SVEUČILIŠTE U ZAGREB
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Marko Smetko

APLIKACIJA ZA EVIDENCIJU PUTNIH NALOGA

ZAVRŠNI RAD

Mentor:

Dr. sc. Tihomir Orehovački, viši asistent

Varaždin, rujan 2015.

SVEUČILIŠTE U ZAGREB

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Marko Smetko

Izvanredni student

Broj indeksa: Z37171/08 IZV

Smjer: Primjena informacijskih tehnologija u poslovanju

Preddiplomski stručni studij

APLIKACIJA ZA EVIDENCIJU PUTNIH NALOGA

ZAVRŠNI RAD

Mentor:

Dr. sc. Tihomir Orehovački, viši asistent

Varaždin, rujan 2015.

Sadržaj

1. Uvod

Ovaj rad bavi se problematikom razvoja aplikacije za generiranje i arhiviranje putnih naloga te njima pripisanim troškovima. Sastoji se od praktičnog dijela u kojem se stvara funkcionalna aplikacija koja je iskoristiva u praksi i teoretskog dijela u kojem je opisan proces nastanka aplikacije sa pripadajućim detaljima i opisima.

U teoretskim dijelu opisuje se putni nalog kao dokument. U tom dijelu istražuje se svrha i važnost putnog naloga. Prikazuje se njegova struktura, koje su stavke obavezne te koje opcionalne. Pretpostavlja se kako je putni nalog pravdajući dokument upotrijebljen u stvarnim slučajevima i pretpostavlja se doza odgovornosti koju donosi korištenjem. Nakon teoretske osnove dokumenta, pogled se usmjeruje na postojeća rješenja na tržištu, kako bi se dobila ideja o smjeru u kojem se razvijala aplikacija u ovom projektu. U ovim koracima ne radi se kopija postojećih aplikacija, već se na vlastiti način realizira jedinstvena ideja koja je inspirirana provjerenim rješenjima korištenih od stvarnih ljudi u stvarnim situacijama.

Kod definiranja smjera razvoja aplikacije, programsko sučelje u kojem će se razvijati buduća aplikacija mora biti opisano. Programsko sučelje podrazumijeva razvojni program za programiranje, sustav za upravljanje bazama podataka i alate za obradu grafike. Svaki program koji arhivira dokumente i za potrebe stvaranja novih koristi kataloške vrijednosti, treba imati bazu podataka. Njezin dizajn, kao i odnosi u njoj, dobivaju se iz stavki dokumenata koji se žele stvoriti u aplikaciji, pri čemu se mora obratiti pažnja da su zapisi u njoj napisani uz najmanju moguću redundanciju koja ne narušava funkcionalnost same aplikacije. Kad je baza razvijena, trebalo je napisati algoritme kako bi se moglo korisniku omogućiti funkcionalno i ugodno iskustvo s aplikacijom. Za svaki isplanirani algoritam potrebno je opisati kako će se pretvoriti u izvorni kod koji podrazumijeva davanje funkcije prisutnim elementima u aplikaciji. Nakon pisanja koda, potrebno je isplanirati testiranje i opisati zamišljene nadogradnje koje se vežu uz praktični dio zadatka. U ovom radu slijedili su se opisani koraci.

Praktični dio sastoji se od izrade baze podataka i stvaranja aplikacije koja je opisana u teoretskom dijelu. Za izradu baze korišten je sustav za upravljanje bazama podataka, te plan i shema koji su objašnjeni u teoretskom dijelu. Općenito, baza koja se izrađuje mora sadržavati sve potrebne entitete i atribute kako bi bila pouzdana i kako joj se ne bi naknadno trebala mijenjati struktura. Kad postoji pouzdana baza, tada se počinje s izradom aplikacije. Njezina izrada objašnjena je u teoretskom dijelu rada. Jednom kad je aplikacija u potpunosti

funkcionalna, pažnja se usmjeruje na vizualne detalje koji podrazumijevaju stilove teksta, pozadinske slike, boje, veličinu pojedinih elemenata i njihov raspored.

Svaku izrađenu aplikaciju potrebno je ispitati, za što su potrebni dobrovoljci koji će je detaljno testirati. Ispitivanje se obavlja direktno na aplikaciji, a do zaključaka se dolazi promatranjem i interakcijom s dobrovoljcima. Ispitivanjem se saznaje da li je u praktičnom radu aplikacija stabilna i kakav subjektivni utjecaj ostavlja na korisnika. Iz tih saznanja ispravljaju se eventualni propusti i planiraju potencijalne buduće nadogradnje. Ideje za nadogradnje se, nakon ispitivanja, opisuju u teoretskom dijelu koji je u ovom radu posljednja i zaključna komponenta.

2. Prikupljanje informacija

Planirajući aplikaciju, počinje se od izgleda dokumenta putnog naloga. Budući da provjerena rješenja već postoje na tržištu, za pomoć pri dizajnu dokumenta uzeti su stvarni primjeri. Dokumenti koji su uzeti za primjer su putni nalozi iz aplikacija poduzeća Interbiz-a, Relago-a i PBZ-a. Polazi se od pretpostavke da su zajedničke stavke na različitim putnim nalozima obavezne, a one koje nisu zajedničke su opcionalne. Informacije dobivene od primjera dokumenata nisu dovoljna garancija o ispravnosti dokumenata. Kako bi se osigurala ispravnost dokumenta potrebno je provjeriti da li je dokument u skladu sa zakonom i pravilnicima u Republici Hrvatskoj. Putni nalozi podliježu Zakonu o putnim ispravama, Zakonu o porezu na dohodak i Pravilniku o troškovima službenog putovanja. Zakoni i pravilnici koji se odnose na dokumente dostupni su svim građanima na Internetu.

Kada su poznati svi detalji oko dokumenata, iz konkurentskih aplikacija uzimaju se ideje za upis podataka i interakciju s korisnikom. Postojeće aplikacije temelje se na principu dohvata stavaka iz ugrađenih kataloga podataka pa se za razvoj ove aplikacije isto koristi sličan princip. Iz aplikacija viđenih u uredima Hrvatske Pošte i FINA-e viđeno je da djelatnici sve podatke unose brzo putem tipkovnice. Iz toga se može zaključiti da je miš komponenta koja kod upisa podataka negativno utječe na brzinu i ergonomiju upisa te bi se trebao kod uvesti kao alternativa, a način unosa preko tipkovnice trebao bi biti primarni način unosa.

Kada je na temelju otprije navedenih podataka dizajnirana baza podataka te su napisani algoritmi koji se žele implementirati, treba se doći do znanja nužnih za pisanje koda. Budući da se za razvoj ove aplikacije koristi Microsoft-ova verzija Visual Basic-a, onda je logično da se koristi njihova podrška. Za potrebe programiranja informacije se uzimaju iz Microsoft-ove baze znanja koja se zove MSDN. Ona je praktični izvor informacija jer u sebi ima ugrađene algoritme za pretraživanje informacija o elementima, skriptama, strukturama, naredbama, ugrađenim funkcijama i procedurama itd. Osim toga sadrži primjere na konkretnim slučajevima gdje se može vidjeti kako napisati neki dio koda. Za probleme kod konkretnih slučajeva postoji i MSDN forum gdje drugi korisnici Microsoft-ovih razvojnih rješenja iznose svoje probleme, komentare i iskustva kako bi drugi sudionici mogli to koristi kao pomoć. Informacije nađene na MSDN-u su dovoljne za razvoj ovakve aplikacije, spajanje na bazu podataka i implementaciju svih zamišljenih algoritama, jer pokrivaju sva područja s kojima se netko tko razvija ovakvu aplikaciju može susresti.

3. Motivacija za izradu aplikacija

Korištenje vozila u poslovne svrhe, kao i odlazak na poslovni put mora se dokumentirati. To dokumentiranje služi za različite svrhe od vođenja evidencije o vozilima, troškovima puta, statistici, pravdanja korištenja vozila itd. Najvažniji dokument kod dokumentiranja je putni nalog. On sam po sebi može imati mnogo stavaka što radnicima, koji ih ispunjavaju, uzima mnogo vremena, pogotovo ako se ispunjava za veći broj vozila. Tu može pomoći aplikacija koja već ima u sebi unaprijed definirane vrijednosti koje se pozivaju kako bi se ubrzao proces upisa podataka u sam nalog, a i smanjila mogućnost pogreške kod upisa.

Na primjer ako se svaki dan koriste isti brojevi vozila i isti vozači, tada je veća mogućnost da se pogrešno ispuni radni nalog kod ručnog upisa, nego kada se vrijednosti odabiru iz kataloga. Elimininiranjem takvih grešaka, poduzeće izbjegne neželjene probleme koji bi inače nastali zbog ljudskog faktora.

Sljedeći problem koje ima ručno vođenje nad računalnim je arhiviranje, pretraživanje i vođenje statistike. Ručno arhiviranje svodi se na čuvanje velike količine papira, što je staromodno i sve više odumire jer nije ekološki, uzima puno više radne snage, manje je pouzdano, svaka radnja radi se ručno, što zbog ljudskog faktora može dovesti do greške, te svako vođenje statistike i pretraživanje podataka je mnogo zahtjevnije u odnosu na računalna rješenja.

S aplikacijom moguće je eliminirati navedene probleme ručnog unosa. Arhiva dokumenata pospremljenih u nekoj bazi podataka ne zauzima toliko mjesta koliko sami papiri. Pretraživanje na računalu je neusporedivo lakše od ručnog pretraživanja jer bilo koji putni nalog na računalu može biti pozvan u zanemarivom vremenskom intervalu.

Statistiku o bilo kojoj stvari koju poduzeće treba, može dobiti pozivom funkcije u aplikaciji koja će brzo i bez greške ljudskog faktora dati željeni rezultat. Takvo dobivanje statistike je mnogo bolja alternativa klasičnom računanju kakvo još uvijek postoji u mnogim poduzećima koja iz navike, financijskih razloga ili nekog drugog razloga ne vode svoje dokumente uz pomoć računala. U svakom slučaju službenih vozila u Republici Hrvatskoj ima dovoljno i tržište za aplikaciju koja bi vodila putne naloge postoji. Što je veće poduzeće sa što više vozila i više službenih ruta koje se dokumentiraju, aplikacija im je isplativija. Cijene aplikacija za vođenje putnih naloga kreću se između 700 i 2000 kuna, što nije visoka cijena kada se uzme u obzir cijena ostale programske potpore. Uzevši u obzir sve nabrojeno zaključuje se da bi razvoj aplikacije za putne naloge bila isplativa investicija.

4. Putni nalozi

4.1 Općenito o putnim nalozima

Putni nalog je dokument koji služi za dokumentiranje službenog putovanja zaposlenika nekog poduzeća. Kako službeno putovanje spada u troškove poduzeća, mora se dokumentirati adekvatnim dokumentom i zbog toga se koriste putni nalozi. Putni nalog sam po sebi sadrži 3 zasebne cjeline.

Prvi i najvažniji dio je putni nalog. U tom dijelu odgovorna osoba mora potvrditi putovanje potpisom i pečatom. Obavezno mora biti navedeno polazište i odredište, ime osobe koja putuje, datum putovanja, tip i registracijska oznaka vozila, dnevnica, predujmovi te svrha putovanja. S tim dokumentom pravda se putovanje i eventualno korištenje službenog vozila. Taj isti dokument je potvrda policiji da je korištenje službenog vozila opravdano, ukoliko dođe do rutinske policijske provjere prometa.

Drugi dio kod putnog naloga je izvještaj o putovanju kojeg ispunjava putnik ili vozač. Kod njega se navodi vrijeme polaska i dolaska, početna i završna kilometraža, opis putovanja, moguće promjene kod putovanja, te se evidentiraju troškovi koji su nastali tokom putovanja kao što su cestarine, mostarine, troškovi goriva, troškovi parkinga, troškovi hotela i noćenja itd. U praksi je to prazan obrazac koji se ručno ispunjava tokom putovanja i prilaže se uz putni nalog i obračun.

Zadnji dio putnog naloga je obračun. On se ispunjava nakon predanog izvještaja o putovanju zajedno sa svim računima i ostalim dokumentima s kojim se pravdaju troškovi. Njega obično ispunjava računovodstvo ili osoba odgovorna za troškove putovanja. Sam obračun se sastoji od dnevnice, predujmova, troškova kilometraže i troškova nastalih tokom putovanja.

4.2 Svojstva i svrha putnih naloga

Putni nalozi su papirnati dokumenti i obavezno ih je nositi tijekom službenog putovanja[Pravilnik o naknadama putnih i drugih troškova na službenom putovanju, Članak 6, 2009.]. Budući da je to službeni dokument i da je zakonski propisana obaveza koristiti ga, on mora sadržavati obavezne i nedvosmislene stavke jer se njegov sadržaj veže uz Zakon o porezu na dohodak i Zakon o putnim ispravama hrvatskih državljana. On služi za kao dozvola za korištenje službenog vozila, ako se koriste te je dokaz službenim tijelima da vozilo nije ukradeno ili zlorabljeno.

Da bi dokument bio ispravan i vjerodostojan u očima službenih državnih tijela, on po zakonu mora sadržavati podatke o tvrtci s obaveznim osobnim identifikacijskim brojem, rutom kretanja i imenom putnika ili vozača, namjerom i potpisom odgovorne osobe u poduzeću[Članak 14. stavka 2 Pravilnika o porezu na dohodak,2006.].

Ukoliko putni nalog nije ispravan državni službenik može korištenje službenog vozila vidjeti kao krađu ili zlouporabu.

Osim samog naloga važno je voditi brigu o izvještaju jer se uz pomoć njega i prikupljenih računa izrađuje obračun. Obračun se izrađuje zato jer se neki troškovi u putnom nalogu mogu pravdati kao porezno priznati trošak. U trenutku pisanja ovog rada, zakoni u Republici Hrvatskoj dopuštaju da se troškovi cestarina, mostarina, tunelarina, hotelskog smještaja kod službenog putovanja uključe u porezno priznate troškove. Isto tako su dnevnice za putovanja dalja od 35 kilometara od sjedišta neoporeziva, kao i cijena kilometraže putovanja, ukoliko ta cijena ne prelazi 2 kune po kilometru. Porezne stavke ne moraju biti navedene u obračunu, ali moraju sadržavati točne bruto podatke o troškovima, te sve pripadajuće pravdajuće dokumente, koji su u pravilu računi, kako bi računovodstvo moglo pravdati trošak. Isto tako u napomeni mora biti navedeno da li je vozač zaposlenik poduzeća ili vanjski suradnik, jer ako je u putnom nalogu naveden vanjski suradnik onda se ne može pravdati na isti način kao kod nekog zaposlenog u poduzeću, već sa zakonski propisanom tarifom koja ovisi o primarnoj djelatnosti poduzeća.[Zakon o porezu na dohodak, Članak 9,točka 10, 2015.]

4.3 Putni nalozi u praksi

U nekom poduzeću kada se planira službeno putovanje ispunjava se putni nalog, njega može sastaviti sam zaposlenik za sebe ili za to odgovorna osoba u poduzeću, ali obavezno mora imati potpis i pečat od nadređenog. U njemu se navodi datum izdavanja dokumenta, datum putovanja, odredište, polazište, namjera putovanja i ukoliko se koristi službeno vozilo, onda obvezan tip vozila i registracijska oznaka. Nakon ispisa i ovjere, zaposlenik koji putuje nosi putni nalog sa sobom zajedno s izvještajem o putovanju.

Pri početku samog putovanja u izvještaj se upiše datum i vrijeme kada je započeto putovanje i početna kilometražu. Tokom putovanja, koristeći službeno vozilo, zaposlenika može zaustaviti službena osoba iz policije, tada uz vozačku dozvolu i prometnu dozvolu mora predočiti putni nalog koji je prethodno potvrđen jer vozilo koji se koristi glasi na poduzeće i dokazuje da je njegovo korištenje opravdano [Zakon o putnim ispravama Hrvatskih državljana,Zakon Republike Hrvatske, 2015].

Tijekom putovanja zaposlenik nailazi na cestarine, mostarine, tunelarine,trošak hotela itd. Svi ti troškovi upisuju se u izvještaj i za njih zaposlenik mora imati račune kako bi se ti troškovi mogli pravdati. Osim troškova upisuju se eventualne promjene u ruti, kao što su zatvorena prometnica, kvar ili elementarna nepogoda.

Pri povratku s putovanja upisuje se vrijeme dolaska i završna kilometraža vozila te se predaje putni nalog zajedno s izvještajem i pripadajućim dokumentima za pravdanje troškova osobi koji radi obračun, ukoliko to ne radi zaposlenik sam za sebe. U obračun se uvrštavaju dnevnice, troškovi kilometraže i ostali troškovi koji se moraju pravdati računima, te od tih stavki odbija se predujam, ukoliko ga je zaposlenik dobio prije putovanja. Nakon obračunavanja troškova, putni nalog, izvještaj i obračun zajedno s pravdajućim dokumentima, potrebno je predati računovodstvu na obradu i arhiviranje.

5. Kritički osvrt na postojeće aplikacije

5.1 Analiza postojećih aplikacija

Kako bi se dobila ideja o tome što je potrebno napraviti kako bi ova aplikacija bila na razini postojećih, mora se istražiti što rade drugi programerski timovi. Na tržištu ima dovoljan broj takvih programa jer mnoga poduzeća u svojem poslovanju koriste putne naloge.

Postojeće aplikacije grubo se mogu podijeliti na dvije skupine. Prva skupina su Windows aplikacije od kojih je većina kao i aplikacija opisana u ovom radu, napravljena u Visual Studio[1]-u. Osim u Visual Studio-u, postoje i aplikacije napravljene u Delphi[2]-u koje zahtijevaju drugi skup potpornih datoteka. Delphi je programski jezik baziran na Pascal-u za razvoj konzolnih,Windows,web i mobilnih aplikacija.[Enbarcadero,2015.,dostupno od 15.Svibanj.2015. na: www.embarcadero.com] .Ali one su rjeđe jer Delphi nije toliko široko rasprostranjen kad se radi o ovakvim relativno malim aplikacijama zbog toga jer Visual Studio nudi mnogo više podrške.

Druga skupina aplikacija su web aplikacije. Bazirane su većinom na ASP[3]-u ili PHP[4]-u potpomognute Javascriptom[5]. Njihov dizajn je u pravilu dorađeniji jer se ipak radi o web stranicama kojima je prikaz prvotna svrha i samim time mogućnosti dizajna su mnogo veće. Osim toga budući da za dizajn koriste kaskadne stilske stranice onda je taj dizajn i brzo prilagodljiv i izmjenjiv. Glavni nedostatak takvih aplikacija je ovisnost o internetskoj vezi bez koje ih je nemoguće koristi.

5.2 Aplikacija Relago putni nalozi

Poduzeće Relago iz Zaprešića razvila je web aplikaciju za izradu i arhiviranje putnih naloga. Napravljena je na Microsoftovoj Azure platformi koja je specijalizirana za kreiranje, distribuiranje i upravljanje Cloud usluga kao što je ova.

Dizajn aplikacije je u bijelo - plavim tonovima, isto kao i dizajn izgleda poduzeća te se po cijeloj strukturi drži tih dizajnerskih smjernica. Ne sadrži kontrastne kombinacije boja i ulazni elementi

[1] Visual Studio - Skupina razvojnih alata i servisa za kreiranje aplikacija na Microsoft-ovoj platformi

[2] Delphi je programski jezik baziran na Pascal-u za razvoj konzolnih,Windows,web i mobilnih aplikacija

[3] ASP- Microsoft-ov programski jezik za stvaranje dinamičkih web stranica

[4] PHP- Neovisni server side programski jezik za stvaranje dinamičkih web stranica

[5] Javascript[5]- Skriptni programski jezik za web programiranje na strani korisnika

su logički poredani na postavljenom formularu. Nedostatak dizajna je taj što nema nikakve podjele u skupine i zbog toga je narušena čitljivost na obrascu za upis.

Ergonomski gledano, aplikacija bi se mogla doraditi . Upis je ovisan o korištenju miša te je poprilično nezgrapno izmjenjivati kontrole s miša na tipkovnicu i obrnuto. To nije efektivno kao „mousefree“ dizajn kakav se koristi u aplikacijama s velikim brojem upisa.

Prednost ovakve aplikacije je to što nije ovisna o Windows operativnom sustavu. Za korištenje aplikacije potreban je samo internet preglednik i pristup Internetu. Budući da se baza podataka nalazi u oblaku, ova aplikacija nije namijenjena korisnicima koji imaju skepsu prema uslugama oblaka i sigurnije se osjećaju kad se sva radnja radi na njihovom računalu.

U osnovnoj izvedbi aplikacija ne dozvoljava ispis u PDF i Word formatu, već se to dodatno naplaćuje. Dodatne mogućnosti dodatno se naplaćuju i ne mogu se na lagan način personalizirati obzirom da se radi o Cloud usluzi koja je napravljena tako da bude univerzalna. Još jedan nedostatak je taj što ne postoji mogućnost da se aplikacija kupi i koristi neovisno, već se iznajmljuje na mjesečnoj bazi. Prednost se nalazi u tome što se podaci drže u oblaku te je poduzeću koje unajmljuje prostor na oblaku garantirana sigurnost podataka, ne postoji mogućnost da kvar na računalu ili zlonamjerni software uništi podatke, a nedostatak je taj što vlasnik oblaka može poduzeću uskratiti pristup ukoliko se usluga ne plati na vrijeme.

Osim svega navedenoga uvijek su mogući hakerski napadi na usluge oblaka i mogućnost zloupotrebe osobnih podataka od strane zaposlenika u poduzeću koje nudi uslugu, ali to je rizik koji se mora prihvatiti ukoliko se odlučuje na ovakvo rješenje.

5.3 Aplikacija Inter-biz putni nalozi

Poduzeće Inter-biz iz Varaždina odlučila se za Windows aplikaciju kao svoje poslovno rješenje za putne naloge. Na svojoj Web stranici nude demonstrativnu verziju programa. Na stranici nije navedeno u kojem razvojnom okruženju je nastala. Ovo rješenje je najsličnije rješenju opisanom u ovom radu jer je napravljeno da funkcionira u Windows okruženju.

Dizajn aplikacije nije na razini današnjih dizajnerskih standarda. Izgled sučelja podsjeća na dizajn Windowsa 95 gdje nema toniranih prijelaza pri čemu je dizajnirano s visokim kontrastom i plosnato. Kao bazu podataka koristi SQL server[6] i ta baza za razliku od baza na web aplikacijama se nalazi na tvrdom disku na računalu te zbog toga aplikacija nije ovisna o Internetu. Osim toga SQL baza aplikaciji omogućuje mrežni rad.

[6] SQL server – Microsoftov sustav za upravljanje bazama podataka

Kontrole i ulazi su logički i pregledno posloženi. Ergonomija programa je poprilično dobra. Manevriranje po kontrolama i ulazima je zadovoljavajuća. Izvještaji programa dizajnirani su konzervativno bez elemenata koji bi se previše isticali ili bi odudarali od sheme boja korištene u ostatku programa. Oni su jednostavni i funkcionalni, što i jest najbitnija stvar kod njih. Ključni podaci na dokumentu su podcrtani tako da se brzo mogu iščitati bez mukotrpnih traženja po dokumentu.

Cijena ovog programa je 800kn+PDV, što u odnosu na ostala rješenja nije pretjerana cijena za takav proizvod, a prednost je što nakon kupljenog proizvoda on ostaje kod kupca aplikacije i dugoročno se takvo rješenje više isplati od aplikacije u najmu. Nedostatak ovakvog rješenja je to što je korisnik ovisan o računalu na kojem je instalirana aplikacija, a problemi mogu nastati kvarom na računalu. U najgorem slučaju ako se pokvari tvrdi disk na računalu kojem je instalirana aplikacija i postavljena baza, teško se može doći do podataka.

5.4 Usporedba razvijene aplikacije s postojećim rješenjima

Analizirajući postojeće aplikacija i potrebe ljudi koji ih koriste, razvoj aplikacije odvijao se u smjeru povećanja omjera dobivenog i uloženog. Iako web aplikacije imaju svoje prednosti, razvijena Windows aplikacija je napravljena tako da nije ovisna o Internet vezi i ne ograničava stranku koja bi ju eventualno kupila s brojem unosa dokumenata, putnika ili korisnika.

Za razliku od mnogih, ova aplikacija može imati neograničen broj korisnika, putnika, vozila, mjesta i dokumenta. Namijenjena je za jednokratnu kupnju i korištenje na računalu i zbog toga je eliminiran problem mjesečnog plaćanja, udaljene baze i eventualnih promjena koje bi se pojavile zbog napretka web tehnologija. Baza se nalazi na računalu korisnika tako da nikakva treća osoba ne može pristupiti podacima poduzeća kao što je slučaj kod web aplikacija.

Ergonomija je mnogo bolja nego kod postojećih aplikacija jer ima dva moda rada koji su jednako prilagođeni stalnom i povremenom načinu korištenju. Upis u formulare je lakši jer je se može učiniti samo upotrebom tipkovnice, ali alternativno može se koristiti i miš za navigaciju. Mogućnost krivog upisa je smanjena na minimum jer se sve moguće važne stavke dohvaćaju uz pomoć formulara za dohvat i svaki korak se provjerava pri čemu se ispravne i neispravne stavke označuju odgovarajućom shemom boja. Dizajn aplikacije je moderniji i pregledniji od aplikacije koju ima Inter-biz i bolje se uklapa u dizajn Windows-a 7 i 8 koji se u trenutku pisanja rada i kreiranja aplikacije koriste.

Za slučaj da se želi napraviti sigurnosna kopija dovoljno je samo kopirati Access bazu podataka na sigurno mjesto kao što je oblak ili eksterna memorija. Takav jednostavni mehanizam sigurnosne pohrane se može automatizirani u samom operativnom sustavu.

Kada potencijalni klijent kupuje program, osim funkcionalnosti, jedan od većih faktora je cijena. Budući da, nakon što je aplikacija napravljena, ne postoje dodatni troškovi s posjedovanjem i održavanjem može se ponuditi poprilično konkurentna cijena u odnosu na postojeće aplikacije. Inter-biz svoju aplikaciju naplaćuje 800kuna+PDV, a kod poduzeća Relago najam aplikacije u svojem najslabijem izdanju, do 5 djelatnika, košta 420 kuna godišnje. Kada se uzme u obzir to što aplikacija u ovom projektu nudi, ako se ponudi po nižoj cijeni za očekivati je da na tržištu bude poprilično konkurentna.

6. Razrada problema

Kad se promatra putni nalog kao dokument, vidi se da nije jednostavan dokument koji se ispisuje odjednom i od jedne osobe, već je složena cjelina od tri dijela koju ispunjava i potvrđuje više osoba. Zbog takvog oblika dokumenta mora se imati i više mehanizama za upis i arhiviranje pojedinih dijelova. Kao arhiva koristi se baza podataka koja mora biti logički strukturirana tako da predstavlja iste dokumente u realnom svijetu i veze među njima.

Ako se počne od dokumenata, oni predstavljaju zasebne entitete u bazi, a stavke koje se ispunjavaju u dokumentima predstavljat će atributi. Osim dokumenta, ostali entiteti su osobe koje putuju, vozila koja se koriste i mjesta s kojih i na koja se ide. Kad je to definirano moraju se napraviti formulari i elementi na formularima takvi da se uz najmanji mogući napor mogu upisivati ili odabirati vrijednosti koje će se ispisati i arhivirati. To mora biti napravljeno po uzoru na programe koji se koriste na Fini ili u poslovnim bankama, jer ti programi su napravljeni za maksimalno brz upis i navigaciju pomoću tipkovnice, pa se korištenje miša svodi na minimum, ali još uvijek postoji ako alternativa.

Kad postoji ideja o tome kako i kamo s upisom, moraju se odrediti ograničenja tko može što upisati u pojedinu stavku i u kakvim ograničenjima mora ta stavka biti. Isto tako mora se ograničiti u koje dijelove programa pojedinci imaju pristup, a u koje ne. To se učini tako da se svakom korisniku daju podaci za prijavu u obliku korisničkog imena i lozinke s kojima će se prijaviti u program i ovisno o korisničkim pravima koja mu dodjeljuje administrator, biti će mu dopuštene ili uskraćene određene funkcije u aplikaciji. Administrator će imati sve funkcije, čitač će moći samo pregledavati arhivu i ispisivati prazne izvještaje, dok će upisivač moći upisivati nove naloge i raditi obračun. Korisnički račun administratora je namijenjen direktoru ili nekoj odgovornoj osobi u poduzeću. Čitač je namijenjen zaposlenicima koji putuju da mogu pratiti vlastite putne naloge i obračune, dok je upisivač namijenjen računovođi u poduzeću, jer su oni obično zaduženi za njihovo stvaranje i obračunavanje.

Osim standardnih korisnika napravljen je programer mod čiji podaci za prijavu se ne nalaze u bazi i on ima sva prava kao i administrator s dodatnim funkcijama gdje će moći pratiti parametre programa u realnom vremenu. Takav backdoor[7] je postavljen iz razloga što u slučaju greške u programu se može vidjeti što ga uzrokuje prateći parametre s kojima rukuje.

Osim generiranja dokumenata, aplikacija zajedno sa svojom bazom mora biti pouzdana arhiva. Otvaranje i pregledavanje dokumenata jedan po jedan nije praktično. Iz tog razloga kod arhiva

[7] Backdoor je programerski ulaz u aplikaciju koji zaobilazi standardnu prijavu korisnika

mora se imati tablični prikaz dokumenata s mogućnošću prikazivanja ključnih stavaka u realnom vremenu kako se prolazi kroz tablicu. Ako se pretpostavi da arhiva može biti poprilično velika, onda samo sirovi tablični prikaz sam po sebi nije praktičan bez mogućnosti filtriranja. Filter mora biti postavljen po identifikaciji dokumenta, datumu i po ključnim stavkama kao što su putnici i mjesta navedena u dokumentu. Iz arhive se treba moći ispisati bilo koji već generirani dokument.

Da bi ispisivali dokumenti trebalo je izraditi adekvatne izvještaje. Oni će se morati generirati pomoću Microsoftovog modula za izvještaje koji je ugrađen u Visual Studio, jer osim .NET frameworka 4.5[8] ne iziskuje nikakve dodatne potporne datoteke. Izvještaji trebaju biti čitljivo dizajnirani i trebaju imati sve podatke kako bi se preko njih mogle obavljati funkcije za koje su namijenjeni. Na njih će se stavke koje se pojavljuju jednom, kao što su identifikacija, putnik, polazište i odredište, ispisivati uz pomoć parametara koje će obraditi sam program, dok će stavke poput ostalih troškova, kojih može biti mnogo, iščitavati direktno iz baze, a u toj bazi će sve vrijednosti, pa čak i brojčane, biti definirane kao tekst tako da ih modul za izvještaje lakše i nedvosmisleno prikaže onako kako bi trebali biti prikazani.

Naposljetku program se treba prilagoditi dvjema vrstama korisnika. Jedni su oni koji program koriste usput, povremeno ili u pozadini, a drugi su oni koji ga aktivno koriste kao glavni dio svog posla pa su zbog toga kreirana dva specifična formulara, i s njima, dva režima rada. Za one koji koriste program povremeno, program se može pokrenuti u mini modu. To je režim rada gdje je prozor manji tako da ne zauzima previše prostora na radnoj površini i većina funkcija će se odmah nalaziti na formularu. Drugi način rada je namijenjen stalnom radu i nazvan je maxi mod. On zauzima cijeli ekran i sve funkcije kategorički su poredane preko padajućih izbornika, a uz to ima i status traku koja prikazuje korisne informacije.

Program je napravljen tako da sve što se radi s bazom, ne radi na način kao sa stalnom vezom, nego vezu po potrebi otvara i zatvara tako da baza može biti ručno otvorena kad se ne radi na programu. Tu se otvara mogućnost daljnje nadogradnje gdje više aplikacija može koristiti tu istu bazu.

[8].NET Framework je instalacijski paket koji postavlja komponente potrebne za rad određenih programa od Microsoft-a

7. Programska podrška

Kako bi se napravila aplikacija dostojna pojave na tržištu mora se koristiti adekvatni razvojni software koji će raditi na operativnom sustavu Microsoft Windows zbog toga jer je to najčešće korišteni operativni sustav u Hrvatskoj. Aplikacija radi na Windows-ima XP, Vista, 7, 8, 8.1 i budućem Windows-u 10. Za razvoj ove aplikacije koriste se većinom Microsoftovi programi za koje je dovoljno instalirati .Net Framework. Kao sustav za upravljanje bazama podataka koristi se Microsoft Access 2010 jer baza ne treba biti kompleksna i mora biti lako prenosiva. Dobra alternativa bi bila SQL server, ali za ovakav projekt jednostavno nema potrebe koristiti kompliciranije programske alate. Kao razvojni alat za izradu same aplikacije koristi se Microsoft Visual Studio 2013 jer je jedan od najmoćnijih alata za razvoj Windows aplikacija. Od mnogih programskih jezika koje podržava radi se u Visual Basic-u jer se u njemu može relativno brzo izraditi aplikacija ovog tipa, isto tako u sebi već sadrži većinu elemenata koji trebaju za izradu aplikacije. Za sam dizajn i grafiku u programu radi se u Adobe Master Collection-u CS6 i to najviše u programu Adobe Photoshop CS6. Grafika se većinom svodi na izradu manjih grafičkih elemenata i skidanje grafičkih predložaka s Interneta nakon čega se prilagođavaju zamišljenom dizajnu aplikacije.

7.1 Microsoft Visual Studio 2013 –VB.net

Za ovaj projekt koristi se Visual studio 2013 koji je u trenutku pisanja najnovija inačica. Kao jezik koristi se Basic, točnije Microsoftova inačica VisualBasic.net. Visual Basic .NET je sljedeća generacija programskog jezika Visual Basic koja predstavlja jednostavan i brz način za stvaranje aplikacija baziranih na .NET-u, u koje su uključeni XML Web servisi i Web aplikacije[MSDN Baza znanja,Microsoft,2015.].

Prednost korištenja Visual Studio-a je ta što u sebi već ima ugrađene elemente koji se pozivaju po potrebi, što uvelike olakšava posao jer ih korisnik ne treba sam definirati. Dijelovi izvornog koda su različito obojani i logičkim smislom uvučeni, kako bi izvorni kod izgledao preglednije. Osim već gotovih formulara, gumba, raznih elemenata za unos i ispis teksta i slika, Visual Studio ima ugrađene elemente za izradu izvješća (eng. report) kao i ugrađene mehanizme za spajanje na vanjske baze podataka. U ovom projektu aplikacija se spaja na Access 2010 bazu podataka, a RDLC modul za izvješća koristi se za ispis dokumenata.

Alternativa Visual Basicu bi bila Visual C# koji ima širi spektar mogućnosti, ali zahtijeva više pisanja koda i programiranja i nema neke gotove elemente poput inputbox-a koji ubrzavaju izradu same aplikacije.

Od noviteta u ovoj inačici su korištenje Microsoftove cloud usluge za spremanje vlastitih postavki korisničkog sučelja, unaprijeđenu navigaciju izvornom kodom pomoću karte koda kao i prikaz korištenja procesorskog vremena kod svakog pokrenutog elementa.

Od negativnih strana mora se navesti relativno veliko povećanje u korištenju procesorskih i memorijskih resursa kao i nekompatibilnost projekata s određenim starijim verzijama. Dizajn sučelja je pojednostavljen i u stilu ostalih Microsoftovih aplikacija za 2013. godinu. Samo sučelje izgleda relativno konzervativno jer je Microsoft ciljao na takozvani „*clean look*“. Nedostatak takvog dizajna je to što u monotoniji boja sve izgleda isto i korisnik se ne može snalaziti tako lako budući da ima mnogo različitih opcija i tekstova, a ikonice i grafika čiji se dizajn godinama nije drastično mijenjao, sada je dobio novi plosnati dizajn pri čemu se koriste uvijek iste boje i teže je razlikovati jednu kontrolu od druge. Zaključak što se dotiče dizajna je taj da je Microsoft mijenjao praktičnost starog dizajna za izgled nekog njihovog zacrtanog dizajna koji kompromitira preglednost i ergonomiju samog sučelja.

7.2 DBMS – Sustav za upravljanje bazama podataka

DBMS (eng. *Data base management sistem*) je sustav za upravljanje bazama podataka. U ovom projektu korišten je Access 2010. jer spada u jednostavnije verzije Microsoftovih programa ovoga tipa.

Alternativa Access-u bi bila SQL server zbog mnogo većih mogućnosti što se tiče sigurnosti i multithreading-a[9], ali za ovu aplikaciju bi to bilo vremenski zahtjevnije i kompliciranije rješenje za implementaciju. Osim toga Access ima tu prednost što ima lako prenosivu bazu podataka i izgledom je pristupačniji prosječnom korisniku jer je zamišljen da ga mogu koristiti ljudi kod kuće, a ne samo profesionalci. Mogućnosti Microsoft Access-a nadilaze mogućnosti bazičnih sustava za upravljanje bazama podataka jer on osim upravljanja entitetima i svojstvima te vezama među njima, ima i mogućnosti stvaranja izvješća, formulara za upis, makronaredbi itd. Sučelje je veoma intuitivno, čak toliko da bi netko bez ili s vrlo malo znanja SQL jezika moglo mnogo napraviti. Postoji grafičko sučelje za stvaranje tablica i dodavanje njima pripisanih svojstava u vidu redaka te svakom možemo precizno odrediti dozvole i ograničenja. Isto tako inicijalizacija veza je isto napravljena grafički, tako da se upotreba SQL jezika može svesti na minimum.

[9] Multithreading - proces kontroliranja toka većeg broja dretvi

Skoro svaku radnju u Access-u možemo raditi na tri načina. Prvi način je preko čarobnjaka gdje korak po korak nas program vodi prema tome što želimo u bazi kreirati. Drugi način je dizajn menadžer koji je grafičko sučelje u kojem entitete i veze među njima vidimo kao tablice. Takvim načinom puno je lakše raditi jer je pregledan i nije potrebno poznavanje SQL jezika. Treći način je direktno pisanje SQL jezika. To je najteži način upravljanja bazom. Zahtijeva dobro poznavanje SQL jezika i dobro opće znanje o bazama podataka da bi se napravila čak i najjednostavnija operacija koja se preko ostala dva navedena načina napravi puno lakše. Kompliciranost direktnog upisa SQL koda ima prednost što se sve puno preciznije može definirati, tj. mogu se definirati funkcije koje se ne nalaze kod čarobnjaka i dizajn menadžera.

7.3 Dizajn grafike

Da bi korištenje programa bilo ergonomski mora se voditi briga o izgledu programa. Gotovi elementi koji se koriste u .NET-u dizajnirani su po uzoru na ostale proizvode od Microsoft-a i ne trebaju velike preinake da bi se uklopili u sučelje operativnog sustava Windows. Elementi moraju biti posloženi logički i ako se nadovezuju na istu cjelinu moraju biti i u jednakim razmacima. Elementi koji su određeni kao ulazi imaju žutu pozadinsku boju koja nakon verifikacije prelazi u crvenu ili zelenu ovisno o tome je li upis dobar ili ne. Takav način dizajna programa omogućuje lakše uočavanje i popravljanje grešaka. Gumbi za krajnje operacije, kao što je generiranje naloga, imaju povećan font i crvenu boju slova. Pozadine pojedinih formulara su jednobojne, da ne skreću previše pažnje od ostalih elemenata na njoj. Boje koje se koriste kao pozadinske su standardne boje Windows operativnog sistema. Manji grafički zahvati i slike obrađene su u programu Adobe Photoshop CS6. Pod manje zahvate se misli na obrezivanja, prilagodbu rezolucije, jednostavnije crteže i modulaciju boja. Slike koje se nalaze u programu skinute su s Google pretraživača za slike te su rezolucijom i bojama lagano prilagođene stilu programa.

8. Izrada aplikacije

Kako bi se razvio neki software, u pravilu je na računalu na kojem se program razvija potreban moćniji hardware od računala na kojem se nalazi gotova prevedena aplikacija. Aplikacija u ovom projektu razvijena je preko Visual Studio-a 2013., koji je instaliran na Acerovom prijenosnom računalu s AMD A8 procesorom i 4 jezgre, 8 Gigabajta radne memorije i Solid state drive-om od 256GB s Marvellovim kontrolerom, dok se za grafičke performanse koristi AMD Radeon HD 8750 s 2GB vlastite memorije. Na tu konfiguraciju instalirana je 64-bitna verzija Windowsa 7 Professional jer je taj operativni sustav već neko vrijeme na tržištu i provjereno dobro radi za potrebe ovog projekta. Osim operativnog sustava na računalo je bilo potrebno dodatno instalirati Microsoft Office 2010.

Nakon instaliranog software-a generirana je baza podataka čija je shema napravljena definiranjem poznatih entiteta i njihovih atributa te vezama među njima. Kada je baza podataka napravljena, spremljena je na sigurno mjesto koje nije na vanjskoj memoriji i koje nije ograničeno sigurnosnom politikom samog operativnog sustava. Lokacija baze podataka je bitna stavka jer je potrebna kod definiranja povezivnog stringa (eng. connection string). Povezivni string je tekst potreban za spajanje nekog programa na izvor ili bazu podataka[Alessandro Del Sole, 2010,Str.515]. Nakon što je riješeno pitanje baze podataka, napravljena je sama aplikacija koja popunjava bazu podataka s podacima i prema tim podacima generira izvještaje. Poslovi aplikacije prikazani su uz pomoć dijagrama toka i preko njih su implementirani algoritmi u aplikaciju.

Za olakšano programiranje razvijen je osobni skup pravila za lakše snalaženje u kodu i u samom programu. Prvo pravilo je da za dio koda koji se izvršava više puta obavezno se koristi funkcija, što ujedno jest i njihova svrha u Visual Basic-u. Alternativa tome bila bi kopiranje dijela izvornog koda na mjesta gdje je potreban umjesto da se poziva funkcija, pri čemu bi se u slučaju neke prepravke ili greške više vremena izgubilo na prepravljanje programskog koda. Sve funkcije su kratko nazvane po tome što rade, a riječi su odvojene s „_“ što olakšava traženje funkcija i poboljšava čitljivost izvornog koda ukoliko ne postoje komentari. Ukoliko se usred programiranja dogodi da se nađe bolje alternativno rješenje, onda prvotnu ideju valja zadržati kao komentar, jer program taj dio koda neće čitati, a ukoliko se nova metoda pokaže lošom onda još uvijek postoji mogućnost povratka na staro rješenje. Najvažnije pravilo od svih je da se nijedan element ne ostavi s imenom kojeg je postavio sustav. Prilikom stvaranja novog gotovog elementa okruženje mu daje ime po tipu elementa zajedno s rednim brojem. Alternativa pri imenovanju je takva da se imenuje svaka kontrola prema shemi

„skraćenica_naziv_funkcije_elementa“. Skraćenice se mogu vidjeti u tablici 8.1[10]. Poštujući pravila uvelike se ubrzava traženje željenih dijelova, pisanje i razumljivost koda.

Tablica 8.1 Popis skraćenica

Element	**Skraćenica u imenu**
Textbox	txt
Label	lbl
Combo box	cmb
DateTimePicker	dtp
NumericUpDown	nud
Form	frm
Datatable	table
Button	btn
Radiobutton	rbn
Checkbox	chk
Connection	con

8.1 Kratki opis poslova aplikacije

Prvi posao koji aplikacija mora napraviti je spriječiti neovlašteni ulaz i dodijeliti prava ovisno o tipu korisnika. Kada se aplikacija pokrene prvo se prikaže prozor za unos korisničkog imena i lozinke te dok se ne upiše ispravno korisničko ime i lozinka ne može se pristupiti niti jednom dijelu programa. Nakon prvog koraka pojavljuje se početni prozor te se omoguće sve funkcije koje su specifičnom korisniku dozvoljene.

Da bi program radio onako kako je zamišljen i prema individualnim potrebama, u njemu se moraju namjestiti postavke. Njihova je dužnost postaviti parametre koji će se poslije u glavnim dijelovima programa pozivati ili automatski ili po potrebi.

Postavke se sastoje od postavki putnika, mjesta, vozila, osnovnih postavki i postavki korisnika. U postavke putnika upisuju se zaposlenici koji putuju i njihovi podaci koji će se poslije stavljati u putni nalog. Tamo se mogu dodavati novi i brisati ili mijenjati postojeći putnici. Kod upravljanja mjestima može se dodati, mijenjati ili brisati mjesto koje se kasnije dohvaća kao polazište i odredište. Kod njih se upisuje ili mijenja naziv mjesta, općine ili grada, poštanski broj

[10] Popis skraćenica-Tablica usporedbe imena elemenata u Visual Studio-u i njima pripadajućim skraćenicama

jer postoji više mjesta s istim ili sličnim imenom, ali imaju jedinstven poštanski broj i ne nalaze se u istoj županiji.

Postavke vozila služe za dodavanje, mijenjanje i brisanje vozila kojima se putuje i koja se navode u putnom nalogu. Kod vozila upisuje se model vozila, registracijska oznaka, vlasništvo i zadnja kilometraža koja će kad se povuče u sljedeći putni nalog biti početna kilometraža putovanja. Isto tako početna kilometraža se mijenja svakim obračunom kad se upiše završna kilometraža putovanja jer završna kilometraža putovanja je početna kilometraža sljedećeg putovanja.

Osnovnim postavkama i postavkama korisnika pristup ima samo administrator i korisnik u programer modu. Svrha osnovnih postavki je postaviti informacije o poduzeću i logo u zaglavlje dokumenta. Najveću zadaću kod postavki imaju postavke korisnika. One su bitne jer u njima se dodaju novi korisnici i mijenjaju ili brišu postojeći. U njima, osim što se upisuju podaci, definiraju se prava koja korisnik ima pri korištenju programa, zbog čega je to dozvoljeno samo administratoru. On mora u programu odrediti tko može stvarati dokumente, a tko ih može samo pregledavati.

Kad su postavke podešene, može se početi sa stvaranjem dokumenata, što je i najvažniji posao razvijene aplikacije. Prvi dokument koji program mora kreirati je putni nalog. U njega se preko mehanizama za dohvat upisuje putnik, vozilo, predujam, dnevnica, datum dokumenta, datum putovanja, rok povrata izvještaja, polazište, odredište i svrha putovanja. S upisanim podacima program generira putni nalog koji se može ispisati uz pomoć pisaća ili spremiti u .doc, .xls i .pdf formatu. Ujedno se upisani podaci mogu spremaju u bazu za naknadni ispis.

Poslije putnog naloga slijedi izvještaj. On je najjednostavniji dokument i ne zahtjeva upis parametara. Poziva se pritiskom gumba nakon čega se prikazuje njegov izgled sa svrhom da se ispiše.

Sljedeći dokument u nizu je obračun troškova. Podaci koji su potrebni kod tog dokumenta dijelom su upisani u putnom nalogu, a dijelom su ručno upisani u izvještaj. Oni koji su upisani u putni nalog, kao što su dnevnice, predujmovi i početni kilometri, upisat će se automatski, a podaci na izvještaju koje putnik upisuje tokom putovanja nisu upisani nigdje u bazi i oni se ručno unose. Program mora, s obzirom na unesenu završnu kilometražu, izračunati ukupan broj kilometara i cijenu kilometraže. Nakon toga se pribrajaju dnevnice i ostali troškovi koji su nastali tokom putovanja od kojih se oduzima predujam.

Nakon što je izvršen izračun, dokument se mora ispisati i pospremiti u arhivu. Aplikacija u bazu podataka sprema sve putne naloge i obračune za slučaj potrebe naknadnog ispisa. Arhive moraju

omogućiti tablični prikaz svih dokumenata. Arhiva mora biti pregledna tako da se navigacijom kroz tablicu dokumenata moraju u realnom vremenu prikazati detalji dokumenta. Kada je u arhivi mnogo dokumenta koriste se filtri i filtriracija dokumenata koji su od interesa korisniku. Arhiva ima mogućnost pretraživanja po identifikatoru, ruti, datumu i putniku. Kad se nađe traženi dokument u arhivi, posao aplikacije je da ga prikaže i pripremi za ispis ili za spremanje u neki od .pdf, .doc ili .xls formata.

Osim dijelova programa koji su namijenjeni korisniku, ima i dio programa koji je namijenjen programeru za rješavanje mogućih incidenata. Taj dio programa zove se programer mod. U programer modu dodan je formular koji u realnom vremenu ispisuje parametre programa. Svrha tog moda je vidjeti što se događa u programu i uz pomoć prikazanih parametara vidjeti zbog čega je došlo do incidenta.

8.2 Glavni dijelovi programa

U daljnjem tekstu objasnit će se glavni dijelovi programa od kojih se ističu određeni formulari i izvještaji. Od formulara, ističu se formulari za postavke, formulari za upis putnog naloga, upis obračuna, arhivski formulari, formulari za dohvat i izvještaje. Svaki od njih posebnog je dizajna jer imaju različite funkcije i različite načine na koje se njima upravlja. Tok rada na pojedinim dijelovima programa razlikuje se zbog specifičnosti poslova koje obavljaju. Formulari za upis nisu slični arhivskim formularima, a formulari za postavke slični su arhivskim, ali moraju obavljati određene poslove koje arhivski formulari nemaju, dok kod formulara koji služe za ispis dokumenta naglasak se stavlja na preglednost i lakoću navigacije dokumentom. U svakom slučaju, svaki dio programa mora biti maksimalno logičan, pregledan i ugodan za rad.

8.2.1 Formular za dohvat

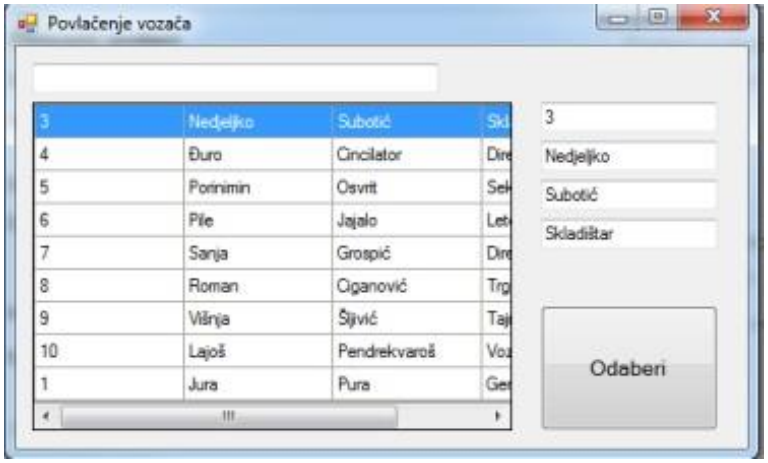
Formulari za dohvat su formulari koji se pozivaju tokom upisa nekog dijela dokumenta. Oni su u pravilu malih dimenzija i jednostavnog dizajna. Njihova svrha je odabrati iz baze koji će se parametri povući u formular u koji se upisuje. U pravilu se otvaraju automatski kada se pritisne bilo koja tipka na određenim Textbox-ovima[11]. Kad se otvori neki formular za dohvat tada se u skrivenom Stack formularu upiše da je taj određeni formular za dohvat otvoren. U ovoj aplikaciji su tri formulara za dohvat: putnici, vozila i mjesta. Dizajnirani su tako da je na vrhu filtar koji filtrira tablicu s mogućim unosima koji se nalaze ispod njega. Ta tablica je prikazana pomoću

[11] Textbox - .Net komponenta koja omogućuje korisniku unos teksta sa mogućnošću upisa u više reda ili mogućnošću upisa lozinke (Izvor: Microsoft Visual Studio – opis komponenti)

Datagridview-a[12]. Za filtriranje nije potrebna dodatna radnja, kao na primjer pritisak na gumb Enter ili klik na gumb koji potvrđuje uneseno filtriranje, već se događa u stvarnom vremenu kako se upisuje u Textbox.

Navigacija može biti mišem ili navigacijskim tipkama gore i dolje, tako da je praktično bilo kome bez obzira preferira li miš ili tipkovnicu. Navigacijom kroz tablicu automatski se upisuju odabrane vrijednosti u bočne Textbox-ove tako da je odabir vidljiv i nedvosmislen. Kako bi se potvrdio odabir dovoljno je pritisnuti tipku Enter ili kliknuti na gumb ispod bočnih Textbox-eva i tada će se odabrane vrijednosti same kopirati na za to određena mjesta na upisnom formularu. Veličina im je fiksna jer na zadanoj veličini su vidljivi svi podaci i sve kontrole su ergonomski smještene. Ne bi imalo smisla raditi ovaj formular većim zbog toga jer je to pomoćni formular koji u ovoj veličini zadovoljavajuće obavlja svoju funkciju. Primjer formulara za dohvat nalazi se na slici 8.2.1.1.[13]

Slika 8.2.1.1 Formular za dohvat na primjeru vozača



8.2.2 Formular za upravljanje postavkama

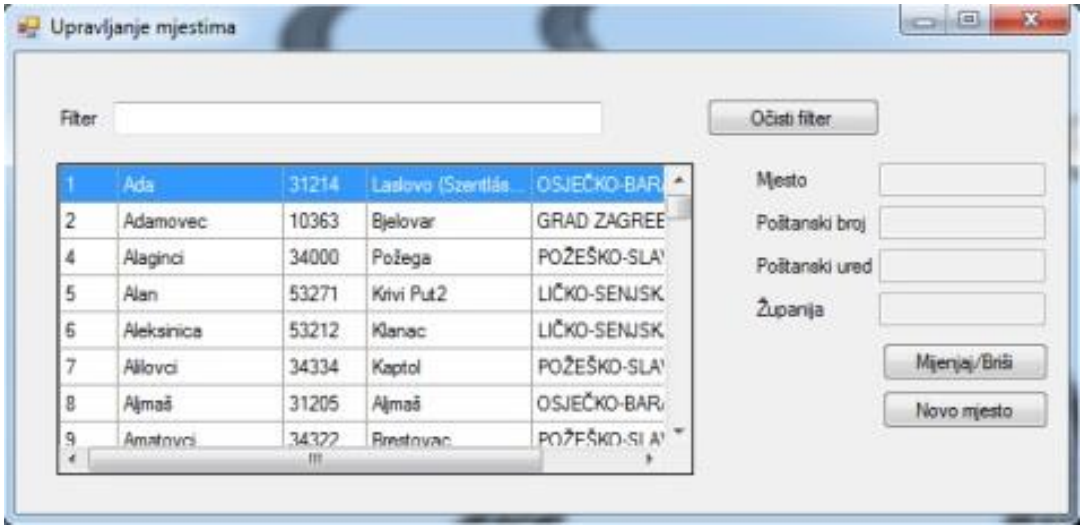
Formulari za upravljanje postavkama koriste se za postavke mjesta, putnika i vozila. Oni imaju funkciju kataloga za stavke koje se dohvaćaju kada se stvara novi putni nalog. Po dizajnu su jako slične formularima za dohvat jer isto kao i oni imaju filtar na vrhu i ispod njega tablicu u Datagridview-u. Bočni Textbox-evi služe za upis novih stavki koje su namijenjene za dohvat isto kao i promjenu njihovih vrijednosti. Sadrži sedam gumba koji se prikazuju i sakrivaju po potrebi ovisno o tome koja radnja se radi na njoj. Taj je mehanizam ugrađen da korisnika ne zbunjuju funkcije koje u tom trenutku nije moguće realizirati i isto tako da eliminira nastajanje potencijalne pogreške.

[12] Datagridview - .NET komponenta koja omogućuje prikaz stupaca i redaka podataka kao skup ćelija otvorenih za mogućnost prilagodbe (Izvor: Microsoft Visual Studio – opis komponenti)

[13] Slika 8.2.1.1 Formular za dohvat na primjeru vozača

Na primjer ne može se usred promjene jedne stavke zatražiti da se dodaje nova dok radnja nije dovršena ili dok nije naznačeno da je korisnik odustao od nje. Veličina ovog formulara, isto kao kod formulara za dohvat je fiksna, jer je korištena minimalna veličina kod koje sve izgleda pregledno. Povećanjem veličine elementi bi samo bili više razmaknuti, vidljivost podataka se ne bi poboljšala i estetika samog formulara bi se narušila, stoga se odluka da ovaj formular bude fiksne veličine čini logičnom. Primjer formulara za upravljanje postavkama može se vidjeti na slici 8.2.2.1[14].

Slika 8.2.2.1 Formular za upravljanje postavkama na primjeru upravljanja mjestima



8.2.3 Minimod i Maximod formulari

Budući da razvijena aplikacija ima dva moda, za njih su potrebna dva različita formulara. To su početni formulari s kojima je moguće doći do svakog dijela programa.

Minimod formular napravljen je minimalistički. Na sebi ima kontrole koje su poredane jedne ispod druge. Veličina formulara je relativno mala jer je zamišljena kao nenametljiv način korištenja programom. Njegova veličina je fiksna jer kad bi bila napravljena manjom, izgubila bi funkcionalnost, a kad bi bila veća, izgubila bi smisao i njezina preglednost se ne bi promijenila.

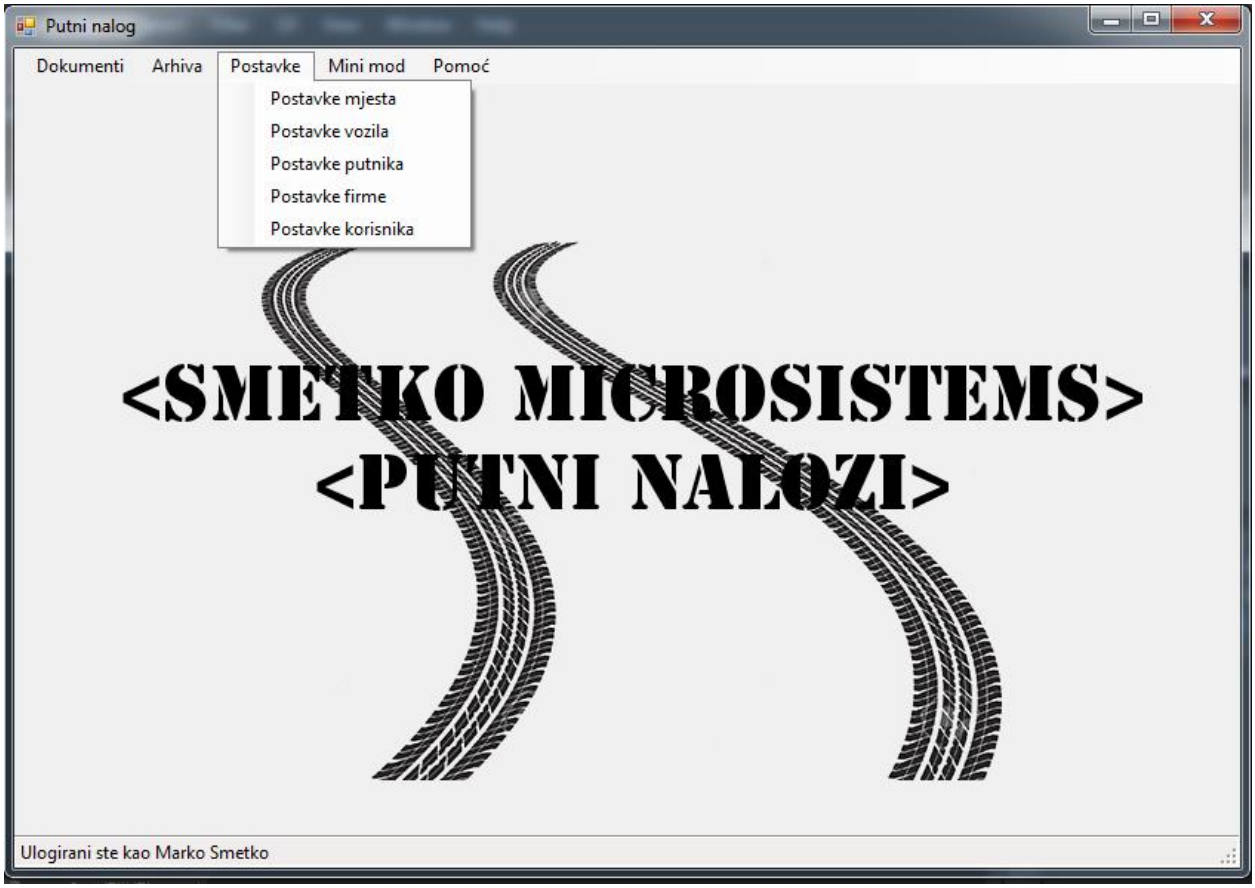
Suprotno Minimodu, Maximod formular je zamišljen da bude veći zbog toga jer je namijenjen stalnom korištenju. On služi kao pozadina koja se koristi preko cijelog zaslona dok se s nje preko padajućih izbornika kategoriziranih prema poslovima i funkcijama pozivaju drugi formulari. Na dnu ima statusnu traku koja prikazuje status korisnika i eventualne greške s funkcionalnošću programa, ako se pojave. Veličina, iako je zamišljena da zauzima cijeli ekran, nije fiksa, već ju korisnik može postaviti kako mu odgovara jer ne utječe na funkcionalnost. Funkcionalnost kod

[14]Slika 8.2.2.1 Formular za upravljanje postavkama na primjeru upravljanja mjestima

smanjivanja Maximoda nije kompromitirana jer minimalna veličina to sprečava. Iz Minimoda se u Maximod prelazi preko pritiska na gumb i ta radnja se pamti u postavkama korisnika tako da kod prijave se program pokreće u modu kojeg je zadnji pojedini korisnik koristio.

Primjer maximod formulara može se vidjeti na slici8.2.3.1.[15]

Slika 8.2.3.1 Primjer formulara maximod režima rada



8.2.4 Formular za unos putnog naloga

Formular za unos putnog naloga je najvažniji i najkompleksniji formular u aplikaciji i služi za generiranje i arhiviranje putnog naloga. Sastoji se od 4 manja modula čiji su elementi raspoređeni u Groupbox-eve[16]. To je učinjeno jer se skupina kontrola koja se koristi za upis može na jednostavniji način omogućiti i onemogućiti, ovisno o tome kad je za to potreba. Taj princip se koristi tako da korisnik može lakše i po logičnom poretku unositi stavke koje se prelaskom na drugi Groupbox verificiraju. Boje na ovom formularu imaju važnu ulogu. Korisnik samo ispunjava stavke koje su označene žutom bojom. Ukoliko je usred verifikacije neka stavka

[15]Slika 8.2.3.1 Primjer formulara maximod režima rada

[16] Groupbox-.NET komponenta koja postavlja okvir oko ostalih komponenti sa opcijom stavljanja naslova(Izvor: Microsoft Visual Studio – opis komponenti)

ispravno ispunjena ona će dobiti zelenu pozadinu, u suprotnom će pozadina pocrveniti i neće se aktivirati sljedeći Groupbox dok se ne prepravi naznačena greška.

Svaki modul ima gumb za potvrdu podataka i pomištavanje podataka, osim modula za osnovne podatke jer je prejednostavan. Kad je formular pokrenut, inicijalno su svi Groupbox-evi onemogućeni osim onog s osnovnim podacima jer se on ispunjava prvi. On se sastoji od dva Datetimepicker-a[17], Textbox-a koji prikazuje broj putnog naloga, Textbox-eva za unos putnika i pripadajućih Labela kojima imenujemo kontrole. Datetimepicker-i služe za odabir datuma koji su automatski postavljeni da prikazuju današnji datum kao zadanu vrijednost. Kod njih je postavljen mehanizam takav da datum nastajanja dokumenta ne može biti kasniji od datuma putovanja pri čemu je mogućnost takve greške eliminirana. Textbox-u s brojem putnog naloga je onemogućen pristup jer se taj broj automatski upisuje kao broj putnog naloga, pri čemu pročita iz baze zadnji broj i uveća ga za jedan.

Osim datuma, u osnovne podatke unose se podaci o putniku koji sadrže četiri Textbox-a. Jedan je sakriven jer je korisniku nebitan, on sadrži identifikator putnika koji može biti prikazan u nekoj budućoj verziji kod programer Mod-a. Textbox za unos putnika zadan je da bude prvi odabrani element kad se formular otvori i on je žute boje, preko njega se poziva formular za dohvat i ispunjavaju ostali Textbox-evi.

Kad je uspješno ispunjen Groupbox s osnovnim podacima, program se prebacuje na ispunu podataka o vozilu. Isto imamo grupu Textbox-ova od kojih je jedan za identifikator skriven i jedan za dohvat žut. Osim njih u aplikaciji su dva gumba od kojih jedan služi za vraćanje svih elementa u Groupboxu na početnu vrijednost, a drugi za potvrdu unosa pri čemu unesene vrijednosti prolaze kroz verifikaciju i ukoliko su ispravne omoguće pristup sljedećem Groupbox-u.

Sljedeći Groupbox sadrži specifične pojedinosti o putovanju koje su sve brojčane i zbog čega se sastoji samo od Numericupdown[18] elemenata. Oni su praktični jer je njihova sadržana vrijednost broj po tipu podatka. Tipkama gore dolje može se mijenjati njihova vrijednost, isto tako mogu se direktno upisivati brojevi u njih, pri čemu u svojstvima se može zadati korak kod promjene vrijednosti isto kao i decimalna mjesta. Svaki Textbox podešen je tako da se pritiskom na tipku Enter pozicionira na sljedeći element.

[17] Datetimepicker - .Net komponenta koja omogućuje odabir datuma i vremena sa opcijom prikaza odabira (Izvor: Microsoft Visual Studio – opis komponenti)

[18]Numericupdown - .Net komponenta koja prikazuje brojčanu vrijednost koju korisnik može mijenjati klikom ili tipkama kontrolu u veličini koraka koji je prethodno postavljen(Izvor: Microsoft Visual Studio – opis komponenti)

Prvi elementi su za duljinu putovanja i rok povrata izvještaja. Budući da se radi o danima, namješteni su tako da prikazuju samo cijele brojeve. Između njih postoji mehanizam takav da minimalna dopuštena vrijednost kod povrata izvještaja je trajanje putovanja plus još tri dana, jer putnik ima pravo na dodatno vrijeme da popuni izvještaj, a i izbjegava se scenarij gdje bi rok povrata izvještaja bio raniji od roka povrata s puta.

Sljedeće se ispunjava visina dnevnice po danu. Njezin element je namješten da bude decimalni broj s dva decimalna mjesta. Mijenjajući vrijednosti kod dnevnice i broju dana putovanja automatski se računa ukupni zbroj dnevnica koji se prikazuje u Label-u[19] jer je to vrijednost koja se ne upisuje, ali je korisniku korisna.

Zadnja vrijednost koja se ispunjava je predujam, njegov element ima ista svojstva kao dnevnica jer se isto radi o novčanoj vrijednosti, ali za razliku od nje početna mu je vrijednost nula jer u praksi se u većini putnih naloga ne isplaćuje.

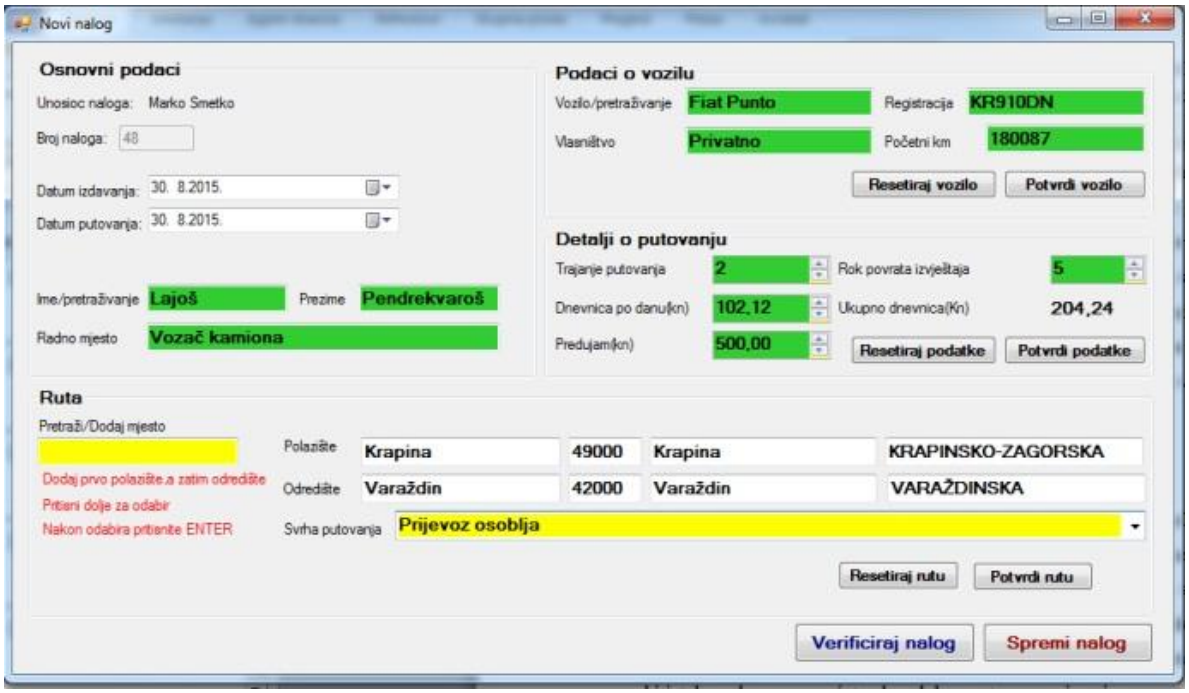
Ispod njih nalaze se 2 gumba od kojih jedan vraća cijeli modul na početno stanje, a drugi verificira i omogućava zadnji modul gdje se upisuje ruta i svrha putovanja.

Groupbox u kojeg se upisuje ruta je po veličini najveći i sadrži najviše stavki, zbog čega zauzima skoro cijelu širinu formulara. Polazište i odredište se ispunjavaju uz pomoć formulara za dohvat koji se poziva u oba slučaja s istog mjesta. Dohvat s prvog upisa upisat će se u polazište, a drugog u odredište. Polazište i odredište imaju po četiri dijela svaki i za to zasebna četiri Textbox-a. Kod njih ručni upis nije onemogućen jer ponekad se kod upisa putnog naloga u rutu treba upisati neka specifičnost, kao što je kvart ili lokalni naziv naselja ili ulice. Ispod rute nalazi se Combobox koji služi za unos svrhe podataka. On je korišten jer je kombinacija padajućeg izbornika i Textbox-a. U njega se ručno upisuje svrha putovanja ili odabire jedna od već predefinirah opcija koja je upisana u njegova svojstva. Na dnu ovog Groupbox-a je gumb za vraćanje svih elemenata na početak i gumb koji pokreće verifikaciju, nakon koje se pokazivač premjesti na gumb za arhiviranje i ispis putnog naloga. Taj se gumb nalazi na dnu samog formulara i ne pripada niti jednom modulu. Pritiskom na njega pokreće se još jednom verifikacija svih upisanih stavki i nakon što se pokažu kao ispravne, pohrane se u bazu i pokrene formular za prikaz ispisa putnog naloga. Osim njega postoji i verifikacijski gumb koji samo služi za verifikaciju svih modula. On je samo za provjeru upisa kada korisnik nije siguran o ispisu. Primjer formulara za unos putnog naloga može se vidjeti na slici 8.2.4.1[20].

[19] Label - .NET komponenta koja prikazuje tekstualnu informaciju o nekoj kontroli (Izvor: Microsoft Visual Studio - opis komponenti)

[20]Slika 8.2.4.1 Primjer formulara za unos putnog naloga sa ispunjenim stavkama

Slika 8.2.4.1 Primjer formulara za unos putnog naloga sa ispunjenim stavkama



8.2.5 Formular za obračun troškova

Formular za obračun troškova, kao i formular za putni nalog, služi za upis podataka koji se arhiviraju i od njih se stvara dokument. Budući da se njegov sadržaj zasniva na putnom nalogu i izvještaju s putovanja zajedno s pripadajućim potvrđujućim dokumentima, ima svoje specifičnosti.

Prvi dokument na kojem se temelji obračun je putni nalog, njegovi podaci su već upisani u bazi i oni se pozivaju, dok podatke s izvještaja putovanja korisnik mora sam upisati. Formular mora biti takav da uz pomoć ta dva dokumenta korisnik može napraviti ispravan obračun. Kako bi se moglo započeti s obračunom, prvo se učita putni nalog na temelju kojeg se obračun radi. Za svaki putni nalog se radi samo jedan obračun pa su tako i njihovi brojevi, odnosno identifikatori jednaki. Na vrhu formulara se nalazi Textbox i gumb uz pomoć kojeg se učitava putni nalog na temelju kojeg se temelji trenutni obračun. U gornjem desnom kutu je Datetimepicker koji određuje pri pokretanju datum stvaranja obračuna. Korisniku je omogućeno da taj datum može izmijeniti ako je potrebno. Ispod se nalaze dva groupbox-a koji su jedan ispod drugoga. Prvi sadrži samo podatke iz putnog naloga i na njega se ništa ne upisuje i sadrži samo Label-e. Budući da putni nalog ima mnogo stavki koje bi bile bez imenovanja nečitljive, postavljene su vrijednosti tako da su crne one koje imenuju neki podatak ili svojstvo, a kestenjasto crvene su one koje prikazuju vrijednosti.

Drugi Groupbox služi za unos troškova s izvještaja putovanja. Prvo se unosi kilometraža nakon putovanja u Textbox koji dopušta da se u njega unose samo brojeve. Postoji mehanizam koji ne

dopušta da korisnik nastavi dalje s radom dok nije napisan broj kilometara koji je veći. Nakon unosa kilometraže, unosi se cijena po kilometru u za to predviđeni Numericupdown koji je zbog cijene kilometara, koje u pravilu variraju oko dvije do tri kune, namješten na početnu vrijednost nula s korakom od 0,10, odnosno 10 lipa. Nakon unosa kilometraže omogućuje se unos ostalih troškova koji su nastali tokom putovanja.

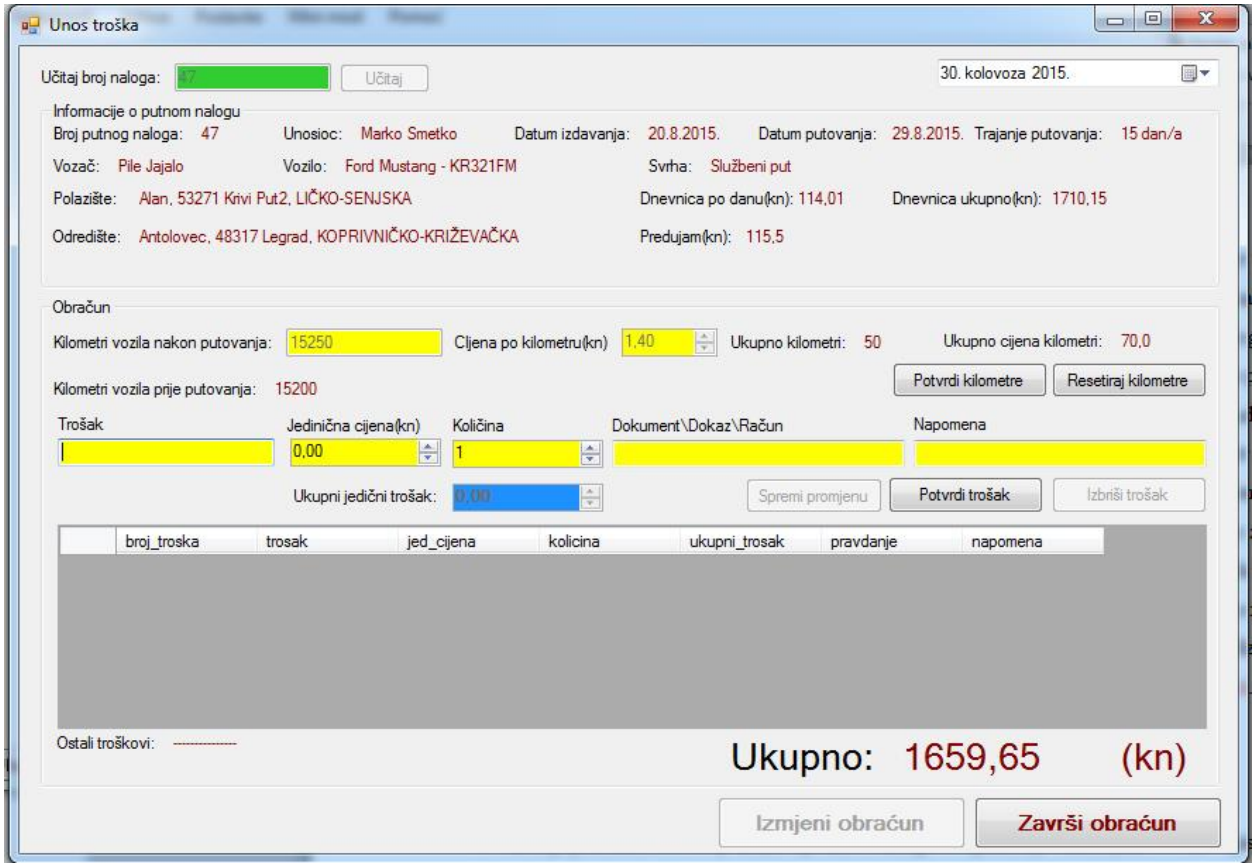
Budući da je zapis ostalih troškova zamišljen kao skup koji se sadrži od naziva, cijene, količine, dokumenta s kojim se dokazuje trošak i napomene. Naziv se upisuje u Textbox, to je kratki opis troška, u ravnini s njim su dva Numericupdown-a. Prvi je namješten da ima dva decimalna mjesta jer predstavlja cijenu u kunama, a korak je namještan na 1,00, odnosno jednu kunu, dok drugi služi za količinu nanesenog troška. U pravilu je količina jedan i početna vrijednost joj je jedan, ali u nekim slučajevima, kao na primjer kod vožnje trajektom kada se kupi u isto vrijeme i povratna karta s istom cijenom i to se pravda istim dokumentom, onda je praktično da se može mijenjati količina umjesto da se dvaput unosi. Sljedeće se mora upisati dokazni dokument koji je isto kombinacija slova i brojeva pa se i za njega koristi Textbox. Nakon njega upisuje se napomena, ona je opcionalna i stoga je dopušteno ne ispuniti njezin Textbox, za razliku od ostalih stavki troška koje moraju imati ispunjen minimum pri čemu nijedna vrijednost ne smije biti nula. Za potvrdu troška napravljen je gumb koji pokrene mehanizam koji provjerava jesu li ispunjeni minimalni uvjeti da bi trošak bio ispravan nakon čega se vrijednosti upišu u Datagridview. Upisom novih troškova automatski se računaju ukupni ostali troškovi i ukupni troškovi koji su prikazani ispod Datagridview-a.

Budući da se tijekom upisa mogu dogoditi greške, mora se omogućiti prepravak zapisa u Datagridview-u. Zapis se prepravlja dvostrukim klikom na zapis u tablici pri čemu se vrijednosti vrate natrag gdje su prvotno bile upisane i sakrije se gumb za pohranu i prikažu se kraj njega dva nova gumba, jedan je za spremanje promjena pri čemu ih pohrani opet u isto mjesto na tablici, a drugi je za brisanje zapisa u potpunosti.

Kada je sve uspješno upisano onda obračun treba pohraniti i iz upisanih stavki napraviti dokument za ispis. Za to su napravljena dva gumba na dnu formulara. Ta dva gumba nikad nisu prikazana u isto vrijeme. Iako oba gumba pokreću inicijalnu verifikaciju upisanih stavki i na kraju pokreću prikaz za ispis, jedan je prikazan kod mijenjanja postojećeg obračuna, a drugi je prikazan kada se radi obračun iznova, jer kod tih radnji postupak arhiviranja nije isti. Da bi sve bilo pregledno formular je fiksne veličine od 880x610 piksela. Mijenjanje veličine dodatno bi

zakompliciralo dizajn ovog formulara i zbog toga nije implementirano. Primjer formulara za unos troškova može se vidjeti na slici 8.2.5.1[21].

Slika 8.2.5.1 Slika Primjer formulara za unos troškova sa učitanim putnim nalogom



8.2.6 Izvještaj putni nalog

Zbog praktičnosti ovaj izvještaj kao i drugi u aplikaciji zamišljeni su za ispis na A4 format papira u portretnoj orijentaciji. Budući da je A4 format ograničenih dimenzija od 210mm x 297mm, zbog ograničenih mogućnosti pisaća i mogućeg uvezivanja dokumenta u fascikle moraju se uračunati margine. Nakon postavka margina mora podešena je optimalna veličina elemenata koja je čitljiva i dizajnirana u skladu s potrebama.

Kao kod većine putnih naloga logo i informacije o poduzeću smještene su na vrh. Logo se nalazi u gornjem desnom uglu i veličine je od 2cm x 2cm što ga čini uočljivim, ali u isto vrijeme nije prevelik. Naziv, adresa i osobni identifikacijski broj su isto kao i kod većine izvještaja s desne strane loga raspoređeni u jednakim razmacima u zaglavlju dokumenta.

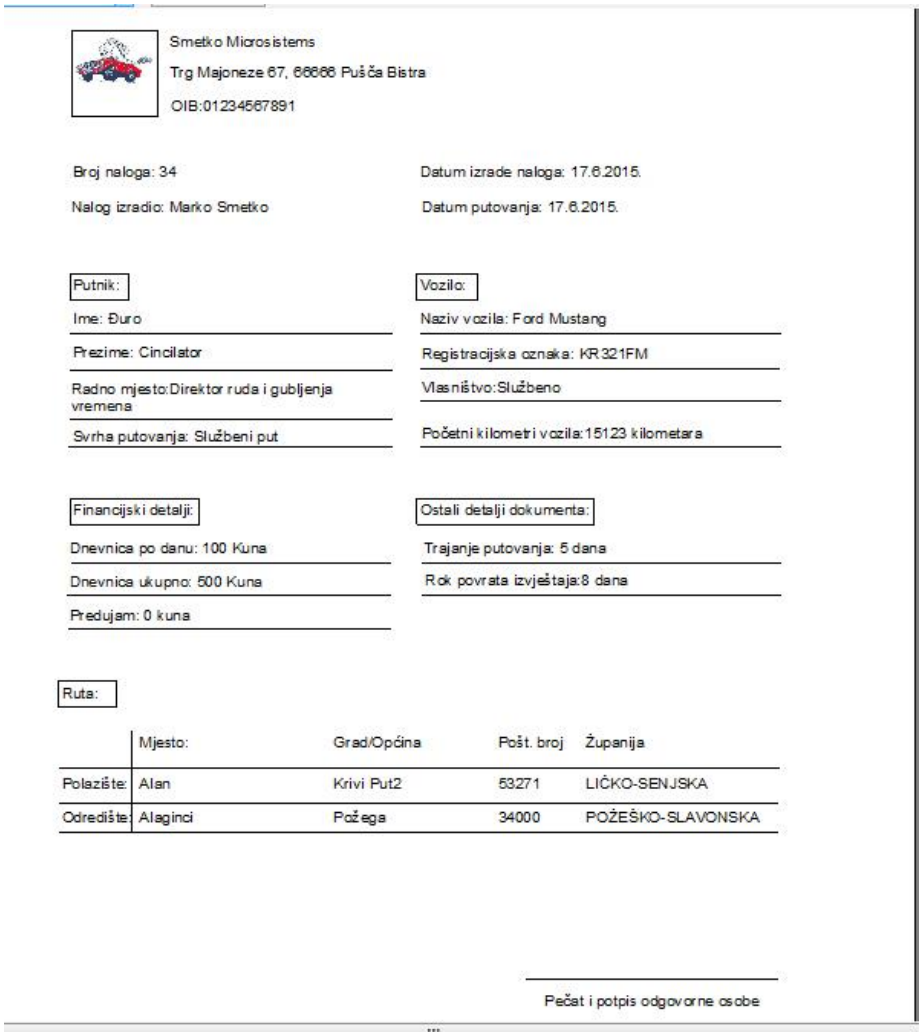
Osnovni podaci o dokumentu nalaze se ispod zaglavlja. U njih spadaju broj naloga,osoba koja je izradila dokument,datum dokumenta i datum putovanja. Zbog ograničene širine raspoređeni su u dva stupca. Kako bi se logički rasporedili ostali podaci potrebno je razmišljati kao službena

[21]Slika 8.2.5.1 Slika Primjer formulara za unos troškova sa učitanim putnim nalogom

osoba koja provjerava putnika i vozilo. Stoga su sljedeći podaci za unos oni o putniku i vozilu, a tek ispod njih manje važne informacije o dnevnicama i predujmovima.

Od ostalih podataka preostaje samo ruta. Ona zahtjeva veću količinu mjesta na dokumentu i zbog toga je ispod ostalih podataka po cijeloj širini stranice, što ih čini uočljivima. Ispod svih podataka u donjem desnom uglu postavljeno je mjesto za pečat i potpis odgovorne osobe jer to čini ispisani dokument važećim. Primjer ispisa putnog naloga vidljiv je na slici 8.2.6.1[22].

Slika 8.2.6.1 Primjer ispisa putnog naloga



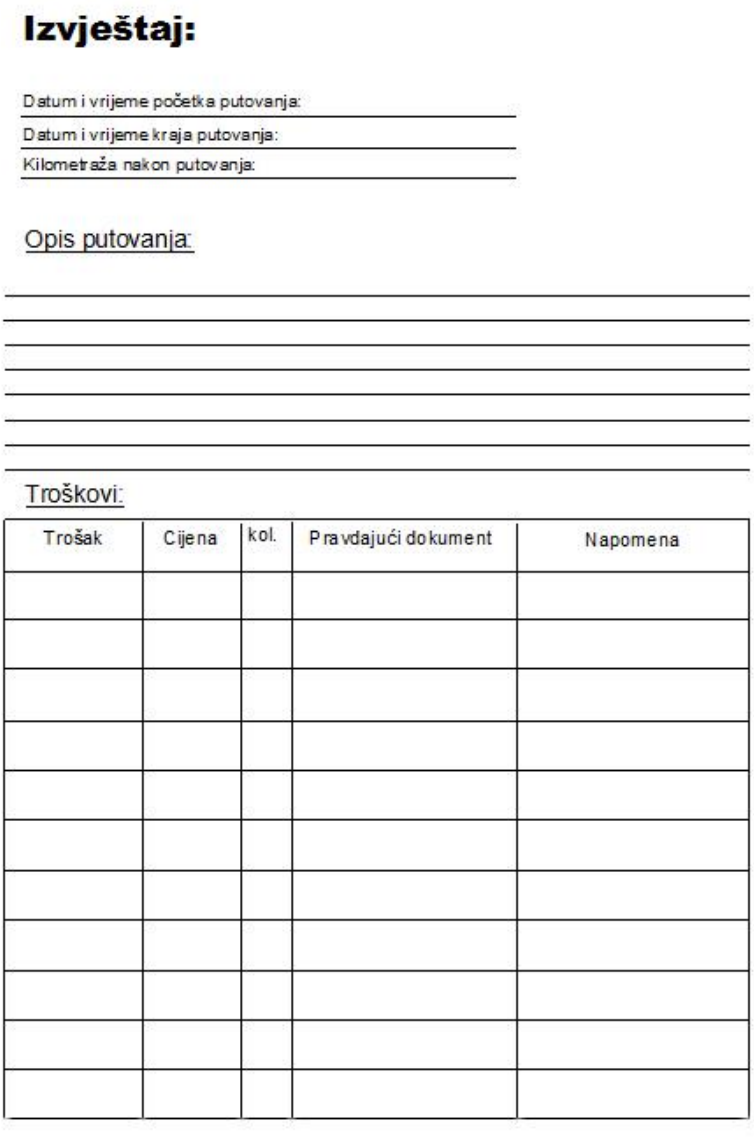
8.2.7 Izvještaj s puta

Izvještaj s puta je najjednostavniji među izvještajima jer se na njemu ne nalaze nikakvi parametri niti podaci iz arhive. On je dizajniran tako da putnik može na njemu zapisati podatke do kojih je došao tokom putovanja. Putnik je dužan sve stavke izvještaja ispuniti. Mjesta koja se ispunjavaju su vidljivi na izvještaju u obliku crta ili tablica. Na vrh je stavljeno mjesto gdje putnik ispunjava datum i točno vrijeme u kojem je krenuo na putovanje i vratio se s putovanja, te završnu

[22]Slika 8.2.6.1 Primjer ispisa putnog naloga

kilometražu vozila. Ispod toga jedna trećina dokumenta je posvećena pismenom opisu putovanja kojeg je putnik dužan napisati. Budući da je to slobodan tekst, na predviđeno mjesto su postavljene crte međusobno razmaknute 5 mm što je prosječnoj osobi dovoljno za pisanje čitljivog teksta. Ispod pismenog opisa putovanja postavljena je tablica u koju se ispunjavaju troškovi koji su nastali tokom putovanja. Podijeljena je na pet stupaca koji predstavljaju troškove koji će se upisivati u obračun nakon kraja putovanja. Svaka ćelija je visine jedan centimetar što omogućuje da se neke stavke mogu pisati i u dva reda ukoliko se za to pokaže potreba. Primjer praznog izvještaja s puta vidljiv je na slici 8.2.7.1[23].

Slika 8.2.7.1 Primjer praznog izvještaja s putovanja



Izvještaj:

Datum i vrijeme početka putovanja:

Datum i vrijeme kraja putovanja:

Kilometraža nakon putovanja:

Opis putovanja:

Troškovi:

Trošak	Cijena	kol.	Pravdajući dokument	Napomena

[23]Slika 8.2.7.1 Primjer praznog izvještaja s putovanja

8.2.8 Izvještaj obračuna troškova

Obračun troškova je posljednji od tri dokumenta koji se rade za putne naloge. Na vrhu se nalazi naslov dokumenta koji je veći od ostalih naslova sekcija i podnaslova u daljnjem tekstu dokumenta. Ispod njega nalazi se datum obračuna. Taj podatak je bitan, ali nije ključan za obračun pa je za njega korištena zadana veličina slova bez podebljana ili podcrtavanja. Ispod njega je većim podebljanim slovima napisan broj radnog naloga odnosno obračuna. Taj je podatak tako napisan jer je bitna stavka dokumenta koja mora biti uočljiva.

Troškovi su prikazani u četiri sekcije: Obračun kilometara, dnevnice, predujmovi i ostali troškovi. To je napravljeno iz razloga da se pogledom na dokument korisnik lakše snalazi obzirom da su podaci logički sortirani uzduž dokumenta. Naslov svake sekcije napisan je većim fontom, podebljan je i podcrtan. Ispod njega nalaze se stavke koje ulaze u račun glavnog troška pojedine sekcije. One nisu posebno naznačene, za razliku od ukupnog troška sekcije koji je podebljan i koji ulazi u račun za ukupnu vrijednost obračuna. Sekcije su odvojene međusobno horizontalnim linijama da ih je lakše razlikovati. Sekcija s ostalim troškovima odudara od ostalih sekcija jer ona sadrži tablicu s troškovima načinjenim tokom putovanja koje direktno dohvaća iz baze podataka dok su ostale stavke prikazane pomoću parametara. Ispod tablice nalazi se ukupni izračun troškova, odnosno svota za isplatu. Svota za isplatu je cilj izrade obračuna i predstavlja najbitniju brojku na dokumentu i zbog toga je napisana najvećom veličinom slova koja je ujedno i podcrtana i podebljana. Da bi dokument bio potpun, u donji lijevi dio izvještaja je postavljeno mjesto za potpis odgovorne osobe koja je napravila obračun.

8.3 Shema baze podataka

Svaka aplikacija ovog tipa treba imati bazu podataka iz koje se izvlače kataloški podaci i u koju se spremaju obrađeni podaci. Da bi se napravila baza koja odgovara pojedinačnim potrebama potrebno je identificirati entitete koji se pojavljuju u poslu koji aplikacija treba obavljati. Za svaki entitet se definiraju atributi. Entitetima treba pridružiti identifikatore koji su atributi koji jednoznačno identificiraju određeni entitet i deskriptore koji opisuju svojstva tog entiteta. Kad su definirani entiteti i atributi potrebno je odrediti u kojim su odnosima ti entiteti.

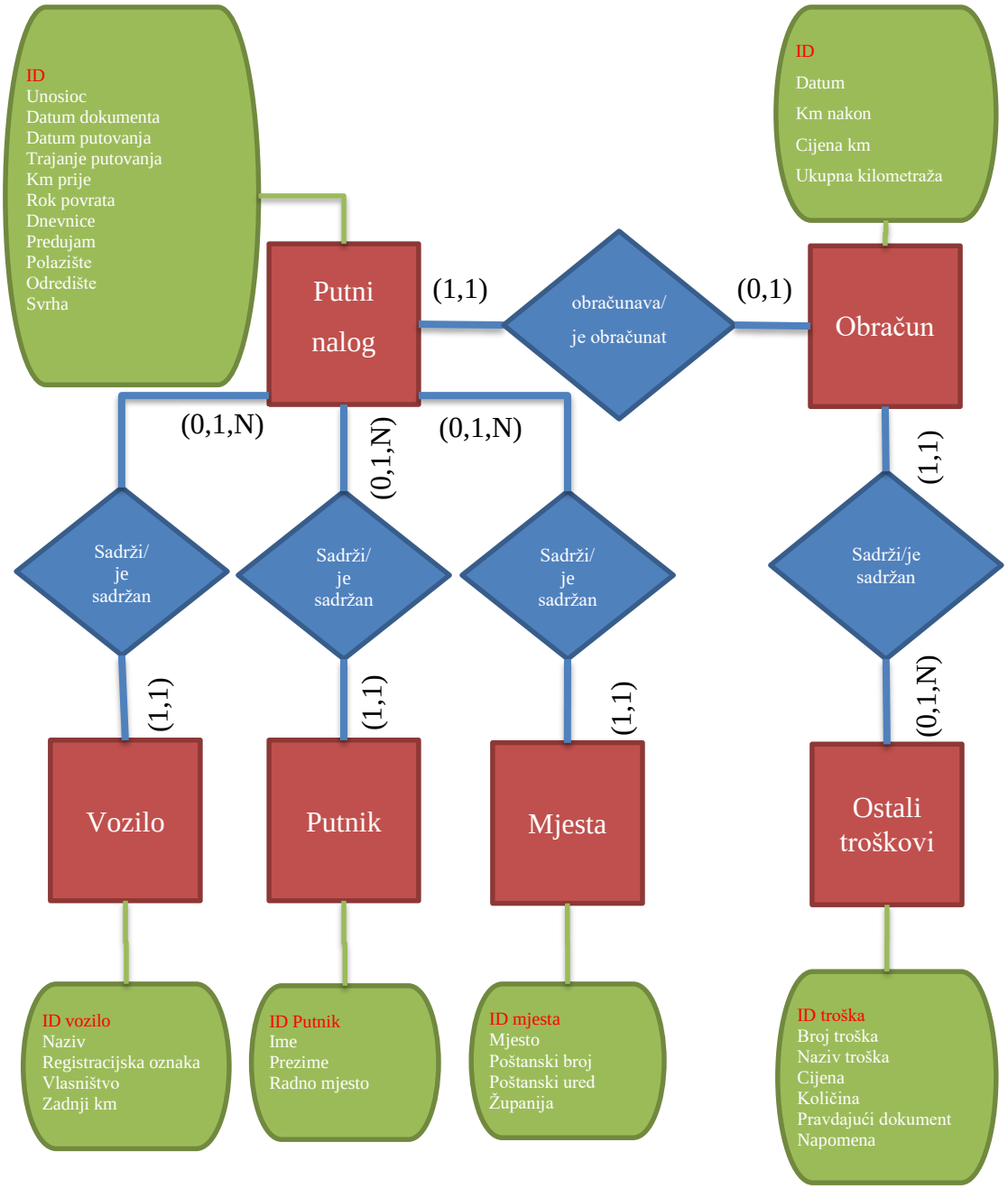
Grafički prikaz odnosa entiteta i njihovih atributa naziva se ERA modelom. Taj model je prvi korak prema stvaranju funkcionalne baze podataka. Grafički prikaz koji se koristi kao skica za bazu podataka nalazi se na slici ERA modela[24]. Ona sadrži glavne entitete koji su obojani

[24] Slika 8.3.1 ERA model

crveno, njihovi atributi su obojani zeleno, a tekst identifikacijskih atributa označen je crvenom bojom. Relacije između entiteta obojane su plavo i tip veze napisan je ISO notacijom. Počevši od glavnog entiteta koji je putni nalog, koji osim identifikatora, dnevnica, svrhe i predujmova, treba sadržavati još i informacije o putniku, mjestu i vozilu. S vozilima i putnicima ima relaciju 1:N, jer putni nalog može sadržati samo jedno i samo jedno vozilo i putnika, dok se vozilo i putnik mogu pojaviti u nijednom, jednom ili više putnih naloga. Mjesta i putni nalozi su isto u odnosu 1:N, ali mjesto se pojavljuje kod putnog naloga na mjestu polazišta i na mjestu odredišta. Putni nalog može ima jedno i samo jedno mjesto kao polazište, i jedno i samo jedno mjesto kao odredište, dok jedno mjesto može biti u nijednom, jednom ili više putnih naloga, ali ne na mjestu polazišta i odredišta u isto vrijeme. Budući da se u praksi mjesto često treba dodatno specificirati i da je takav odnos za ovaj projekt kompliciran za implementaciju i rad stoga je lakše improvizirati kreiranje finalne baze s programskim rješenjima i direktnim upisom u tablicu putnog naloga.

Obračun je još jedan entitet koji je vezan uz putni nalog. Oni su međusobno u vezi 1:1 jer se za svaki putni nalog obračunava 0 ili 1 obračun, a obračun ne može postojati bez putnog naloga i zbog toga je obračun obračunat iz jednog i samo jednog putnog naloga. Na obračun se nadovezuju ostali troškovi. Oni predstavljaju troškove nastale tokom putovanja koju mogu, ali i ne moraju postojati i koji se iz izvještaja ispunjavaju u obračun. Zbog toga se obračun i ostali troškovi nalaze u vezi 1:N, što znači da jedan trošak može biti sadržan u jednom i samo jednom obračunu, dok obračun može sadržavati nijedan,jedan ili više troškova.

Slika Error! Use the Home tab to apply Naslov 2 to the text that you want to appear here.**.1 ERA model**



8.4 Stvaranje baze podataka

Za stvaranje baze podataka korišten je Microsoft Access kao sustav za upravljanje bazom podataka. U njemu je stvorena datoteka pod nazivom „PN.accdb“ koja je baza podataka koju aplikacija koristi. Naziv datoteke i mjesto gdje je spremljena su važni za definiranje povezivnog stringa s kojim se aplikacija spaja na nju. Kada je datoteka generirana i na pravom mjestu, mora se oblikovati njezin sadržaj koji će se sastojati od tablica koje će predstavljati entitete i njihove atribute, a između njih alatom za odnose definirat će se veze između njih. Tablice su stvorene po

uzoru na ERA model. Stvorivši tablicu putnih naloga u tablici definirani su stupci koji predstavljaju atribute.

Kod definiranja svakog stupca moraju se definirati sljedeća svojstva: koji je tip podatka, u kojem je formatu prikazan, je li unos obavezan, jesu li dopušteni dvostruki ulazi i koji raspon ili veličinu imaju. U pravilu je prvi atribut identifikator koji je ujedno i primarni ključ. Primarni ključ je uvijek obavezan za unos i ne može biti prazan, a u razvijenoj aplikaciji to je broj putnog naloga koji je ujedno po tipu podatka cijeli broj, što ga čini u ovakvom slučaju praktičnim. Putni nalog kao dokument nema opcionalnih stavki tako da svi atributi u bazi moraju obavezno biti upisani. Od atributa obavezni za upis su oni koji su navedeni u ERA modelu i još se dodaje po jedan za svaki atribut koji se nadovezuje na putni nalog u vezi 1:N. Ti atributi predstavljaju vanjske ključeve i moraju biti istog tipa kao i primarni ključevi kod entiteta s kojima se nalaze u vezi. U ovom konkretnom primjeru to su putnik i vozilo.

Mjesta nisu u vezi s putnim nalozima jer improvizacijom preko aplikacije se upisuju u zasebne atribute koji su definirani kao tekstualne vrijednosti. To je napravljeno zbog toga jer se mjesta nalaze u putnom nalogu na dva mjesta i taj atribut je nerijetko sklon izmjenama od strane korisnika.

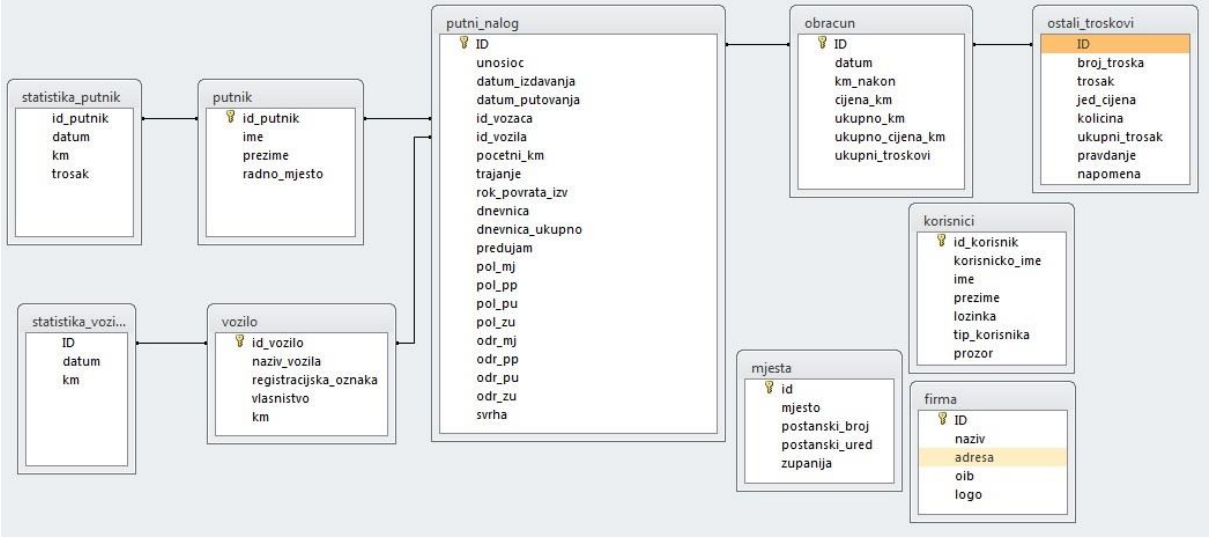
Nakon putnog naloga po istom se principu stvaraju i druge tablice pazeći na to da se točno definira tip podatka tako da odgovara potrebama dokumenta i da je pogodan za korištenje uz aplikaciju. Sukladno logici kod atributa kilometri, količine i identifikatori definirani su kao cijeli brojevi, novčane vrijednosti kao decimalni brojevi ograničeni na dvije decimale, datumi kao datumi kratkog formata, dok su imena, nazivi i opisne vrijednosti definirani kao tekst. Jedino odstupanje u toj logici nalazi se kod ostalih troškova gdje su i brojčane vrijednosti definirane kao tekst jer u trenutku njihova upisa su one već obrađene i njihov naknadan dohvat olakšava prikaz u tablici dokumenta za ispis.

Osim entiteta, atributa i veza iz prvotno zamišljenog ERA modela, u stvarnom primjeru su dodane još neke veze i tablice. Za putnike i vozila stvorene su nove tablice koje prikazuju njihovu statistiku i nazvane su statistika putnika i statistika vozila. One se nalaze u vezi 1:N s tablicama putnik i vozilo. Njihova je zadaća bilježiti kilometražu putnika i vozila kroz vrijeme i namijenjene su za neku buduću verziju aplikacije jer u ovoj se njihovi podaci zasad nigdje ne prikazuju.

U bazi još postoje dvije tablice koje nisu vezane s drugim entitetima, te dvije tablice su korisnici i poduzeće. Korisnici sadrži korisnike aplikacije, te njihova korisnička imena, lozinke, prava i preferirani režim rada. Ona služi za prijavu u program. Druga slobodna tablica je poduzeće i ona

ima samo jedan zapis u kojem su definirani podaci o poduzeću. Iako ovo nije uobičajeno, napravljeno je iz razloga da sve što poduzeće zatreba bude pohranjeno u jednoj datoteci. Primjer takve baze vidi se na slici 8.4.1[25]

Slika Error! Use the Home tab to apply Naslov 2 to the text that you want to appear here.**.1 Baza**



8.5 Dijagrami toka i algoritmi

Prije nego se počne s pisanjem koda moraju se definirati procesi koje aplikacija mora obavljati da bi bila korisna potencijalnom korisniku. Svaki proces se mora dobro definirati na način da se odredi kako počne, koje čimbenike obuhvaća i koji mu je krajnji cilj. Da se ispravno definira neki proces, mora se podijeliti na korake tako da su toliko precizni da za njih nije potrebna nikakva dodatna inteligencija. Takav precizan opis koraka u procesu naziva se algoritmom. Zbog toga što je u algoritmu najjednostavnije moguće pojašnjen proces, pogodan je za pisanje koda. Precizno napisani koraci potrebni su za pisanje koda zbog toga jer računalo nije zamišljeno da donosi zaključke, već da izvršava direktne instrukcije. Pišući algoritme za neki proces vidi se da često ne idu linearnim tokom, pa se dolazi u situaciju da se algoritam želi prikazati slikom. Grafička reprezentacija algoritma naziva se dijagramom toka. Elementi dijagrama toka su standardizirani. To znači da svaki oblik u dijagramu ima svoje značenje.

Osnovni elementi dijagrama toka su početak i kraj procesa, ulazi, izlazi, definiranje varijabli i konstanti, obrada i odluka. Međusobno su elementi spojeni strelicama u smjeru u kojem se algoritam odvija. Jednom kad se naprave dijagrami toka, pisanje koda ide brzo jer tad u jednostavnoj grafičkoj reprezentaciji vidi se u svakom koraku što se traži da se napiše u kodu da bi aplikacija radila ono za što je namijenjena. Često kod pisanja koda, elemente iz dijagrama u

[25]Slika 8.4.1 Baza podataka u praksi

kodu se piše kao funkcije koje se pozivaju po potrebi. To je praktično ukoliko se neka radnja ponavlja.

8.6 Planiranje ključnih algoritmi u aplikaciji

Kod zamišljanja ove aplikacije planirano je da postoji jedno mjesto koje će sadržavati: privremeno pohranjene podatke, zastavice o otvorenim prozorima, prijavljenim korisnicima i ostalim parametrima. Taj je koncept inspiriran radom procesora jer za svoj rad koristi registre za privremenu pohranu koji se nazivaju stack ili stog registri, i flag ili zastavice koje su u nekom registru zapisane kao jedan bit koji ovisno tome da li je jedan ili nula naznačuje neko stanje. Za tu zadaću je zadužena Stack formular. Njezina je zadaća da sadrži te privremeno pohranjene podatke i zastave. Za tu zadaću namjerno je uzet formular jer je u zamišljenom programer modu najlakši za prikaz tih podataka koji mogu biti korisni kad se stvori neka moguća greška. Taj formular je bitan jer se svi dalje razvijeni algoritmi njime koriste.

Prvi algoritam koji se služi stogom i zastavicama je prijava u program. U njemu objašnjeno kako se upisano korisničko ime i lozinka uspoređuje s vrijednostima postavljenim za programer mod ili vrijednostima iščitanima iz baze podataka. Nakon što je usporedbom omogućen ulaz privremeno se spremaju podaci iz baze u stog, stavi se zastavica da postoji prijavljen korisnik nakon se prikazuje početni formular od moda koji je specificiran u bazi.

Glavni formular služi kao mjesto gdje se otvaraju drugi prozori. Budući da se svaki formular može otvoriti samo jednom, mora se za to napraviti i adekvatan algoritam. U njemu je specificirano kako se prije prikaza formulara mora provjeriti da li je već otvoren. Informacije o otvorenom formularu zapisane su u obliku zastavice. Ako nije otvorena, onda se otvori, a ako jest, ispiše se poruka da je već otvorena. Prihvativši takav princip mora se voditi briga o tome da se zastavica vrati u stanje neotvorenoga nakon što se prozor zatvori.

Kad je prijava i navigacija kroz program riješena, potrebno je napraviti algoritme za upis podataka. Prvo se upisuju kataloški podaci jer na temelju njih će se temeljiti putni nalog i obračun. Kod kataloških vrijednosti mora se voditi briga da algoritam pokriva njihovo pretraživanje, upis, izmjenu i brisanje. Kad su kataloške vrijednosti postavljene, treba napraviti jednostavniji, njemu sličan algoritam za dohvat tih kataloških vrijednosti u formular putnog naloga. On se sastoji samo od koraka za pretraživanje, odabir i kopiranja podataka na za to predviđeno mjesto.

Jednom kad je riješeno pitanje kataloških vrijednosti može se početi s glavnim algoritmom za stvaranje putnog naloga. To je najkompleksniji algoritam u cijelom projektu. Zbog svoje

kompleksnosti podijeljen je na više modula koji je samostalan algoritam. Moduli kod putnog naloga se međusobno nadovezuju kako bi na kraju nalog bio pohranjen u bazu i ispisan.

Drugi važan dokument je obračun čiji algoritam obuhvaća učitavanje podataka iz putnog naloga i upis podataka s izvještaja nakon kojeg se taj određeni obračun pohranjuje u bazu i ispisuje. Kada je omogućen upis putnog naloga i obračuna, tada se za njih mora napraviti arhivski prikaz i mogućnost naknadnog pregleda i ispisa. Algoritmi za arhive su slični algoritmima kataloških vrijednosti. Razlika je u tome što kod arhivskih podataka nije omogućeno manipuliranje podacima, već samo filtriranje i pokretanje prikaza dokumenta.

Kad su definirani ključni algoritmi u aplikaciji, moraju se implementirati odgovarajućim redoslijedom. Prvo je implementirana prijava korisnika, a zatim navigacija. Kad je implementirana navigacija kroz aplikaciju, implementirane su kataloške vrijednosti u program prvo u vidu manipuliranja njima, a zatim dohvata istih u za to predviđena mjesta. Nakon kataloških vrijednosti počelo se s implementacijom algoritama glavnog dokumenta u aplikaciji koji je putni nalog. Putni nalog je potreban kako bi se napravio obračun koji je sljedeći po redu. Nakon što je riješen upis svih dokumenata, ostaje samo implementacija algoritma za arhive obračuna i putnih naloga. Kada se ne bi poštivao taj redoslijed, pisanje koda bi se uvelike zakompliciralo i zbog toga nije preporučljivo ići direktno u pisanje koda bez plana implementacije algoritama.

8.7 Implementacija algoritama i pisanje koda

Kada su prikazani svi algoritmi uz pomoć dijagrama toka, može se početi s izradom aplikacije. Nakon što je u Visual Studio-u napravljen novi Visual Basic projekt, počelo se s programiranjem. Kod nastanka projekta sustav stvori prvi formular i ta formular je zamišljena da prikazuje uvodni logo. Formular prikazuje logo na tri sekunde i onda postaje nevidljiv. Budući da je taj formular nevidljiv on služi kao Stack formular, a ulogu inicijalnog formulara će preuzeti drugi formular koja će se automatski otvoriti nakon što Stack formular nestane.

Stvorivši ta dva formulara pri samom početku stvaranja programa postavili su se temelji za pisanje daljnjeg koda. Implementacijom algoritama i daljnjim pisanjem koda usputno će se prema potrebama stvarati nove formularu i na njima elementi koji odgovaraju koracima opisanim u dijagramima toka. Kod pisanja koda moraju se poštivati osobna pravila jer s njima izvorni kod aplikacije postaje mnogo razumljiviji kada ga se ponovo čita ili kad se u njemu traži greška.

Kada se stvori novi element mora mu se promijeniti zadano ime u ono kakvo je propisano osobnim pravilima kako bi ono imalo smisla jer je teško raspoznati elemente kada su diferencirani samo brojem. Ukoliko se kod ovog koraka poštuje sve isplanirano, tada će se mogućnost greške smanjiti, a ukoliko se greška pojavi, olakšat će se njezin pronalazak. Ako su se dobro implementirali svi algoritmi i poštivala sva pravila, može se dobiti pouzdana aplikacija koju je lakše dalje razvijati.

8.7.1 Pokretanje aplikacije i prijava korisnika

Kao što je u planu implementacije navedeno, pokretanje aplikacije i prijava korisnika je prvi korak u pisanju koda. Uvodno su generirana dva formulara.

Prvi formular mora prikazivati logo i zbog toga je on postavljen u njegova svojstva kao pozadina. Pri pokretanju programa on se prva prikazuje, budući da je u dijagramu navedeno da bude prikazan samo tri sekunde nakon čega se mora prikazati sljedeći formular, koristi se Timer[26]. On funkcionira na način da odbrojava vrijeme nakon čega izvrši zadana radnju. Kada se stavi na formular koji prikazuje logo, u njegovim ga je svojstvima potrebno uključiti i namjestiti mu interval na vrijednost 3000, jer ta vrijednost predstavlja milisekunde. Dvostrukim klikom na Timer, sustav sam definira proceduru koja se provede kada vrijeme na Timeru istekne. U proceduru se upisuje kod s kojim se onemogućuje Timer, te se formular s logom postavlja kao sakriven, a formular koji služi za prijavu korisnika se prikazuje. Nastavak pisanja koda napisan je na formularu za prijavu. Da bi se korisnik mogao prijaviti potrebno je postaviti dva Textbox-a i gumb s kojim se potvrđuje prijava. Maksimalnu duljinu teksta treba ograničiti na duljinu koja je ograničena u bazi podataka. Textbox u koji se upisuje lozinka mora biti maskiran, a to se radi preko svojstva PasswordChar gdje je postavljen simbol „§“ koji prekriva sve upisane simbole u lozinci.

Korisničko ime i lozinka se potvrđuje klikom na gumb i za to je definirana nova procedura. U toj proceduri uzimaju se upisane vrijednosti iz Textbox-eva i uz pomoć uvjeta provjerava se ispravnost prijave. Iz dijagrama toka vidi se da iz uvjeta mogu izaći tri moguća izlaza. Uvjeti se provjeravaju preko uvjetnih naredbi If i Elseif. U prvom izrazu If naredbe uspoređuje se upisano korisničko ime i lozinka s u kodu postavljenim vrijednostima koja su postavljena za pokretanje programer moda. U tijelo uvjeta programer moda upisuje se kod koji se izvršava ukoliko je uvjet ispunjen. U tijelu treba prvo ponovo postaviti Stack formular sa svim njezinim elementima vidljivom. Vidljivost se mijenja pomoću svojstva Visible koji kod elemenata koji se žele

[26]Timer- .NET komponenta koja pokreće neki događaj nakon isteka zadanog intervala (Izvor: Microsoft Visual Studio - opis komponenti)

prikazati mora imati vrijednost True. U drugom uvjetu koji je alternativa programer modu iz baze podataka se učitava korisnik i njegovi podaci koji moraju odgovarati korisničkom imenu i lozinci. Da se nešto iščita iz baze potrebno je imati vezu s bazom, te se nju mora definirati kao novu klasu OleDB.OleDbConnection[27].

Veza za spajanje na bazu mora sadržavati povezivni string koji isto mora biti deklariran. Vrijednost povezivnog stringa zapisana je u Stack formular i od tud se dohvaća. Kada su parametri veze postavljeni tad se ona uz pomoć ugrađene procedure Open otvori i tada je aplikacija spojena na bazu. Za dohvat iz baze prvo se mora definirati tekstualna varijabla u kojoj je napisan upit za dohvat podataka o klijentu koji se želi prijaviti. Select upit koji se koristi je fiksan sve do dijela gdje se navode korisnička imena i lozinke, a taj varijabilni dio upita je napravljen jednostavnim spajanjem tekstualnih vrijednosti iz Textbox-a. Nakon formiranog upita definira se OleDb.DataAdapter[28] u kojeg je upisan upit i ime veze na bazu. On može popuniti tablicu podataka koji odgovaraju ranije napisanom upitu. Podacima u tablici podataka pristupa se tako da preko svojstva Rows pristupa redu, a svojstva Item pristupa ćeliji u kojoj se podatak nalazi. Budući da su korisnici jedinstveni uvijek će u svojstvu Rows pisati 0,a to znači da je to prvi red u tablici. Podaci iz tablice učitat će se u Stack formular tako da se može njima dalje manipulirati.

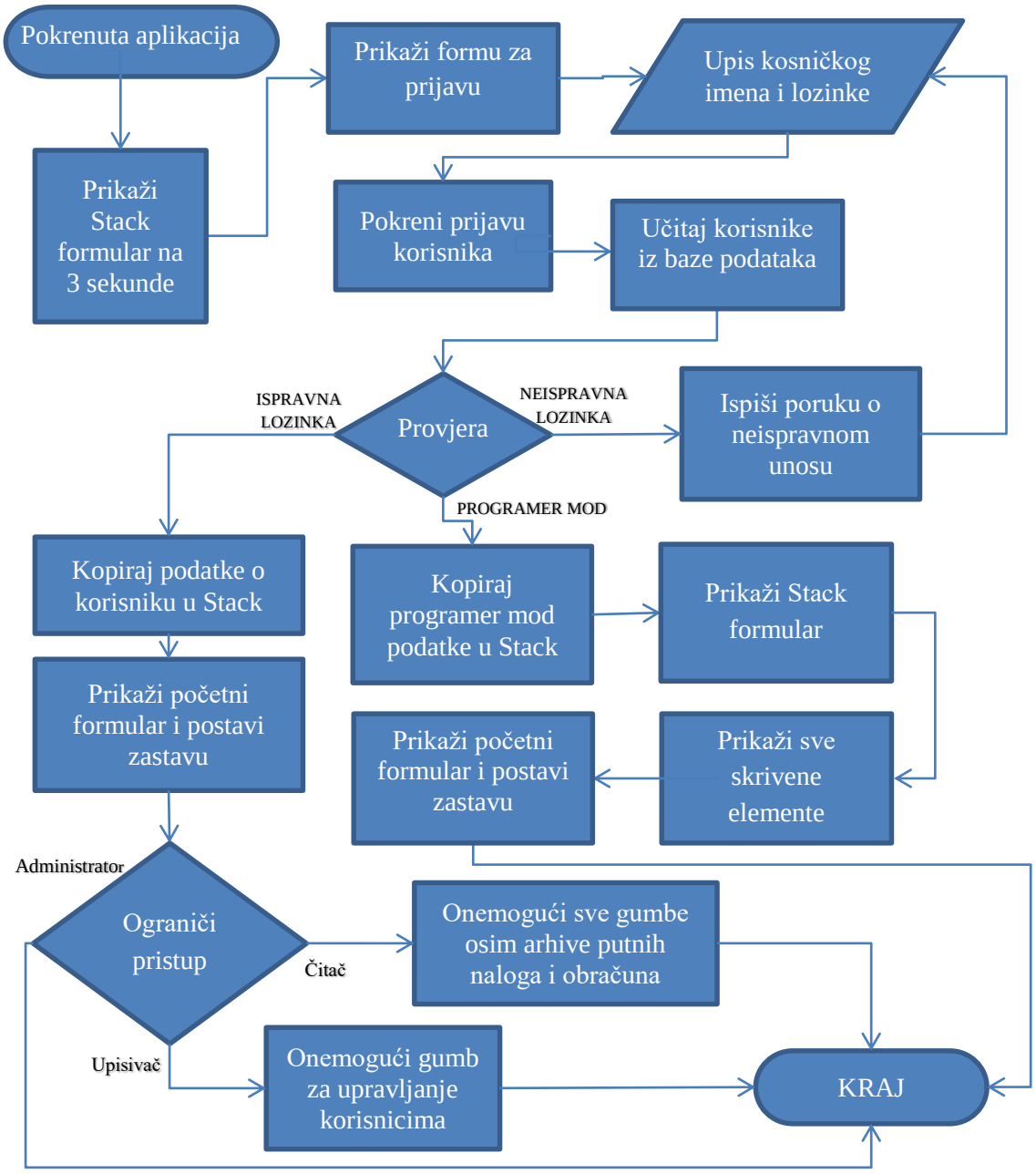
Za slučaj da je korisnik nađen u bazi, napisana je funkcija pod nazivom prikazi_potrebno, koja se poziva odmah nakon uspješnog dohvata iz baze. U tijelu te funkcije je uvjet koji ovisno o tome koji je tip korisnika preko svojstva Visible sakriva gumbe koji nisu namijene tom korisniku. Sav kod vezan uz prijavu korisnika zbog spajanja na bazu je upisan u tijelu naredbe Try, koja pokušava izvršiti kod, i ako ne uspije u tijelu njemu nadovezane naredbe Catch je napisan kod koji se izvršava za slučaj da upit nije vratio niti jedan rezultat. U tom kodu ispisuje se poruka preko Messagebox-a da korisničko ime i lozinka nisu dobro upisani i vraća se na prijavu korisnika. Na kraju se u tijelo naredbe Finally pokreće procedura koja zatvora vezu bez obzira da li je prvotni dio koda uspješno proveden ili ne. Svi implementirani koraci mogu se vidjeti na slici8.7.1.1[29].

[27]OleDB.OleDbConnection - .NET klasa koja reprezentira otvorenu vezu prema izvoru podataka (Izvor: Microsoft Visual Studio - opis komponenti)

[28] OleDb.DataAdapter - .NET klasa koja predstavlja skupinu instrukcija za punjenje setova podataka iz izvora podataka (Izvor: Microsoft Visual Studio - opis komponenti)

[29] Slika 8.7.1.1 Dijagram toka za prijavu korisnika

Slika 8.7.1.1 Dijagram toka za prijavu korisnika



8.7.2 Upravljanje prozorima

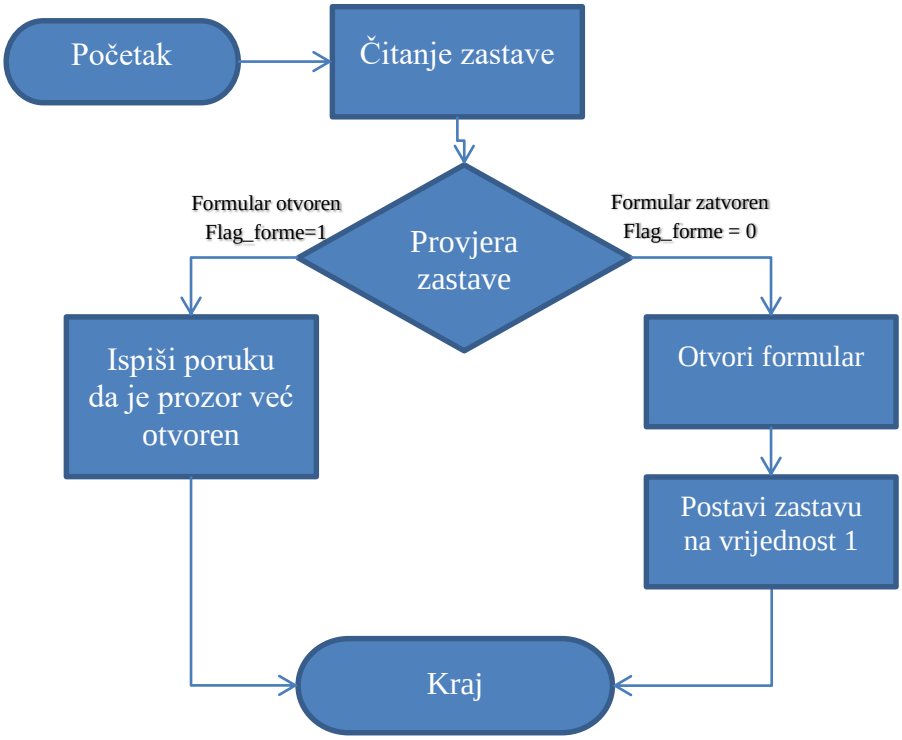
Nakon prijave u aplikaciju, treba postojati način da se pristupi omogućenim dijelovima te aplikacije. Budući da je aplikacija dizajnirana tako da se svaki ključni dio nalazi u zasebnom formularu, odnosno prozoru, treba način da mu se pristup. Kao što je vidljivo na slici 8.7.2.1[30] dijagrama toka otvaranja prozora, to je vrlo jednostavan algoritam za implementaciju. U njemu se vidi da je stanje svakog prozora opisano zastavom, što u budućem razvoju aplikacije

[30]Slika 8.7.2.1 Dijagram toka otvaranje prozora

omogućuje da se iz jednog mjesta može iščitati stanje svakog prozora odjednom. Takav pristup nije obavezan ali pridonosi kvaliteti aplikacije.

Algoritam se pokreće pritiskom na Gumb ili element padajućeg izbornika koji su namijenjeni za otvaranje prozora. U Stack formular za svaki prozor postoji globalna varijabla za svaki prozor koji se otvara ovom logikom. Ta varijabla je cjelobrojnog tipa i predstavlja zastavicu. Inicijalna vrijednost svake zastavice je 0, ali kad je formular otvoren mijenja joj se vrijednost u 1 što je vidljivo iz algoritma. Da bi se implementirao ovaj algoritam u proceduru gumba prvo se upisuje uvjet koji provjerava da li pročitana zastavica taj određeni prozor ima vrijednost 0. Ako se uvjet pokaže istinitim, preko procedure Show prikazuje se formular koji je u kodu definiran kao klasa. Kad je formular otvoren, vrijednost zastavice treba se postaviti na 1. Ukoliko uvjet nije bio ispunjen preko MessegeBox-a[31] se ispisuje poruka da je prozor već otvoren. Da bi ovaj princip funkcionirao, kod svakog zatvaranja prozora automatski se pokreće procedura koja vraća zastavicu na vrijednost 0.

Slika 8.7.2.1 Dijagram toka otvaranje prozora



[31] MessegeBox - .NET klasa koja prikazuje prozor sa tekstom i gumbom sa svrhom da informira ili uputi korisnika (Izvor: Microsoft Visual Studio - opis komponenti)

8.7.3 Manipuliranje kataloškim podacima na primjeru putnika

Za implementaciju ovog algoritma uzet je samo primjer putnika jer je princip kod mjesta i vozila skoro isti. Jedine razlike su u stavkama koje se mijenjaju i tablica u bazi koja se za određeni slučaj koristi.

Iz slike 8.7.3.1[32] na dijagramu toka vidimo nekoliko operacija koje će se izvršavati nad bazom podataka. Te operacije obuhvaćaju dohvat podataka iz baze, upis podataka u bazu, mijenjanje podataka u bazi i brisanje podataka iz baze. Te korake iz dijagrama je najbolje implementirati kao funkcije koje pozivamo kad ih se zatreba. Budući da se sve nabrojene funkcije vežu na bazu podataka, koristi se ista veza koja je definirana kao javna. Kod spajanja na bazu uvijek se koristi struktura Try-Catch-Finally kako bi se izbjegao zastoj programa. Ta struktura se nalazi u svim navedenim funkcijama koje se spajaju na bazu. Ispod svih Catch naredbi nalazi se kod koji ispisuje informaciju o grešci, a ispod svih Finally naredbi nalazi se kod koji zatvora vezu.

Dio koda koji se spaja na bazu nalazi se ispod Try naredbe. Prvi koraci koji su napisani u kodu su učitavanje putnika iz baze i njihov prikaz. Za to je napisana funkcija nazvana dohvati u kojoj se prvo otvora veza s bazom. Kada je veza otvorena, napisan je upit koji dohvaća sve podatke iz tablice putnik. Budući da je taj upit lako moguće mijenjati uz pomoć spajanja stringova gdje je kombiniran ručno uneseni dio teksta i onaj koji korisnik unaša sam, ova funkcija može posluži i za filtriranje.

Kad je upit napisan, preko adaptera puni se tablica u objektu iz klase Dataset[33], i tu istu tablicu postavimo kao izvor podataka za Datagridview. Funkciju dohvati poziva se kod samog pokretanja formulara i kod promjene teksta u Textbox-u čime je riješen problem filtriranja. Radnjama dodavanja korisnika,mijenjanja i brisanja pristupa se uz pomoć gumba. Daljnji koraci implementirani su u proceduru nastalu klikom na gumb. Kod procedure gumba za novog korisnika prvo se mora dohvatiti najveći ID iz tablice putnik i uvećati za 1. Taj korak je implementiran pozivom funkcije get_max. Ta funkcija sadrži Select Max[34] upit koji je ugrađen u OleDb instrukciju i izvršava se skalarno, što znači da pri izvšavanju upita uzima se samo prva vrijednost prvog stupca iz dobivenog rezultata. Povučena vrijednost zatim se čita kao cijeli broj koji se uveća za 1 i pomoću naredbe return vraća. Vrijednost vraćena iz funkcije se upisuje u Textbox predviđen za upis ID-a.

---

[32]Slika 8.7.3.1 Dijagram toka manipulacije kataloških vrijednosti u tablici putnik

[33] Dataset - .Net klasa koja predstavlja skupinu podataka u memoriji(Izvor: Microsoft Visual Studio - opis komponenti)

[34] Select Max – Funkcija u SQL jeziku koja vraća najveću vrijednost u stupcu

U sljedećem koraku trebalo je svojstva Visible postaviti na vrijednost False kod gumba za pokretanje promjene i brisanja, a nakon toga postaviti vrijednosti svojstva Enabled na True kod svih Textbox-ova koji trebaju sadržavati podatke korisnika. Promjenom tih svojstava sakriven je gumb koji korisnik ne treba i omogućen je upis podatka o putniku. Po istom principu gumbi za spremanje putnika i odustajanje učinjeni su vidljivim.

Kad su Textbox-evi omogućeni i odgovarajući gumbi vidljivi, tada je upis podataka o novom korisniku moguć. Da bi se upisani podaci spremili u bazu, u proceduri gumba za spremanje putnika mora biti pozvana funkcija za upis. U toj funkciji preko spajanja tekstualnih tipova podataka kombinira se Insert upit sa vrijednostima iz Textbox-ova da bi se dobio konačan upit koji se mora provesti nad bazom. On se provodi ugradnjom OleDb naredbe[35] i provođenjem te preko veze sa bazom. Naredba se provodi nad bazom preko ExecuteNonQuery[36] funkcije. Nakon upisa sljedeći korak je postaviti svojstva svih elementa u početno stanje. Iz dijagrama se vidi da kroz taj korak prolazi i kod brisanja,promjene i odustajanja, pa je onda logično da taj korak implementiran kao funkciju koja će se pozivati nakon svakog od tih slučajeva. Ta funkcija zove se Odustani jer kod koji sadrži izvršava se prilikom odustajanja. Ako je pri odabiru radnje izbor bio Promjena/Brisanje putnika tada u njegovu proceduru mora biti upisan dio koda koji je opisan u prva tri koraka nakon izbora. Prva dva koraka se provode isto kao kod gumba za novog putnika, što znači da su implementirana tako da mijenjaju svojstva elementima koje se žele omogućiti i prikazati, a pritom sakriti elemente koji se ne koriste u ovim koracima. Omogućavanje odabira korisnika klikom na redak u tablici konkretno znači da se odabirom na redak u Datagridview-u sve njegove stavke prepišu u Textbox-eve namijenje za stvake putnika gdje ih je moguće mijenjati. Za taj korak poziva se funkcija nazvana povezi_textboxe. Unutar te funkcije postoji cjelobrojna varijabla koja predstavlja broj reda u kojem je trenutno odabrani putnik. U ostatku funkcije uz pomoć te varijable pristupa se podacima putnika koji je odabran gdje se sa svakim odabirom upisuju njegovi podaci u za njih predviđene Textbox-ove.

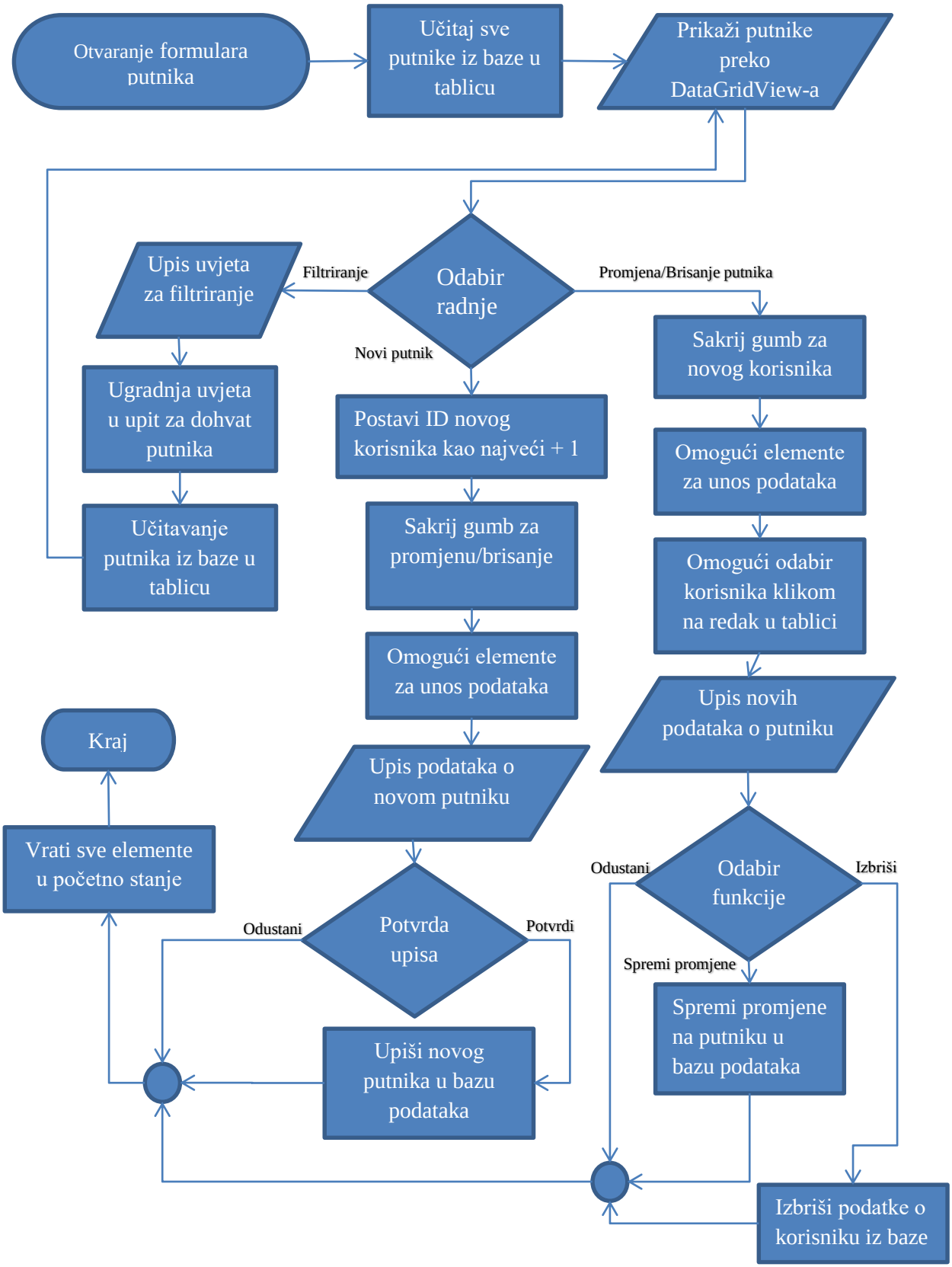
Jednom kad su podaci putnika upisani lako ih je mijenjati. U trenutku kad je moguće mijenjati podatke, gumbi za odustajanje, promjenu zapisa i brisanje zapisa su vidljivi. Gumb za odustajanje sadrži samo funkciju za vračanje elemenata u početno stanje. Gumbi za spremanje promjena i brisanje pozivaju svaki svoju funkciju čiji sadržaj sadrži dio koda koji izvršava upit nad bazom, po kojem se te funkcije razlikuju.

[35] OleDb naredba ili OleDb command - .NET klasa koja predstavlja upit ili uskladištenu proceduru koja se izvodi nad bazom podataka (Izvor: Microsoft Visual Studio - opis komponenti)

[36] ExecuteNonQuery - .NET ugrađena funkcija koja zajedno izvršava upit nad bazom podataka i vraća količinu zahvaćenih redaka kao cijeli broj (Izvor: Microsoft Visual Studio - opis komponenti)

U funkciji za promjenu zapisa koristi se Update upit, dok u funkciji za brisanje koristi se Delete upit. Vrijednosti za potrebe upita uzete su iz Textbox-ova u koje su učitani podaci korisnika. Nakon tih funkcija u proceduri oba gumba poziva se još funkcija za vračanje svih elemenata u početno stanje s čim je cijeli algoritam u potpunosti implementiran.

Slika 8.7.3.1 Dijagram toka manipulacije kataloških vrijednosti u tablici putnik



8.7.4 Dohvat podatka iz kataloga na primjeru putnika

Dohvat iz kataloga uveden je zbog bržeg upisa. Zbog toga jer se ovim postupkom zaobilazi ručni upis, ujedno je i smanjena mogućnost greške. Algoritam za dohvat implementira se na formular za dohvat. Iako se ovaj algoritam odnosi na putnika, razlika između tog algoritma i ostalih za dohvat je u tablici iz baze koju čitaju i mjesto u koje se ti podaci dohvaćaju.

Cijeli postupak počinje pozivom formulara za dohvat iz formulara putnih naloga i završava njegovim upisom u taj isti formular. Za implementaciju algoritma za dohvat koristi se dijagram dohvata kataloških vrijednosti. Iz njega se vidi da ima mnogo zajedničkih elemenata sa dijagramom toka manipulacije kataloških vrijednosti. To je zbog toga jer koristi isti princip filtriranja i prikaza. Jednom kad je formular otvoren postavlja se zastava u Stack formularu na vrijednost 1, nakon čega učitavaju podaci o putnicima iz baze i prikazuju u Datagridview-u. Prikaz je implementiran pozivom iste funkcije kao kod manipulacije kataloškim vrijednostima. Kraj filtriranja prekida pritisak na tipku dolje. Prekidom filtriranja omogućuje se izbor odabira iz filtrirane tablice Datagridview-a.

Za odabir ili promjenu odabira kreirana je procedura koja poziva identičnu funkciju kao i kod manipulacije kataloškim vrijednostima. Ta funkcija sadrži dio koda koji prepisuje podatke iz odabranog reda u Textbox-ove koji su namijenjeni za prikaz svih podataka korisnika. Kada je odabran putnik i kad su njegovi podaci vidljivi u Textbox-ovima, mora se potvrditi njegov izbor. Za potvrdu izbora generirana je procedura kod klika na gumb za potvrdu, ali kao alternativa tu istu proceduru pokreće pritisak na tipku Enter kada u trenutku kad se bira putnik. U proceduri u kojoj se potvrđuje izbor putnika prvo se vrijednosti iz Textbox-ova izbora prepišu u Textbox-eve na formularu za upis putnog naloga. Kad su podaci upisani, preostaje samo zastavicu o otvorenom formularu vratiti na vrijednost 0, nakon čega treba zatvoriti formular za dohvat. Postupak dohvata vidi se na slici 8.7.4.1[37]

[37] Slika 8.7.4.1 Dijagram toka dohvata na primjeru putnika

Slika 8.7.4.1 Dijagram toka dohvata na primjeru putnika

8.7.5 Dohvat i prikaz izvještaja na primjeru putnog naloga

Kada se neki putni nalog koji je tek upisan ili neki postojeći iz arhive želi prikazati, za treba imati formular sa adekvatnim elementima i kodom koji će prikazati taj dokument i omogućiti mu ispis. Za potrebe prikaza koristi se formular koji po cijeloj površini ima ReportViewer[38]. Algoritam vidljiv na slici dijagrama toka prikaza putnog naloga implementiran je u proceduru pokretanja formulara. Budući da se taj formular s tim algoritmom poziva po potrebi sa više mjesta, napravljen je tako da dio programa koji poziva prikaz izvještaja privremeno pohrani broj izvještaja u Stack formular, a onda prilikom pokretanja formulara s izvještajem taj se broj učita i na temelju njega prikaže putni nalog koji se traži. Identifikator putnog naloga zapisan je u

[38] ReportViewer -.NET komponenta za prikaz izvještaja (Izvor: Microsoft Visual Studio - opis komponenti)

skriveni Textbox na Stack formularu. Ukoliko zapis ne postoji, što znači da je njegov Textbox prazan ispisuje je poruka o grešci preko MessageBox-a. Ukoliko je identifikator uspješno pročitan, ugrađuje se u upit s kojim učitavaju podaci iz tablica putni_nalog, putnik i vozilo u dataset. Podatke je potrebno premjestiti u varijable tekstualnog tipa. Ukoliko postoji neki podatak koji nije tekstualnog tipa, koristi se ugrađena funkcija ToString[39].

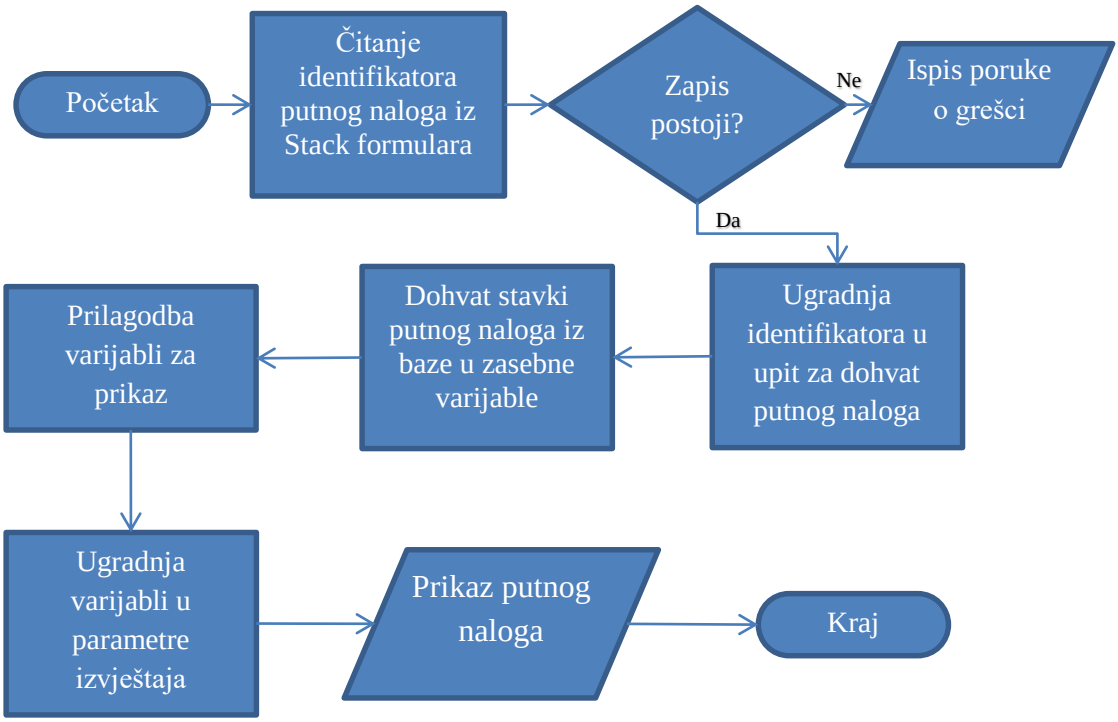
Jednom kad su vrijednosti smještene u varijable treba ih prilagoditi za prikaz na izvještaju. Prilagodba podrazumijeva promjenu vrijednosti putnog naloga iz zasebnih varijabli u oblik koji prilagođen za ispis. To je implementirano tako da su točke iz decimalnih brojeva zamijenjene zarezima,a kod datuma su izuzeti sati,minute i sekunde. Za tu radnju koristi se ugrađena funkcija Replace.

Kada su svi podaci prilagođeni, treba ih staviti u parametre koje izvještaj mora prikazati. Za potrebe prikaza trebalo je za svaku stavku u izvještaju napraviti parametar. Parametar se definira tako da se u njega upisuju dvije stavke. Prva stavka je ime parametra na koji se veže u izvještaju. Druga stavka je tekst koji taj parametar sadrži i u njega su stavljene tekstualne varijable koje su prilagođene za prikaz na izvještaju. Da bi se parametri prikazali izvještaju koji predstavlja putni nalog treba ih se dodati u njega. Parametri se dodaju tako da u ReportViewer namjesti izvještaj putnog naloga kao zadani i preko funkcije SetParameters postave postave otprije definirani parametri. Broj unesenih parametara i broj parametara definiranih u izvještaju putnog naloga mora biti isti, i nakon što su ti uvjeti ispunjeni može se pokrenuti prikaz za ispis. Koraci opisani u ovom postupku vidljivi su na slici 8.7.5.1[40]

[39] ToString – . NET ugrađena funkcija koja unesenu vrijednost vraća kao tekstualan tip podatka(Izvor: Microsoft Visual Studio - opis komponenti)

[40] Slika 8.7.5.1 Dijagram toka dohvata i prikaza izvještaja na primjeru putnih naloga

Slika 8.7.5.1 Dijagram toka dohvata i prikaza izvještaja na primjeru putnih naloga



8.7.6 Upis putnog naloga

Da bi se mogao upisati putni nalog, trebalo je prvo biti riješeno pitanje manipulacije kataloškim vrijednostima, dohvata iz kataloga i prikaza dokumenta. Upis je podijeljen po tipu podataka koji se upisuju. Algoritam po kojem pisan kod za ovaj dio programa napisan je tako da upis bude maksimalno brz, pri čemu se mehanizmima za provjeru kod svakog dijela programa želi spriječiti mogućnost pogreške.

Budući da je sam algoritam upisa relativno kompleksan, podijeljen je na četiri modula. Svaki modul promatra se kao zasebni algoritam koji se nadovezuje na ostale module. Elementi koji se vežu na određeni modul su smješteni u zasebne Groupbox-eve. To je napravljeno zato da se mogu omogućiti ili onemogućiti svi elementi nekog modula odjednom.

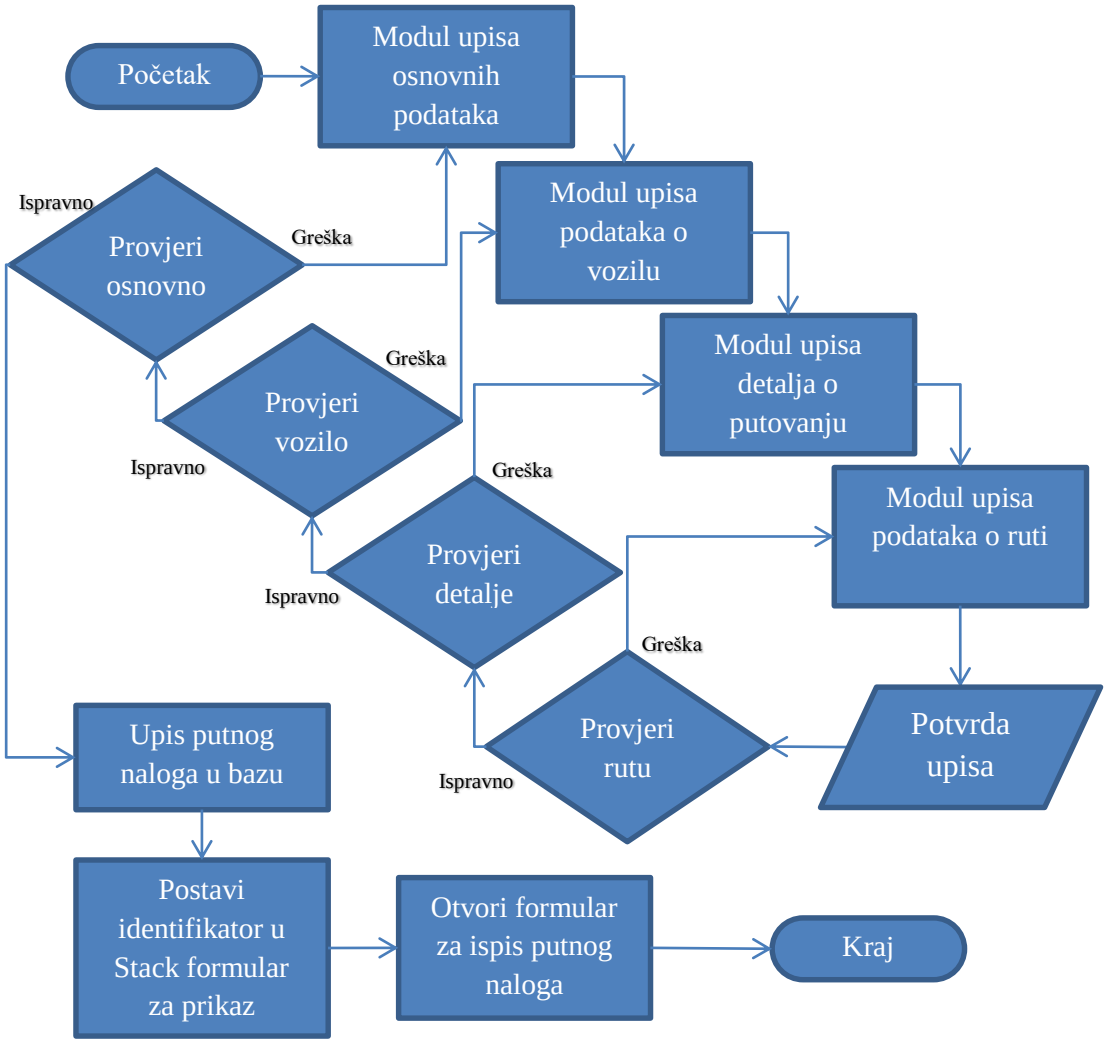
Iz dijagrama toka upisa putnog naloga[41] vidi se da se kroz module slijedno upisuju podaci i nakon potvrde upisa obrnutim redoslijedom provjeravaju se upisani podaci. Ukoliko se usred provjere identificira greška, aktivira se modul na kojem je greška kako bi se ista ispravila. Provjere su napravljene tako da se uz pomoć If uvjeta provjerava da li su upisani podaci dobro upisani, pri čemu se provjerava duljina zapisa i tip podatka koji je upisan. Ukoliko neki uvjet nije

[41]Slika 8.7.6.1 Dijagram toka upisa putnog naloga

zadovoljen, omogući se ponovni upis pri čemu se element u kojem je krivi upis oboji u crveno. Boja pozadine nekog elementa mijenja se pomoću svojstva BackColor.

Nakon što su upisani podaci prošli sve provjere, upisuju se u bazi u tablicu putnog naloga. Upis je implementiran na isti princip kao kod manipulacije kataloškim vrijednostima. Nakon uspješnog upisa u bazu, identifikator tj. broj putnog naloga se upisuje u Stack formular na mjesto putnog naloga za prikaz, nakon čega se otvara formular za prikaz putnog naloga. Otvaranjem formulara za prikaz čita se postavljeni identifikator putnog naloga i preko njega se otvara tek upisani putni nalog kojeg se tad može ispisati.

Slika 8.7.6.1 Dijagram toka upisa putnog naloga



Modul upisa osnovnih podataka je prvi modul u koji se upisuju podaci potrebni za stvaranje novog putnog naloga. U osnovne podatke ubrajaju se broj putnog naloga, datum dokumenta i putovanja, ime,prezime i radno mjesto putnika. Tijek upisa vidljiv je u Dijagramu toka modula upisa osnovnih podataka. Prvi korak je dohvat zadnjeg broja putnog naloga i njegovo uvećanje za jedan kako bi predstavljao identifikator naloga koji se upisuje. Taj je korak implementiran na

isti način kao kod manipulacije kataloškim vrijednostima. Poziva se funkcija get_max samo što je ovaj put u njoj izmijenjen upit tako da uzima najveći broj putnog naloga iz tablice putni_nalog. Kad je putni nalog dohvaćen, treba upisati datume dokumenta i putovanja u za njih namijenjene DateTimePicker-e.

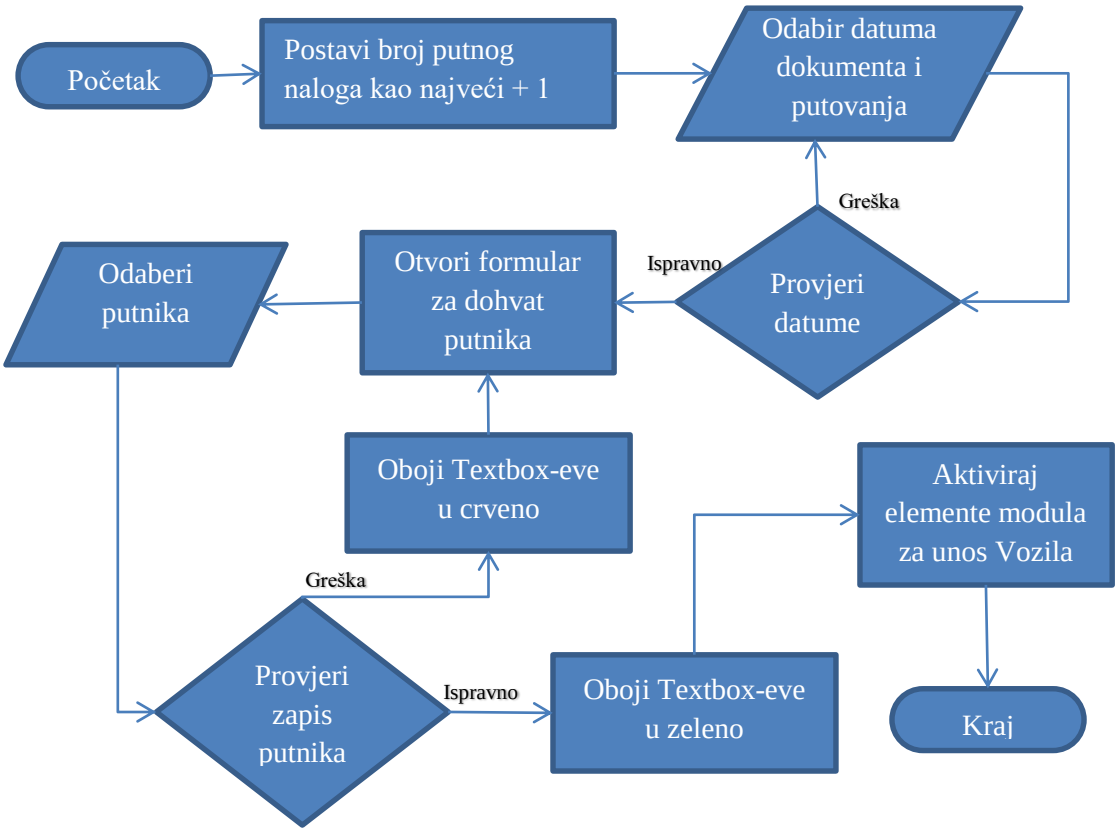
Nakon unosa datuma pokreće dio koda koji provjerava da datum putovanja nije raniji od datuma dokumenta. Taj dio napravljen je preko If uvjeta koji ako nije zadovoljen vraća se na upis datuma koji je neispravan. Ukoliko se ne upišu datumi koriste se zadani današnji datum za vrijednost oba datuma. Podaci putnika se dohvaćaju preko formulara za dohvat. On se može pozvati pritiskom na bilo koju tipku prilikom upisa u žuti Textbox. Korak u kojem se odabire putnik odnosi se na dijagram toka dohvat kataloških vrijednosti na primjeru putnika. Kad su vrijednosti dohvaćene i upisane, izvršava se procedura u kojoj se preko If uvjeta provjera minimalna duljina kod svake stavke. Ukoliko je pogrešno upisana neka stavka, u svojstvo BackColor postavi se crvena boja čime se pozadinska boja Textbox-eva promijeni u crveno,

nakon čega se cijeli postupak upisa vraća na odabir putnika. Duljina se mjeri preko funkcije Len[42]. Ukoliko je If uvjet ispunjen i sve stavke zadovoljavaju minimalnu duljinu, svojstvo BackColor se promijeni tako da Textbox-evi budu zeleni, nakon čega se mora aktivirati modul za unos vozila. Modul je jednostavno aktivirati jer su svi njegovi elementi sadržani u jednom Groupbox-u i promjenom njegovog svojstva Enabled na vrijednost True, on i svi njegovi elementi postaju spremni za upis. Koraci implementira za ovaj modul vidljivi su na dijagramu toka koji se nalazi na slici Slika 8.7.6.2[43]

[42] Len - .NET ugrađena funkcija koja vraća broj znakova u stringu kao cijeli broj (Izvor: Microsoft Visual Studio - opis komponenti)

[43] Slika 8.7.6.2 Dijagram toka modula za unos osnovnih podataka putnog naloga

Slika 8.7.6.2 Dijagram toka modula za unos osnovnih podataka putnog naloga



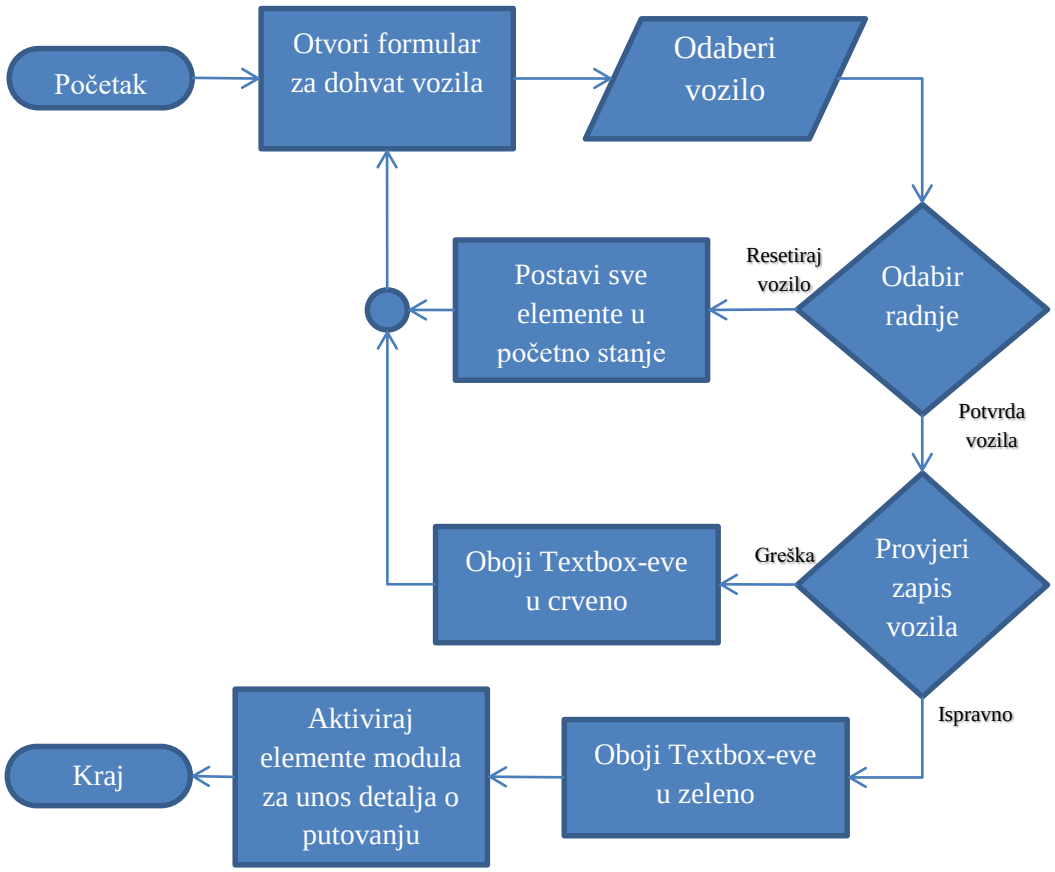
Modul upisa podataka o vozilu sadrži podatke o modelu vozila, registracijskoj oznaci i kilometrima vozila prije putovanja. Najjednostavniji je od svih modula jer ima najmanje elemenata i provjera zapisa. Na modulu je jedan žuto obojani Textbox kod kojeg se pritiskom na tipku pokreće procedura koja otvara formular za dohvat vozila. Formular za dohvat vozila je na dijagramu toka na slici 8.7.6.3[44] prikazan kao ulazni element dok je zapravo kao kod mjesta i putnika zasebni algoritam orijentiran brzom unosu podataka.

Nakon što su povučeni i upisani podaci u za to predviđene Textbox-eve mora se birati daljnja radnja. Radnja se bira preko gumba Reset i Potvrda vozila. Kod svakog gumba postoji procedura koja se pokreće klikom na gumb. U procedurama se nalazi dio koda koji se odnosi na ostale elemente iza odabira radnje u dijagramu toka. Za slučaj resetiranja je u proceduru upisan dio koda koji sve vrijednosti elemenata u modulu vraća na početnu vrijednost. Ukoliko se klikne na gumb za potvrđivanje vozila pokreće se procedura u kojoj se provjerava ispravnost unosa preko If uvjeta. Provjerava se minimalna duljina teksta koja je upisana i ukoliko nije ispunjen uvjet promijeni se svojstva elementa u koje su upisani podaci tako da imaju crvenu boju pozadine, nakon čega se postupak vraća na početak sa funkcijom Focus kod elementa kojim se poziva

[44]Slika 8.7.6.3 Dijagram toka modula unosa vozila

formular za dohvat vozila. Ukoliko je uvjet istinit, svojstva elemenata u koje su upisani podaci se postave tako da imaju zelenu boju pozadine nakon čega se aktivira modul sa detaljima putovanja tako da se omogući Groupbox u kojem se nalazi.

Slika 8.7.6.3 Dijagram toka modula unosa vozila



Modul s detaljima o putovanju sadrži samo brojčane vrijednosti i ujedno je jedini modul koji za ne poziva ni jedan formular za dohvat podataka iz kataloga. Svi ulazni elementi su Numericupdown jer su praktičniji za brojčani tip upisa od Textbox-a. Oni dopuštaju da se u njima definira minimalna i maksimalna vrijednost, isto kao i koliko decimala ima koji broj. Ako su ta svojstva definirana za svaku stavku, broj mogućih grešaka se svodi na minimum. Modul se zasniva na koracima prikazanim na dijagramu toka upisa detalja putovanja na slici 8.7.6.4[45]. Prvi korak je upis broja dana putovanja. Ako se prilikom upisa promijeni početna zadana vrijednost pokreće se procedura u kojoj je dio koda koji mijenja svojstva elementa za upis roka povrata izvještaja tako da je minimalna vrijednost koja se može upisati broj dana proveden na putu uvećan za tri. Uvećava se za tri jer se ide s pretpostavkom da putnik mora dobiti barem 3 dana za vraćanje dokumenta. Minimalni upisani broj dana namješta se preko svojstva Minimum. U sljedećem koraku rok povrata neće se moći upisati ukoliko je ispod minimuma.

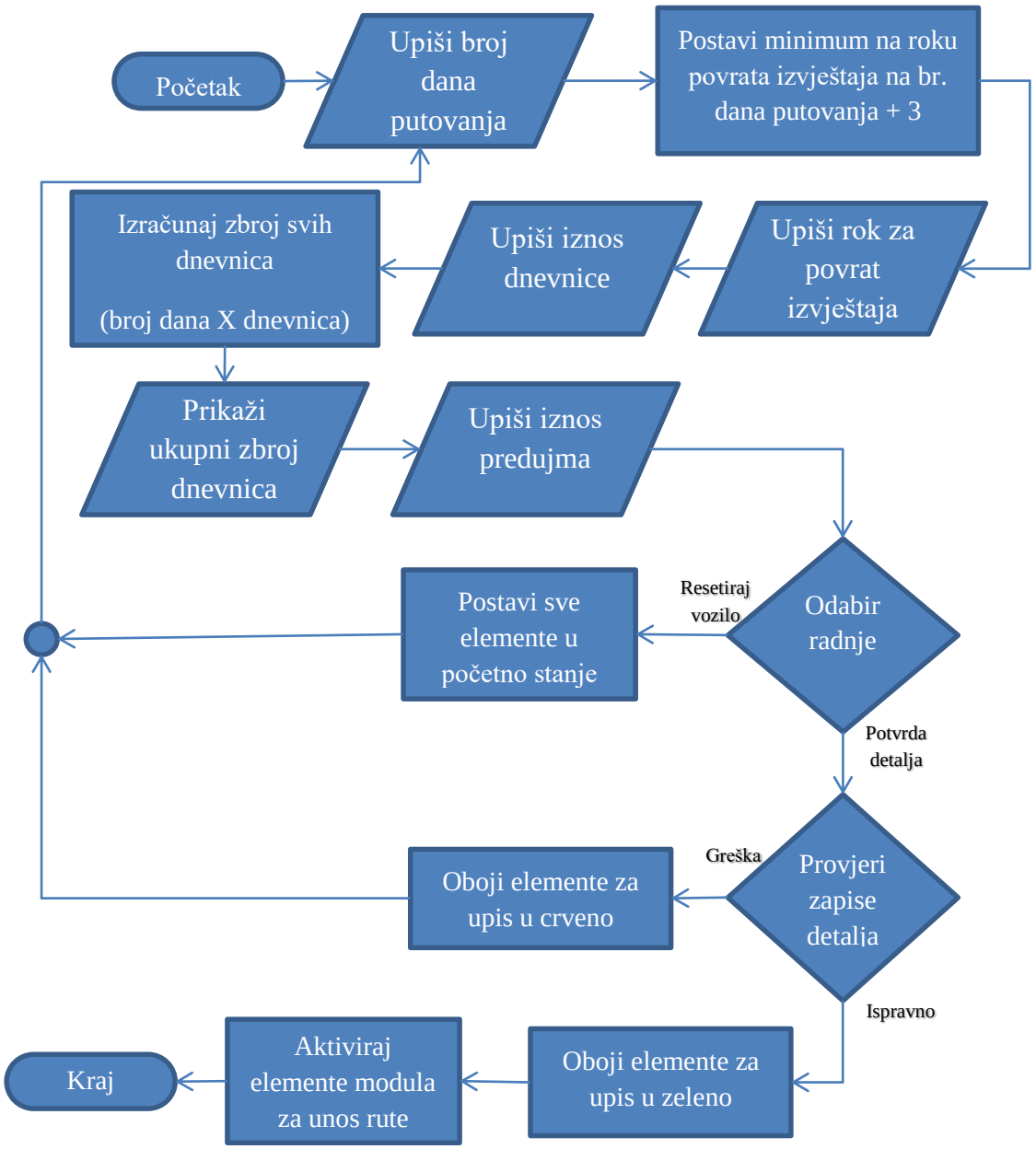
[45]Slika 8.7.6.4 Dijagram toka modula za upis detalja o putovanju (Autor: Marko Smetko)

Nakon upisa roka slijedi upis iznosa dnevnice. On sadrži dvije decimale i prilikom promijene bilokakve vrijednosti kod inicijalno postavljene dnevnice automatski se računa ukupni broj dnevnica. Ukupni broj dnevnica dobiven je množenjem vrijednosti iz upisnih elemenata broja dana putovanja i iznosa dnevnice. Dobiveni umnožak prikazan je u tekstu Label-e.

Zadnji upis koji se upisuje je predujam. Budući da se radi o kunama isto sadrži dvije decimale kao i dnevnica. Nakon što su svi podaci upisani, imamo dva gumba za biranje daljnje radnje.

Klikom na gumb za resetiranje podataka pokrećemo procedura koja sadrži dio koda kojim vračamo sve elemente na početno stanje. Klikom na gumb za potvrdu podataka pokreće se procedura koja provjerava preko If uvjeta jesu li upisane vrijednosti u okvirima koji bi bili realni. Ukoliko uvjet nije zadovoljen pokreće se u dio koda koji oboji elemente za upis u crveno nakon čega se ponovo postupak vraća na početak upisa. Ukoliko je uvjet zadovoljen elementi se oboje u zeleno i aktivira se modul za unos rute tako da se svojstvo Enabled kod njegovog Groupbox-a postavi na vrijednost True.

Slika 8.7.6.4 Dijagram toka modula za upis detalja o putovanju

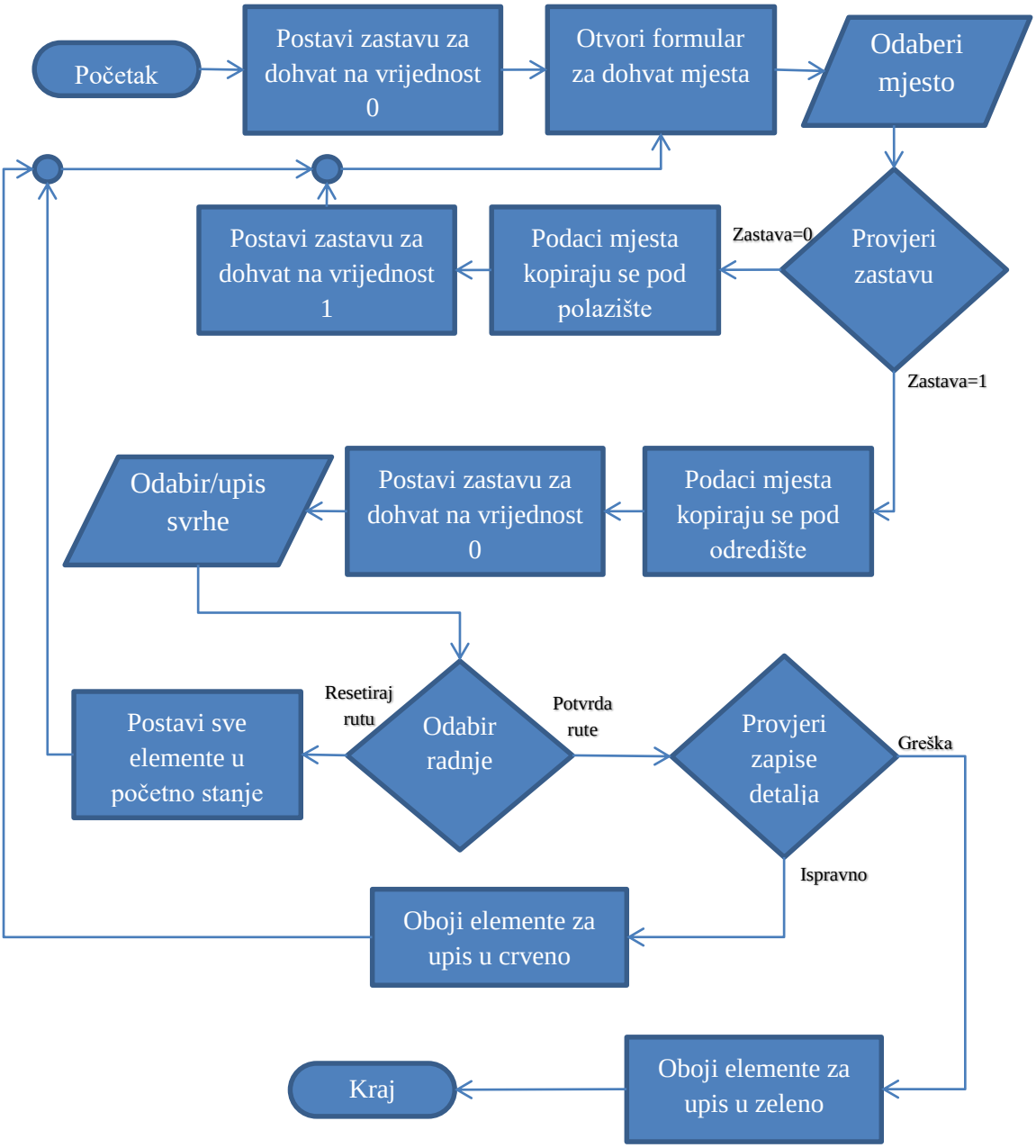


Modul za upis rute služi za upis odredišta, polazišta i svrhe putovanja, pri čemu ima jedinstven način upisa polazišta i odredišta uz pomoć jednog formulara za dohvat. Iz dijagrama toka upisa rute na slici 8.7.6.5[46] vidljivo je da je prvi korak postavljanje zastavice na vrijednost 0. Zastavica je javna varijabla cjelobrojnog tipa kojoj proceduri pokretanja se dodaje početna vrijednost. Kada je zastavici vrijednost 0 tada će daljnjim koracima mjesto koje se dohvaća upisati pod polazište. U proceduri koja se pokreće preko pritiskom tipke u žutom Textbox-u nalazi se dio koda koji pokreće formular za dohvat. Nakon odabira u formularu za dohvat mjesta, preko If uvjeta koji se nalazi u proceduri gumba za potvrdu na formularu za dohvat, provjerava se zastava

[46] Slika 8.7.6.5 Dijagram toka modula za unos rute

i ukoliko je 0, odabrano mjesto kopira se u Textbox-eve predviđene za polazište. Kad je polazište odabrano vrijednost zastave se u istoj proceduri promijeni na vrijednost 1 i postupak se vraća ponovo na poziv formulara za dohvat mjesta nakon čega se cijeli postupak do provjere zastave ponavlja. Ukoliko je preko If uvjeta potvrđeno da je vrijednosti iz zastave jednaka 1, mjesto koje se dohvaća upisuje se u Textbox-eve namijene odredište. Kada su upisani odredište i polazište zastavica se vraća natrag na 0 ukoliko se kasnije odluči ponoviti upis. Sljedeće se preko Combobox-a upisuje svrha putovanja, ili se odabere jedna od već postavljenih vrijednosti. Kad su upisane sve stavke radnja se bira kao i u dva prošla modula s gumbima za odustajanje i potvrdu. Ukoliko se klikne na gumb za odustajanje pokreće se procedura koja sadrži dio koda koji sve elemente koji sadrže upisane podatke vraća na početno stanje. Procedura pokrenuta klikom na gumb za potvrdu sadrži dio koda koji preko If uvjeta provjerava jesu li upisane vrijednosti u minimalnim okvirima. Tu se konkretno provjerava da upis nije prazan. Ukoliko uvjet nije zadovoljen, svojstvo Backcolor se kod upisnih elemenata promijeni tako da prikazuje crvenu boju nakon čega se postupak vraća na ponovni upis. Ukoliko je uvjet zadovoljen, na isti princip upisni elementi dobivaju zelenu boju poleđine.

Slika 8.7.6.5 Dijagram toka modula za unos rute



8.7.7 Obračun putovanja

Obračun je zasnovan na podacima iz putnog naloga i podacima koje unosimo iz izvještaja nakon putovanja. Budući da su identifikator putnog naloga i obračuna isti, prilikom upisa identifikatora, stavke putnog naloga moraju se same upisati i pribrojati u obračun, dok se ostali troškovi moraju obračunavati redom kako se upisuju. Algoritam obračuna putovanje implementiran je na formular za obračun troškova i prikazan je na slici 8.7.7.1[47].

[47] Slika 8.7.7.1 Dijagram toka unosa i izmjena obračuna troškova

Proces počinje upisom broja putnog naloga putnog naloga u Textbox, nakon čega se pokreće procedura inicijalizirana klikom na gumb ili pritiskom tipke Enter nakon upisa. Procedura sadrži upit koji dohvaća vrijednosti iz tablice putnog naloga koji su u vezi s upisanim identifikatorom. Dohvaćeni podaci učitavaju se u tablicu Dataset-a. Kad su podaci učitani treba provjeriti da li postoje redovi u toj tablici. Provjera je implementirana tako da se ugrađenom funkcijom Count dobiva se broj redaka i taj broj se postavlja u If uvjet gdje se provjerava da li postoji jedan ili nijedan redak zapisa. Ukoliko nema nijedan zapis ispisuje se poruka preko Messagebox-a o tome da putni nalog s upisanim identifikatorom ne postoji. Ukoliko je ispunjen uvjet da broj redaka nije nula, učitani podaci iz tablice prepišu se u za to predviđene Label-e.

Sljedeće se identifikator upiše u upite za dohvat obračuna troškova i ostalih troškova. Ovaj korak služi za to, a se vidi da li je napravljen već obračun za taj putni nalog. Po istom principu kao kod putnog naloga dohvaćaju se podaci iz baze o obračunu i ostalim troškovima u tablice dataset-a. Provjerom broja zapisa u dohvaćenoj tablici obračuna provjerava se da li je obračun napravljen ili ne. Broj redaka dobiven je funkcijom Count i uvršten je u If uvjet koji provjerava da li je broj redaka nula ili jedan. Ukoliko je uvjet zadovoljen i broj redaka vraćen funkcijom Count jednak nuli, proces se nastavlja direktno na elementu za upis kilometara nakon putovanja. Ukoliko je ispunjen uvjet da je broj redaka jednak jedan, podaci iz tablice obračuna se upisuju u upisne elemente obračuna, a podaci iz tablice ostalih troškova se upisuju u Datagridview.

Kad proces dođe do koraka u kojem se upisuju ili mijenjaju završni kilometri, mora se voditi briga da budu veći od početnih kilometara jer u protivnom se radi o grešci. Nelogično je da vozilo ima manje kilometra nakon putovanja, pa se zbog toga, kada se s upisa kilometraže želi prijeći na sljedeći upisni element pokreće procedura koja provjerava jesu li upisane vrijednosti ispravne. Upisane vrijednosti su tekstualnog tipa i pretvaraju se u cjelobrojne varijable preko funkcije ToInt32 koja se poziva iz klase Convert. Dobivene brojčane vrijednosti se stavljaju u If uvjet gdje se provjera je li kilometraža nakon putovanja veća. Ukoliko uvjet nije ispunjen ispisuje se poruka da je upis neispravan, dok ukoliko je uvjet ispunjen proces se seli na upis cijene po kilometru.

Upisom cijene pokreće se procedura koja računa ukupnu cijenu kilometraže koja je umnožak razlike u kilometrima i cijene po kilometru. Kad je cijena izračunata, procedura omogućuje sve elemente potrebne za manipulaciju ostalim troškovima. Upis novog troška je zadana radnja kad korisnik dođe do upisa troškova, dok se odabir promijene ili brisanja troška pokreće dvostrukim klikom na trošak u tablici Datagridview-a. U toj proceduri vrijednosti iz odabranog retka se prepisuju u upisne elemente za trošak. Prepisane vrijednosti se tamo mogu mijenjati dok se

krajnja radnja potvrđuje s gumbima koji imaju procedure za mijenjanje zapisa u tablici i brisanje zapisa iz tablice. Kod upisa novog troška aktivan je samo gumb za potvrdu troška. Klikom na njega pokreće se procedura koja postavlja novi trošak u tablicu i oslobađa upisne elemente za novi upis. Nakon svake radnje upisa, mijenjanja i brisanja poziva se funkcija koja zbraja sve troškove i oduzima predujam kao bi se izračunala krajnja vrijednost obračuna.

Kad su uneseni svi troškovi obračun potvrdimo preko gumba. U proceduri gumba prvo se poziva funkcija koja upisuje podatke o obračunu u bazu gdje je tablica s obračunima. Nakon što je unesen obračun, u istoj proceduri se poziva funkcija koja upisuje ostale troškove u bazu podataka. Budući da ostalih troškova ima više i zapisani su u tablici, njihov se upis izvršava pomoću For petlje koja počinje s nulom i završava s brojem redaka umanjenim za jedan. Unutar petlje se red tablice upisuje Insert upit pomoću kojeg se upisuju troškovi jedan po jedan. Kad je upis u bazu u potpunosti izvršen, identifikator obračuna se upisuje u Stack formu nakon čega se otvara formular za prikaz obračuna koja taj identifikator pročita i na temelju njega prikaže obračun za ispis.

Slika 8.7.7.1 Dijagram toka unosa i izmjena obračuna troškova

8.7.8 Arhiviranje na primjeru obračuna

Algoritmi arhive najsličniji su algoritmima u formularima za dohvat i formularima za manipuliranje kataloškim podacima. Arhiva je zamišljena kao tablični prikaz generiranih dokumenata s mogućnošću filtriranja sadržaja gdje bi se kod odabira mogle prikazivati sve stavke kako bi korisnik mogao odabrati traženi dokument za potrebe njegovog naknadnog ispisa. Aplikacija ima arhivu putnih naloga i arhivu obračuna troškova. Obje arhive su po algoritmima i stilovima gotovo identične, jedina je razlika u tome što arhiva obračuna ima mehanizam koji uz stavke putnog naloga prikazuje i stavke obračuna pri čemu onemogućuje ispis ukoliko obračun ne postoji. Za arhivu se koristi formular sličan formularu za dohvat, samo što je zbog potreba prikaza većih dimenzija i na nju je implementiran algoritam vidljiv na slici 8.7.8.1[48] Dijagrama toka upravljanja arhivom.

Budući da se obračun temelji na putnim nalozima, prvi korak je učitavanje putnih naloga iz baze podataka. Pokretanjem samog formulara pokreće se procedura koja učitava sve zapise putnih naloga u tablicu Dataset-a, nakon čega se ta tablica prikaže u Datagridview-u. Za potrebe filtriranja na vrhu formulara je Textbox u kojeg se upisuje uvjet za filter. Upisani sadržaj ugrađuje se u Select upit s kojim se dohvaćaju podaci u tablicu Dataset-a, nakon čega se tablica prikazuje ponovo u Datagridview-u. Tim postupkom je filtriranje uspješno implementirano i prekida se klikom na red u Datagridview-u, pri čemu se omogućuje odabir dokumenta preko klika na red u tablici.

Radnje klika i promjene odabira pokreću procedure koje pozivaju dvije funkcije. Prva funkcija upisuje vrijednosti iz tablice u Label-e zbog preglednosti i čitljivosti samog odabira. Sljedeća funkcija ima implementirane sve korake do potvrde odabira. U njoj su prvo definirana dva upita. Prvi dohvaća obračun koji ima identifikator iz odabranog reda, a drugi dohvaća ostale troškove koji se vežu uz obračun i isto imaju taj identifikator. Pomoću prvog upita se u tablicu Dataseta učita obračun nakon čega se provjerava broj zapisa u njemu. To se provjerava pomoću If uvjeta gdje se kao uvjet postavlja da je broj redaka jednak jedan. Ukoliko uvjet nije zadovoljen i ne postoji niti jedan zapis, za odabrani putni nalog nije napravljen obračun što znači da se pokreće dio koda koji preko svojstva Hidden sakriva Datagridview s ostalim troškovima i preko svojstva Enabled onemogućuje pokretanje gumba koji poziva formular za ispis. Ukoliko je uvjet zadovoljen omogućuje se gumb za poziv ispisa i Datagridview za prikaz ostalih troškova pri čemu se preko drugog upita puni tablica Dataset-a iz koje on uzima podatke koje prikazuje. U tom trenutku navigacijom po tablici korisnik može vidjeti podatke iz putnog naloga i obračuna

[48] Slika 8.7.8.1 Dijagram toka arhive na primjeru obračuna troškova (Autor: Marko Smetko)

ukoliko je napravljen. Omogućen mu je prikaz svih stavaka oba dokumenta zajedno s mogućnošću ispisa. Odabir se potvrđuje klikom na gumb koji u svojoj proceduri postavlja identifikator u Stack formular. Nakon toga, poziva formular za prikaz obračuna koji čita ranije postavljen identifikator i na temelju njega prikazuje obračun koji je odabran iz arhive.

Slika 8.7.8.1 Dijagram toka arhive na primjeru obračuna troškova

8.8 Analiza ključnih dijelova aplikacije

Jednom kad su postavljeni svi elementi i napisan sav kod, treba još jednom provjeriti koliko je aplikacija dobro napravljena za potrebe korisnika, koliko je sigurna i da li obuhvaća sve segmente posla koje treba obavljati. Analizom se prolazi kroz svaki dio aplikacije i ističu se eventualni propusti, moguće buduće nadogradnje i samo iskustvo s aplikacijom.

Kod pokretanja aplikacije treba se proći prijava korisnika koja je i prvi dio koji se analizira. Prijava je jednostavno zamišljena kao ulaz u program i dodjela prava preko korisničkog imena i lozinke. Pozitivne strane su što je sama prijava za korisnika jednostavna i zamišljena je na način kakav je viđen na Operativnim sustavu Windows, Google-u, Facebook-u i brojnim drugim aplikacijama. Kao alternativa moglo se implementirati neko drugo rješenje u vidu prijave preko 3D barkoda,otiska prsta ili nekog drugog biometrijskog obilježja, ali takva bi prijava zahtijevala dodatno programiranje i dodatnu opremu kao što su barkod čitači i biometrijski skeneri. Nedostatak ovog načina prijave je taj što su korisničko ime i lozinka zapisani u Access bazu iz koje ih je vrlo lako iščitati, pa je sigurnost ulaska u program kompromitirana. Jedno rješenje bi bilo provjera lozinke preko hash vrijednosti i to rješenje bi se trebalo implementirati u nekoj budućoj inačici aplikacije. Hash funkcija ili hash algoritam je funkcija za sažimanje i identificiranje podataka [Domjan, Juren i Mlikota,2006,str. 3].

Nakon prijave, korisnik dolazi do mini i maxi moda s kojima treba prolaziti kroz aplikaciju. Svaki prozor u aplikaciji može se otvoriti samo jednom za što postoji mehanizam koji sprečava da se neki prozor dvaput otvori. Elementi za otvaranje formulara su posloženi logički tako da prosječnom korisniku ne bi trebao biti problem u kratkom vremenu naći to što traži. Izbor boja je konzervativan jer koristi standardnu Windows shemu boja zbog toga da ne odskače previše od stila samog operativnog sustava. U nekoj budućoj inačici može se dodati mogućnost da korisnik sam bira poleđinu i shemu boja. Upis putnog naloga dizajniran je za jednostavan i brz upis, te ne zahtjeva nikakva posebna računalna znanja.

Prilikom prelaska iz modula u modul provjerava se svaki zapis što smanjuje mogućnost pogreške. Aktiviranje modula i upis redoslijedom pojednostavljuju korisniku upis podataka. Dodatna pomoć koja bi se mogla u budućoj inačici ugraditi je mogućnost dodavanja predloška za putni nalog i mogućnost stvaranja novih putnih naloga iz tih predložaka.

Sljedeći važni upis je obračun. Kod njega, kao i kod upisa putnog naloga, poslije svakog upisa omogući se sljedeći upis čime se poštuje redoslijed i time olakšava upis. Postoji mogućnost i ispravka obračuna ukoliko je krivo obračunat. Kontrole koje korisnik treba se prikazuju po

potrebi, a one koje ne treba se sakrivaju i zbog toga se korisnik ne može napraviti neku krivu radnju. Kao buduću nadogradnju mogao bi se ugraditi barkod na putni nalog koji bi služio za učitavanje stavaka za obračun umjesto da se broj putnog naloga ručno upisuje.

Arhive za ispis napravljene su kao tablični prikaz podataka za koji se u realnom vremenu prikazuju sve stavke dokumenta koji želimo ispisati ispod. Stavke dokumenta podijeljene su na zasebne sekcije kako bi korisniku bilo lakše pročitati podatak koji traži. U nekoj budućoj inačici kod arhive bi se mogao dodati dokument ispisan kao kopija. Taj bi dokument imao vodeni žig preko cijele stranice da se ne zamijeni s originalnim dokumentom, ali bi po svim ostalim stavkama bio identičan originalu. Takve dokumente ponekad treba imati za uvid dok se želi izbjeći gubitak originala ili zloupotreba dokumenta.

9. Testiranje aplikacije

Prije nego što se aplikacija pusti u rad, mora se testirati njezina ispravnost. Neispravne aplikacije, kao i aplikacije sklone incidentima mogu imati negativan učinak na poslovanje poduzeća koje tu aplikaciju koristi, isto kao i na poduzeće koje je tu aplikaciju razvilo. U današnjem modernom društvu gdje se informacije brzo šire, lako je doći do komentara i mišljenja drugih korisnika o nekom proizvodu, pa je u interesu poduzeća koje razvija neku aplikaciju da ne skupi negativan publicitet. Kako bi se izbjegao taj problem aplikacija prolazi kroz dvije faze testiranja.

U prvoj fazi, osoba koja je razvila aplikaciju za svaki dio posla koji ta aplikacija obavlja, zamišlja sve moguće kombinacije interakcija s aplikacijom kako bi se pronašao eventualni propust u aplikaciji. Kod upisa novih dokumenta prvo je proveden upis i promjena na minimalno pet dokumenta kako bi se isprobala funkcionalnost kod normalnog načina upisa. Kad normalni upis prolazi, daljnje testiranje zahtijeva namjerno izazivanje incidenta. Incidenti su se kod testiranja aplikacije izazivali na sljedeće načine: pokušaj upisa krivim redoslijedom, pokušaj upisa slova u elemente namijenjene brojčanim vrijednostima i obrnuto, ostavljanje neispunjenih stavaka, pokušaj promjene dohvaćenih stavaka koje bi trebale biti fiksne, pokušaj potvrde odabira bez samog provođenja odabira, višestruki pokušaj otvaranja formulara za dohvat, pokušaj ručnog brisanja redaka iz tablica, pokušaj upisa negativnih brojeva i pokušaj upisa nelogičnih datuma. Ako je aplikacija sve pokušaje prošla bez incidenta, može se smatrati spremnom za testiranje korisnika. Navigacija kroz aplikaciju je isprobana na način da se otvore svi mogući prozori i onda se pokušavaju otvoriti još jednom. Ukoliko je ponovno otvaranje zaustavljeno, algoritam za otvaranje prozora je dobro implementiran i time se taj dio aplikacije može smatrati ispravnim. Postavke kataloških vrijednosti testiraju se na isti način kao i upisi dokumenta jer se sastoje od većine istih elemenata i način upisa je sličan. Kod njih je važno da su elementi za njihov upis usklađeni s ograničenjima definiranim u bazi. Ako je nekoj stavki u aplikaciji dopuštena veća veličina od one koju predstavlja u bazi, nastat će incident. Isto tako mora se voditi briga o tome da obavezna polja u bazi ne smiju biti prazne vrijednosti u aplikaciji.

Kad aplikacija uspješno obavlja sve poslove za koje je namijenjena i kad ne izazove niti jedan incident koji se htio namjerno izazvati, spremna je za drugu fazu testiranja. U drugoj fazi testiranja aplikaciju su testirali korisnici koji nisu sudjelovali u njezinom razvoju i prvi put su se susreli s njom. U aplikaciju su prijavljeni kao administratori tako da im je dopušten pristup svakom dijelu programa s kojim se neki korisnik može susresti. Za potrebe ovog testiranja uzet je uzorak od 5 ljudi u razdoblju od 20 do 50 godina starosti. Kako bi se usput vidjelo koliko je

aplikacija korisniku dobro prilagođena, ljudima koji ju testiraju nije bio objašnjen niti jedan postupak i sve što rade u njoj su morali otkriti sami. Takvim pristupom maksimalno se izaziva mogućnost nastanka incidenta te se testira ergonomija same aplikacije. Promatranjem interakcije testnih korisnika s aplikacijom nisu primijećeni incidenti i aplikacija je sve poslove obavila kao što je zamišljeno. Budući da su oba testiranja aplikacije obavljena bez incidenta, aplikacija se može smatrati dovoljno dobrom za plasiranje većem broju korisnika.

10. Pogled s korisničke strane

Da bi se dobila realna slika o iskustvu s aplikacijom, potrebno ju je gledati onako kako bi ju vidio potencijalni korisnik. Promatranjem, zapisivanjem komentara i usmenom interakcijom s testnim korisnicima koji prolaze kroz sve procese upisa, ispisa, arhiviranja i postavki, dobiva se novi pogled na aplikaciju koji se s razvojnog stajališta vjerojatno ne bi uočio.

Nakon što je korisnik prijavljen u aplikaciju, prikazan mu je glavni formular. Kod oba korisnička moda rada, svaki element za otvaranje novog prozora je logički smješten. Prvi su elementi s kojima se poziva stvaranje novih dokumenata, zatim su arhive, postavke i na kraju promjena moda. Nitko od ispitanih korisnika nije imao probleme s pronalaskom djela aplikacije kojeg je tražio.

Nakon same navigacije kroz aplikaciju korisnici se susreću s upisom samog putnog naloga. Na prvi pogled vide se mjesta za upis stavaka koja su obojana žuto. Pritiskom bilo koje tipke pokreće se formular za dohvat koji je jednostavan i nema zbunjujući efekt na testne korisnike. Upisom jedne stavke aktivira se sljedeći modul i omogućuje se sljedeća stavka za upis. Budući da su svi upisi imenovani i aktivirani logičkim redoslijedom, onda prosječnom korisniku nije teško snaći se kod upisa.

Kad je upis izvršen aplikacija prikazuje ispis putnog naloga. Putni nalog je čitljiv i sve stavke su posložene u sekcije po istom principu kako su bile upisivane. U takvom se rasporedu podataka, osoba koja je putni nalog generirala, može lako snaći. Ispis izvještaja putovanja je relativno jednostavan i od strane korisnika za taj dio nisu bili postavljani nikakvi upiti. Sljedeći dokument koji se upisuje je obračun. Upisom broja naloga aplikacija sama navodi korisnike prema sljedećim stavkama za upis. Način upisa stavaka poznat je iz upisa putnog naloga i nove stavke su svi korisnici mogli upisati s lakoćom. Mogućnost promjene i brisanja ostalih troškova nije vidljiva iz sučelja i ta opcija se korisnicima trebala objasniti.

Kad kod neke buduće verzije bude omogućena pomoć u kojoj će se naći objašnjenje za korake brisanja i promjene ostalih troškova. Postavke aplikacije su jednostavne i nemaju previše mogućnosti koje bi zbunjivale korisnike. Testni korisnici su bez problema savladali manipulaciju kataloških vrijednosti i postavljanje osnovnih postavki poduzeća.

Zadnja stvar koja se testira je arhiva. Arhive putnih naloga i obračuna napravljene su prema istom konceptu. Zbog kompromitirane preglednosti, stavke izbora iz tablice postaju vidljive u naznačenim mjestima ispod. Svaka stavka je imenovana i njezina vrijednost je naznačena

crveno. Iako su određeni testni korisnici rekli da izgled arhive na prvi pogled izgleda nagurano, mogli bi se naviknuti na taj dizajn jer su im sve stavke koje traže prikazane i nisu potrebni nikakvi dodatni napori za pristup željenim podacima.

Iz bilješki iskustava testnih korisnika donesen je zaključak da je aplikacija jednostavno i funkcionalno koncipirana, te da ju može koristiti prosječan korisnik s prosječnim znanjem o računalima.

11. Moguća buduća poboljšanja i nadogradnje

U stvarnom svijetu rijetko koja aplikacija plasirana na tržište s vremenom ostaje onakva kakvom je prvotno zamišljena. Zbog promjena na tržištu, korisničkih navika, platforma i samih poslova koje neka aplikacija treba obavljati, često ju je potrebno nadograđivati. Neke nadogradnje se rade kako bi se uklonili nedostaci ili sigurnosni propusti na samim aplikacijama, dok se drugi implementiraju kako bi se povećala funkcionalnost aplikacije ili se učinila poželjnijom na način da joj se osvježi izgled ili unaprijedi ergonomija korištenja.

Iz analize aplikacije i korisničkih iskustava vidi se da bi i aplikaciji za putne naloge određene nadogradnje povećale vrijednost. Ako je neka aplikacija sigurnija, pouzdanija i ima funkcije koje povećavaju produktivnost tada je implementacija nadogradnji opravdana. Kod razvijene aplikacije prvotno je gledano na ono što je glavni posao aplikacije, dok su implementiranje pomoći, sigurnosti i nekih sporednih funkcionalnih komponenti stavljene u drugi plan. Implementiranje nadogradnji nije uvijek jednostavno. Često se zbog nadogradnje aplikacije treba promijeniti i struktura baze podataka. To može biti riskantan potez jer postoji mogućnost gubitka podataka. Jedan način postavljanja nadogradnji je taj da se ručno zamijeni aplikacija i ručno restrukturira baza. Taj način se koristi kad nema mnogo aplikacija u opticaju.

Kada bi postojalo mnogo aplikacija u opticaju, tada bi implementiranje nadogradnje ručnom metodom zahtijevalo mnogo ljudskog rada i morao bi se postaviti red čekanja koji bi usporio cijeli proces. Zbog toga kompanije kao što su Adobe i Microsoft čiji se programski paketi prodaju u velikim količinama imaju zasebne aplikacije ili module koji služe samo za nadogradnje. Takav modul ili aplikacija bi zahtijevala dodatne napore od strane programera. Isplativost takvog poduhvata bi se trebala uzeti u obzir kada bi postojala dovoljna količina aplikacija u opticaju pri čemu bi troškovi i vrijeme ručne nadogradnje nadilazili troškove i napore razvoja dodatne aplikacije za nadogradnju. Logičan potez kod ove aplikacije je pustiti ju na tržište i vidjeti koliko će se proširi i u početku bi se ručno nadograđivala aplikacija korisnika. Ukoliko aplikacija bude proširena na veći broj korisnika, u obzir će se uzeti plan o razvoju aplikacije čija bi svrha bila samo da implementira nadogradnje.

11.1 Instalacija i protupiratska zaštita

Uobičajena je praksa na tržištu da se aplikacije dostavljaju korisniku kao paket koji se pokrene i instalira na računalo umjesto da se dijelovi aplikacije kopiraju na njima pripadajuća mjesta i namještaju ručno. Za potrebe instalacije potrebno je imati jednostavnu aplikaciju koja bi postavila glavnu aplikaciju. Ona je zamišljena da korak po korak vodi korisnika prema tome gdje

želi da aplikacija bude smještena, koje parametre želi i da potvrdi da je on vlasnik aplikacije. Smještaj aplikacije sastoji se od jednostavnog mehanizma koji postavlja datoteke aplikacije na mjesto koje je korisnik definirao da želi i da postavlja bazu podataka na mjesto koje je definirano u samoj aplikaciji. Tijekom instalacije korisnik mora dokazati da je on vlasnik aplikacije na način da upiše slijed slova i brojeva fiksne dužine koji je dobio kada je aplikaciju kupio. Ključ se generira po nekom proizvoljnom algoritmu pri čemu se prava vrijednost daje korisniku tokom kupnje, a hash vrijednost ključa se pohranjuje na aktivacijskom serveru u bazu podataka. Hash vrijednosti se koriste jer ako se slučajno dogodi da se dođe do baze na aktivacijskom serveru, ne može se dobiti ključ jer za hash funkciju ne postoji inverzna funkcija. Teoretski bi se iz hash funkcije moglo doći do ključa metodom pokušaja, ali za takav poduhvat je potrebna enormna količina vremena kad se uzme u obzir veličina ključa i broj mogućih kombinacija znakova u njemu. Kada bi se ključ upisao kod instalacije njegova bi vrijednost bila pretvorena u hash vrijednost i tada bi se na aktivacijskom serveru tražilo da li postoji podudarnost. Ukoliko ne postoji podudarnost, ključ je nevaljan ili je već iskorišten i instalacija se ne bi mogla provesti. Ukoliko postoji podudarnost i ključ nije iskorišten instalacija se nastavlja. Kada su sve datoteke na mjestu tada se uz aplikaciju u jednu datoteku postavlja vrijednost dobivena iz hash funkcije spoja ključa i serijskog broja matične ploče. Obje vrijednosti su jedinstvene za računalo korisnika i ako se iz njih napravi hash vrijednost koja se provjerava uvijek prilikom pokretanja aplikacije, tada se osigurava da aplikacija nije ilegalno prekopirana na neko drugo računalo. Implementacijom ovih koraka smanjuje se mogućnost zlouporabe aplikacije. Napori potrebni za otkrivanje vrijednosti iz hash funkcija višestruko nadilaze vrijednost same aplikacije što potencijalnim piratima uvelike otežava mogućnost zlouporabe.

11.2 Unaprjeđenje sigurnosti prijave u sustav

Analizom aplikacije uočeno je da bi se pronalaskom baze i njezinim otvaranjem mogli iz nje izvući svi podaci svih korisnika potrebni za prijavu. Kada bi se onemogućilo da se podaci za prijavu vide u bazi, uvelike bi se smanjila mogućnost neovlaštenog ulaska u aplikaciju. Podaci bi se štitili uz pomoć hash funkcije. Lozinka bi umjesto zapisa u izvornom obliku bila zapisana kao hash vrijednost. Aplikacija bi tokom prijave upisanu lozinku pretvarala u hash vrijednost i uspoređivala ju s hash vrijednošću u tablici. S korisničke strane se način prijave ne bi mijenjao, što znači da novi korak u sigurnosti ne utječe na korisničko iskustvo s aplikacijom. Kad se gleda od strane nekoga tko želi ukrasti lozinku i zloupotrijebiti aplikaciju, dolazak do nje je uvelike otežan. Kada bi se ukrala hash vrijednost lozinke, ne bi se mogla iskoristiti za prijavu i ne može se iz nje dobiti lozinka u izvornom obliku. Tada ostaje rješenje nasumičnog pokušaja i promašaja

koje ovisno o jačini lozinke određuje koliko je potrebno vremena za njezino probijanje. Čim lozinka ima više znakova,slova ili brojki u kombinaciji, manje su šanse da ju se otkrije.

11.3 Predlošci putnog naloga

Mnoga poduzeća često rade rutinska službena putovanja. To su većinom trgovačka poduzeća poput pekara koja svaki dan distribuiraju svoje proizvode po istim mjestima. U praksi se često događa da putnici svakodnevno koriste ista vozila i prolaze kroz iste rute. Ako su kod takvih putovanja isti vozači, vozila i ruta onda se putni nalozi kod takvih putovanja razlikuju samo po manjim detaljima poput datuma. Izrada takvih naloga može se pojednostaviti i automatizirati na način da nije potrebno upisivati svaki put sve stavke. Rješenje se sastoji u tome da se u postojeći mehanizam za generiranje putnih naloga ugradi modul za manipulaciju predlošcima. Ideja je preuzeta iz aplikacija za Internet i mobilno bankarstvo koje za generiranje rutinskih naloga plaćanja koriste predloške koji su pretpostavljeni ili generirani iz postojećih naloga plaćanja. Po uzoru na taj koncept u budućoj verziji aplikacije, ugradit će se postavke za manipulaciju predlošcima. Za te potrebe u bazi će se dodati nova tablica koja bi trebala biti slično koncipirana kao tablica putnih naloga i ona bi trebala sadržavati katalog predložaka. U postavkama predložaka po istom principu kao i kod ostalih kataloških vrijednosti moći će se dodavati, mijenjati i brisati predlošci. U arhivi putnih naloga isto će postojati opcija koja će iz postojećeg putnog naloga moći napraviti predložak. Spremljeni predlošci pozivali bi se kad bi se radio novi putni nalog. Predložak bi se odabrao pomoću gumba koji bi pozivao formular za dohvat predloška. Tako koncipiran dodatak bi olakšao izradu rutinskih putnih naloga i dodao novu vrijednost aplikaciji i učinio ju kvalitetnijom.

11.4 Centar za rješavanje incidenata

Unatoč testiranjima, nije u potpunosti isključena mogućnost da se incident dogodi. S gledišta programera, mora se znati zbog čega je taj incident nastao i da li je krivnja u aplikaciji ili u okruženju na računalu korisnika. Ukoliko je propust u aplikaciji, potrebno je otkriti kako je nastao i kako ga ukloniti. Ako je uzrok incidenta na računalu korisnika na kojem je instalirana aplikacija, tada se uzrok pokušava zaobići tako da se okružje prilagodi aplikaciji ili ukoliko je to moguće, da se aplikacija prilagodi da bude kompatibilna s okruženjem. Korisnik često nema iste poglede kao programer. Njemu je u interesu da se uzrok incidenta riješi u što kraćem vremenskom roku. Kad se pretpostavi da je korisničko znanje o aplikaciji ograničeno, za očekivati je da u slučaju nastanka incidenta može nastati problem u komunikaciji između programera i korisnika. Kad bi se komunikacija između programera i korisnika svela na

minimum pri čemu bi se slanje informacija o incidentu automatiziralo, to bi pogodovalo jednoj i drugoj strani. Korisnik ne bi morao ulagati dodatne napore za objašnjenje incidenta, dok programer na jednostavan način dobiva rezultate o zadnjem stanju aplikacije prilikom kojeg je incident nastao. Centar za rješavanje incidenata zamišljen je kao dio aplikacije koji bi se aktivirao automatski u situacijama kada se neka radnja ne bi uspjela izvršiti. Za potrebe implementacije ove nadogradnje potrebno je postaviti mehanizme koji bi u zasebnu datoteku bilježili upotrijebljene SQL upite, za slučaj da se uzrok incidenta nalazi u njima. Aplikacija bi u slučaju incidenta pokrenula formular koji bi obavijestio korisnika o slanju izvještaja o incidentu prema programeru. Izvještaj o incidentu sastojao bi se od datoteke sa zapisanim upitima, parametrima programa na Stack formularu i informacijama sustava koje je moguće povući iz operativnog sustava Windows. Izvještaj bi se slao preko FTP[49] protokola na neki zakupljeni server gdje bi ga programer mogao preuzeti. Ovaj korak bio bi prednost pred ostalim aplikacijama za putne naloge jer na njihovim rješenjima za putne naloge nije viđeno da je takav sličan koncept pomoći korisniku ugrađen.

[49] FTP(eng. *File transfer protocol*) je standardni mrežni protokol koji se koristi za premještanje datoteka s jednog hosta na drugi.

12. Zaključak

Zadatak ovog završnog rada bio je izrada aplikacije za vođenje putnih naloga te njima pripadajućim troškovima i tu izradu upotpuniti teoretskim temeljima na kojima je ta aplikacija razvijena.

Prvi korak u rješavanju zadatka bio je istraživanje informacija o putnim nalozima. Iz zakona i pravilnika dobivene su informacije o tome čemu služi putni nalog, koje su obavezne stavke kod putnog naloga i što čini putni nalog pravno važećim. Kao pomoć korišteni su primjeri putnih naloga iz različitih poduzeća kako bi se došlo do dizajna korištenog u ovom radu. U mnogim primjerima putnih naloga korišten je dizajn orijentiran individualnim potrebama pojedinog poduzeća, dok je kod ovog projekta cilj bio napraviti dizajn dokumenta koji je primjenjiv kod većeg broja poduzeća. Taj cilj je postignut jer su se za primjer uzeta postojeća rješenja koja već imaju svoje mjesto na tržištu. Iz analize postojećih aplikacija uočljivo je da postoje neke vidljive prednosti i nedostaci u odnosu na razvijenu aplikaciju iz ovog rada, ali budući da su postojeća rješenja već prihvaćena na tržištu aplikacija i koristi ih veći broj poduzeća, izvlači se zaključak da je niska cijena najbolje rješenje za konkurentsku prednost kada bi aplikacija bila na tržištu.

Analizirajući dokumente i aplikacije na tržištu dobila se ideja o tome u kojem će se smjeru razvijati aplikaciju. Na temelju tih znanja definirani su entiteti, njihovi atributi i veze između njih, nakon čega je napravljen ERA model. On je poslužio kao plan za izradu funkcionalne baze podatka u Access-u. Stvarna baza sadrži neke izmjene i dodatke kako bi se pojednostavio upis iz aplikacije i ujedno zadržali svi podaci koje neko poduzeće može imati.

Kad se gleda sa sadašnjeg stajališta gdje je aplikacija u potpunosti funkcionalna bez obzira na razlike između stvarne baze i ERA modela, može se zaključiti da je odluka o implementiranim improvizacijama u bazi bila opravdana. Jednom kad je baza došla u zamišljene okvire, tad se moglo početi sa stvaranjem aplikacije. U planu izrade aplikacije prvo su se napravile bazne stvari kao što su prijava korisnika, navigacija kroz aplikaciju, kataloške vrijednosti i postavke, umjesto da se direktno radnja usredotočila na krajnju funkciju. Zbog te odluke razvoj aplikacije je tekao linearno, pri čemu se misli da nije postojala potreba da se dorađuje jedan dio aplikacije samo da zadovolji trenutne potrebe potrebne za razvoj dijela aplikacije koji se radio prije njega. Po tom principu se za svaki dio aplikacije prvo stvorio formular i na njega su se dodavali svi elementi koji bi korisniku bili potrebni za rad, a kod se implementirao u te elemente uz pomoć otprije isplaniranih dijagrama toka. Budući da su te metode rezultirale u izradi potpuno funkcionalne aplikacije, može se iz toga zaključiti da razvoj aplikacije bio adekvatno isplaniran. Postoji

mogućnost da se na isti način implementiraju buduće nadogradnje jer se polazi od pretpostavke da bi se upotrebom istih metoda mogla nastaviti uspješnost u daljnjem razvoju aplikacije.

Verziju aplikacije koja je imala implementirane sve isplanirane algoritme bilo je potrebno testirati kako bi se uklonili nedostaci. Testiranjem krajnje verzije nisu bili izazvani incidenti i zbog toga je dana testnim korisnicima na dodatno testiranje. Dodatnim testiranjem od strane testnih korisnika pokazalo se da je rad aplikacije stabilan, a i ujedno se promatranjem testnih korisnika i interakcijom s njima dobilo subjektivno mišljenje o samoj aplikaciji. Korisnici su iznijeli većinom pozitivne komentare na rad s aplikacijom, dok su negativni komentari inspirirali neke od ideja opisanih u budućim nadogradnjama aplikacije.

Ako je koncepcija aplikacije zadovoljavajuća uzorku testnih korisnika, može se očekivati da će biti zadovoljavajuća i potencijalnim kupcima na tržištu aplikacija. Uzevši u obzir sve do sad napravljeno, razvijena je ispravna aplikacija s mogućnošću uspjeha na tržištu, pri čemu su postupak izrade i svi njegovi aspekti opisani u teoretskom dijelu i sastoji se od analize dokumenata, analize postojećih aplikacija, opisa razvojnog sučelja, izrade baze, planiranja i izrade svih aspekata aplikacije, testiranja i eventualnih nadogradnja. Može se zaključiti da su svi aspekti ovog rada teoretski i praktično pokriveni te zbog toga se ovaj zadatak može smatrati uspješno obavljenim. Sva prikupljena eksplicitna i tacitna znanja mogu biti upotrijebljena u daljnjem razvoju ove aplikacije, kao i drugih aplikacija ovog tipa koje se rade po sličnom principu.

13. Literatura

1. Del Sole Alessandro, *Visual Basic .NET Unleashed,* 2010.,Str.515,Izdavač:SAMS
2. Domjan, Juren i Mlikota, Predavanje elektroničko poslovanje:Hash funkcije,2006.,str. 3 dostupno od 14.Kolovoza.2015. na ftp.foi.hr/nastava/elektronickoposlovanje/hash
3. Članak 14. stavka 2 Pravilnika o porezu na dohodak,NN,2006., dostupan od 1.veljače.2015.na http://narodne-novine.nn.hr/clanci/sluzbeni/2005_08_95_1884.html
4. MSDN Baza znanja,Microsoft,215.,dostupno od 1. Veljače 2015 na: msdn.microsoft.com/en-us/library/aa903378(v=vs.71).aspx
5. Pravilnik o naknadama putnih i drugih troškova na službenom putovanju, 2009, dostupan od 1.Veljače 2015. na: www.azoo.hr/images/razno/Pravilnik_o_naknadama_putnih_trokova.pdf
6. Zakon o putnim ispravama Hrvatskih državljana,2015. Dostupan od 1.Veljače na http://www.zakon.hr/z/448/Zakon-o-putnim-ispravama-hrvatskih-državljana
7. Zakon o porezu na dohodak, Članak 9,točka 10.,2015 Dostupan od 1. Veljače na http://www.zakon.hr/z/85/Zakon-o-porezu-na-dohodak

14. Popis tablica

15. Popis slika