

MATEMATIČKI FAKULTET
BEOGRADSKI UNIVERZITET

MASTER TEZA

Aplikacija za vođenje evidencije prihoda i rashoda
zasnovana na .NET tehnologiji

Vesna Kaplarević

Mentor: prof. Dušan Tošić

Beograd, Jun 2009

Sadržaj

Predgovor	3
Uvod u finansijski softver	4
1.0 Opis Aplikacije	5
1.1 Instalacija aplikacije	5
1.2 Uputstvo za korišćenje aplikacije	8
2.0 Korišćene Tehnologije i Implementacija Rešenja	17
2.1 SQL Server 2005 Express Izdanje	17
2.2 .Net Framework 3.5 – LINQ (Language Integrated Query)	18
2.2.1 Proširenje C# jezika	18
2.2.2 LINQ	19
Zaključak	32
Literatura	33

Predgovor

Master teza pod naslovom „Aplikacija za vođenje evidencije prihoda i rashoda zasnovana na .Net tehnologiji“ predstavlja primer pojednostavljenog finansijskog softvera. Pomoću nje se na vrlo jednostvan način mogu obavljati osnovne finansijske transakcije evidentiranja svih prihoda i rashoda, kao i pratiti sve promene u krajnjem finansijskom bilansu. Takođe, u svakom trenutku je moguće sačuvati finansijski izveštaj radi eventualne dalje analize kao poseban Excel dokument.

Cilj ovog rada jeste da pruži smernice za razvoj poslovnih aplikacija, bez prethodnog računovodstveno-finansijskog obrazovanja, korišćenjem Microsoft .Net Framework 3.5 i Visual Studio 2008 okruženja. Stoga posebno može biti zanimljiv studentima kao i svim zainteresovanima za izučavanje novih tehnologija.

Polazeći od toga ovaj rad je na odgovarajući način strukturiran – komponovan od dva dela :

1. Opis aplikacije
2. Opis korišćenih tehnologija i implementacija rešenja

U prvom delu, nakon objašnjenja kako se instalira aplikacija pomoću instalacionog čarobnjaka, veći deo posvećen je detaljnom opisu korišćenja aplikacije, što ujedno može poslužiti krajnjim korisnicima i kao uputstvo za upotrebu aplikacije.

Drugi deo se više odnosi na opis korišćenih tehnologija. Dotaknuta su osnovna obeležja sistema za upravljane relacionom bazom podataka – SQL Server 2005 Express Edition, dok je najviše pažnje posvećeno .Net Framework-u 3.5 i LINQ-u(Language Integrated Query -ju), kao njegovoj najbitnijoj komponenti.

U ovom potpoglavlju data je podela LINQ-a u podgrupe i objašnjene su njihove specifičnosti i razlike. Više pažnje je posvećeno delu LINQ To SQL uz propratni kod – korišćen u samoj implementaciji. Dublje je analiziran njen doprinos programiranju i napravljena je paralela sa različitim pristupima u manipulaciji podataka korišćenim u prethodnim verzijama .Net Framework-a.

Uvod u finansijski softver

Uspešno poslovanje kako manjih preduzeća, tako i većih kompanija bilo bi nemoguće postići bez detaljne i sistematične analize prihoda i rashoda u budžetu, kao i faktora koji na njih utiču. Informativnu podršku upravi svakog preduzeća pruža računovodstvo.

Računovodstvo predstavlja nezamenljiv način upravljanja i rukovođenja modernim preduzećem.

Iako ne postoji univerzalna definicija računovodstva, savremeno računovodstvo bi se moglo definisati kao osnovna informaciona delatnost u preduzeću, i o preduzeću. Može se sagledati i kao skup sinhronizovanih i međusobno usklađenih metoda putem kojih se obezbeđuje evidentiranje poslovnih događaja, kao i njihova analiza i kontrola.

Osnovni zadatak računovodstvene funkcije je da, u skladu sa zakonom, prikuplja, obrađuje i distribuira računovodstvene informacije u okviru, ali i izvan preduzeća.

Redovni, kao i neposredni prihodi i rashodi budžeta, predstavljaju osnovne elemente krajnjeg finansijskog rezultata. Budžet se može definisati kao finansijski plan za određeno vreme.

Posebno treba istaći ERP sisteme koji, najkraće rečeno, prate sve aspekte poslovanja jedne kompanije. Oni realizuju, ili bolje rečeno, omogućavaju integraciju kompletnog funkcionisanja poslovnog sistema pomoću jedinstvenog softverskog rešenja.

ERP je akronim od „**Enterprise Resource Planing**“, odnosno „**Poslovni Informacioni Sistem**“. Ova finansijska rešenja obuhvataju sve standardne poslovne funkcije i imaju mogućnost prilagođavanja konkretnim potrebama preduzeća, podržavajući međunarodne standarde (tzv. implementacija softverskog rešenja).

Naime, implementirani ERP sistem je u mogućnosti da integriše poslovanje različitih delova firme (kao npr. računovodstvo, prodaja, proizvodnja, itd.) u jednu jedinstvenu celinu. Tako se dobija sistem preko kojeg je moguće sa jedne strane upravljati svim ljudskim i materijalnim resursima, a sa druge planirati, razvijati i pratiti poslovne procese i procedure. Dužina procesa implementacije ERP softvera je različita za svaku kompaniju - zavisi kako od same veličine firme, tako i od broja modula koji se implementiraju. U toku procesa podešavanja i integracije vrše se modifikacije ERP sistema koje će zadovoljiti specifične potrebe svake industrije.

Postoji pet glavnih razloga zašto se firme odlučuju na implementaciju ERP sistema, a to su : integrisani finansijski podaci, integrisani podaci o nalogima kupaca, standardizacija informacija o ljudskim resursima, standardizacija i ubrzanje procesa proizvodnje i smanjenje lagera.

Danas postoji veliki broj kompanija koje proizvode poslovni softver, među kojima su vodeće:

- SAP("Systems, Applications and Products in data processing"),
- Microsoft,
- IBM,
- ORACLE

U ogromnoj paleti njihovih proizvoda, koje nude globalnom tržištu, nalazi se i finansijski softver koji u mnogome olakšava računovodstveni proces svakog preduzeća.

Među najpoznatije komercijalne ERP softverske pakete spadaju :

- Proizvodi kompanije SAP koji su stekli gotovo kulni status u svetu ERP rešenja. Najpoznatija ERP softverska rešenja su „**mySAP Business Suite**” za velike kompanije i „**SAP Business One**” za mala i srednja preduzeća. Više informacija o proizvodima ove kompanije može se naći na sledećoj internet adresi - www.sap.com.

- Microsoft (www.microsoft.com) je na tržištu ERP softvera prisutan kroz liniju poznatu pod imenom „**Microsoft Dynamics**” (ranije poznata kao Microsoft Business Suite) u čijem su sastavu sledeći ERP paketi:

- MS Dynamics NAV (ranije Navision) - za male i srednje firme, sadrži veliki broj delatnosti, sa naglaskom na kvalitetnim finansijama
- MS Dynamics AX (ranije Axapta) - za velike (i veće srednje) firme, kompletno rešenje (za sve delatnosti) sa izvanrednom proizvodnjom
- MS Dynamics SL (ranije Solomon) - za manje i srednje firme, sa posebnim naglaskom na vođenju projekata
- MS Dynamics GP (ranije Great Plans) - sličan NAV-u, ali je rađen za USA tržište

-Takodje manje IT firme i kompanije plasiraju na tržište svoja finansijska rešenja koja ništa manje nisu uspešna u zadovoljavanju i postizanju istih ciljeva. Naravno, svaki od tih proizvoda se može prilagoditi potrebama klijenata u okviru zakona.

Ovaj master rad, „Aplikacija za vođenje evidencije prihoda i rashoda“, predstavlja upravo osnovu jedne takve aplikacije realizovane u .NET tehnologiji, primenjivu u manjim preduzećima ili čak domaćinstvima, koja se može dalje nadograđivati u zavisnosti od samog poslovanja i potreba.

Kao što je naglašeno u predgovoru, kroz sledeća dva poglavlja biće detaljno opisano kako se koristi aplikacije, njena implementacija i korišćene tehnologije.

1.0 Opis Aplikacije

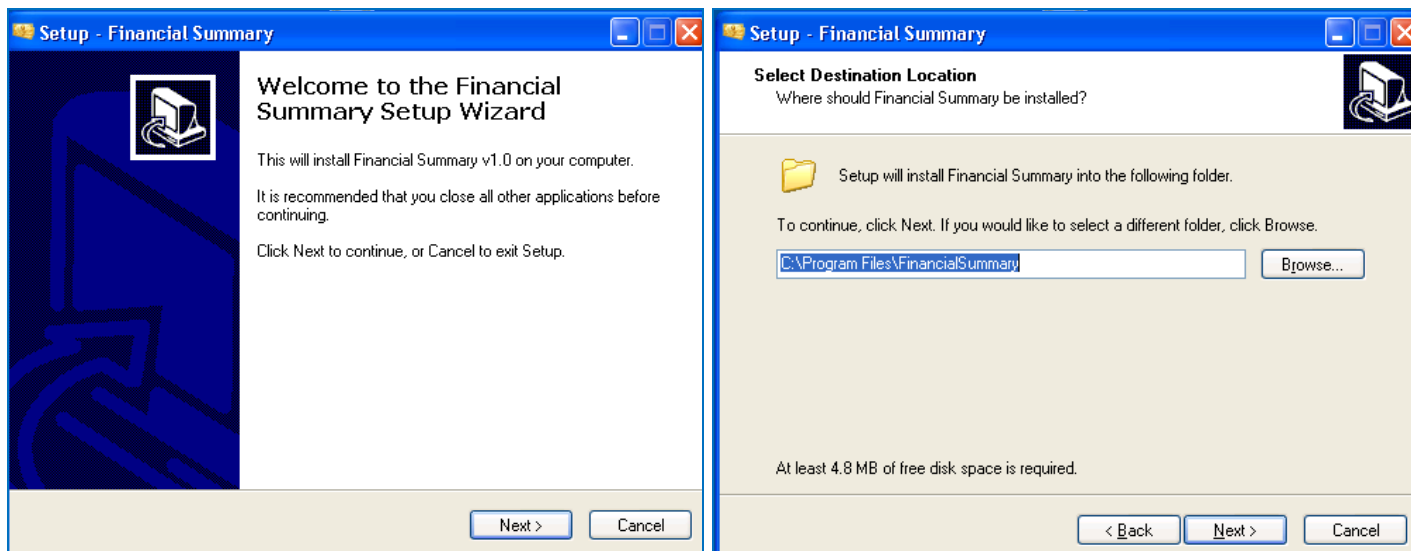
Osnovna namena aplikacije jeste da se omogući vođenje evidencije prihoda i rashoda i na taj način omogući korisniku da u svakom trenutku ima uvid u količinu novca kojim raspolaže. Pored upisivanja svih dnevnih troškova, mesečnih računa, plata i drugih primanja, dostupni su i pregledi protoka novca po različitim kriterijuma što omogućava korisnicima detaljan uvid u to na šta i na koji način troše novac. Detaljniji opis mogućnosti aplikacije se nalazi u drugom potpoglavlju 1.2.

Ovo poglavlje se može podeliti u dve celine :

- Instalacija Aplikacije – pokretanjem instalacionog čarobnjaka
- Uputstvo za korišćenje aplikacije

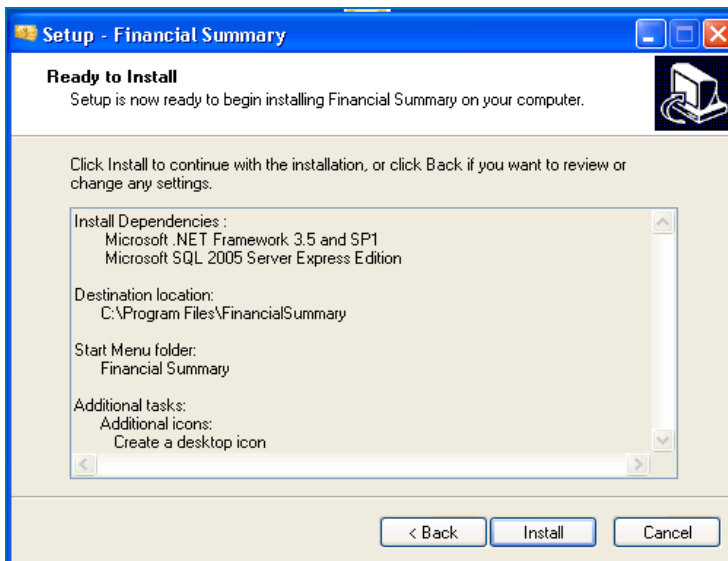
1.1 Instalacija Aplikacije

Pokretanjem FinancialSummary.exe fajla startuje se čarobnjak za instalaciju koji vodi korisnika kroz sam proces.



U prvom koraku se proverava da li je instaliran operativni sistem Windows XP SP2 i ukoliko jeste instalacija se nastavlja i dalje se proverava, na osnovu vrednosti odgovarajućih registara, da li je instaliran .NET Framework 3.5, .NET Framework 3.5 SP1 i SQL Server 2005 Express izdanje o kojima će biti nešto više reči u sledećem poglavlju. Bitno je napomenuti da, ukoliko njihova instalacija postoji, neće biti navedeni u listi proizvoda čija instalacija je neophodna za rad aplikacije.

Na sledećoj slici prikazan je prozor našeg instalera na kom je prikazan spisak svih programa čija je instalacija neophodna za pokretanje aplikacije :

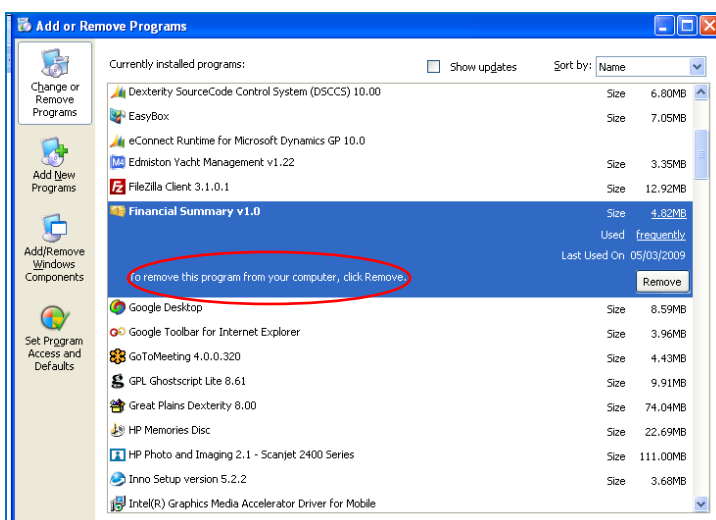


Takođe, bitno je napomenuti da se njihova instalacija, ukoliko je potrebna, izvršava u tihom modu, tj. korisnik ne mora da podešava parametre i unosi podatke što bi bio slučaj pri standardnoj instalaciji. Ukoliko ipak postoji razlog zbog kojeg korisnik želi da ih svojeručno unese, na CD-u koji je priložen uz ovu master tezu, nalaze se njihovi instalacioni exe fajlovi. Inicijalna instalacija, zbog .NET Framework-a i SQL Express Server-a, oduzima malo više vremena nego u slučaju kada oni već postoje.

Nakon što se instalira SQL Server, pokreće se deo aplikacije za kreiranje baze podataka i inicijalno punjenje baze podacima neophodnim za pokretanje aplikacije, kao što su valute i njihovi ISO kodovi, osnovne kategorije troškova, registruje se Admin korisnik sa administratorskim pravima pristupa aplikaciji čije korisničko ime i šifra se mogu koristiti kako bi se registrovali ostali korisnici. U sledećem potpoglavlju biće detaljnije objašnjen sam način korišćenja aplikacije.

Po završetku instalacije kreiraće se prečica na desktopu kao i stavka u listi programa u start meniju pomoću kojih se može pokrenuti aplikacija.

Deinstalacija programa je moguća iz Control Panel-a klikom na ime aplikacije u „Add or Remove Programs“ listi.



1.2 Uputstvo za korišćenje aplikacije

Aplikacija je realizovana tako da se može koristiti u tri režima, tačnije, postoje tri nivoa prava pristupa aplikaciji kroz koje se realizuje bezbednost podataka u njenom korišćenju.

To su :

- Administrator – sa apsolutnim pravima
- Power User – napredni režim, u kojem korisnik ima manja prava od administratora
- User – osnovni režim u kojem korisnik može samo unositi svoje prihode i rashode i praviti izveštaje vezane samo za njih.

Po pokretanju aplikacije prikazaće se prozor za logovanje korisnika sa odgovarajućim pravima :



U nastavku, korišćenje aplikacije će biti objašnjeno pod pretpostavkom da je ulogovan administrator korisnik i na taj način će se proći kroz najvažnije opcije aplikacije, a zatim će biti objašnjeno kojim delovima imaju pristup korisnici sa manjim pravima.

Pri prvom pokretanju koristi se korisničko ime i šifra korisnika sa administratorskim pravima pristupa aplikaciji, koji je registrovan prilikom instalacije :

Korisničko ime : admin

Korisnička šifra : FS_Admin

Da bi pristupio aplikaciji korisnik, koji je prethodno registrovan, mora izabrati valutu koja će biti osnovna pri unosu svih novčanih transakcija.

Meni pri samom vrhu prozora sadrži sledeće elemente dostupne korisniku :

- Income – unos prihoda
- Outcome – unos rashoda
- Category – pregled kategoria troškova
- Report – izveštaji
- Money Exchange – setovanje vrednosti kursa valuta različitih od izabrane osnovne
- Admin – registrovanje novih korisnika, kategorija troškova i njihova aktivacija i deaktivacija, pravljenje rezervne kopije baze, kao i eksportovanje transakcija u Excel dokument.

Prvo će biti objašnjen Admin deo(Administracija), pošto je potrebno prvo podesiti aplikaciju i registrovati ostale korisnike.

Admin (Administracija)

Kao što je već rečeno, u administrativnom delu aplikacije mogu se registrovati novi korisnici tako što im se dodeli određeno pravo pristupa, korisničko ime i šifra. Dalje, mogu se registrovati nove kategorije troškova ili aktivirati/deaktivirati već postojeće, napraviti rezervna kopija baze (backup fajl) i exportovati transakcije u Excel fajl.

Jedino administrator može pristupiti administrativnom delu.

U nastavku ćemo opisati detaljnije meni administrativnog dela aplikacije.

User (Korisnik) :

Svaki korisnik je jedinstven prema dodeljenom korisničkom imenu.

Editovanje imena korisnika nije moguće jedino u slučaju kada korisnik sa novim izabranim imenom već postoji. Ukoliko za korisnika, čije ime ili rola se menja, već postoje unete transakcije, automatski će se izmeniti i njihovi detalji u smislu da će se odnositi na istog korisnika samo će njegovo ime ili rola pretrpeti izmene.

Brisanje korisnika nije dozvoljeno ukoliko postoje bilo kakve novčane transakcije vezane za njega.

USERNAME	ROLE	PASSWORD
admin	Administrator	FS_Admin
admin2	Administrator	Admin2@123
admin3	Administrator	Admin3@123
Mark	Power User	Mark@123
Peter	Power User	Peter@123
Sam	User	Sam@123

Category (Kategorije Troškova):

Prilikom unošenja transakcija rashoda, korisnik mora izabrati kojoj vrsti troškova, odnosno kategoriji, ona pripada. Na primer, to može biti plaćanje školovanja, različitih kurseva, putovanja, osiguranja, ali i svakodnevnih potreba kao što su hrana, piće, komunalije itd.

Postoje predefinisane kategorije i podkategorije kreirane pri instalaciji. Takođe, moguće je registrovati nove i postaviti njihov aktivan ili neaktivan status.

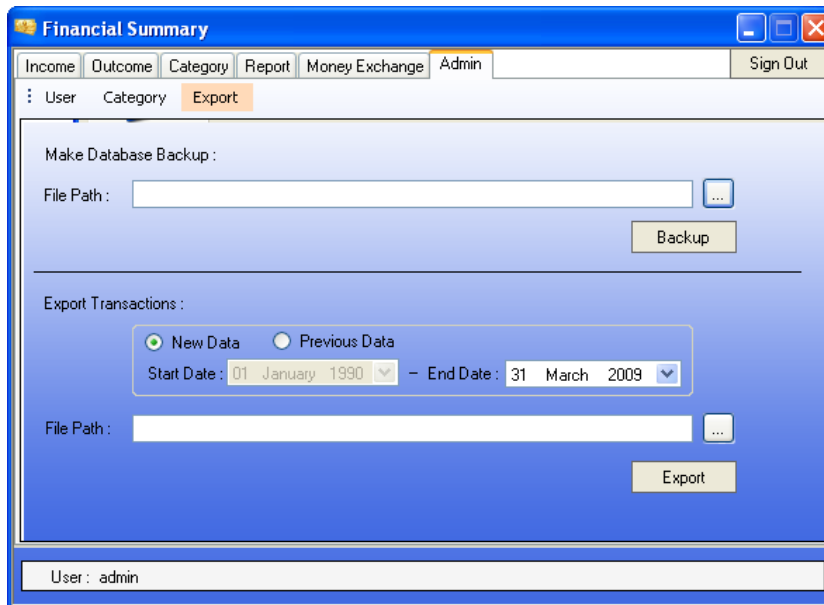
Editovanje kategorija je dozvoljeno samo za podkategorije. Slično editovanju korisnika, biće izmenjene i relevantne transakcije.

Brisanje je dozvoljeno samo ukoliko ne postoje transakcije, u suprotnom može se samo promeniti stanje u kojem se nalazi ordedena podkategorija na aktivno ili neaktivno stanje, što se na transakcije odražava tako da ukoliko je neaktivna – odgovarajuća transakcija se ne može ni editovati ni brisati dok se njeno stanje ne vrati na aktivno.

CATEGORY	SUBCATEGORY	DESCRIPTION	PREFIX	ACTIVE
EDUCATION	Membership & As...		EDC	<input checked="" type="checkbox"/>
EDUCATION	Training & Certific...		EDC	<input type="checkbox"/>
HEALTH	Beauty Services ...		HLT	<input checked="" type="checkbox"/>
HEALTH	Crew Medical		HLT	<input checked="" type="checkbox"/>
HEALTH	Dentist		HLT	<input checked="" type="checkbox"/>
HEALTH	Insurance		HLT	<input type="checkbox"/>

Export (Pravljenje Excel Izveštaja unetih transakcija i rezervne kopije baze podataka):

Export stavka podmenija Admin dela, prikazana na sledećoj slici, nudi korisniku mogućnost pravljenja rezervne kopije podataka, kao i ekportovanja novčanih transakcija, što će biti u nastavku objašnjeno.



- **Pravljenje rezervne kopije baze podataka („backup“ fajl)**
Pritiskom na dugme, pored teksta za putanju dokumenta, otvoriće se dijalog za čuvanje fajlova („Save File Dialog“), gde će korisnik moći da izabere ime i mesto čuvanja rezervne kopije baze (dokument sa ekstenzijom .bak). Inicijalno pri otvaranju ovog dijaloga za čuvanje fajlova, otvara se folder na putanji ~\My Documents\Financial Summary Files\Database Backups, što naravno ne obavezuje korisnika da sačuva fajl na predloženoj lokaciji.
Pritiskom na dugme za pravljenje rezervne kopije („Backup“) napraviće se .bak dokument sa prethodno izabranim imenom, koji će biti sačuvan na izabranoj putanji.
- **Eksportovanje novčanih transakcija**
Slično prethodnom, pritiskom na dugme za otvaranje dijaloga za čuvanje dokumenata, inicijalno će se otvoriti folder na putanji ~\My Documents\Financial Summary Files\Exports koji se može promeniti (služi samo kao preporuka lokacije gde bi mogli da se sačuvaju svi Excel fajlovi).
Kreira se Excel dokument sa više listova – prvi list sadrži sve transakcije vezane za sve korisnike, a zatim slede ostali za svakog korisnika posebno. Dalje se te eksportovane transakcije neće pojavljivati kao aktivne za editovanje i brisanje na Income (Prihodi) i Outcome (Rashodi) delu, jedino će se moći napraviti njihov izveštaj na Report (Izveštaj) prozoru.

Prilikom eksportovanja može se napraviti izbor između novih transakcija ili već eksportovanih u zavisnosti od potrebe korisnika. Bitno je spomenuti da će zaključni bilans, pri eksportovanju transakcija, biti početni za dalji unos transakcija. U Excel dokumentu taj preneti početni bilans se nalazi u polju „Balance Bought Fwd“ (Preneti Bilans).

U primeru exportovanog fajla, prikazanom na sledećoj slici, istaknuto je polje koje predstavlja preneti bilans kao i lista korisnika po kojima je napravljen zaključni list:

1	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	
	Userna	Role	Date	Tran Type	Description	Category	Subcategory	Supplier	Rec No.	ExpDate	ISO Exch	Amount Exchange	ISO Code	Income	Outcome	Balance	
2												BALANCE BOUGHT FWD :		0.00			
3	admin	Admin	31/01/2009	INCOME	Jan-09				INC1		RSD	50,000.00	RSD	50,000.00	0.00	50,000.00	
4	admin	Admin	04/02/2009	INCOME	income				INC8		RSD	200.00	RSD	200.00	0.00	50,200.00	
5	admin	Administrator	28/02/2009	INCOME	income				INC6		RSD	35,000.00	RSD	35,000.00	0.00	85,200.00	
6	admin	Admin	02/03/2009	INCOME					INC4		RSD	100.00	RSD	100.00	0.00	85,300.00	
7	admin	Admin	04/03/2009	OUTCOME	nails	HEALTH	Beauty Services & Therapy	BeautyShop	EDC1		RSD	2,000.00	RSD	0.00	2,000.00	83,300.00	
8	admin	Admin	05/03/2009	OUTCOME		HEALTH	Dentist	SavaDent	EDC2		RSD	3,000.00	RSD	0.00	3,000.00	80,300.00	
9	admin	Admin	05/03/2009	INCOME					INC5		EUR	100.00	RSD	8,000.00	0.00	88,300.00	
10	Mark	Power	05/03/2009	INCOME	Febr Income				INC9		EUR	400.00	RSD	32,000.00	0.00	120,300.00	
11	Sam	User	10/03/2009	OUTCOME		LIVING COSTS	Communications	Telekom	EDC11		RSD	3,000.00	RSD	0.00	3,000.00	117,300.00	
12	admin	Admin	12/03/2009	OUTCOME	test	EDUCATION	Membership & Association	maxi	EDC8		EUR	100.00	RSD	0.00	8,000.00	109,300.00	
13	admin	Admin	13/03/2009	INCOME	test				INC7		EUR	1,000.00	RSD	80,000.00	0.00	189,300.00	
14	admin	Admin	14/03/2009	OUTCOME	movie	LEISURE	Entertainment	TACKWOOD	EDC4		EUR	680.86	RSD	0.00	54,468.80	134,831.20	
15	Sam	User	18/03/2009	OUTCOME		LEISURE	Shopping	Maxi	EDC10		RSD	3,000.00	RSD	0.00	3,000.00	131,831.20	
16	admin	Admin	19/03/2009	OUTCOME		LIVING COSTS	Wine & Beverage	WineShop	EDC5		RSD	4,574.40	RSD	0.00	4,574.40	127,256.80	
17	admin	Admin	26/03/2009	OUTCOME	post paid	LIVING COSTS	Communications	Telekom	EDC6		RSD	1,000.00	RSD	0.00	1,000.00	126,256.80	
18	admin	Admin	31/03/2009	INCOME	sales				INC2		RSD	2,000.00	RSD	2,000.00	0.00	128,256.80	
19	admin	Admin	31/03/2009	INCOME					INC3		RSD	100.00	RSD	100.00	0.00	128,356.80	
20	admin	Admin	01/04/2009	OUTCOME		HEALTH	Dentist	SavaDent	EDC3		EUR	100.00	RSD	0.00	9,567.00	118,789.80	
21	admin	Admin	01/04/2009	OUTCOME	test	EDUCATION	Membership & Association	maxi	EDC7		EUR	100.00	RSD	0.00	9,567.00	109,222.80	
22	Mark	Power	15/04/2009	OUTCOME		LIVING COSTS	Communications	telekom	EDC9		RSD	2,000.00	RSD	0.00	2,000.00	107,222.80	
23														Totals :	207,400.00	100,177.20	107,222.80

Money Exchange (Razmena Novca)

Korišćenjem ove aplikacije, unosi se aktivni kurs valute različite od osnovne (izabrane pri pristupu aplikaciji). Svaki unos je jedinstven u odnosu na osnovnu valutu, tj. izabrani ISO kôd i datum.

Editovanje je dozvoljeno samo ukoliko ne postoje eksportovane transakcije sa datim kursom za određeni dan. Može se promeniti vrednost kursa za određenu valutu i dan. Tada će se automatski ponovo izračunati iznosi i bilansi svih relevantnih transakcija.

Brisanje kursa za određenu valutu i dan je dozvoljeno samo ukoliko ne postoje transakcije vezane za taj izabrani kurs.

ISO CODE	DESCRIPTION	EXCH. DATE	EXCH. RATE	SYMBOL
EUR	European Union	27/03/2009	95.2200	€
EUR	European Union	31/03/2009	95.6700	€
USD	United States	17/03/2009	88.0000	\$
USD	United States	30/03/2009	89.4300	\$

Income i Outcome (Prihodi i Rashodi)

Stavke menija *Income* i *Outcome* su deo naše aplikacije koji se odnosi na unos prihoda i rashoda. Dalje će biti objašnjene mogućnosti koje one nude korisniku, a mogu se koristiti otvaranjem prozora koji su prikazani na sledećim slikama.

Sve unete transakcije su jedinstvene prema osnovnoj valuti i broju računa (Receipt No polje).

Broj računa se automatski uvećava pri svakom unošenju, ali je takođe korisniku dozvoljeno i da unese proizvoljan broj računa od kojeg će se dalje svaki put povećavati ta vrednost za jedan.

Na sledećim slikama prikazani su prozori aplikacije na kojima se mogu unositi transakcije prihoda, odnosno transakcije rashoda :

DATE	USERNAME	REC NO.	DESCRIPTION	AMOUNT	BALANCE
31/01/2009	admin	INC1	Januar2009	50000.00	50000.00
31/03/2009	admin	INC2	sales	2000.00	52000.00
31/03/2009	admin	INC3		9567.00	61567.00
31/03/2009	admin	INC4		100.00	61667.00
31/03/2009	admin	INC5		8943.00	70610.00

DATE	USERNAME	REC NO.	DESCRIPTION	CATEGORY	SUBCA
04/03/2009	admin	EDC1	nails	HEALTH	Beauty S
05/03/2009	admin	EDC2		HEALTH	Dentist
11/03/2009	admin	EDC3		HEALTH	Dentist
14/03/2009	admin	EDC4	movie	LEISURE	Entertain

Ukoliko je ulogovan administrator, omogućeno mu je da unosi, menja ili briše transakcije kako za sebe tako i za bilo kog drugog korisnika. Moguće je izabrati kao alternativnu valutu neku za koju je u Admin delu prethodno određen kurs u odnosu na osnovnu. Naravno, prikazaće se jedino onaj kurs koji je poslednji registrovan u odnosu na izabrani datum transakcije.

Kada je reč o kategorijama troškova – korisniku će biti ponuđene samo one koje su na Admin delu podešene da budu aktivne. Ukoliko se javi potreba za aktiviranjem ili deaktiviranjem neke od njih, ili možda registrovanjem novih, to se može uraditi na već spomenutom Admin delu aplikacije u Category podmeniju.

Takođe, potrebno je ponoviti, da ukoliko neeksportovana transakcija ima trošak vezan za kategoriju koja je kasnije deaktivirana, neće se moći menjati niti brisati sve dok ne bude ponovo aktivirana.

Posebnu pažnju zaslužuje editovanje transakcija menjanjem datuma transakcije ili valute koja je različita od osnovne. Pošto su usko povezana ova dva parametra, menjanjem datuma promeniće se i lista valuta za koje je registrovan kurs. Tačnije, moguće je izabrati samo one valute, različite od osnovne, čiji je kurs registrovan pre datuma transakcije.

Category (Kategorije Troškova)

Opcija *Category* omogućava pregled aktivnih kategoria troškova.

Report (Izveštaj)

Opcija *Report* omogućava pravljenje izveštaja prihoda i rashoda prema različitim kriterijuma i njihovo eksportovanje u Excel dokument.

Pri pravljenju izveštaja bira se korisnik i tip transakcija - mogu se izabrati sve ili samo transakcije prihoda, odnosno rashoda.

Sa desne strane postoje opcije koje omogućavaju korisniku da napravi izveštaj koji se odnosi samo na nove transakcije (New), sve (All) ili one koje su već eksportovane (Exported).

U drugom delu prozora nalaze se razni filteri kao što su datum transakcije, kategorija troškova, broj računa. Naravno, ukoliko se kao tip transakcije izabere Prihod (Income) sva filter polja, sem datuma i broja računa, biće nedostupna.

U našem primeru napravljen je izveštaj za neeksportovane transakcije vezane za admin korisnika kao i Excel fajl dobijen pri eksportovanju :

Financial Summary

Income | Outcome | Category | Report | Money Exchange | Admin | Sign Out

Username: admin

Type: All

Date: All

Category: All

Subcategory: All

Supplier: All

Receipt No.: All

New
 All
 Exported

Show

User: admin

ReportForm

USERNAME	ROLE	DATE	TRANSACTION TYPE	DESCRIPTION	CATEGORY
admin	Administrator	31/01/2009	INCOME	Januar2009	
admin	Administrator	04/03/2009	OUTCOME	nails	HEALTH
admin	Administrator	05/03/2009	OUTCOME		HEALTH
admin	Administrator	11/03/2009	OUTCOME		HEALTH
admin	Administrator	14/03/2009	OUTCOME	movie	LEISURE
admin	Administrator	19/03/2009	OUTCOME		LIVING COST
admin	Administrator	31/03/2009	INCOME	sales	
admin	Administrator	31/03/2009	INCOME		
admin	Administrator	31/03/2009	INCOME		
admin	Administrator	31/03/2009	INCOME		

Export to Summary Excel

Pritiskom na dugme "Export to Summary Excel", finansijski izveštaj prikazan na prethodnom prozoru će se otvoriti u Excel dokumentu, što pokazuje sledeća slika :

1	Username	Role	Date	Transaction Type	Description	Category	Subcategory	Supplier	Receipt No.	Date Exported	ISO Exchange Code	Amount Exchange	ISO Code	Income	Outcome	Balance
2												BALANCE BOUGHT FWD :		0.00		
3	admin	Administrator	31/01/2009	INCOME	Jan-09				INC1		RSD	50,000.00	RSD	50,000.00	0.00	50,000.00
4	admin	Administrator	04/03/2009	OUTCOME	nails	HEALTH	Beauty Services & Therapy	BeautyShop	EDC1		RSD	2,000.00	RSD	0.00	2,000.00	48,000.00
5	admin	Administrator	05/03/2009	OUTCOME		HEALTH	Dentist	SavaDent	EDC2		RSD	3,000.00	RSD	0.00	3,000.00	45,000.00
6	admin	Administrator	11/03/2009	OUTCOME		HEALTH	Dentist	SavaDent	EDC3		RSD	10,000.00	RSD	0.00	10,000.00	35,000.00
7	admin	Administrator	14/03/2009	OUTCOME	movie	LEISURE	Entertainment	TACKWOOD	EDC4		RSD	680.86	RSD	0.00	680.86	34,319.14
8	admin	Administrator	19/03/2009	OUTCOME		LIVING COSTS	Wine & Beverage	WineShop	EDC5		RSD	4,574.40	RSD	0.00	4,574.40	29,744.74
9	admin	Administrator	31/03/2009	INCOME	sales				INC2		RSD	2,000.00	RSD	2,000.00	0.00	31,744.74
10	admin	Administrator	31/03/2009	INCOME					INC3		EUR	100.00	RSD	9,567.00	0.00	41,311.74
11	admin	Administrator	31/03/2009	INCOME					INC4		RSD	100.00	RSD	100.00	0.00	41,411.74
12	admin	Administrator	31/03/2009	INCOME					INC5		USD	100.00	RSD	8,943.00	0.00	50,354.74
13													Totals :	70,610.00	20,255.26	50,354.74

Kao što je već spomenuto, postoje tri nivoa pristupa aplikaciji. Jedan je administratorski i sva prethodna obašnja i primeri su vezani za taj pristup. Preostala dva su Power User i User, koji takođe zaslužuju da budu detaljnije opisani:

- Power User**
 Jedina razlika između ovog i administratorskog prava pristupa jeste u tome što korisnik koji pripada grupi „Power User“ nema pristup Admin delu aplikacije. Može i dalje registrovati kurs za izabrane valute u odnosu na osnovni i unosti novčane transakcije kako za sebe tako i za sve ostale korisnike.
- User**
 Osnovni režim u kojem korisnik ima pristup samo Income, Outcome, Category i Report delu aplikacije. Takođe, može unositi samo svoje prihode i rashode i praviti izveštaje vezane samo za njih.

2.0 Korišćene tehnologije i implementacija rešenja

2.1 SQL Server 2005 Express Izdanje

SQL Server 2005 Express, sistem za upravljane relacionom bazom podataka, je verzija SQL Servera 2005 koja se koristi i koja je dizajnirana za izgradnju jednostavnih aplikacija za rad sa podacima.

Osnovne karakteristike su :

- Brzo i lako preuzimanje sa interneta
- Besplatan za preuzimanje i korišćenje
- Uprošćen korisnički interfejs instalacije, tiha instalacija za ugrađeno korišćenje, integrisana primena preko opcije „Click Once“
- Takođe podrazumeva podešavanje bezbednosti i prava pristupa, kao i podršku za Windows proveru identiteta
- Lako korišćenje i upravljanje, napredan optimizator koji automatski optimizuje upite; korišćenje uskladištenih procedura
- Bogate funkcionalnosti nad bazom podataka kao što su : uskladištene procedure, prikazi, okidači (trigeri), kursori, indeksi, transact-SQL podrška, XML podrška
- Transact-SQL je proširen SQL jezik nad bazom podataka i predstavlja osnovu čitave programibilnosti Microsoft SQL Servera dodavanjem funkcionalnosti koje nisu deo standarda SQL upitnog jezika – veliku ponudu tipova podataka, privremenih objekata, uslovni tok procesa (IF . . . ELSE, WHILE, GO TO, WAITFOR), kontrolu transakcija, izuzeci i njihova kontrola, kao i mnoge druge.

Navedimo primer uskladištene procedure u kojoj su, radi bolje funkcionalnosti, kombinovani Transact-SQL izrazi i SQL izrazi.

Primenjeno je poslovno pravilo : 'Obriši samo podatke starije od dve godine.'

```
CREATE PROCEDURE purge_sales
  @order_id VARCHAR(20)
AS
IF @order_id IS NULL
  RAISERROR 50001 "Error! Please supply transaction order number."
ELSE
  DELETE temp_sales
  WHERE ord_num = @order_id AND DATEDIFF(year, ord_date, GETDATE()) > 2
GO
```

SQL Server 2005 pojednostavljuje upravljanje tako što nudi SQL integrisanu upravljačku konzolu - Server Management Studio za nadgledanje SQL Server relacione baze podataka i za upravljanje njome, kao i usluge integracije, analize, izrade izveštaja.

SQL Server Management Studio Express izdanje nije deo naše instalacije i može se odvojeno preuzeti sa sledeće internet adrese:

<http://www.microsoft.com/downloads/details.aspx?FamilyId=C243A5AE-4BD1-4E3D-94B8-5A0F62BF7796&displaylang=en> .

2.2 .Net Framework 3.5 – LINQ (Language Integrated Query)

2.2.1 Proširenje C# jezika

U .NET Framework-u 3.5 postoji niz proširenja. Jedno od njih odnosi se na lambda izraze. Lambda izrazi su novine uvedene sa .Net Framework-om 3.5, i mogu se koristiti za građenje optimizovnih LINQ upita. Kao povratna vrednost može se koristiti ključna reč 'var' koja predstavlja implicitni tip lokalne promenljive kojoj se tip određuje upravo prema njoj dodeljenoj vrednosti i koja takođe spada u proširenje C# jezika.

Objasnićemo njihova osnovna značenja kako bi se lakše razumeli primeri koda koji će biti kasnije navedeni.

Upotreba var ključne reči

Ključna reč var se može upotrebiti, bez navođenja tipa podataka, pri deklaraciji lokalne promenljive. Umesto da se od programera zahteva da eksplicitno definiše tip promenljive, on se određuje na osnovu vrednosti izraza koji joj je dodeljen. Ovo je odlika slabo tipiziranih jezika i u ovom slučaju omogućava udobniji rad.

Primera radi, mogli bismo da koristimo var rezervisanu reč za deklarisanje tri promenljive:

```
var ime = "Ana";  
var godine = 32;  
var zenskiRod = true;
```

Isti efekat bismo postigli i sledećom deklaracijom, ali u prethodnom slučaju sam prevodilac određuje tipove promenljivih.

```
string ime = "Ana";  
int godine = 32;  
bool zenskiRod = true;
```

Baš zbog njene osobine da predstavlja slabo tipiziranu lokalnu promenljivu, obavezna je njena inicijalizacija prilikom deklaracije. U suprotnom, pri kompajliranju bi došlo do greške, što bi bio slučaj u sledećem primeru :

```
var ime;
```

Delegati, Anonimne metode i Lambda izrazi

Pod delegatom možemo smatrati tip (što znači takođe da može biti prosleđen metodi kao parametar) koji referencira ili „pokazuje na“ metodu. To znači da delegat mora da pokaže na metodu koja prima određene parametre i određeni povratni tip. To su jedini zahtevi koji moraju biti ispunjeni, sama implementacija metode se ne razmatra u ovom kontekstu.

Anonimne metode predstavljaju način kako da se, umesto imena metode na koju pokazuje delegat, kao parametar prosledi telo te metode. Glavna dobit ostvarena korišćenjem anonimnih metoda je koncizniji i čitkiji kod, ne zahteva se da se pozivana metoda prethodno definiše.

Lambda izrazi koriste lambda operator => , leva strana tog operatora predstavlja ulazne parametre, dok je desna strana izraz nad njima. Lambda izraz ima jednostavnu i konciznu sintaksu za pisanje inline metoda. Upotrebom ovih lambda izraza gubi se potreba za upotrebom reči 'delegate' ili obaveznom upotrebom tipa parametra (koji se sada implicitno određuje).

Značenje operatora => bi se moglo iskazati rečima : „uzmi parametre sa leve strane, i primeni ceo izraz sa desne nad njima“.

2.2.2 LINQ

Upiti nad podacima i manipulacija nad njima oduvek je bila i uvek će biti osnovni deo programerskog posla. Možemo reći da se formati podataka menjaju, proširuju, ali potrebe su iste, što je dalje i uslovalo ideju da se koncept upita, manipulacije i menjanje podataka podigne na sledeći nivo - čime se postiže čistiji i deklarativniji kod.

LINQ je novi programski model za pristup podacima koji se integriše direktno u .NET jezike. Sam termin LINQ ima sveobuhvatno značenje, tj. predstavlja unificirani API pomoću kojeg se grade strogo tipizirani upiti nad različitim izvorima podataka – relacionom bazom podataka, XML dokumentima, različitim setovima podataka, listama i sl.

Prema kategorizaciji tipova podataka LINQ možemo podeliti na četiri dela :

- LINQ To Object
LINQ To Object se, glibo rečeno, može primeniti nad svakim objektom koji implementira IEnumerable<T> interfejs uključujući i proste nizove, kao i generičke i negeneričke kolekcije podataka.
- LINQ To DataSet
LINQ To DataSet nadograđuje ADO.NET DataSet programski model dozvoljavajući manipulaciju skupovima podataka kao što su: DataSet, DataTable, DataRow. Potrebno je pozivati System.Data.DataSetExtensions.dll koji proširuje System.Data prostor imena.

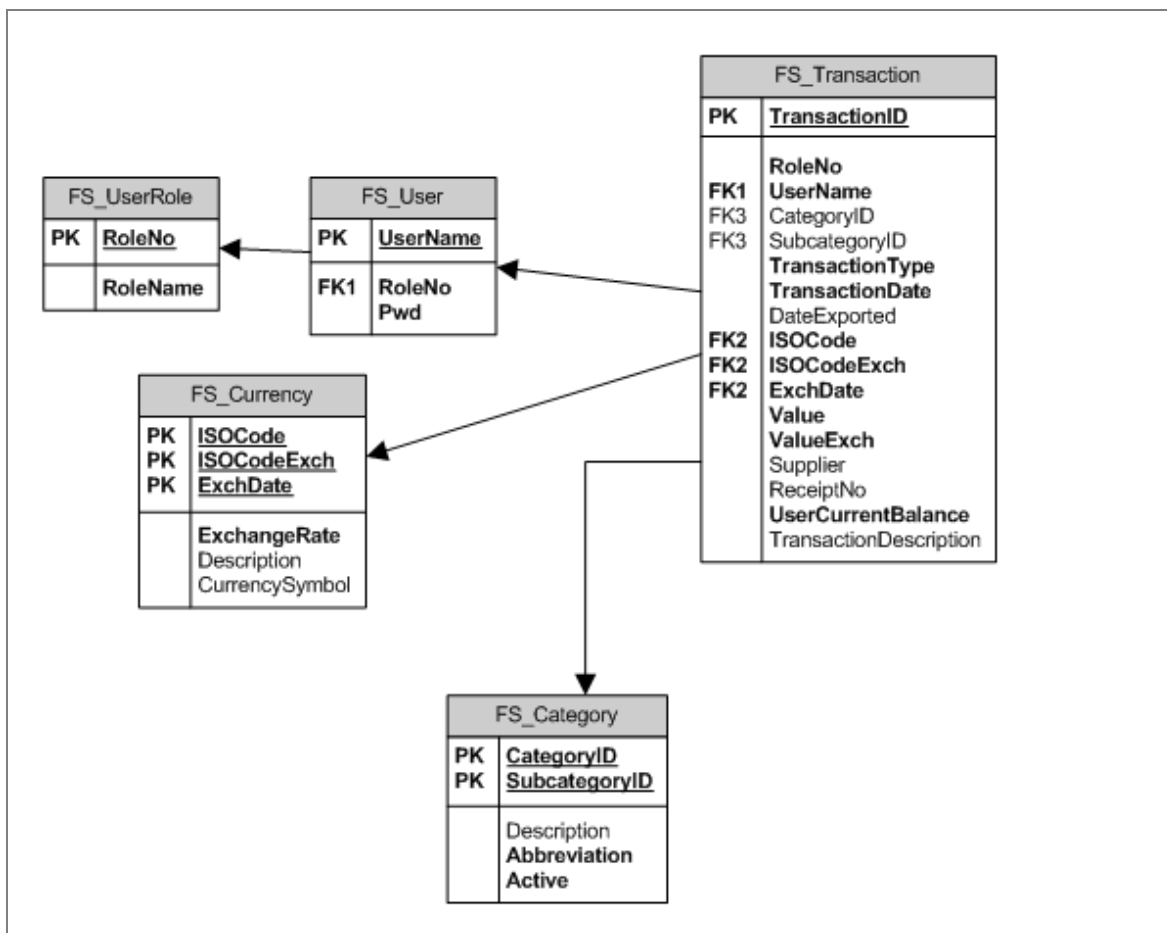
Data Set omogućava rad sa podacima pri raskinutoj konekciji, tj. ne održava konekciju sa izvorom podataka, već se nakon njihovog ažuriranja vrši usaglašavanje sa originalnim izvorom (bazom). Data Set zapravo predstavlja celokupan skup podataka smešten u memoriju i obuhvata tabele, relacije i definisana ograničenja.

- LINQ To SQL
LINQ To SQL omogućava interakciju sa relacionom bazom podataka preko klasa entiteta koje predstavljaju tabele relacione baze podataka.
Potrebno je pozivati System.Data.Linq.dll koji proširuje System.Data prostor imena.
U izradi aplikacije korišćen je upravo LINQ To SQL koji će u nastavku biti detaljnije objašnjen.
- LINQ To XML
LINQ To XML omogućava primenu LINQ izraza nad XML dokumentima. Potrebno je pozivati System.Xml.Linq.dll koji proširuje System.Data prostor imena.

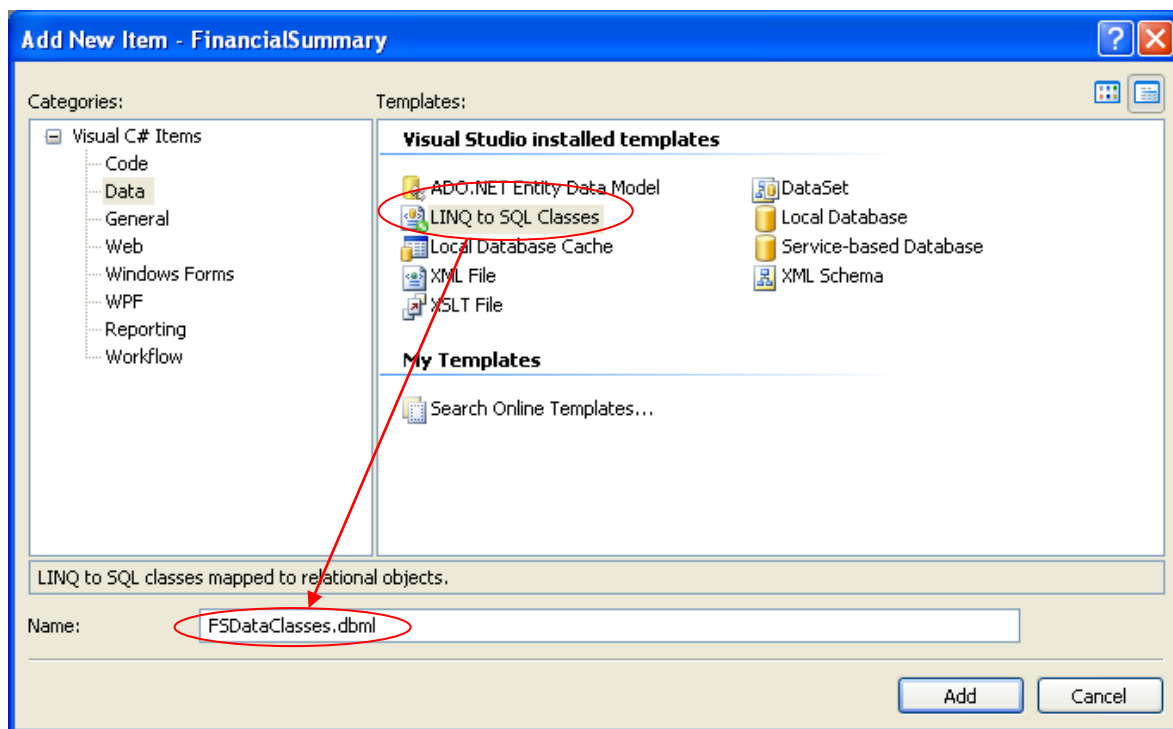
LINQ To SQL

LINQ To Sql je O/RM („Object-relational mapping“) implementacija koja omogućava modelovanje relacione baze podataka pomoću .NET klasa. LINQ se dalje može koristiti za realizaciju upita nad bazom, kao i izvršavanje svih operacija kreiranja/čitanja/ažuriranja/brisanja (u daljem tekstu CRUD (Create/Read/Update/Delete) operacija). Ovaj programski model potpuno podržava transakcije, poglede nad bazom, kao i uskladištene procedure, omogućava olakšano integrisanje validacije podataka i biznis-logike u sam model baze.

Osnovni sloj naše aplikacije je relaciona baza podataka u kojoj će se nalaziti svi potrebni podaci. Sledećom šemom su predstavljene sve tabele i veze između njih potrebne za njeno funkcionisanje:

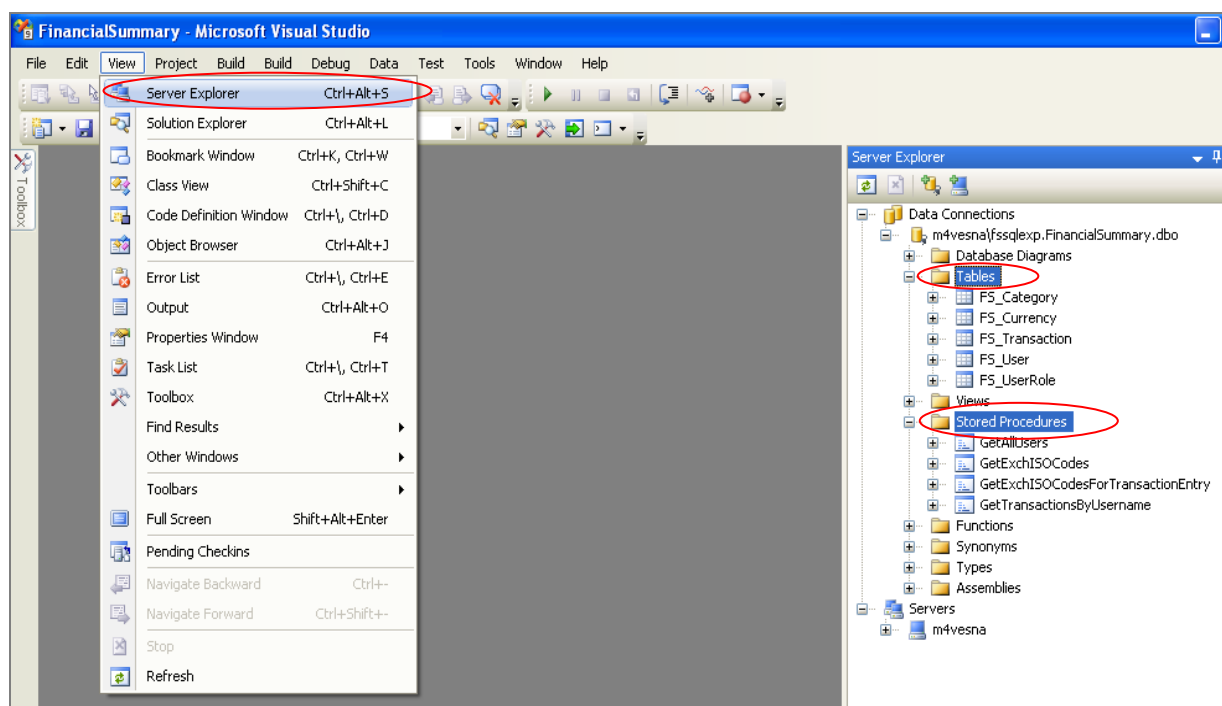


Dodavanjem projektu 'LINQ To Sql' stavke u Visual Studio okruženju automatski se generiše LINQ To Sql dizajner pomoću kojeg se vrlo lako mogu modelovati klase koje će predstavljati relacionu bazu podataka. Takođe, automatski se generiše DataContext klasa koja je glavna nit između relacione baze podataka i LINQ-a.

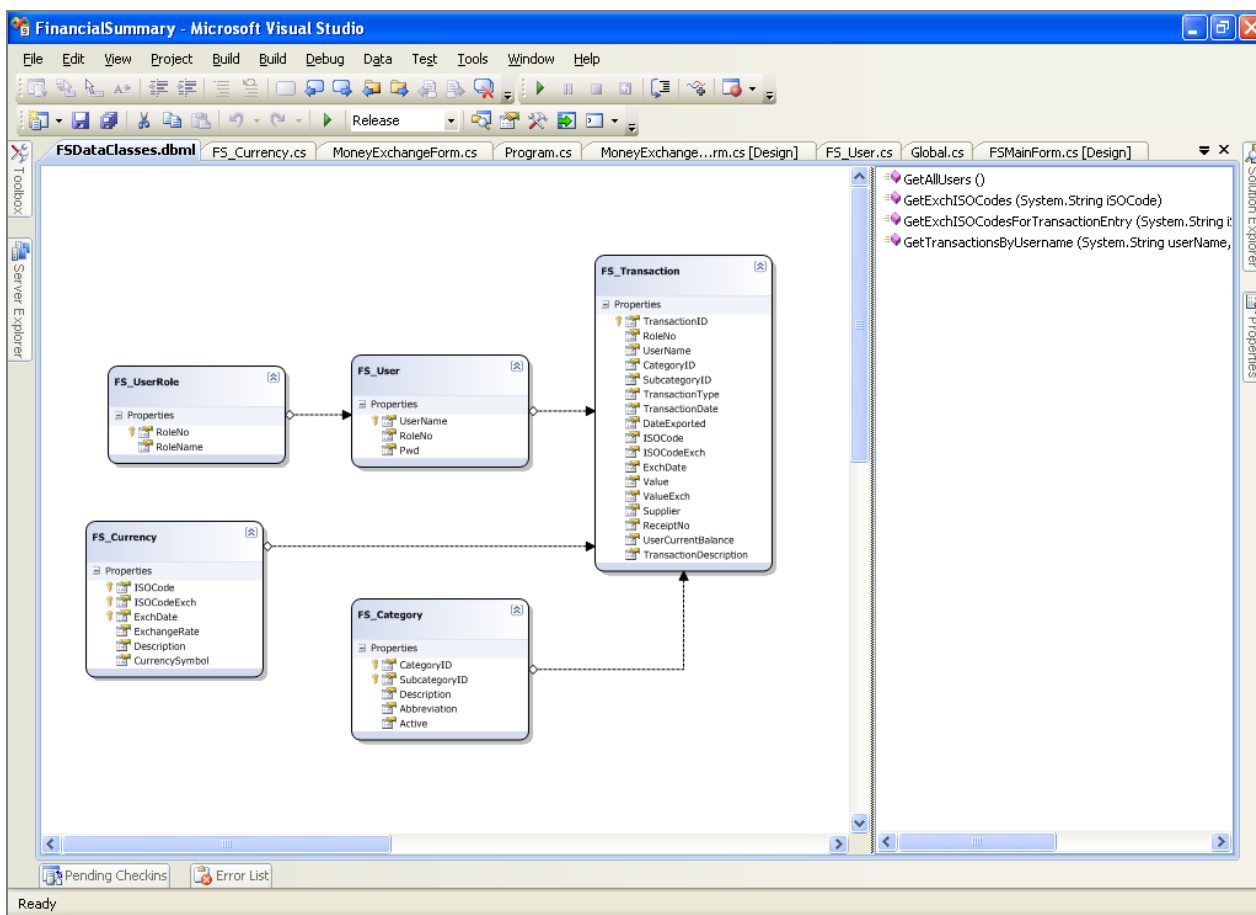


Ukoliko šema relacione baze podataka već postoji, pomoću LINQ To Sql dizajnera mogu se vrlo lako napraviti klase entita koje odgovaraju tabelama baze, što se može pokazati na primeru baze naše aplikacije.

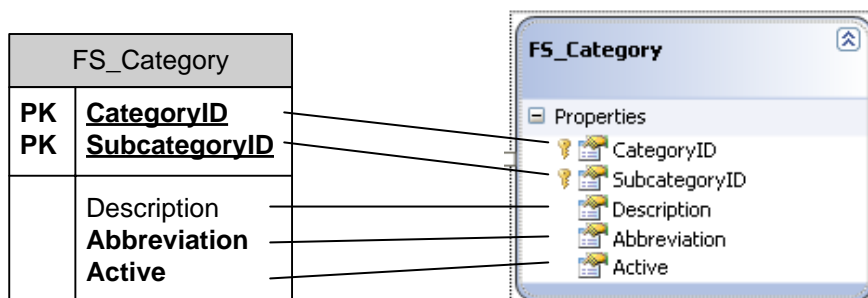
Jedan od načina za postizanje toga jeste otvaranjem Server Explorer-a u okruženju Visual Studio 2008, selektovanjem tabela, pogleda, uskladištenih procedura i njihovim prevlačenjem na prozor dizajnera, što je delimično prikazano na sledećoj slici.



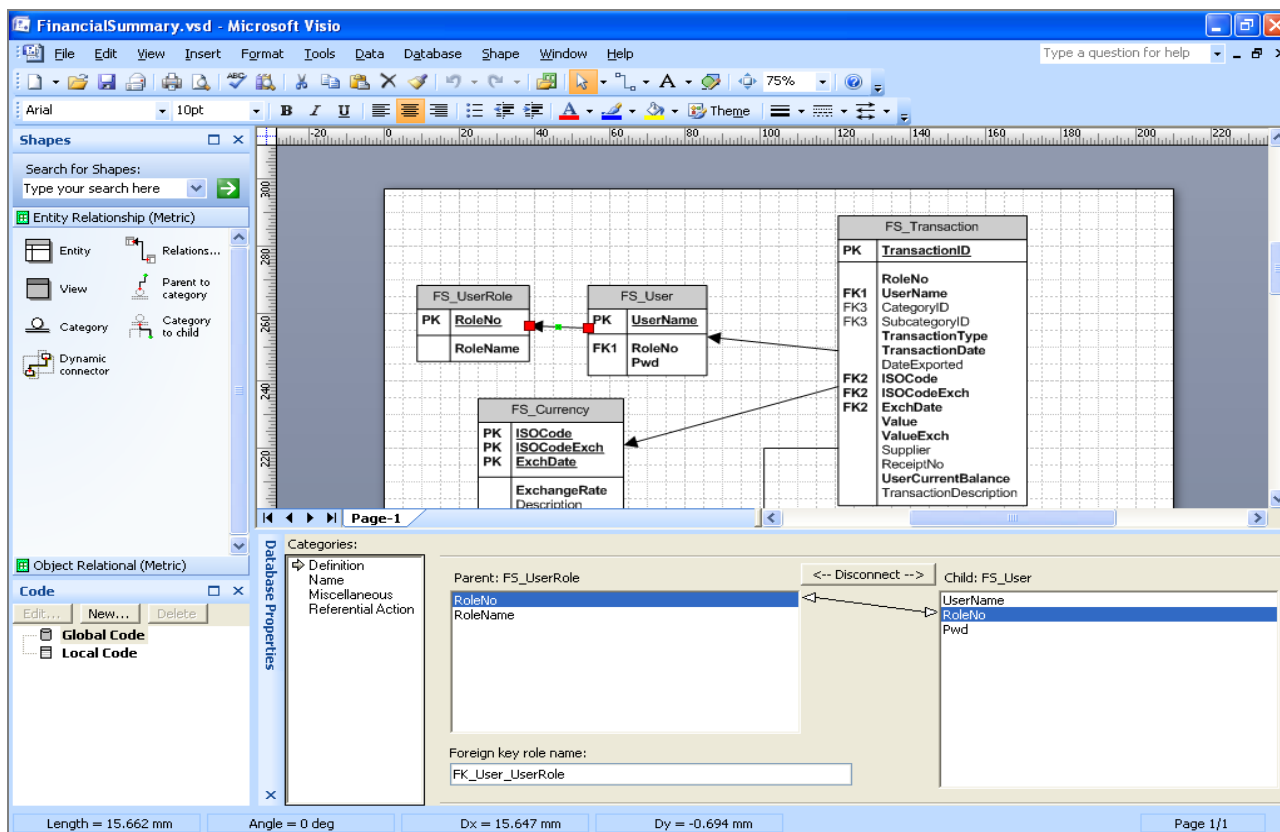
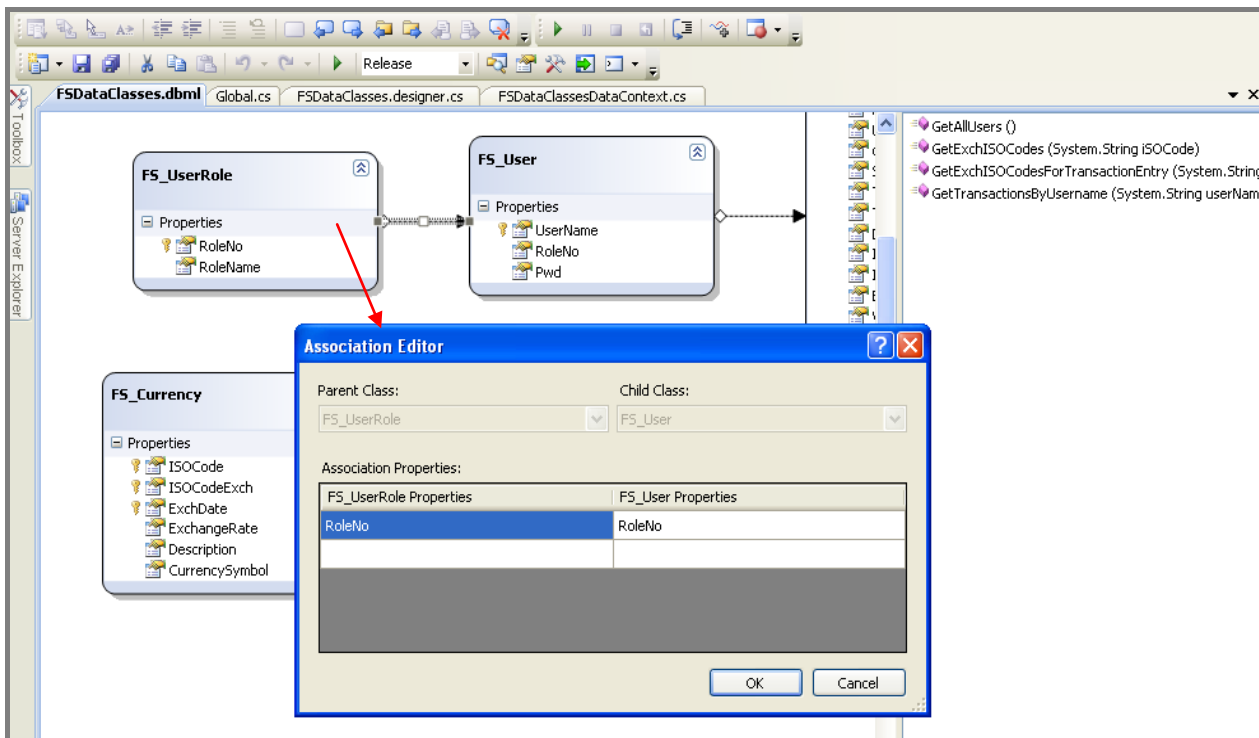
U trenutku kreiranja tabela u LINQ To Sql dizajneru (bilo prevlačenjem iz Server Explorer-a u VS-u, bilo njihovim direktnim kreiranjem) za svaku od njih definisaće se po jedna klasa koja predstavlja upravo preslikavanje tabela u relacionoj bazi podataka. U našem primeru kreiraće se pet takvih klasa koje će predstavljati ekvivalente tabela relacione baze podataka, čija je šema prikazana na jednoj od prethodnih slika. To su klase (tabele) sa imenima : FS_UserRole, FS_User, FS_Transaction, FS_Currency, FS_Category. Sadržaji ovih klasa, kao i njihova međusobna povezanost prikazani su pomoću UML-dijagram na sledećoj slici.



LINQ To SQL omogućuje modelovanje klasa koje se najčešće nazivaju klasama entiteta, a njihove instance entitetima. Klase entiteta preslikavaju tabele baze podataka, a njihovi atributi predstavljaju same kolone tabele koju klasa predstavlja. Svaki entitet, instanca klase entiteta, je upravo jedan red (rekord) same tabele.



U trenutku dodavanja tabela Visual Studio proverava odnose između tabela u bazi i na osnovu njih automatski generiše odnose zavisnosti klasa entiteta koji su predstavljeni stelicama i odgovaraju primarnom/stranom ključu referentnih tabela i njihovih odnosa u bazi. U našim primerima napravljeno je uporedno poređenje i na taj način predstavljene baze u VS-u 2008 i šeme baze u Visio okruženju.



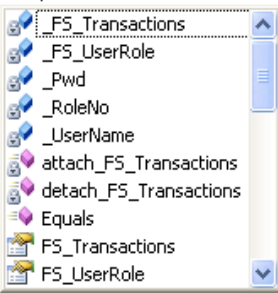
U primeru iz naše aplikacije odnos između FS_UserRole i FS_User tabela omogućava da se pomoću RoleNo atributa entiteta klase FS_User, može pristupiti klasi entiteta FS_UserRole kojoj odgovarajući korisnik pripada. Takođe, FS_UserRole klasa, na osnovu RoleNo atributa, može dobiti kolekciju instanci FS_User klase koje pripadaju određenoj roli. Na sledećoj slici prikazan je deo deo kôda u kom se na osnovu imena korisnika dobija kojoj kategoriji pripada.

```

int roleNo = 0;
FSDataClassesDataContext dcdbc = new FSDataClassesDataContext();
dcdbc.Connection.ConnectionString = @"data source=localhost\FSSQLEXP;initial catalog=FinancialSummary;";

var roleNoDb = from user in dcdbc.FS_Users
                where user.UserName == username select user.RoleNo;
roleNo = roleNoDb.First();
return roleNo;

```

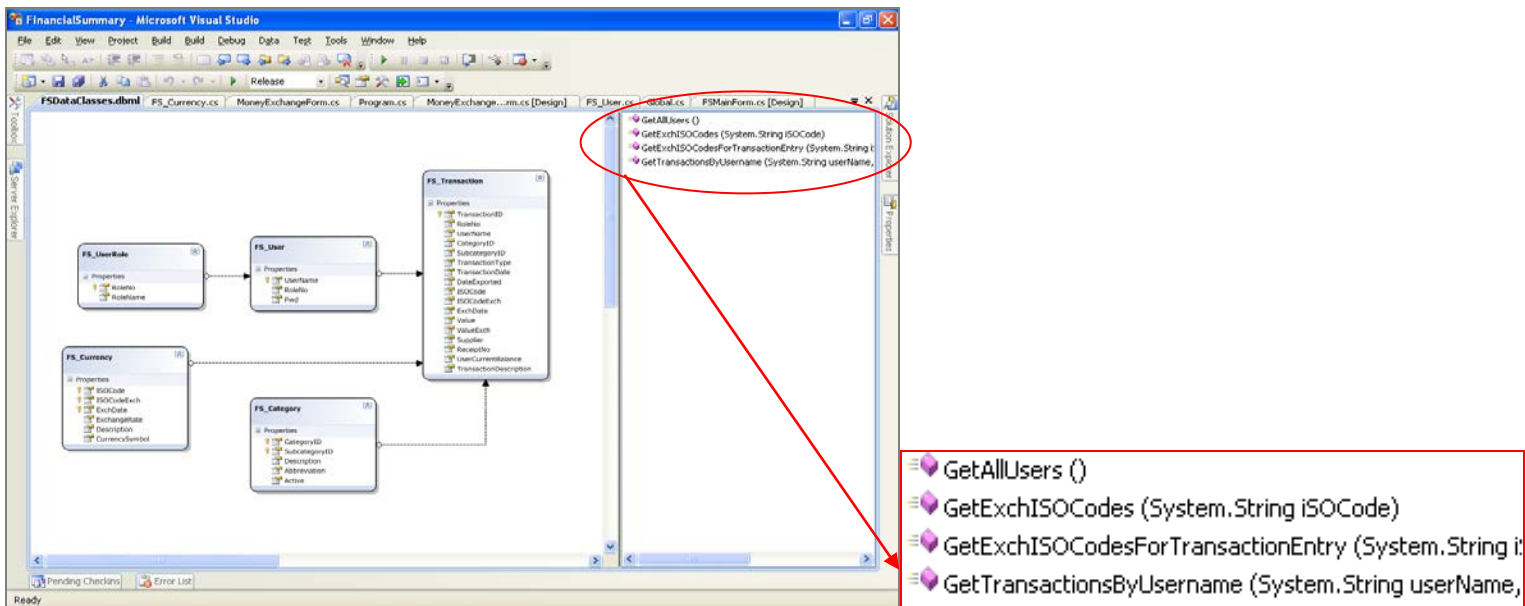


Nasuprot DataSet/DataAdapter objektima, koji su u Visual Studiu 2005 korišćeni za uzimanje podataka iz baze i njihovu manipulaciju, korišćenjem LINQ To Sql-a ne mora se razmišljati o detaljima sql upita i data nivou na način na koji je to do sada moralo. Naime, programer se može prvenstveno skoncentrisati na definisanje klase entiteta, njihovo preslikavanje i međusobne odnose, a LINQ To SQL O/RM implementacija će se „pobrinuti“ za generisanje odgovarajuće logike izvršavanja sql upita.

Svaka klasa kreirana na ovaj način je definisana kao „nepotpuna“ (partial) klasa – što zapravo znači da se dodatni atributi, metode i događaji mogu dodavati i to ne mora biti u istom fajlu.

Takođe, vredno je spomenuti da se sama imena tabela i njihovih odnosa, kao i njihove specifičnosti mogu promeniti tako da imaju drugačije vrednosti od onih automatski dodeljenih od strane LINQ To Sql dizajnera u trenutku njihovog kreiranja.

Desna strana LINQ To Sql dizajnera sadrži listu uskladištenih procedura definisanih nad bazom podataka kojima se dalje može pristupiti preko instance DataContext klase. Na sledećoj slici prikazana je, unutar LINQ To SQL dizajner prozora, lista definisanih uskladištenih procedura nad našom bazom.



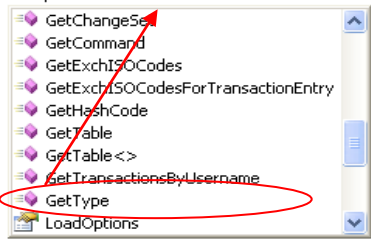
Na sledećoj slici prikazan je primer definisane uskladištene procedure, i njen poziv u kodu :

Procedura `GetTransactionsByUserName` vraća kolekciju transakcija koje odgovaraju određenom korisniku za određenu valutu i tip transakcije; kao ulazne parametre prima ime korisnika (`UserName`), korišćenu valutu (ISO Code), i tip transakcije (`TransactionType`) – koji može biti prihod (Income) ili rashod (Outcome).

```
ALTER PROCEDURE dbo.GetTransactionsByUsername
(
    @UserName varchar(50),
    @ISOCODE varchar(5),
    @TransactionType varchar(20)
)
AS
SELECT TransactionDate, UserName, ReceiptNo, ISOCODEExch, ValueExch,
    TransactionDescription, TransactionID, CategoryID, SubcategoryID,
    Supplier, ISOCODE, Value, UserCurrentBalance
FROM FS_Transaction
WHERE UserName = @UserName
    AND ISOCODE = @ISOCODE
    AND TransactionType = @TransactionType
    AND DateExported IS NULL
RETURN
```

```
public static void FillTransactionDGV(DataGridView dgv, string userName)
{
    FSDataClassesDataContext dcdc = new FSDataClassesDataContext();
    dcdc.Connection.ConnectionString = @"data source=localhost\FSSQLEXP;initial catalog=FinancialSumme

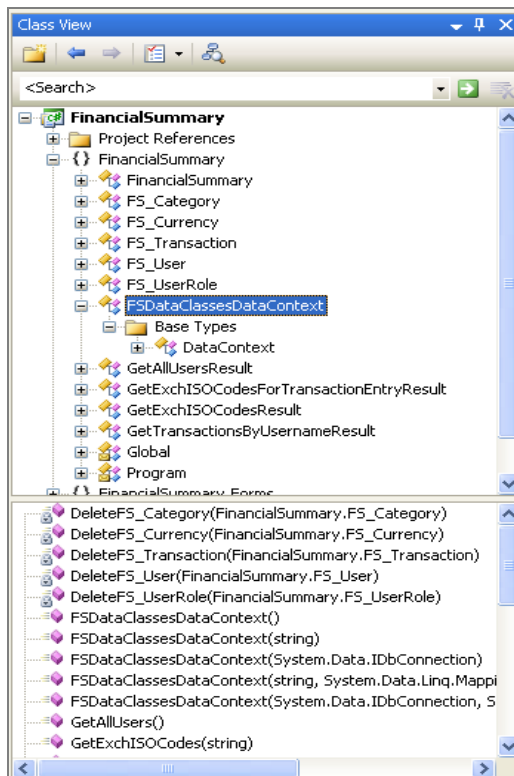
    dgv.DataSource = dcdc.GetTransactionsByUserName(userName, Global.ISOLoggedIn, trnType);
}
```



LINQ To Sql će u trenutku kreiranja klasa entiteta automatski generisati CRUD izraze nad entitetima. Ove podrazumevane metode se takođe mogu promeniti prema potrebama specifikacije samog projekta, tako što bi se prvo definisale odgovarajuće uskladištene procedure koje bi zamenile podrazumevane.

Primetićemo iz datog primera da je, pre izvršavanja bilo kog LINQ izraza, potrebno najpre instancirati `DataContext` klasu, koja igra ključnu ulogu u izvršavanju upita nad bazom i svim njenim promenama.

Za svaki LINQ To Sql dizajner fajl generiše se DataContext klasa čiji atributi predstavljaju tabele, kao i metode koje odgovaraju uskladištenim procedurama baze podataka. Otvaranjem prozora Class View u VS-u, koji je prikazan na sledećoj slici, mogu se videti svi atributi i metode ove klase.



Ova klasa kao svoj atribut ima konekcioni string, koji se može dodeliti pri samom instanciranju klase u njenom konstruktoru ili kao vrednost „ConnectionString“ atributa u već kreiranoj instanci.

```
string connStr = @"data source=localhost\FSSQLEXP;initial catalog=FinancialSummary;uid=sa;pwd=FSummary@123";
FSDataClassesDataContext dcdc = new FSDataClassesDataContext(connStr);
```

ili

```
string connStr = @"data source=localhost\FSSQLEXP;initial catalog=FinancialSummary;uid=sa;pwd=FSummary@123";
FSDataClassesDataContext dcdc = new FSDataClassesDataContext();
dcdc.Connection.ConnectionString = connStr;
```

Na taj način otvaranje i zatvaranje konekcije ka bazi vezano je za instancu DataContext klase, što olakšava rad programeru.

DataContext klasa je ključna i u preslikavanju tabela i manipulaciji podacima, u čemu veliku ulogu imaju generičke kolekcije. Kao primer se može uzeti metoda `GetTable<T>()`, koja kao rezultat vraća generičku kolekciju `Table<T>()`. Generička klasa `Table` implementira interfejs `IQueryable<T>` koji nasledjuje `IEnumerable<T>` interfejs, nad kojom se dalje mogu izvršavati LINQ upiti kao nad tabelom.

Pošto kreiramo bazu podataka, dalje možemo vrlo lako manipulirati podacima koristeći mnogobrojne LINQ mogućnosti.

Osnovne funkcije kreiranja, čitanja, ažuriranja i brisanja pokazaćemo na primeru tabele FS_Category naše baze, odnosno klasi pod istim imenom koja njoj odgovara:

- Stvaranje novih objekata i njihovo unošenje u bazu podataka :

U našem primeru generiše se nova instanca klase FS_Category, dodeljuju se vrednosti njenim atributima (koji predstavljaju kolone tabele), zatim se dodaje taj objekat kolekciji objekata i poziva metoda SubmitChanges() - koja kreira i izvršava INSERT upit nad bazom podataka.

```
FS_Category category = new FS_Category();
class FinancialSummary.FS_Category
category.CategoryID = categoryID;
category.SubcategoryID = subcategoryID;
category.Description = description;
category.Abbreviation = prefix;
category.Active = active;
dcdc.FS_Categories.InsertOnSubmit(category);
dcdc.SubmitChanges();
```

- Čitanje :

Uzimanje reda tabele koji zadovoljava određeni kriterijum vrlo lako se postiže korišćenjem LINQ upita, koji će detaljnije biti kasnije objašnjeni. U našem primeru uzimaju se sve kategorije koje imaju aktivan status:

```
var categories = from category in dcdc.FS_Categories
                 where category.Active
                 select category;
```

- Ažuriranje/brisanje postojećih slogova se vrši tako što se prvo pročita, tj.dovuče red tabele, odnosno odgovarajući objekat i ažuriraju se vrednosti njenih atributa, odnosno obriše se iz kolekcije :

Ažuriranje :

```
FS_Category categoryOld = dcdc.FS_Categories.Single(x => x.CategoryID == oldCategoryID
                                                    && x.SubcategoryID == oldSubcategoryID);

categoryOld.Description = description;
categoryOld.Active = active;
dcdc.SubmitChanges();
```

Brisanje :

```
FS_Category category = dcdc.FS_Categories.Single(x => x.CategoryID == categoryID
                                                    && x.SubcategoryID == subcategoryID);
dcdc.FS_Categories.DeleteOnSubmit(category);
dcdc.SubmitChanges();
```

Sve promene koje su napravljene nad instancama klasa koje odgovaraju odgovarajućim tabelama u bazi, biće zaista izvršene nad bazom tek nakon poziva funkcije SubmitChanges(). U trenutku pozivanja, LINQ To Sql dinamički konstruiše i izvršava metode kreiranja i čuvanja, brisanja, ažuriranja.

Takođe, bitno je napomenuti da pri ažuriranju redova tabele, tj. instanci odgovarajuće klase, kôd će se izvršiti samo nad onim redovima čije su vrednosti kolona zaista promenjene. Drugim rečima, ažuriranjem se ne bi smatrao deo koda koji bi kolonama određenog reda u tabeli dodelio vrednost koju već sadrži.

U trenutku poziva SubmitChanges() metode sve promene će biti podrazumevane pod jednom transakcijom, što znači da će se ili sve izvršiti uspešno nad bazom podataka ili neće.

Bitno je napomenuti da je u .NET Framework-u 3.5 ostavljeno prostora i za pisanje i izvršavanje sql upita na način na koji su programeri do sada i navikli.

Primera radi, prethodni kod brisanja objekta, odnosno reda tabele, identičan je sledećem :

```
string sql = @"DELETE FROM FS_Transaction WHERE CategoryID = {0} AND SubcategoryID = {1}";
object[] parametri = new object[] { oldCategoryID, oldSubcategoryID };
var izlaz = dcdc.ExecuteCommand(sql, parametri);
```

Jedan od neizbežnih zadataka pri radu sa podacima jeste njihova validacija. LINQ To Sql daje programerima efikasan način kako da validaciju podataka definišu samo jednom, bez ponavljanja koda, što vodi ka lakšem održavanju koda projekta i njegovoj boljoj čitljivosti.

Naime, kada se definišu klase entiteta korišćenjem LINQ To Sql dizajnera u okruženju VS 2008, određena pravila validacije će se automatski generisati, tačnije tipovi atributa klasa entiteta odgovaraće tipovima definisanih kolona tabela šeme baze podataka i nikakvi dodatni koraci nisu potrebni da bi se ona koristila. To bi značilo da, ukoliko bi se pokušalo dodeljivanje vrednost tipa string atributu tipa decimal, došlo bi do greške pri kompajliranju. Drugi primer bi bio ako bi se pokušalo dodeljivanje NULL vrednost atributu koji je definisan tako da ne može imati nedefinisane vrednosti.

Naravno, kako je svaka klasa entiteta generisana pomoću LINQ To Sql dizajnera, nepotpuna (partial) klasa, ove metode validacije se mogu predefinisati i njihova definicija se ne mora nalaziti u istom fajlu.

Primer predefinisane metode validacije unetih podataka :

Funkcija `OnValidate()` unutar nepotpune klase (koja je ekvivalent istoimene tabele u bazi podataka) `FS_Transaction` vrši dodatne provere da li vrednosti, dodeljene atributima instance ove klase, zadovoljavaju određene uslove :

- Vrednost rashoda/prihoda ne može biti jednaka nuli, niti negativna
- Opis transakcije ne može imati više od 200 proizvoljnih znakova
- Broj računa mora počinjati sa 3 karaktera iza kojih sledi makar jedan broj
- Naziv dobavljača ne može imati više od 50 proizvoljnih karaktera (slučaj ako se unosi transakcija rashoda)

U samoj funkciji, radi efikasnije validacije, korišćeni su regularni izrazi, što se može videti na sledećoj slici :

```
partial void OnValidate(System.Data.Linq.ChangeAction action)
{
    Regex regVal = new Regex(@"(?!\^0*$) (?!\^0*[\.\,]0*$)^\d+([\.\,]\d{0,2})?");
    if (!regVal.IsMatch(ValueExch.ToString()))
    {
        throw new Exception("Invalid Amount.");
    }
    regVal = new Regex(@"^\.(0,200)$");
    if (!regVal.IsMatch(TransactionDescription))
    {
        throw new Exception("Invalid Description.");
    }

    regVal = new Regex(@"^[a-zA-Z]{3}\d{1,}$");
    if (!regVal.IsMatch(ReceiptNo))
    {
        throw new Exception("Invalid Receipt No.");
    }
    if (trnType == "OUTCOME")
    {
        regVal = new Regex(@"^\.(1,50)$");
        if (!regVal.IsMatch(Supplier))
        {
            throw new Exception("Invalid Supplier.");
        }
    }
}
```

Dalje se u samoj parcijalnoj klasi `DataContext` naglasi da se, prilikom validacije podataka pri čuvanju, koristi naša proširena `OnValidate()` metoda na sledeći način :

```
public partial class FSDataClassesDataContext
{
    partial void InsertFS_Transaction(FS_Transaction instance)
    {
        this.ExecuteDynamicInsert(instance);
    }
    //...
}
```

Ova metoda će biti automatski pozvana svaki put pri pozivanju `SubmitChanges()` metode pri čuvanju novog sloga (izvršavanje `INSERT` komande).

Osnovna sintaksa u kreiranju LINQ upita bila bi sledeća :

```
var result = from [identifikator] in [kolekcija objekata]
let [izraz]
where [logički izraz]
order by [[izraz](ascending/descending)],
        [opcionalna ponavljanja izraza]
select [izraz]
group [izraz] by [izraz] into [izraz]
```

Navešćemo i tabelu operatora koji se koriste u izgradnji LINQ upita :

Tip Operatora	Operator	Opis
Agregacija	<u>Average</u>	Izračunava prosek nad skupom numeričkih vrednosti.
	<u>Count</u>	Izračunava broj rekorda skupa, Povratna vrednost je tipa int
	<u>LongCount</u>	Povratna vrednost je tipa long.
	<u>Max</u>	Pronazi maksimalnu vrednost.
	<u>Min</u>	Pronalazi najmanju vrednost.
	<u>Sum</u>	Računa zbir numeričkih vrednosti.
Konkatenacija	<u>Concat</u>	Spaja dve sekvence.
Konverzija	<u>Cast</u>	Vrši konverziju datog skupa u određeni tip.
	<u>OfType</u>	Filtrira elemente skupa određenog tipa.
	<u>ToArray</u>	Kreira niz od datog skupa objekata.
	<u>ToDictionary</u>	Kreira Dictionary(Rečnik) objekat.
	<u>ToList</u>	Kreira listu.
	<u>ToLookup</u>	Kreira lookup.
	<u>ToSequence</u>	Vraća argument funkcije kao IEnumerable tip.
Element	<u>DefaultIfEmpty</u>	Predstavlja podrazumevanu vrednost elementa ukoliko je prazan skup.
	<u>ElementAt</u>	Vraća element na određenom mestu određen indeksom.
	<u>ElementAtOrDefault</u>	Vraća element određen indeksom, ili podrazumevanu vrednost ukoliko je indeks izvan opsega.
	<u>First</u>	Vraća prvi element niza.
	<u>FirstOrDefault</u>	Vraća prvi ili podrazumevanu vrednost ukoliko takav element ne postoji.
	<u>Last</u>	Vraća poslednji element niza.
	<u>LastOrDefault</u>	Vraća poslednji element niza ili podrazumevanu vrednost ukoliko takav element ne postoji.
	<u>Single</u>	Vraća jedan element niza.
	<u>SingleOrDefault</u>	Vraća jedan element niza ili podrazumevanu vrednost

		ukoliko takav element ne postoji.
Jednakost	<u>SequenceEqual</u>	Proverava da li su dva niza jednaka.
Kreiranje	<u>Empty</u>	Vraća prazan niz datog tipa.
	<u>Range</u>	Generiše segment integer vrednosti.
	<u>Repeat</u>	Generiše niz ponavljanjem odgovarajuće vrednosti dati broj puta.
Grupisanje	<u>GroupBy</u>	Grupiše elemente datog niza.
Spajanje	<u>Join</u>	Unutašnje spajanje (inner join) dve kolekcije elemenata na osnovu zajedničkih atributa
Sortiranje	<u>OrderBy</u>	Sortira rastuće prema jednom ili više datih ključeva.
	<u>ThenBy</u>	Sortira već sortiranu kolekciju prema odgovarajućim ključevima.
	<u>OrderByDescending</u>	Sortira kolekciju u opadajućem poretku prema jednom ili više datih ključeva.
	<u>ThenByDescending</u>	Sortira već sortiranu kolekciju u opadajućem poretku preko jednom ili više datih ključeva.
	<u>Reverse</u>	Obrće redosled elemenata u datoj kolekciji.
Izdvajanje	<u>Skip</u>	Preskače određen broj elemenata u datoj kolekciji.
	<u>SkipWhile</u>	Preskače elemente u kolekciji dokle god je ispunjen while uslov, tj. njegova vrednost je true
	<u>Take</u>	Uzima dati broj elemenata, ostali deo zanemaruje.
	<u>TakeWhile</u>	Uzima elemente date kolekcije dokle god je ispunjen while uslov, ostali deo kolekcije se zanemaruje.
Kvantifikatori	<u>All</u>	Proverava da li svi elementi kolekcije zadovoljavaju određeni uslov.
	<u>Any</u>	Proverava da li bilo koji element kolekcije zadovoljava dati uslov.
	<u>Contains</u>	Proverava da li data kolekcija sadrži dati element.
Restrikcije	<u>Where</u>	Filtrira kolekciju na osnovu datog kriterijuma.
Projekcije	<u>Select</u>	Projekcija nad datom kolekcijom elementa.
	<u>Distinct</u>	Eliminiše duplikate iz kolekcije elemenata.
	<u>Except</u>	Vraća razliku dve kolekcije.
	<u>Intersect</u>	Vraća presek dve kolekcije.
	<u>Union</u>	Vraća uniju dve kolekcije.

3.0 Zaključak

„Aplikacija za vođenje evidencije prihoda i rashoda“ predstavlja primer pojednostavljenog finansijskog softvera. Njena prednost je jednostavna instalacija, mogućnost uvida u trenutno stanje za svakog korisnika i u svakom trenutku, kao i čuvanje i prikaz izveštaja u vidu Excel dokumenta za određeni period i njihovo eventualno ponovno kreiranje.

Ova aplikacija se može prilagoditi svakom vidu poslovanja preko registrovanja korisnika sa određenim pravima pristupa aplikaciji, kreiranju kursnih listi, kao i definisanja novih kategorija troškova.

Moglo bi se razmišljati u smeru daljeg proširenja ove aplikacije čija implementacija, zahvaljujući LINQ-u, bi se mogla vrlo efikasno i koncizno realizovati.

Primeru radi, jedna od ideja bi bila vezivanje bankarskog računa za svakog korisnika ili grupu kojoj pripada, kao i svaku kategoriju prihoda i rashoda. Na taj način omogućilo bi se realizovanje dvojnog knjigovodstva i pravljenje izveštaja prema određenom računu.

Takođe, druga ideja bi mogla biti unapređivanje administrativnog dela kategorija, tako da se određenim korisnicima dodeljuju određene kategorije.

Možemo nesumljivo zaključiti da finansijska softverska rešenja daju nezanemarljiv doprinos efikasnosti računovodstvenog procesa i da su danas neophodni deo svakog uspešnog poslovanja.

4.0 Literatura

- Jason Price, Mike Gunderloy; „Mastering Visual C# .NET“; Sybex
- Fabrice Marguerie, Steve Eichert, Jim Wooley; „LINQ in Action“; Manning
- Andrew Troelsen; „Pro C# 2008 and the .NET 3.5 Platform“, Fourth Edition; Apress
- „Leksikon računovodstva i poslovnih finansija“; Savez računovodstvenih i finansijskih radnika Srbije
- Dr.Kata Škarić-Jovanović; “Finansijsko Računovodstvo”; Ekonomski fakultet u Beogradu
- <http://weblogs.asp.net/scottgu>
serija tutorijala : „LINQ To SQL“
 „Anonymous Types“
 „Extension Methods“
- <http://en.wikipedia.org>
- <http://www.microsoft.com/scg/sql/editions/express/features.msp>
- <http://msdn.microsoft.com/netframework>