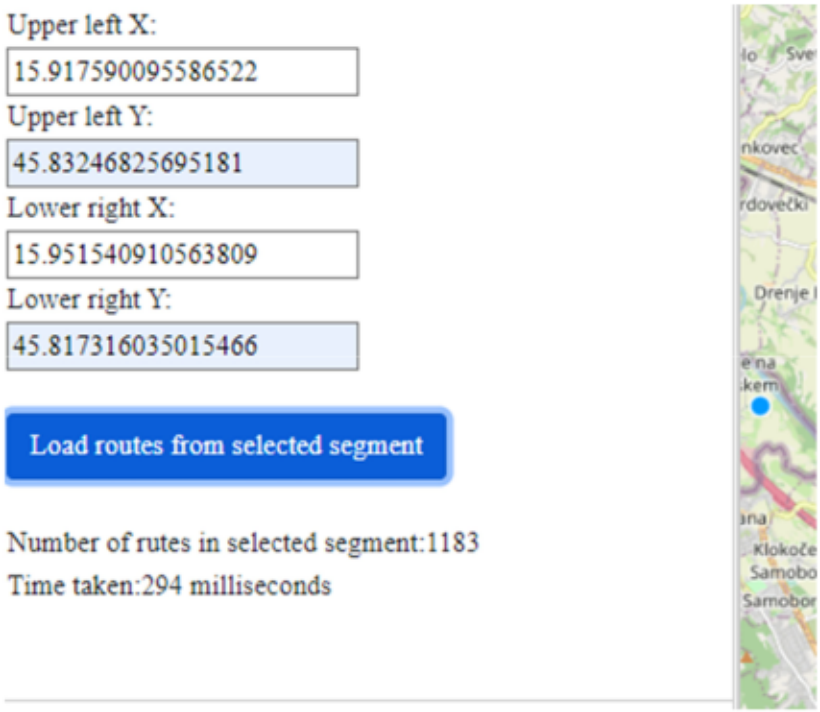
U drugom testiranju odabrano je područje omeđeno koordinatama X: 15.867 Y: 45.804 i X: 15.892 Y: 45.792 te učitali rute u tom području. U odabiru je učitano 4901 ruta te je trajanje učitavanja ruta iznosilo 354 milisekundi. Zatim smo odabrali rutu broj 5160 i prikazali je na karti. Dalje smo duplim klikom na karti dobili informacije o ruti koji nam pokazuju:

- pređenu udaljenost rute: 2493m
- datum početka vožnje : 14.4.2013.
- vrijeme početka vožnje: 12:28:23

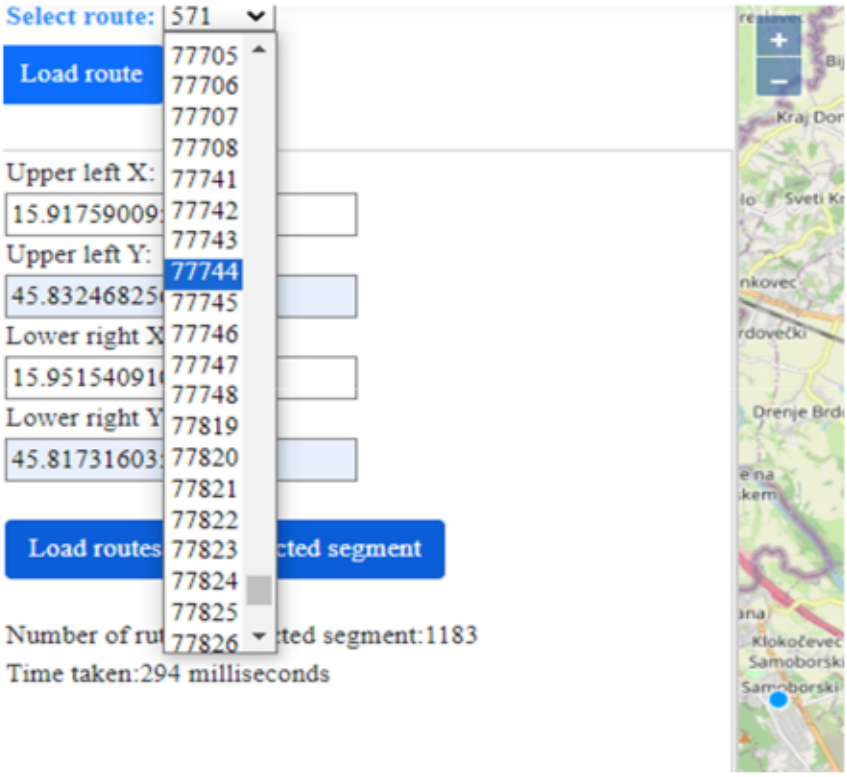
5.3. Analiza 3

Za treće testiranje odabire se područje nad naseljem Črnomerec. Proces treće analize može se vidjeti na slikama 26, 27, 28, 29 i 30.

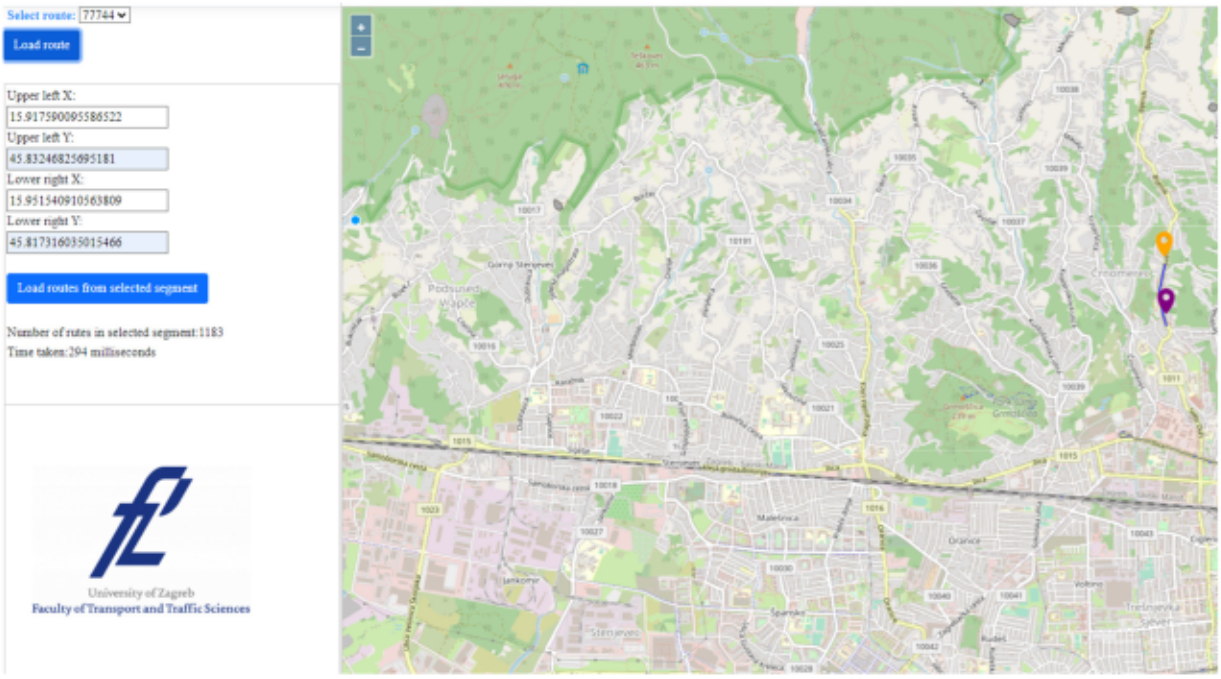
Slika 26: Odabir ruta u području (Analiza 3)



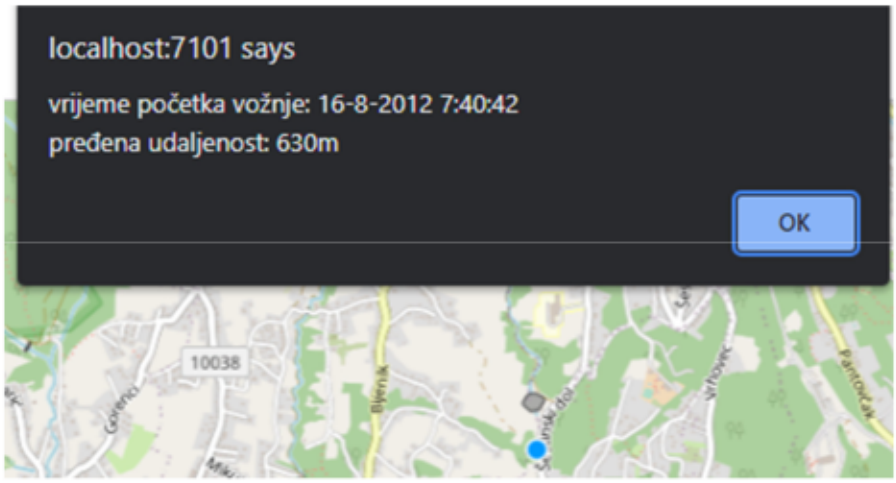
Slika 27: Učitavanje ruta iz pravokutnika u izbornik (Analiza 3)



Slika 28: Biranje rute iz izbornika (Analiza 3)



Slika 29: Crtanje odabrane rute (Analiza 3)



Slika 30: Prikaz informacija o ruti (Analiza 3)

U trećem testiranju odabrano je područje omeđeno koordinatama X: 15.917 Y: 45.832 i X: 15.951 Y: 45.817 te učitali rute u tom području. U odabiru su učitane 1183 rute te je trajanje učitavanja ruta iznosilo 294 milisekundi. Zatim smo odabrali rutu broj 77744 i prikazali je na karti. Dalje smo duplim klikom na karti dobili informacije o ruti koji nam pokazuju:

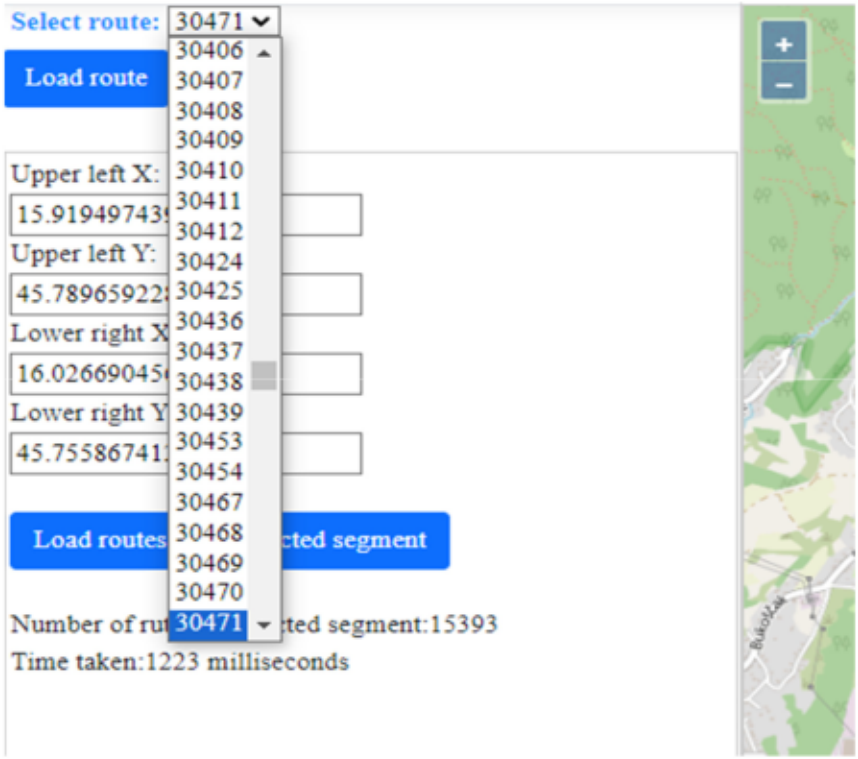
- pređenu udaljenost rute: 630m
- datum početka vožnje : 16.8.2012.
- vrijeme početka vožnje: 7:40:42

5.4. Analiza 4

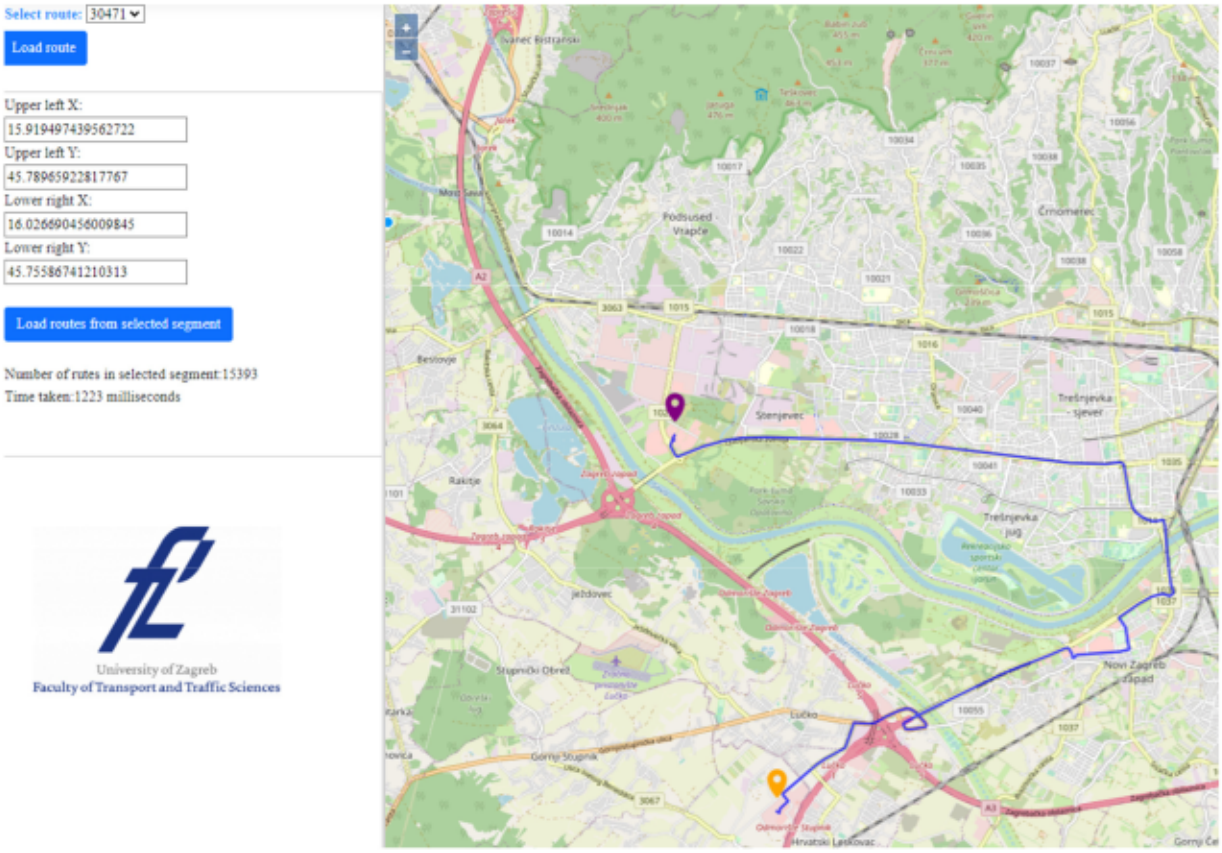
Za četvrto testiranje odabire se područje nad Novim Zagrebom. Proces četvrte analize može se vidjeti na slikama 31, 32, 33, 34 i 35.

Slika 31: Odabir ruta u području (Analiza 4)

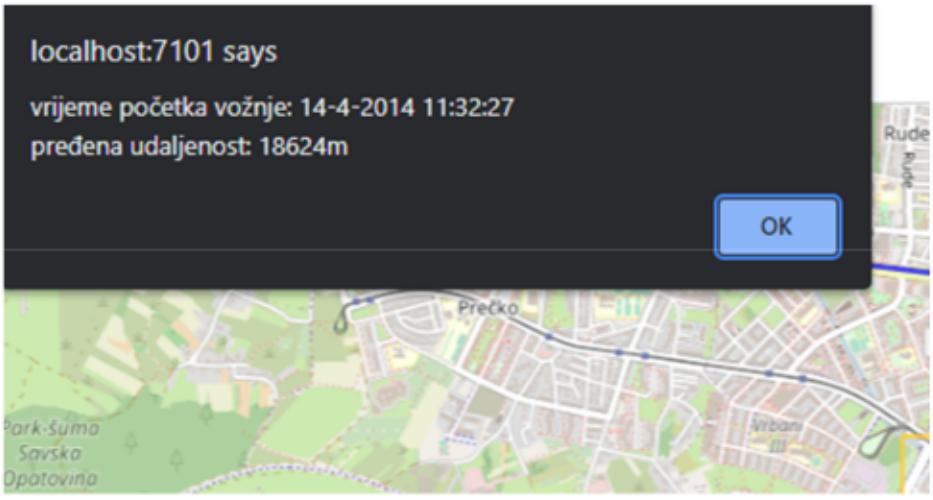
Slika 32: Učitavanje ruta iz pravokutnika u izbornik (Analiza 4)



Slika 33: Biranje rute iz izbornika (Analiza 4)



Slika 34: Crtanje odabrane rute (Analiza 4)



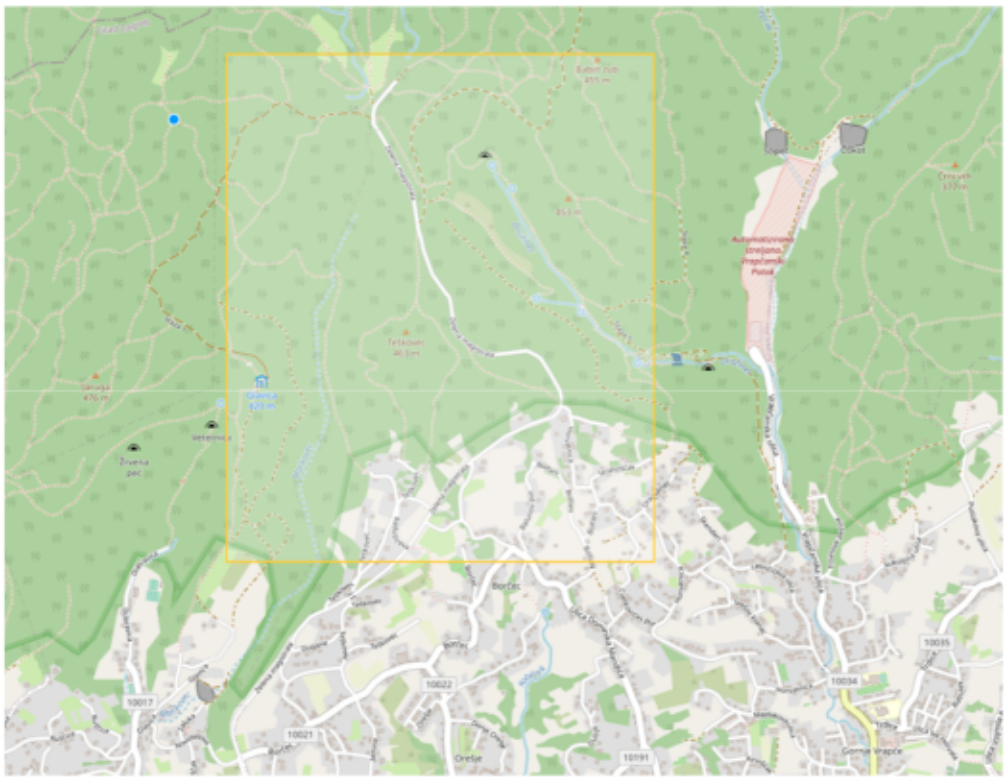
Slika 35: Prikaz informacija o ruti (Analiza 4)

U četvrtom testiranju odabrano je područje omeđeno koordinatama X: 15.919 Y: 45.789 i X: 16.026 Y: 45.755 te učitali rute u tom području. U odabiru su učitane 15393 rute te je trajanje učitavanja ruta iznosilo 1223 milisekundi. Zatim smo odabrali rutu broj 30471 i prikazali je na karti. Dalje smo duplim klikom na karti dobili informacije o ruti koji nam pokazuju:

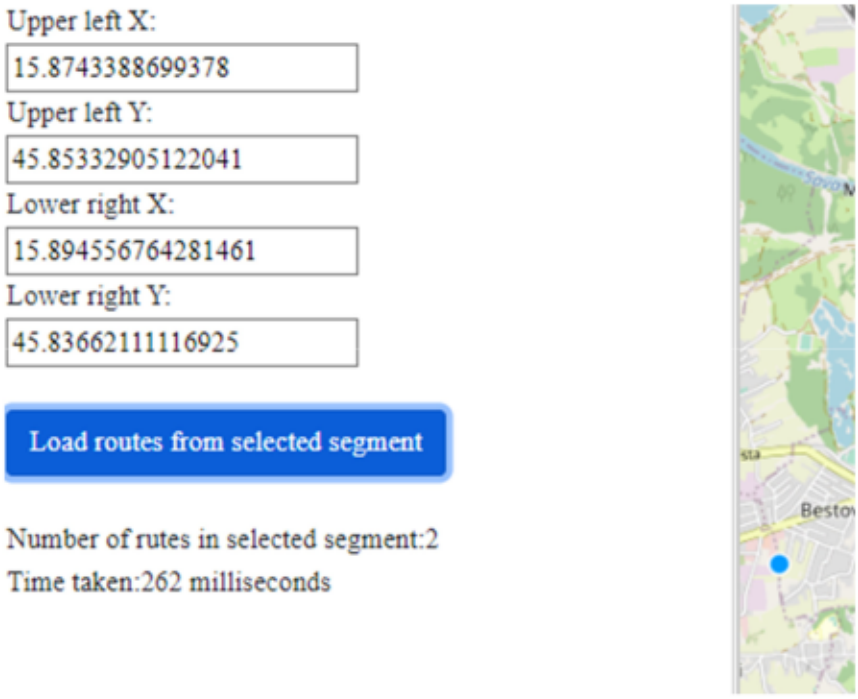
- pređenu udaljenost rute: 18624m
- datum početka vožnje : 14.4.2014.
- vrijeme početka vožnje: 11:32:27

5.5. Analiza 5

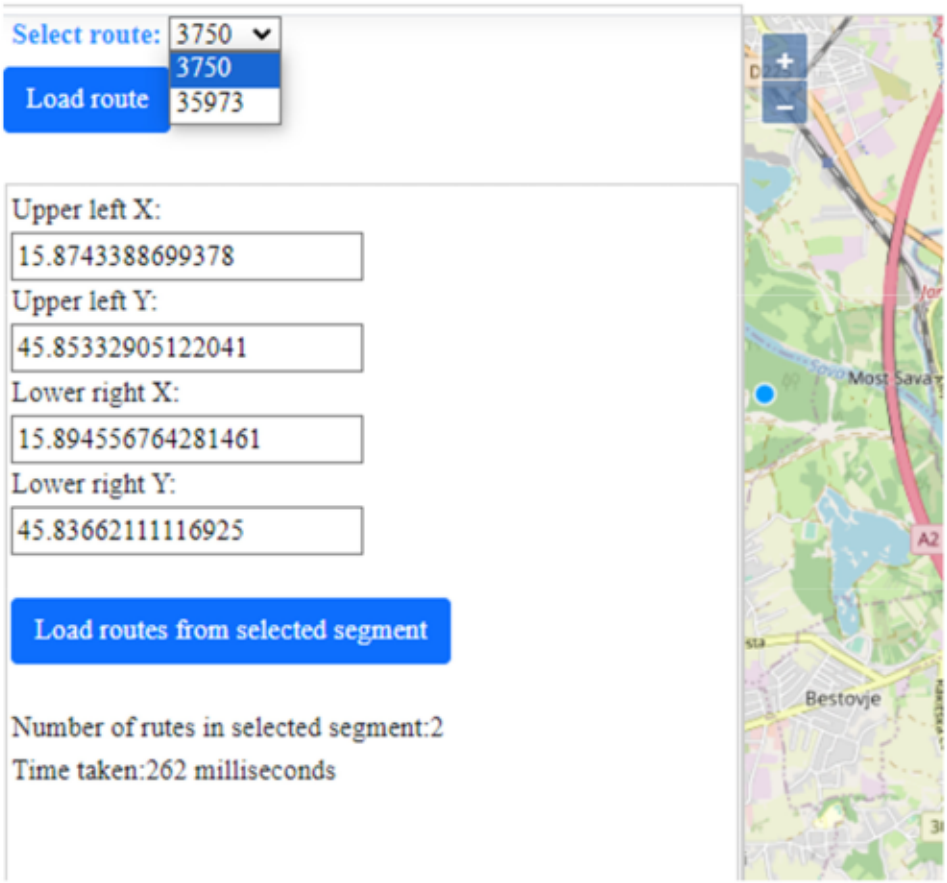
Za peto testiranje odabire se područje nad naseljem Gornjeg Vrapča i Medvednice. Proces pete analize može se vidjeti na slikama 36, 37, 38, 39 i 40.



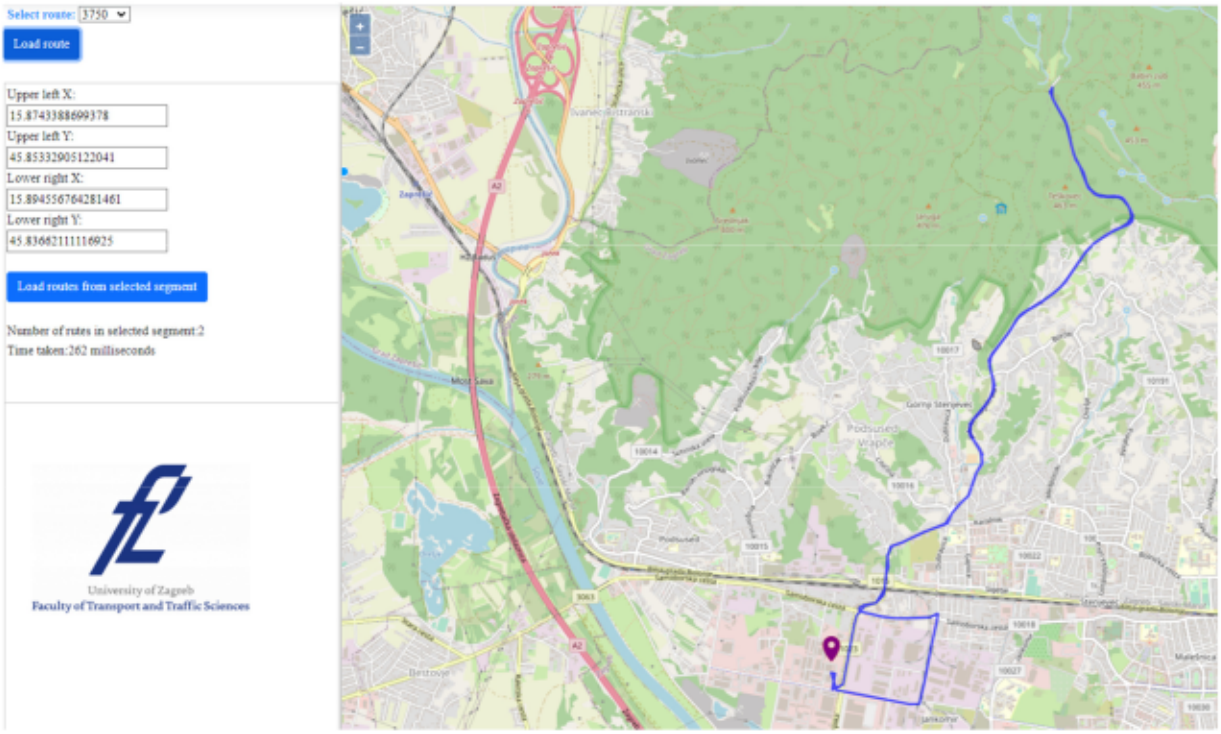
Slika 36: Odabir ruta u području (Analiza 5)



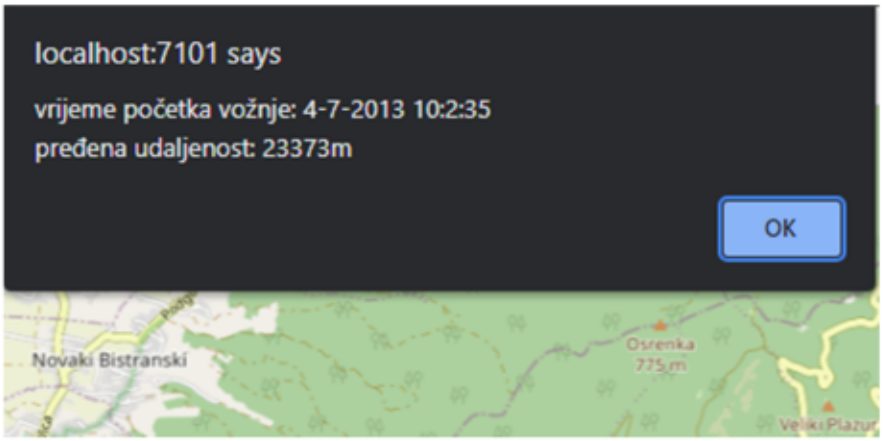
Slika 37: Učitavanje ruta iz pravokutnika u izbornik (Analiza 5)



Slika 38: Biranje rute iz izbornika (Analiza 5)



Slika 39: Crtanje odabrane rute (Analiza 5)



Slika 40: Prikaz informacija o ruti (Analiza 5)

U petom testiranju odabrano je područje omeđeno koordinatama X: 15.874 Y: 45.853 i X: 15.894 Y: 45.836 te učitali rute u tom području. U odabiru su učitane 2 rute te je trajanje učitavanja ruta iznosilo 262 milisekunde. Zatim smo odabrali rutu broj 3750 i prikazali je na karti. Dalje smo duplim klikom na karti dobili informacije o ruti koji nam pokazuju:

- pređenu udaljenost rute: 23373m
- datum početka vožnje : 4.7.2013.
- vrijeme početka vožnje: 10:02:35

6. Zaključak

U ovom radu opisani su prikupljeni GPS podaci o trajektorijama vozila na području poligona grada Zagreba. Dizajnirano je automatsko izrađivanje MS baze podataka kroz napravljenu Delphi aplikaciju. Ogromna količina podataka je obrađena i pripremljena za unos u bazu podataka. Obrađena skupina podataka se unijela u MS bazu podataka.

Izrađena baza podataka povezana je sa sučeljem Web aplikacije u kojoj se prikazuju GPS trajektorije praćenih vozila unutar definiranog poligona. Prikazane trajektorije su opisane s tri informacije, a to su datum početka kretanja, vrijeme početka kretanja i udaljenost koje je to vozilo prešlo. Aplikacija je zamišljena s ciljem pružanja krajnjem korisniku informacije o trajektorijama vozila koja želi pratiti.

Rezultat rada je potpuno funkcionalna web aplikacija za prikaz podataka dobivenih iz GPS trajektorija praćenih vozila. Korisnici većih flota vozila mogu lagano naći korist kroz ovu aplikaciju.

Postoji više dijelova ovog sustava na koje bi se moglo nadograditi različitim idejama ili potrebama krajnjeg korisnika. Sustav je napravljen tako da se na njega mogu dodavati razne ideje kojima bi postigli željeno funkcionalnost kroz mogućnost dodavanja dodatnih informacija o rutama ili dodavanjem dodatnih funkcionalnosti pri odabiru ruta.

Buduće istraživanje bi moglo uključivati obradu smjerova kretanja vozila pomoću umjetne inteligencije s ciljem pronalaska poveznica između početnih i krajnjih točaka trajektorija.

7. Popis literature

[1] Science Direct, »Estimating congestion zones and travel time indexes based on the floating car data,« [Mrežno]. Dostupno na: https://www.sciencedirect.com/science/article/pii/S0198971521000119#s0025. [Pristupljeno 6. 9. 2023.].

[2] OpenStreetMap Foundation, »OpenStreetMap,« [Mrežno]. Dostupno na: https://www.openstreetmap.org/about. [Pristupljeno 3. 9. 2023.].

[3] HOT OSM, »learnosm,« [Mrežno]. Dostupno na: https://learnosm.org/en/osm-data/data-overview/. [Pristupljeno 3. 9. 2023.].

[4] »Planet OSM,« [Mrežno]. Dostupno na: https://planet.openstreetmap.org/. [Pristupljeno 6. 9. 2023.].

[5] Oracle, »MySQL,« [Mrežno]. Dostupno na: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html. [Pristupljeno 10. 8. 2020.].

[6] W. Buchanan, Mastering Delphi Programming, Camden: Palgrave Macmillan, 2003.

[7] »Performance Comparison from Delphi 2010 to XE6 (Part 2),« [Mrežno]. Dostupno na: http://blogs.riversoftavg.com/index.php/2014/05/12/performance-comparison-from-delphi-2010-to-xe6-part-2/. [Pristupljeno 9. 3. 2016.].

[8] »The Delphi Geek: Built For Speed,« [Mrežno]. Dostupno na: https://www.thedelphigeek.com/2010/06/built-for-speed.html. [Pristupljeno 9. 3. 2016.].

[9] »Discussion on Hacker News about Delphi being alive,« [Mrežno]. Dostupno na: https://news.ycombinator.com/item?id=7614236. [Pristupljeno 9. 3. 2016.].

[10] »Microsoft,« [Mrežno]. Dostupno na: https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019. [Pristupljeno 10. 8. 2020.].

[11] »Microsoft,« [Mrežno]. Dostupno na: https://learn.microsoft.com/en-us/visualstudio/releases/2019/release-notes#16.7.0. [Pristupljeno 10. 8. 2020.].

[12] »Microsoft,« [Mrežno]. Dostupno na: https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet. [Pristupljeno 10. 8. 2020.].

[13] GeeksforGeeks, »GeeksforGeeks,« [Mrežno]. Dostupno na: https://www.geeksforgeeks.org/mvc-framework-introduction/. [Pristupljeno 6. 9. 2023.].

8. Popis slika

9. Popis tablica

Sveučilište u Zagrebu
Fakultet prometnih znanosti
Vukelićeva 4, 10000 Zagreb

IZJAVA O AKADEMSKOJ ČESTITOSTI I SUGLASNOSTI

Izjavljujem i svojim potpisom potvrđujem da je diplomski rad
(*vrsta rada*)
isključivo rezultat mojega vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju upotrijebljene bilješke i bibliografija. Izjavljujem da nijedan dio rada nije napisan na nedopušten način, odnosno da je prepisan iz necitiranog rada te da nijedan dio rada ne krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Svojim potpisom potvrđujem i dajem suglasnost za javnu objavu završnog/diplomskog rada pod naslovom Izrada web aplikacije za prikaz podataka dobivenih iz GPS trajektorija praćenih vozila , u Nacionalni repozitorij završnih i diplomskih radova ZIR.

Student/ica:

U Zagrebu, 07.09.2023

ALEKSANDAR KNEŽEVIĆ
(ime i prezime, *potpis*)