UNIVERZA V MARIBORU
FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO

Aljaž Soderžnik

RAZVOJ TEHNIČNIH SPECIFIKACIJ INFORMACIJSKEGA SISTEMA ZA VODENJE VOZIL

Diplomsko delo

Maribor, avgust 2018

UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO

Aljaž Soderžnik

RAZVOJ TEHNIČNIH SPECIFIKACIJ INFORMACIJSKEGA SISTEMA ZA VODENJE VOZIL

Diplomsko delo

Maribor, avgust 2018

RAZVOJ TEHNIČNIH SPECIFIKACIJ INFORMACIJSKEGA SISTEMA ZA VODENJE VOZIL

Diplomsko delo

Študent:	Aljaž Soderžnik
Študijski program:	Visokošolski-strokovni
	Računalništvo in informacijske tehnologije
Mentor:	izr. prof. dr. Domen Mongus

ZAHVALA

Želel bi se zahvaliti mentorju, izr. prof. dr. Domnu Mongusu, za strokovno pomoč, usmerjanje in potrpežljivost v času pisanja diplomske naloge.

Iskrena hvala vsem, ki ste mi na različne načine pomagali, da bi bilo pravljeno delo čim bolj kakovostno.

Na koncu bi se želel zahvaliti tudi svoji partnerki Anji, ki me je spodbujala pri študiji in izdelavi diplomske naloge.

Hvala!

Razvoj tehničnih specifikacij informacijskega sistema za vodenje vozil

Ključne besede: vodenje vozil, tehnična specifikacija, geografsko ograjevanje, spletna storitev

UDK: 004.777:621.396.969.3(043.2)

Povzetek

To diplomsko delo naslavlja tehnologije sledenja in vodenja vozil. V ta namen smo najprej preučili trenutno stanja tehnike na področju, čemur je sledil pregled obstoječih rešitev slovenskih podjetij, ki omogočajo tovrstne funkcionalnosti. Ugotovili smo, da obstoječe rešitve nudijo dobro podporo za ogled statusa vozila in njegovo vodenje od točke A do točke B, učinkovite informacijske rešitve za rabo parkirnih prostorov pa danes še ne obstajajo. To je zlasti očitno v kontekstu tovornega prometa, kjer so parkirišča izrazito prezasedena. Iz tega razloga smo zasnovali informacijsko rešitev, ki omogočala tudi tovrstne funkcionalnosti. Izdelali smo tudi tehnične specifikacije in implementirali spletno storitev, ki temelji na koncept geografskega ograjevanja v ogrodju .NET in omogoča spremljanje zasedenosti in vodenje rezervacij parkirišča.

Development of technical specifications for vehicle management information system

Key words: vehicle guidance, technical specifications, geofencing, web service

UDK: 004.777:621.396.969.3(043.2)

Abstract

This diploma thesis focuses on vehicle tracking and routing technologies. After initial analysis of the contemporary state-of-the-art, we carried out a detailed examination of commercial products offered by Slovenian industry. While most of the solutions allow for efficient monitoring of status of vehicles and their navigation from point A to B, there is a considerable lack of ICT solutions that would allow for efficient management of parking spaces. This aspect is particularly important in the context of generally overcrowded parking space. For this reason, we designed ICT solution that provides the key functionalities for parking space management and define its technical specifications. Implementation of parking-space booking and utilization monitoring was also done a form of .NET geofencing service.

KAZALO VSEBINE

1 UVOD 1

2 TEHNOLOGIJE ZA SLEDENJE IN VODENJE VOZIL 3

2.1 GPS 3

2.2 GLONASS 5

2.3 Struktura sporočila NMEA 6

2.4 Geografsko ograjevanje 8

3 PREGLED OBSTOJEČIH REŠITEV 10

3.1 EasyTracker 10

3.2 Track.si 12

3.3 KJE-SI 13

3.4 Primerjava rešitev in ugotovitve 14

4 PREDLAGANA REŠITEV 16

5 TEHNIČNE SPECIFIKACIJE 19

5.1 Namen aplikacije 19

5.2 Struktura aplikacije 19

5.3 Uporabniški vmesnik 20

5.4 Spletni vmesnik 20

5.5 Podatkovna baza 23

5.6 Delovanje aplikacije 24

6 IMPLEMENTACIJA KONCEPTA GEOGRAFSKEGA OGRAJEVANJA..... 26

6.1 Uporabljene tehnologije..... 26

6.2 Implementacija 28

6.3 Testiranje 35

7 SKLEP 41

VIRI IN LITERATURA..... 43

KAZALO SLIK

Slika 2.1: Primer uporabe navidezne geo-meje 8
Slika 3.1: Shema delovanja sistema Track.si 12
Slika 4.1: Primer območja geo-meje 16
Slika 4.2: Diagram poteka predlagane rešitve 18
Slika 5.1: Diagram strukture aplikacije 19
Slika 5.2: Hierarhija klicev metode in pomožnih metod 25
Slika 6.1: Ustvarjanje WCF projekta 28
Slika 6.2: Entitetno-relacijski diagram podatkovne baze 29
Slika 6.3: Vmesnik spletnega servisa 30
Slika 6.4: Spremenljivka geofenceDB 30
Slika 6.5: Metoda LogVehicleStatus 31
Slika 6.6: Metoda CheckGeofences 32
Slika 6.7: Metoda CheckState 33
Slika 6.8: Usmeritev vozila v namembno podjetje 34
Slika 6.9: Preusmeritev vozila na najbližje prosto parkirišče 34
Slika 6.10: Testna situacija 35
Slika 6.11: Poizvedba in odgovor 1. scenarija 36
Slika 6.12: Tabela VehicleLogs po 1. scenariju 37
Slika 6.13: Poizvedba in odgovor 2. scenarija 37
Slika 6.14: Tabela VehicleLogs po 2. scenariju 38
Slika 6.15: Poizvedba in odgovor 3. scenarija 38
Slika 6.16: Tabela VehicleLogs po 3. scenariju 39
Slika 6.17: Poizvedba in odgovor 4. scenarija 39
Slika 6.18: Tabela VehicleLogs po 4. scenariju 40

KAZALO TABEL

TABELA 2.1: PRIMERJAVA GPS IN GLONASS 5
TABELA 2.2: TIPI SPOROČIL NMEA 6
TABELA 2.3: PODROBEN PREGLED SPOROČILA NMEA $GPGGA 7
TABELA 3.1: PRIMERJAVA SLOVENSKIH SISTEMOV ZA VODENJE IN SPREMLJANJE VOZIL 15
TABELA 5.1: ZGRADBA TABELE COMPANY 23
TABELA 5.2: ZGRADBA TABELE GEOFENCE 23
TABELA 5.3: ZGRADBA TABELE PARKING 24
TABELA 5.4: ZGRADBA TABELE VEHICLELOGS 24

SEZNAM UPORABLJENIH KRATIC

API - Application Programming Interface

GLONASS – Globalnaja Navigacionnaja Sputnikovaja Sistema

GPS – Global Positioning System

IDE - Integrated development environment

IIS - Internet Information Services

NMEA – National Marine Electronics Association

RFID - Radio-frequency identification

SQL - tructured Query Language

WCF - Windows Communication Foundation

XML - Extensible Markup Language

1 UVOD

Slovensko avtocestno omrežje je obremenjeno s tranzicijskim tovornim prometom. Leta 2015 je odsek štajerske avtoceste (Trojane – Vransko) v povprečju dnevno prevozilo 5.467 vozil, ki so bila težja od sedmih ton. Na odseku primorske avtoceste (Unec – Postojna) je ta številka znašala v povprečju 6.778 tovornjakov na dan. Zasnova parkirišč in počivališč je večinoma že zelo stara, obseg tovornega prometa pa se je z zadnjem desetletju približno potrojil. Ob slovenskih avtocestah, ki so skupno dolge 650 kilometrov, so urejena parkirišča za približno 1.400 tovornjakov. Po vsakem izmed obeh obremenjenih krakov torej dnevno vozi približno petkrat več tovornjakov, kot je ob vsem avtocestnem križu zanje namenjenih parkirnih mest [11].

Tako v Sloveniji, kot tudi v Evropi se srečujemo s pomanjkanjem parkirnih prostorov za tovorna vozila, kar pride še posebej do izraza med vikendi, ko veljalo omejitve za vožnjo tovornih vozil. Dejstvo je, da počivališča ob slovenskih avtocestah niso bila projektirana in zgrajena za današnji obseg tovornega prometa. Leta 2003, zadnje leto pred vstopom Slovenije v Evropsko unijo, je cestninsko postajo Tepanje prevozilo dobrih 650.000 vozil, že leta 2009 pa jih je bilo za slab milijon več, kar posledično pomeni velik pritisk na počivališča [15].

Ob slovenskem avtocestnem križu je trenutno že nekaj več kot 60 počivališč, od tega 12 tako imenovanih majhnih počivališč brez bencinskega servisa, preostalih 51 pa je opremljenih z bencinskim servisom. Skupno je na počivališčih nekaj več kot 1.400 parkirnih mest za tovorna vozila. Uradno so sicer počivališča namenjena le kratkotrajnim postankom, njihova širitev pa zahteva daljši postopek sprejemanja ustreznih aktov na pristojnem ministrstvu [11].

Problematika parkiranja tovornih vozil se ne konča na avtocestah ampak se izraža tudi v urbanih okoljih, predvsem v manjših mestih. V slovenskih mestih, kjer se industrijska

območja prekrivajo s stanovanjskimi, lokalni deležniki zaznavajo problematiko velikega obsega tovornih vozil. Do težav prihaja zaradi pomanjkanja parkirnih prostorov za tovorna vozila pri posameznih podjetjih zato le-ta zasedajo javne prometne površine (parkirajo na cesti in pločniku) in povzročajo težave v prometu (zastoji, varnost). V kolikor v neposredni bližini podjetij ne najdejo primernega mesta, krožijo po območju dokler ne najdejo mesta za parkiranje. Problematika parkiranja tovornih vozil je izpostavljena tako s strani lokalnih deležnikov kot tudi podjetij, ki vršijo dostavo. Med samo izdelavo diplomskega dela smo se zato osredotočili na naslednje naloge:

- preučiti problem in pregledati obstoječe rešitve sledenja in vodenja vozil,
- na tej osnovi pripraviti idejno zasnovo informacijske rešitve.

Pri tem smo si zadali smo si naslednje cilje:

- predlagati informacijsko rešitev, ki bi reševala problematiko parkiranja vozil v območjih, ker se prepletajo industrijska in stanovanjska območja,
- pripraviti tehnično specifikacijo predlaganega informacijskega sistema,
- pripraviti nekatere delne rešitve, ki bi lahko bile uporabljene za vodenje tovornih vozil (izdelava konceptov in algoritmov).

V 2. poglavju tega diplomskega dela najprej predstavimo različne tehnologije za sledenje in vodenje vozil. V 3. poglavju sledi pregled obstoječih rešitev na tem področju, pri čemer primerjamo tri najbolj popularne produkte slovenskih proizvajalcev za sledenje in pregled vozil. V 4. poglavju smo predlagali lastno rešitev, ki bi jo lahko integrirali v obstoječe sisteme, da bi dobili celovit sistem.. V 5. poglavju smo za predlagano rešitev izdelali tehnično specifikacijo in jo v 6. poglavju tudi implementirali kot spletno storitev. Diplomsko nalogo pa zaključimo z izvedenimi testi.

2 TEHNOLOGIJE ZA SLEDENJE IN VODENJE VOZIL

Kot smo dejali že uvodoma, je eden izmed ciljev diplomskega dela zasnovati informacijsko rešitev, ki bo omogočala sledenje vozilom in jim glede na njihov položaj pošiljati ustrezna sporočila in obvestila o nadaljnji usmeritvi. V tem poglavju zato predstavljamo poznane in preverjene tehnologije, ki nam omogočajo določanje položaja in sledenje premikajočim objektom.

Glede na njihov domet jih ločimo na globalne in lokalne. V osnovi poznamo štiri globalne sisteme za določanje položaja, ki za določanje lege komunicirajo s sateliti (ameriški GPS, ruski GLONASS, evropski Galileo in kitajski BeiDou-2). Za določanje položaja v prostoru (npr. skladišča ali police v trgovini) pa uporabljamo tehnologije kot so radiofrekvenčna identifikacija (v nadaljevanju RFID) in različne oddajnike (najpogosteje Bluetooth). V nadaljevanju smo na kratko predstavili le globalni tehnologiji GPS in GLONASS, saj tehnologije za lokalno določanje položaja niso predmet tega diplomskega dela.

2.1 GPS

Globalni sistem pozicioniranja (angl. Global Positioning System, v nadaljevanju GPS) je storitev v lasti Združenih držav, ki uporabnikom zagotavlja storitve določanja položaja, navigacije in določanja časa [5].

Začetki sistema GPS segajo v zgodnja sedemdeseta leta prejšnjega stoletja, ko je Ameriško ministrstvo za obrambo želelo zagotoviti robusten in stabilen satelitski navigacijski sistem. Pri snovanju sistema so se precej opirali na zamisli in zasnove mornariških znanstvenikov. Leta 1978 je ministrstvo za obrambo zagnalo svoj prvi satelitski navigacijski sistem, sistem s 24 sateliti pa je postal popolnoma operativen v letu 1993 [10].

GPS se uporablja za podporo kopenskih, morskih in zračnih vozil, geodetskih in geofizičnih raziskav, v kartografiji, geodeziji, kmetijstvu in transportnih sistemih ter v številnih drugih aplikacijah. Ena izmed njegovih najpomembnejših, a najmanj cenjenih funkcij, je zagotovo globalna dostava natančnega in skupnega časa uporabnikom mobilne in fiksne telefonije [3].

Sam sistem je v osnovi razdeljen na tri segmente [5]:

- vesoljski,
- nadzorni in
- uporabniški segment.

Vesoljski segment

Vesoljski segment je sestavljen iz 31 satelitov, ki krožijo po šestih enako razmaknjenih krožnicah, na višini približno 20.200 km. Vsak izmed satelitov zemljo obkroži dvakrat dnevno. Združene države so zavezane k ohranjanju razpoložljivosti 24 operativnih satelitov vsaj 95 % časa. Takšna postavitev bi omogočala, da bi uporabniki imeli dostop do vsaj štirih satelitov iz katerekoli točke na zemlji, trenutna sestava (31 satelitov) pa še dodatno povečuje učinkovitost samega sistema. Sistem redno vzdržujejo in posodabljajo z novimi sateliti, trenutno so v uporabi sateliti štirih različnih generacij (od leta 1990 do 2016) [5].

Nadzorni segment.

Nadzorni segment navigacijskega sistema GPS sestavlja globalna mreža zemeljskih objektov, ki sledijo satelitom, spremljajo njihove prenose, jim pošiljajo ukaze in izvajajo analize. Nadzorni segment trenutno vključuje glavno nadzorno postajo, nadomestno nadzorno postajo, 11 ukaznih in nadzornih anten ter 16 nadzornih mest [5].

Uporabniški segment

Uporabniški segment je sestavljen iz sprejemnika GPS, ki sprejema signale satelitov in uporablja prenesene podatke za izračun uporabnikovega tridimenzionalnega položaja in časa [5].

2.2 GLONASS

Globalnaja Navigacionnaja Sputnikovaja Sistema, v nadaljevanju GLONASS, je ruska različica ameriškega sistema GPS. Razvoj sistema GLONASS je leta 1976 začela takratna Sovjetska zveza. Danes je sistem del ruske vesoljske agencije in je hkrati njen najdražji projekt; za vzdrževanje, razvoj in posodabljanje sistema namreč namenijo približno tretjino letnega proračuna. Sistem se je do razpada Sovjetske zveze primarno uporabljal predvsem za vremensko določanje položaja, merjenje hitrosti in časa. S kratko življenjsko dobo satelitov (3 leta) so le redki verjeli v uspešnost sistema. Leta 2001 pa je tedanji in sedanji ruski predsednik, Vladimir Putin, označil projekt kot prednostno nalogo svoje vlade, kar je vodilo k povečanju sredstev namenjenih projektu. Šest let kasneje, leta 2007 je izdal odlok predsednika Ruske federacije, s katerim je GLONASS odprl vrata za neomejeno javno uporabo. Ruska vlada bi naj za projekt porabila približno 5 milijard ameriških dolarjev (4.25 milijard evrov) do leta 2011, pričakujejo se dodatna vlaganja v višini 10 milijard dolarjev (8.5 milijard evrov) do leta 2020 [12].

V tabeli 2.1 je prikazana primerjava lastnosti obeh predstavljenih satelitskih sistemov.

Tabela 2.1: Primerjava GPS in GLONASS

Lastnost	GPS	GLONASS
Lastnik	Združene države	Ruska federacija
Število satelitov	31	vsaj 24
Število krožnic	6	4
Natančnost	3.5 – 7.8 m	5 – 10 m
Višina orbite	21150 km	19130 km
Perioda orbite	11 ur 58 minut	11 ur 16 minut

Poleg ameriškega sistema GPS in ruskega sistema GLONASS poznamo še evropskega – Galileo in kitajskega – BeiDou, ki komercialno nista tako razširjena in jo v diplomskem delu ne bomo podrobneje predstavili.

2.3 Struktura sporočila NMEA

National Marine Electronics Association (v nadaljevanju NMEA) je leta 1957 ustanovljeno združenje s strani trgovcev elektronike. Zveza je bila ustanovljena z namenom boljše komunikacije med trgovci in proizvajalci elektronike. Danes v svetu GPS-a je NMEA standarden podatkovni format, ki ga podpirajo vsi proizvajalci sprejemnikov GPS. NMEA je v navigaciji to, kar je ASCII v svetu digitalnega računalništva [4].

Namen strukture NMEA je uporabnikom omogočiti mešanje in usklajevanje strojne in programske opreme. Podatki formatirani v NMEA podatkovni strukturi omogočajo razvijalcem programske opreme, da napišejo programsko opremo za širok spekter GPS sprejemnikov, ne da bi morali pisati prilagojen vmesnik za vsak sprejemnik posebej. Brez standarda NMEA bi bilo pisanje programske opreme za GPS sprejemnike časovno potratno in finančno zelo drago [4].

Poleg osnovnega in hkrati najbolj popularnega tipa NMEA sporočila, $GPGGA, poznamo več alternativnih tipov NMEA sporočil (glej tabelo 2.2), kateri vsebujejo različne dodatne podatke [4].

Tabela 2.2: Tipi sporočil NMEA

Sporočilo	Opis
GGA	Čas, položaj in podatki povezani s popravki
GLL	Položaj in podatki o popravkih
GSA	Splošni podatki o satelitih
GSV	Podrobni podatki o satelitih
MSS	Informacije o statusu sprejemnika
RMC	Položaj, hitrost in čas
VTG	Vektorska tirnica in hitrost nad tlemi
ZDA	Datum in čas

V nadaljevanju si bomo pogledali primer sporočila NMEA $GPGGA in ga v tabeli 2.3 podrobno razdelali. Primer je povzet po [4].

Imamo naslednje sporočilo:

*$GPGGA,181908.00,3404.7041778,N,07044.3966270,W,4,13,1.00,495.144,M,29.200,M,1.0,0000*40*

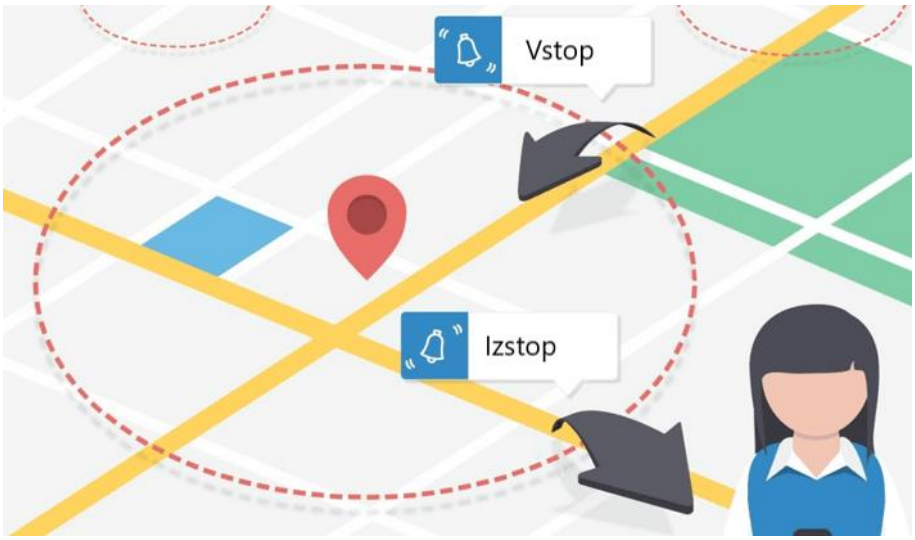
Tabela 2.3: Podroben pregled sporočila NMEA $GPGGA

Znak	Razlaga
$	Vsa NMEA sporočila se začnejo z znakom $, vsak posamezen podatek je ločen z vejico
GP	GP nam pove, da gre za GPS položaj (GL bi označeval GLONASS)
181908.00	Predstavlja časovni žig v času UTC
3404.7041778	Zemljepisna širina; število decimalnih mest je variabilno
N	N označuje, da gre za severno zemljepisno širino
07044.3966270	Zemljepisna dolžina; število decimalnih mest je variabilno
W	W označuje, da gre za zahodno zemljepisno dolžino
4	Kazalnik kakovosti natančnosti
13	Označuje število uporabljenih satelitov za določitev koordinate
1.0	Horizontalna razredčitev natančnosti
495.144	Označuje nadmorsko višino antene
M	Označuje enoto merjenja nadmorske višine (M – meter, F – čevelj)
29.200	Označuje geoidno ločitev
1.0	Označuje starost popravka (neobvezno)
0000	Označuje korekcijsko postajo (neobvezno)
***40**	Kontrolna cifra

2.4 Geografsko ograjevanje

Geografsko ograjevanje (angl. geo-fencing) je lokacijska storitev v kateri aplikacija ali druga programska oprema uporablja informacijo o lokaciji, da sproži vnaprej programiran ukaz, ko mobilna naprava ali oznaka RFID vstopi ali izstopi iz območja. Slednje je omejeno z virtualno pregrado, znano kot geo-meja (angl. geofence) [17].

Odvisno od konfiguracije geo-meje lahko le-ta sproži potisna obvestila, pošlje besedilno sporočilo oz. opozorilo, omogoči sledenje voznemu parku, onemogoči določeno tehnologijo ali daje podatke o trženju na podlagi položaja. Nekatere geo-meje so nastavljene za spremljanje dejavnosti na varovanih območjih, tako lahko odgovorne osebe spremljajo vsakogar, ki vstopi in zapusti določeno območje. Podjetja lahko geografsko ograjevanje uporabljajo tudi za spremljanje zaposlenih na terenu, avtomatizacijo vodenja delovnih ur in spremljanje premoženja podjetja [17]. Slika 2.1 prikazuje primer uporabe geo-meje.



Slika 2.1: Primer uporabe navidezne geo-meje

Da bi uporabili funkcionalnosti geografskega ograjevanja, mora skrbnik ali razvijalec v programski opremi, ki podpira GPS ali RFID, najprej vzpostaviti navidezno mejo okoli določene lokacije. To je lahko tako enostavno kot krog, ki je narisan 100 metrov okrog

lokacije v Google Zemljevidih, kot je določeno z uporabo API-jev pri razvoju aplikacije za mobilne naprave. Ta navidezna geo-meja bo sprožila odziv, ko pooblaščena naprava vstopi ali zapusti to območje [17].

Geografsko ograjevanje ni uporabno samo za mobilne aplikacije, pač pa se uporablja tudi za nadzor in sledenje vozil v ladijski industriji, živine v kmetijski industriji in brezpilotnih letalnikov. Skoraj vsak letalnik je predhodno programiran, da zna ugoditi geo-mejam, ki so po navadi postavljene na letališčih, prizorišč na prostem in drugih potencialno ogroženih območjih. [17].

Virtualne geo-meje so lahko aktivne ali pasivne. Aktivne geo-meje zahtevajo, da ima uporabnik omogočene lokacijske storitve in odprto mobilno aplikacijo. Pasivne geo-meje so vedno aktivne (delujejo v ozadju) in se zanašajo na Wi-Fi in mobilne podatke namesto na GPS [13].

Primeri uporabe geo-mej [13]:

- **Upravljanje flote:** Geo-meja lahko opozori dispečerja, ko voznik tovornjaka skrene z načrtovane poti.
- **Upravljanje s človeškimi viri:** Pametna kartica zaposlenega pošlje varnostno opozorilo, če zaposleni poskuša vstopiti v nepooblaščeno geo-mejno območje.
- **Upravljanje premoženja:** Skrbnik lahko nastavi opozorila, tako da se sproži alarm, ko elektronska naprava (mobilni telefon, tablični računalnik,…) zapusti območje

3 PREGLED OBSTOJEČIH REŠITEV

V tem poglavju smo na spletu poiskali in pregledali nekaj obstoječih rešitev, ki obravnavajo našo tematiko. Pogledali smo njihove ključne funkcionalnosti, glavne prednosti ter morebitne slabosti. Ker so takšni sistemi zasnovani z mislijo zadovoljiti čim več različnih naročnikov oz. uporabnikov, smo se pri funkcionalnostih rešitve omejili le na tiste funkcionalnosti, ki so ključne za sledenje in vodenje vozil. Tako nas niso zanimali npr. sistemi za analizo, ustvarjanje poročil o prevoženi poti in poročila o delu. Pri pregledu obstoječih rešitev smo se omejili na rešitve slovenskih proizvajalcev.

Pregled obstoječih rešitev je ključnega pomena za naslednja poglavja, ko smo načrtovali in implementirali delne koncepte lastne informacijske rešitve za vodenje in spremljanje tovornih vozil. S tem smo dobili vpogled v ključne funkcionalnosti, ki bi jih podobna rešitev naj zajemala.

3.1 EasyTracker

EasyTracker je rešitev domžalskega podjetja Bent Excellent d.o.o. in je bil v osnovi razvit za potrebe njihovega podjetja s prvotno dejavnostjo prodaje in razvoza čistilne opreme. Do danes so sistem razširili z mnogimi funkcionalnostmi in je primeren za uporabo v avtoprevozništvu, taksi službah, podjetjih s prodajo na terenu, trgovinah z lastno dostavo in drugih podjetij, katere dejavnost vključuje transport. Sistem pa lahko po željah naročnika prilagodijo njihovim potrebam. Sistem deluje v kombinaciji z obstoječimi napravami Garmin ali TimoCom. Sistem deluje preko spletnega portala, kamor se uporabnik prijavi s svojim uporabniškim imenom in geslom in ima takoj na razpolago pregled nad stanjem njegovih vozil ter uporabi drugih funkcionalnosti sistema [1].

Ključne funkcionalnosti sistema EasyTracker [1]:

- **Načrtovanje poti:** Rešitev omogoča planiranje poti in pregled realizacije načrtovanih opravi. Tako lahko podjetje načrtuje dostavo pošiljk, delo serviserjev, sestanke komercialistov in taksi vožnje. Načrtovane poti so pregledno prikazane tako na zemljevidu kot na seznamu. Sistem zna sam predlagati optimalni vrstni red obiska dodanih točk. Istočasno sistem preračuna skupno število kilometrov in predviden čas vožnje za posamezno vozilo, da se posameznemu vozniku ne doda preveč dela za delovni dan. Sistem omogoča avtomatski prenos vseh načrtovanih opravil v navigacijsko napravo Garmin v samem vozilu. Prenos se naredi avtomatsko in na daljavo.
- **Terminal:** Sistem omogoča brezplačno dvosmerno komunikacijo z voznikom preko sporočil. Če voznik pošlje dispečerju sporočilo izven delovnega časa dispečerja se sporočilo lahko pošlje na dispečerjev e-mail naslov in ga lahko zlahka prebere na svojem mobilnem telefonu.
- **Pregled trenutne lokacije vseh vozil:** Za vsako vozilo je možno preveriti njegovo trenutno lokacijo kot tudi čas zadnjega gibanja. To je možno preveriti na zemljevidu ali pa samo v tekstovni obliki. Vozila, ki se trenutno gibljejo imajo na zemljevidu označeno tudi smer vožnje.
- **Dodeljevanje voznika vozilu:** V primeru menjave vozil med različnimi vozniki sistem omogoča ročno (dispečer vnese podatek v sistem) ali avtomatsko (s pomočjo čitalca v vozilu in voznikove kartice) dodeljevanje voznika vozilu.
- **Tahograf:** Poleg lažjega pregleda kronologije vožnje, sistem opozarja tudi na prekoračena pravila (predolge vožnje, premalo počitka,…).
- **Zaklepanje vozil:** Zaklepanje parkiranih vozil poveča varnost voznega parka, saj uporabnika pravočasno opozori na nedovoljene premike. Obvestilo o nedovoljenem premiku uporabnik prejme na elektronski naslov ali mobilni telefon v obliki SMS sporočila.
- **Wirefence (geografsko ograjevanje):** Modul wirefence omogoča označevanje območij na zemljevidu. Ko vozilo zapusti določeno območje se sproži alarm in sistem preko besedilnega sporočila o tem obvesti odgovorno osebo.

3.2 Track.si

Sistem sledenja Track.si je v celoti razvit v Sloveniji, v ljubljanskem podjetju TRACK d.o.o. Podjetje uporabnikom omogoča neposredno sodelovanje pri širitvi zanimivih funkcionalnosti sistema. Tudi strežniki za delovanje sistema se nahajajo v Sloveniji, kar pripomore k visoki odzivnosti aplikacije. Podjetje ponuja rešitev Track.si v cenovno različnih paketih, odvisno od višine cene so nam potem na voljo različne funkcionalnosti. Vsi paketi, ne glede na ceno pa v osnovi zajemajo osnovno storitev GPS sledenja, ki zagotavlja pregled nad stanjem vozil v realnem času. Za delovanje sistema Track.si je potrebno v vozilo vgraditi eno izmed njihovih GPS sledilnih naprav in se z uporabniškim imenom in geslom povezati na eno izmed aplikacij, ki jih ponujajo (mobilna ali namizna) [16]. Shema delovanja sistema je prikazana na sliki 3.1.

Slika 3.1: Shema delovanja sistema Track.si

Ključne funkcionalnosti sistema Track.si [16]:

- **Sporočila – in komunikacija z voznikom:** Sporočila omogočajo komunikacijo med upravljalcem Track.si programske opreme in vozniki, ki imajo v vozilih naprave.
- **Dogodki in obveščanje:** Sistem omogoča, da smo ob določenem dogodku obveščeni na sms ali e-mail. Dogodki so lahko na primer aktivacija alarma, prvi premik vozila, in prekoračitev hitrosti.
- **Identifikacija s ključki iButton:** Sistem Track.si podpira identifikacijo uporabnikov vozil s ključki iButton. Vsak voznik se identificira s svojim ključkom iButton tako, da ga prisloni na bralnik ključkov. Na ta način ima sistem točno evidenco kdo in kdaj je uporabljal določena vozila.

- **Nadzor goriva CAN:** Nadzor goriva CAN je posebna funkcija sistema Track.si, ki omogoča natančen pregled nad porabo goriva vozila. Poročilo o porabi goriva vsebuje prevožene razdalje in porabljeno gorivo, sistem pa izračuna povprečno porabo glede na ta dva podatka.
- **Iskanje lokacij, predviden čas prihoda:** Funkcija iskanje lokacij / predviden čas prihoda omogoča računanje razdalje in predviden čas do prihoda izbrane naprave na lokacijo. Izberemo napravi in želeno lokacijo (cilj), sistem Track.si pa bo izračunal razdaljo od trenutne lokacije naprave do cilja.
- **Stanje naprav:** Stanje naprav je glavno okno, ki se odpre ob prijavi v aplikacijo. Razdeljeno je na dva dela: seznam z napravami in karto, ki prikazuje trenutno stanje naprav.
- **API funkcije:** Sistem ima na voljo nabor API funkcij, ki omogočajo partnerjem povezovanje drugih informacijskih sistemov s sistemom Track.si. Vsaka funkcija ima prvi vstopni parameter API key (API ključ). Na podlagi API ključa se določa seznam naprav do katerih lahko partnerjev sistem dostopa. S klicanjem različnih funkcij tako lahko dobimo podatke o stanju sledilne naprave, njenih postankih in ostale statistične podatke.

3.3 KJE-SI

Sistem sledenja KJE-SI je razvilo istoimensko slovensko podjetje iz Zagorja ob Savi. Sam sistem je zelo podoben prej omenjenemu sistemu Track.si. Tudi to rešitev sestavlja v vozilo vgrajena sledilna naprava, status naprave pa lahko potem spremljavo v spletni aplikaciji na osebnem računalniku ali mobilnem telefonu, v realnem času, z zamikom 5-10 sekund. Tudi tukaj imamo možnost izbire med več paketi in različnimi sledilnimi napravami, ki jih lahko vgradimo kar sami ali pa to delo prepustimo avtoelektrikarju [6].

Ključne funkcionalnosti sistema [6]:

- **Pregled vozil:** Ob odpiranju aplikacije KJE-SI za sledenje vozil, se prikaže osnovno okno aplikacije, kjer je na zemljevidu seznam vseh sledilnih naprav za katera ima

uporabnik pooblastila. Vozila, ki se premikajo so prikazana z zeleno barvo, vozila, ki mirujejo vsaj 5 minut (časovni interval je možno spremeniti) pa so obarvana rdeče. Spletna aplikacija podatke posodablja vsakih 10 sekund, kar je tudi običajni interval pošiljanja podatkov iz GPS komunikacijske naprave nameščene v vozilu.

- **Dvosmerna komunikacija:** Sistem sledenja omogoča dvosmerno komuniciranje med navigacijskimi napravami Garmin v vozilu in spletno aplikacijo KJE-SI. Iz aplikacije lahko pošiljamo vozniku na Garmin klasično sporočilo in sporočilo s položajem. Voznik potrjuje prejem sporočila, ali pošilja nova sporočila direktno operaterju v pisarno. Komunikacije je brezplačna, kar je še posebej dobrodošlo kadar se vozilo nahaja v tujini.
- **Geografsko ograjevanje:** Sistem omogoča prikaz obvestil, ko vozilo prispe na določeno lokacijo, mogoče je tudi pošiljanje podatka, kako dolgo se je vozilo na tej lokaciji zadržalo.
- **Simulacija vožnje:** Aplikacija omogoča tudi simulacijo vožnje, ki jo krmilimo z navigacijskimi gumbi – na ta način je mogoča popolna rekonstrukcija z vsemi podrobnostmi vožnje za poljuben datum zgodovine.

3.4 Primerjava rešitev in ugotovitve

Že po hitrem pregledu lastnosti in funkcionalnosti v prejšnjih podpoglavjih omenjenih rešitev sledenja, slovenskih proizvajalcev, lahko vidimo njihove podobnosti, razlike in prednosti posamezne storitve.

Takoj lahko opazimo, da je glavna razlika sistemov v načinu pridobivanja lokacijskih in drugih podatkov. Sistem EasyTracker za pridobivanje podatkov uporablja kar obstoječe Garmin oz. TimoCom naprave v vozilu, medtem ko sistema Track.si in KJE-SI za sledenje vozilom uporabljata svoje naprave, ki jih je potrebno naknadno vgraditi.

Podobnosti med samimi sistemi so predvsem v glavnih funkcionalnostih, ki jih nudijo. Vsi sistemi namreč pričakovano nudijo osnovne podatke o statusu vozil, storitev geografskega

ograjevanja, možnost komunikacije med voznikom in operaterjem ter proženje obvestil oz. alarmov ob določenih dogodkih. Izmed treh primerjanih sistemov po funkcionalnostih najbolj izstopa sistem EasyTracker saj omogoča najširši nabor dodatnih funkcionalnosti. Pozitivno pa nas je presenetila predvsem dodatna storitev sistema Track.si, ki programerjem oz. razvijalcem informacijskih sistemov omogoča klicanje API funkcij. Še najbolj skop z naborom funkcionalnosti je bil sistem KJE-SI, vendar tudi ta sistem omogoča vse, kar potrebujemo za osnovno sledenje vozilom. V tabeli so prikazane ključne prednosti obravnavanih sistemov in predlog uporabe sistema v praksi.

Tabela 3.1: Primerjava slovenskih sistemov za vodenje in spremljanje vozil

	EasyTracker	Track.si	KJE-SI
Prednost	Širok nabor dodatnih funkcionalnosti	Omogočano klicanje API funkcij	Minimalističnost
Predlog uporabe	Za podjetja z večjim številom voznega parka, ki želijo optimizirati svoje stroške in dobiti storitev za enostavno načrtovanje optimalne poti	Za podjetja, ki jim je glavno, da imajo ves čas pregled nad stanjem svojih vozil in podjetja, ki bi želela storitev sledenja vozil integrirati v svoj obstoječi informacijski sistem (API funkcije)	Za taksi službe in nezahtevna podjetja, ki bi rada spremljala stanje svojih vozil na terenu

4 PREDLAGANA REŠITEV

Po pregledu obstoječih rešitev smo ugotovili, da le-te zelo dobro rešujejo problematiko sledenja in vodenja vozil s pomočjo vgradnih oz. z uporabo obstoječih GPS sistemov v vozilu, prav tako nudijo tudi dovršene podporne funkcionalnosti o statusu ter pregledu vozila in izdelavo najrazličnejših analitičnih poročil. Med primerjanimi rešitvami pa nismo našli take, ki bi celovito reševale naš problem parkiranja tovornih vozil v urbanih okoljih, zato smo se odločili sami zasnovati shemo in ključne funkcionalnosti, ki bi jih naj takšna celovita rešitev zajemala.

Poglavitna ideja predlagane rešitve je, da bi sistem znal vozilo preusmeriti na prostor, kjer vozilo ne moti okolice (parkirišče ali počivališče), če je namembno podjetje vozila zasedeno. To v praksi pomeni, da bi sistem s pomočjo geografskega ograjevanja zaznal, kdaj je vozilo dovolj blizu namembnemu kraju, da začne s pregledovanjem prostih kapacitet v namembnem kraju vozila in ga na podlagi tega podatka ustrezno usmeri v podjetje oz. na parkirišče za tovorna vozila. Dober primer takega območja geo-meje, kjer bi sistem začel poizvedovati o prostih kapacitetah v namembnem podjetju bi bil npr. ob izvozu iz avtoceste (slika 4.1).

Slika 4.1: Primer območja geo-meje

Glede na to, da je na tržišču že veliko zmogljivih sistemov za sledenje in vodenje vozil, bi našo rešitev oblikovali podobno kot sistem Track.si, ki smo ga obravnavali v prejšnjem poglavju, na osnovi API funkcij, ki bi jih lahko razvijalci najrazličnejših rešitev implementirali v svoj sistem. Tako se nam ne bi bilo potrebno osredotočati na samo sledenje in vodenje vozil iz točke A do točke B ampak bi predlagano rešitev integrirali v že obstoječ sistem in bi tako dobili celovito rešitev, ki bi reševala našo problematiko.

Prednosti rešitve

Predlagana rešitev bi razbremenila promet na območjih kjer se prepletajo stanovanjska in industrijska območja. Voznikom več ne bi bilo potrebno obračati vozil, krožiti po območju in iskati ustreznega prostora za parkiranje vozil. Posledično bi nehali zasedati javne prometne površine (ceste in pločnike) ter povzročati težave v prometu (zastoji, varnost).

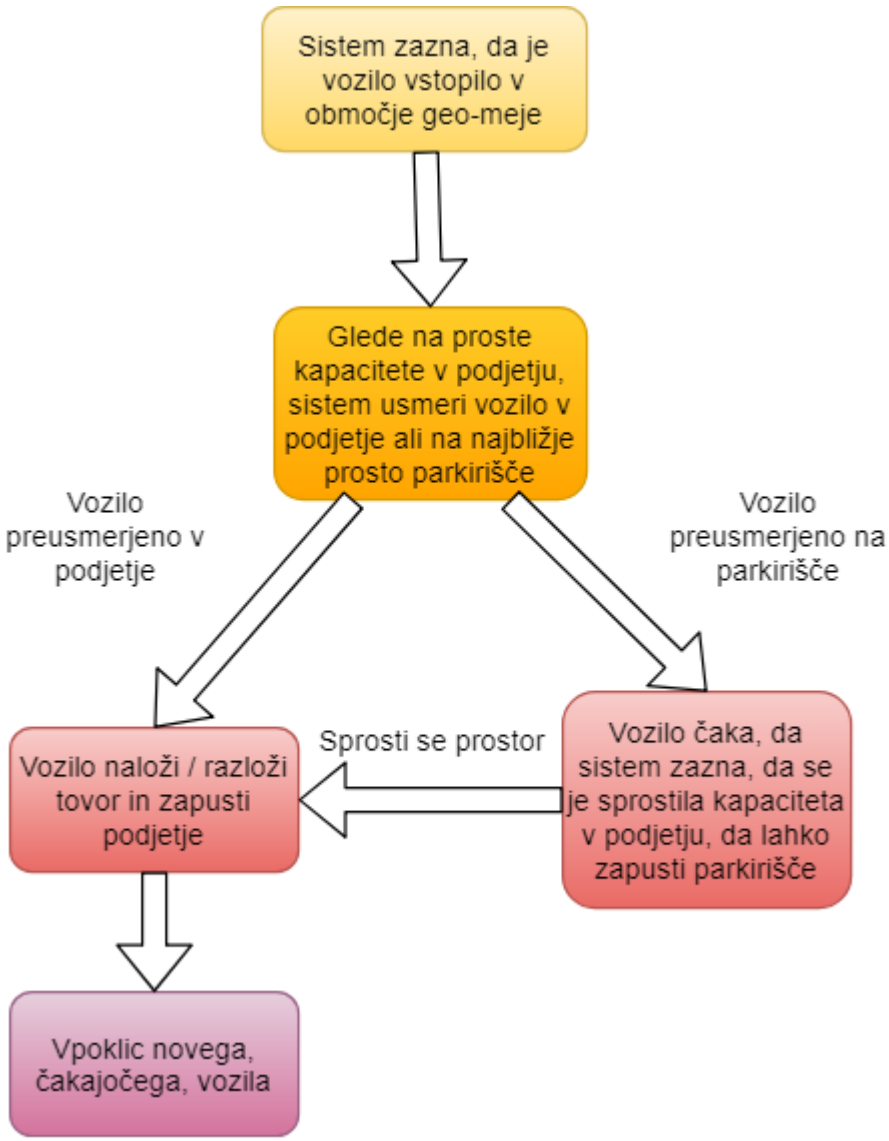
Priložnosti rešitve

Glede na vsesplošno pomanjkanje parkirnih prostorov za tovorna vozila, bi takšna rešitev po sej verjetnosti zahtevala vlaganja v izgradnjo nove infrastrukture (parkirišča) oz. preureditev obstoječe ali do zdaj neuporabljene infrastrukture, kar bi prineslo nove priložnosti za zaposlitev, podjetja z neizkoriščenim prostorom, pa bi lahko tak prostor preuredila v parkirišča in ga oddajala za potrebe parkiranja vozil. Urejena parkirišča blizu avtocest bi lahko oddajali tudi tovornim vozilom v tranzitu, ki bi jih izkoriščala kot počivališča in s tem razbremenili že tako preobremenjena avtocestna počivališča.

Omejitve rešitve

Rešitev, ki smo si jo zamislili bi pomenila dodatna vlaganja podjetij v informacijski sistem, saj bi morali nekako poskrbeti za zaznavanje prostih nakladalno/razkladalnih kapacitet v samem podjetju. Tukaj bi lahko uporabili parkirne senzorje podobne takim v parkiriščih sodobnih nakupovalnih centrov ali pa kakšno drugo tehnologijo (računalniški vid,…).

Slika 4.2 prikazuje diagram poteka procesa predlagane rešitve.



Slika 4.2: Diagram poteka predlagane rešitve

5 TEHNIČNE SPECIFIKACIJE

V tem poglavju smo napisali tehnično specifikacijo za predlagano rešitev, kar nam je služilo kot smernica v naslednjem poglavju, ko smo se lotili implementacije določenih konceptov rešitve.

5.1 Namen aplikacije

Namen aplikacije oz. spletnega servisa je ob pravem trenutku usmeriti tovorno vozilo na pravi kraj, z namenom zmanjšanja gneče in povečanja varnosti na cestah. Sistem za odločanje o usmeritvi vozila uporabi podatke, o lokaciji vozila (ki jih vozilo vsakih N sekund pošilja v sistem), o prostih kapacitetah nakladalnih/razkladalnih ramp v namembnem podjetju in prostih kapacitet bližnjih parkirišč za tovorna vozila.

5.2 Struktura aplikacije

Glavni sestavni deli aplikacije so uporabniški vmesnik, spletni vmesnik in podatkovna baza.

Uporabniški vmesnik bo klical metodo spletnega vmesnika ter prikazal vrnjeno statusno sporočilo, ki ga bo prejel s strani spletnega servisa, ta pa se bo v svojih metodah povezoval na podatkovno bazo in pridobil ustrezne podatke za svoj izračun.

Slika 5.1: Diagram strukture aplikacije

5.3 Uporabniški vmesnik

Izbira tehnologije v kateri bo izdelan uporabniški vmesnik je prepuščena razvijalcem že obstoječega programskega sistema, ki bo integriral našo rešitev. Uporabniški vmesnik je lahko mobilna aplikacija (Android oz. iOS) ali pa namizna Windows aplikacija. Edini pogoj je, da mora izbrana tehnologija omogočati klicanje spletnih servisov in na enostaven način prikazati vrnjen odgovor, ki je v obliki besedila.

5.4 Spletni vmesnik

Spletni vmesnik spletnega servisa je izdelan v Microsoftovem razvojnem okolju Visual Studio 2017, v programskem jeziku C, s pomočjo ogrodja Windows Communication Foundation.

Spletni servis bo imel glavno metodo *LogVehicleStatus*, katero bo klical uporabniški vmesnik. V metodi *LogVehicleStatus* pa se bodo klicale pomožne metode *CheckGeofences*, *CheckState* in *CheckForFreespace*. Vhodne iz izhodne parametre glavne in pomožnih metod bomo podrobneje opisali v nadaljevanju.

LogVehicleStatus

Je glavna metoda spletnega servisa, vhodni parametri metode so naslednji:

- Integer vehicleId – identifikacijska številka vozila,
- Double xPos – X koordinata položaja vozila,
- Double yPos – Y koordinata položaja vozila,
- Integer companyId – identifikacijska številka namembnega podjetja.

Izhodni parameter metode je *result* (String) – napotki za usmeritev vozila na parkirišče oz. v podjetje.

Metoda na podlagi vhodnih podatkov in podatkov, ki jih pridobi iz podatkovne baze, odjemalcu vrne sporočilo z navodili kam se naj usmeri vozilo. Če je v namembnem podjetju še prostor za nova tovorna vozila, sistem vozilo usmeri tja, v nasprotnem primeru pa vozilo usmeri na najbližje prosto parkirišče za tovorna vozila.

CheckGeofences

Je pomožna metoda, ki jo kliče glavna metoda, da preveri ali je vozilo vstopilo/izstopilo v kakšno geo-mejo.

Vhodni parameter metode je *vehicleId* (Integer) – identifikacijska števila vozila.

Izhodni parameter metode je *result* (String) – besedilni opis, ki pove ali je vozilo vstopilo v geo-mejo ali ne.

Metoda s pomočjo parametra *vehicleId*, dostopa do podatkovne baze in pridobi položaj ustreznega vozila. Nato preveri ali je v območju katere izmed geo-mej, katerih podatki so prav tako shranjeni v bazi. Nato metoda kliče še eno pomožno metodo (*CheckState*), ki preveri ali je vozilo vstopilo ali izstopilo ali pa sploh ni v nobeni geo-meji. Če metoda *CheckState* vrne status, da je vozilo vstopilo v geo-mejo, metoda *CheckGeofences* pokliče metodo *CheckForFreespace*, ki preveri ali je v namembnem podjetju prostor za tovorno vozilo oz. katero je najbližje parkirišče s prostim mestom. V primeru, da metoda *CheckState* vrne, da vozilo ni v geo-meji, metoda *CheckGeofences* vrne status »Vozilo ni v dosegu nobene geo-meje«.

CheckState

Je pomožna metoda, ki preveri ali je vozilo vstopilo v geo-mejo ali je iz nje izstopilo ali se morda v njej še vedno nahaja oz. ali sploh ni v dosegu nobene geo-meje.

Vhodna parametra:

- *VehicleId* (Integer) – identifikacijska številka vozila,

- *geofenceId* (Integer) – identifikacijska številka geo-meje v kateri se vozilo trenutno nahaja.

Izhodni parameter je *result* (Integer) – status o stanju položaju vozila v geo-meji (0 – vozilo se ne nahaja v nobeni geo-meji; 1 – vozilo je vstopilo v geo-mejo; 2 – vozilo je izstopilo iz geo-meje; 3 – vozilo se še vedno nahaj v isti geo-meji kot ob prejšnjem preverjanju lokacije).

Metoda primerja trenutno geo-mejo (*geofenceId*) z geo-mejo, ki je bila v shranjena v podatkovno bazo ob prejšnjem merjenju položaja vozila in na podlagi ugotovitev vrne ustrezen status.

CheckForFreespace

Še zadnja pomožna metoda, ki preveri ali ima namembno podjetje prostor, da sprejme vozilo, če podjetje prostora nima, metoda poišče najbližje prosto parkirišče in vozilo preusmeri tja.

Vhodna parametra:

- *vehicleId* (Integer) – identifikacijska številka vozila,
- *companyId* (Integer) – identifikacijska številka namembnega podjetja.

Izhodni parameter je *result* (String) – opis stanja kam naj se usmeri vozilo.

Ko je poklicana, metoda dostopa do podatkovne baze in preveri ali je v namembnem podjetju (companyId) še kaj prostora za tovorno vozilo. Če je prostor, metoda vrne status »Pojdi v namembno podjetje«. V primeru, da prostora ni, metoda dostopa do seznama parkirnih mest, ki je shranjen v podatkovni bazi in poišče takšno prosto parkirišče, ki je najbližje namembnemu podjetju in v statusu vrne naziv tega parkirišča ter podatek o številu prostih parkirnih mest.

5.5 Podatkovna baza

Podatki se lahko nahajajo v poljubni podatkovni bazi, ki ima tipizirane lastnosti (Microsoft SQL Server, MySQl, PostgreSQL, MongoDB,…).

Baza mora vsebovati naslednje tabele:

- *Company* – podatki o namembnih podjetjih (tabela 5.1),
- *Geofence* – podatki o geo-mejah (tabela 5.2),
- *Parking* – podatki o parkiriščih za tovorna vozila (tabela 5.3),
- *VehicleLogs* – status in položaj vozil (tabela 5.4).

Tabela 5.1: Zgradba tabele Company

Lastnost	Opis
ID	Identifikacijska št. podjetja
Name	Naziv podjetja
xPos	X koordinata podjetja
yPos	Y koordinata podjetja
freeSpace	Št. prostih parkirnih mest v podjetju

Tabela 5.2: Zgradba tabele Geofence

Lastnost	Opis
ID	Identifikacijska št. geo-meje
xPos	X koordinata središča geo-meje
yPos	Y koordinata središča geo-meje
Radius	Polmer geo-meje
Name	Naziv geo-meje

Tabela 5.3: Zgradba tabele Parking

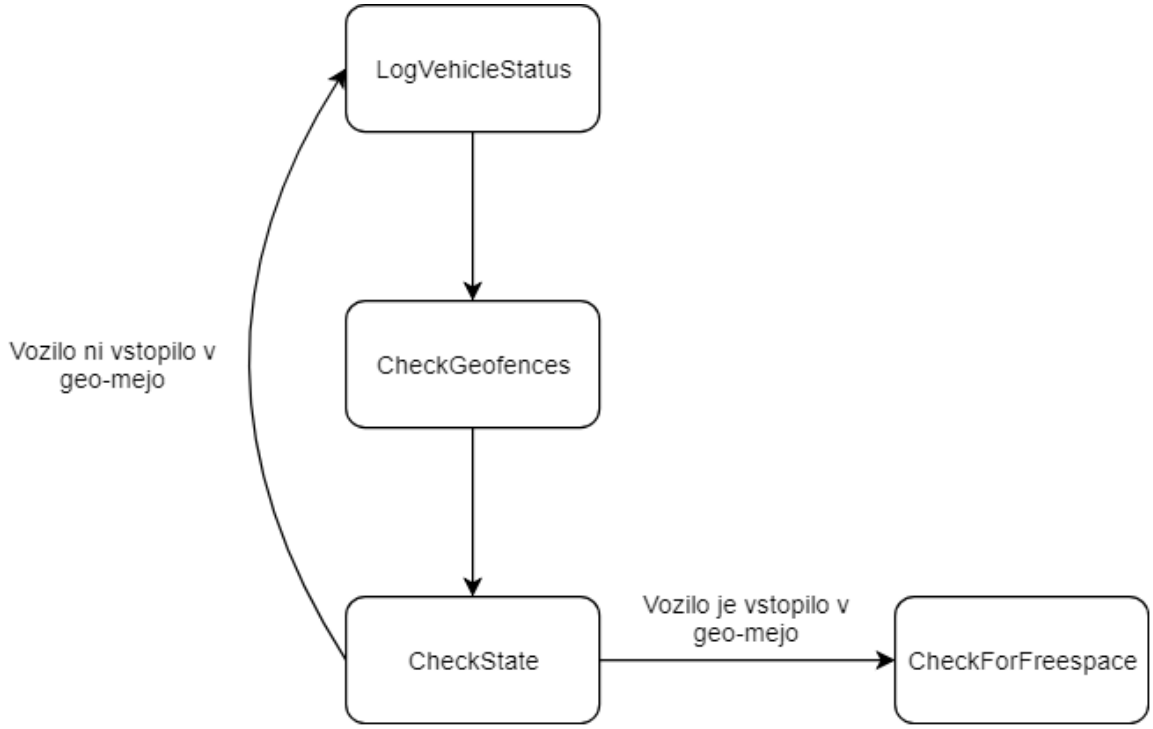
Lastnost	Opis
ID	Identifikacijska št. parkirišča
Name	Naziv parkirišča
freeSpace	Število prostih parkirnih mest
xPos	X koordinata parkirišča
yPos	Y koordinata parkirišča

Tabela 5.4: Zgradba tabele VehicleLogs

Lastnost	Opis
ID	Identifikacijska številka dnevniškega zapisa
vehicleId	Identifikacijska številka vozila
currentGeofenceId	Identifikacijska številka geo-meje v kateri se vozilo trenutno nahaja
companyId	Identifikacijska številka namembnega podjetja
xPos	X koordinata položaja vozila
yPos	Y koordinata položaja vozila

5.6 Delovanje aplikacije

Odjemalčeva aplikacija vsakih N sekund kliče glavno metodo spletnega servisa, ki preveri ali je vozilo vstopilo v geo-mejo. Če je vozilo vstopilo v geo-mejo metoda začne preverjati kam naj usmeri tovorno vozilo. V primeru, ima namembno podjetje prostor, vozilo usmeri tja, v nasprotnem primeru pa vozilo preusmeri na najbližje prosto parkirišče. Slika 5.2 prikazuje hierarhijo klicev glavne metode in pomožnih metod.



Slika 5.2: Hierarhija klicev metode in pomožnih metod

6 IMPLEMENTACIJA KONCEPTA GEOGRAFSKEGA OGRAJEVANJA

V tem poglavju smo se lotili implementacije koncepta geografskega ograjevanja kot spletno storitev v ogrodju .NET. Najprej smo opisali tehnologije, ki smo jih uporabili za implementacijo nato pa smo s pomočjo slik programske kode podrobneje opisali sam postopek implementacije.

6.1 Uporabljene tehnologije

V naslednjem podpoglavju bomo našteli in na kratko opisali glavne tehnologije, katere so bile uporabljene za implementacijo koncepta geografskega ograjevanja. Za naslednje tehnologije smo se odločili, ker smo jih obravnavali skozi izobraževanje na fakulteti in si tako posledično pridobili ustrezno predznanje za njihovo uporabo.

C#

C# je eleganten in tipiziran, objektno usmerjen programski jezik, ki razvijalcem omogoča, da razvijajo cel spekter različnih, varnih in robustnih aplikacij, ki se izvajajo v ogrodju .NET. C# lahko uporabimo za ustvarjanje namiznih Windows aplikacij, spletnih storitev, aplikacij odjemalec-strežnik, aplikacij podatkovne baze in še veliko več [7].

Sintaksa programskega jezika C# je zelo izrazita, vendar je vseeno preprosta in enostavna za učenje. Sintaksa z zavitimi oklepaji je takoj prepoznavna za vsakogar, ki je seznanjen s programskimi jeziki C, C++ ali Java. Razvijalci, ki poznajo katerega od teh jezikov, so običajno sposobni začeti produktivno razvijati v C# v zelo kratkem času. C# s sintakso poenostavlja veliko kompleksnih stvari jezika C++ in nudi zmogljive funkcionalnosti(tipi ničelnih vrednosti, enumeracije, delegati, lambda izrazi in direktni dostop do pomnilnika), ki jih ni mogoče najti v Javi [7].

Microsoft Visual Studio

Visual Studio je integrirano razvojno okolje (v nadaljevanju IDE) podjetja Microsoft. Namenjeno je razvoju programov za operacijske sisteme Windows, spletnih strani, spletnih aplikacij, spletnih storitev in aplikacij na osnovi ogrodja .NET. IDE je program, bogat s funkcijami, ki se lahko uporablja za številne vidike razvoja programske opreme. Poleg standardnega urejevalnika in razhroščevalnika, ki ga zagotavlja večina IDE-jev, Visual Studio vključuje tudi prevajalnike, orodja za samodokončanje kode, grafične urejevalnike in še veliko drugih funkcionalnosti za lažji razvoj programske opreme [8].

Windows Communication Foundation

Windows Communication Foundation (v nadaljevanju WCF) je ogrodje za razvijanje storitveno usmerjenih aplikacij. Z uporabo WCF lahko podatke pošiljamo ko asinhrona sporočila iz ene končne točke storitve v drugo. Končna točka storitve je lahko del stalno razpoložljive storitve, ki jo gosti strežnik IIS, ali pa je storitev gostiteljica v aplikaciji. Sporočila, ki se prenašajo so lahko tako enostavna kot en sam znak ali beseda, poslana kot XML, ali tako zapletena kot tok binarnih podatkov [9].

Microsoft SQL Server

Microsoft SQL Server je sistem za upravljanje relacijskih podatkovnih baz, ki podpira številne aplikacije za obdelavo transakcij, poslovne inteligence in analitike v poslovnih okoljih IT. Je eden izmed treh vodilnih tehnologij podatkovnih baz na tržišču, skupaj z Oracle Database in IBM-ovim DB2 [14].

Tako kot druga programska oprema za upravljanje relacijskih podatkovnih baz je tudi Microsoft SQL Server zgrajen na podlagi strukturiranega povpraševalnega jezika (v nadaljevanju SQL), ki ga uporabljajo administratorji podatkovnih baz in drugi IT strokovnjaki za poizvedovanje podatkov in upravljanje podatkovnih baz. SQL Server je vezan na Transact-SQL, Microsoftovo implementacijo jezika SQL, ki v standardni jezik doda niz lastnih programskih razširitev [14].

Entity Framework

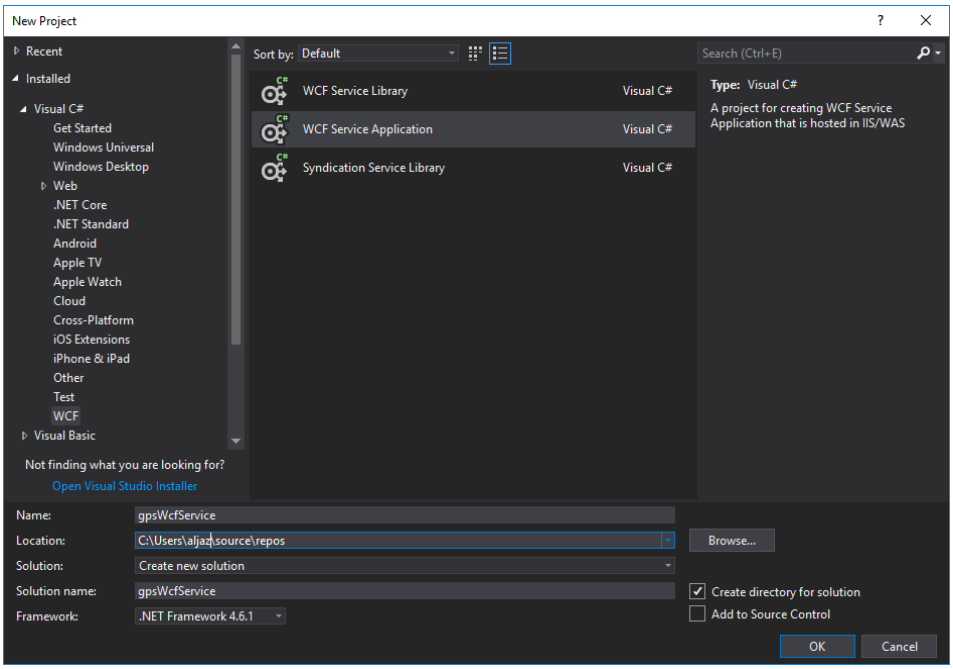
Entity Framework je odprtokodno ogrodje, podprto s strani Microsofta, za objektno-relacijsko mapiranje za aplikacije razvite v okolju .NET. Razvijalcem omogoča, da uporabljajo objekte domensko specifičnih razredov, ne da bi se pri tem osredotočali na tabele in stolpce v podatkovni bazi, kjer so ti podatki shranjeni. Z uporabo ogrodja Entity Framework lahko razvijalci pri obdelavi podatkov delajo na višji stopnji abstrakcije in lahko ustvarijo in vzdržujejo podatkovne aplikacije z manj kode v primerjavi s tradicionalnimi aplikacijami [2].

6.2 Implementacija

Po smernicah iz prejšnjega poglavja, ko smo napisali tehnično specifikacijo za predlagano rešitev, smo se v tem poglavju lotili same implementacije.

Ustvarjanje projekta

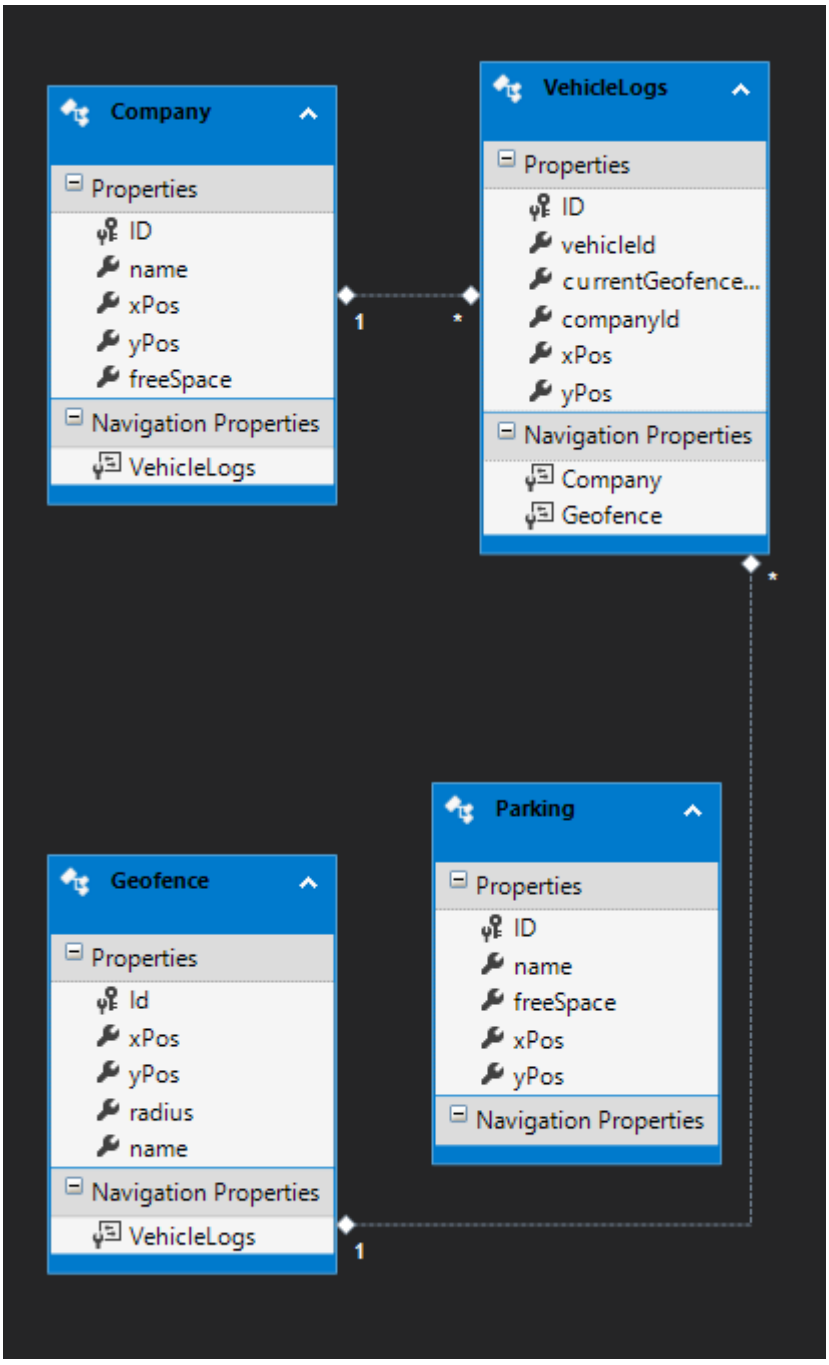
V orodju Visual Studio smo ustvarili nov WCF projekt, kot je prikazano na sliki 6.1.



Slika 6.1: Ustvarjanje WCF projekta

Podatkovna baza

V samem okolju Visual Studio smo tudi ustvarili lokalno podatkovno bazo z modelom, ki je bil opisan v prejšnjem poglavju tega dela. Entitetno-relacijski diagram podatkovne baze je prikazan na sliki 6.2.



Slika 6.2: Entitetno-relacijski diagram podatkovne baze

Vmesnik:

V vmesniku smo definirali glavno metodo servisa in njene vhodne parametre na način kot prikazuje slika 6.3.

```
[ServiceContract]
public interface IService1
{
    [OperationContract]
    string LogVehicleStatus(int vehicleId, double xPos, double yPos, int companyId);
}
```

Slika 6.3: Vmesnik spletnega servisa

Razred Service1

Service1 je osrednji (main) razred spletnega vmesnika. V njem je implementirana metoda *LogVehicleStatus*, ki je definirana v vmesniku. Razred vsebuje globalno spremenljivko *geofenceDB* (slika 6.4), ki je povezava do podatkovne baze, implementirana je s pomočjo ogrodja Entity Framework in nam omogoča objektni dostop do tabel in stolpcev v podatkovni bazi (ni potrebe po uporabi jezika SQL). Poleg globalne spremenljivke *geofenceDB*, razred *Service1* vsebuje še glavno metodo *LogVehicleStatus* in pomožne metode *CheckGeofences*, *CheckState* in *CheckForFreespace*.

```
private bazaGpsEntities geofenceDB = new bazaGpsEntities();
```

Slika 6.4: Spremenljivka geofenceDB

LogVehicleStatus

Sledila je implementacija glavne metode spletnega servisa. Ko se metoda *LogVehicleStatus* pokliče prvič za izbrano vozilo, se to vozilo doda v tabelo *VehicleLogs*, ko se potem pokliče naslednjič vsakih N sekund se vnos v tabeli *VehicleLogs* samo posodobi z zadnjo lokacijo vozila. Vsakič ko v programski kodi dostopamo do podatkovne baze in spreminjamo podatke, je treba spremembe tudi shraniti. To dosežemo z metodo *SaveChanges*(). V implementaciji smo uporabili tudi *try-catch* stavek, ki v primeru napake ne zaustavi

delovanja aplikacije ampak samo javi za kakšno napako gre. Celotna implementacija metode *LogVehicleStatus* je prikazana na sliki 6.5.

```
// Metoda doda vozilo v dnevnik - ob začetku vožnje; oz. posodobi pozicijo vozila med vožnjo
public string LogVehicleStatus(int vehicleId, double xPos, double yPos, int companyId)
{
    string result = "";

    VehicleLogs logToAdd = new VehicleLogs();
    logToAdd.vehicleId = vehicleId;
    logToAdd.xPos = xPos;
    logToAdd.yPos = yPos;
    logToAdd.companyId = companyId;
    logToAdd.currentGeofenceId = 0;

    try
    {
        VehicleLogs exsistingLog = geofenceDB.VehicleLogs.Where(x => x.vehicleId == vehicleId).FirstOrDefault();

        // Preverimo ali v dnevniku že obstaja zapis za vozilo s tem ID-jem
        if (exsistingLog != null) // Če obstaja, zapis posodobimo z novimi podatki
        {
            exsistingLog.xPos = xPos;
            exsistingLog.yPos = yPos;
            exsistingLog.companyId = companyId;
            exsistingLog.currentGeofenceId = 0;
            result = "Status: UPDATE - Vozilo z ID " + vehicleId + " je bilo posodobljeno";
        }
        else  // Če vozilo s tem ID-jem še ne obstaja, ga dodamo v dnevnik
        {
            geofenceDB.VehicleLogs.Add(logToAdd);
            result = "Status: NEW - Vozilo z ID " + vehicleId + " je bilo dodano v bazo";
        }

        // Shranimo spremembe na podatkovni bazi
        geofenceDB.SaveChanges();

         result += ". " + CheckGeofences(vehicleId);

    }
    catch (Exception e)
    {
        result = e.ToString();
    }

    return result;
}
```

Slika 6.5: Metoda LogVehicleStatus

CheckGeofences

Nato smo implementirali pomožno metodo *CheckGeofences* (slika 6.6), ki preveri ali je vozilo na območju katere izmed geo-mej, ki so shranjene v podatkovni bazi v tabeli *Geofence*. To naredi na način, da primerja razdaljo od trenutne lokacije vozila do središča geo-meje z dolžino polmera geo-meje, če je razdalja enaka ali manjša polmeru geo-meje potem vemo, da je vozilo v območju dotične geo-meje. Ko dobimo ID geo-meje v kateri se nahaja vozilo, kličemo še eno pomožno metodo – *CheckState*, da preverimo ali je vozilo

vstopilo, izstopilo ali pa se še vedno nahaja v določeni geo-meji. Če nam metoda *CheckState* vrne status, da je vozilo vstopilo v geo-mejo, kličemo metodo *CheckForFreespace*, ki preveri ali lahko vozilo parkira v namembnem podjetju ali pa ga je treba morebiti preusmeriti na najbližje prosto parkirišče.

```
// Metoda preveri ali je vozilo vstopilo ali izstopilo iz kakšne geo-meje in ustrezno usmeri vozilo -
//odjemalec med vožnjo kliče metodo vsakih N sekund
private string CheckGeofences(int vehicleId)
{
    string result = "";

    // Pridobimo vozilo iz podatkovne baze
    VehicleLogs vehicleLog = geofenceDB.VehicleLogs.Where(x => x.vehicleId == vehicleId).FirstOrDefault();

    // Trenutna koordinata vozila
    GeoCoordinate vehiclePos = new GeoCoordinate(vehicleLog.xPos, vehicleLog.yPos);

    // Iz baze pridobimo seznam vseh geo-mej
    List<Geofence> geofences = geofenceDB.Geofence.ToList();

    // Inicializiramo privzeto geo-mejo
    int geofenceId = 0;

    // Preverimo razdaljo od vozila do vsake geo-meje
    foreach (Geofence geofence in geofences)
    {
        GeoCoordinate geofencePos = new GeoCoordinate(geofence.xPos, geofence.yPos);
        double distance = vehiclePos.GetDistanceTo(geofencePos);

        // Preverimo ali je vozilo v območju geo-meje
        if (distance <= geofence.radius) // Vozilo je v območju geo-meje
        {
            // Nastavimo spremenljivko na ID od geo-meje v kateri se nahaja vozilo
            geofenceId = geofence.Id;
            break;
        }
    }

    if (CheckState(vehicleId, geofenceId) == 1) // Vozilo je vstopilo v geo-mejo
    {
        result = CheckForFreespace(vehicleId, vehicleLog.companyId);
    }
    else if (CheckState(vehicleId, geofenceId) == 0) // Vozilo ni vstopilo v geo-mejo
    {
        result = "Vozilo ni v dosegu nobene geo-meje";
    }

    return result;
}
```

Slika 6.6: Metoda CheckGeofences

CheckState

Naslednja pomožna metoda – *CheckState* (Slika 6.7), primerja ID geo-meje, ki jo dobi kot vhodni parameter in ID geo-meje, ki je zabeležen kot zadnja lokacija vozila. Na podlagi primerjav, vrne status ali je vozilo vstopilo v določeno geo-mejo, ali je iz nje izstopilo ali je

morda še vedno v isti geo-meji kot pri zadnjem merjenju lokacije ali pa vozilo sploh ni v nobeni geo-meji.

```
// Metoda preveri ali je vozilo vstopilo, zapustilo ali pa se še vedno nahaja v geo-meji.
//0 - ni v geo-meji, 1- vstopilo, 2 - izstopilo, 3 - še vedno je v enaki geo-meji
private int CheckState(int vehicleId, int geofenceId)
{
    int result = 0;

    VehicleLogs log = geofenceDB.VehicleLogs.Where(x => x.vehicleId == vehicleId).First();

    if (log.currentGeofenceId == 0 && geofenceId == 0) // Vozilo ni v geo-meji
    {
        result = 0;
    }
    else if (log.currentGeofenceId == 0 && geofenceId != 0) // Vozilo je vstopilo v geo-mejo
    {
        log.currentGeofenceId = geofenceId;
        geofenceDB.SaveChanges();
        result = 1;
    }
    else if (log.currentGeofenceId != 0 && geofenceId == 0) // Vozilo je izstopilo iz geo-meje
    {
        log.currentGeofenceId = geofenceId;
        geofenceDB.SaveChanges();
        result = 2;
    }
    else if (log.currentGeofenceId == geofenceId) // Vozilo se še vedno nahaja v geo-meji
    {
        result = 3;
    }

    return result;
}
```

Slika 6.7: Metoda CheckState

CheckForFreespace

V primeru da metoda CheckState javi, da je vozilo vstopilo v območje geo-meje se proži metoda CheckForFreespace. Metoda najprej iz podatkovne baze pridobi podatke o namembnem podjetju vozila, če ima podjetje prosto mesto za vozilo, vrne status, da naj se vozilo usmeri v namembno podjetje (slika 6.8).

```
// Metoda preveri ali ima namembno podjetje prostor, da vozilo sparkira oz. če nima, vozilo usmeri
//na najbližje parkirišče
private string CheckForFreespace(int vehicleId, int companyId)
{
    string result = "";
    // Dobimo namembno podjetje iz baze
    Company destinationCompany = geofenceDB.Company.Where(x => x.ID == companyId).First();
    // Preverimo ali ima namembno podjetje prosto parkirno mesto za vozilo
    if (destinationCompany.freeSpace > 0) // Če ga ima, vozilo usmerimo v podjetje
    {
        result = "Pojdi v namembno podjetje (" + destinationCompany.name + ")";
    }
```

Slika 6.8: Usmeritev vozila v namembno podjetje

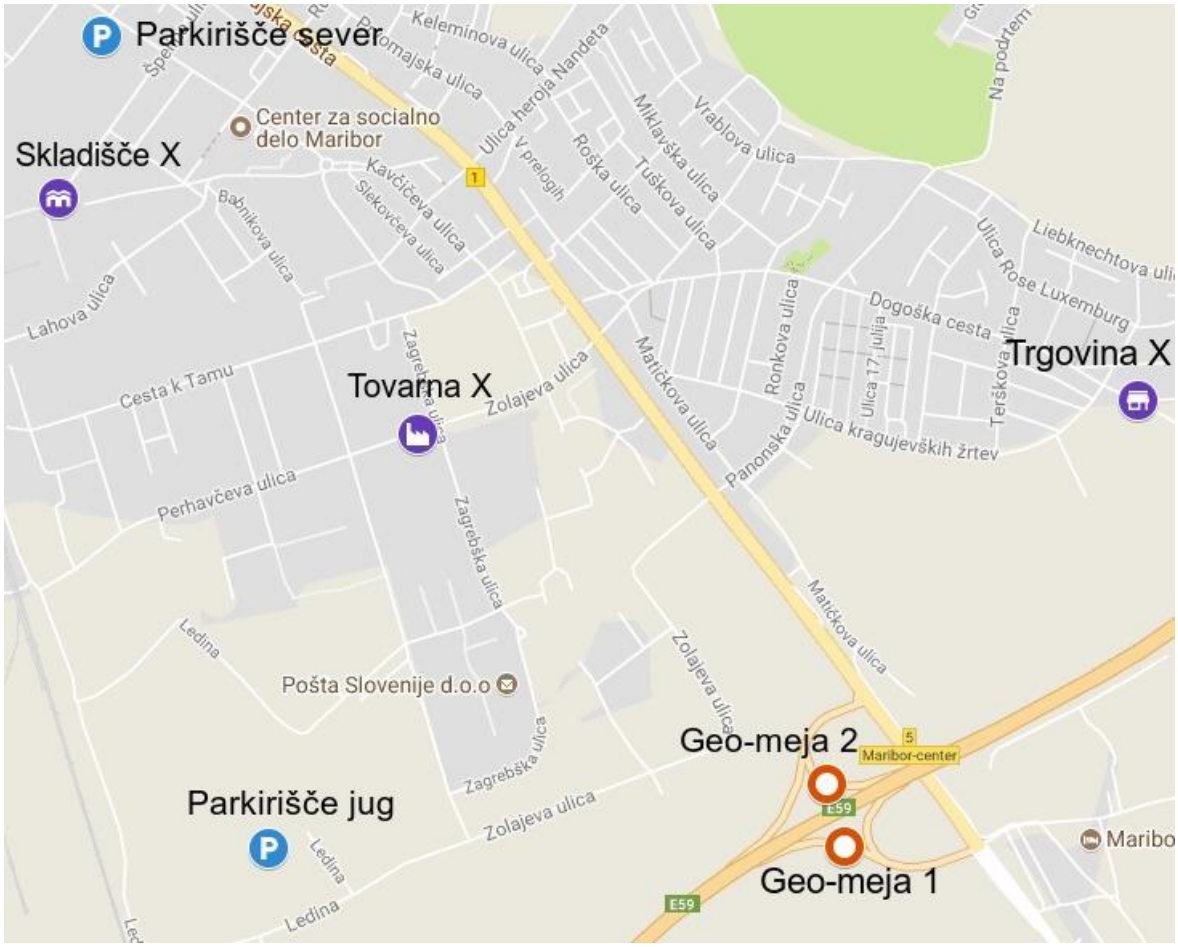
V primeru da podjetje nima prostora pa metoda poišče najbližje prosto parkirišče v bližini podjetja (slika 6.9).

```
else // Če ga nima, poiščemo namembnemu podjetju najbližje prosto parkirišče
{
    // Pridobimo seznam vseh prostih parkirišč iz baze
    List<Parking> freeParkings = geofenceDB.Parking.Where(x => x.freeSpace > 0).ToList();

    // Če v seznamu ni prostih parkirišč, metoda to sporoči
    if (freeParkings.Count == 0)
    {
        result = "Namembno podjetje je zasedeno in v bližini prav tako ni nobenega prostega parkirišča.";
    }
    else
    { // Če obstajajo prosta parkirišča, metoda poišče najbližje parkirišče podjetju in sporoči ime tega parkirišča

        Parking closestParking = null;
        double distanceCompanyParking = 0;

        GeoCoordinate companyPos = new GeoCoordinate(destinationCompany.xPos, destinationCompany.yPos);

        foreach (Parking parking in freeParkings)
        {
            GeoCoordinate parkingPos = new GeoCoordinate(parking.xPos, parking.yPos);

            if (closestParking == null) // Kot najbližje parkirišče dodelimo prvo parkirišče v seznamu
            {
                closestParking = parking;
                distanceCompanyParking = companyPos.GetDistanceTo(parkingPos);
            }
            else // Vsa naslednja parkirišča preverimo, če so bližje kot trenutno najbližje parkirišče
            {
                if (companyPos.GetDistanceTo(parkingPos) < distanceCompanyParking) // Našli smo bližje parkirišče
                {
                    closestParking = parking;
                }
            }
        }
        result = "Podjetje je zasedeno. Pojdi na parkirišče " + closestParking.name + ", ima še "
            + closestParking.freeSpace + " prostih mest.";
    }
}


return result;
```

Slika 6.9: Preusmeritev vozila na najbližje prosto parkirišče

6.3 Testiranje

V tem podpoglavju smo testirali našo spletno storitev WCF in preverili, da vse deluje po pričakovanjih. Za namene testiranja smo tabele v podatkovni bazi napolnili s testnimi podatki, da dobimo situacijo, ki jo prikazuje slika 6.10. V tabelo *Company* smo dodali tri podjetja, v tabelo *Geofence* in *Parking* pa po dve geo-meji oz. parkirišča. Spletni servis smo testirali kar v testnem orodju WCF Test Client, ki je dol okolja Visual Studio in je namenjen testiranju implementiranih spletnih storitev WCF.



Slika 6.10: Testna situacija

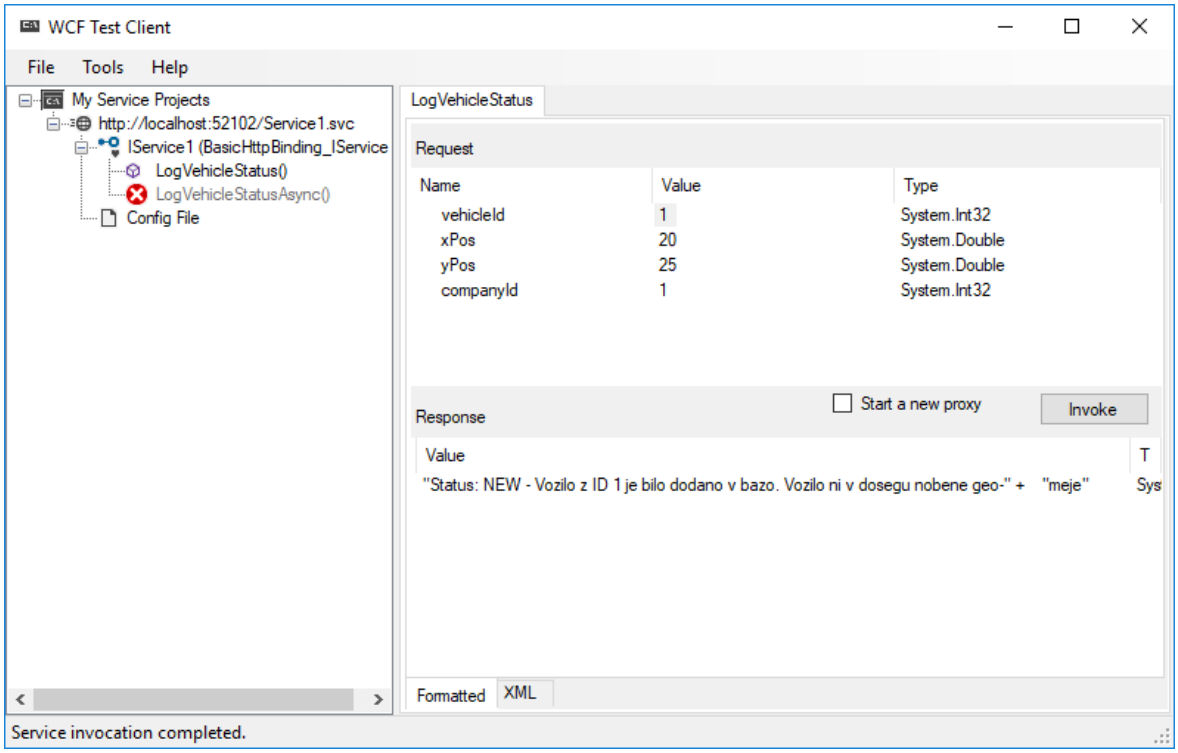
Testirali smo vse štiri možne scenarije, in sicer:

- Vozilo ni v geo-meji.
- Vozilo je v geo-meji, v namembnem podjetju je prostor za vozilo.

- Vozilo je v geo-meji, v namembnem podjetju ni prostora za vozilo.
- Vozilo je v geo-meji, v namembnem podjetju in na parkirišču ni prostora za vozilo.

Primer 1: Vozilo ni v geo-meji

V prvem scenariju smo testirali ali metoda vrne pravilni odgovor na zahtevo, če vozilo ni v nobeni geo-meji. Spletni servis smo poklicali s koordinatama (X = 20, Y = 25), ki sta zelo daleč od obeh dejanskih geo-mej (X1 = 46,51798, Y1 = 15,68051 in X2 = 46,51913, Y2 = 15,68007). Vhodne parametre spletnega servisa prikazuje slika 6.11.



Slika 6.11: Poizvedba in odgovor 1. scenarija

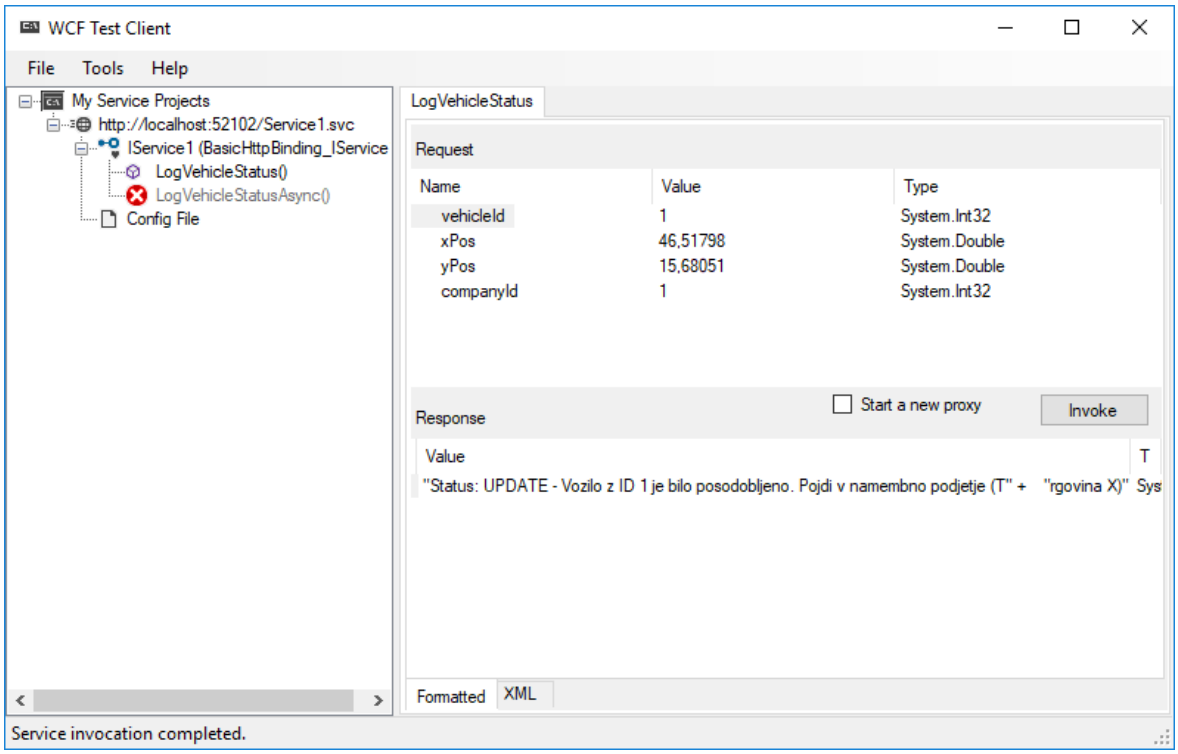
Iz slike 6.11 je možno razbrati, da je bil odgovor, ki smo ga prejeli (»Status: NEW – Vozilo z ID 1 je bilo dodano v bazo. Vozilo ni v dosegu nobene geo-meje«) ustrezen glede na scenarij testa. Vozilo je bilo tudi uspešno dodano v tabelo *VehicleLogs* (slika 6.12) in je tudi pravilno zabeleženo da vozilo ni v nobeni geo-meji, saj je vrednost stolpca *currentGeofenceId* enaka 0.

	ID	vehicleId	currentGeofen...	companyId	xPos	yPos
▷	8	1	0	1	20	25

Slika 6.12: Tabela VehicleLogs po 1. scenariju

Primer 2: Vozilo je v geo-meji, v namembnem podjetju je prostor za vozilo

V drugem testnem scenariju bomo vhodne podatke klicanega servisa prilagodili tako, da bo vozilo v dosegu geo-meje 1. V tabeli *Company* bomo namembnemu podjetju prav tako priredili podatek v stolpcu *freeSpace*, da je v podjetju vsaj eno prosto mesto za tovorna vozila. Vhodne parametre in odgovor servisa za scenarij 2 prikazuje slika 6.13.



Slika 6.13: Poizvedba in odgovor 2. scenarija

Tudi v tem primeru nam spletni servis vrača pravilni rezultat, in sicer: »Status: UPDATE – Vozilo z ID 1 je bilo posodobljeno. Pojdi v namembno podjetje (Trgovina X)«. Sistem je prepoznal, da je vozilo z ID-jem 1 že v podatkovni bazi in je podatke o vozilu samo posodobil, ter v odgovoru javil, da je v namembnem podjetju prostor. Stanje v tabeli *VehicleLogs* po drugi poizvedbi prikazuje slika 6.14.

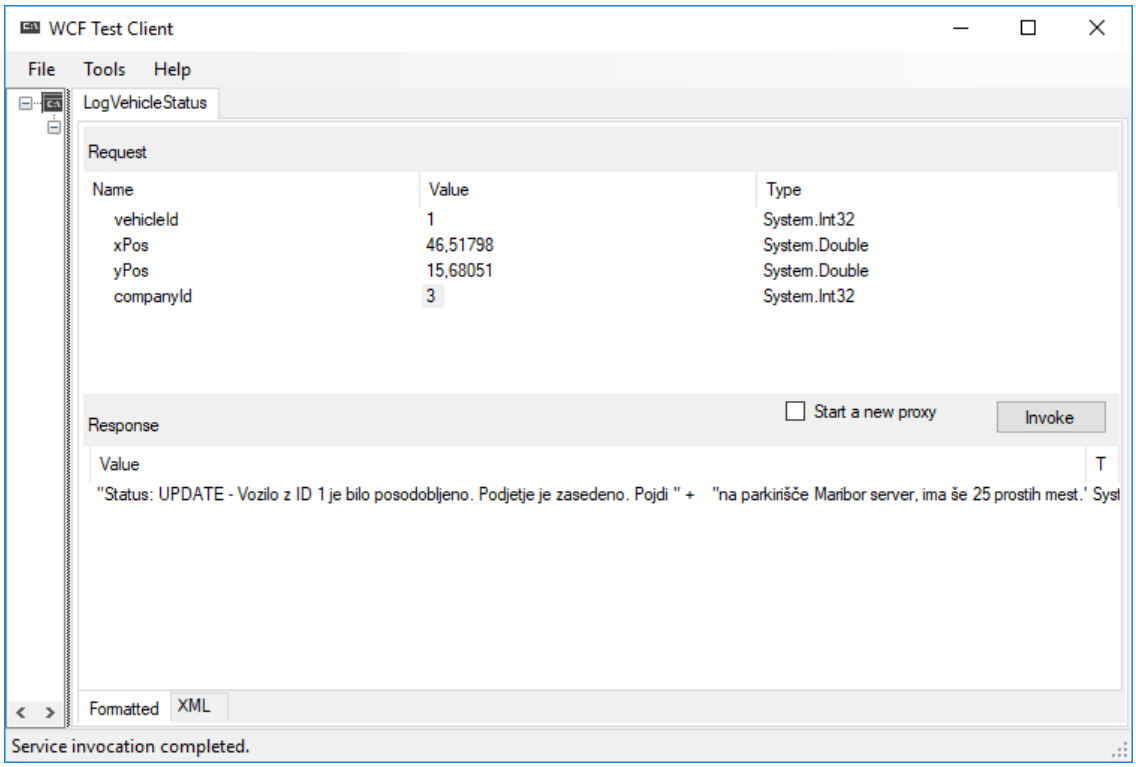
	ID	vehicleId	currentGeofen...	companyId	xPos	yPos
▷	8	1	1	1	46,51798	15,68051

Slika 6.14: Tabela VehicleLogs po 2. scenariju

Opazimo lahko, da so se posodobili podatki o lokaciji vozila, kot tudi v kateri geo-meji se vozilo nahaja.

Primer 3: Vozilo je v geo-meji, v namembnem podjetju ni prostora za vozilo

V tretjem testnem primeru nas je zanimalo ali metoda vrača pravilni rezultat, če v podatkovni bazi priredimo podatek, da v namembnem podjetju ni prostora za tovorno vozilo. V tem primeru smo vozilo preusmerili v Skladišče X (companyId = 3), temu podjetju smo v podatkovni bazi prav tako priredili podatek o prostih mestih za tovorna vozila (freeSpace = 0). Če sklepamo po sliki 6.10 bi nas moral sistem preusmeriti na Parkirišče sever, saj je to parkirišče najbližje Skladišču X. Vhodne parametre in rezultat metode prikazuje slika 6.15.



Slika 6.15: Poizvedba in odgovor 3. scenarija

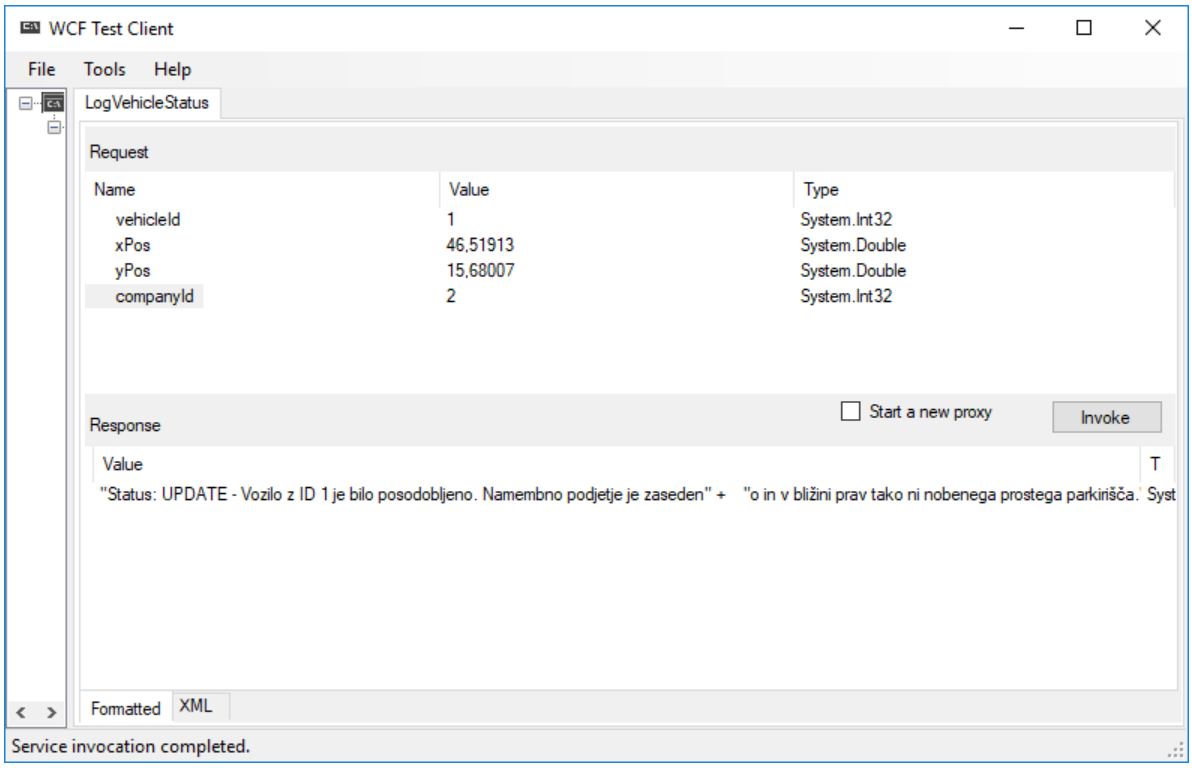
Tudi tokrat je metoda delovala po pričakovanjih in vozilo preusmerila na Parkirišče sever. Ponovno je zaznala, da je vozilo z ID-jem 1 že v podatkovni bazi in je smo posodobila podatke o vnosu (slika 6.16).

	ID	vehicleId	currentGeofen...	companyId	xPos	yPos
▷	8	1	1	3	46,51798	15,68051

Slika 6.16: Tabela VehicleLogs po 3. scenariju

Primer 4: Vozilo je v geo-meji v namembnem podjetju in na parkirišču ni prostora

V zadnjem scenariju smo preverili ali spletni servis vrne pravilen rezultat v primeru, da ni prostora ne v namembnem podjetju in prav tako ne na nobenem parkirišču v bližini podjetja. Za namen scenarija smo vozilo postavili v geo-mejo 2 in mu določili namembno podjetje Tovarna X (companyId = 2). Podjetju in obema parkiriščema v tabeli *Parking* smo nastavili stolpec *freeSpace* na vrednost 0. Ko smo uredili vse potrebne spremembe in poklicali servis, smo dobili rezultat, ki ga prikazuje slika 6.17.



Slika 6.17: Poizvedba in odgovor 4. scenarija

Metoda vrne odgovor: »Status: UPDATE – Vozilo z ID 1 je bilo posodobljeno. Namembno podjetje je zasedeno in v bližini prav tako ni nobenega prostega parkirišča.«, kar je bil tudi pričakovan odziv metode. Stanje v tabeli VehicleLogs po 4. scenariju prikazuje slika 6.18.

	ID	vehicleId	currentGeofen...	companyId	xPos	yPos
▷	8	1	2	2	46,51913	15,68007

Slika 6.18: Tabela VehicleLogs po 4. scenariju

Razvidno je, da se je posodobil podatek o trenutni geo-meji v kateri se nahaja vozilo, kot tudi lokacija samega vozila ter ID namembnega podjetja.

Zaključimo lahko, da je bilo testiranje uspešno, saj metoda deluje pravilno v vseh različnih scenarijih.

7 SKLEP

Sistemi sledenja in vodenja vozil so v informacijski dobi nepogrešljiv pripomoček deležnikom transportne panoge. Strankam dajejo vpogled o lokaciji njihovega tovora, podjetjem pa pregled nad položajem in stanjem vozil, možnost komunikacije z voznikom in preusmerjanje na manj obremenjene transportne poti.

Kot smo ugotovili, vse obstoječe rešitve, ki smo jih obravnavali zelo dobro prikazujejo stanje vozila in osnovne podatke o vozilu. Nekateri izmed sistemov vključujejo tudi kopico najrazličnejših pomožnih funkcionalnosti (stanje goriva, zaklepanje vozila). Vendar pa menimo, da se bodo z nenehnim povečevanjem obsega tovornega prometa, sistemi za sledenje in vodenje vozil morali povezati z lokalnimi podjetji in parkirišči, da bodo zagotovili celovito reševanje gneče v prometu, ki nastaja zaradi rasti števila tovornih vozil.

Zato smo v diplomskem delu predlagali rešitev, ki bi ta problem reševala z uporabo tamponskih parkirišč, ko bi v namembnem podjetju zmanjkalo prostora za tovorna vozila. Sistem bi vodil kapaciteto prostih mest v posameznem podjetju in na registriranih parkiriščih za tovorna vozila in vozilo ob pravem trenutku ustrezno preusmeril na pravo mesto. Vozilo bi preusmeril, ko bi le-to vstopilo v območje geo-meje, ki bi bile pametno postavljene v okolju (izvozi iz avtocest, glavne ceste blizu podjetja).

Rešitev smo implementirali kot spletno storitev WCF, kamor tovorno vozilo kot vhodne parametre vsakih nekaj sekund pošilja svojo lokacijo in pa ID namembnega podjetja. Če sistem zazna, da je podjetje v geo-meji blizu namembnega podjetja, ga ustrezno usmeri na prosto mesto.

Slabost take rešitve so dodatni stroški za podjetje, saj bi morali vložiti v nakup tehnologije, ki bi zaznavala prosta mesta na njihovih nakladalno/razkladalnih rampah. Prav tako bi bilo potrebno v lokalnih okoljih zgraditi parkirišča za tovorna vozila in infrastrukturno ustrezno opremiti obstoječa.

Menimo, da bi z uporabo takšne rešitve močno povečali varnost v prometu, saj tovorna vozila ne bi več parkirala na pločnikih in drugih javnih površinah. Hkrati pa bi se zmanjšala gneča na cestah, saj vozila ne bi več krožila po območju in iskala prostih mest za parkiranje.

VIRI IN LITERATURA

[1] EASYTRACKER. Rešitve sledenja. 2012. Dostopno na: https://www.easytracker.si/si/resitve-sledenja/ [20. 7. 2018].

[2] Entity Framework Tutorial. What is Entity Framework? 2018. Dostopno na: http://www.entityframeworktutorial.net/what-is-entityframework.aspx [26. 7. 2018].

[3] FAA. GNSS Frequently Asked Questions – GPS. 2016. Dostopno na https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/faq/gps/#4 [8. 7. 2018].

[4] Gakstatter, E. What Exactly Is GPS NMEA Data? 2015. Dostopno na: http://gpsworld.com/what-exactly-is-gps-nmea-data/ [9. 7. 2018].

[5] GPS. The Global Positioning System. 2017. Dostopno na: https://www.gps.gov/systems/gps/ [8. 7. 2018].

[6] KJE-SI. Sledenje vozil. 2018. Dostopno na: https://kje-si.si/sledenje-vozil/ [20. 7. 2018].

[7] Microsoft. Introduction to the C# Language and the .NET Framework. 2015. Dostopno na: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework [26. 7. 2018].

[8] Microsoft. Visual Studio overview. 2018. Dostopno na: https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide [26. 7. 2018].

[9] Microsoft. What Is Windows Communication Foundation. 2017. Dostopno na https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf [26. 7. 2018].

[10] NASA. Global Positioning System History. 2012. Dostopno na: https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html [8. 7. 2018].

[11] Pavšič, G. Počivališča ob avtocestah: koliko časa še kaos tovornih vozil? 2017. Dostopno na: https://siol.net/avtomoto/zgodbe/pocivalisca-ob-avtocestah-koliko-casa-se-kaos-tovornih-vozil-437538 [5. 7. 2018].

[12] Pinto, D. What is GLONASS? Is it Better Than GPS? 2017. Dostopno na: https://www.techworm.net/2017/08/what-is-glonass-better-gps.html [9. 7. 2018].

[13] Rouse M. geo-fencing (geofencing). 2018. Dostopno na: https://whatis.techtarget.com/definition/geofencing [19.7. 2018].

[14] Rouse M. Microsoft SQL Server. 2017. Dostopno na: https://searchsqlserver.techtarget.com/definition/SQL-Server [26. 7. 2018].

[15] Terzič, M. Avtocestna počivališča: Niso bila zgrajena za današnji obseg. 2010. Dostopno na: https://www.dnevnik.si/1042378840 [5. 7.2018].

[16] Track.si. Funkcije sistema. 2018. Dostopno na: http://track.si/funkcije-sistema/ [20. 7. 2018].

[17] White, S. K. What is geofencing? Putting location to work. 2017. Dostopno na: https://www.cio.com/article/2383123/mobile/geofencing-explained.html [19. 7. 2018].