UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

ZAKLJUČNA NALOGA

ZAKLJUČNA NALOGA

IZDELAVA APLIKACIJE ZA PRIKAZOVANJE NEVARNIH CESTNIH ODSEKOV V REPUBLIKI SLOVENIJI

ANDREJ GODEC

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Izdelava aplikacije za prikazovanje nevarnih cestnih odsekov v Republiki Sloveniji

(Developing application for displaying dangerous road sections in the Republic of Slovenia)

Ime in priimek: Andrej Godec
Študijski program: Računalništvo in informatika
Mentor: doc. dr. Branko Kavšek
Somentor: dr. Branko Bogunovič

Koper, september 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Ključna dokumentacijska informacija

Ime in PRIIMEK: Andrej GODEC

Naslov zaključne naloge: Izdelava aplikacije za prikazovanje nevarnih cestnih odsekov v Republiki Sloveniji

Kraj: Koper

Leto: 2013

Število listov: 46 Število slik: 10 Število tabel: 4

Število prilog: 3 Število strani prilog: 5 Število referenc: 16

Mentor: doc. dr. Branko Kavšek

Somentor: dr. Branko Bogunovič

UDK:

Ključne besede: programski, produkt, aplikacija, cestni, odseki, nesreče

Izvleček:
V zaključni nalogi bo predstavljen razvoj spletne aplikacije za prikazovanje nevarnih cestnih odsekov v Republiki Sloveniji. Glavni cilj aplikacije je seznaniti se z nevarnimi cestnimi odseki, in to na podlagi preteklih prometnih nesreč. S tem želimo voznike opozoriti na previdnost na določenih cestnih odsekih. Cestni odseki bodo urejeni glede na točnost in verodostojnost določenih atributov. Na voljo imamo podatke od leta 1995 do 2011, vendar smo se zaradi sprememb in posodobitev cestnega omrežja osredotočili le na podatke med leti 2007 in 2011. Podatki o prometnih nesrečah so zbrani in dostopni na spletni strani policije Republike Slovenije. Namenjeni so statistični obdelavi in so javno dostopni. Pri razvoju programskega produkta bo na praktičnem primeru povzeto teoretično znanje, ki je osnova za uspešno realizacijo končnega produkta. Zaključna naloga bo služila kot dokumentacija za nadaljnje delo.

Key words documentation

Name and SURNAME: Andrej GODEC

Title of final project paper: Developing application for displaying dangerous road sections in the Republic of Slovenia

Place: Koper

Year: 2013

Number of pages: 46 Number of figures: 10 Number of tables: 4

Number of appendices: 3 Number of appendix pages: 5 Number of references: 16

Mentor: Ph.D. Branko Kavšek

Co-Mentor: Ph.D. Branko Bogunovič

UDC:

Keywords: software, product application, road, sections of an accident

Abstract:
The main objective of this research is the development of web application that will graphically present spatial and temporal variation of traffic accidents and thus identity dangerous road sections, based on past accidents. Road sections will be filtered according to the accuracy and veracity of choosen attributes. Data sets are available from 1995 to 2011, but due to the changes and modernization of the road network, we have focused only on the data between 2007 and 2011. Data on traffic accidents are collected and published on the website of the Police of the Republic of Slovenia. They are available to public, with purposes of statistical analysis. Alongside the software product development the theoretical knowledge will be used as basis on a case study which is significant for the successful comletion of the final product. This thesis will be used as documentation for further development.

Zahvala

Zahvalil bi se rad somentorju, dr. Branku Bogunoviču za pomoč pri oblikovanju in izdelavi zaključne naloge ter mentorju dr. Branku Kavšku.
Zahvalil bi se svoji družini, ki mi je vsa ta leta študija stala ob strani tako moralno kot finančno.
Rad bi se seveda zahvalil še osebju in sošolcem na UP FAMNIT, ki so mi nudili pogoje za pridobivanje dodatnega znanja in pomoči.
Zahvaljujem se tudi Ireni Kavrečič Holcman, prof. za lektorski pregled zaključne naloge.

Kazalo vsebine

Seznam tabel

Seznam slik

Seznam prilog

Seznam kratic

GIS	Geografski informacijski sistemi
GIST	Generalized Search Tree ali v slovenščini posplošeno iskalno drevo
GIN	Generalized Inverted Tree ali v slovenščini posplošeno obrnjeno drevo
GPS	Global Positioning System ali v slovenščini globalni sistem pozicioniranja
GURS	Geodetska uprava Republike Slovenije
LINUX	Odprto-kodeni operacijski sistem
OSM	OpenStreetMap ali v slovenščini odprto-kodni ulični zemljevid
PostgreSQL	Podatkovna baza
Python	Programski jezik
SQL	Structured Query Language ali v slovenščini strukturirani povpraševalni jezik
SP − GiST	Space-Partitioning Trees Generalized Search Tree ali v slovenščini prostorsko usmerjeno generalizirano iskalno drevo
UML	Unified Modeling Language ali v slovenščini poenoteni jezik modeliranja

1 UVOD

Življenja si dan danes brez svetovnega spleta sploh ne moremo predstavljati. Z njegovim prihodom se je začela informacijska doba, kjer so postale informacije gonilo programskih produktov. Vloga programskih produktov v post-industrijski dobi igra ključno vlogo pri prenosu informacij in zagotavljanju kakovosti storitev. Z njimi se srečamo na vsakem koraku, in to ob uporabi osebnih, tabličnih računalnikov, pametnih telefonov in televizij. V zadnjih letih se je s prodorom pametnih telefonov segment trga aplikacij razširil na mobilne naprave in s tem omogočil uporabo novih orodij pri razvijanju in zadovoljevanju potreb uporabnikov. Aplikacije lahko iz boljše preglednosti delimo po kategorijah. Mednje sodijo tudi take, ki nas opozarjajo o nevarnostih, kot na primer aplikacija o obveščanju pred neurji, ki nas pravočasno obvesti o bližajočem se neurju in morebitni ujmi.

1.1 Motivacija

Motivacijo za izdelavo zaključne naloge smo dobili na podlagi relativno velikega števila prometnih nesreč na naših cestah, ki se končajo s smrtnim izidom in hudimi telesnimi poškodbami. Vzrokov za prometne nesreče je veliko, kot je neprilagojena hitrost, nepravilna smer vožnje, neupoštevanje pravil o prednosti, neustrezna varnostna razdalja, vožnja pod vplivom alkohola ter ostale naravne in tehnične podrobnosti, kot so vremenske razmere, neustrezno cestišče in mnogi drugi dejavniki. Kljub posodobitvam cestnega omrežja in sodobnim tehnologijam v avtomobilski industriji nam števila prometnih nesreč ni uspelo drastično zmanjšati. Država je poskušala ukrepati z različnimi represivnimi sredstvi, kot so postreni nadzor in večje denarne globe za prometne prekrške, vendar neuspešno. V primeru, da bi voznika predhodno obvestili o nevarnih odsekih, na katerih se je v preteklosti zgodilo veliko prometnih nesreč, obstaja verjetnost, da bi voznik omenjeni del previdneje odvozil in se s tem izognil morebitni prometni nesreči.

1.2 Cilji

Cilji zaključnega dela predstavljajo dokumentacijo same aplikacije in sistem za iskanje ter grafično upodobitev kritičnih cestnih odsekov na podlagi števila prometnih nesreč. Postavitev sistema ima za osnovo podatke o prometnih nesrečah med leti 2007 in 2011, na podlagi katerih določimo kategorizacijo cestnih odsekov glede na število prometnih nesreč in s tem ustrezno definiramo stopnjo kritičnosti. Na podlagi trenutne lokacije se bodo uporabniku izrisale ceste v njegovi bližini in ga s pomočjo enostavnega in za uporabnika prijaznega grafičnega vmesnika opozorile na potencialne nevarne odseke. Avtomatizacija bo uporabniku zagotavljala hiter dostop do ključnih informacij.

1.3 Struktura zaključne naloge

V naslednjih poglavjih si bomo pogledali definicijo problema, s katerim se srečujemo. Zatem bomo prešli na študijo izvedljivosti, kjer si bomo pogledali obstoječe rešitve. Nadaljevali bomo z analizo in definicijo zahtev, s čimer bomo postavili teoretične temelje samega projekta. Zatem bo sledila še realizacija s poglavjema načrtovanje in izvedba, kjer se bomo lotili praktične izvedbe. Zaključili bomo s testiranjem aplikacije, kjer bomo izpostavili morebitne napake, težave in dodatne funkcionalnosti, ki bi izboljšale delovanje. Za konec bomo še podali svoje mnenje o morebitnem nadaljnjem razvoju in vzdrževanju aplikacije.

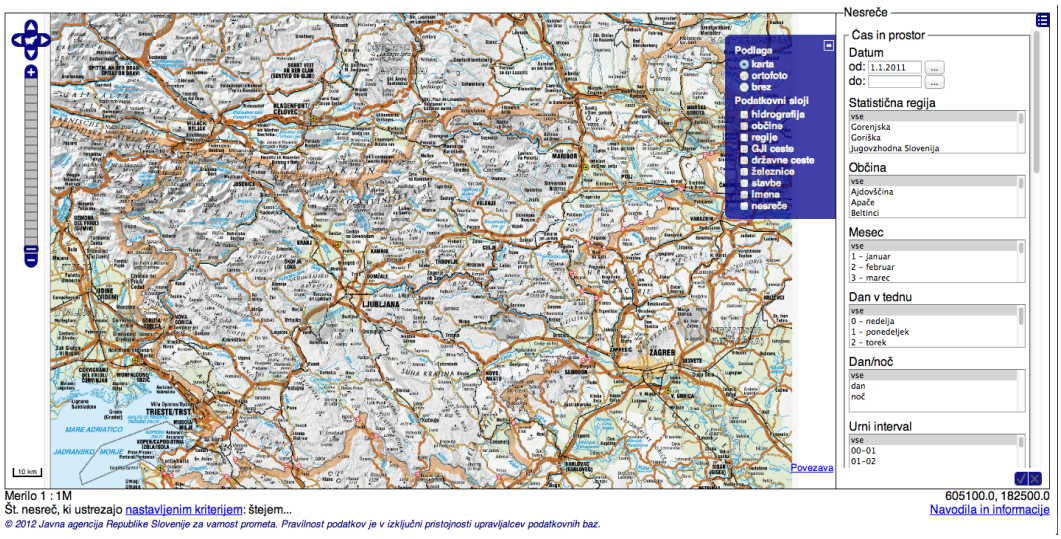
2 DEFINICIJA PROBLEMA

2.1 Opis problema

Z razvojem omenjenega programskega produkta želimo voznike predhodno obvestiti o nevarnih cestnih odsekih. Ne glede na vremenske razmere, stanje cestišča in vozila ter ostale udeležence v prometu je vsak voznik sposoben varno odpeljati potencialno nevarne cestne odseke, le v primeru, da bi ga o nevarnostih predhodno obvestili.

2.2 Obstoječe rešitve

Pri pregledu obstoječih aplikacij in rešitev smo ugotovili, da ni spletne aplikacije, ki bi nam prostorsko prikazala nevarne cestne odseke. Sicer podobna rešitev že obstaja, prikazana na sliki 2.1, vendar je za končnega uporabnika neprijazna in nepregledna, saj je namenjena predvsem splošnemu pregledovanju podatkov o prometnih nesrečah na cestah in avtocestah [13].



Slika 2.1: Spletna aplikacija, namenjena pregledovanju podatkov o prometnih nesrečah na področju Republike Slovenije

3 ŠTUDIJA IZVEDLJIVOSTI

3.1 Koristi projekta

Projekt je zasnovan za širok spekter uporabnikov. Mednje sodijo tako vozniki kot tudi pristojni organi. Voznikom motornih vozil bo sistem omogočil varno vožnjo po slovenskih cestah in avtocestah. S tem bi zmanjšali število prometnih nesreč. Koristi bi imeli prav tako pristojni organi (policija, agencija za varnost v prometu), saj bi z manjšim številom prometnih nesreč bilo tudi manj telesnih poškodb in nesreč s smrtnim izidom. Posledično bi bilo na cestah manj urgentnih voženj. Bolnišnice bi bile razbremenjene pacientov, udeleženih v prometnih nesrečah.

3.2 Izvedljivost

Na razpolago za realizacijo projekta imamo dobre tri mesece. Ob upoštevanju predhodno določenih časovnih rokov bo projekt izveden pravočasno. Denarna sredstva so omejena, zato bomo pri realizaciji projekta uporabljali izključno odprto-kodno programsko opremo. Projekt bom izdelal s pomočjo somentorja.

Projekta se bomo lotili z zbiranjem in vnašanjem podatkov, ki so dostopni na spletni strani policije. Nato bo sledil vnos podatkov v podatkovno bazo, kjer bomo ustrezno prečistili določene podatke. Zaključili bomo z grafičnim prikazom nevarnih cestnih odsekov.
Soočili se bomo z dvema ključnima problemoma:

1. V atributu ceste imamo poleg cest tudi podatke o ulicah, kar nam otežuje ločevanje med cestami in ulicami.

2. Povezava zemljevida z našimi podatki v podatkovni bazi.

Glede na to, da so podatki javno dostopni na svetovnem spletu, ne bomo kršili zakonov.

4 ANALIZA IN DEFINICIJA ZAHTEV

V fazi analize in definicije sistema se bomo osredotočili na uporabnikove zahteve in pričakovanja. Uporabili bomo objektno orientiran pristop, ki ga bomo predstavili s pomočjo UML diagramov in razčlenitve analize na dva dela [4].

UML ("Unified Modeling Language") ali Poenoteni jezik modeliranja je družina grafičnih oznak, podprtih z meta-modelom, ki pomagajo pri opisovanju in oblikovanju programskih sistemov, zlasti programske opreme, ki se jo gradi s pomočjo objektno usmerjenega pristopa. UML diagrame delimo glede na njihov namen v dve skupini:

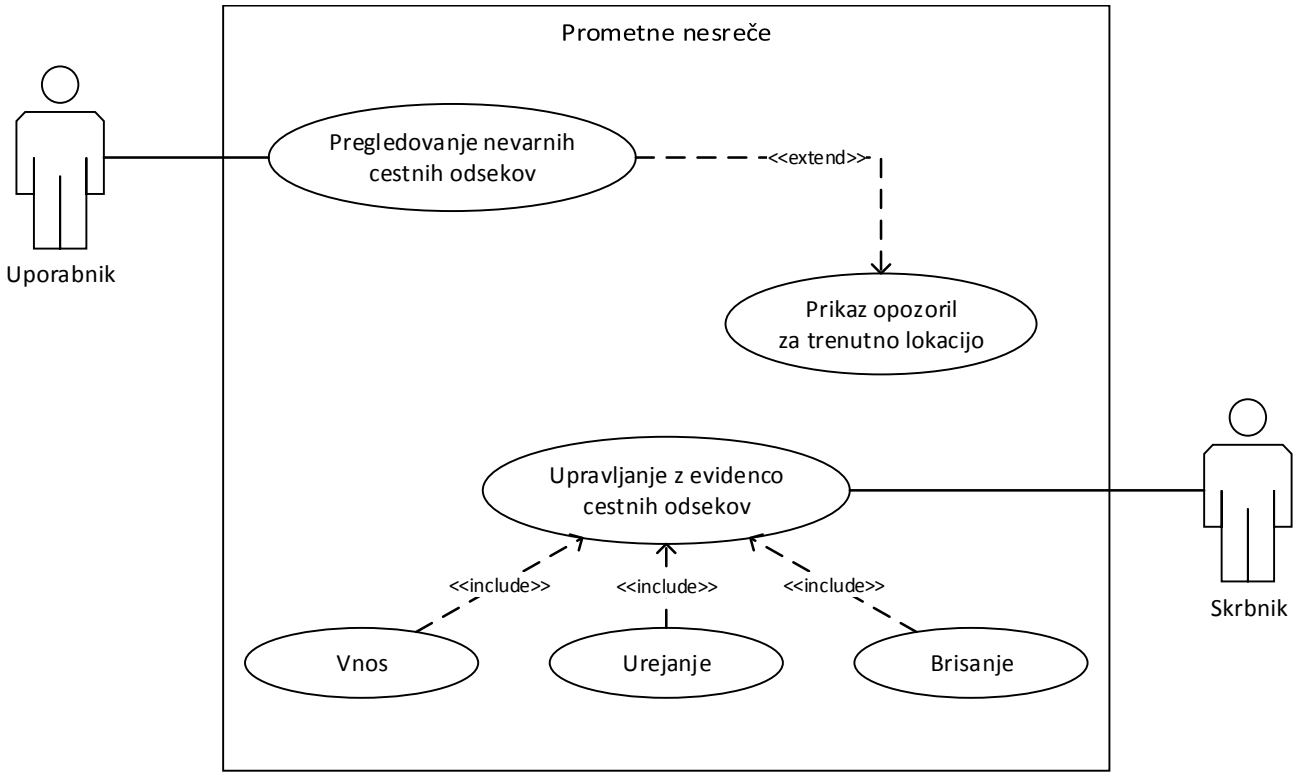
- Strukturne (razredni, komponentni, objektni in paketni diagram)
- Vedenjske (aktivnosti, sekvenčni, primer uporabe, komunikacijski, časovni in diagram stanj)

Pri predstavljanju interakcije s sistemom bodo zlasti koristni UML diagrami [3]. Nato bomo zahteve obravnavali in obdelali na funkcijske in nefunkcijske. Med funkcijske zahteve uvrščamo primere uporabe uporabnika s samim sistemom, kjer je podrobno opisana interakcija na človeku najbolj razumljiv način [1]. Medtem ko pa na drugi strani imamo nefunkcijske zahteve, ki ju ločimo na sledeči lastnosti:

- Obratovalne lastnosti (varnost, uporabljivost, zahtevana prepustnost sistema)
- Evolucijske lastnosti (možnost testiranja, zmožnost vzdrževanja, razširljivost, nadgradljivost)

Nefunkcijske zahteve se sprva ne izražajo kot samoumevne, ampak se jih, slej ko prej občuti pri fazi načrtovanja in izvedbi.
Uporabnikove zahteve bodo modelirane na podlagi vseh pridobljenih informacij s pomočjo preučevanja zgoraj navedenega primera aplikacije, na študiji izboljšav, predlogov in nasvetov strokovnjakov z danega področja ter študijem literature, ki igra ključno vlogo pri sami analizi kot tudi pri projektu.

4.1 Funkcijske zahteve



Slika 4.1: Diagram uporabe za prometne nesreče

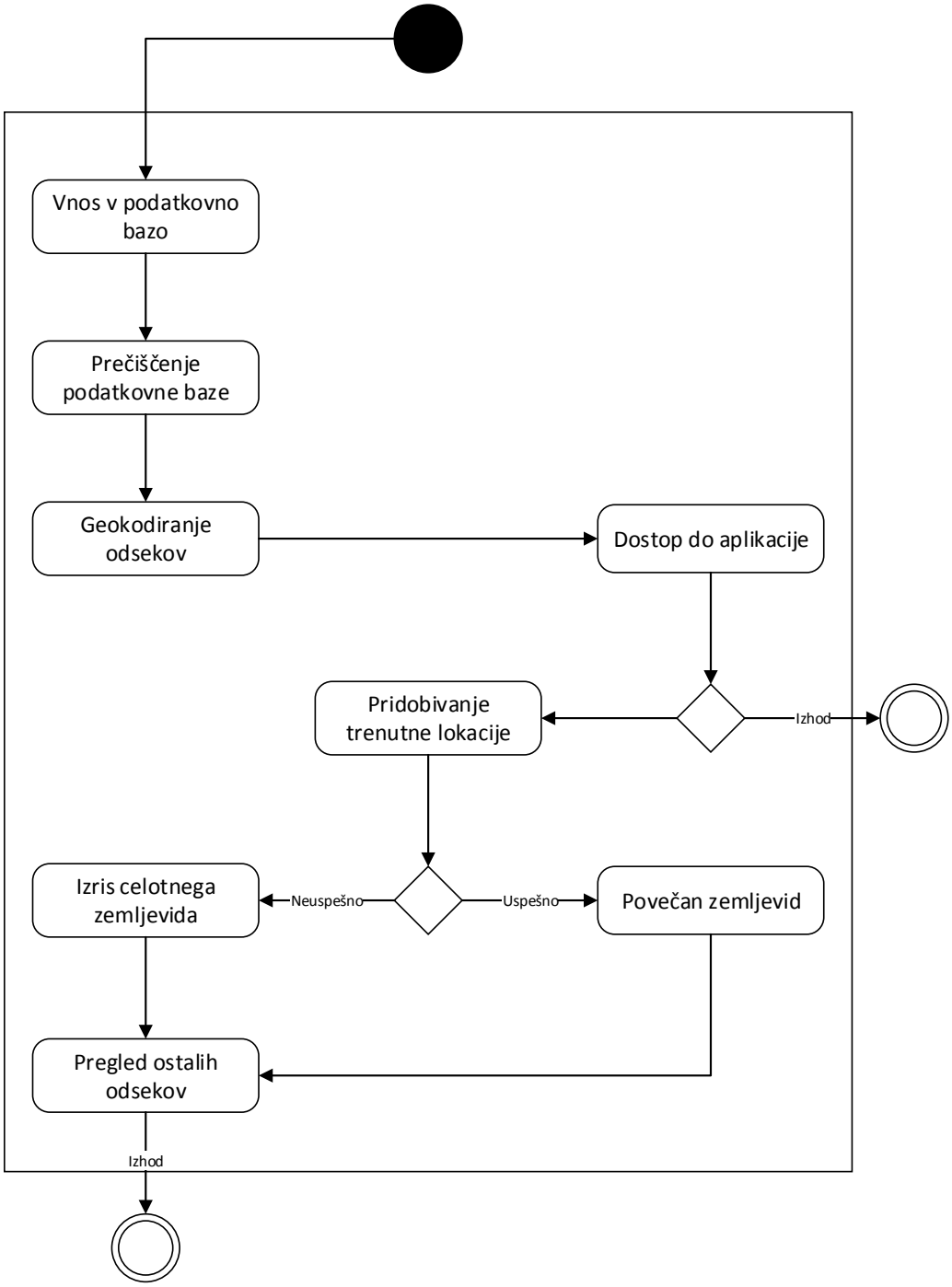
Diagram primera uporabe na sliki 4.1 nam nazorno predstavlja interakcijo uporabnika s sistemom. Vemo, da je lahko akter "voznik" kdorkoli, vendar smo se zaradi lažjega razumevanja omejili na specifičnega uporabnika. Skrbnik je oseba z administrativnimi pravicami, ki ima dostop in nadzor nad podatkovno bazo, kjer so shranjeni podatki o prometnih nesrečah.

Sistem je zamišljen za čim bolj preprosto uporabo. Uporabnik mora imeti na dosegu le potrebne informacije. V primeru, da mu nudimo široko paleto izbirnosti atributov, uporabniku otežimo hitro in enostavno posredovanje informacij ključnega značaja.

Opis poteka:

Uporabnik spletne aplikacije "Pregledovanje nesreč" ima na razpolago pregledovanje prometnih nesreč, ki so se pripetile na območju Republike Slovenije. Te se mu izrišejo na zaslonu na pregleden in minimalističen način.

Aktivnostni diagram



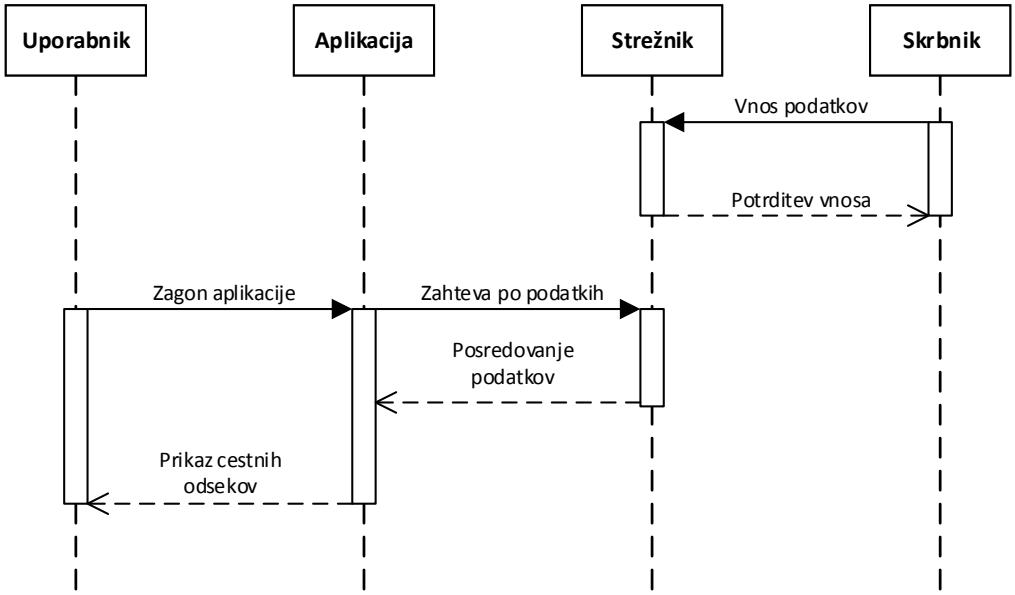
Slika 4.2: Diagram aktivnosti za prikaz prometne nesreče

Aktivnostni diagram na sliki 4.2 nazorno prikazuje potek aktivnosti delovanja aplikacije.

1. Skrbnik vnese podatke o nesrečah v podatkovno bazo.

2. Skrbnik primerno uredi ter prečisti bazo tako, da je primerna za uporabo.
3. Skrbnik geografsko določi lokacije prometnih nesreč.
4. Uporabnik zažene aplikacijo.
   4.1. Uporabnik zapusti aplikacijo.
   4.2. Uporabnik preko aplikacije strežniku posreduje svojo trenutno lokacijo.
      4.2.1. Prikaz približanega zemljevida glede na trenutno lokacijo.
      4.2.2. Prikaz celotnega zemljevida.
5. Pregled vseh ostalih cestnih odsekov.
6. Uporabnik zapusti aplikacijo.

Sekvenčni diagram



Slika 4.3: Sekvenčni diagram nevarnih cestnih odsekov

Sekvenčni diagram na sliki 4.3 nazorno prikazuje zaporedje komunikacije med objekti [3]. Sprva skrbnik naloži podatke o prometnih nesrečah v podatkovno bazo, ki se nahaja na strežniku. Tako so podatki na razpolago uporabniku. Ko uporabnik zažene aplikacijo, le-ta poda zahtevo po podatkih o prometnih nesrečah. Po obdelavi strežnik posreduje podatke aplikaciji, ki uporabniku izriše nevarne cestne odseke.

4.2 Nefunkcijske zahteve

4.2.1 Obratovalne lastnosti

Varnost

Za varnost moramo ustrezno poskrbeti, saj ne želimo, da bi bil kdo oškodovan. Pri tem se moramo ozirati na naše pomanjkljivosti in slabosti. Določene stvari opazimo šele pri testiranju in jih zato rešujemo sproti. Tako bomo že predhodno poskrbeli za varnost pri določitvi uporabniških skupin in njihovih pravic, ki jim bodo dodeljene.

Uporabljivost

Moto za čim širši spekter uporabnikov je angleški izraz "Keep it simple" ali po slovensko "Majhno je dosti". S tem mislimo na popoln avtomatizem same aplikacije, tako da uporabniku ponudimo to, kar resnično potrebuje in želi. Ne želimo ga obremenjevati z nepotrebnimi informacijami, ki povprečnega človeka odvračajo od same uporabniške izkušnje in s tem naredijo aplikacijo s časom tudi kompleksnejšo za vzdrževanje, česar pa ne želimo.

Zahtevana prepustnost sistema

Ob predpostavki, da želimo nuditi storitev stotim uporabnikom hkrati, bo za to zadostovala dvajset megabitna linija ter povprečni moderni računalnik nižjega srednjega cenovnega razreda.

4.2.2 Evolucijske lastnosti

Možnost testiranja

Možnost testiranja je ena ključnih lastnosti, saj s tem opazimo, kje se pojavijo napake in druge anomalije. Več o testiranju bomo opisali v poglavju Testiranje.

Zmožnost vzdrževanja

Če želimo zagotoviti kakovosten programski produkt, moramo upoštevati možnost razvoja glede na spreminjajoče potrebe in biti moramo sposobni vzdrževati sistem, da ne uide izpod nadzora. Ključnega pomena je tudi držati korak s časom, saj moramo produkt nenehno spreminjati in prilagajati glede na okolico in tehnološke spremembe, zato da aplikacija ne zastara. Za urejanje in nadzorovanje bo zadolžen skrbnik sistema.

Razširljivost

Sistem mora biti razširljiv za dodatne funkcionalnosti, ki nam omogočajo širšo uporabljivost sistema. Kot razširitve načrtujemo uporabo različnih slojev, ki bodo poenostavili in polepšali uporabniško izkušnjo, poleg tega pa bi s pomočjo različnih orodij lahko nenehno dopolnjevali aplikacijo, da bi bila čim lepša in enostavnejša za uporabo.

Nadgradljivost

Ker nesreča nikoli ne počiva, se prometne nesreče dogajajo vsakodnevno. Smiselno bi bilo, da v sistem nenehno vnašamo nove prometne nesreče. S tem bi bilo poskrbljeno za ažurnost in širšo uporabo same aplikacije. V naši izvedbi omenjena funkcionalnost zaenkrat še ne bi bila prisotna.

5 NAČRTOVANJE

V tej fazi načrtujemo sistem in aplikacije. Pogledali si bomo izbrana orodja in tehnologije pristopa za kasnejšo izvedbo projekta. Začeli bomo s predstavljenimi programskimi ogrodji, ki bodo uporabljena. Sledila bo zasnova podatkovne baze, nato izvedba povezovanja le-te z grafičnim vmesnikom.

5.1 Uporabljena programska oprema

Kot smo že omenili, bo v nadaljevanju uporabljena izključno odprto-kodna programska oprema, in to zaradi samih omejenih sredstev in zaradi tega, ker želimo podpirati odprto-kodne rešitve, ki so ravno tako dobre kot marsikateri komercialni produkt, če ne celo boljše.

5.1.1 Linux

Linux je odprto-kodni operacijski sistem z določenimi deli prenesenimi iz Unix sistema, razvit in poimenovan po Linusu Torvardsu, ki ga je daljnega leta 1991 razvil kot študent. Poleg Applovega Mac OS X in Microsoftovega Windowsa predstavlja zelo dobro alternativo, saj je popolnoma odprto-koden. Linux je v svojem pomenu besede le jedro operacijskega sistema. Imamo različne "glavne" distribucije Linuxa, kot so Arch Linux, Debian, Fedora, Gentoo, Slackware, openSUSE, Univention Corporate Server. Med njimi najbolj je popularna Ubuntu distribucija, ki izhaja iz Debiana. Za razvojem Ubuntuja stoji angleško podjetje Canonical pod vodstvom Marka Shuttlewortha. Operacijski sistem Ubuntu ima redni cikel posodobitev na vsakih šest mesecev, torej dvakrat letno meseca aprila in oktobra. Vsaki dve leti pa izide verzija z dolgoročno podporo (Long term support - LTS) [5]. Za naše potrebe smo se odločili za 64-bitno verzijo strežniškega Ubuntu 12.04 LTS operacijskega sistema z dolgoročno podporo. Poleg vsega naštetega ima tudi dobro podporo skupnosti, ki skrbi za redno zalaganje s programi v ubuntujevem repozitoriju.

5.1.2 PostgreSQL

Za izbor podatkovne baze, s katero bomo operirali podatkovne vnose (urejali, filtrirali, sortirali, dopolnjevali, brisali, vzdrževali), smo izmed razširjenih odprto-kodnih baz kot so Cassandra, Couchbase, Hadoop, MongoDB, MySQL, MariaDB, Neo4j, Riak, Redis, izbrali podatkovno bazo PostgreSQL, saj edina omogoča obdelavo in dobro integracijo za geografske podatke, s katerimi bomo delali.

PostgreSQL je odprto-kodni sistem za upravljanje relacijskih podatkovnih baz, ki je začel svojo pot kot projekt Univerze Berkeley v Kaliforniji leta 1982. Ta je bil prvotno pod licenco BSD, zdaj pa spada pod okrilje PostgreSQL licence. Ima dolgo zgodovino, ki sega skoraj do začetka relacijske baze podatkov [6]. Trenutna aktualna različica je 9.2. Poleg odprto-kodnosti nam omogoča dodatne funkcionalnosti in razširitve. V našem primeru je uporaben predvsem zaradi razširitve PostGIS, ki je predstavljena v nadaljevanju. PostgreSQL ima poleg že vgrajenih programskih jezikov (SQL, pgsql in C) tudi možnost razširitve na ostale programske jezike za pisanje funkcij in kasnejšo uporabo. Mednje med drugimi štejemo tudi Javo, Python, Ruby, Perl.
Podatkovna baza je sama po sebi sestavljena iz PostgreSQL aplikacije, katero lahko upravljamo preko vmesnika z ukazno vrstico. Za lažje upravljanje lahko uporabimo različna grafična ogrodja. Ta se na podlagi baze poveže z vmesnikom. Namizna aplikacija pgAdmin je namenjena grafičnemu prikazu na določenem operacijskem sistemu, medtem ko pa je na drugi strani spletni grafični vmesnik phpPgAdmin, ki deluje neodvisno od zahtev operacijskega sistema, saj za svojo uporabo potrebuje le spletni brskalnik. Narejen je v programskem jeziku php, ki teče na strežniku ali računalniku, kjer je nameščen [9].

PostGIS

PostGIS razširja funkcionalnost v objektni relacijski bazi PostgreSQL za podporo geografskim predmetom. PostGIS v bistvu omogoča PostgreSQL strežniku "prostorsko " uporabo podatkovne baze za geografske informacijske sisteme (GIS) [15].
Razvoj PostGIS-a se je začel v Refractions Research kot projekt v odprto-kodno prostorsko tehnologijo baz podatkov. PostGIS je izdan pod GNU General Public License. Kot lahko vidimo v tabeli 5.1, je PostGIS še v fazi nadaljnjega razvoja, saj mu razvijalci nenehno dodajajo in dopolnjujejo funkcionalnosti [7]. PostGIS ne služi samo podatkovnim bazam, temveč tudi strežnikom, ki prikazujejo prostorske podatke, s tem da omogoča povezavo med strežnikom in podatkovno bazo.

Verzija	
PostGIS	PostgreSQL
1.3	7.2-7.4, 8.0-8.4
1.4	8.2-8.4
1.5	8.3-8.4, 9.0-9.1
2.0	8.4, 9.0-9.2
2.1	9.0-9.2

Tabela 5.1: Podprta verzija PostGIS glede na verzijo PostgreSQL

5.1.3 Python

Python je sodoben, visoko nivojski programski jezik, primeren za veliko število različnih programskih opravil. Pogosto je naveden kar kot "skriptni jezik", čeprav dandanes ta razlika ni pomembna. Python je uporabljen za pisanje spletnih sistemov, namiznih aplikacij, iger, znanstvenih programov ter celo programov javnih služb in drugih delov na višjih ravneh različnih operacijskih sistemih. Python podpira širok nabor programskih tehnik, od enostavnih postopkov programiranja, pa vse tja do objektnega in funkcionalnega programiranja [10]. Poleg vsega naštetega Python zasledimo tudi na vseh popularnejših operacijskih sistemih in za delo v geografskih informacijskih sistemih.

5.1.4 Apache Tomcat

Apache Tomcat (ali enostavneje Tomcat, prej znan tudi pod imenom Jakarta Tomcat) je odprto-kodni spletni strežnik, ki ga je razvil(o) Apache Software Foundation (ASF). Tomcat implemenitra Java Servlet in JavaServer Pages (JSP) specifikacije iz Sun Microsystems in zagotavlja zagon programskega jezika Jave preko strežnika, ki zna pretvoriti Javo v HTTP obliko. Več o tem lahko preberemo v referenci [12].

5.1.5 GeoServer

GeoServer je odprto-kodna programska oprema strežnika, napisana v Javi, ki omogoča uporabnikom, da delijo in urejajo geoprostorske podatke. Zasnovan za večuporabnost objavlja podatke iz kateregakoli glavnega vira prostorskih podatkov z uporabo odprtih standardov. Kot projekt namenjen skupnosti se GeoServer razvija, testira in posodablja s strani raznih skupin posameznikov in organizacij iz vsega sveta. GeoServer je referenčna izvedba Open Geospatial Consortium (OGC), Web Feature Service (WFS) ter Web Coverage Service (WCS) standardov, kot tudi visoko zmogljive storitve

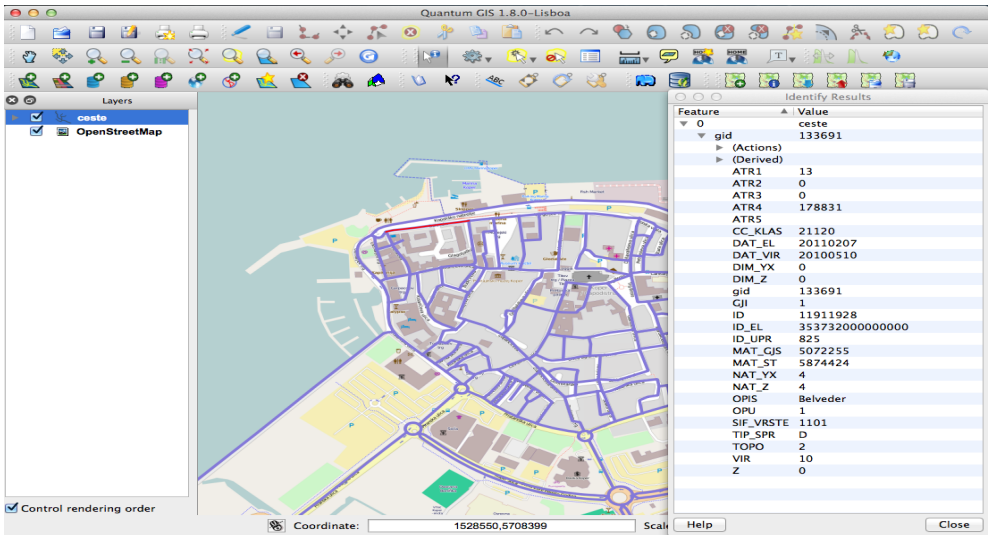
Web Map Service (WMS). GeoServer predstavlja osrednjo komponento geoprostorskega spleta [11].

5.1.6 OpenLayers

OpenLayers je odprto-kodna JavaScript knjižnica za prikaz zemljevidov v spletnih brskalnikih. To zagotavlja API za izgradnjo bogatih spletnih geografskih aplikacij podobnim Google Maps ali Bing Maps. Knjižnica je prvotno temeljila na Prototype JavaScript okvirju [8].

5.1.7 Quantum GIS

Quantum GIS (ali krajše poimenovan QGIS) je odprto-kodni namizni program, ki ima široko podporo na vseh operacijskih sistemih. Narejen je za uporabno v geografskih informacijskih sistemih (GIS), ki nam omogočajo veliko funkcionalnosti: pregled podatkov, pregledovanje in urejanje vektorskih in rasterskih slik ter izvajanje statistike. Poleg tega ima dobro integracijo s PostgreSQL bazo. QGIS nam nudi tudi širok spekter različnih razširitev ter funkcij. Omogoča tudi SQL poizvedbe ter pisanje funkcij v programskem jeziku Python [16].
Za testno orodje prikazano na sliki 5.1 in med samo izvedbo aplikacije bomo uporabili Quantum GIS. Za njega smo se odločili zato, ker nam omogoča uvoz "shape-fileov" v PostgreSQL.



Slika 5.1: Primer uporabe QGIS-a z vektorskima slojema Open Street Maps (OSM) ter GURS-ovih cest s prikazom atributa na izbranem cestnem odseku, prikazanim z rdečo barvo.

5.2 Načrtovanje podatkove baze

Podatke za načrtovanje podatkovne baze smo dobili na spletni strani policije [14], kjer sta na razpolago dve datoteki. To sta datoteki dogodki in osebe. Prometne nesreč, ki se vodijo za vsako osebo smo izločili, saj niso pomembne za prostorski prikaz prometnih nesreč. Podatki v datoteki osebe vsebujejo naslednje atribute (19 atributov):

1. Številka zadeve - to je enoznačna številka zadeve, pod katero policija vodi posamezno prometno nesrečo.
2. Klasifikacija nesreče glede na posledice.
3. Upravna enota, na območju katere se je zgodila prometna nesreča.
4. Datum nesreče.
5. Ura nesreče.
6. Indikator, ali se je nesreča zgodila v naselju ali izven.
7. Kategorija ceste, na kateri je prišlo do nesreče.
8. Oznaka ceste ali šifra naselja, kjer je prišlo do nesreče.
9. Tekst ceste ali naselja, kjer je prišlo do nesreče.
10. Oznaka odseka ceste ali šifra ulice, kjer je prišlo do nesreče.
11. Tekst odseka ali ulice, kjer je prišlo do nesreče.
12. Točna stacionaža ali hišna številka, kjer je prišlo do nesreče.
13. Opis prizorišča nesreče.
14. Glavni vzrok nesreče.
15. Tip nesreče.
16. Vremenske okoliščine v času nesreče.
17. Stanje prometa v času nesreče.
18. Stanje vozišča v času nesreče.
19. Stanje površine vozišča v času nesreče.

Atribut	Prvotno	Po fitriranju
Številka zadeve	številka	številka
Klasifikacija nesreče	B, U, H, L, S	H, L, S
Občina	ime občine	isto
Kategorija ceste ali naselja	H, L, N, T, V, 0, 1, 2, 3, 4, 5	H, L, N, V, 0, 1, 2
Opis kraja	Ž, A, C, E, K, M, N, O, P, R, V, Z	C, N
Vzrok	CE, HI, NP, OS, PD, PV, SV, TO, VO, VR	CE, HI, PD, PV, SV, VO, VR
Tip nesreče	ČT, BT, NT, OP, OS, PP, PR, PZ, TO, TV	ČT, BT, NT, OP, PP, PR, PZ, TO, TV
Vreme	D, J, M, N, O, S, T, V	D, J, M, O, S, T, V
Stanje prometa	E, G, N, R, Z	G, N, R, Z
Stanje vozišča	BL, MO, OS, PN, PP, SL, SN, SP, SU	BL, MO, PN, PP, SL, SN, SP, SU
Vrsta Vozišča	AH, AN, AZ, MA, OS	AH, AN, AZ, MA

Tabela 5.2: Filtriranje podatkovne baze glede na izbrani atribut

Opis atributov glede na kratice, podane v tabeli 5.2: Klasifikacija nesreče je razdeljena na:

- B, U = brez poškodbe
- H = hujša telesna poškodba
- L = lažja telesna poškodba
- S = smrt

Kjer bi odstranili B in U iz razloga, ker ne gre za veliko škodo, temveč morda le za kakšno zvito pločevino.
Kategorija ceste je razčlenjena:

- H = hitra cesta
- L,V = lokalna cesta
- T = turistična
- 0 = avtocesta
- 1,2 = glavna cesta
- 3,4,5 = regionalna cesta

Pri kategorizaciji ceste bi bile odstranjene regionalne in turistične ceste, saj za njih nimamo dovolj podatkov.
Opis kraja nesreče ima kot možne vrednosti podane sledeče lastnosti:

- Ž = železniški prehod
- A = avtobusna postaja
- C = cesta
- E = železniško postajališče
- K = kolesarska steza ali pločnik
- M = krožno križišče
- N = naravno okolje
- O = naravovarstveno območje
- P = parkirni prostor
- R = križišče
- T = predor
- V = vlak

- Z = prehod za pešce

Pri opisu bi odstranili vse razen naravnega okolja in ceste, saj nas ostali podatki ne zanimajo.
Vzrok prometne nesreče:

- CE = nepravilnosti na cesti
- HI = neprilagojena hitrost
- NP = nepravilnosti pešca
- OS = ostalo
- PD = neupoštevanje pravil o prednosti
- PR = nepravilno prehitevanje
- PV = premiki z vozilom
- SV = nepravilna stran / smer vožnje
- TO = nepravilnosti na tovoru
- VO = nepravilnosti na vozilu
- VR = neustrezna varnostna razdalja

Izbrisana atributa naj bi bila ostalo in nepravilnosti na tovoru, saj si pod atributom ostalo, ne predstavljamo, kaj bi to bilo.
Tip prometne nesreče:

- ČT = čelno trčenje
- BT = bočno trčenje
- NT = naletno trčenje
- OP = oplaženje
- OS = ostalo
- PP = povoženje pešca
- PR = prevrnitev vozila
- PZ = povoženje živali
- TO = trčenje v objekt
- TV = trčenje v stoječe parkirano vozilo

Vreme na dan prometne nesreče:

- D = deževno
- J = jasno
- M = megla
- N = neznano
- O = oblačno
- S = sneg
- T = toča
- V = veter

Intenzivnost prometa:

- E = neznano
- G = gost
- N = normalen
- R = redek
- Z = zastoji

Stanje vozišča:

- BL = blatno
- MO = mokro
- OS = ostalo
- PN = poledenelo - neposipano
- PP = poledenelo - posipano
- SL = snežno - pluženo
- SN = snežno - nepluženo
- SP = spolzko
- SU = suho

Vrsta vozišča:

- AH = hrapav asfalt / beton
- AN = nevaren asfalt / beton
- AZ = zglajen asfalt / beton
- MA = makadam
- OS = ostalo

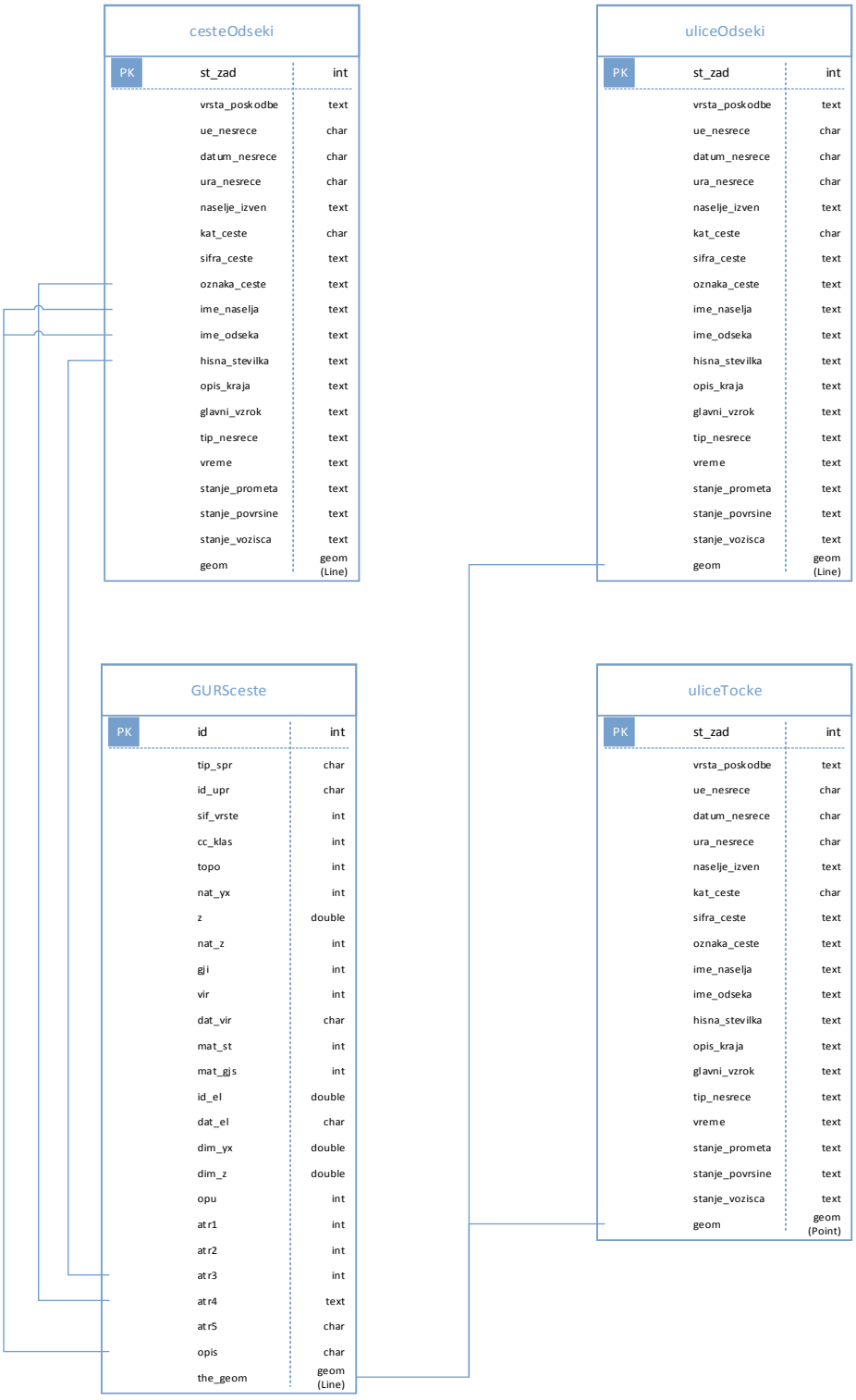
Pri pregledu vseh naštetih atributov smo ugotovili, da so nekateri irelevantni, vsaj za izdelavo naše spletne aplikacije, ampak jih bomo pustili v bazi, saj bodo lahko uporabljeni v nadaljnjih korakih razvoja in dopolnitev.
Podatkovna baza mora imeti tudi naše primarne ključe tabel za lažje iskanje in povezovanje (relacije) med tabelami. Primarni ključ v podatkovni bazi bo številka zadeve, saj je določena enolično. Med drugim bo potrebno iz baze očistiti neveljavne argumente, kot so neznane vrednosti (ostalo, neznano, ...) in prometne nesreče brez fizičnih poškodb, ki so nastale v primeru lažjega trčenja brez večje materialne in telesne škode. Za tem bo sledilo preimenovanje vseh kratic atributov. Poleg tega bo potrebno izbrati indekse za povezovanje tabel, saj na ta način želimo pospešiti poizvedbo. PostgreSQL nam razpolaga sledeče indekse:

- B-drevo
- GIST
- GIN
- SP-GiST
- razpršilni indeks

Za našo spletno aplikacijo smo uporabili GiST (Generalized Search Tree) ali po slovensko (usmerjena generalizirana iskalna drevesa), ki nam omogoča, da porabimo manj časa za zagon določene poizvedbe z SQL stavkom in je geometrično podprt.

5.2.1 ER diagram

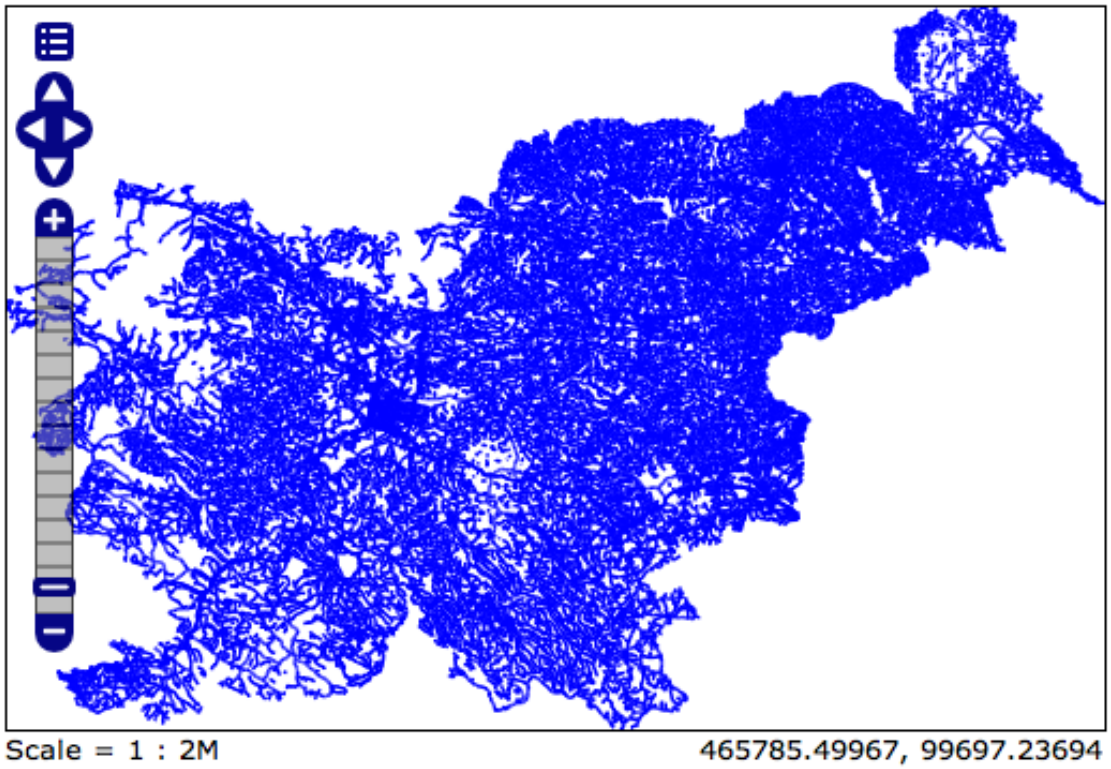
Entitetno-relacijski diagram nam prikaže, kako bomo v fazi izvedbe povezali med seboj tabele in katere tabele bomo imeli za prikaz določenih podatkov.



Slika 5.2: Entitetno-relacijski diagram

5.3 Grafični vmesnik

Ena od ključnih faz projekta je grafični vmesnik, ki bo služil posredovanju informacij končnemu uporabniku. Za grafični vmesnik bo uporabljeno orodje GeoServer, ki bo skrbelo za nemoteno posredovanje podatkov, pridobljenih iz podatkovne baze in bo služil kot vmeseni sloj za povezavo na odprto-kodno orodje OpenLayers. To bo končnemu uporabniku na prijazen način posredovalo želene podatke. Programsko orodje GeoServer je prikazano na sliki 5.3, s prostorsko razporeditvijo cestnega omrežja v Republiki Sloveniji.
Za podlago celovitega sloja bomo izbrali Open_Stret_Map sloj, ki je odprto-kodni.



Slika 5.3: Izgled aplikacije na podlagi GeoServerja, kjer lahko vidimo ceste v Sloveniji, obarvane z modro barvo, s povečavo, merilom in koordinatnim sistemom

6 IZVEDBA

V tem poglavju je predstavljena implementacija aplikacije, kjer bo opisano po principu "od tal k nebu", kar pomeni, da bomo začeli z osnovo postavitve operacijskega sistema, na katerem bomo operirali s podatkovno bazo in GeoServerjem.

6.1 Postavitev zahtevanih orodij

V prvem koraku smo namestili na osebni računalnik operacijski sistem Linux - Ubuntu server 12.04 LTS, ki nam že med samo namestitvijo ponudi možnost opcijske izbire namestitve PostgreSQL podatkovne baze, pri čemer smo potrdili možnost namestitve. Po vzpostavitvi delovanja operacijskega sistema smo nato namestili dodatek PostGIS za podatkovno bazo PostgreSQL 9.2, ki se ga namesti zelo enostavno, saj je v Ubuntujev repozitoriju (sudo apt-get install ...). Za varnost in delo s PostgreSQL je bilo potrebno še nastaviti nastavitveni datoteki *pg_hba.conf* in *postgresql.conf* in nato smo se poizkusili povezati na bazo preko terminalskega okna s sledečim ukazom:

```
psql –h localhost –U postgres
```

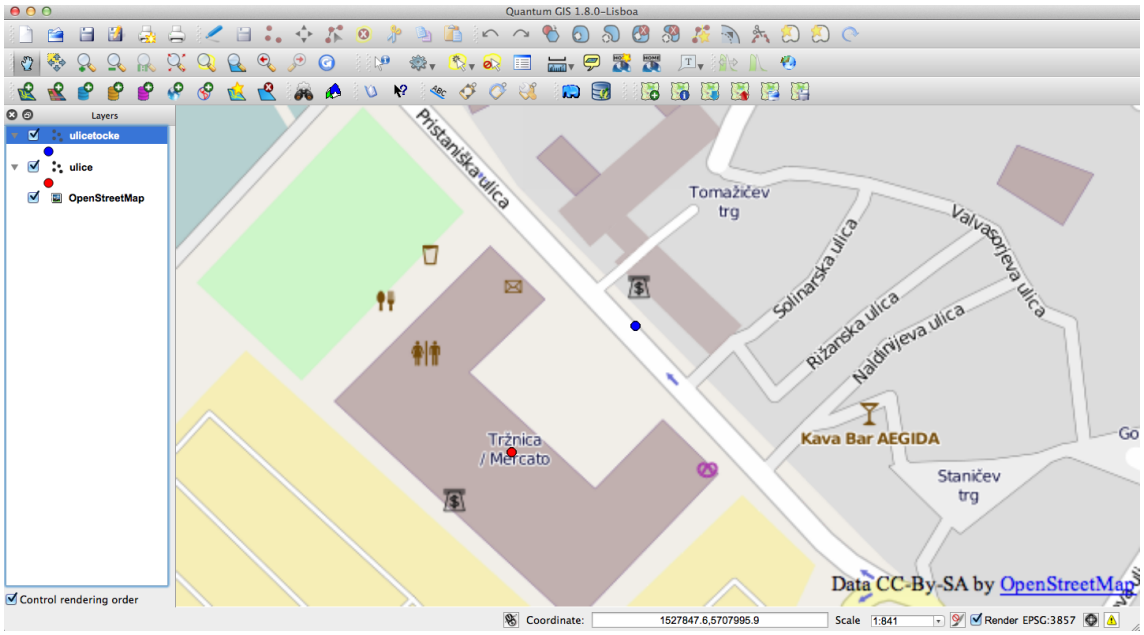
Sledile so namestitve razširitev, kot so: plpythonu, hstore knjižnic za uporabo indeksov. Šele tedaj smo začeli uporabljati podatkovno bazo, kamor smo uvozili, v tabelo imenovano prometne, združene podatke petih let. Potem smo podatke prečistili (odstranili in uredili). Določene atribute smo uporabili za delitev podatkov na ceste in ulice. Preverili smo ulice, ali vsebujejo pravilne naslove za realizacijo geokodiranja s funkcijo, ki je prikazana v priponki A. Za izvedbo geokodiranja ulic so bili izbrani atributi, ki predstavljajo zapis naslova (mesto, ulica, hišna številka). Nato smo kasneje naredili podobno za cestne odseke, ki pa smo jih geokodirali preko "Shapefile". Tega nam je posredoval GURS - Geodestka uprava Republike Slovenije, kjer poleg samega "shapefile"-a vsebuje datoteka še:

- .shp — oblikovni format
- .shx — oblikovni indeks
- .dbf — atribut

Tega smo uvozili v PostgreSQL in cestne odseke povezali po atributu: oznake cestnega odseka, dolžine odseka in opisa.
Potem smo morali popraviti celoletno GURS-ovo mapo v "shapefile", ki smo ga transformirali v koordinatni sistem EPSG 3794 - Slovene National Grid ali slovensko nacionalno mrežo. Premik je bil izveden s pomočjo funkcije *ST_Affine*, ki se nahaja v PostGISu, tako da smo dobili ujemanje z ostalimi "shapefili", kar nam je omogočilo v nadaljnjem koraku povezavo in prekrivanje s sloji iz OSM (Open Street Maps).

Po končani fazi povezovanja prometnih nesreč, storjenih na cestnih odsekih s podatkovno bazo GURS-a, ki vsebuje geokodirane cestne odseke, smo povezali na način: opisa odseka, številke oznake cest in stacionarne točke, kjer se je prigodila prometna nesreča, z različnimi SQL stavki, ki so vsebovali: podstavke, delitev na particije in primerjanje vsake izmed črk vsebovanosti med dvema primerjanima besedama.

Ker izdelujemo aplikacijo za prikaz nevarnih cestnih odsekov, smo ulice, ki jih imamo v točkovni obliki, locirali na najbližjo cesto s pomočjo PostGIS funkcije najbližje točke (*ST_Closestpoint*) in funkcije zbiranja (*ST_Collect*). Na praktičnem primeru s pomočjo QGIS lepo vidimo, kako nam točko iz koprske tržnice, predstavljeno z rdečo piko, umesti na najbližjo cesto, ki je Pristaniška ulica ali promenada in ne na parkirišče, saj smo izbrisali nesreče, ki so se zgodile na parkiriščih.



Slika 6.1: Primer alociranja točke prometne nesreče z ulice (označene z rdečim krogcem) na najbližjo cesto (označeno z modrim krogcem)

Za zaključek temeljnega dela lociranja smo na podlagi prekrivanja cestnih odsekov s cesto geolocirali prometne nesreče, ki so se zgodile v bližini ulic, alocirali na cestne odseke GURS-a s pomočjo funkcij razširjanja točk in funkcije, ki omogoča presek, (*ST_Intersects*), dobili cestni odsek.

6.2 Grafični vmesnik

Grafični vmesnik predstavlja zadnji korak v fazi izvedbe, poleg same podatkovne baze, ki služi za dostop do podatkov. Orodja GeoServer in OpenLayers bodo skrbela za prikazovanje cestnih odsekov in točk. GeoServer predstavlja povezavo med podatkovno bazo in sloji za izris.
Za osnovni sloj bomo uporabili OSM (Open Street Map), saj je odprto-kodni in omogoča lep in enostaven prikaz. Drugi sloji so:

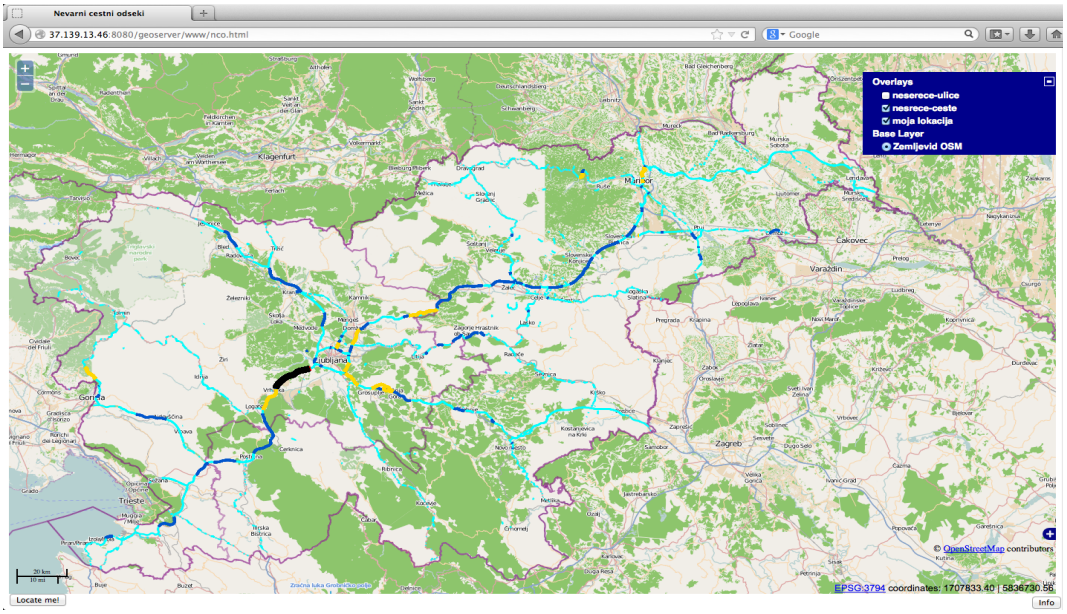
- ulice, ki so predstavljene kot točke;
- ulice, ki so predstavljene kot cestni odseki;
- ceste, ki so predstavljene kot cestni odseki;

Pri OpenLayerjih so uporabljene med drugim še ostale programske komponente:

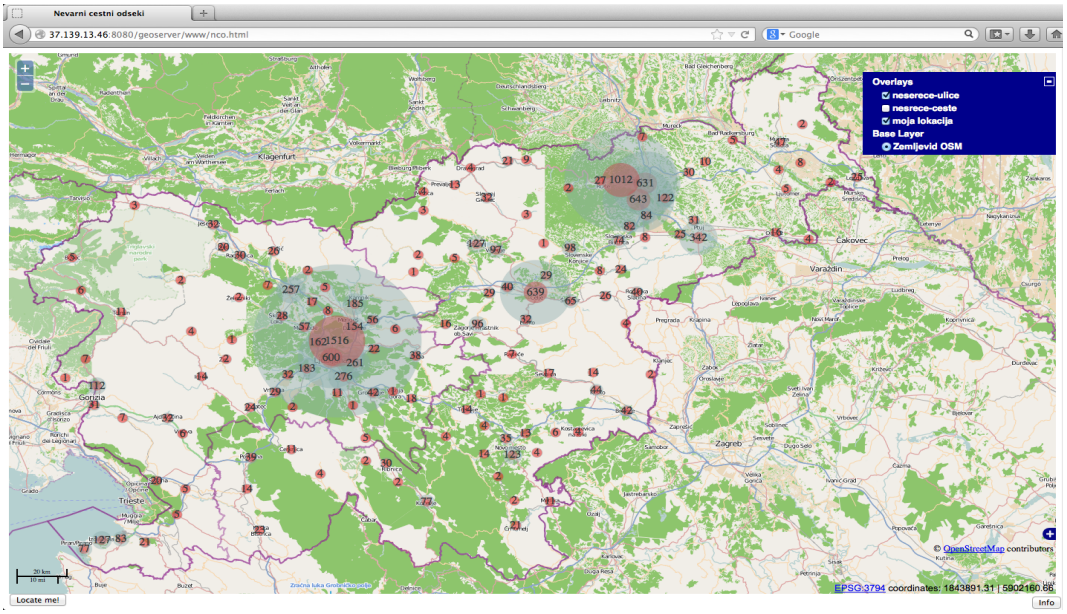
- določanje položaja
- prikazovanje določenih slojev
- prikazovanje vrednosti atributov

Zaradi zagotavljanja čim preprostejše uporabe spletne aplikacije je aplikacija narejena tako, da je čim bolj "prijateljsko" naravnana. Na sledečima slikama 6.2 in 6.3 lahko vidimo končno verzijo aplikacije. S pomočjo spletnega naslova enostavno pridemo do aplikacije, ki izriše zemljevid in komponente v celotni velikosti odprtega okna. Levo v zgornjem kotu imamo dva gumbka (plus in minus), ki predstavljata povečavo ali pomanjšavo zemljevida. V desnem spodnjem kotu je na razpolago gumbek za določanje našega trenutnega položaja. Na levi strani ravno sta prav tako dve komponenti. Prva se nahaja zgoraj in nam prikazuje sloje, ki jih lahko izberemo. Le osnovnega sloja ne moremo odstraniti. Ta komponenta je vidna že pri dostopu na spletno stran, medtem ko je druga komponenta postavljena nižje in je uvodoma skrita ter prikaže pomanjšavo zemljevida za lažjo orientacijo. Cestni odseki so prikazani s petimi barvami, podobno kakor so označene smučarske proge. Relativno malo nesreč je predstavljeno s turkizno barvo, saj bi zelena barva bila premalo vidna. Barve se nato stopnjujejo: temno modra od 10 do 20 nesreč, rumena od 20 do 30 nesreč, rdeča od 40 do 60 in kjer je bilo več

kot 60 nesreč, je odsek obarvan s črno, kamor lahko uvrstimo metaforično prispodobo črni odsek.



Slika 6.2: Prikaz sloja s cestnimi odseki



Slika 6.3: Prikaz sloja ulic s pomočjo krogcev združenih v enote

7 TESTIRANJE

Testiranje je pri razvoju našega programskega produkta zadnja faza, kjer pred predajo programskega produkta širši množici uporabnikov, pregledamo, ali vse komponente delujejo pravilno.

Testiranje bo predstavljeno po načrtu, kjer bodo podani opisi komponent in ugotovitve. Potekalo bo na dva načina. Prvi način bo ta, da testira razvijalec sam, drugi način, kjer sodeluje zaokrožena skupina do treh ljudi, torej treh neodvisnih preizkuševalcev.

7.1 Načrt testiranja

Elementi testiranja	Pričakovani rezultat
Geokoordinirane ulice	pravilno geokoordinino
Ujemanje "shp" datotek	prekrivanje
Geokodiranje cestnish odsekov	prekrivanje
Varnost	varen sistem
Prepustnost	do 100 uporabnikov
Pomnilnik	zasedenost

Tabela 7.1: Elementi, ki bodo testirani in pričakovani rezultat

Pri načrtu testiranja smo v večini uporabljali odprto-kodno orodje QGIS, saj nam nudi pri iskanju in odkrivanju napak široko paleto svojih funkcij.
Poleg naštetega orodja smo aplikacijo testirali tudi sami preko spletnega vmesnika.

7.2 Najdene napake in popravki

Napake smo zaradi izčrpnosti filtriranja baze srečevali na vsakem koraku. Največ napak se je pojavilo zaradi neprečiščene podatkovne baze, s katero smo operirali, saj so ulice in cestni odseki bili podani v površni obliki in nam jih je določene napačno geokodirano, ali pa izpisalo ene in iste večkrat.
Večjo težavo je predstavljalo prekrivanje dveh "shape" datotek, torej datoteke pridobljene iz OSM (Open Street Maps) in datoteke pridobljene z GURS-a (Geodetske

uprave Republike Slovenije). Pri prekrivanju smo uporabili zamik GURS-ove datoteke. Po zaključeni fazi smo ugotovi še nekatere napake kot na primer lociranje ulic na napačno cesto. Za primer naj navedemo Prešernovo cesto v občini Izola, kjer nam je določilo geokordinato bližje napačni cesti. Takih primerov je bilo še več. Potreben je popravek in boljše razvrščanje, kar pa zahteva veliko zamudnega dela, saj bi morali pogledati par tisoč točk.
Drugo večjo napako smo ugotovili na odseku Izola - Belvedere, ki pa se je pojavila na cesti Belveder v Kopru, saj smo geokodirali na opis ceste, ker na identifikacijsko številko ceste/odseka nismo uspeli. Poleg tega to ni edini primer, saj je večina lokalnih cest vprašljivih zaradi točnosti podatkov, ki nam jih je posredovala policija.
Tabela 7.2 prikazuje elemente testiranja na podlagi predhodnega določila rezultata in trenutnega rezultata po testiranju.

Elementi testiranja	Pričakovan rezultat	Rezultati testiranja
Geokoordinirane ulice	pravilno geokoordinino	zadovoljivo
Ujemanje “shp” datotek	prekrivanje	v redu
Geokodiranje cestnish odsekov	prekrivanje	zadovoljivo
Varnost	varen sistem	v redu
Prepustnost	do 100 uporabnikov	v redu
Pomnilnik	zasedenost	v redu

Tabela 7.2: Elementi, ki so bili testirani, njihov pričakovani rezultat in rezultat testiranja

8 ZAKLJUČEK IN NADALJNJE DELO

V zaključni nalogi sem predstavil, kako izgleda načrtovanje programskega produkta skozi faze, od same zastavitve problema, pa do izvedbe in testiranja. Na splošno je bilo zahtevno uporabiti enega izmed ustaljenih razvojnih modelov, kot so: "Build & Fix", linearni model, linearni model s povratki, izdelavo prototipa [2]. Naš programski produkt je mešanica vseh teh, kar je posledica sprva nejasno zastavljene analize potreb, nakar je to posledično delovalo kot nekakšna gradnja prototipa z lastnostmi, sprva zastavljenega, linearnega modela. Največ preglavic je seveda povzročila podatkovna baza.

Leta 2014 naj bi slovenska policija dobila tablične računalnike, in tako bi se dalo nastalo aplikacijo razširiti za uporabo popisa prometnih nesreč, saj ima velika večina le-teh vgrajen GPS sprejemnik. Nastala aplikacija bi policiji olajšala delo, saj bi točno geolocirala lokacijo prometne nesreče.
Aplikacijo za vnos prometnih nesreč ter detekcijo nevarnih cestnih odsekov bi se dalo nadgraditi z algoritmi strojnega učenja, ki bi lahko sami avtomatsko predlagali nevarne odseke glede na zgodovino njihovega beleženja.

Literatura

[1] A. Cockburn, *Writing Effective Use Cases*, Addison-Wesley Professional, 2001. *(Citirano na strani 5.)*

[2] M. Cohn, *Agile Estimating and Planning*, Prentice Hall, 2005. *(Citirano na strani 27.)*

[3] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language, Third Edition*, Addison-Wesley Professional, 2003. *(Citirano na straneh 5 in 8.)*

[4] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, Addison Wesley Professional, 2004. *(Citirano na strani 5.)*

[5] C. Negus, *Linux Bible 2010 Edition: Boot Up to Ubuntu, Fedora, KNOPPIX, Debian, openSUSE, and 13 Other Distributions*, Wiley, 2010. *(Citirano na strani 11.)*

[6] R. Obe in L. Hsu, *PostgreSQL: Up and Running*, O'Reilly Media, 2012. *(Citirano na strani 12.)*

[7] R. Obe in L. Hsu, *P. Ramsey*, PostGIS in Action. Manning Publications; Pap/Psc edition, 2011 *(Citirano na strani 12.)*

[8] A. S. Perez, *OpenLayers Cookbook*, Packt Publishing, 2012. *(Citirano na strani 14.)*

[9] S. Riggs in H.Krosing, *PostgreSQL 9 Admin Cookbook*, Packt Publishing, 2010. *(Citirano na strani 12.)*

[10] E. Westra, *Python Geospatial Development*, Packt Publishing, 2010. *(Citirano na strani 13.)*

[11] B. Youngblood in S. Iacovella, *GeoServer Beginner's Guide*, Packt Publishing, 2013. *(Citirano na strani 14.)*

Viri

[12] Apache Tomcat `http://tomcat.apache.org` 2013 *(Citirano na strani 13.)*

[13] Javna agencija Republike Slovenije za varnost v prometu `http://nesrece.avp-rs.si` 2013 *(Citirano na strani 3.)*

[14] Policija Republike Slovenije `http://www.policija.si` 2013 *(Citirano na strani 15.)*

[15] PostGIS 2.0 on Ubuntu 12.04 `http://trac.osgeo.org/postgis/wiki/UsersWikiPostGIS20Ubuntu1204` 2013 *(Citirano na strani 12.)*

[16] Quantum GIS Project `http://www.qgis.org` 2013 *(Citirano na strani 14.)*

Priloge

Priloga A: Geokodiranje

Funkcija, ki nam geokodora prometne nesreče storjene na ulicah s pomočjo programskega jezika Python v PostgreSQL z SQL stavkom.

```
1  CREATE OR REPLACE FUNCTION dogodki.geocoding(param_sql text,
       param_db text)
2    RETURNS SETOF dogodki.address2lonlat AS
3  $BODY$
4  import psycopg2
5  import sys
6  from time import sleep
7  from pygeocoder import Geocoder
8
9  con = None
10
11 try:
12     con = psycopg2.connect(database=param_db, user='postgres',
           password='postgres', host='localhost', port='5432' )
13     cur = con.cursor()
14     cur.execute(param_sql)
15
16     rows = cur.fetchall()
17
18     for row in rows:
19         results = Geocoder.geocode("%s" % ",".join(map(str, row
               [0:])))
20         sleep(0.1)   # Wait for a while(100ms) to get respond from
               Google API
21         # Concatenate finalResult: [Address,Lon,Lat]
22         yield("%s" % ",".join(map(str, row[0:])),results[0].
               coordinates[1],results[0].coordinates[0])
23
24
25 except psycopg2.DatabaseError, e:
26     print 'Error %s' % e
27     sys.exit(1)
28
```

```
29 finally:
30
31     if con:
32         con.close()
33 $BODY$
34   LANGUAGE plpythonu VOLATILE
35   COST 100
36   ROWS 1000;
37 ALTER FUNCTION dogodki.geocoding(text, text)
38   OWNER TO postgres;
```

Priloga B: SQL stavki

Primeri začetnih SQL stavkov, ki so bili uporabljeni pri delu s podatkovno bazo.

```
1  --Uvoz vseh podatkov zdruzenih v eno datoteko
2
3  -- Naredimo Shemo
4  CREATE SCHEMA dogodki
5
6  -- Create table for saving all tabings
7  CREATE TABLE dogodki.pn(st_zad int,
8          vrsta_poskodbe char(1),
9          ue_nesrece char(4) NOT NULL,
10         datum_nesrece char(10),
11         ura_nesrece char(2),
12         naselje_izven text,
13         kat_ceste char(1),
14         sifra_ceste char(5),
15         oznaka_ceste int,
16         ime_naselja text,
17         ime_odseka text,
18         hisna_st text,
19         opis_kraja text,
20         glavni_vzrok char(2),
21         tip_nesrece char(2),
22         vreme char(1),
23         stanje_prometa char(1),
24         stanje_povrsine char(2),
25         stanje_vozisca char(2));
26
27 -- Kopiranje iz datoteke v bazo
28 -- Jezik kodiranja UTF8 ali latin1
29 SET client_encoding = 'UTF8';
30 COPY dogodki.pn
31       FROM '/Users/Shared/PN.TXT' DELIMITER AS '$';
32
33
34 -- iskanje neznanih atributov, za kasnejsi izbris
35
```

```
36 -- vrsta poskodbe
37 select * from dogodki.pn where vrsta_poskodbe='B' or vrsta_poskodbe
      ='U'
38
39 -- obcina (MNZ,neznano,izdano v drugi obcini)
40 select ue_nesrece from dogodki.pn group by ue_nesrece order by
      ue_nesrece desc
41
42 -- kategorija ceste
43 -- select kat_ceste from dogodki.pn group by kat_ceste
44 select * from dogodki.pn where kat_ceste='T' or kat_ceste='3' or
      kat_ceste='4' or kat_ceste='5'
45
46 -- kraj dogodka (C in N ostane)
47 select opis_kraja from dogodki.pn2 group by opis_kraja
48
49 -- glavni vzrok
50 select * from dogodki.pn where glavni_vzrok='NP' or glavni_vzrok='
      OS' or glavni_vzrok='TO'
51
52
53 -- tip_nesrece
54 select * from dogodki.pn where tip_nesrece='OS'
55
56 -- vreme
57 select * from dogodki.pn where vreme='N'
58
59 -- stanje_prometa
60 select * from dogodki.pn where stanje_prometa='E'
61
62 -- stanje_povrsine
63 select * from dogodki.pn where stanje_povrsine='OS'
64
65 --stanje_vozisca
66 select * from dogodki.pn where stanje_vozisca='OS'
67
68
69 -- delitev ulic in cest
70 select * into dogodki.cest from dogodki.pn where ime_naselja like
      '%-%'
71 select * into dogodki.ulice from dogodki.pn where ime_odseka not
      like '%-%'
```

Priloga C: Programska oprema

Priložena je zgoščenka (Compact Disk ali CD), s programsko opremo in ostalimi stvarmi kot so: poizvedbe, funkcije, itd.