UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

NENAD PANIĆ

RAZVOJ IN VPELJAVA PROGRAMA ZA IZDELAVO POROČILA O POTNEM NALOGU

DIPLOMSKO DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Marko Bajec

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00540/2011
Datum: 04.02.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **NENAD PANIĆ**

Naslov: **RAZVOJ IN VPELJAVA PROGRAMA ZA IZDELAVO POROČILA O POTNEM NALOGU**

DEVELOPMENT AND IMPLEMENTATION OF AN APPLICATION FOR BUSINESS TRIP REPORTS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Razvoj poslovnih aplikacij je zahteven proces, ki zajema veliko aktivnosti poleg samega programiranja. Na enostavnejšem praktičnem primeru prikažite uporabo metodologij razvoja informacijskih rešitev. Skladno z izbrano metodologijo izdelajte vse postopke, ki jih metodologija zahteva; izbiro metodologije utemeljite. Na koncu podajte in komentirajte pridobljene izkušnje.

Mentor:

prof. dr. Marko Bajec

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Zahvala

Najprej bi se rad zahvalil doc. dr. Marku Bajcu za mentorstvo, informacije, ter čas, ki mi ga je posvetil pri izdelavi diplomskega dela. Prav tako se zahvaljujem sošolcem, prijateljem in družini, ki so mi pomagali in me spodbujali pri pisanju. Na koncu pa velja še posebna zahvala staršem, ki so me v času študija podpirali in mi ga tudi omogočili.

KAZALO

KAZALO SLIK

POVZETEK

Diplomska naloga obsega izdelavo in vpeljavo programa za izdelavo poročila o potnem nalogu. Razlogi za izdelavo te aplikacije so predvsem nepreglednost nad potnimi stroški in pomanjkanje raznih podatkov v zvezi s potnimi nalogi ter krajši čas izdelave poročila in preprostost uporabe programa.

V uvodnem poglavju sem se lotil opisa problemske domene, katero sem reševal v nadaljevanju diplomskega dela. S programom se pridobi večji nadzor (pregled) nad potnimi stroški in sortiranjem le teh, glede na različne dejavnike. Poleg tega je v uvodnem delu predstavljen tudi namen aplikacije, znotraj tega pa so razloženi cilji, ter dodatne funkcije, ki jih obsega nov program.

V nadaljevanju sem predstavil metodologijo IE (informacijski inženiring), saj sem med delom uporabljal smernice, ki jih ta metodologija ponuja.

V kasnejših poglavjih navajam razlage, kako sem pregledal obstoječi način dela ter ga preučil in s tem pridobil okvirno sliko problema, ter postopoma opisujem posamezne faze razvoja programa. Po posvetovanju z vodstvom združbe sem začel načrtovati nov program in nove funkcije, ki so bile potrebne za boljša in preglednejša poročila potnih nalogov. Za izdelavo programa sem uporabil Microsoft Visual Studio 2008 ter SQL Server 2005, za končno poročilo pa Crystal Reports. V zadnjih poglavjih sem opisal testiranje programa in namestitev.

Ključne besede: izdelava poročil o potnem nalogu, informacijski inženiring, faze razvoja aplikacije

ABSTRACT

This thesis involves the development and introduction of a program to produce the travel order reports. The reasons for making this application are mainly in lack of transparency over travel costs and various information on the travel order. Also the new program requires much less time and effort to create the travel order.

In the introductory chapter I started with the description of the problem domain, which I have been working with through this thesis. With this program I acquired greater control (inspection) of travel costs and sorting those, depending on various factors. Moreover, in the introductory section I presented the purpose of the application and within this I explained the objectives and additional features covered by the new program.

In the following chapters I presented the methodology IE (information engineering) and guidelines that this methodology offers.

In the subsequent chapters I described how I have reviewed existing methods of work and examine it in order to obtain an indicative picture of the problem, and gradually I describe the different phases of program development. After consulting with the leadership of groups I began to plan a new program and new features that were necessary for better and more transparent report of the journey. To make this program I used Microsoft Visual Studio 2008 and SQL Server 2005 and for the creation final report I used Crystal Reports. In the last chapter I described the testing and installing the program.

Keywords: producing travel order reports, information engineering, stages of program development

1. UVOD

1.1. Opredelitev problema

V družinskem podjetju se ukvarjamo z mednarodnim transportom. Ta dejavnost zahteva dokaj natančna poročila o potnih stroških, saj so ti zelo pomembni za računovodstvo. Po vsakem opravljenem prevozu je potrebno urediti in sešteti posamezne potne stroške, ter jih pripraviti (sortirati) za knjiženje. Poleg tega je treba voditi evidenco porabe denarja v različnih državah in spremljati razlike porabe pri vsakem prevozu.

Kar zadeva same stroške, je seveda potrebno tudi njihovo sortiranje po raznih dejavnikih (vrsta stroška, kraj, znesek …). Pri plačevanju le teh je potrebno voditi tudi evidenco plačila z bančno kartico ali z gotovino. Ker prevoz poteka po več državah, se pojavlja tudi problem preračunavanja valut po tečaju na tisti dan, ko je strošek nastal. Prav tako nas zanimajo tudi statistični podatki prevozov, kot so na primer: trajanje prevoza, povprečna poraba goriva, prevoženi kilometri, kupljeno gorivo in povprečna poraba denarja.

Le nekateri izmed teh problemov so se do sedaj reševali z uporabo zvezkov v Microsoft Excelu. V razpredelnice so se vnašali stroški in s tem smo dobili le veliko nepregledno tabelo, iz katere smo odčitali podatke, ki so nas zanimali, vendar to še zdaleč ni bilo dovolj za končno poročilo potnega naloga.

Zgoraj omenjeni statistični podatki so se računali ročno, kar je dopuščalo človeško napako in to je vodilo v nepopolna poročila, ki so kasneje povzročala težave pri knjiženju. Te napake je bilo potrebno odpraviti, problem pa je bil najti napačen seštevek v celotnem poročilu. Problem je bil tudi pri iskanju starejših poročil, saj je vsak prevoz predstavljal en Excelov zvezek, shranjen na trdem disku; torej, nikjer ni bilo podatkovne baze, ki bi nam omogočila hitro iskanje starejših poročil in bi združila vse podatke na enem mestu. Če tu omenim še čas in napor, ki je bil porabljen za izdelavo (starega) poročila, pridemo do vzroka, zakaj sem se odločil narediti aplikacijo, ki nam bo omogočila enostavno in hitro izdelavo poročila.

1.2. Cilji

Glavni cilj projekta je narediti uporabniku prijazno aplikacijo, ki bo omogočala rešitev problemov, navedenih v prejšnji točki. Če na kratko ponovim, mora program omogočati naslednje:

- združevati vse podatke o prevozih na enem mestu
- vnos stroškov
- sortiranje stroškov po raznih kriterijih
- seštevek raznih stroškov kot podlaga za knjižbo
- preračunavanje valut
- posredovanje statističnih podatkov prevozov
- hitro iskanje in urejanje starejših poročil
- izdelavo kvalitetnega in preglednega poročila o potnem nalogu.

Ker so pri transportnem podjetju najpomembnejša delovna sredstva tovorno in priklopno vozilo, sta seveda vzdrževanje in amortizacija le teh zelo pomembna.
To nas pripelje do dodatnega cilja tega projekta, saj je smiselno vključiti podatke o delovnih sredstvih in obveščanje o raznih dogodkih. Dobro je tudi vedeti, kdaj naj pričakujemo razne dogodke v zvezi z vzdrževanjem, saj si s takimi podatki lahko pomagamo načrtovati kratkoročne obveznosti vzdrževanja delovnega sredstva. Če na te podatke nismo pozorni, se lahko kaj hitro zgodi, da spregledamo kakšen dogodek, še posebej, če moramo vzdrževati več vozil, in to nas pripelje do velikih, nepričakovanih stroškov. Če na kratko povzamem, pridem do dodatne funkcije programa, ta pa nam mora omogočati dostop do naslednjih podatkov:

- čas do izteka registracije
- čas do naslednjega rednega servisa tovornega vozila
- kdaj so bili zamenjani večji rezervni deli
- kdaj so bile zamenjane pnevmatike.

Kot končni cilj projekta naj še omenim, da mora končni program dopuščati možnost dodatnega razvoja in širitev aplikacije, saj je področje transporta zelo obsežno in so dodatne funkcionalnosti programa zelo dobrodošle. Naj tu omenim samo povpraševanje po prevozih in ponudbo teh preko svetovnega spleta.

2. METODOLOGIJA IE

2.1. Metodologija na splošno

Na splošno metodologija pomeni skupek metod, postopkov in standardov, ki sestavljajo zaključeno celoto pri izvajanju inženirskih pristopov k razvoju produkta. Glede samih definicij jih obstaja več.

- Metodologija je priporočena zbirka filozofij, faz, postopkov, pravil, tehnik, orodij, dokumentacije, upravljanja in izobraževanja za razvijalce IS (Maddison).

- Metodologija razvoja IS je priporočen način razvoja IS, ki temelji na filozofiji in množici principov (Avison, 2003).

- Metodologija je množica dogovorov (konvencij), s katerimi se (projektna) skupina/organizacija strinja (Cockburn, 2002).

- Metodologija je vse, kar redno delamo, da bi dosegli končni rezultat – delujoča PO pri končnem uporabniku (Cockburn, 2002).

Metodologija razvoja sistema pomeni postopen način razvoja informacijskega sistema, ki vključuje uporabo različnih tehnik in orodij, celovit v smislu korakov življenjskega cikla razvoja.

Izbor metodologije je eden prvih in zelo pomembnih korakov pri načrtovanju in izdelavi aplikacij. Z metodologijo razbijemo projekt na majhne, zaokrožene aktivnosti, katere določajo natančno zaporedje ter aktivnosti. Za predstavitev dovolj natančnega in strukturiranega opisa sistema uporabljajo diagramske in druge tehnike modeliranja, ter s tem podajo razumljivo sliko celotnega sistema tako razvijalcem, kakor tudi končnim uporabnikom.

2.2. Informacijski inženiring (IE)

Informacijski inženiring ali metodologija informacijskega inženiringa sodi med strukturne metodologije razvoja načrtovanja in informacijskih sistemov. To metodologijo sta prvič predstavila leta 1981 James Martin in Clive Finkelstein. Sčasoma sta se nekoliko oddaljila v pristopu, v praksi pa je prevladal Martinov pristop.

Kaj je informacijski inženiring

Metodologija IE je natančen pristop k planiranju, analizi, načrtovanju in implementaciji programskih rešitev v organizaciji. Organizaciji ali podjetju omogoča maksimalno izrabo svojih virov (kapitala, kadrov in IS) pri doseganju zastavljenih ciljev. Informacijski inženiring je opredeljen kot integrirana in razvojna množica opravil in tehnik, ki izboljšajo poslovne

komunikacije znotraj podjetja in s tem omogočajo usmerjanje ljudi, procedur in sistemov k uresničevanju njihovega poslanstva.

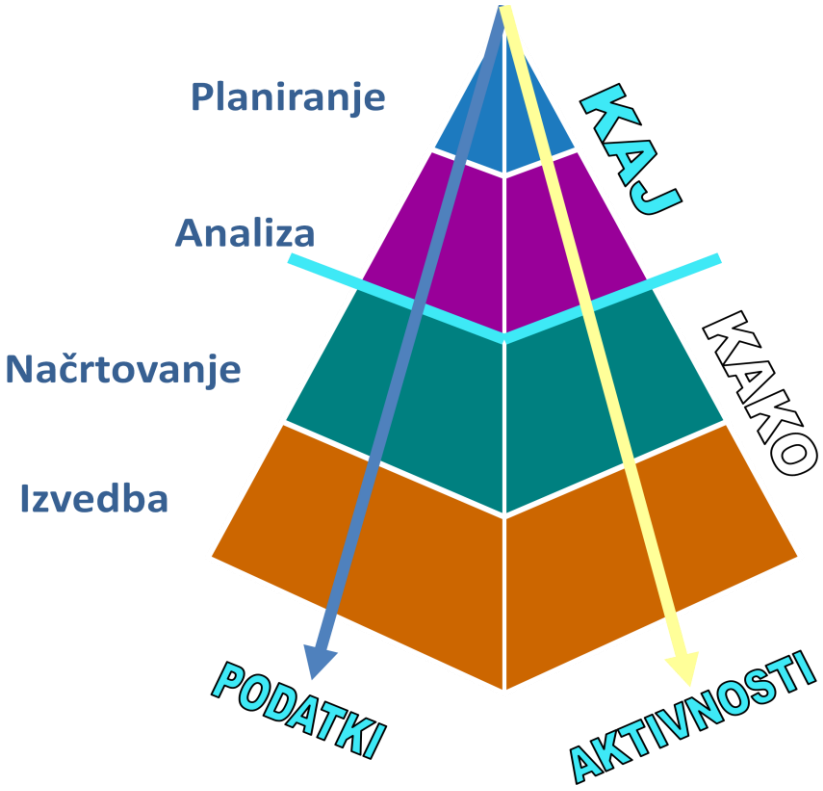
IE je večnamenski in podpira:

- načrtovanje v organizaciji
- prenovo poslovnih procesov
- razvoj aplikacij
- načrtovanje IS
- prenovo sistemov.

2.2.1. Faze razvoja informacijskega sistema po metodologiji informacijskega inženiringa

IE v grobem zajema naslednje štiri faze razvoja:

- **strateško načrtovanje IS (planiranje)**
- **analizo funkcijskih področij**
- **načrtovanje aplikacijskega sistema**
- **izvedbo IS.**



Slika 1: Faze razvoja po metodologiji IE

2.3. Zakaj informacijski inženiring

Metodologijo IE sem izbral zato, ker ta podpira in narekuje razvoj sistema od začetka do konca. Skozi celoten projekt sem se lahko opiral na faze razvoja in v vsakem trenutku sem vedel, kako sem do določene točke prišel in kam me bo ta pripeljala. Poleg tega ima IE jasno in enostavno opredeljene tehnike dela, ki so nepogrešljive v tovrstnih projektih. Enostavno razumljive in natančno definirane tehnike so ključnega pomena pri razvoju informacijskih sistemov.

Nekaj pomembnih tehnik, ki jih uporablja IE:

- **Analiza entitet**

ugotovi vse objekte, o katerih podjetje zbira, hrani in obdeluje podatke. Rezultat analize entitet je klasifikacija entitet v entitetne tipe. Na osnovi analize entitet se izdela entitetni model (model entitet).

- **Funkcionalna analiza in odvisnost (medsebojni vpliv) procesov**

Funkcionalna analiza obravnava poslovne funkcije (glavne poslovne aktivnosti) podjetja in jih razčleni na elementarne poslovne postopke.

- **Analiza logike procesiranja podatkov**

opiše zaporedje akcij, ki jih izvajajo poslovni procesi in prikaže, kateri podatki so uporabljeni pri določeni akciji.

- **Navzkrižno preverjanje podatkov in procesov**

S to tehniko naredimo navzkrižno matrico med podatki in procesi in preverimo, če so zajeti vsi potrebni podatki in procesi ter popolnost le-teh.

- **Analiza toka podatkov in analiza podatkov**

naredi primerjavo med modelom poslovnega področja in IS, ki bo podpiral izvedbo osnovnih funkcij in procesov. Analiza se izvede tako, da se uporabita tehniki toka podatkov in analize podatkov.

3. PLANIRANJE

Projekt sem začel s fazo planiranja. Namen tega je narediti načrt, ki bo omogočal normalen potek projekta in bo na koncu pripeljal do rezultata, ki bo podpiral vse potrebe. Tu sem se najprej seznanil s problemom in ga preučil. Nato sem pregledal dosedanji način reševanja problema in s tem pridobil informacije o tem, katere funkcije bo moral bodoči program podpirati. Na koncu sem še opredelil bodoče faze razvoja in vse skupaj predstavil v časovnem planu razvoja.

3.1. Proučitev problema in poslovnih ter tehnoloških zahtev

Vsak pregled prevoza, oz. potnega naloga se prične s kupom računov na mizi. Te je seveda potrebno podrobno preučiti in na koncu predstaviti v natančnem poročilu. Težave so nastajale na mestih, ko je bilo potrebno iz tega kupa narediti naslednje:

- posvečati posebno pozornost večjim stroškom
- deliti stroške glede na vrsto
- seštevati specifične vsote
- deliti stroške glede na kraj
- deliti stroške glede na datum
- preračunavati valute
- voditi posebno evidenco o natakanju goriva
- vnašati in shranjevati podatke
- deliti stroške glede na način plačila - z bančno kartico ali gotovino
- posebej obravnavati stroške, ki so nastali v Sloveniji
- sestaviti pregledno in natančno poročilo
- iskati starejša poročila
- odgovoriti na vprašanja, povezana s starejšimi nalogi.

Prav tako je potrebno voditi osnovne podatke o prevozu, kot so:

- datum odhoda / prihoda
- relacija prevoza
- stanje kilometrov vozila ob odhodu / prihodu
- stanje rezervoarja ob odhodu / prihodu
- prejeta vsota denarja ob odhodu
- ostanek denarja ob prihodu
- kdo je prevoz opravljal
- s katerim vlečnim in priklopnim vozilom je bil prevoz opravljen
- koliko časa je prevoz trajal
- ali je na prevozu prišlo do nepredvidenih stroškov.

Po pogovoru z vodstvom podjetja o omenjenih težavah je bil cilj narediti program, ki bo omogočal učinkovito in hitro rešitev vseh težav ter bo omogočal tudi hrambo podatkov na enem mestu in dopuščal možnost za nadaljnji razvoj. Projekt sem nadaljeval z določitvijo tehnoloških zahtev in razvojne tehnologije.

Tu bi moral dodati kaj o stroških razvoja, ker pa smo družinsko podjetje in ker program ne bo zahteval nobenih zunanjih virov za delovanje in se bo aplikacija izvajala na enem računalniku, posebnih stroškov razvoja takorekoč ne bo.

Potrebna bo seveda tudi dokumentacija, s katero bodo lahko uporabniki dobro spoznali program in vse funkcije, ki jih bo le ta ponujal.

Tehnološke zahteve

Ker bo program samostojen, so tehnološke zahteve dokaj majhne. Za normalno delovanje bo potrebno imeti:

- osebni ali prenosni računalnik
- tipkovnico, miško
- A4 tiskalnik
- dostop do interneta.

Računalnik bo moral delovati na osnovi Windows in imeti naložen DOT NET FRAMEWORK. Potrebna bo tudi podatkovna baza, ki bo nameščena na lokalnem strežniku, dostop do nje pa bo lahko potekal preko več računalnikov.

Določitev razvojne tehnologije

Program bo v celoti razvit z naslednjimi tehnologijami:

- Microsoft Visual Studio 2008 (kodiranje)
- Sybase Power Designer 10 (načrtovanje podatkovne baze)
- Microsoft SQL Server 2005 (izdelava podatkovne baze)
- Crystal Reports (izdelava poročila)
- Microsoft Office Word (dokumentacija)

Uporabniške zahteve

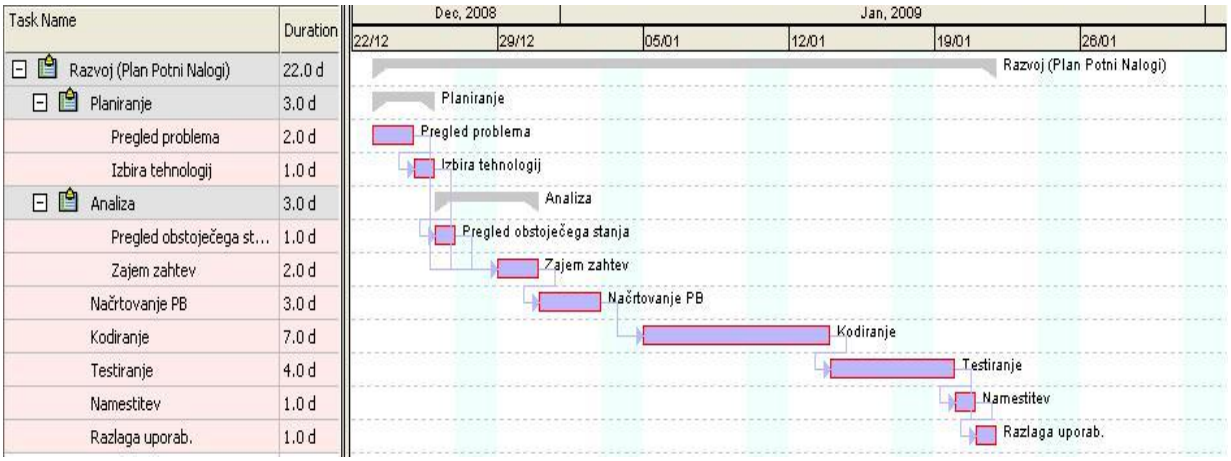
Program mora biti preprost za uporabo in na vsakem koraku naj bo razvidno, kateri koraki so še potrebni do uspešnega zaključka naloga. Podoba programa mora biti nekako standardna za win32 aplikacije in osnovo Windows. Priporočljivo je tudi vključiti pomoč na vsakem koraku, za primer, da uporabnik ni prepričan, če pravilno uporablja funkcije programa.

3.2. Časovni plan razvoja

Po končanem seznanjanju s problemom sem se lotil izdelave časovnega načrta izdelave programa. Pri tem sem si pomagal s programom Project Planner. To je orodje, ki pomaga časovno načrtovati projekte in omogoča izdelavo Ganttovih diagramov. Na podlagi teh vidimo našo kritično pot projekta in vse odvisnosti med opravili.

Za vsa potrebna opravila sem določil čas trajanja, in sicer na podlagi dosedanjega znanja in izkušenj na področju programiranja, ter na podlagi kompleksnosti svojega projekta.

V Ganttovem diagramu spodaj so prikazana vsa opravila ter odvisnosti med njimi, razvidno je, da se eno opravilo ne more začeti, dokler ni končano predhodno opravilo.



Slika 2: Prikaz Ganttovega diagrama projekta

4. ANALIZA

Namen analize je najti odgovor na vprašanje, kaj naj IS podpira, kaj se izvaja v poslovnih funkcijah in katere podatke te rabijo.

Analiza služi kot:

- sredstvo za definicijo zahtev
- osnova za dogovor med naročnikom in izvajalcem
- osnova za kasnejše faze razvoja.

4.1. Opredelitev obstoječega stanja

Dosedanji način dela je potekal preko uporabe zvezkov v Microsoft Excel-u. Stroški so se vnašali v preglednice ter se seštevali glede na vrsto. Ta način dela je bil zelo zamuden in odprt za človeške napake. Večino potrebnih funkcij se je opravljalo ročno. Potrebne specifične vsote so se odčitale iz preglednice ter se seštevale s pomočjo kalkulatorjev.
Vsak nalog je predstavljal svoj zvezek, shranjen na disku, torej ni bilo nobene podatkovne baze, ki bi olajšala iskanje starejših nalogov.

Spodaj so z zeleno barvo označene funkcije, ki jih je stari način dela podpiral. Tiste, ki niso označene, so se izvajale ročno.

- Posvečanje posebne pozornosti večjim stroškom
- Delitev stroškov glede na vrsto
- Delitev stroškov glede na kraj
- Delitev stroškov glede na datum
- Seštevanje specifičnih vsot
- Preračunavanje valut
- Vodenje posebne evidence o gorivu
- Vnašanje in shranjevanje podatkov
- Delitev stroškov glede na način plačila - z bančno kartico, gotovino
- Posebno obravnavanje stroškov, ki so nastali v Sloveniji
- Sestavljanje preglednega in natančnega poročila
- Iskanje starejših poročil
- Najti odgovore na vprašanja, povezana s starejšimi nalogi

4.2. Zajem zahtev

Zajem in specifikacija zahtev je ena pomembnejših aktivnosti pri razvoju programa, oziroma informacijskega sistema. Bistvo zajema je pridobiti informacije o vseh specifikacijah bodočega sistema, ali drugače povedano, potrebno je navesti vse funkcije, ki jih bo program omogočal.

V svojem projektu sem torej moral v novem programu ohraniti vse funkcije prejšnjega načina dela in seveda dodati še veliko novih funkcij. Velik poudarek sem namenil skrajšanje časa izdelave poročila in shranjevanju podatkov v podatkovno bazo, ter shranjevanju dokumentov v elektronski in papirni obliki. Razlog tiči v tem, da je po zakonu potrebno imeti vse dokumente shranjene na računalniku za najmanj dve leti nazaj, v papirni obliki pa za vsa leta nazaj.

Vse zahteve sem razdelil v dve skupini:

funkcionalne- kaj naj bi nov sistem delal

nefunkcionalne- kako dobro naj bi bila neka lastnost zagotovljena (stopnja kakovosti).

4.2.1. Funkcionalne zahteve programa

V tem poglavju navajam zahteve, ki se nanašajo na želeno, oziroma potrebno funkcionalnost aplikacije.

Program bo podpiral naslednje funkcionalnosti:

- **vnos začetnih podatkov prevoza**

  - relacija
  - ime in priimek voznika
  - tovorno in vlečno vozilo
  - datum odhoda
  - stanje števca (km)
  - stanje goriva
  - prejeti denar
  - številka potnega naloga

Ti podatki bodo pomembni za vsak prevoz, saj se bodo na podlagi teh računali statistični podatki, ki jih bom razložil v kasnejših poglavjih.

- **vnos potnih stroškov**

  - datum
  - kraj
  - vrsta stroška
  - znesek
  - valuta
  - način plačila
  - znesek preračunan v evre
  - ali je znesek obračunan (plačan v SLO)

Te podatke je potrebno vnesti v program, saj se na njihovi podlagi izračunavajo zneski za knjižbo. Vsi se odčitajo iz plačanih računov.

- **vnos podatkov o natakanju goriva**

  - datum
  - kraj
  - stanje števca (km)
  - količina goriva
  - znesek
  - valuta
  - način plačila

Ti podatki služijo za informacijo o porabi goriva, saj je to največji strošek pri prevozu in je ta evidenca zelo pomembna. Prav tako so potrebni za računanje statističnih podatkov.

- **vnos prejetih dovolilnic**

  - vrsta (za katero državo je potrebna)
  - številka dovolilnice

Potreba po teh podatkih je pogojena z nadzorom uradnih dokumentov, saj pri vsakem prevozu potrebujemo dovolilnice in vedeti moramo, koliko smo jih že porabili, ter koliko jih imamo še na razpolago.

- **vnos podatkov o končanem prevozu**

  - datum prihoda
  - stanje števca ob prihodu (km)
  - stanje goriva
  - ostanek denarja

Ob vnosu teh podatkov se potni nalog zaključi, to pomeni, da je prevoz končan in da imamo vse potrebno za izdelavo poročila. Prav tako ti podatki služijo tudi za statistiko.

Vsi zgoraj omenjeni vnosi bodo seveda dopuščali možnost urejanja, se pravi, ponovnega dodajanja, brisanja in popravljanja vnosov. Bistvo teh vnosov je, da so ti potrebni za izdelavo končnega poročila po prevozu.

- **izdelava poročila o potnem nalogu**

Izdelava poročila predstavlja ključno funkcijo programa, saj nam bo podal podroben rezultat prevoza na pregleden način. Prikazoval bo vse potrebne podatke o prevozu, stroškovnik prevoza, podatke o natakanju goriva in dovolilnicah. Poleg tega pa bo vse stroške uredil po kriterijih, kot so: vrsta, kraj, način plačila, stroške, pri katerih smo oproščeni plačila davka in bo na koncu prikazal skupni rezultat.

- **računanje statističnih podatkov**

Kot sem že v drugem poglavju omenil, je bil cilj aplikacije prikazovanje statističnih podatkov prevoza. To pomeni, da bomo imeli na voljo vpogled o prevoženih kilometrih, o kupljenem gorivu, porabljenem gorivu, o povprečni porabi in trajanju prevoza, in sicer za vsak prevoz posebej; prav tako pa bomo imeli možnost izračunati te podatke za določeno obdobje, kar pomeni, da bomo lahko izvedeli statistične podatke za točno določen čas, ki nas bo zanimal (n.pr.: zanimajo nas prevoženi kilometri za tovorno vozilo x v času od 01.01.2008 do 10.03.2008).

- **obveščanje o bližajočih se dogodkih**

Program bo obveščal tudi o bližajočih se obveznostih vzdrževanja sredstev. Za vsako tovorno in priklopno vozilo se bodo vnašali podatki o registraciji vozil, o rednih servisih, o menjavah pnevmatik in o zadnji menjavi pnevmatik. S temi informacijami bomo pridobili, da nas bo aplikacija opozarjala na datum ponovne registracije. Prav tako nam bo teoretično sporočala, kdaj bo potrebno menjati pnevmatike in koliko časa imamo še do naslednjega rednega servisa. Ta funkcija programa je zgolj informativna, a zelo pomembna, saj na te vrste stroškov ne smemo pozabiti, ker predstavljajo visoke denarne zneske.

- **vnos valut**

Program bo seveda moral omogočati tudi vnos in preračunavanje med valutami, saj je vse tuje valute potrebno preračunati v evre. Program bo zahteval le vnos potrebnih valut, torej tistih držav, preko katerih je prevoz potekal. Potrebno bo pa tudi ločiti plačila z gotovino in bančno kartico.

- **računanje specifičnih zneskov za knjiženje stroškov**

Kot sem že v prejšnjem poglavju omenil, bo potrebno tudi računati posebne vrste stroškov, kot osnovo za knjiženje. To pomeni, da bomo lahko v programu določili, da nam izračuna določeno vrsto stroška, ki ga potrebujemo za knjižbo (n.pr.: zaradi povračila davkov nas zanima, koliko stroškov, ki niso povezani s špedicijo, je nastalo v Sloveniji in koliko teh je bilo plačanih z bančno kartico).

- **hitri pregled naloga**

Tu bomo imeli takojšen vpogled v obstoječi potni nalog in tukaj se bodo lahko sproti pregledovali in seštevali stroški. Prav tako pa nas bo program opozarjal na napake, ki so povezane s preračunavanjem.

- **predogled in tiskanje naloga**

Program bo seveda omogočal tudi predogled poročila in tiskanje. Uporabnik bo imel možnost izbire med tiskanjem celotnega poročila in tiskanjem posameznih segmentov. Slednje pomeni, da bo lahko na končno poročilo dodajal katerekoli segmente poročila (statistični podatki, stroškovnik, evidenca o gorivu, evidenca dovolilnic).

4.2.2. Nefunkcionalne zahteve programa

Poleg zahtev, ki funkcionalno določajo sistem, je potrebno poskrbeti tudi za tiste faktorje, ki neposredno ne določajo sistema, vendar so zelo pomembni pri njegovem delovanju in je od njih odvisno nemoteno delovanje. To so torej zahteve, ki se nanašajo na tehnične in druge nevsebinske zahteve sistema.

- **Varnost**

  Podatkovna baza je ključnega pomena za delovanja programa, zato je zelo pomembno, da je ta zaščitena pred dostopom nepooblaščenih oseb.

- **Odzivnost**

  Uporabniki zahtevajo, da so aplikacije, ki jih redno uporabljajo, dovolj hitre, oz. odzivne. To je pomembno zgolj ob zagonu programa in ob kreiranju končnih poročil ter pri shranjevanju podatkov.

- **Zanesljivost**

  Zanesljivost programa je zelo pomembna. Ta mora ob vsakem zagonu omogočati vse svoje funkcionalnosti. Pred dejansko vpeljavo programa ga je treba torej dobro testirati in preprečiti vse možne potencialne napake tako glede kodiranja kot tudi s strani uporabnika.

- **Uporabnost**

  Program mora biti uporabniku prijazen in enostaven za uporabo. Omogočati mora, da lahko v vsakem trenutku dobimo odgovor na katerokoli vprašanje, ki se nanaša na opravljene potne naloge. Priporočljivo je, da je izgled programa prijeten in da nima vpadljivih barv, ter da so vsa uporabniška okna narejena na podoben način.

4.3. Funkcije programa

Da bi program izpolnjeval zadane naloge, mora biti sposoben izvrševati sledeče funkcije:

- možnost vnosa in pregledovanje po vseh pomembnih podatkih,
- predogled in izpis dokumentov,
- možnost ponovnega vnašanja in popravljanja podatkov.

Značilnosti za uporabnika

Program lahko uporabljajo uporabniki brez posebnega in natančnega usposabljanja za delo s programom, zato je med uporabo programa potrebna uporaba dokumentacije ali navodil.

4.4. Splošne omejitve

Pri izdelavi programa so upoštevani naslednji pogoji in omejitve:

- potrebno je zagotoviti varno in natančno uporabo programa zaradi zavarovanja vnesenih podatkov
- program je odprt za nadaljnje dograditve in dopolnitve
- za dodajanje novih funkcij programa je potrebno, da uporabnik poda zahtevo za nove funkcije z natančno opredeljeno dokumentacijo.

Predpostavke in odvisnosti

Program deluje na operacijskih sistemih Windows 98, 2000 in XP.

Uporabniški vmesnik

Program deluje v Windows delovnem okolju, kar pomeni, da je grafični vmesnik prilagojen, oz. podoben takemu, kot je delovno okolje. Program za vse glavne funkcije vsebuje kratko pomoč s klikom na gumb »Pomoč«. Natančni izgled in vsebina vseh potrebnih izpisov se lahko dopolnita in spremenita po dogovoru z uporabnikom. O vseh spremembah se lahko dogovorimo, vendar je priporočljiva pisna dokumentacija zahtev.

Komunikacijsko povezovanje

Program ne vsebuje sklopov za dislocirano povezovanje, kar pomeni, da v osnovni verziji ni možna dislocirana uporaba. Program bo deloval znotraj lokalnega omrežja.

Internet

Zaželeno je, da ima uporabnik dostop do interneta zaradi uporabe podatkov s tečajne liste.

4.4.1. Zahteve za namestitev programa

Da bo lahko program tekel na ciljnem računalniku, bo na tem potrebno namestiti ogrodje. NET Farmework (v nadaljevanju ogrodje), saj programsko okolje Visual Studio deluje na tem ogrodju. Poleg tega pa bo potrebno tudi vključiti knjižnice za prikazovanje poročil Crystal Reports-a. Več o tem v poglavju Namestitve programa.

4.4.1.1. Ogrodje Microsoft.NET

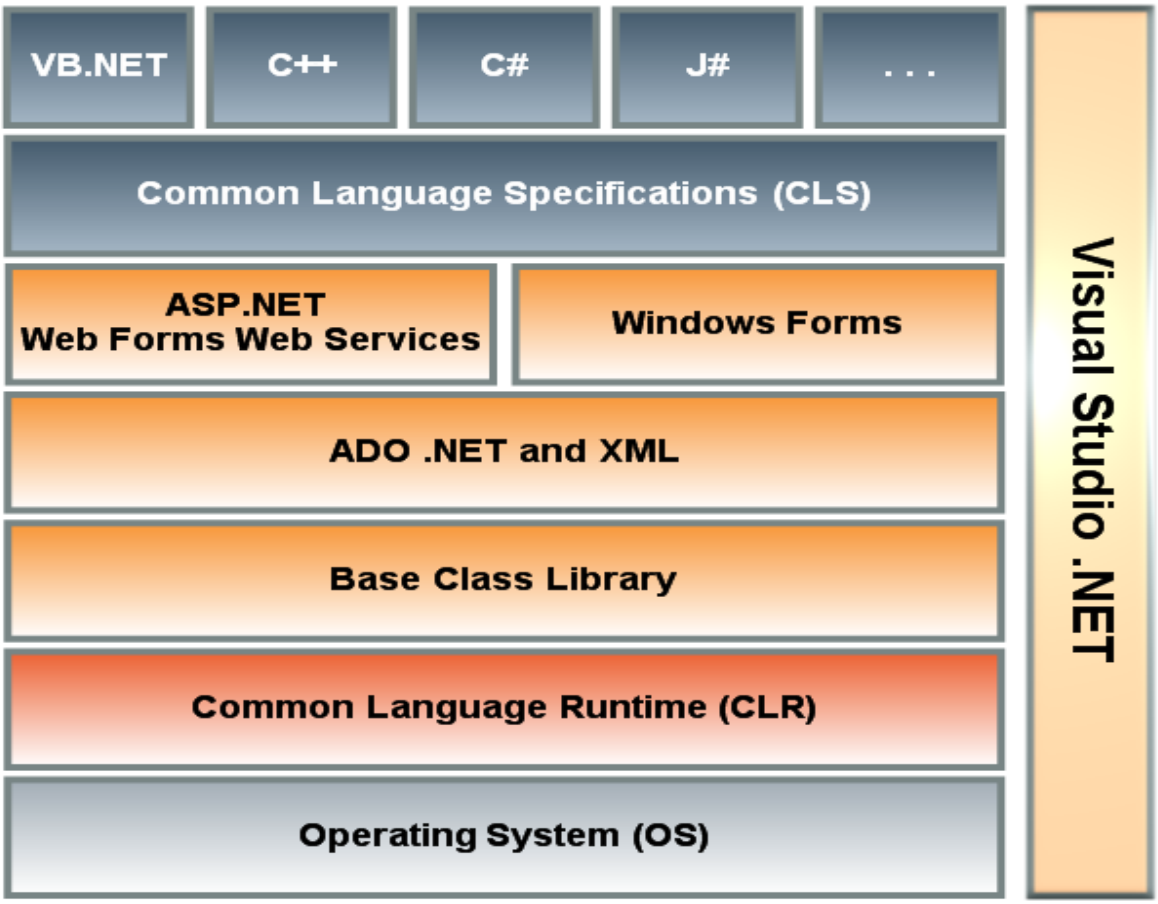
Ogrodje.NET je ogrodje programske opreme na Microsoft Windows operacijskih sistemih. Ta vsebuje obsežno knjižnico programskih rešitev, ki preprečijo pogoste težave pri programiranju, in virtualni stroj, ki nadzira izvršitev programov, napisanih posebej za omenjeno ogrodje. Programske rešitve, ki sestavljajo glavno knjižnico ogrodja (BLC - Base Class Library), poskrbijo za vrsto programskih zahtev na številnih področjih, med katerimi so tudi uporabniški vmesniki, dostop do podatkov, povezljivosti s podatkovnimi bazami, kriptografije, spletne aplikacije, numerični algoritmi in omrežne povezave. Preostali del knjižnice uporabljajo programerji, ki jo kombinirajo s svojimi lastnimi kodami. Na tak način nastajajo aplikacije.

Ogrodje.NET radikalno posega v samo jedro osnove Windows. Določa arhitekturo, ki strogo temelji na konceptih objektne tehnologije in komponentnega razvoja, torej, na temeljih, ki jih je postavil COM+. Razvijalcem ponuja pregledno, konsistentno razvojno ogrodje, katerega se bo lažje naučiti in bo nudil večjo produktivnost od dosedanjega modela programiranja pod Windows.

Osnova je skupno izvajalno okolje za programske jezike (CLR). Kot že ime pove, to okolje nudi nizkonivojske storitve za izvajanje programov. Zelo zanimiv pa je način, kako CLR izvaja programe. CLR vpeljuje vmesni jezik (IL – Intermidiate Language). Izvorna koda programov se po novem ne prevede v strojno kodo, pač pa v vmesno kodo. Pri zagonu aplikacije se vmesni jezik pretvori v strojno kodo.

V ogrodju.NET lahko kličemo metode objektov in komponent brez ozira na programski jezik, v katerem so napisane – in to brez ukvarjanja z namestniki in s skeleti, s posredniki zahtev objektov in podobnim sistemskimi koncepti. Vsi ti še obstajajo, vendar so konsistentno skriti pred razvijalcem. V nekem programskem jeziku (n.pr.: C++) lahko dedujemo od objekta, napisanega v drugem programskem jeziku. Možnosti so praktično neomejene in na prvi pogled zelo obetavne. Ker za vsem stoji vmesni jezik, je on tisti, ki določa, kaj je možno in kaj ne. Z drugimi besedami, vmesni jezik določa najmanjši skupni imenovalec, vendar na ta način v sistem vnaša tudi nekaj omejitev.

Ogrodje.NET je torej temelj za izgradnjo aplikacij nove generacije, ki vsebuje nov celovit programski model, ta pa temelji na standardih in je korak proti integraciji.
Sestavljata ga **skupno izvajalno okolje** (Common Language Runtime - CLR) in **e**.NET (BCL).



Slika 3: Pregled ogrodja .NET

Razlaga pojmov:

- CLS

(Common Language Specification) je podmnožica CTS, ki jo morajo implementirati vsi prevajalniki, ki prevajajo v CLR. V bistvu je to najmanjši skupni imenovalec podatkovnih tipov, ki jih morajo jeziki poznati in podpirati, da je za njih smiselno napisati prevajalnik v CLR. Nepredznačenih celoštevilskih tipov, razen 8-bitnega zloga, v njemu pač ni, ker jih tudi ni v starem VB-ju, ali pa, recimo, v J#, ki uporablja Java sintakso. To ne pomeni, da ni možno uporabljati nepredznačenih in ostalih tipov, ki so implementirani v CTS, vendar ti tipi pač ne spadajo v CLS.

- CTS

(Common Type System) je sistem podatkovnih tipov, ki je namenjen podpori velikega števila objektnih in proceduralnih jezikov. N.pr. F# kot izvedenka Fortran sintakse in Eiffel.NET kot implementacija Eiffel sistema, se oba zanašata na isti CTS, čeprav je en strogo proceduralen, drugi pa strogo objekten jezik. Tukaj najdemo vse primitivne tipe in ogrodje za izgradnjo poljubnih kompleksnih tipov.

- CLR

(Common Language Runtime) je okolje, sestavljeno iz JIT, GC in standardne knjižnice. JIT (Just-In-Time compiler) skrbi za sprotno prevajanje CLR kode v strojno kodo in izvajanje le te. GC (Garbage Collector) skrbi za delo s pomnilnikom, standardne knjižnice pa so tisto, kar da okolju dejansko substanco. CLR je ekvivalent JVM + standardni knjižnici za Javo.

- ADO.NET

ADO.NET podatkovna tehnologija omogoča preprost in zmogljiv dostop do podatkov. Tako kot njegov predhodnik pozna »connected« - dostop do podatkov, omogoča pa tudi »disconnected« ali prekinjen dostop. Podatkovna povezava se vzpostavi in ostane odprta le, dokler se izvajajo operacije iz vira podatkov, kar je uporabno predvsem pri podatkovnih zbirkah z več uporabniki.

Dostop do podatkov pri ADO.NET temelji na dveh entitetah: DataSet-u, v katerem hranimo podatke na lokalnem računalniku in Data Provider-ju, ki vsebuje razrede za upravljanje s podatki in podatkovno bazo.

- ASP.NET

ASP.NET (Active Server Pages) je Microsoftova tehnologija, ki omogoča kreiranje dinamičnih spletnih strani. Dinamične spletne strani se od navadnih spletnih strani razlikujejo po tem, da vsebujejo tudi elemente, ki se dinamično spreminjajo med samim pregledovanjem spletne strani. Ti dinamični elementi ASP se kreirajo na samem strežniku in so naknadno prevedeni v kodo HTML. Dinamične strani omogočajo, da se uporabniku predstavi vsebina, ki jo je na strežniku na podlagi uporabnikovih zahtev in/ali podatkov, ki so zapisani v nekem viru (razne vrste podatkovnih baz, dokumenti XML, oddaljen računalniški sistem, ki komunicira s strežnikom …), generiral nek program.

- BASE CLASS LIBRARY

Base Class Library (BCL) je standardna knjižnica, ki je na voljo v vseh jezikih, ki uporabljajo ogrodje.NET. Ogrodje vključuje BCL, da zapre v ohišje številne skupne funkcije, kot so datoteke za branje in pisanje, podatkovne baze, medsebojni vplivi in XML manipulacije, katere programer uporablja za lažje delo. BCL ima veliko večji pomen kot ostale standardne knjižnice pri drugih jezikih, ki ne uporabljajo ogrodja.NET.

4.5. Podatkovna baza

Podatkovna baza mora biti implementirana na tak način, da bo zadoščala vsem uporabniškim potrebam in zahtevam, ki so bile omenjene v tem poglavju. Podatkovna baza bo načrtovana s pomočjo Power Designer-ja, implementirana pa v celoti z Microsoft Sql Serverjem 2005. Nameščena bo na istem računalniku kot sama aplikacija, na katerem se bodo izvajale tudi varnostne kopije. Več o tem v poglavju Izvedbe.

4.5.1. Normalizacija podatkovne baze

Normalizacija je proces, ki zagotavlja, da relacije, oz. tabele ne vsebujejo dvoumnih ali redundantnih podatkov. Lahko bi tudi rekli, da je normalizacija postopek, s katerim pridemo do množice primernih relacij, ki ustrezajo vsem potrebam poslovne domene.

Prednosti takih podatkovnih baz se odražajo predvsem v enostavnejšem dostopu do podatkov ter vzdrževanju le-teh, veliko bolj so učinkovite, s takim načinom pa tudi bolje izrabljamo diskovne kapacitete.

Relacija je v prvi normalni obliki, če:

- nima ponavljajočih atributov; to pomeni, da ne obstajajo atributi ali skupine atributov, ki bi imele več vrednosti pri isti vrednosti ostalih atributov (na presečišču ene vrstice in enega stolpca je več vrednosti)
- ima definiran primarni ključ in določene funkcionalne odvisnosti.

Relacija je v drugi normalni obliki, če:

- je v prvi normalni obliki
- ne vsebuje parcialnih odvisnosti, kar pomeni, da noben atribut, ki ni del ključa, ni funkcionalno odvisen le od dela primarnega ključa, temveč od celotnega ključa.

Relacija je v tretji normalni obliki, če:

- je v drugi normalni obliki
- ne vsebuje tranzitivnih funkcionalnih odvisnosti, kar pomeni, da med atributi, ki niso del primarnega ključa, ni funkcionalnih odvisnosti.

Funkcionalna odvisnost opisuje razmerja med atributi relacije. Če sta A in B atributa relacije R, potem je atribut B funkcionalno odvisen od atributa A (notacija: A→ B), če je vsaki vrednosti atributa A pridružena natanko ena vrednost atributa B. V tem primeru atribut A predstavlja **determinanto** za atribut B. Za funkcionalno odvisnost lahko rečemo, da je to lastnost pomena (semantike) atributov relacije.

4.5.2. Primer normalizacije dela podatkovne baze

Začetni problem:

Prevoz (id_prevoza, st_naloga, datum_odh, datum_prih, drz_odh, drz_prih, km_odh, km_prih, litri_odh, litri_prih, prejeti_denar, ostanek_denar, id_komp, oznaka, reg_do, zadnji_servis, menjava_gum, opombe, id_voznik, ime, priimek, naslov, st_voz_dov, veljavnost**)**

1. NORMALNA OBLIKA

Določitev funkcionalnih odvisnosti:
F = **{**id_prevoza → (st_naloga, datum_odh, datum_prih, drz_odh, drz_prih, km_odh, km_prih, litri_odh, litri_prihod, prejeti_denar, ostanek_denarja), id_komp → (oznaka, reg_do, zadnji_servis, menjava_gum, opombe), id_voznik → (ime, priimek, naslov, st_voz_dov, veljavnost)**}**

Določitev primarnih ključev:
Prevoz (id_prevoza, st_naloga, datum_odh, datum_prih, drz_odh, drz_prih, km_odh, km_prih, litri_odh, litri_prih, prejeti_denar, ostanek_denar, **id_komp**, oznaka, reg_do, zadnji_servis, menjava_gum, opombe, **id_voznik**, ime, priimek, naslov, st_voz_dov, veljavnost**)**

2. NORMALNA OBLIKA

Tu razbijemo relacijo tako, da bodo vsi atributi odvisni od svojega ključa.
Oznaka # pomeni tuj ključ.

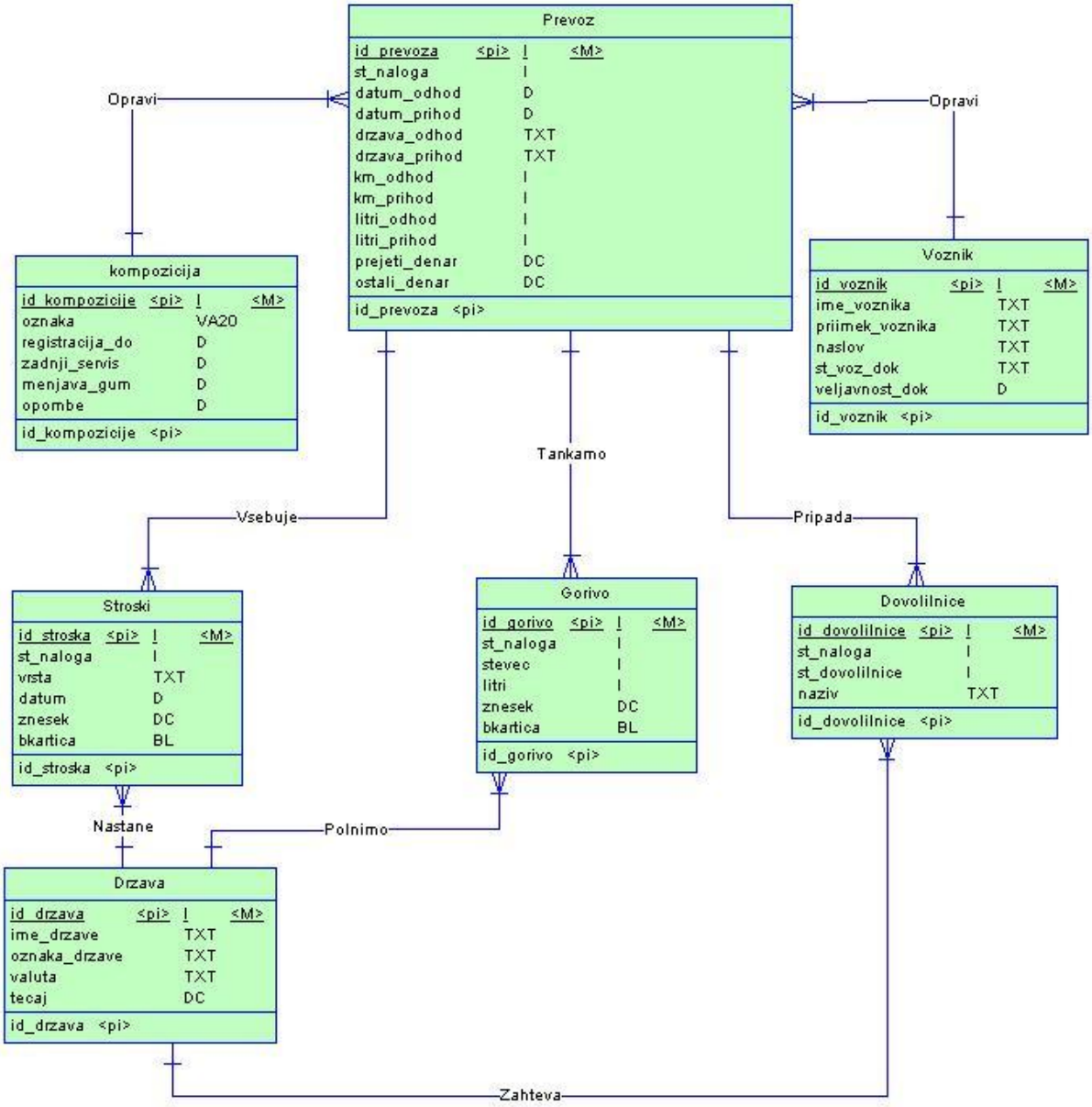
Prevoz (id_prevoza, **#id_komp, #id_voznik**, st_naloga, datum_odh, datum_prih, drz_odh, drz_prih, km_odh, km_prih, litri_odh, litri_prih, prejeti_denar, ostanek_denar**)**
Kompozicija (id_komp, oznaka, reg_do, zadnji_servis, menjava_gum, opombe **)**
Voznik (id_voznik, ime, priimek, naslov, st_voz_dov, veljavnost**)**

3. NORMALNA OBLIKA

Relacija se že nahaja v tretji normalni obliki, kjer več ni funkcionalnih odvisnosti med atributi.

4.5.3. Izdelava logičnega podatkovnega modela (LDM)

Na naslednjem diagramu sem predstavil tabele in povezave, ki bodo osnovale aplikacijo. Spodnji model bo uporabljen v končnem programu. Prav tako so tukaj opisane tabele in njihovi atributi ter podatkovni tipi. Relacije so poimenovane z imenom, ki nakazujejo medsebojne odnose med tabelami. Podatkovna baza je v tretji normalni obliki.



Slika 4: Logični podatkovni model programa

5. NAČRTOVANJE

5.1. Razvojna programska oprema

- **Microsoft Visual studio**

Microsoft Visual Studio je integrirano razvojno orodje (IDE), katerega lahko uporabimo za razvijanje konzolnih aplikacij, aplikacij z grafičnimi vmesniki, internet aplikacij, strežniških servisov, mobilnih aplikacij itd.. Zasnovan je bil za razvoj aplikacij z Microsoft .NET Framework 2.0 ogrodjem.

.NET ogrodje obsega skupno izvajalno okolje in zbirko raznovrstnih razrednih knjižnic. Ker se vsi programski jeziki v ogrodju .NET Framework prevajajo v vmesni jezik, so lahko posamezni deli aplikacije napisani v različnih visokih jezikih. Vmesni jezik je univerzalen jezik, ki povezuje visoke programske jezike (VB, C++, C# ...) in strojni jezik. Tako je lahko ena aplikacija v Visual Studiu napisana hkrati v več jezikih, kar omogoča stabilnejše in varnejše računalniško okolje. Podprti programski jeziki so: C, C++, C#, Visual Basic, .NET jeziki, JavaScript, CSS, Chrome, F#, Python, Ruby, XML/XSLT, HTML/XHTML, Microsoft J# in drugi.

Razvojno okolje vsebuje:

1. urejevalnik programske kode
2. orodje za načrtovanje grafičnih vmesnikov
3. orodjarno
4. orodje za načrtovanje projektov
5. razhroščevalnik
6. prevajalnik
7. pregledovalnik objektov, dokumentov, rešitev
8. kontrolnik izvirnih datotek in verzij.

Po želji lahko vgradimo tudi svoje dodatke, ki so v pomoč pri razvoju.

- **Programski jezik C#**

V zadnjih letih smo bili le redko priče nastanku novega programskega jezika. Na tržišču prevladujejo C++, java in basic, poleg tega pa srečamo številne specializirane jezike, ki jih večinoma uporabljajo posamezni proizvajalci. Internet je na to področje prinesel nove dimenzije, predvsem z javo, ki omogoča izdelavo programov, delujočih na različnih računalnikih in operacijskih sistemih pravzaprav brez spremembe v kodi. Javo so hitro prevzeli številni proizvajalci, tudi Microsoft, čeprav ne brez zapletov.

Videti je, da je Microsoftu celoten zaplet okoli jave in dejstvo, da ne nadzorujejo njenega razvoja, postalo zelo moteče. Zato so se odločili razviti lasten programski jezik, ki so ga razvijali pod delovnim imenom Cool. Konec junija pa so uradno predstavili nov jezik, ki so ga nato poimenovali C# (Cis).

Že samo ime pove, da programski jezik izhaja iz sintakse jezika C++, s čimer so hoteli zagotoviti mehak prehod za uporabnike, vajene jezikov C in C++. Obenem pa so v jeziku poenostavili nekatere težavne dele, kot so novo samodejno zbiranje "smeti" (pomnilnika, ki ga koda ne potrebuje več) ter samodejno deklariranje spremenljivk. V sintaksi naj bi se zgledovali tudi po Visual Basicu in orodjem za hitro izdelavo programov RAD. S tem se C# po strukturi in načinu dela močno približuje javi. Toda za razliko od tega je C# močno usmerjen v poslovno rabo, kjer zna tesno sodelovati z nastajajočim standardom za poslovno izmenjavo podatkov XML. Prvenstveno naj bi ga uporabljali za razvoj komponent za spletno rabo, ki bi kot storitve tvorile sestavne dele večje celote. Jezik je zasnovan na tak način, da ga je mogoče preprosto izboljševati in razširjati, brez skrbi, da bi s tem zgubili združljivost z obstoječimi programi.

Toda Microsoft poudarja, da C# ni tekmec javi, podjetje pa bo za zdaj še naprej podpiralo tudi javo. Jezik so uvrstili v skupno strategijo, ki so jo poimenovali .Net, združuje pa vse od operacijskih sistemov, do namenskih programov in razvojnih orodij. Microsoft trenutno nima neposrednih načrtov, da bi podporo za C# prenesel tudi na druge osnove. Zato pa so napovedali, da so jezik ponudili kot odprt standard organizaciji ECMA (European Computer Manufacturer's Association), enemu od teles, ki skrbi za standardizacije na področju računalništva. S tem želijo pritegniti druge ponudnike, da bi C# prenesli na druge osnove, obenem pa skušajo narediti to, kar Sunu ni uspelo (ali pa ni hotel).

Deset najpomembnejših razlogov za uporabo Visual C# :

- Bogata dediščina jezika C++

Visual C# učinkovito izkorišča dediščino jezika C++, ki razvijalcem ponuja trdne temelje za inteligentno predmetno in komponentno usmerjeno programiranje, in jezik, ki je takoj domač in udoben poznavalcem C++ in Jave, ne glede na raven njihovega znanja.

- Sistem tipov, ki temelji na predmetih

Visual C# razvijalcem ponuja sodoben in intuitiven sistem tipov, ki temelji na predmetih in odpravlja potrebo po zapletenih kazalcih in predlogah drugih jezikov, ki so pogosto vir napak.

- Dostop do ogrodja Microsoft .NET Framework

Visual C# razvijalcem omogoča dostop do funkcij ogrodja Microsoft .NET Framework, sestavljenega iz zanesljive in varne knjižnice zbirk razredov, omrežnih funkcij, razredov za podatkovni dostop in številnih drugih funkcij.

- Komponentno usmerjen razvoj

Visual C# razvijalcem ponuja zmogljiv komponentno usmerjen razvojni jezik, ki podpira lastnosti, stvarna kazala, delegate, dedovanje, spremljanje številk različic, atribute in še veliko več.

- Pripombe XML

Visual C# razvijalcem omogoča uporabo pripomb v obliki XML (Extensible Markup Language) za uporabno in prilagodljivo dokumentiranje izvorne kode.

- Na standardih temelječ jezik

Visual C# razvijalcem ponuja na standardih temelječ jezik, ki zagotavlja udeležbo skupnosti razvijalcev in dosledno inovacijo.

- Interaktivne spletne storitve XML

Visual C# razvijalcem omogoča uvajanje in uporabo bogatih interaktivnih spletnih storitev XML, ki z združevanjem programske opreme z različnih osnov skrajšajo čas, potreben za razvijanje.

- Razvijanje za katerokoli napravo

Visual C# razvijalcem omogoča uporabo istih orodij in znanja pri razvoju programske opreme tako za zmogljive namizne računalnike, kot za številne različne ročne in brezžične naprave.

- Pomnilnik v slogu programskega jezika C

Visual C# omogoča razvijalcem, da po potrebi uporabijo upravljanje pomnilnika in kazalce, znane iz programskega jezika C, namesto da bi morali pisati kode v drugih jezikih in z drugimi orodji.

- Integrirano razvojno okolje Visual Studio .NET

Visual C# razvijalcem ponuja večkrat nagrajeno razvojno okolje Visual Studio .NET, ki podpira seznam opravil, urejevalnike lastnosti, tehnologijo Microsoft IntelliSense®, orodja za oblikovanje obrazcev in še mnogo več.

- **Crystal Reports**

Crystal Reports je močna, dinamična in odzivna rešitev za kreiranje, oblikovanje in enostavno vizualizacijo raznih poročil. Prenos le teh lahko poteka preko spleta ali integriranih aplikacij znotraj podjetja. Končnim uporabnikom omogoča pregledovanje enostavnih poročil, ki so narejena tako, da hitro povzamejo bistvo podatkov. Glede na to, kako namerava zaposleni uporabiti poročilo, ga lahko izvozi v različne oblike. To pomeni manj pritiska na razvijalce in na zaposlene, ki bi bili odgovorni za kreiranje in dostavljanje poslovnih poročil.

- **Microsoft SQL Server**

Za izdelavo podatkovne baze sem izbral Microsoft SQL Server, ki je najučinkovitejša podatkovna baza na operacijskem sistemu Windows NT. Vsa poslovna logika se izvaja direktno na strežniku v sami bazi podatkov v obliki shranjenih procedur (Stored Procedure), kar predvsem pripomore k dobremu odzivnemu času sistema tudi v primeru velikih baz podatkov.

Microsoft SQL Server je prva podatkovna baza, ki na vseh nivojih (strežnik, delovna postaja, prenosnik) uporablja enako kodno osnovo in zagotavlja stoodstotno združljivost kode, podpira samokonfiguracijo in samouglaševanje. Uporabniki srednjih in velikih sistemov bodo imeli zlasti koristi od storitev OLAP (Online Analytical Processing), ker je Microsoft SQL Server tudi prva podatkovna baza z integriranim OLAP strežnikom.

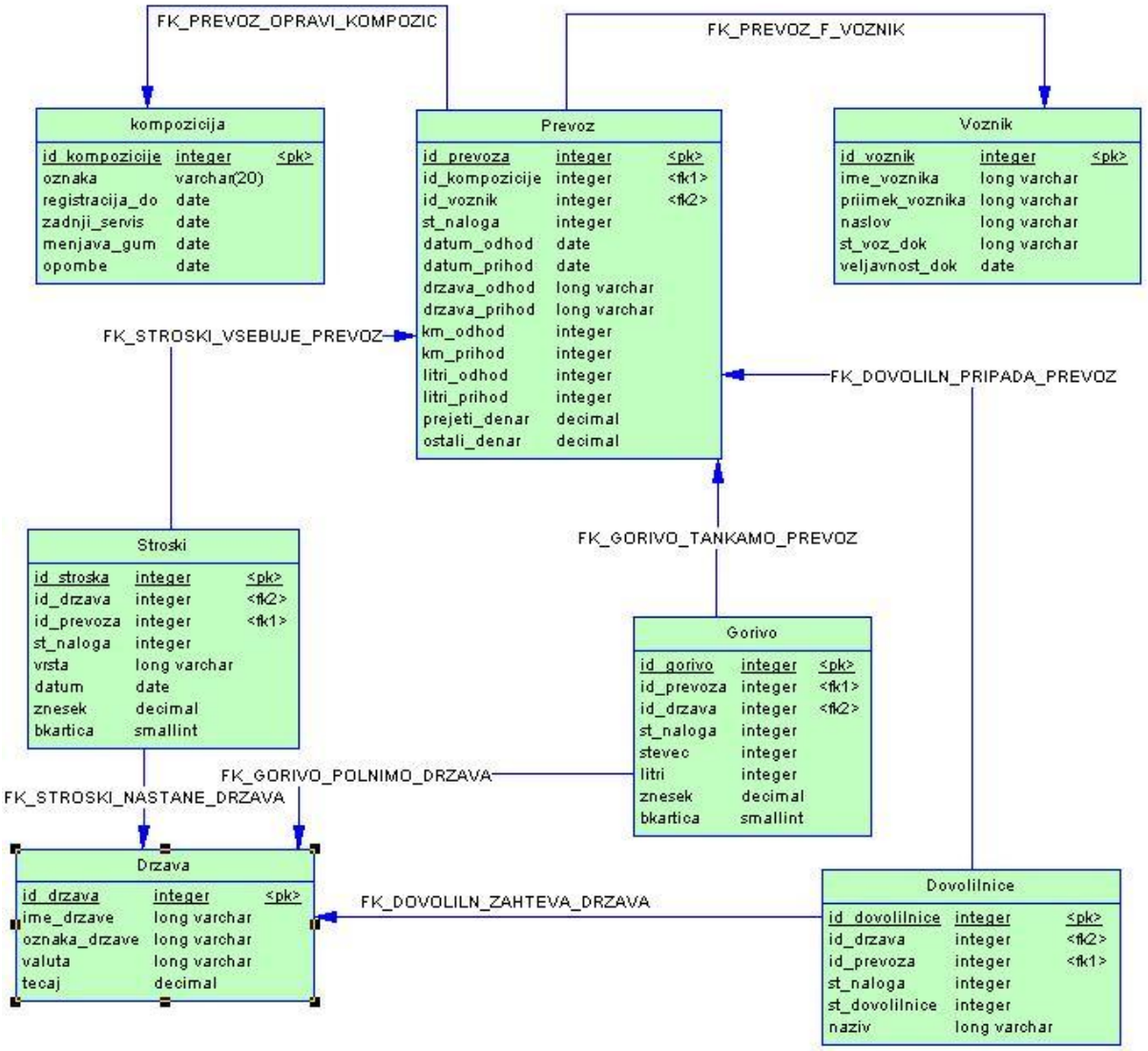
Storitve za prenos in transformacijo podatkov DTS (Data Transformation Services) poenostavljajo uvoz in izvoz heterogenih podatkov s pomočjo vmesnikov OLE DB in ODBC ali besedilnih datotek.

SQL Server izpolnjuje tudi specifične zahteve sodobnega mobilnega računalništva. Ponuja skalabilnost od prenosnih računalnikov do simetričnih večprocesorskih strežnikov in omogoča najširšo izbiro namestitev iste aplikacije. Vdelane možnosti replikacije podatkov, ki jih prinaša SQL Server, pa zagotavljajo samodejno sinhronizacijo spremenjenih podatkov med različnimi bazami podatkov.

SQL Server prinaša bistvene prednosti tudi v obvladovanju vseh tehnik upravljanja s podatki. Tu gre predvsem za povezave med posameznimi bazami podatkov, kot n.pr. prenosi podatkov med različnimi lokacijami. SQL Server podpira enostaven način izgradnje aplikacije, ki dobiva podatke dejansko tam, kjer nastajajo (brez programiranja prenosov). Nadalje omenjam replikacijo podatkov, kar pomeni osveževanje podatkov v več različnih bazah, kar spet pomeni dejanski vpogled v ažurirano bazo podatkov ne glede na to, na katerem mestu je izvor podatkov. Dosledna vgradnja v Backoffice okolje nam omogoča tudi učinkovito uporabo dodatnih funkcij kot n.pr. pošiljanje dokumentov po faxu, po elektronski pošti, uporabo interneta itd. ...

5.2. Načrtovanje fizične podatkovne baze

Fizični podatkovni model je izpeljan iz logičnega podatkovnega modela. V tem diagramu lahko vidimo, kako so dejansko zapisane tabele v podatkovni bazi, z vsemi pripadajočimi primarnimi in tujimi ključi.



Slika 5: Fizični podatkovni model programa

5.2.1. Opis posameznih entitet in relacij

ENTITETE

- **Prevoz**

Hrani osnovne podatke o odhodu, oziroma prihodu posameznega prevoznika. Primarni ključ je id_prevoza, vsebuje pa še dva tuja ključa: id_voznika in id_kompozicije, saj vsak prevoz opravi eden ali več voznikov z eno kompozicijo.

Polja tabele:
id_prevoza: glavni ključ
id_kompozicije: tuj ključ, ki nam pove, s katero kompozicijo se prevoz opravlja
id_voznika: tuj ključ, ki nam pove, kdo opravlja prevoz
st_naloga: pod katero številko naloga se bo prevoz knjižil
datum_odhod: kdaj se je prevoz začel
datum_prihod: kdaj se je prevoz zaključil
drzava_odhod: začetna točka prevoza
drzava_prihod: ciljna država
km_odhod: stanje kilometrov ob odhodu
km_prihod: stanje kilometrov ob prihodu
litri_odhod: stanje goriva v rezervoarju ob odhodu
litri_prihod: stanje goriva v rezervoarju ob prihodu
prejeti_denar: koliko denarja je voznik prejel ob odhodu
ostali_denar: razlika denarja

- **Kompozicija**

Kompozicija predstavlja sklop vlečnega in priklopnega vozila. Entiteta hrani podatke, ki so potrebni za obveščanje o bodočih dogodkih, kot so redni servis, menjava gum …

Polja tabele:
id_kompozicije: glavni ključ
oznaka: registrska številka priklopnega in vlečnega vozila
registracija_do: datum, ko poteče registracija
zadnji_servis: kdaj je bil na vozilu opravljen zadnji servis
menjava_gum: datum zadnje menjave gum
opombe: menjave večjih rezervnih delov

- **Voznik**

Entiteta voznik hrani vse potrebne podatke o posameznem vozniku.

Polja tabele:
id_voznik: glavni ključ
ime_voznika: ime voznika
priimek_voznika: priimek voznika
naslov: bivališče voznika
st_voz_dok: id vozniškega dovoljenja
veljavnost: do kdaj dovoljenje velja

- **Stroški**

Ta vsebuje vse podatke, potrebne za obvladovanje stroškov (datum, kraj, znesek, način plačila, valuta). Vsebuje še dva tuja ključa, in sicer id_države in id_prevoza, saj tako lahko v tabeli stroškov lepo vidimo, kateremu prevozu pripada določen strošek.

Polja tabele:
id_stroška: glavni ključ
id_drzave: tuj ključ, ki nam pove, kje je strošek nastal
id_prevoza: tuj ključ, ki določa, kateremu prevozu pripada strošek
st_naloga: pod katero številko se bo strošek knjižil
vrsta: parkirnine, cestnine, kazni, špedicije …
datum: kdaj je strošek nastal
znesek: višina stroška
bkartica: plačilo z gotovino ali z bančno kartico

- **Gorivo**

Tukaj hranimo podate o vsakem točenju goriva na določenem prevozu. Ti podatki so potrebni za vodenje posebne evidence o porabljenem gorivu kot največjem strošku. Prav tako vsebuje dva tuja ključa, ki sta enaka kot pri entiteti stroškov.

Polja tabele:
id_goriva: glavni ključ
id_prevoza: pod kateri prevoz spada natakanje goriva
id_drzava: v kateri državi se gorivo kupuje
st_naloga: pod kateri nalog se bo knjižil strošek
stevec: stanje kilometrov ob nakupu goriva
litri: koliko litrov se natoči
znesek: znesek stroška
bkartica : način plačila

- **Država**

Ta entiteta ima funkcijo šifranta. Vanj se vnašajo države, preko katerih prevoz poteka, povezan je pa z vsemi entitetami, ki vsebujejo celico, kraj, oziroma državo.

Polja tabele:
id_drzava: glavni ključ
ime_drzave: za katero državo gre
oznaka_drzave: mednarodna kratica
valuta: katero valuto ima država
tecaj: vrednost tečaja v evrih (zaradi preračunavanja)

- **Dovolilnice**

Če prevoz poteka preko držav, ki niso članice EU, potem potrebujemo še entiteto dovolilnice, s katero urejamo prejete in porabljene dovolilnice v sklopu posameznega prevoza. Vsebuje dva tuja ključa: id_države (kateri državi pripada dovolilnica), ter id_prevoza(na katerem prevozu je bila porabljena).

Polja tabele:
id_dovolilnice: glavni ključ
id_drzava: kateri državi pripada dovolilnica
id_prevoza: na katerem prevozu se je porabila
st_naloga: pod katero številko naloga spada
st_dovolilnice: fizična številka na dovolilnici
naziv: vrsta dovolilnice (tranzitna, tretja, bilateralna)

RELACIJE

- **Prevoz – Voznik:** En voznik opravi enega ali več prevozov
- **Prevoz – Kompozicija:** Ena kompozicija opravi enega ali več prevozov
- **Prevoz – Stroški:** En prevoz vsebuje enega ali več stroškov, oziroma na enem prevozu nastane najmanj en strošek.
- **Prevoz – Gorivo:** Na enem prevozu lahko natočimo gorivo enkrat ali večkrat
- **Prevoz – Dovolilnice:** Na enem prevozu porabimo eno ali več dovolilnic
- **Stroški – Država:** V eni državi nastane eden ali več stroškov
- **Gorivo – Država:** V eni državi lahko polnimo rezervoar enkrat ali večkrat
- **Dovolilnice – Država:** Ena država zahteva eno ali več dovolilnic

5.3. Podoba uporabniškega vmesnika

Vsak uporabnik, ki prvič vidi vnosno masko, mora v nekaj sekundah ugotoviti smiselnost postavljenega uporabniškega vmesnika, zato je podoba in jasnost le tega zelo pomembna. Če podoba vmesnika ni primerna, se lahko uporabniki zmedejo in se ob tem zmotijo pri sami uporabi. Zaradi tega sem se opiral na uporabnikom že znane uporabniške vmesnike (standardni Windows programi) in po teh sem nekako zgradil samo podobo. Tako bodo uporabniki lahko iz izkušenj vedeli, kje se odpre novo okno, kje iskati pomoč itd. …

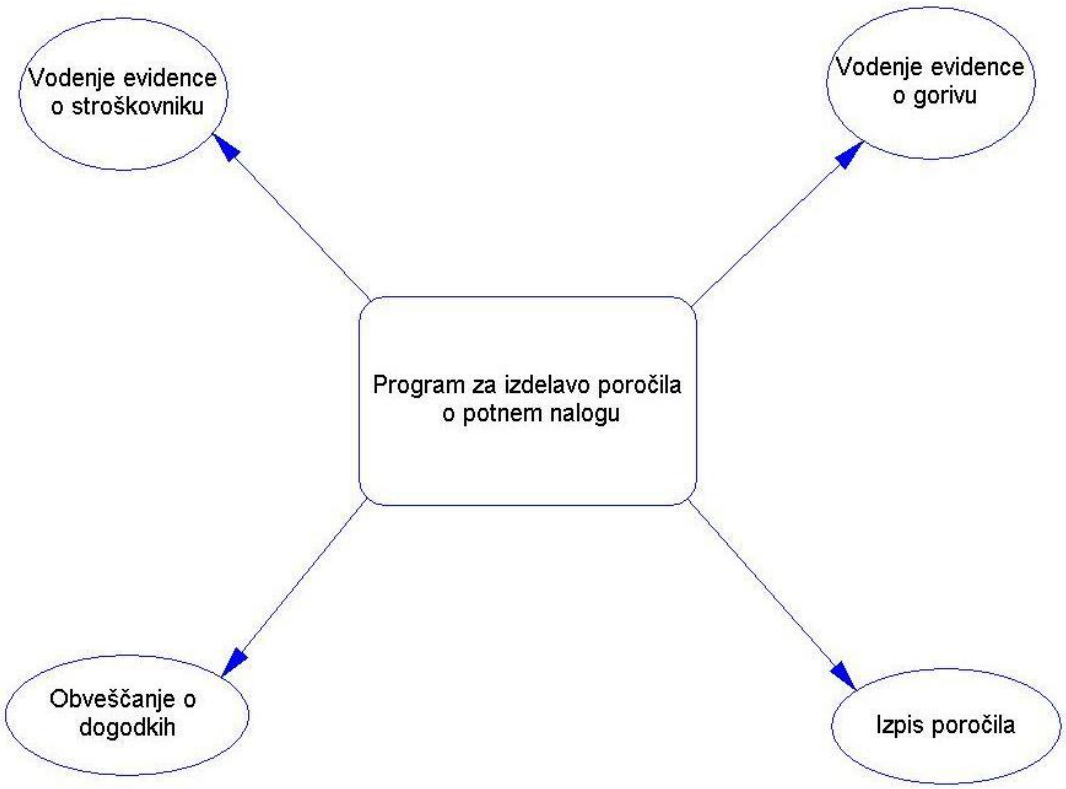
Pri načrtovanju grafičnega vmesnika sem se držal naslednjih opornih točk:

- uporaba primernih barv (ne smejo preveč izstopati)
- dovolj velika pisava zaradi preglednosti in lažjega branja
- delovna površina obsega celoten zaslon
- smiselna razporeditev gumbov (zaporedje)

5.4. Načrtovanje sistema

5.4.1. Glavne funkcionalnosti programa

Na prvem nivoju sem predstavil glavne procese programa. Ti so: vodenje evidenc o stroških ter gorivu, obveščanje o dogodkih in na koncu sam izpis poročila. Spodnji diagram v grobem opisuje funkcionalnost programa.

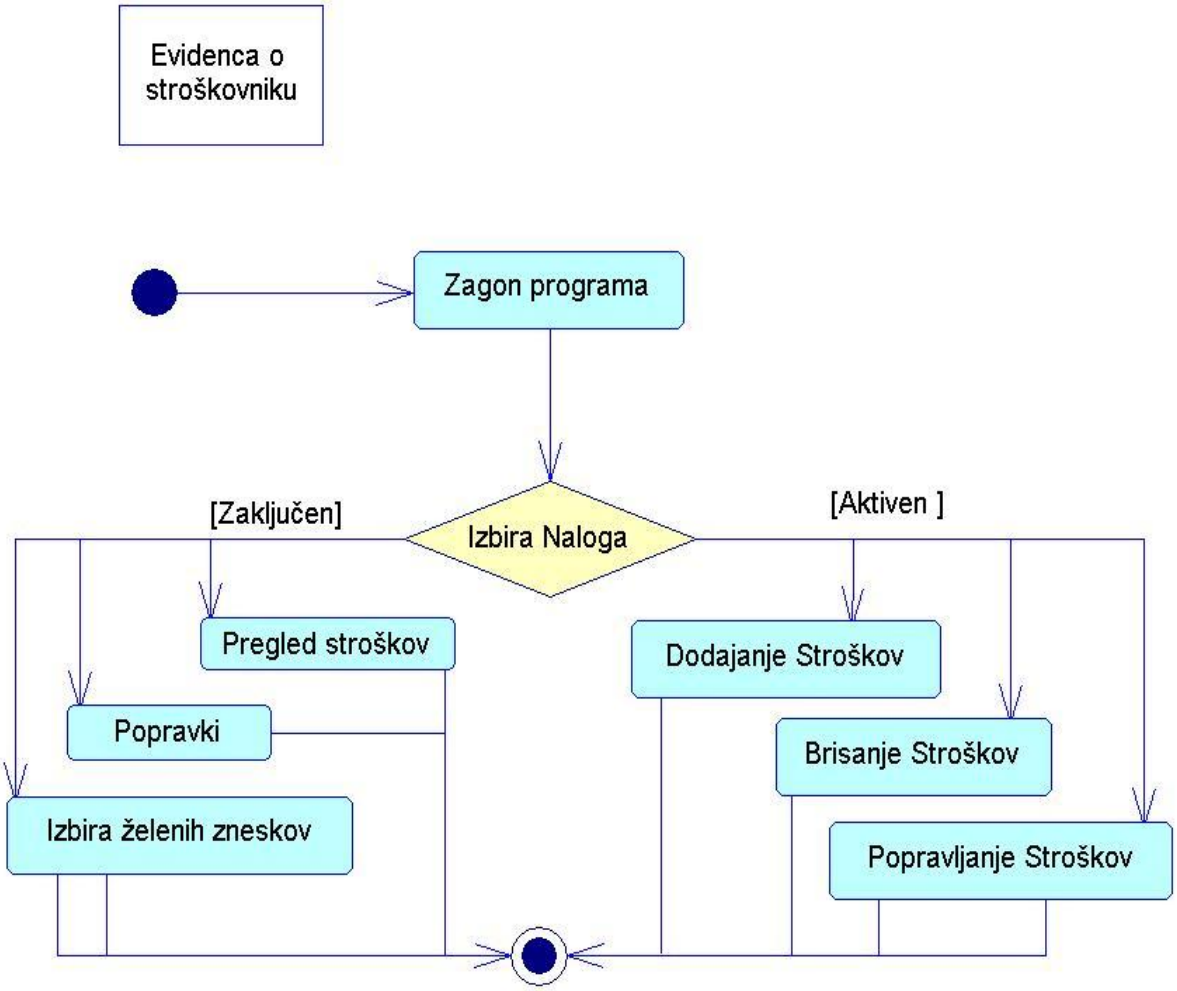


Slika 6: Procesi programa

5.4.2. Evidenca o stroškovniku

Na spodnjem diagramu podrobneje razlagam evidenco o stroškovniku. Uporabnik torej zažene program ter se odloči, kateri stroški ga zanimajo. Na voljo ima že zaključen nalog in nov, oziroma zadnji aktiven nalog.

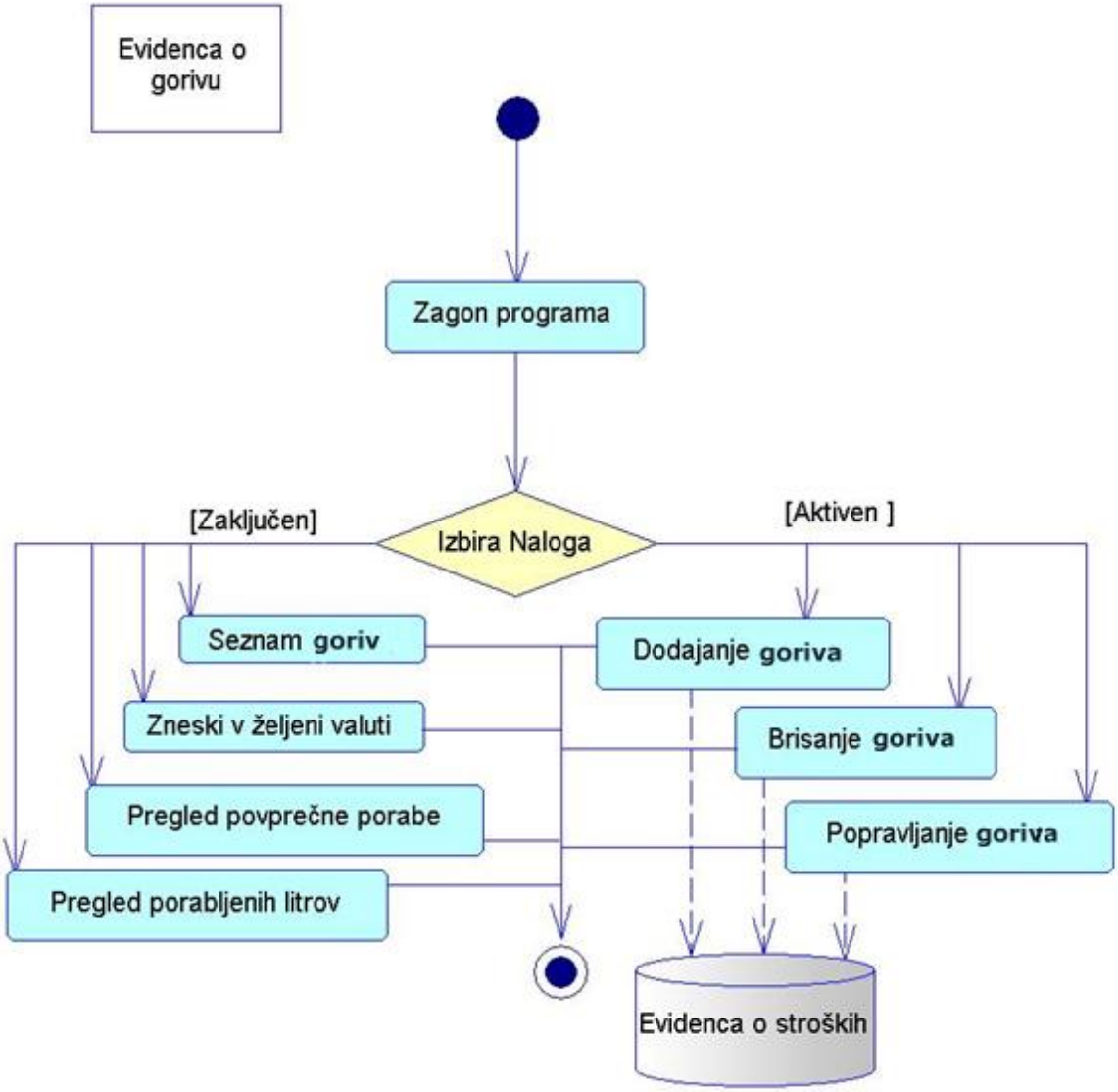
Na podlagi izbire program ponuja različne funkcionalnosti, kot so prikazane na spodnjem diagramu. Procesi se torej delijo glede na izbiro naloga.



Slika 7: Evidenca stroškovnika

5.4.3. Evidenca o gorivu

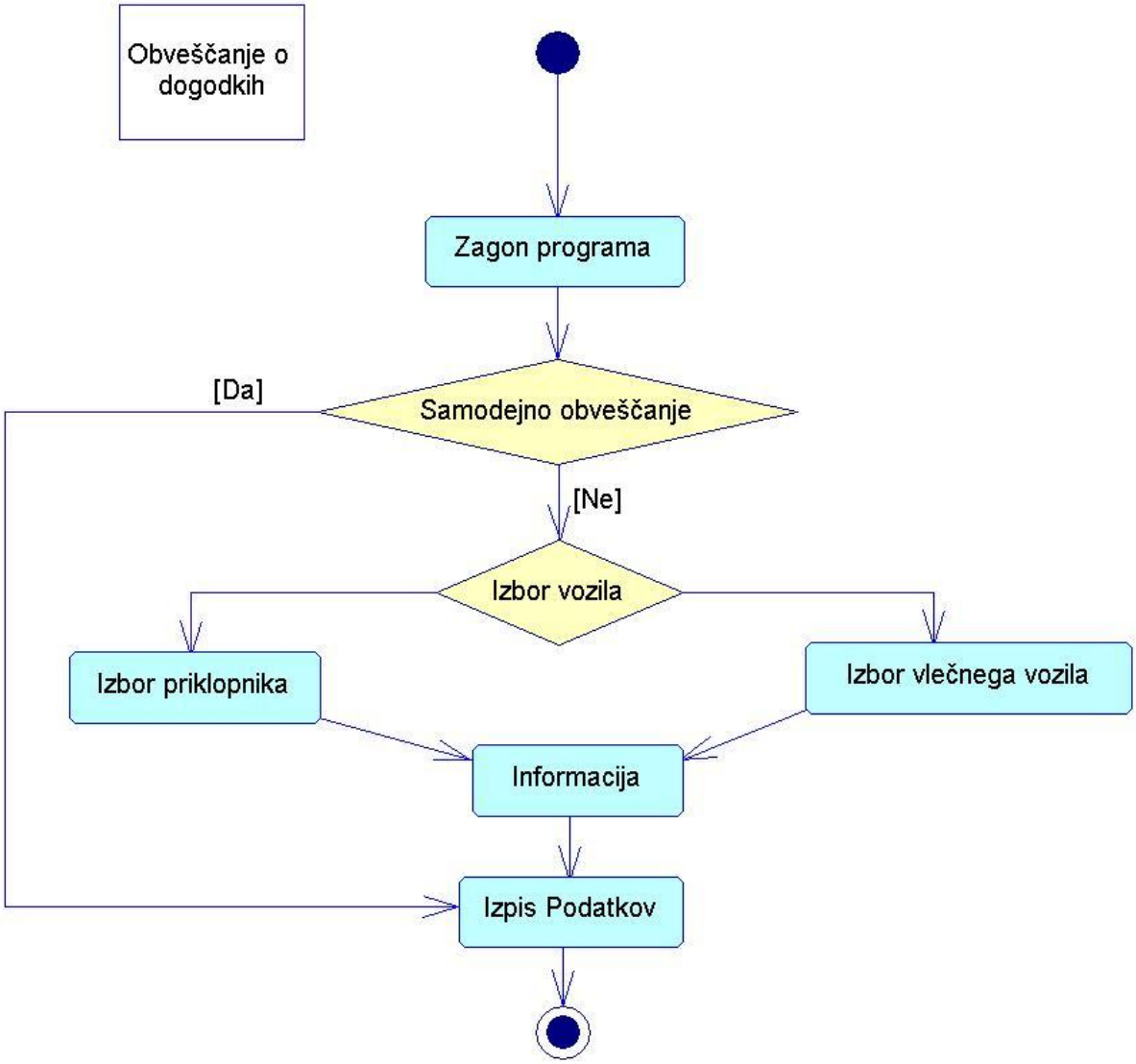
Tukaj se tudi funkcionalnosti delijo glede na izbiro naloga. Procesi so si dokaj podobni kot na prejšnjem diagramu. Glavna razlika je v tem, da so procesi pri aktivnem nalogu povezani s stroškovnikom, saj je vsako nalivanje goriva dejanski strošek, kar pomeni, da takrat, ko vnesemo »nalivanje goriva«, se (potrebni) podatki shranijo tako v evidenco nalivanja goriva, kakor tudi v evidenco stroškov. Evidenca o gorivu je ločena od stroškov zato, ker pri gorivu potrebujemo še dodatne podatke, ki niso vezani na stroške (n.pr. stanje km, iztočenih litrov …).



Slika 8: Evidenca o gorivu

5.4.4. Obveščanje o dogodkih

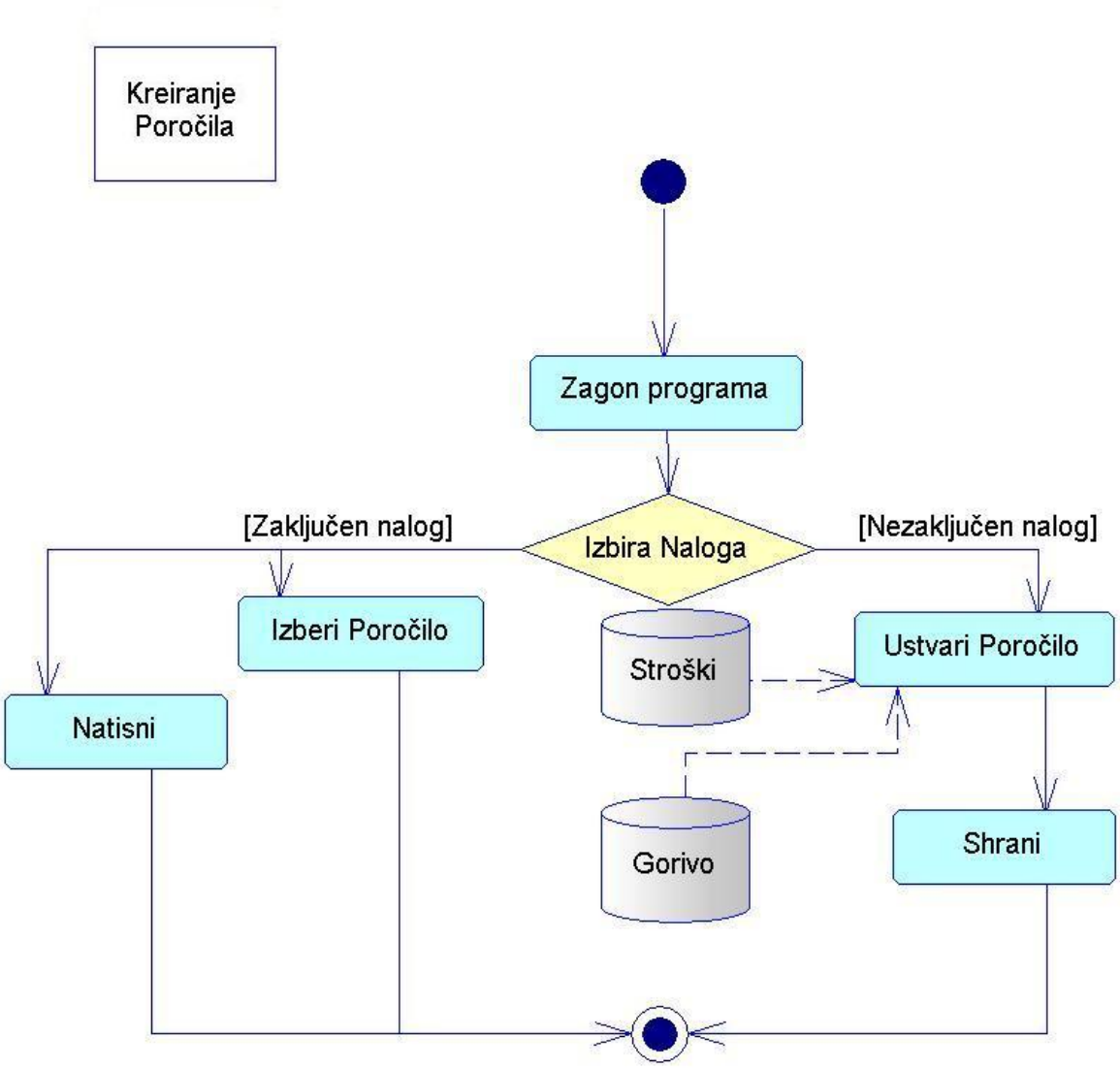
Ko uporabnik zažene program, se sproži tako imenovano samodejno obveščanje, ki nas opozarja o bližajočih se dogodkih, in sicer 30 dni pred rokom. Uporabniki nad to funkcijo nimajo posebnega nadzora, saj program le opozori na dogodek v obliki pojavnega okna. Lahko pa seveda tudi izberemo, kaj točno nas zanima, in sicer tako, da najprej izberemo, katera vrsta vozila nas zanima, nato še, katero vozilo točno in na koncu izberemo, kaj nas zanima pri tem vozilu.



Slika 3: Obveščanje o dogodkih

5.4.5. Kreiranje poročil

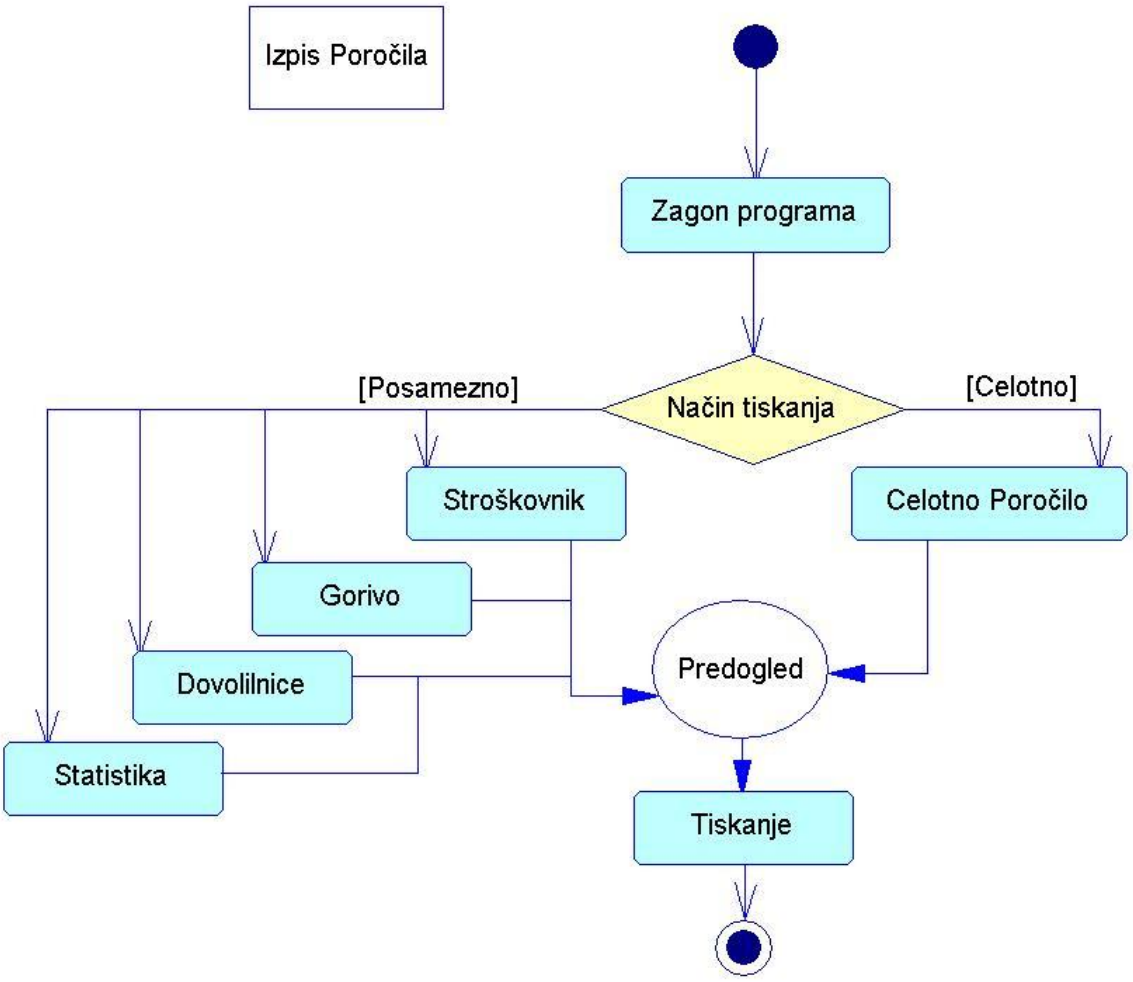
Uporabnik po zagonu programa izbere nalog. V primeru zaključenega naloga pomeni, da je poročilo že bilo ustvarjeno, zato ga lahko izbere in natisne. V nasprotnem primeru pa kreiramo poročilo. S pomočjo stroškovnika in evidence o gorivu ter ostalih potrebnih podatkov se poročilo ustvari, in sicer na način, kot si ga uporabnik želi. To pomeni, da lahko na poročilo dodaja posamezne odseke, na voljo pa ima tudi izbiro celotnega poročila.



Slika 40: Kreiranje poročil

5.4.6. Izpis poročil

Uporabnik lahko tiska tako posamezne odseke poročila, kot tudi celotno poročilo, ki obsega tudi osnovne podatke o prevozu. Enako velja za shranjevanje poročil na trdi disk s tem, da ima na voljo še izbiro shranjevalnega formata.



Slika 11: Tiskanje poročila

6. IZVEDBA

Po zaključku načrtovanja sem se lotil same izvedbe projekta. V tem poglavju bom obrazložil način programiranja, delo s podatkovno bazo ter postopke pri izdelavi končnega poročila.

6.1. Programiranje (kodiranje)

Pri samem programiranju sem se držal osnovnih pravil lepega programiranja. Tu predvsem mislim na poimenovanje komponent, objektov in preglednost programske kode, ter sprotno komentiranje same kode.

Poimenovanje komponent ter spremenljivk:

Razvojno okolje Visual Studio ima nam ponuja veliko komponent in objektov pri programiranju, zato je priporočeno njihovo primerno poimenovanje, saj se v nasprotnem primeru med kodo ne bi znašli, če bi med vrsticami videli poimenovanja, kot so »TextBox1, label2, checkBox6 …«.

Zato sem temu posvetil posebno pozornost in si določil pravila poimenovanja, katerih sem se držal med celotnim projektom. Kar se pa dotika posameznih spremenljivk, sem jih smiselno poimenoval, tako, da se bom v primeru, če bo treba popravljati kodo, hitro znašel.

Pr.	**Tip gradnika**	**Pr.**	**Tip gradnika**	**Pr.**	**Tip gradnika**
cb	Check Box	img	Image	il	Image List
btn	Command Button	lb	Label	lv	List View
cbo	Combo Box	lst	List Box	sb	Status Bar
dat	Data Control	opt	Option Button	tb	Toolbar
dir	Directory List Box	pnl	3D Panel	tv	Tree View
drv	Drive List Box	pic	Picture Box		
fil	File List Box	txt	Text Box		
fra	Frame	tmr	Timer		
hsb	Horizontal Scroll Bar	vsb	Vertical Scroll Bar		

Tabela 1: Predpone pri poimenovanju gradnikov
(predvsem so pomembna imena tistih gradnikov, na katere se sklicujemo v programski kodi)

Uporaba try-catch blokov:

Povsod, kjer sem predvidel možne napake, sem uporabljal try-catch bloke in primerno poimenoval napake. To je bilo predvsem pomembno tam, kjer je prihajalo do gnezdenj le teh, kar pomeni, da lahko na enem mestu pride do več napak.

Komentiranje programske kode in preglednost:

Komentiral sem sproti, še posebej na mestih, kjer so procedure dolge in zapletene. V komentarjih sem navajal, kaj procedura dela, še bolj pomembno pa je, na kakšen način sem rešil problem, da bodo potencialni popravki enostavnejši in hitrejši. Za preglednost in zamike vrstic v kodi pa poskrbi Visual Studio sam.

Postavitev komponent:

Komponente na uporabniških oknih sem postavljal v smiselnem zaporedju in v primerni velikosti, in sicer z namenom, da se uporabnik ne bo trudil iskati posebne gumbe in da pri branju ne bo naprezal oči.

Na sledečih slikah navajam primere programske kode, ki upoštevajo zgoraj omenjena pravila programiranja.

```
private void btn_vnesiGorivo_Click(object sender, EventArgs e)
{
      //POVEZAVA NA PODATKOVNO BAZO
      SqlConnection povezava_gorivo = new SqlConnection(
      @"SERVER=(local);" +
      @"Database= OnRoadApp;" +
      @"Trusted_Connection=Yes");
        povezava_gorivo.Open();
        try
        {
            SqlCommand sql = povezava_gorivo.CreateCommand();
            sql.CommandText = "INSERT INTO Gorivo"
           + "(kraj,datum,stanje_stevca_tank,litri,znesek,valuta,bancna_kartica,st_naloga)"
           + "VALUES(@kraj,@datum,@stanje_stevca_tank,@litri,@znesek,@valuta,@bancna_kartica,@st_naloga)";
            //DOLOČITEV PARAMETROV
            sql.Parameters.Add("@kraj", SqlDbType.Text);
            sql.Parameters.Add("@datum", SqlDbType.DateTime);
            sql.Parameters.Add("@stanje_stevca_tank", SqlDbType.Int);
            sql.Parameters.Add("@litri", SqlDbType.Decimal);
            sql.Parameters.Add("@znesek", SqlDbType.Decimal);
            sql.Parameters.Add("@valuta", SqlDbType.Text);
            sql.Parameters.Add("@bancna_kartica", SqlDbType.Text);
            sql.Parameters.Add("@st_naloga", SqlDbType.VarChar);
            //PRIREDITVE
            sql.Parameters["@kraj"].Value = tb_kraj_gorivo.Text;
            sql.Parameters["@datum"].Value = tb_datum_gorivo.Text;
            sql.Parameters["@stanje_stevca_tank"].Value = tb_stevec_gorivo.Text;
            sql.Parameters["@litri"].Value = tb_litri_gorivo.Text;
            sql.Parameters["@znesek"].Value = tb_znesek_gorivo.Text;
            sql.Parameters["@valuta"].Value = tb_valuta_gorivo.Text;
            string nacin_placila;
            if (cb_bancna.Checked)
                nacin_placila = "BančnaKartica";
            else
                nacin_placila = "Gotovina";
            sql.Parameters["@bancna_kartica"].Value = nacin_placila; ;
            sql.Parameters["@st_naloga"].Value = lb_st_naloga.Text;
            //IZVRŠITEV KOMANDE
            SqlDataReader read = sql.ExecuteReader();
            povezava_gorivo.Close();
```

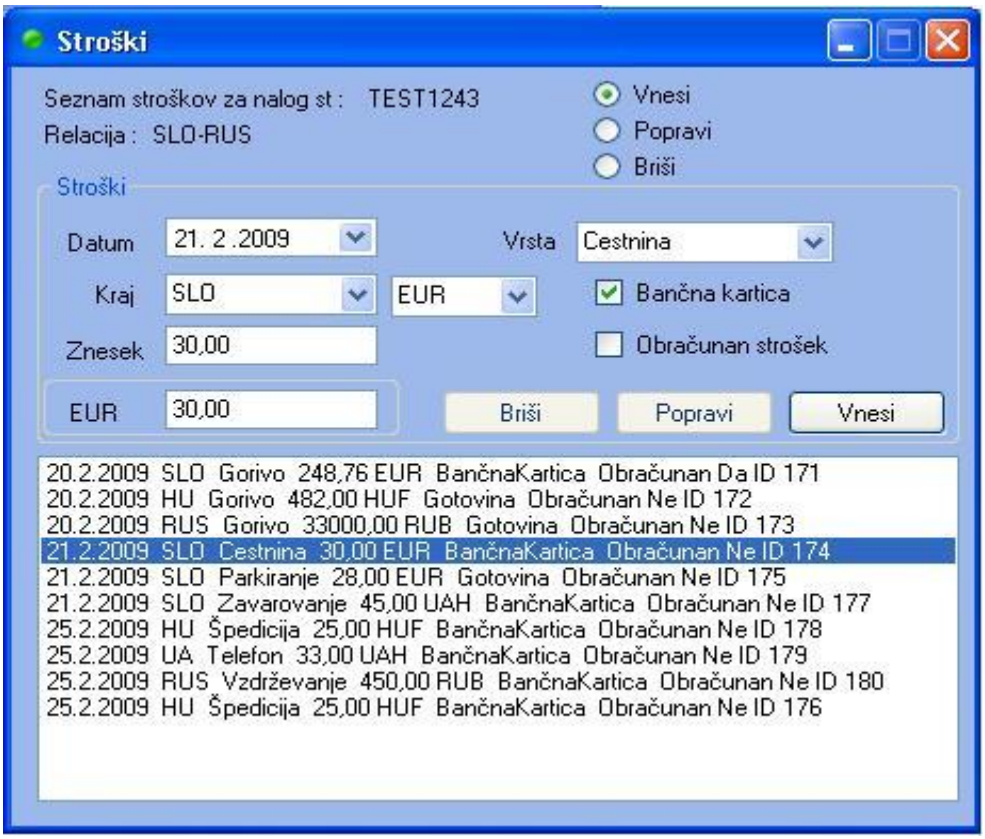
Slika 52: Primer dela kode za vnos podatkov

```
private void btn_updatePrihod_Click(object sender, EventArgs e)
{
    //DOPOLNITEV TABELE PREVOZ OB PRIHODU
    try
    {
        //POVEZAVA NA PODATKOVNO BAZO
        SqlConnection povezava_odhod = new SqlConnection(
                @"SERVER=(local);" +
                @"Database= OnRoadApp;" +
                @"Trusted_Connection=Yes");
        povezava_odhod.Open();
        SqlCommand sql = povezava_odhod.CreateCommand();
        //SQL POIZVEDBA
        sql.CommandText = "UPDATE Odhod_prihod " +
                          "SET " +
                          " prihod_dne = @prihod_dne, " +
                          " stanje_tanka_pr = @stanje_tanka_pr, " +
                          " stanje_stevca_pr = @stanje_stevca_pr, " +
                          " ostanek_denarja = @ostanek_denarja " +
                          "WHERE st_naloga = '" + lb_st_naloga.Text + "'";
        //DODAJANJE PARAMETROV
        sql.Parameters.Add("@prihod_dne", SqlDbType.Text);
        sql.Parameters.Add("@stanje_tanka_pr", SqlDbType.Int);
        sql.Parameters.Add("@stanje_stevca_pr", SqlDbType.Int);
        sql.Parameters.Add("@ostanek_denarja", SqlDbType.Decimal);
        sql.Parameters["@prihod_dne"].Value = dtp_datum.Text;
        sql.Parameters["@stanje_tanka_pr"].Value = tb_stanjeTanka.Text;
        sql.Parameters["@stanje_stevca_pr"].Value = tb_stanje_Stevca.Text;
        sql.Parameters["@ostanek_denarja"].Value = tb_denar.Text;
        SqlDataReader beriPoizvedbo = sql.ExecuteReader();
        povezava_odhod.Close();
        MessageBox.Show("Podatki so uspešno vnešeni","Sporočilo",MessageBoxButtons.OK,MessageBoxIcon.Information);
        Form1.ActiveForm.Close();
    }
    catch
    {
        MessageBox.Show("Vnesite vse podatke!\n.", "Sporočilo" , MessageBoxButtons.OK,MessageBoxIcon.Exclamation);
    }
}
```

Slika 6: Primer dela kode za posodobitev podatkov

```
private void btn_izracunajZneske_Click(object sender, EventArgs e)
{
     string id_naloga = lb_nalogi.SelectedItem.ToString();
     SqlConnection povezava = new SqlConnection(
               @"SERVER=(local);" +
               @"Database= OnRoadApp;" +
               @"Trusted_Connection=Yes");
        povezava.Open();
        SqlCommand sql7 = tst.CreateCommand();
        //ČE NI IZBRAN NOBEN POGOJ IZPIŠI CELOTNO VSOTO STROŠKOV
        if ((cbo_vsota.Text == "") && (cbo_placano.Text == "") &&(cbo_drzava.Text == ""))
        {
            sql7.CommandText = "SELECT SUM (znesek_eur) " +
                               "FROM Stroski " +
                               "WHERE st_naloga='" + id_naloga + "'";
            try
            {
                lb_znesek.Text = Convert.ToString((decimal)sql7.ExecuteScalar());
            }
            catch
            {
                lb_znesek.Text = "0,00";
            }
            povezava.Close();
        }
```

Slika 7: Primer dela kode za izračun zneskov



Slika 8: Uporabniško okno za vnos stroškov

6.2. Delo s podatkovno bazo

Kot sem že omenil, je podatkovna baza implementirana s pomočjo SQL SERVER-ja 2005 in je nameščena na enakem računalniku kot sam program, in sicer iz razloga, ker se ni pokazala potreba po ločevanju podatkovne baze, saj se bo program uporabljal le za računovodstvo na enem računalniku. Vsak dostop do podatkovne baze je realiziran preko takoimenovanih »SQL connection's« (v nadaljevanju povezave), ki se nahajajo v dotNet-ovi knjižnici SQLClient . Tem povezavam podamo tri parametre in sicer: kjer se baza nahaja, kako se le ta imenuje in zaupnost povezave. Primer je podan na zgornjih slikah.

Vsakokrat, ko je bilo potrebno dostopati, spreminjati, oziroma urejati podatkovno bazo, je bilo potrebno ustvariti povezavo do nje. Po uspešni povezavi pa so sledile takoimenovane »SQL command's«, ki nam omogočajo pisanje SQL poizvedb v podatkovni bazi in izvršbo le teh. Pisanje komand je zelo preprosto, saj je potrebno le znanje jezika SQL, oziroma njegove sintakse. Vsak vnos, popravek, izbris in branje iz/v podatkovno bazo je potekal preko teh komand.

Po opravljeni transakciji pa je potrebno vse povezave zapreti, saj jih ne potrebujemo več in ni smisla da ostajajo odprte in s tem uporabljajo pomnilnik.

Varnost:

Poskrbel sem tudi za varnostne kopije podatkovne baze, in sicer s pomočjo orodja »Backup« v sklopu SQL SERVER-ja. Enkrat tedensko se izvaja takoimenovani »incremental backup«, kar pomeni primerjavo trenutne podatkovne baze z zadnjo kopijo in zamenjava le tistih delov, ki so se spremenili.

6.3. Izdelava poročila o potnem nalogu

Kot sem že omenil, sem poročilo izdelal s pomočjo orodja Crystal Reports. Tu sem uporabil vrsto funkcionalnosti, kot so grupiranje podatkov, matematične funkcije, formule, podrobnosti in SQL poizvedbe po podatkih.

Poročilo vsebuje podatke iz vseh tabel v podatkovni bazi, zato sem se pri izdelavi skliceval na unikaten parameter z imenom številka naloga. To je enoličen podatek, po katerem se razlikujejo vsi prevozi. Torej vsak strošek prevoza pripada natanko eni številki naloga.

Zgled in velikost pisave sem prilagodil uporabniškim zahtevam. Odseki poročila in posamezni podatki so prav tako narejeni glede na zahteve in potrebe računovodje.

Poročilo vsebuje osnovne podatke o prevozu, stroškovnik, evidenco porabe goriva, evidenco dovolilnic in razdeljene stroške po spodaj naštetih kriterijih.

- **Obračunani stroški** – pomeni, da vse stroške razdelimo glede na to, ali so nastali v Sloveniji ali ne. To je predvsem pomembno zaradi povračila davkov.
- **Stroški glede na vrsto** – ta odsek vse stroške razdeli glede na vrsto stroška. Tu gre lahko za parkirnine, cestnine, kazni, špedicije …
- **Stroški glede na državo** – tu razdelimo stroške glede na državo, v kateri je strošek nastal.
- **Vsota obračunanih stroškov** – pomeni, da seštejemo vse stroške, ki so nastali v Sloveniji in tiste, ki so nastali izven Slovenije.
- **Skupni stroški** – predstavljajo skupne zneske porabljenega denarja na prevozu.
- **Statistika** – predstavlja zadnji odsek poročila, v katerem so opredeljeni posamezni statistični podatki prevoza.
- **Kriterij plačila** – je kriterij, katerega upoštevajo vsi ostali kriteriji. Pomeni, da stroške deli na tiste, ki so plačani z bančno kartico in tiste, ki so plačani z gotovino.

Naslednji dve sliki prikazujeta primer poročila in vseh opisanih kriterijev.

POROČILO POTNEGA NALOGA

ŠTEVILKA : 233257

Relacija :	SLO-ITAL-RUS	Odhod dne :	13.1.2009	Prihod dne :	10.2.2009
Voznik :	PANIČ BORIS	Stanje števca :	53.195	Stanje števca :	59.759
Vlečno voz. :	KK CZ-558	Stanje tanka :	1.170	Stanje tanka :	795
Priklopno vozilo :	A5-20KK	Prejeti denar :	400,00	Ostanek denarja :	50,00

POTNI STROŠKI

DATUM	KRAJ	VRSTA	ZNESEK	VALUTA	PLAČILO	EUR	OBR	ID
13-jan-09	SLO	Cestnina	9,50	EUR	BančnaKartica	9,50	Da	41
13-jan-09	SLO	Vzdrževanje	750,00	EUR	BančnaKartica	750,00	Ne	73
14-jan-09	SLO	Cestnina	23,60	EUR	BančnaKartica	23,60	Da	42
15-jan-09	ITA	Cestnina	51,73	EUR	BančnaKartica	51,73	Ne	43
15-jan-09	ITA	Cestnina	31,10	EUR	BančnaKartica	31,10	Ne	44
15-jan-09	SLO	Vzdrževanje	160,00	EUR	Gotovina	160,00	Ne	70
16-jan-09	SLO	Cestnina	20,20	EUR	BančnaKartica	20,20	Da	45
16-jan-09	SLO	Parkiranje	16,00	EUR	BančnaKartica	16,00	Da	46
19-jan-09	SLO	Špedicija	20,00	EUR	Gotovina	20,00	Ne	47
19-jan-09	SLO	Cestnina	39,50	EUR	BančnaKartica	39,50	Da	48
19-jan-09	HU	Cestnina	11,50	EUR	Gotovina	11,50	Ne	49
19-jan-09	SLO	Ostalo	257,55	EUR	BančnaKartica	257,55	Ne	50
20-jan-09	HU	Špedicija	3.600,00	HUF	Gotovina	12,07	Ne	51
20-jan-09	UA	Tranzit	101,50	UAH	Gotovina	9,33	Ne	52
23-jan-09	UA	Gorivo	1.156,00	UAH	BančnaKartica	106,29	Ne	38
23-jan-09	UA	Parkiranje	20,00	UAH	Gotovina	1,84	Ne	53
23-jan-09	UA	Brez računov	1.000,00	UAH	BančnaKartica	91,94	Ne	54
24-jan-09	RUS	Gorivo	15.100,00	RUB	BančnaKartica	341,90	Ne	39
25-jan-09	RUS	Parkiranje	300,00	RUB	Gotovina	6,79	Ne	56
26-jan-09	RUS	Hrana	661,00	RUB	Gotovina	14,74	Ne	55
26-jan-09	RUS	Vzdrževanje	6.100,00	RUB	BančnaKartica	138,12	Ne	60
28-jan-09	RUS	Telefon	860,00	RUB	Gotovina	19,17	Ne	57
28-jan-09	RUS	Brez računov	3.800,00	RUB	Gotovina	84,72	Ne	66
30-jan-09	RUS	Gorivo	13.687,00	RUB	BančnaKartica	309,91	Ne	40
30-jan-09	RUS	Vzdrževanje	1.500,00	RUB	Gotovina	33,44	Ne	61
31-jan-09	RUS	Ostalo	1.000,00	RUB	Gotovina	22,30	Ne	58
31-jan-09	UA	Tranzit	17,10	UAH	Gotovina	1,57	Ne	59
8-feb-09	UA	Parkiranje	60,00	UAH	Gotovina	5,52	Ne	62
8-feb-09	UA	Brez računov	895,00	UAH	Gotovina	81,54	Ne	67
9-feb-09	HU	Cestnina	2.760,00	HUF	Gotovina	9,26	Ne	63
9-feb-09	HU	Parkiranje	500,00	HUF	Gotovina	1,68	Ne	64
9-feb-09	SLO	Cestnina	16,10	EUR	BančnaKartica	16,10	Da	65

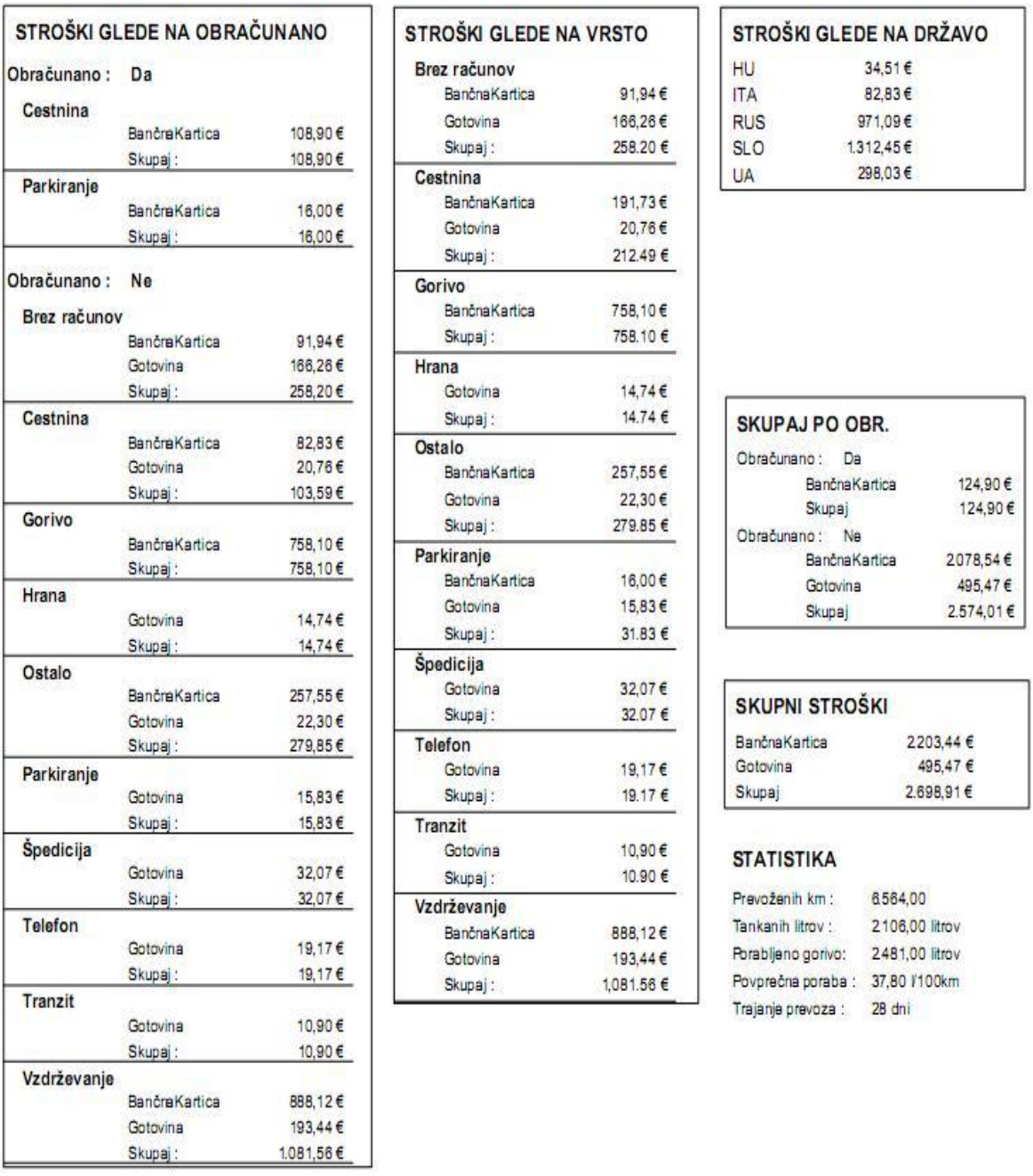
TANKANJE

DATUM	KRAJ	STEVEC	LITRI	ZNESEK	VALUTA	PLAČILO	ID
23-jan-09	UA	55.140	200,00	1.156,00	UAH	BančnaKartica	9
24-jan-09	RUS	55.521	1.000,00	15.100,00	RUB	BančnaKartica	11
30-jan-09	RUS	57.579	906,00	13.687,00	RUB	BančnaKartica	12

DOVOLILNICE

VRSTA	ŠTEVILKA
UKRAINA	243988
RUSIA	085333

Slika 16: Prva stran poročila



STROŠKI GLEDE NA OBRAČUNANO

Obračunano :	Da	
Cestnina		
	BančnaKartica	108,90 €
	Skupaj :	108,90 €
Parkiranje		
	BančnaKartica	16,00 €
	Skupaj :	16,00 €
Obračunano :	Ne	
Brez računov		
	BančnaKartica	91,94 €
	Gotovina	166,26 €
	Skupaj :	258,20 €
Cestnina		
	BančnaKartica	82,83 €
	Gotovina	20,76 €
	Skupaj :	103,59 €
Gorivo		
	BančnaKartica	758,10 €
	Skupaj :	758,10 €
Hrana		
	Gotovina	14,74 €
	Skupaj :	14,74 €
Ostalo		
	BančnaKartica	257,55 €
	Gotovina	22,30 €
	Skupaj :	279,85 €
Parkiranje		
	Gotovina	15,83 €
	Skupaj :	15,83 €
Špedicija		
	Gotovina	32,07 €
	Skupaj :	32,07 €
Telefon		
	Gotovina	19,17 €
	Skupaj :	19,17 €
Tranzit		
	Gotovina	10,90 €
	Skupaj :	10,90 €
Vzdrževanje		
	BančnaKartica	888,12 €
	Gotovina	193,44 €
	Skupaj :	1.081,56 €

STROŠKI GLEDE NA VRSTO

Brez računov	
BančnaKartica	91,94 €
Gotovina	166,26 €
Skupaj :	258.20 €
Cestnina	
BančnaKartica	191,73 €
Gotovina	20,76 €
Skupaj :	212.49 €
Gorivo	
BančnaKartica	758,10 €
Skupaj :	758.10 €
Hrana	
Gotovina	14,74 €
Skupaj :	14.74 €
Ostalo	
BančnaKartica	257,55 €
Gotovina	22,30 €
Skupaj :	279.85 €
Parkiranje	
BančnaKartica	16,00 €
Gotovina	15,83 €
Skupaj :	31.83 €
Špedicija	
Gotovina	32,07 €
Skupaj :	32.07 €
Telefon	
Gotovina	19,17 €
Skupaj :	19.17 €
Tranzit	
Gotovina	10,90 €
Skupaj :	10.90 €
Vzdrževanje	
BančnaKartica	888,12 €
Gotovina	193,44 €
Skupaj :	1,081.56 €

STROŠKI GLEDE NA DRŽAVO

HU	34,51 €
ITA	82,83 €
RUS	971,09 €
SLO	1.312,45 €
UA	298,03 €

SKUPAJ PO OBR.

Obračunano :	Da	
	BančnaKartica	124,90 €
	Skupaj	124,90 €
Obračunano :	Ne	
	BančnaKartica	2.078,54 €
	Gotovina	495,47 €
	Skupaj	2.574,01 €

SKUPNI STROŠKI

BančnaKartica	2.203,44 €
Gotovina	495,47 €
Skupaj	2.698,91 €

STATISTIKA

Prevoženih km :	6.564,00
Tankanih litrov :	2.106,00 litrov
Porabljeno gorivo:	2.481,00 litrov
Povprečna poraba :	37,80 l/100km
Trajanje prevoza :	28 dni

Slika 97: Druga stran poročila

7. TESTIRANJE

Testiranju, oziroma delovanju aplikacije sem se že posvečal v sami izdelavi programa in ob vsaki spremembi. Vsak dodatek (modul), ki ga dodamo pri programiranju, lahko vpliva na nek drug del kode in s tem povzroči nepravilno delovanje samega programa. Iz tega razloga sem ob vsakem dodanem sklopu preveril vse ostale sklope, na katere se je le ta nanašal. Poleg tega sem sproti preverjal, če se v podatkovni bazi vse pravilno izvaja.

Končno testiranje je potekalo štiri dni. Večinoma sem to izvajal sam, tako da sem vnašal razne izmišljene podatke in spremljal razlike v zneskih, ter se tako prepričal, če vse funkcije pravilno delujejo. Za tem sem vnašal nesmiselne stvari, ter tako preveril, če so opozorila na napake pravilno zastavljena in če nam le ta dovolj jasno povedo, kaj je potrebno spremeniti.

Naslednja faza je potekala tako, da sem izbral končnega uporabnika, ki je testiral program. V tem času sem spremljal, kako je oseba uporabljala program in sproti ugotavljal manjše potencialne popravke, v trenutkih, ko so se nekatere stvari uporabniku zdele dvoumne. Poleg tega je uporabnik še izrazil posebne želje o postavitvi komponent, katere sem hitro izvršil.

V zadnji fazi sem še poskušal pospešiti samo delovanje programa z odpravo nepotrebne kode, ali s spreminjanjem posameznih procedur pri izdelavi in shranjevanju poročil. S tem sem zaključil fazo testiranja, ki je prinesla nekaj sprememb, oziroma popravkov, ki so sam program nekoliko izboljšali in skrajšali čas delovanja.

S tem je bil program pripravljen za namestitev in začetek uporabe v praksi.

8. DOKUMENTACIJA

Pred vpeljavo programa sem se lotil še priprave dokumentacije, oziroma navodil za uporabo. Dokumentacija je namenjena vsem uporabnikom, ki bodo upravljali s programom. Opredeljen je način dela s programom, vse skupaj je obogateno s slikami, ki uporabniku pomagajo pri samem delu. Vsebuje pa tudi razlago nekaterih rešitev, predvsem tistih, ki se navezujejo na računovodstvo. Natančno so opredeljeni odseki poročila, ter naštete vrste in podvrste stroškov, ki se nanašajo na določen odsek.

V nadaljevanju poglavja bom prikazal le del dokumentacije in s tem razložil način izdelave.

8.1. Navodila za uporabo

1. ODPIRANJE NOVEGA POTEGA NALOGA

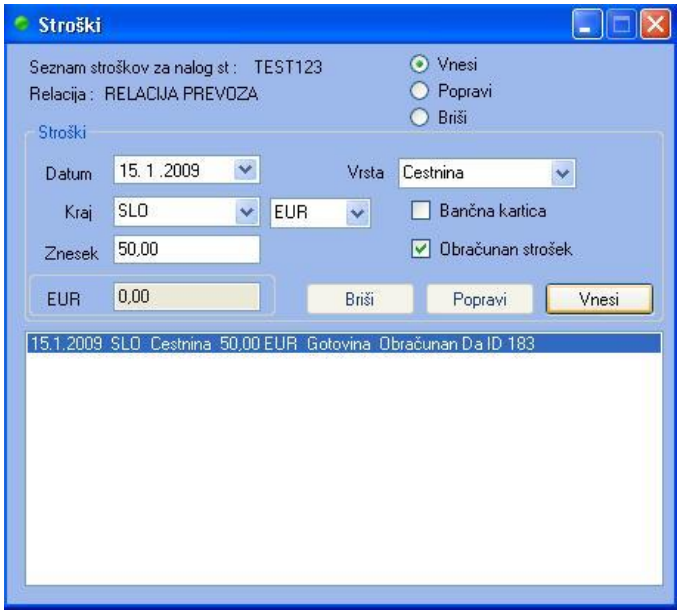
- Za odpiranje novega potnega naloga izberite iz menija »Datoteka«, nato kliknite »Nov nalog«.

- Nato vnesite vse potrebne podatke in pritisnite gumb »Vnesi«. S tem ste odprli nov potni nalog.

- Sedaj se vam je nalog prikazal na spodnjem seznamu in s tem je pripravljen na vnašanje podatkov.

2. VNAŠANJE STROŠKOV

Po uspešno odprtem potnem nalogu kliknite na gumb »Stroški« in odprl se vam bo obrazec za vnašanje stroškov.

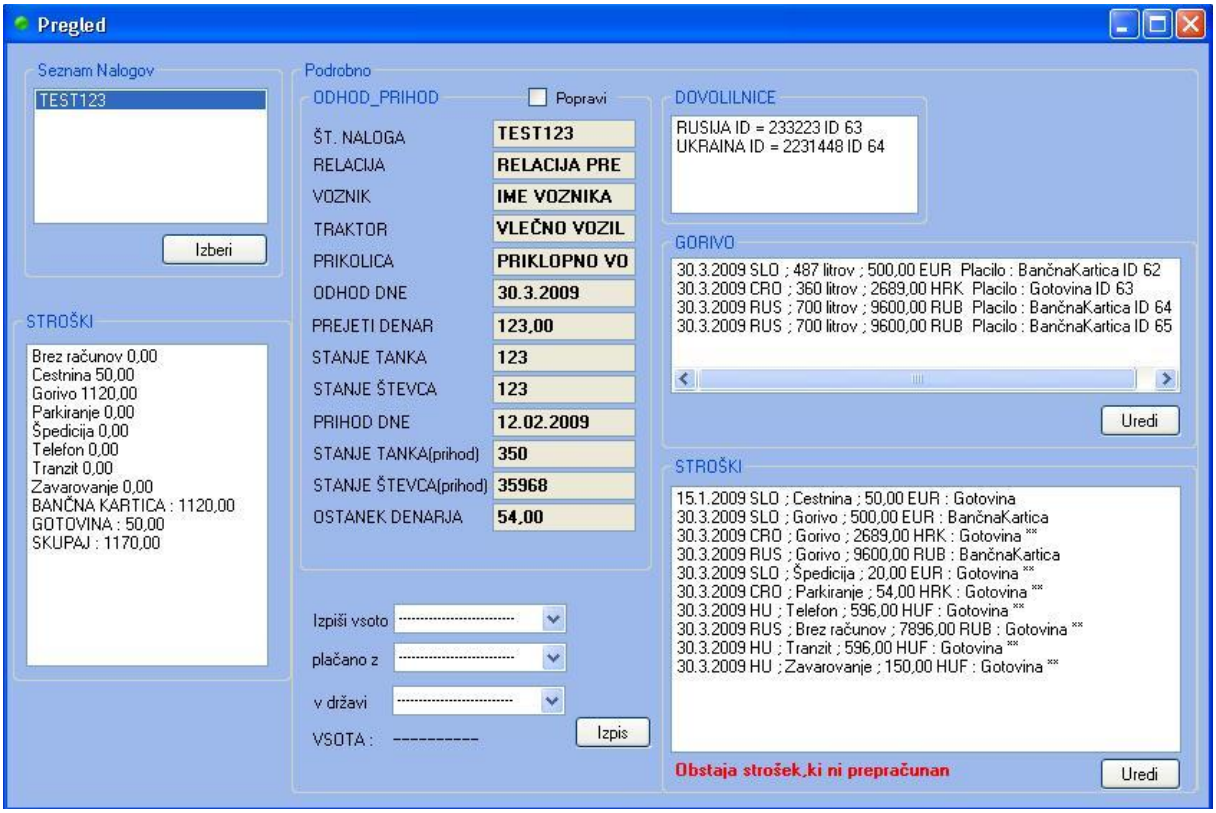


Na obrazcu izpolnite vse potrebne podatke ter pritisnite tipko »Vnesi«. S tem se bo vaš strošek pojavil na spodnjem seznamu. V zgornjem desnem kotu imate na voljo tri možnosti upravljanja s stroški (»Vnesi«, »Popravi«, »Briši«). S klikanjem na te možnosti se bodo aktivirali pripadajoči gumbi, tako da lahko stroške brišete, oziroma popravljate.

Enak način dela velja pri vnašanju dovolilnic in pri gorivu.

3. PREGLED NALOGA

Ko ste vnesli vse potrebne podatke o potnem nalogu, lahko kliknete v meniju na »Uredi«, ter »Preglej/Popravi nalog«. Odpre se vam pogovorno okno, v katerem imate na voljo hiter pregled naloga z vsemi podatki. Tu vas program opozarja, če ste na kaj pozabili, kot na primer na vnos tečajev in podobno. Sporočila se vam izpisujejo v spodnjem desnem kotu z rdečo barvo. Napake so lahko povezane z obračunanimi stroški (ti so lahko le v Sloveniji), s tečaji, tečaji bančne kartice …



Program vas bo opozarjal na napake tako dolgo, dokler ne bo spodaj pisalo »OK« z zeleno barvo in šele takrat bo nalog pripravljen za tiskanje.

Tu imate na voljo tudi izpis posameznih zneskov. Na voljo imate tri vnosna polja in tipko »Izpis«. To funkcijo uporabljate preprosto zato, da izberete katerokoli možnost in kliknete na gumb. Če boste vsa polja pustili prazna, vam bo izpisal vsoto celotnih stroškov prevoza. Če pa recimo izberete samo »plačano z bančno kartico«, pa boste dobili vsoto porabe bančne kartice.

8.2. Razlaga rešitev

V tem delu dokumentacije vam predstavljam razlago rešitev in vseh odsekov poročila. Namen tega je predvsem v razumevanju posameznih rešitev ter zneskov, ki jih boste zasledili v poročilu.

ODSEKI POROČILA

Zadnja stran poročila je sestavljena iz šestih odsekov:

- STROŠKI GLEDE NA OBRAČUNANO
- STROŠKI GLEDE NA VRSTO
- STROŠKI GLEDE NA DRŽAVO
- SKUPAJ PO OBRAČUNU
- SKUPNI STROŠKI
- STATISTIKA

Prvi odsek poročila vam razdeli stroške glede na obračunljivost. Odsek je razdeljen na dva dela, in sicer na »Obračunano: Da« in »Obračunano: Ne«. Prvi del predstavlja vse vrste stroškov, ki so nastali v Sloveniji, in sicer glede na način plačila. Za vsako vrsto stroška vam navaja, ali je bil plačan z gotovino ali z bančno kartico, ter vsoto posamezne vrste. Drugi del tega odseka pa navaja enak opis, vendar za vse tiste stroške, ki **niso** nastali v Sloveniji.

V drugem odseku so stroški porazdeljeni glede na vrsto. To pomeni, da imate tu seznam vseh vrst stroškov in skupne zneske le teh. Kakor pri prvem odseku so tudi tu porazdeljeni glede na način plačila. Razlika je v tem, da se v tem odseku stroški ne delijo glede na to, ali so nastali v Sloveniji ali ne.

Tretji odsek navaja porabo denarja v posamezni državi v času prevoza. Vsi zneski so preračunani v evre, in sicer natančno na dve decimalki.

Četrti odsek ponazarja celotno vsoto stroškov glede na obračunljivost, porazdeljeni so glede na način plačila. Torej to so skupni stroški v Sloveniji in izven nje.

Peti odsek predstavlja celotne stroške prevoza, razdeljene glede na način plačila.

V šestem odseku pa so navedeni statistični podatki prevoza, in sicer: prevoženi kilometri, koliko litrov goriva je bilo natočenih med prevozom, koliko litrov goriva je bilo porabljenega, povprečna poraba goriva in trajanje prevoza.

9. VPELJAVA PROGRAMA

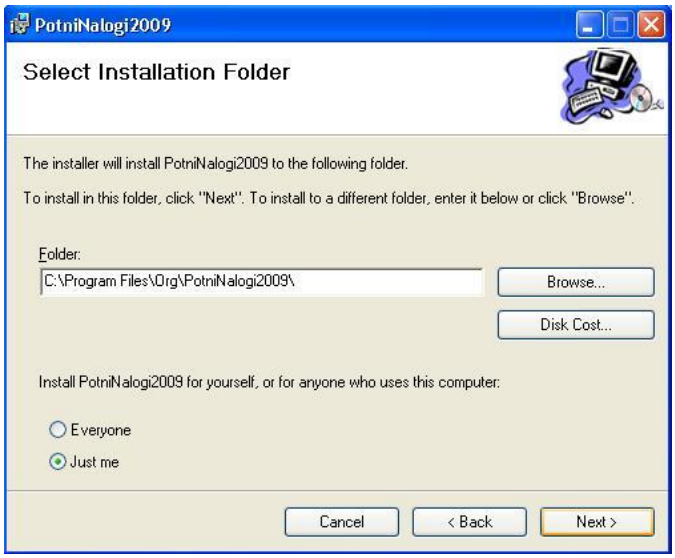
Pred samo namestitvijo progama sem se še dokončno prepričal o pravilnosti delovanja. Bodočega uporabnika sem prosil, da v novi program vnese resnične podatke iz starejših (končanih) nalogov, in sicer z namenom, da se prepričam, če vse funkcije pravilno delujejo. Zanimale so me samo končne vsote stroškov. Pričakoval sem manjša odstopanja (2-3 EUR), in sicer iz razloga, ker bodoči program opravi natančnejši preračun valut. Rezultati so pokazali natančno tisto, kar sem pričakoval in tako sem začel z namestitvijo programa.

9.1. Pakiranje datotek

Za izdelavo »Setup« datoteke sem uporabil Microsot-ovo orodje »Setup Wizard«, ki se nahaja v sklopu Visual Studia. Delo s tem orodjem je dokaj preprosto, saj je program po vnosu EXE datoteke (bodočega programa) v končni paket sam zaznal vse potrebne odvisnosti in vnesel vse dinamične knjižnice (DLL-je), ki so bile potrebne za delovanje programa na ciljnem računalniku. Sledilo je le še nekaj nastavitev o ciljni lokaciji namestitve in poimenovanje posameznih datotek. S tem je bil program pripravljen za namestitev na ciljni računalnik.

9.2. Namestitev

Po izdelavi namestitvene datoteke sem jo zagnal na ciljnem računalniku. Po zagonu me je le ta opozoril, da na računalniku ni nameščena potrebna verzija .NET Framework-a. Potrebno datoteko je avtomatsko prenesel na računalnik, ter jo namestil. Sledilo je še nekaj klikov in program se je uspešno namestil in s tem je bilo vse pripravljeno za njegovo uporabo.



Slika 10: Namestitev programa

10. ZAKLJUČEK

10.1. Rezultati izbranega pristopa

Strukturni pristop k razvoju informacijskih sistemov je najstarejši, vendar še vedno pogosto uporabljan proces razvoja informacijskih sistemov. Zgleduje se po standardnih postopkih razvoja tehničnih izdelkov, pri katerih si **opravila v okviru aktivnosti sledijo v zaporedju**.

V mojem projektu se je izbrana metodologija obnesla odlično, saj sem v vsakem trenutku vedel, kaj je potrebno storiti in kaj bo za tem sledilo. Izvedba projekta je bila še lažja, ker IE podrobno opisuje analizo, ter fizično in logično načrtovanja in ponuja še tehnike dela. Poleg tega so bili izdelki analize (logični podatkovni model, zajem zahtev) in načrtovanja (fizični podatkovni model, diagrami podatkovnih tokov) še kako koristni pri poteku celotnega projekta.

Sama metodologija je načrtovana za zelo obsežne projekte. Zaradi tega sem se moral omejiti le na korake, ki so pri mojem načrtovanju izboljšali sam proces in ga niso zavirali.

Opazil sem tudi eno pomanjkljivost, in sicer, celotna metodologija ne opisuje vzdrževanja informacijskih sistemov, ampak zgolj omeni to področje, in mislim, da bi avtorji metodologije lahko temu posvetili malo več pozornosti.

10.2. Rezultati diplomskega dela

V sklopu diplomskega dela je bil uresničen cilj, ki je bil zastavljen v drugem poglavju. Nastal je uporabniku prijazen program, ki je enostaven za uporabo, predvsem pa izpolnjuje vse zastavljene uporabniške zahteve.

Pri poteku celotnega projekta sem naletel na številne težave, predvidljive in nepredvidljive. Razlog za to je bil predvsem v pomanjkanju izkušenj pri tovrstnih projektih, saj sem prvič samostojno razvil program, ki se bo dejansko uporabljal v praksi v koristne namene. Vse težave sem seveda dokaj hitro odpravil in s tem pridobil nove izkušnje, ki mi bodo v prihodnosti še kako koristile.

Lepa je tudi ta izkušnja, da sem se prav tako prvič soočil z uporabo metodologije razvoja **v praksi**. Omogočala mi je dokaj lepo, tekoče izpeljati celoten projekt, ter mi posredovala veliko novih izkušenj in znanja. Vesel sem, da sem teorijo uporabil v praksi in s tem spoznal, kako so vse faze razvoja ter posamezni koraki pomembni pri tovrstnih projektih. V spominu mi je najbolj ostala faza analize, saj sem tu podrobno spoznal zastavljen problem, ki sem ga nato postopoma reševal.

Diplomsko delo v končni fazi obsega ne le končni produkt, ampak zajema tudi nepozabno izkušnjo in dopolnitev znanja, ter pomoč pri bodočih projektih, za katere verjamem, da bodo dokaj tekoče izpeljani, saj sedaj razumem, kako poteka projekt in na kaj moram biti še posebno pozoren.

11. VIRI

Literatura:

Watson Nagel, Pedersen Reid, Skinner White »Visual C# 2008«, 2008

Nick Randolph, David Gardner »Professional Visual Studio 2008«, 2008

Spletni viri:

Wikipedia, »Crystal Reports«, dostopno na: http://en.wikipedia.org/wiki/Crystal_Reports

Wikipedia, »Dot net framework«, dostopno na: http://en.wikipedia.org/wiki/Dot_net_framework

Izum.si, »Ogrodje.Net«, dostopno na:

http://home.izum.si/cobiss/cobiss_obvestila/2001_2/Html/clanek_04.html

Informacijski inženiring, » Metodologija IE«, dostopno na:

http://colos.fri.uni-lj.si/ERI/RACUNALNISTVO/INFORMATIKA/informacijski_ineniring.html

Sicom, »SQL Server 2005«, dostopno na:

http://www.sicom.si/SQLServer70.htm

Business Objects, »Crystal Reports 2008«, dostopno na:

http://www.businessobjects.com/product/catalog/crystalreports/

Visual C#.NET, »Razlogi uporabe C#«, dostopno na:

http://www.microsoft.com/slovenija/msdn/csharp/razlogi.mspx

Pikanet, »Ogrodje Microsoft .Net« dostopno na:

http://lisa.uni-mb.si/~juric/pikanet.pdf