Univerzitet u Beogradu
Matematički fakultet

Primena Oracle ADF-a u kreiranju modula „Inostrana blagajna“

Master rad

Student

Olga Grujić

Mentor

prof. dr. Dušan Tošić

Beograd, oktobar 2009.

Sadržaj

I PREDGOVOR

Master teza pod naslovom „Primena Oracle ADF-a (*Oracle Application Development Framework*) u kreiranju modula Inostrana blagajna“, predstavlja opis softverskog rešenja za deo ERP sistema („*Enterprise Resource Planing*“, odnosno „Poslovni Informacioni Sistem“) koji, najkraće rečeno, prate sve aspekte poslovanja jedne kompanije. ERP sistemi omogućavaju integraciju kompletnog funkcionisanja poslovnog sistema pomoću jedinstvenog softverskog rešenja.

Modul „Inostrana blagajna“ služi za evidenciju i obračun putnih troškova osobe kaja se upućuje na službeni put u inostranstvo, za izdavanje naloga blagajni novca radi devizne gotovinske uplate/isplate po različitim osnovama, kao i za praćenje dnevnih promena u blagajni novca.

Cilj ovog rada je da objasni suštinu i osnov Oracle ADF-a kroz razvoj poslovnih aplikacija, tj. kroz definisanje poslovnih zahteva, kreiranje objekata i sjedinjavanje u poslovnu logiku.

Ovaj rad je na odgovarajući način komponovan od dva glavna dela:

- opis korišćene tehonologije i alata,
- primena i opis modula „Inostrana blagajna“, koji je kreirao autor ovog rada za demonstraciju korišćenih tehnologija i alata.

U prvom delu objašnjeni su glavni delovi Oracle ADF-a (za Java EE), alat Oracle JDeveloper i njihova suštinska primena, kako bi se u drugom delu kroz dijagrame moglo pratiti samo kreiranje modula.

U drugom delu su opisi izgleda i funkcionalnosti aplikacija. Dat je prikaz ADF modela i fizičkog modela podataka, kao i izgled baze podataka.

II OPIS KORIŠĆENE TEHNOLOGIJE I ALATA

Za implementaciju modula „Inostrana blagajna“ korišćen je Oracle Application Development Framework (Oracle ADF). U ovom poglavlju biće opisan način korišćenja i kreiranja aplikacija pomoću Oracle ADF poslovnih komponenti (*Oracle ADF Business Components,* skraćeno ADF BC), u razvojnom okruženju JDeveloper (verzija 10g).

Oracle ADF BC sadrže poslovnu logiku i povećavaju produktivnost, tako što implementiraju komponente za višestruku upotrebu.

Oracle JDeveloper 10g predstavlja integrisano razvojno okruženje za razvoj aplikacija, koristeći Java standarde, XML i SQL. Zajedno sa Oracle ADF-om ima za cilj pojednostavljenje razvoja Java enterprise edition aplikacija.

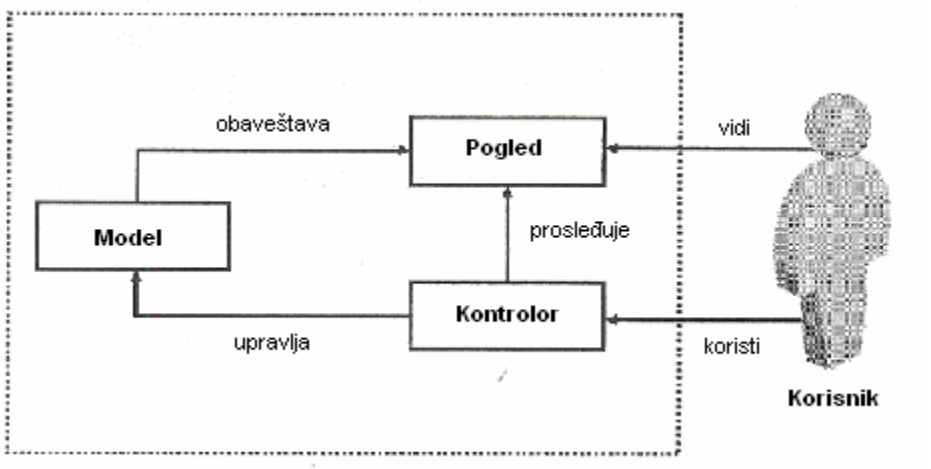
1. Oracle ADF i Oracle JDeveloper 10g

Oracle ADF je Java EE okruženje (*framework*) koje olakšava razvoj aplikacija, pružajući infrastrukturne servise, vizuelan i deklarativan način razvoja.

Oracle Application Development Framework ima sledeće osnovne karakteristike:

- Omogućava programerima da se fokusiraju na aplikaciju, a ne na konfiguraciju
- Kreira održivi kod za višestruku upotrebu
- Implementira dizajn šablone (paterne) koji su korisni za upotrebu pri razvoju aplikacija.

Jedna od često korišćenih arhitektura je MVC šablon (Model-View-Controller) za razvoj aplikacija, slika 1.1.



Slika 1.1 - MVC šablon

MVC arhitektura razdvaja rukovanje domenom modela (*Model*), prezentaciju modela (*View*) i tumačenje korisničkog ulaza (*Controller*).
U MVC arhitekturi, korisnički ulaz, poslovna logika i povratni rezultat korisniku su eksplicitno odvojeni i obrađuju ih tri vrste objekata, od kojih svaki ima određenu ulogu u aplikaciji.

- **ADF Model** je deo ADF sloja (lejera) za kontrolu podataka. Model aplikacije sadrži podatke i operacije koje upravljaju podacima i vrše njihovo modifikovanje. Model obaveštava pogled o promeni svog stanja i odgovara na instrukcije za promenu istog (uglavnom od kontrolora).

- **Pogled (*View*)** određuje kako podaci iz modela treba da budu prikazani. Kontrolor prosleđuje korisnički ulaz pogledu.
  (Na primer, ako pogled treba da prikaže neke redove iz baze, sam upit je ostavljen modelu, a prikaz podataka za korisnika je ostavljen pogledu).

- **Kontrolor** tumači korisničke ulaze mišem i tastaturom zahtevajući promene modela i pogleda na odgovarajući način. Kontrolor šalje korisničke zahteve i selektuje poglede za predstavljanje. On određuje šta se dešava u interakciji korisnika i komponenti (npr. šta se pojavljuje kada korisnik klikne na dugme).

Osnovne karakteristike alata **Oracle JDeveloper 10g** su sledeće: obezbeđuje integrisano razvojno okruženje (IDE), omogućava razvoj, kompajliranje i pokretanje Java aplikacija, kao i korišćenje grafičkog alata za pomoć pri razvoju izvornog (*source*) koda.

Proces razvoja ADF aplikacija obuhvata: definisanje poslovnih zahteva (definiše i modelira bazu), kreiranje objekata koji su u uzajamnoj vezi sa bazom (gradi objekte podataka), konstruisanje korisničkog interfejsa i sjedinjavanje u poslovnu logiku (vezuje

korisnički interfejs i podatke, gradi logiku i tok aplikacije). JDeveloper koristi vizuelno okruženje da ispuni ove zahteve.

1.1 ADF Komponente

Za razvoj Java aplikacija, korišćenjem podataka iz baze, mora da se izvrši preslikavanje Java objekata na relacione tabele. Pored toga, mora da se upravlja transakcijama i da se obezbede adekvatne performanse. To može da se postigne korišćenjem ADF poslovnih komponenti (*ADF Business Components,* skraćeno ADF BC)

Karakteristike ADF komponenti su sledeće:

- sadrže poslovnu logiku za višestruku upotrebu,
- upravljaju transakcijama podataka,
- razdvajaju poslovnu logiku od predstavljanja podataka,
- arhitektura je zasnovana na komponentama
- koriste SQL,
- obezbeđuju većinu koda koji je potreban u aplikaciji.

ADF-BC povećavaju produktivnost tako što implementiraju komponente za višestruku upotrebu. JDeveloper koristi alate za automatsku pomoć (*wizards*) za kreiranje i dodavanje komponenti i čini kreiranje objekata brzim i lakim.

ADF-BC čine tri glavne komponente:

1) **Entiteti** – uopšteno jedan entitet odgovara jednoj baznoj tabeli.
2) **Pogledi** – su kolekcije koje kontrolišu šta programer želi da se prikaže na strani. U globalu, kod pogleda se koristi SQL upit.
3) **Aplikacioni modul** – izlaže ono što korisnik hoće da vidi. Aplikacioni modul je zadužen za upravljanje transakcijama.

Postoje još dve važne komponente, pored prethodno navedenih glavnih komponenti:

4) **Asocijacije** – povezuju entitet-objekte. Automatski se po jedna asocijacija kreira za svaki strani ključ u bazi.
5) **Pogled veza (*View Link*)** – spaja poglede objekata i uglavnom se koriste za vezu glavni-podređeni (*master-detail*).

Nakon kreiranja konekcije sa bazom, komponente se generišu pomoću posebnog alata (čarobnjaka) za generisanje ADF-BC komponenti (*Business Components Wizard*). Ovaj alat kreira tri glavne komponente: entitet, pogled i aplikacioni modul.

U sledećem poglavlju, na primeru kreiranja modula „Inostrana blagajna“, biće prikazani dijagrami komponenti (BC dijagrami) koji obezbeđuju grafičku prezentaciju entiteta,

pogleda i aplikacionog modula, kao i veze između njih i njihovih nasleđenih struktura. Asocijacije se koriste za povezivanje entiteta na dijagramu. Pravac asocijacije identifikovan je strelicom. Pogledi su međusobno povezani pogled vezama (*View Links*).

1.1.1 Entiteti

Entiteti se koriste za definisanje logičke strukture posla. Entitet predstavlja baznu tabelu. Sadrži atribute koji odgovaraju kolonama u baznoj tabeli.

Pri kreiranju komponenti aplikacije, prvo se prave entiteti. Svaki entitet predstavlja objekat u aplikaciji. Pre pamćenja promena u bazi (pre izvršavanja *commit* naredbe), promene prvo vide entiteti.

Posle kreiranja konekcije na bazu, pomoću čarobnjaka *Create Database Connection Wizard*, kreira se prostor za aplikaciju (*Application workspace*). Klikom na File->New->General->Application Workspace, bira se ime aplikacije i lokacija na kojoj će se nalaziti. Zatim se kreiraju odgovarajući projekti u okviru njega, tako što se na File->New->General->Empty Project izabere ime projekta, ime paketa i lokacija na kojoj će se taj projekat nalaziti. (U našem primeru, projekat *ADFInostranaBlagajna* sadrži sve potrebne komponente). Zatim sledi kreiranje entiteta pomoću čarobnjaka *Create Entity Object Wizard*, gde se zadaje ime entiteta, unosi se ime paketa u kojem će se entitet nalaziti i bira se odgovarajuća tabela na koju će se isti odnositi.

Kad se kreira aplikacija, JDeveloper generiše dva fajla za svaki entitet:

Entity.xml – fajl koji sadrži meta podatke (*metadata*) za kreiranje komponenti.

EntityImpl.java – entitet klasa tj. klasa koja implementira entitet. Predstavlja red podataka. Sadrži metode *get* (vraćaju vrednost) i *set* (postavljaju vrednost) za svaki atribut entiteta. Ove metode se generišu automatski. Svi entiteti nasleđuju ovu klasu. Ova klasa pravi i metode za ubacivanje, menjanje, brisanje i zaključavanje redova. Koristi se za upravljanje instanci svakog entiteta.

Neke od metoda koje mogu da se predefinišu u klasi EntityImpl.java su sledeće:

doDML(), koja se izvršava nakon što se izvrše sve provere atributa i entiteta,

beforeCommit() (za proveru više instanci istog entiteta), *remove*() (brisanje entiteta).

Predefinisanje metode se vrši tako što se prvo selektuje odgovarajući java fajl i na toolbar-u (paleti sa alatkama) se izabere *Tools*, a zatim opcija *Override Methods*, gde se selektuju metode koje želite da predefinišete.

1.1.2 Asocijacije

Asocijacije definišu veze između entiteta. One olakšavaju pristup podacima u povezanim entitetima. Mogu biti zasnovane na ograničenjima (*constraint*s) u bazi, a mogu biti i nezavisne od njih. Čine ih glavni i podređeni entitet (*master* i *detail*). U daljem tekstu, primer asocijacije je *IblstavkaIbldokumentAssoc* asocijacija.

JDeveloper automatski kreira asocijacije između entiteta koji predstavljaju tabele u bazi, koje su međusobno povezane referencijalnim integritetom.

1.1.3 Pogledi

Pogled može biti zasnovan na entitetu. U daljem primeru takvi pogledi su sledeći: *IblDnevnicaVw* (zasnovan na *IblDnevnicaEnt*), *IblDnevniceStavkaVw* (zasnovan na *IblstavkaEnt*), *IbldokumentVw* (zasnovan na *IbldokumentEnt*), *IblstavkaVw* (zasnovan na *IblstavkaEnt*).

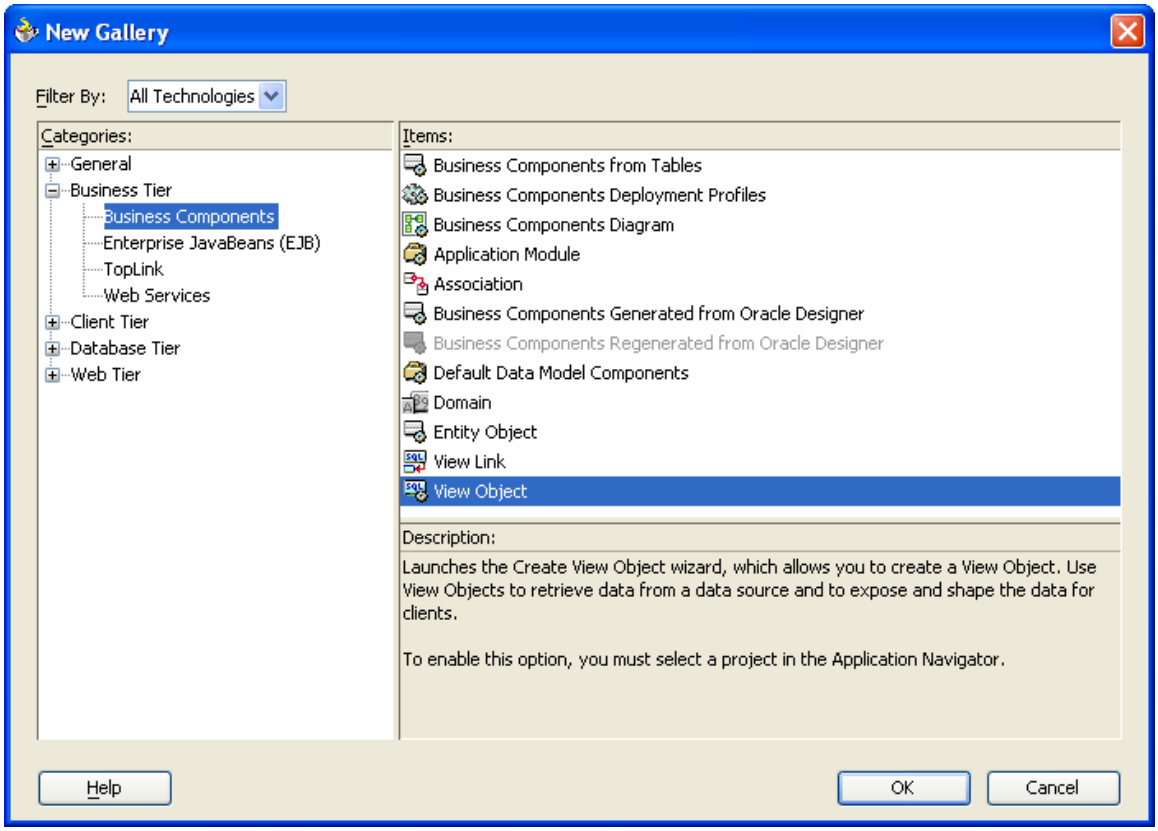
Atributi pogleda prikazuju atribute odgovarajućeg entiteta.

Pogled koristi SQL upit za spajanje, filtriranje, projektovanje i sortiranje podataka.

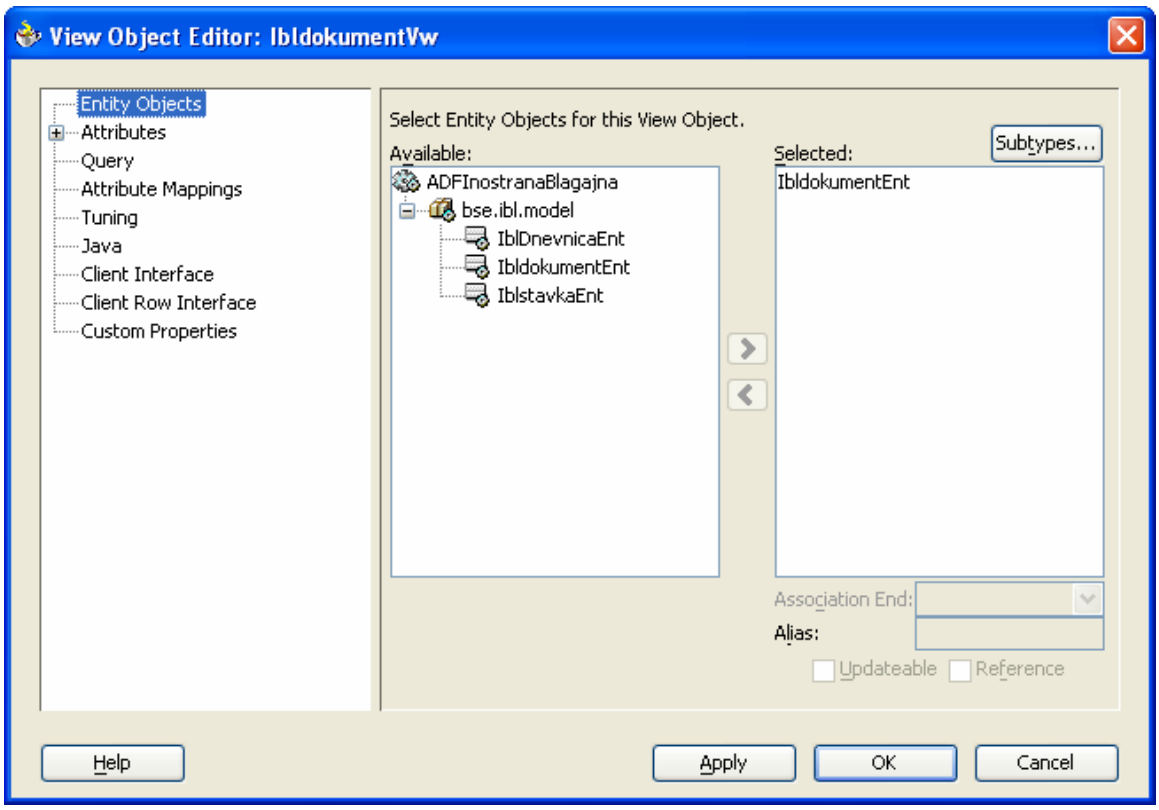
Za kreiranje pogleda se koristi čarobnjak *Create View Object Wizard* (desni klik na odgovarajući projekat, izabere se opcija *New* i otvara se prozor kao na slici 1.2, gde se selektuje pogled objekta kao komponenta koju želimo da kreiramo), navodi se ime pogleda i paketa u kojem će taj pogled da se nalazi. Pri tome pogled ne mora da bude u istom paketu kao i entitet na koji se odnosi ili modul koji ga sadrži. (U našem primeru, pogledu *IblDnevniceStavkaVw* odgovara modul *IblDnevnicaAm*, a entitetu *IblstavkaEnt*, na koji se on odnosi, modul *IbldokumentAm*). Kada se napravi pogled, duplim klikom na njega, otvara se prozor kao na slici 1.3, gde se bira entitet na koji će pogled da se odnosi.

Na ovoj slici se nalazi i polje za selektovanje „*Updateable*“. Ukoliko se ne selektuje ovo polje, sprečava se da pogled menja bilo koje atribute entiteta, a ako je potrebno da korisnik vrši promene podataka nad pogledom, pravi se promenljiv pogled (*updateable view*) i vrši se automatsko sinhronizovanje podataka sa drugim instancama pogleda (promene se odražavaju na odgovarajuće entitete).

Selektovanjem polja „*References*“, informacije iz entiteta se tretiraju kao reference u pogledu.

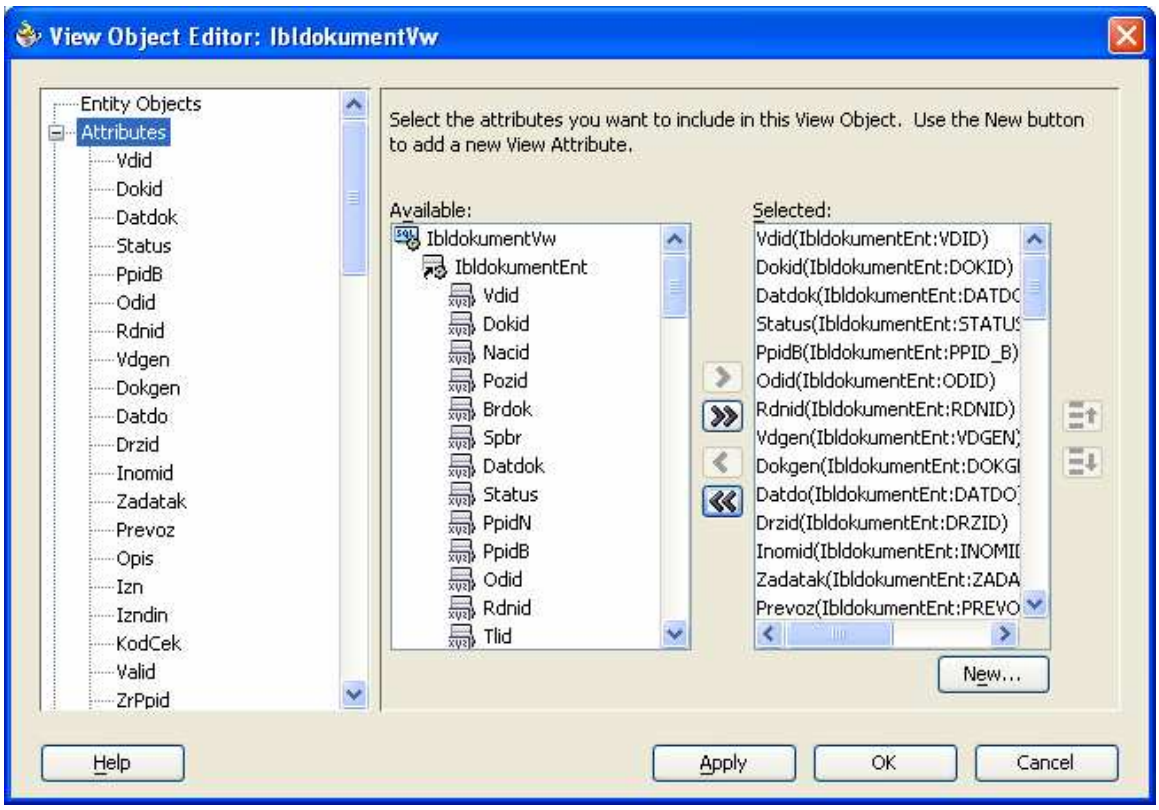

New Gallery
Filter By:
All Technologies
Categories:
General
Business Tier
Business Components
Enterprise JavaBeans (EJB)
TopLink
Web Services
Client Tier
Database Tier
Web Tier
Items:
Business Components from Tables
Business Components Deployment Profiles
Business Components Diagram
Application Module
Association
Business Components Generated from Oracle Designer
Business Components Regenerated from Oracle Designer
Default Data Model Components
Domain
Entity Object
View Link
View Object
Description:
Launches the Create View Object wizard, which allows you to create a View Object. Use View Objects to retrieve data from a data source and to expose and shape the data for clients.
To enable this option, you must select a project in the Application Navigator.
Help
OK
Cancel

Slika 1.2



Slika 1.3

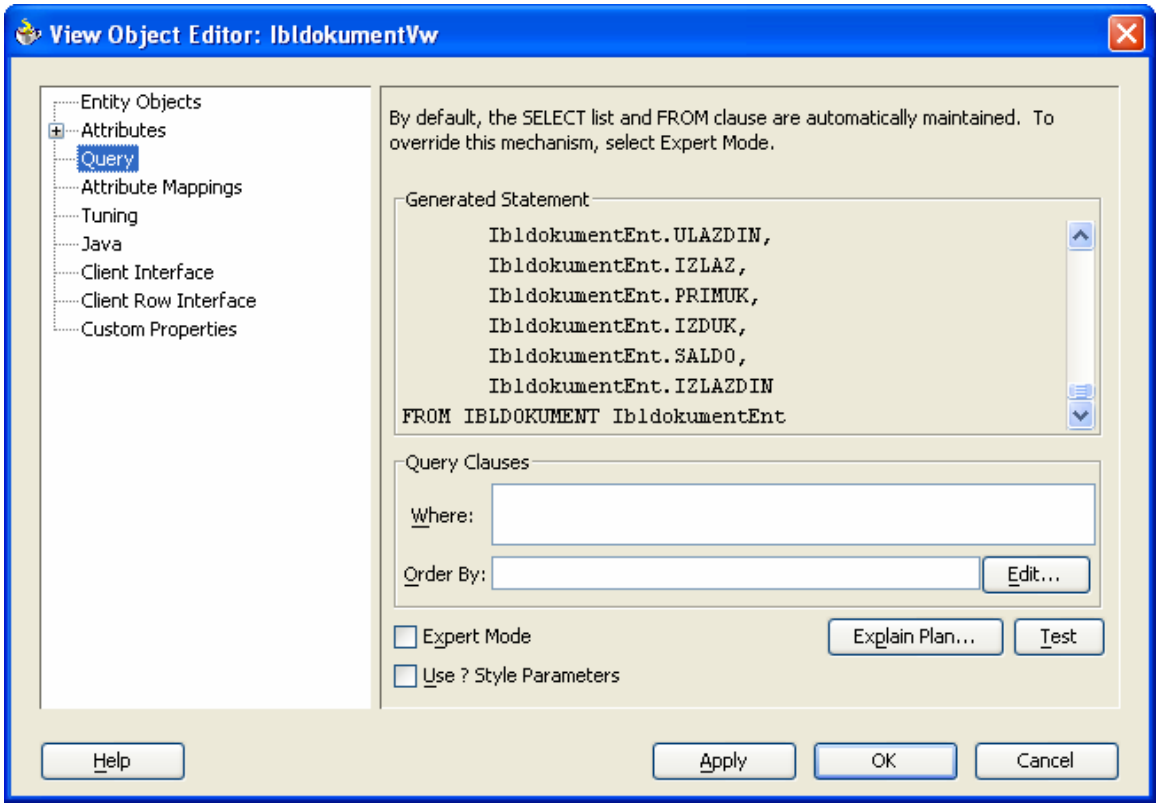
Opcijom *„Attributes“*, biraju se atributi entiteta koje želimo da naš pogled uključuje, slika 1.4 (na slici su predstavljeni svi atributi odgovarajućeg entiteta, a selektovani su oni koje želimo da naš pogled ima i strelicama su prebačeni u krajnji desni deo prozora).



Slika 1.4

Klikom na polje „*New*“, može da se definiše atribut pogleda koji nije zasnovan na atributu entiteta.

Pogledi preuzimaju podatke iz baze korišćenjem SQL upita, slika 1.5.

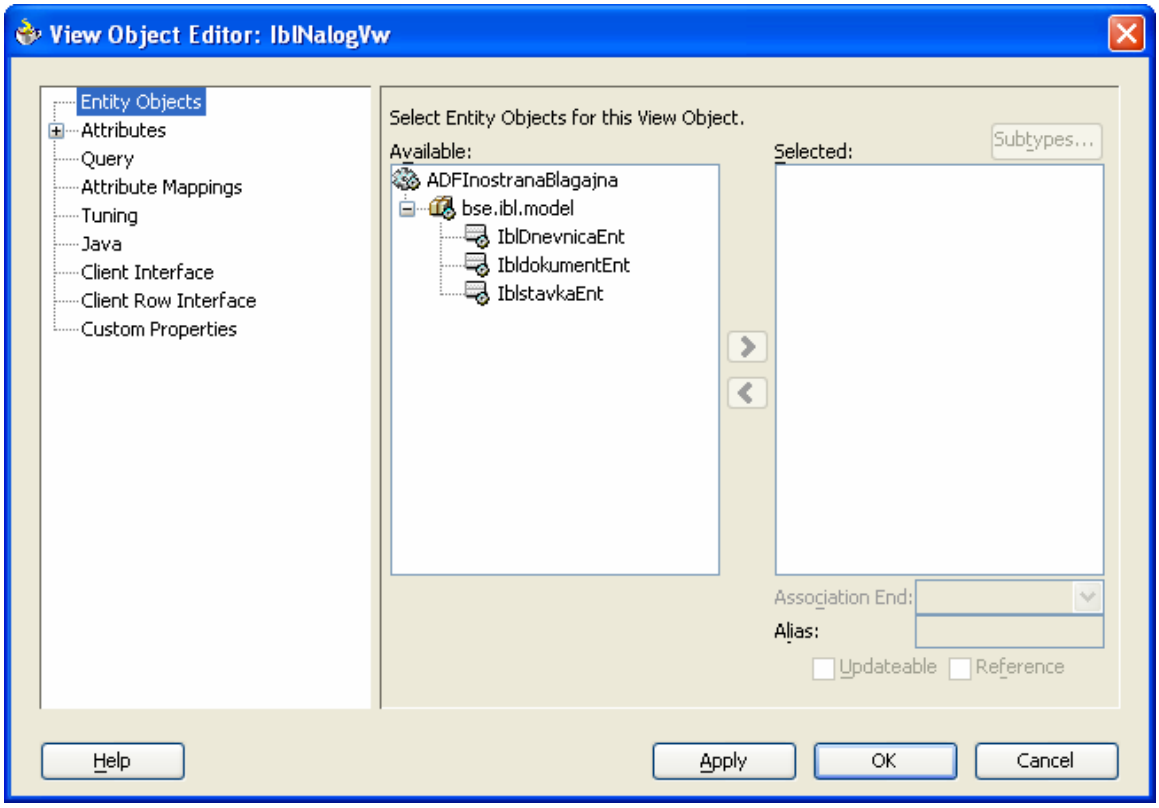


Slika 1.5

Select lista i **from** iskaz se automatski kreiraju na osnovu entiteta na kojem se pogled zasniva. Upit može da se izmeni selektovanje opcije „*Expert Mode*“, koja je korisna ako želimo kompletnu kontrolu nad SQL-upitom.

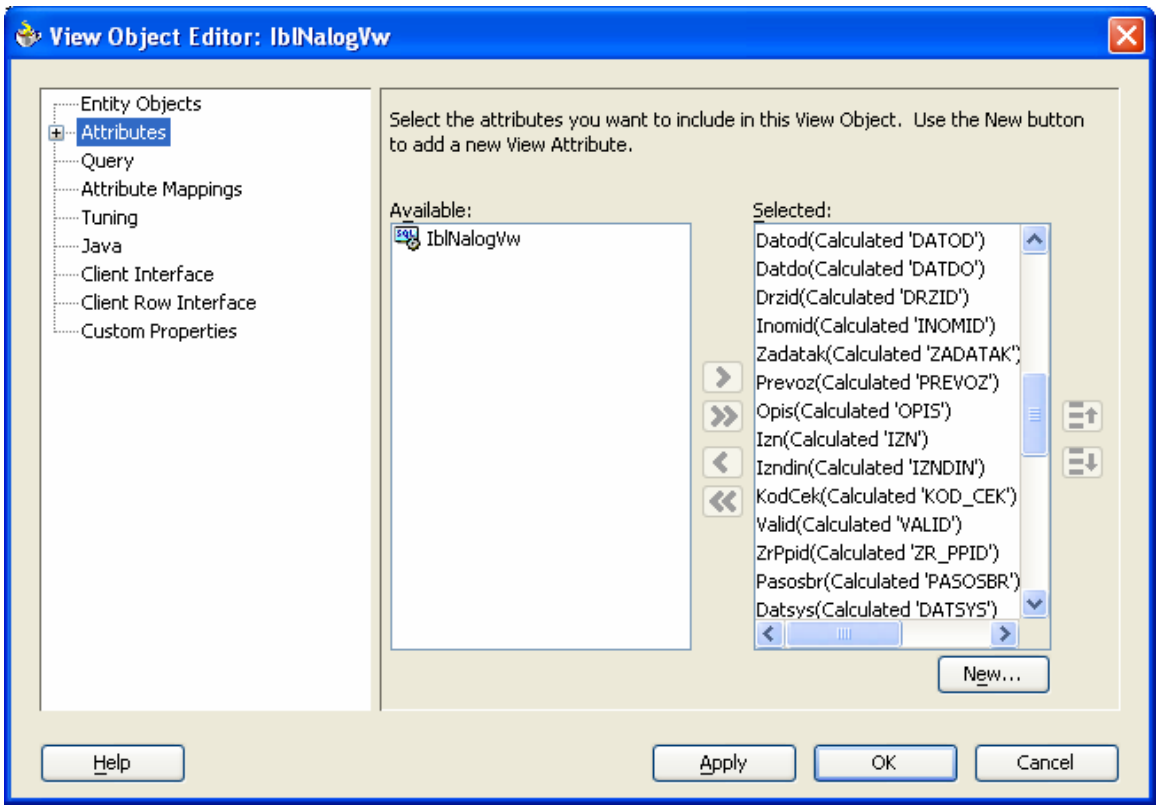
Kolone u SQL upitu odgovaraju atributima entiteta, tako da je bitan njihov redosled u upitu i treba da odgovara redosledu atributa. Mogu da se dodaju *Where* i *Order By* iskazi, ako su potrebni.

U slučaju kada je samo potreban pogled za prikaz podataka, koriste se pogledi samo za čitanje (*read-only*). SQL – pogledi ovog tipa čuvaju promene samo u memoriji i nisu postojani u bazi. Oni se zasnivaju samo na SQL upitu i ne odnose se na neki entitet (slika 1.6). Ne mogu da se koriste za ubacivanje, menjanje, brisanje podataka. U našem primeru takvi pogledi su: *IblNalogVw* i *IblputninalogVw*.



Slika 1.6

Na slici 1.7 vide se atributi koji se dobijaju iz sql upita (bira se stavka „*Attributes*“).



Slika 1.7

Izborom opcije *„Java“* (na prozoru sa prethodne slike) određuje se da li se generiše Java fajl za klasu objekta pogleda (*view object class*) ili za klasu reda pogleda (*view row class*) ili za oba.

a) **ViewObjectImpl** je klasa objekta pogleda i njega i predstavlja. Koristi se za dodavanje i predefinisanje ponašanja koja se odnose na objekat pogleda.
Ova klasa sadrži metode koje se odnose na ceo pogled,
npr. *setWhereClause*() – za postavljanje where klauze u SQL upitu,
findBykey() – nalazi sve redove u pogledu koji odgovaraju datom ključu,
first(), *last*(), *next*() – predstavljaju pozicije redova u pogledu;

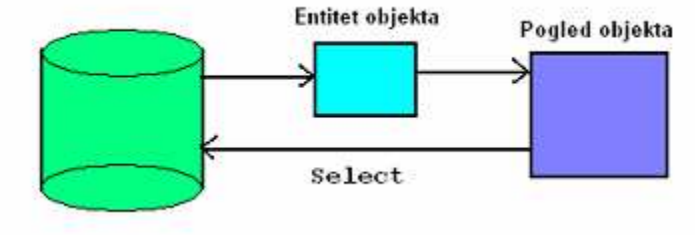
b) **ViewRowImpl** je klasa reda pogleda i njega predstavlja. Koristi se za dodavanje, menjanje ponašanja koja se odnose na red pogleda.
Ova klasa sadrži *get* i *set* metode za uzimanje i postavljanje vrednosti atributa u pogledu.

Obe ove klase se nalaze u paketu **oracle.jbo.server**.
Kada se kreira aplikacija, JDevoloper automatski generiše dva fajla za svaki objekat pogleda:
View.xml – sadrži meta podatke za komponente,

ViewImpl.java – predstavlja klasu objekta pogleda koju nasleđuje klasa ViewObjectImpl.

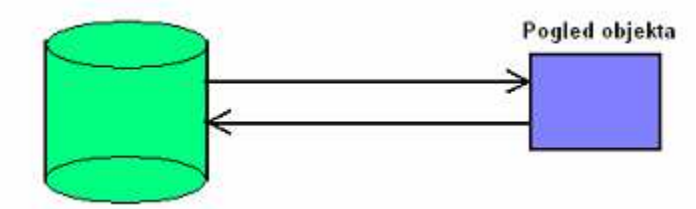
1.1.4 Interakcija između Entiteta i Pogleda: vraćanje i menjanje podataka

Pogled se direktno preko upita odnosi na bazu, tj. dobija podatke koristeći SQL upit i uglavnom je vezan za entitet. Podaci koje upit vraća su sačuvani u keš memoriji objekta entiteta, a zatim se šalju objektu pogleda (slika 1.8):



Slika 1.8

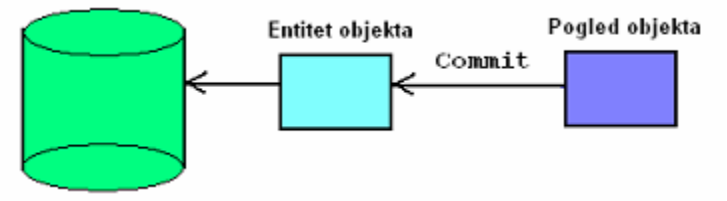
Ukoliko se atribut zasniva samo na pogledu (*Calculated attributes*), tj. ako se dobija SQL upitom pogleda i nije u vezi sa entitetom, onda se takav podatak (netrajni atribut) smešta u keš memoriju objekta pogleda.



Slika 1.9

Pogledi menjaju entitete, a entiteti vrše pamćenje promena u bazi.

Promenom podataka u bazi, pogled menja keš objekta entieta. Kada se transakcija potvrdi (*commit*), entitet vrši ažuriranje (*update)* baze. Entitet garantuje validnost podataka koji su potvrđeni u bazi.



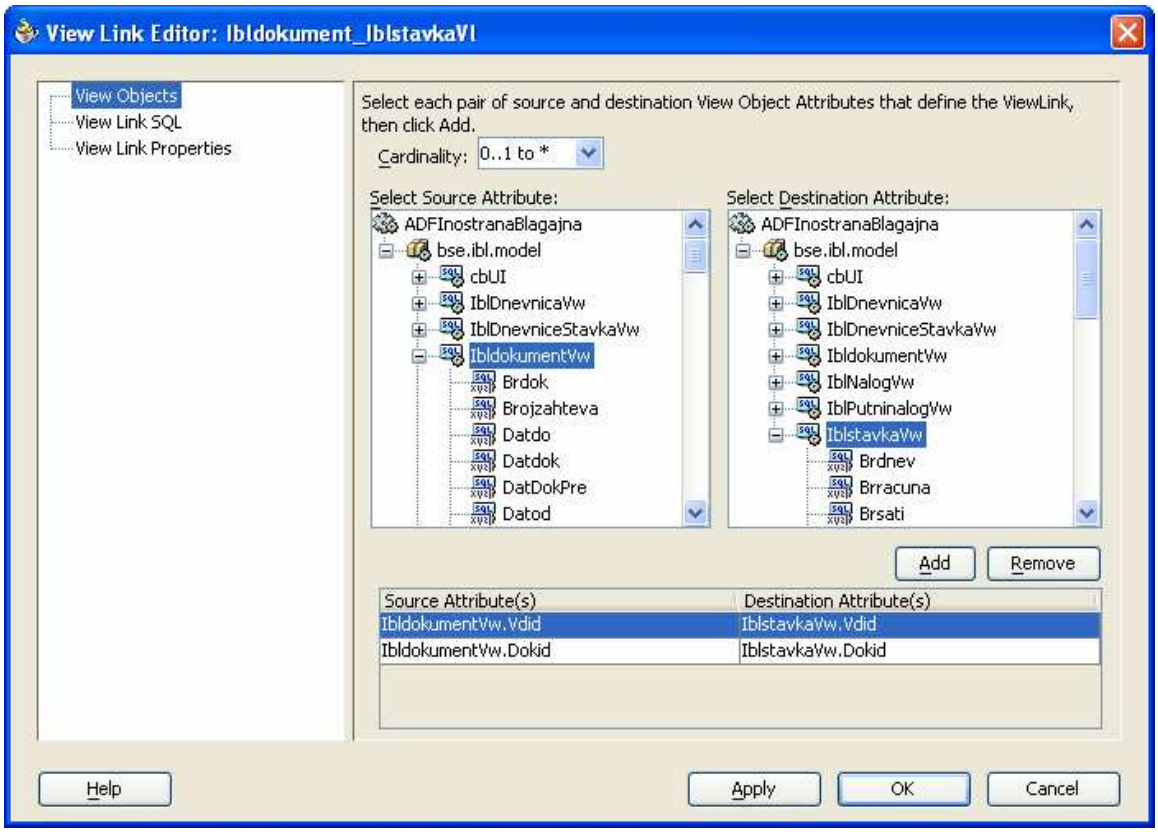
Slika 1.10

1.1.5 Povezivanje pogleda (View Links)

Veza Pogleda (*View Link*) je aktivna veza između objekata pogleda. Pri kreiranju pogled-veza, potrebno je odrediti izvorni (*source*) i ciljni (*destination*) pogled, kao i izvorne i ciljne atribute. Bira se i kardinalnost.

(Desnim klikom na odgovarajući paket izabere se opcija *New View Link* i otvara se prozor kao na slici 1.11).

Pogledi su uglavnom grupisani u veze glavni-podređeni (*master-detail*) i povezani su pogled vezama. U primeru modula „Inostrana blagajna“, postoje sledeće veze pogleda: *Ibldokument_IblstavkaVl*, *Ibldokument_IblPutninalogVl*, *Ibldokument_IblNalogVl*, *Ibldokument_IblDnevnicestavkaVl*.

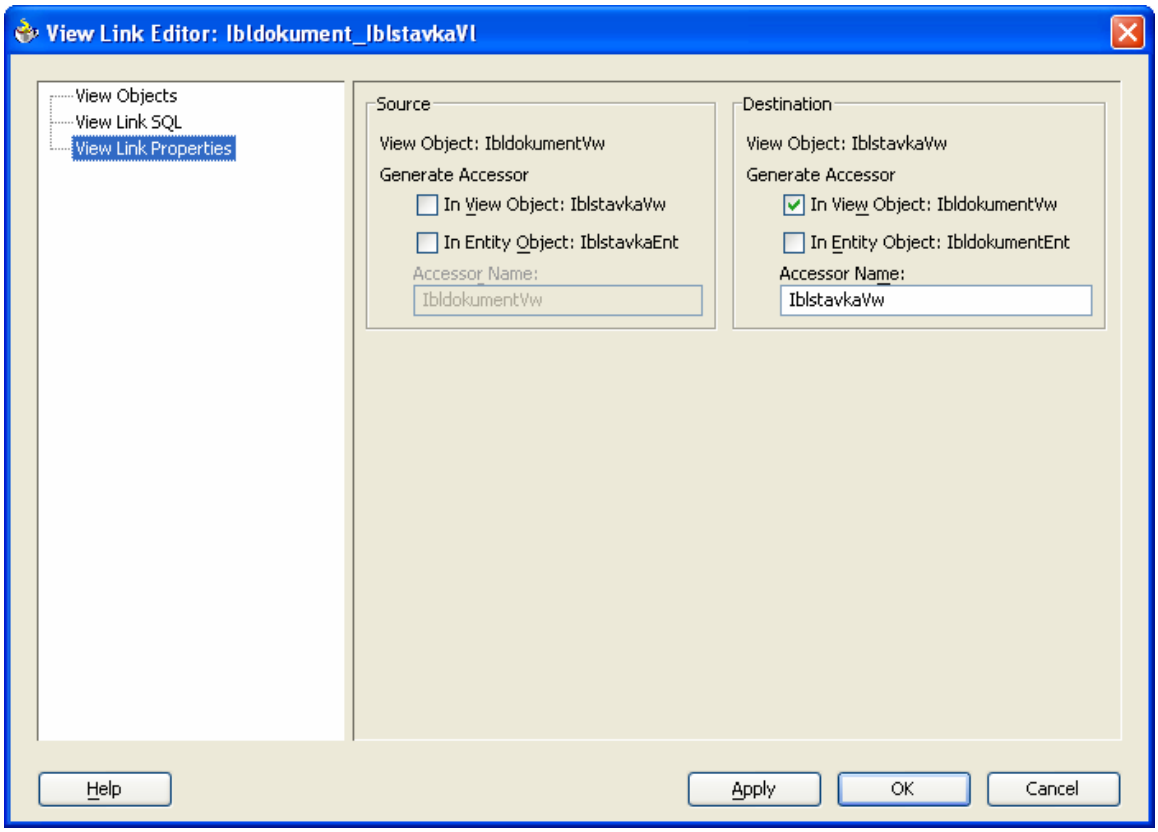


Slika 1.11

Glavni objekat pogleda (npr. *IbldokumentVw*), koji se odnosi na entitet, sadrži primarni ključ (ovde: vdid, dokid) koji odgovara stranom ključu ciljnog pogleda (IblstavkaVw).

SQL upit definiše pogled vezu između glavnog i podređenog pogleda.

Za osobine pogled-veza treba selektovati opciju za pristup pogledu ili entitetu, slika 1.12



Slika 1.12

1.1.6 Aplikacioni modul

Aplikacioni modul služi kao omot za poglede i entitete. On upravlja baznim transakcijama i izvršava zadatke specifične za aplikaciju, npr. obrađivanje korisničkih informacija.

Glavne karakteristike aplikacionog modula su sledeće:

- Predstavlja model podataka koji klijent koristi i ima jednu konekciju na bazu
- Čuva trag svih promena koje se tiču podataka u bazi
- Obezbeđuje metode sa daljinskim pristupom

Svaki aplikacioni modul ima model podataka koji se sastoji od pogleda i pogled-veza (entiteti i asocijacije su implicitno uključeni).

1) <AppMod>.xml – sadrži detaljne meta podatke o uključenim pogledima (uključuje i definicije atributa za sve poglede i njihove veze)
2) <AppMod>Impl.java – sadrži sve metode i njihova ponašanja
3) <AppMod>.java – sadrži deklaracije svih prilagođenih metoda
4) bc4j.xcfg – sadrži svu konfiguraciju i detalje o konekciji

Kada se potvrdi neka promena na aplikacionom modulu, ona se odnosi na sve uključene poglede u njemu, samim tim i na entitete. Transakcija je kao interfejs koji upravlja baznim operacijama. Transaction i DBTransaction su interfejsi koji definišu bazne transakcije. DBTransaction nasleđuje Transaction. Koriste se metode iz ovih interfejsa za pristup transakciji aplikacionog modula (npr. am.getDBTransaction().commit())

Važnije metode su sledeće:

commit – potvrđivanje transakcije; čuvaju se sve promene u bazi

rollback – poništavanje svih promena (poništava transakciju)

connect – za uspostavljanje konekcije na odgovarajuću URL adresu

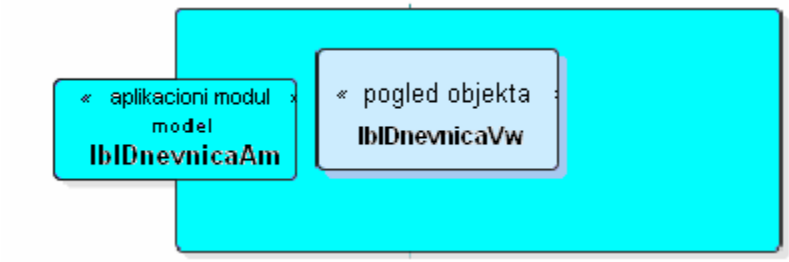
disconnect – za prekid veze servera sa bazom

Aplikacioni modul se kreira pomoću čarobnjaka *Create Application Module Wizard*. Prvo se definiše ime modula i paket u kojem će se nalaziti, zatim se bira pogled objekat na koji će modul da se odnosi.

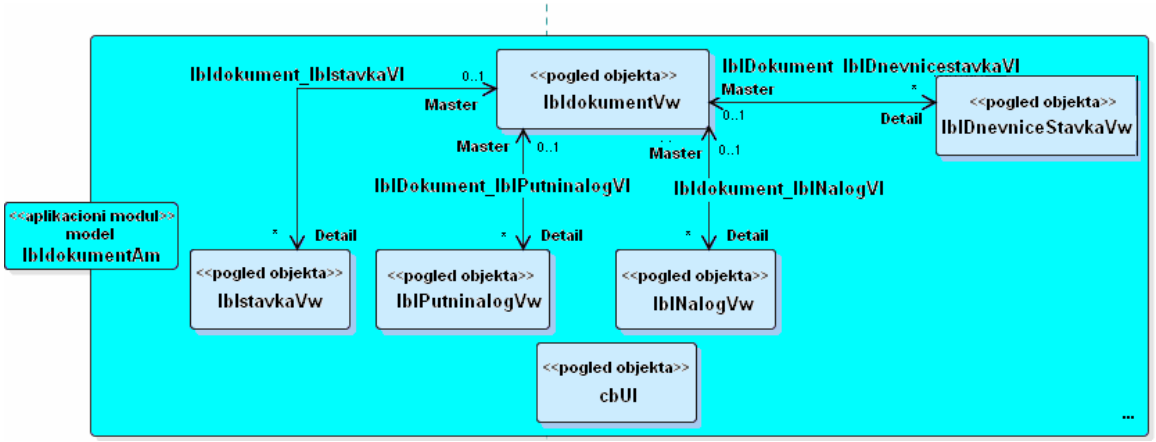
Aplikacioni modul treba da predstavlja jednu logičku jedinicu rada, tj. logički model za rešenje određenog korisničkog zadatka.

U primeru modula „Inostrana blagajna“, postoje dva aplikaciona modula:

IblDnevnicaAm (odnosi se na pogled *IblDnevnicaVw*)



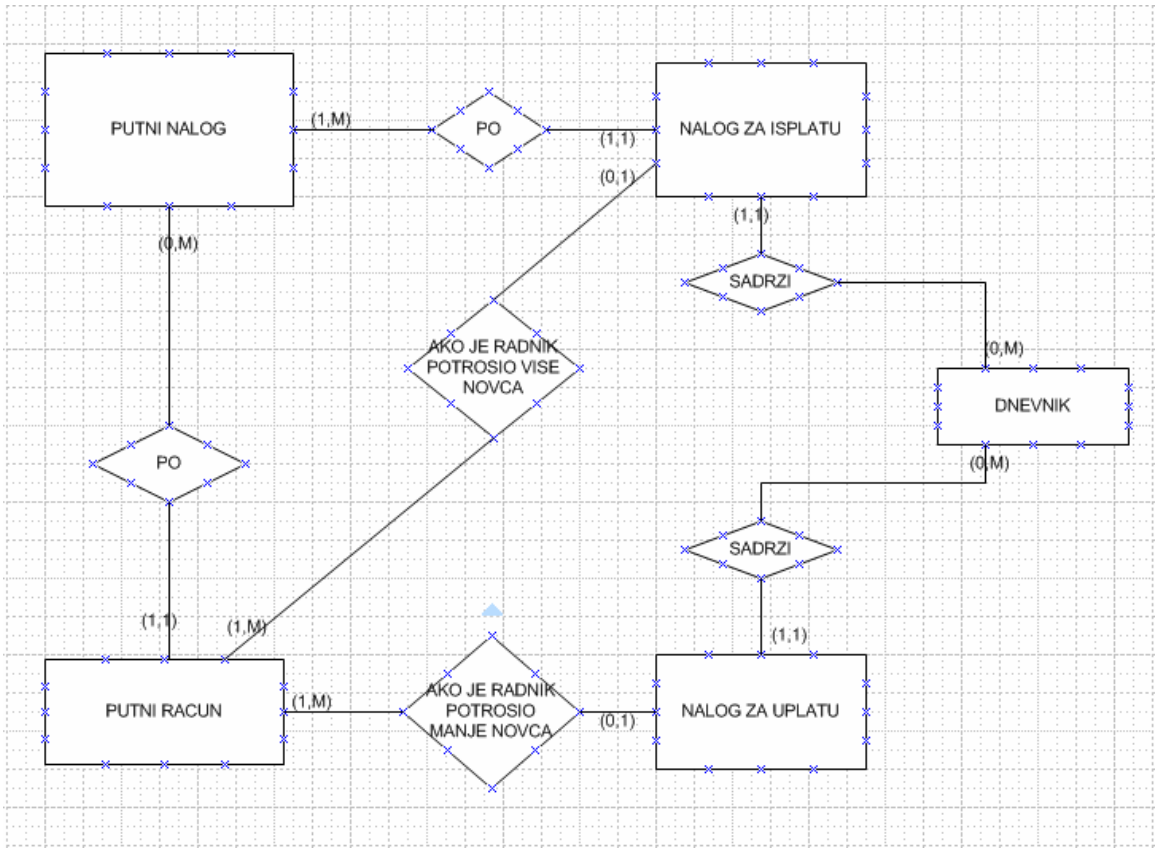
IbldokumentAm (odnosi se na pogled *IbldokumentVw*)



III MODUL INOSTRANA BLAGAJNA

Modul „Inostrana blagajna“ se sastoji od formiranja putnog naloga za određenu osobu (zaposlenog), kao davanje akontacije za službeni put u inostranstvo. Na osnovu ovog naloga se fromira putni račun koji služi za evidenciju putnih troškova po povratku sa službenog puta. Zatim se izdaje nalog blagajni novca radi devizne gotovinske uplate/isplate sa blagajne po različitim osnovama. Na kraju se kroz dnevnik blgajne prate dnevne promene u blagajni novca.

Dijagram poslovne logike modula „Inostrana blagajna“ prikazan je na sledećoj slici:



Slika 2.1 - Dijagram poslovne logike modula „Inostrana blagajna”

Modul „Inostrana blagajna“ implementiran je u tri celine:

Prvi deo je projekat ADFInostranaBlagajna u okviru kojeg postoji paket ibl.model, u kojem se nalaze svi potrebni entiteti, pogledi, aplikacioni moduli, pogled-veze i asocijacije.

Drugi deo je klijentski deo, AppInostranaBlagajna, u kojem je definisan paket ibl.app, gde se nalaze sve potrebne forme koje korisnik vidi, kao i paket ibl.action u kojem su definisane potrebne akcije (u našem primeru akcija obračunavanja dnevnika blagajne novca).

Treći deo je projekat LibInostranaBlagajna, u okviru kojeg su, u paketu ibl.lib, klase BlagajnaSqlExecutor.java i ControlSql.java koje predstavljaju vezu sa bazom. Tu se nalaze i klase InoBlagajnaKonstante.java za definisanje konstanti i klasa ValutaConvertor.java za konverziju novca iz jedne u drugu valutu.

U klasi ControlSql.java definišu se SQL upiti koji su potrebni za aplikacije (selektovanje, ubacivanje, menjanje, brisanje podataka). U klasi BlagajnaSqlExecutor.java (koja nasleđuje klasu SqlExecutor.java) se ti upiti izvršavaju, pozivom odgovarajućih metoda execute() , kojima se kao argument prosleđuje odgovarajući upit.

1. Izlged baze podataka

Usled povećanja složenosti poslovanja, povećavaju se i potrebe za promenama u informacionim tehnologijama. Danas se od IT profesionalaca traži da upravljaju većom količinom informacija i da ih blagovremeno dostavljaju svojim korisnicima, uz stalno povećavanje kvaliteta usluga.

Oracle Database 10g pruža osnov za uspešno dostavljanje velike količine informacija uz visok kvalitet usluga i smanjenje rizika od promena u okviru IT-a i efikasniju upotrebu IT budžeta.

U ovom primeru, kao alat korišćen je **Toad for Oracle** (Žabac)**.** To je moćan vizualni alat koji omogućava brz i lak razvoj baze, kao i aplikacija. On takođe olakšava poslove administracije baze podataka i nudi posebne opcije koje povećavaju produktivnost.

Modul „Inostrana blagajna“ se sastoji od šifarnika i dokumenata. Tabele su logički podeljene i njima su pridružene odgovarajuće šeme.

Pored šifarnika „Dnevnica“, koji se nalazi u šemi iblsif, postoje i šifarnici: država, inomesto, valuta, osoba, vrsta usluge, kurs, kursna lista. Oni se nalaze u šemi sif. Dve glavne tabele su ibldokument i iblstavka, koje se nalaze i šemi ibl.

Izgled baze (logički model) je prikazan na dijagramu 2.2. Dijagram prošireni model objekti i veze (*PMOV*) predstavlja koncept modula blagajna, glavne tabele: ibldokument, iblstavka, dnevnice i njihove veze sa drugim šifarnicima.

Model objekti-veze (*Entity-Relationship Model*) predstavlja najčešće korišćeni model za projektovanje relacionih baza podataka. Struktura MOV je predstavljena preko dijagrama objekti-veze. Na MOV-u se pored objekata (entiteta) i njihovih atributa vide veze i preslikavanja koja postoje između objekata. Objekat, kao elemenat strukture MOV predstavlja ili neki fizički objekat ili koncept realnog sistema.

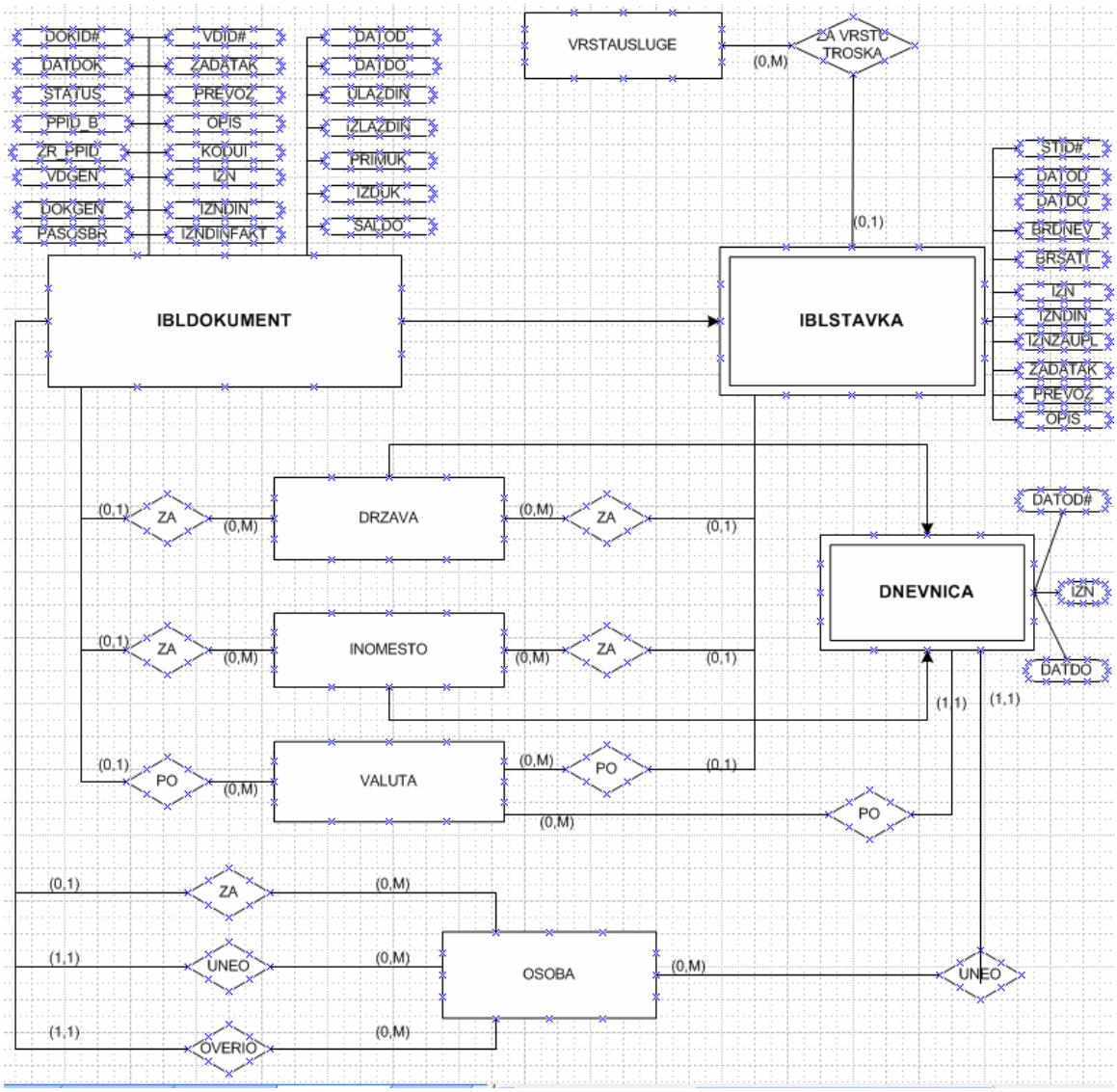
Vrste objekata u MOV-u su:

- Nezavisan objekat - ima osobinu koja ga može jednoznačno identifikovati.
- Zavisan objekat – onaj čija egzistencija i identifikacija zavise od drugog (ili drugih) objekata.
- Slab objekat – onaj koji se ponavlja više puta za određeni nezavisni objekat. Slab objekat je zavisan od postojanja jakog objekta (nezavisnog objekta) .
  (U ovom primeru to su *iblstavka* i *dnevnica*)

- Asocijativni objekat – predstavlja vezu više objekata.

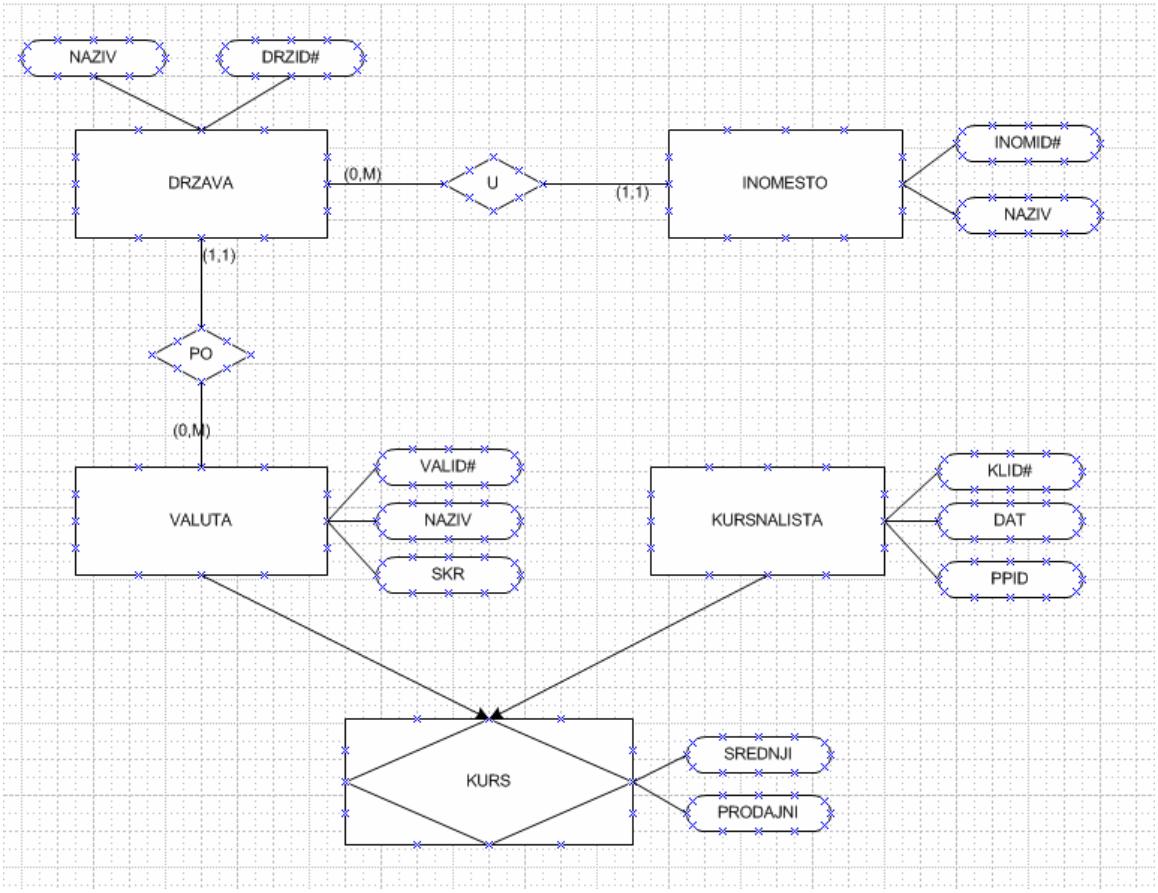
Veze u MOV opisuju način povezivanja (uzajamna dejstva) objekata. One se na dijagramu prikazuju romboidom.
Bitna karakteristika veze je kardinalnost preslikavanja. Zadaju se donja i gornja granica, koje redom predstavljaju najmanji i najveći mogući broj pojavljivanja jednog tipa objekta, za jedno pojavljivanje drugog tipa objekta (slika 2.2).

Objekti i veze mogu imati svoja svojstva, tj. atribute. Atributi su karakteristike ili osobine iskazane kao jedna ili više vrednosti koje opisuju objekat. Na dijagramu se atributi prikazuju elipsom.



Slika 2.2 - Dijagram pmov za koncept modula blagajna

Sledećim dijagramom pmov prikazane su neke tabele iz šeme sif, slika 2.3. Na ovom dijagramu može se primetiti da je entitet kurs, nastao agregacijum entiteta valuta i kursnalista. Agregacija je apstrakcija u kojoj se skup objekata i njihovih međusobnih veza tretira kao novi, jedinstveni agregirani tip. Objekti koji čine agregaciju (valuta, kursnalista) se nazivaju dekomponentama, pa se samim tim inverzan proces naziva dekompozicija.

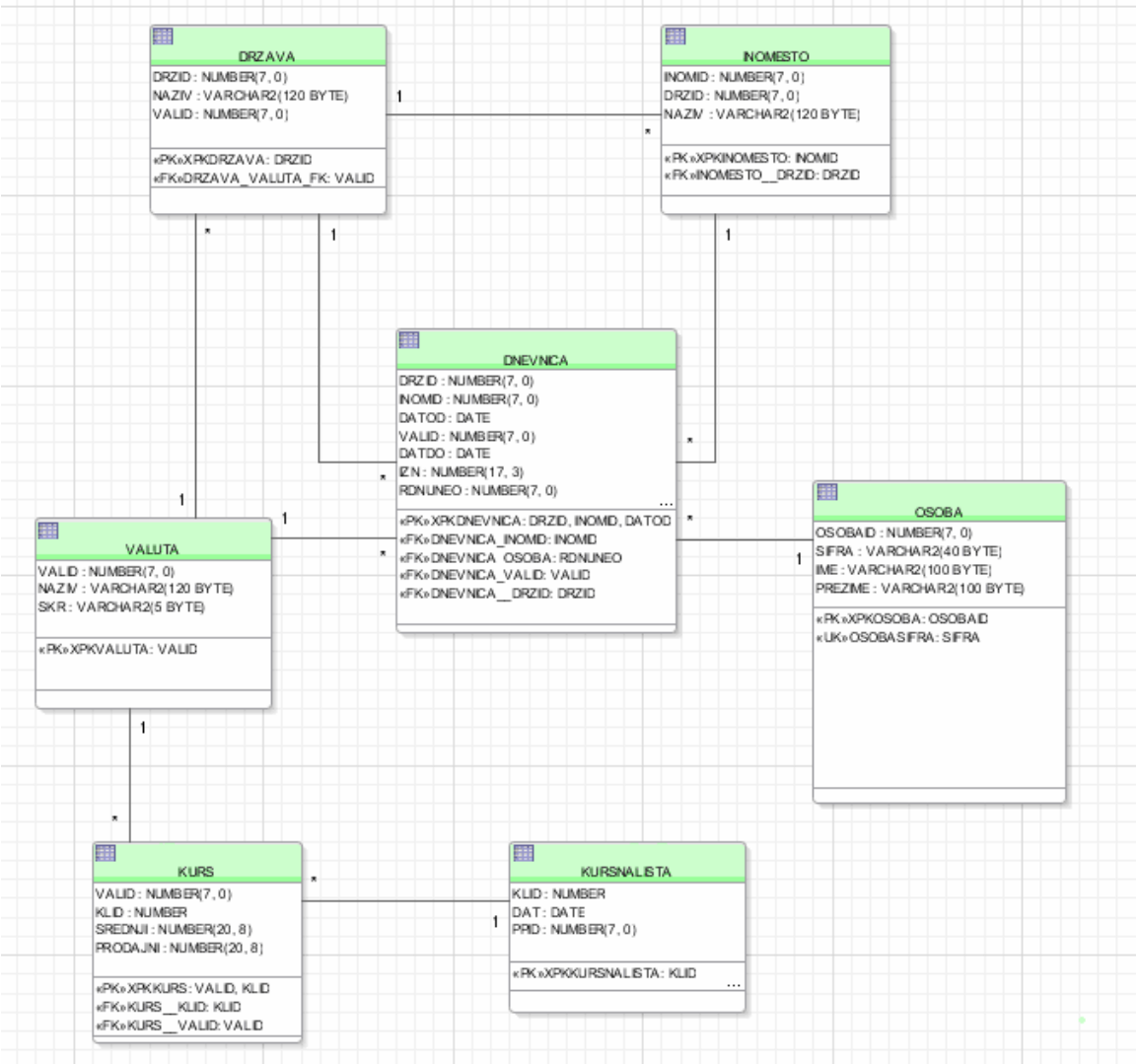


Slika 2.3 - Dijagram pmov za šifarnike

Za formiranje modela, na osnovu postojećih objekata iz baze podataka, koriste se *Database* dijagrami. Moguće je formirati neograničen broj dijagrama što omogućava formiranje modela baze podataka u skladu sa odgovarajućim logičkim celinama.

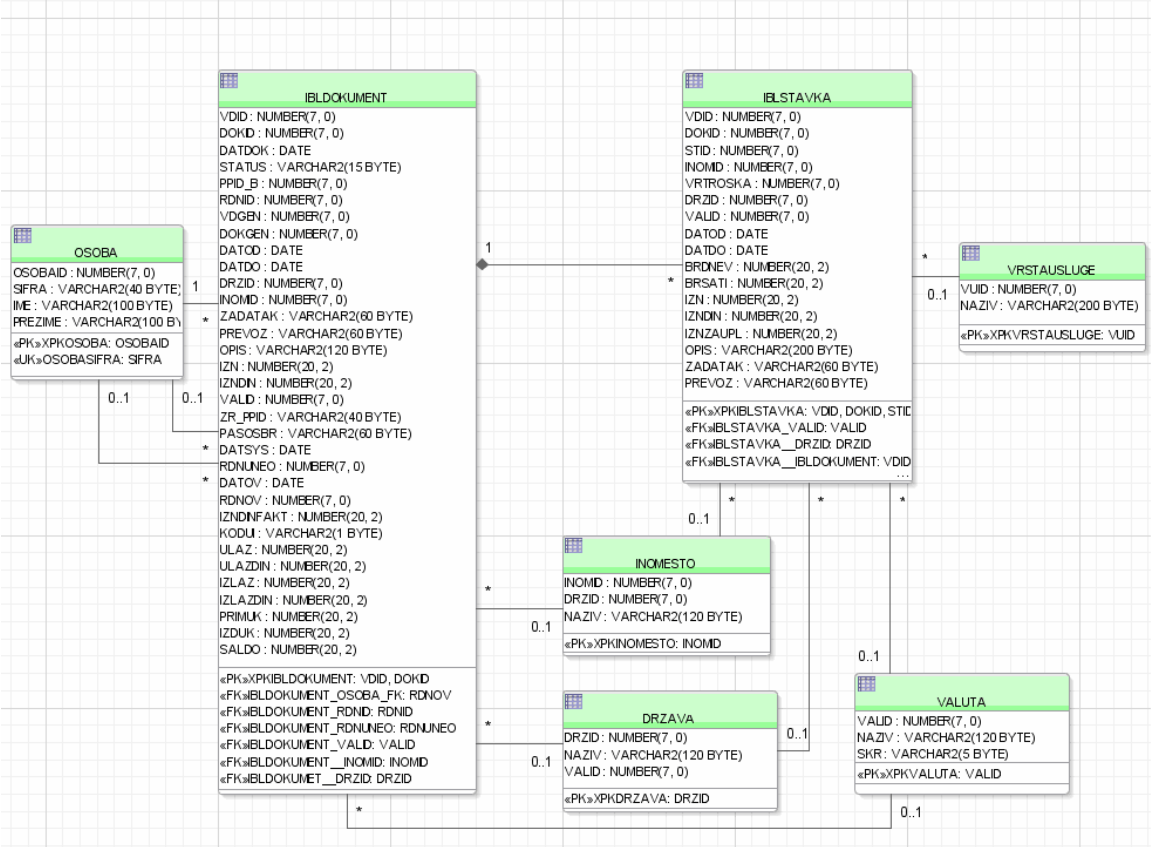
Na ovim dijagramima se mogu videti kolone koje tabela sadrži, njena ograničenja (primarni ključ, strani ključevi tabele) i veze sa drugim tabelama.

Bazni dijagram za koncept dnevnice prikazan je na slici 2.4. Njime je predstavljen izgled tabele dnevnica i njena veza sa drugim šifarnicima.



Slika 2.4 - Bazni dijagram za koncept dnevnica

Bazni dijagram za koncept ibldokument prikazan je na slici 2.5. Na ovom dijagramu su dati izgledi dve glavne tabele: ibldokument i iblstavka, njihova veza, kao i povezanost sa šifarnicima iz šeme sif.



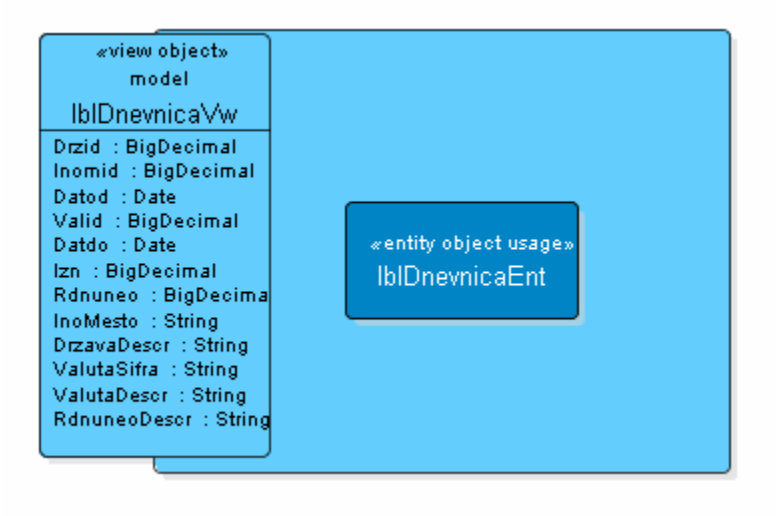
Slika 2.5 - Bazni dijagram za koncept ibldokument

2. Šifarnici

2.1. Dnevnice

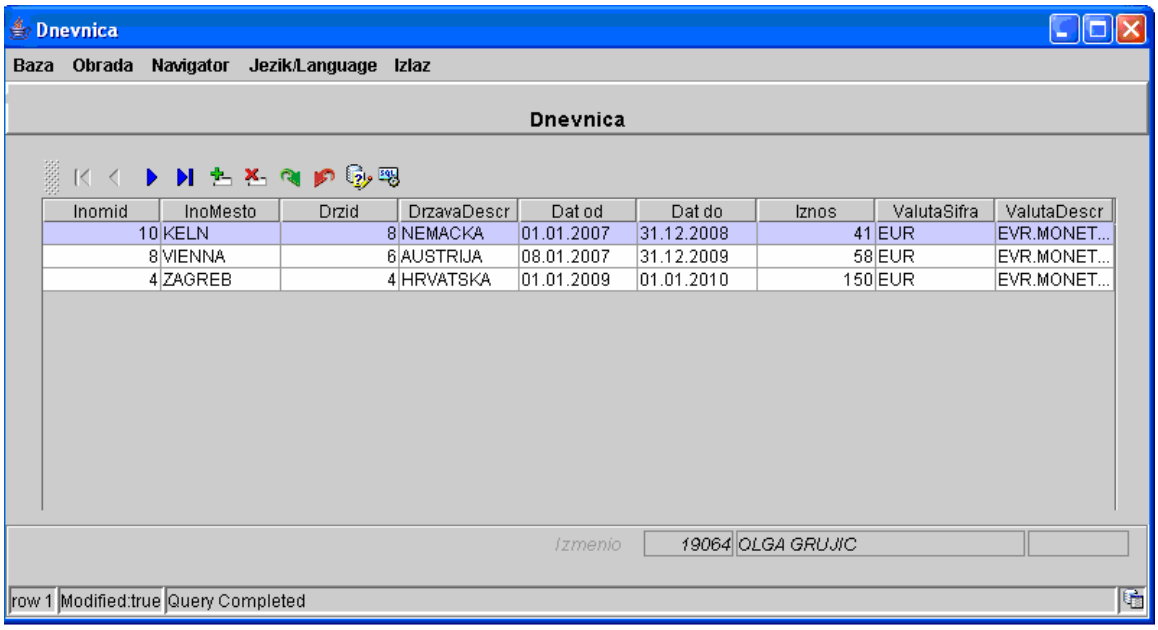
Šifarnik Dnevnice služi za evidenciju dnevnica. Unose se iznosi dnevnica u odgovarajućoj valuti za određeno mesto i državu, u određenom periodu važenja. Tabela u bazi, koja se koristi, je iblsif. DNEVNICA (prikazana na slici 2.4).

Na narednoj slici prikazan je ADF model podataka tj. BC dijagram (*Business Component* dijagram) za šifarnik dnevnica. Na njemu se vidi odgovarajući entitet, pogled koji ga koristi i atributi koje taj pogled sadrži.



Slika 2.6 - ADF model - Dnevnice

Izgled ekranske maske aplikacije za održavanje šifarnika dnevnica prikazan je na slici 2.7:



Slika 2.7 - Ekranska maska aplikacije „Dnevnica”

2.1.1 Opis polja i dozvoljene operacije

U narednoj tabeli dati su opisi polja sa forme „Dnevnica“.

Naziv	Dnevnice
Inomid	Unosi se jedinstveni identifikator mesta (iblsif.dnevnica.inomid). Moguć je izbor iz liste vrednosti (uzimaju se podaci iz tabele inomesto).
InoMesto	Po unosu identifikatora mesta (inomid), ispisuje se naziv mesta (sif.inomesto.naziv)
Drzid	Unosi se jedinstveni identifikator države (iblsif.dnevnica.drzid). Moguć je izbor iz liste vrednosti (kupe se podaci iz tabele država).
DrzavaDescr	Po unosu identifikatora države (drzid) ispisuje se naziv države (sif.drazava.naziv).
Dat od	Unosi se datum početka važenja dnevnice (iblsif.dnevnica.datod).
Dat do	Unosi se datum kraja važenja dnevnice (iblsif.dnevnica.datdo).
Iznos	Unosi se iznos dnevnice (iblsif.dnevnica.izn).
ValutaSifra	Unosi se skraćenica za valutu (iblsif.dnevnica.valid). Moguć je izbor iz liste vrednosti (kupe se podaci iz tabele valuta).
ValutaDescr	Po unosi polja ValutaSifra ispisuje se naziv te valute (sif.valuta.naziv)

Naziv	Polja sa standardnog toolbar-a
Izmenio	Polje sadrži šifru, ime i prezime radnika koji je uneo dokument i datum unosa (kupe se podaci iz tabele osoba). (iblsif.dnevnica.rdnuneo)

Preko ove aplikacije moguće je izvršiti standardne operacije ažuriranja i prikazivanja.

2.2 Ostali šifarnici

Ostale šifarnike čine sledeće tabele:

Osoba: Služi za evidenciju osoba. U ovom primeru unose se podaci o osobi koja se upućuje na službeni put. Služi i za dobijanje informacija o osobi koja je unela ili overila dokument.

Valuta: Služi za evidenciju valute.

Kursna lista: Predstavlja evidenciju kursa na određeni datum i za određenu banku.

Kurs: Služi za evidenciju srednjeg i prodajnog kursa za određenu valutu, po određenom kursu iz kursne liste tj. za određeni dan i banku.

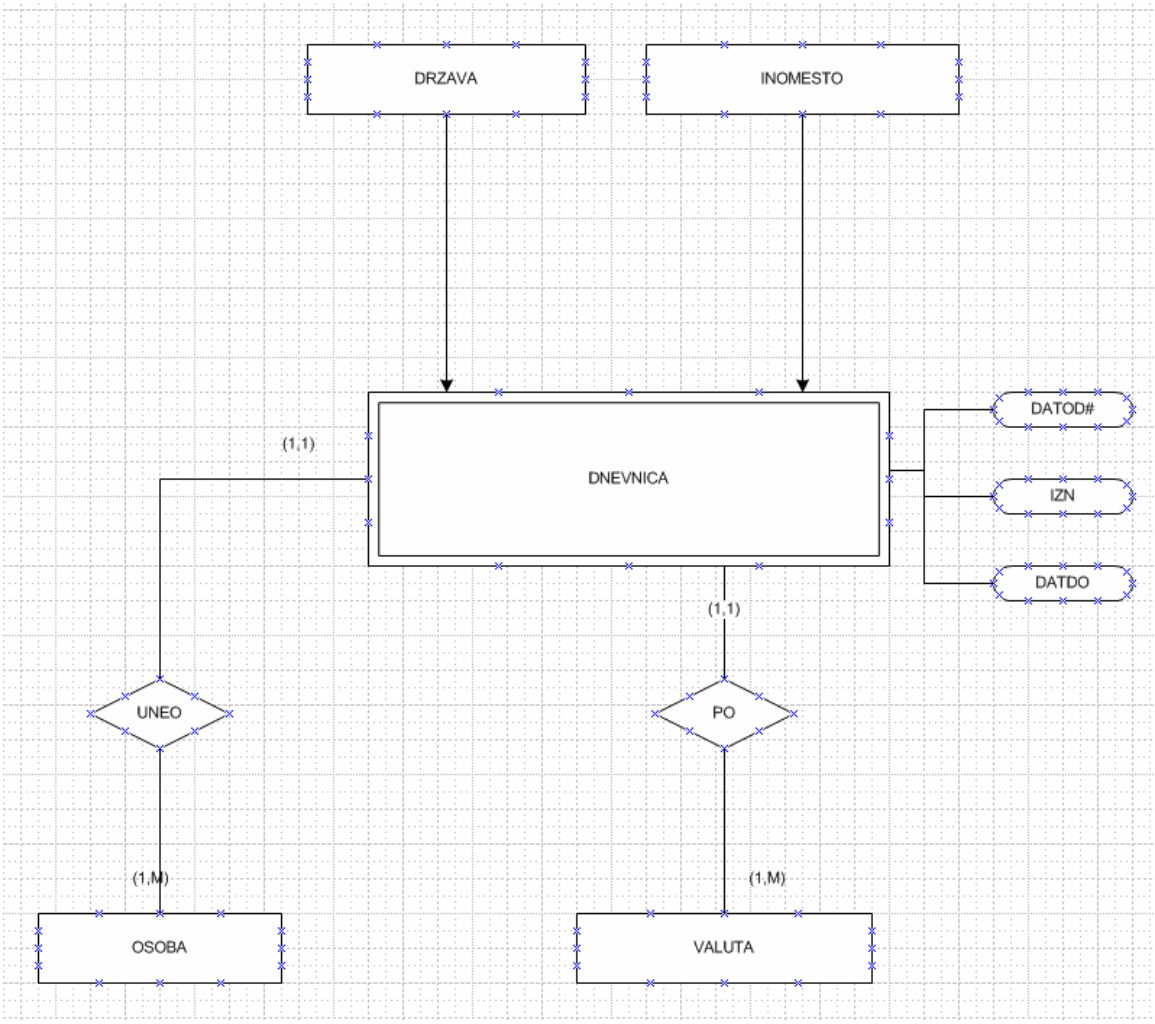
Država: Služi za evidenciju država (u ovom primeru unosi se država u koju osoba putuje).

Inomesto: Predstavlja evidenciju inostranog mesta (povezano je sa državom kojoj to mesto pripada).

Vrsta usluge: Predstavlja evidenciju vrste usluge (u ovom primeru se koristi za pravdanje dnevnica i troška).

Izgled ovih tabela prikazan je u poglavlju 1. „Izgled baze podataka“, slike 2.4 i 2.5.

Veza šifarnika dnevnice sa šifarnicima iz šeme sif prikazana je na narednoj slici:



Slika 2.8 - Veza šifarnika dnevnice sa tabelama iz šeme sif

3. Podmeni dokumenti

Opisi izgleda i funkcionalnosti ekranskih maski za dokumenta koja se unose i obrađuju u sistemu „Inostrana blagajna” su osnovni sadržaj ovog odeljka.

Dokumenta čine:

1. Putni nalog
2. Putni račun
3. Nalog deviznoj blagajni
4. Dnevnik blagajne novca

3.1 Putni nalog

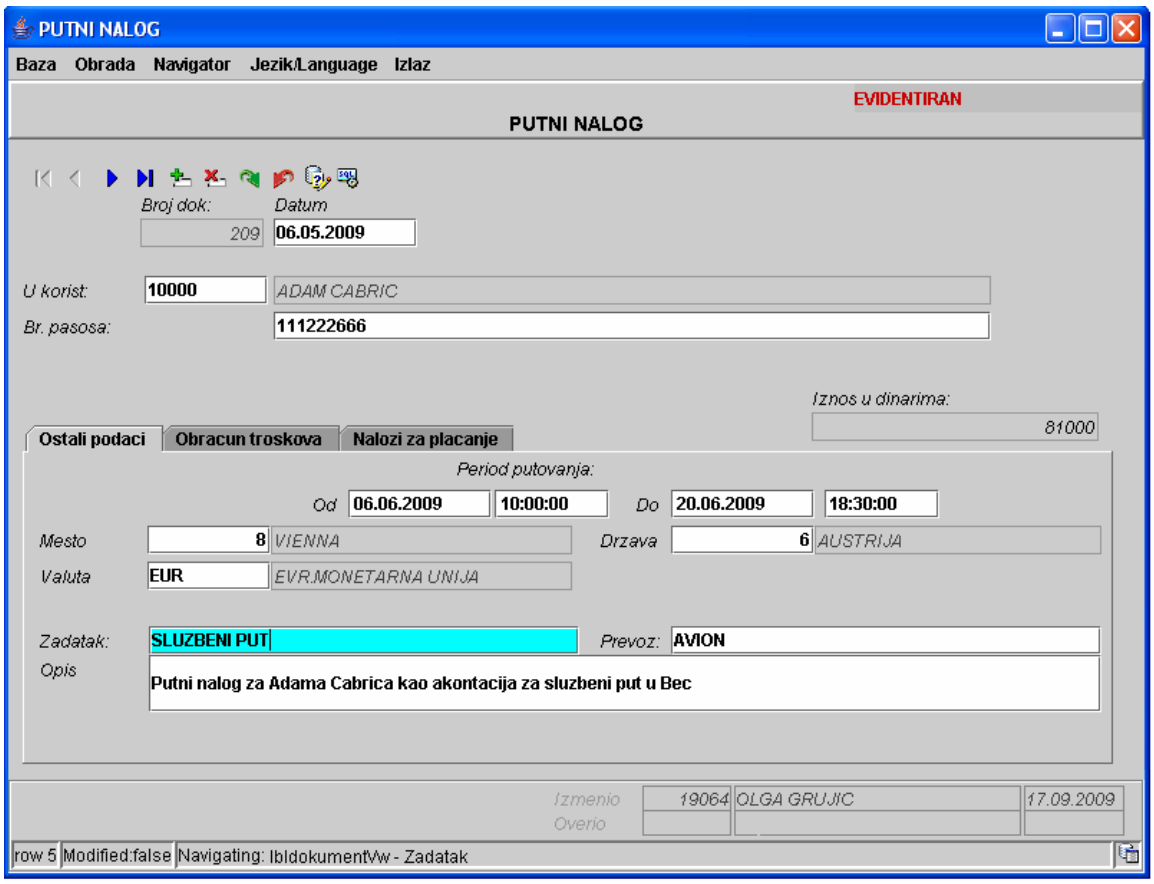
Putni nalog je prvi dokument koji se kreira. Sadrži akontaciju za službeni put u inostranstvo. U daljem tekstu biće opisana poslovna logika aplikacije, kao i polja koja ona sadrži. Takođe će biti prikazan ADF model i fizički model podataka.

3.1.1 Opis poslovne logike i realni slučaj korišćenja

Aplikacija „Putni nalog“ se formira za određenu osobu, kao nalog za službeni put u inostranstvo. Unose se podaci o mestu putovanja, o valuti u kojoj se isplaćuju troškovi, kao i razlog putovanja i prevozno sredstvo. Vrši se obračun troškova na osnovu čega se ispisuje krajnji iznos u dinarima. Na osnovu putnog naloga se izdaje putni račun.

Overom ovog dokumenta se formira novi dokument „Nalog deviznoj blagajni”.

Maska za formiranje putnog naloga za službeni put u inostranstvo prikazana je na slici 2.9:



Slika 2.9 - Putni nalog (zaglavlje i tab „Ostali podaci “)

3.1.2 Opis polja aplikacije

U ovom delu rada, dati su opisi polja koja se nalze na formi „Putni nalog“. Na ovoj formi pored zaglavlja, definisane su tri kartice (taba): „Ostali podaci”, „Obračun troškova”, „Nalozi za plaćanje”.

Zaglavlje

Naziv	Putni nalog
Broj dok	Predstavlja interni broj dokumenta (tj. redni broj naloga) pod kojim se dokument putni nalog pamti u bazi. (ibl.ibldokument.dokid) (U bazi, identifikator vrste dokumenta za „Putni nalog“ je 801 tj. polje vdid iz tabele ibldokument je 801.)

Datum	Unosi se datum naloga. Ovo polje je obavezno inače ne može da se izvrši overa dokumenta (ibl.ibldokument.datdok)
U korist	Unosi se osoba koja se upućuje na put (ibl.ibldokument.rdnid). (Unosi se šifra osobe, na osnovu koje se iz tabele osoba ispisuje njeno ime i prezime)
Br. pasosa	Unosi se broj pasoša osobe koja se upućuje na put. (ibl.ibldokument.pasosbr)
Iznos u dinarima	Ispisuje se odgovarajući iznos u dinarima (kao suma iznosa iz polja Izndin sa taba obračun troškova) (ibl.ibldokument.izndin)

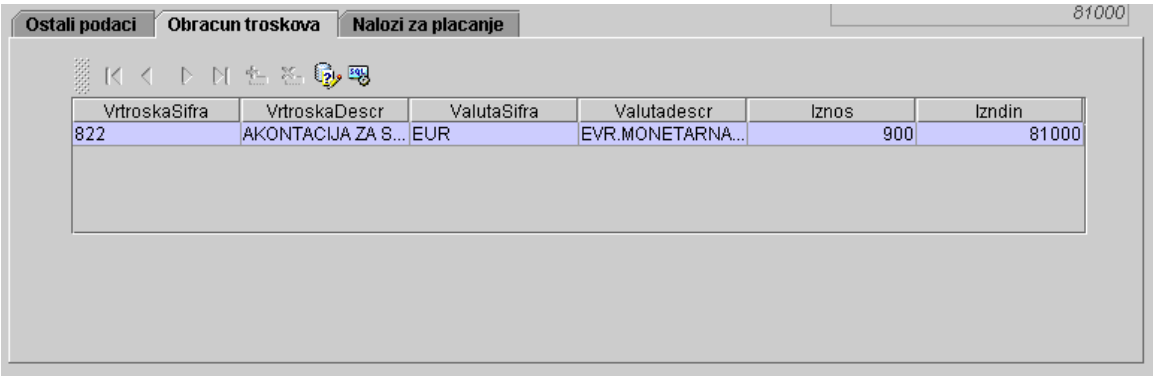
Kartica Ostali podaci

U ovu karticu unose se detaljni podaci o putu: period putovanja, država i mesto u koje osoba putuje, valuta koja se koristi, prevozno sredstvo, razlog putovanja i dodatni opis, ako je potreban.
(Izgled je prikazan na prethodnoj slici 2.9)

Naziv	Putni nalog – Kartica Ostali podaci
Period putovanja	Unosi se datum i vreme polaska (polje Od), kao i datum i vreme povratka (polje Do) sa službenog puta. (ibl.ibldokument.datod, ibl.ibldokument.datdo)
Mesto	Unosi se odredište - grad u koji se putuje (moguće korišćenje liste vrednosti). Unosi se identifikator mesta (ibl.ibldokument.inomid) i ispisuje se naziv mesta (sif.inomesto.naziv).
Država	Unosi se odredište - država u koju se putuje (moguće korišćenje liste vrednosti). Unosi se identifikator države (ibl.ibldokument.drzid) i ispisuje se njen naziv (sif.drzava.naziv).
Valuta	Unosi se valuta (ibl.ibldokument.valid) i ispisuje se pun naziv valute (sif.valuta.naziv). (Moguće korišćenje liste vrednosti).
Zadatak	Unosi se zadatak zbog koga se putuje (ibl.ibldokument.zadatak).
Prevoz	Unosi se tip prevoznog sredstva kojim se putuje (ibl.ibldokument.prevoz).
Opis	Unosi se opis (ibl.ibldokument.opis).

Kartica Obračun troškova

U ovu karticu se unosi iznos u odgovarajućoj valuti koji se daje osobi kao akontacija za službeni put u inostranstvo. Na osnovu tog iznosa ispisuje se krajnji iznos u dinarima. Kartica „Obračun troškova“ je prikazan na slici 2.10.

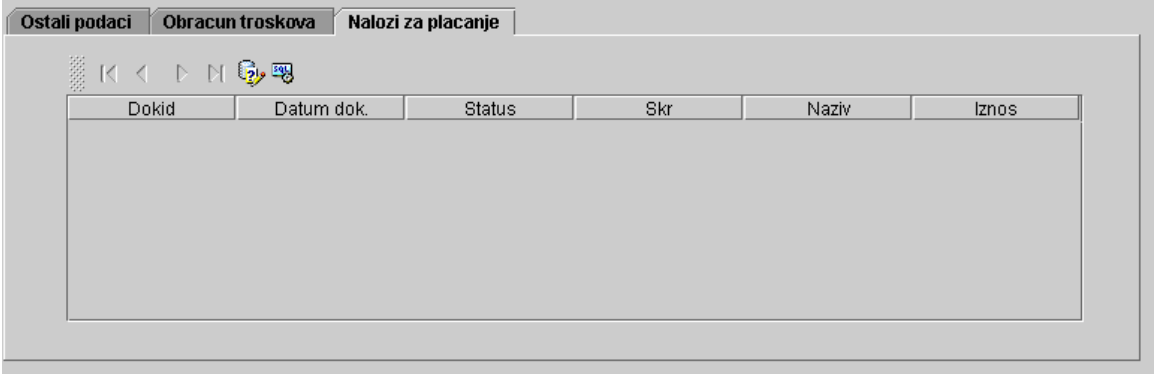


Slika 2.10 – Putni nalog - Obračun troškova

Naziv	Putni nalog – Kartica Obračun troškova
VrtroskaSifra	Unosi se šifra vrste troška (korišćenje liste vrednosti) na osnovu koje se ispisuje opis vrste troška . (ibl.iblstavka.vrtroska)
VrtroskaDescr	Po unosu šifre vrste troška ispisuje se naziv troška. (sif.vrstausluge.naziv)
ValutaSifra	Unosi se šifra valute (korišćenje liste vrednosti) na osnovu koje se ispisuje pun naziv valute. (ibl.iblstavka.valid)
Valutadescr	Po unosu šifre valute, ispisuje se pun naziv valute. (sif.valuta.naziv)
Iznos	Unosi se iznos u valuti na osnovu koga se ispisuje odgovarajući iznos u dinarima (na osnovu kursne liste za taj dan i odgovarajuće banke). (ibl.iblstavka.izn)
Izndin	Ispisuje se odgovarajući iznos u dinarima. Suma iznosa sa ovog polja se ispisuje na zaglavlju dokumenta, na polju „Iznos u dinarima“. (ibl.iblstavka.izndin)

Kartica Nalozi za plaćanje

Polja sa ove kartice se popunjavaju prilikom overe odgovarajućeg „Naloga deviznoj blagajni” (drugi dokument čiji datum odgovara datumu sa ovog putnog naloga i koji se formira prilikom overe istog). Ovaj dokument predstavlja evidenciju overenih naloga deviznoj blagajni za plaćanje na osnovu putnog naloga. (u daljem tekstu biće prikazana slika ove kartice sa popunjenim podacima). Polja sa ove kartice (engl. *Calculated*) se dobijaju SQL-upitom (iz tabela ibldokument i valuta).



Slika 2.11 – Putni nalog - Nalozi za plaćanje

Naziv	Putni nalog – Kartica Nalozi za plaćanje
Dokid	Ispisuje se broj naloga za isplatu (prenosi se broj iz odgovarajućeg naloga blagajni). (ibl.ibldokument.dokid)
Datum dok.	Ispisuje se datum naloga za isplatu (ibl.ibldokument.datdok).
Skr	Ispisuje se skrećeni naziv valute naloga za isplatu (sif.valuta.skr).
Naziv	Ispisuje se pun naziv valute (sif.valuta.naziv).
Iznos	Ispisuje se iznos sa naloga blagajni novca (ibl.ibldokument.izn).

Naziv	Putni nalog – polja sa standardnog toolbar-a
Status	Ispisuje se status naloga za isplatu. Dozvoljene vrednosti su: EVIDENTIRAN I OVEREN. Polje se automatski popunjava. (ibl.ibldokument.status)
Izmenio	Polje sadrži šifru, ime i prezime radnika koji je uneo dokument i datum unosa. (ibl.ibldokument.rdnuneo)
Overio	Sadrži šifru, ime i prezime radnika koji je overio dokument i datum overe. (ibl.ibldokument.rdnov)

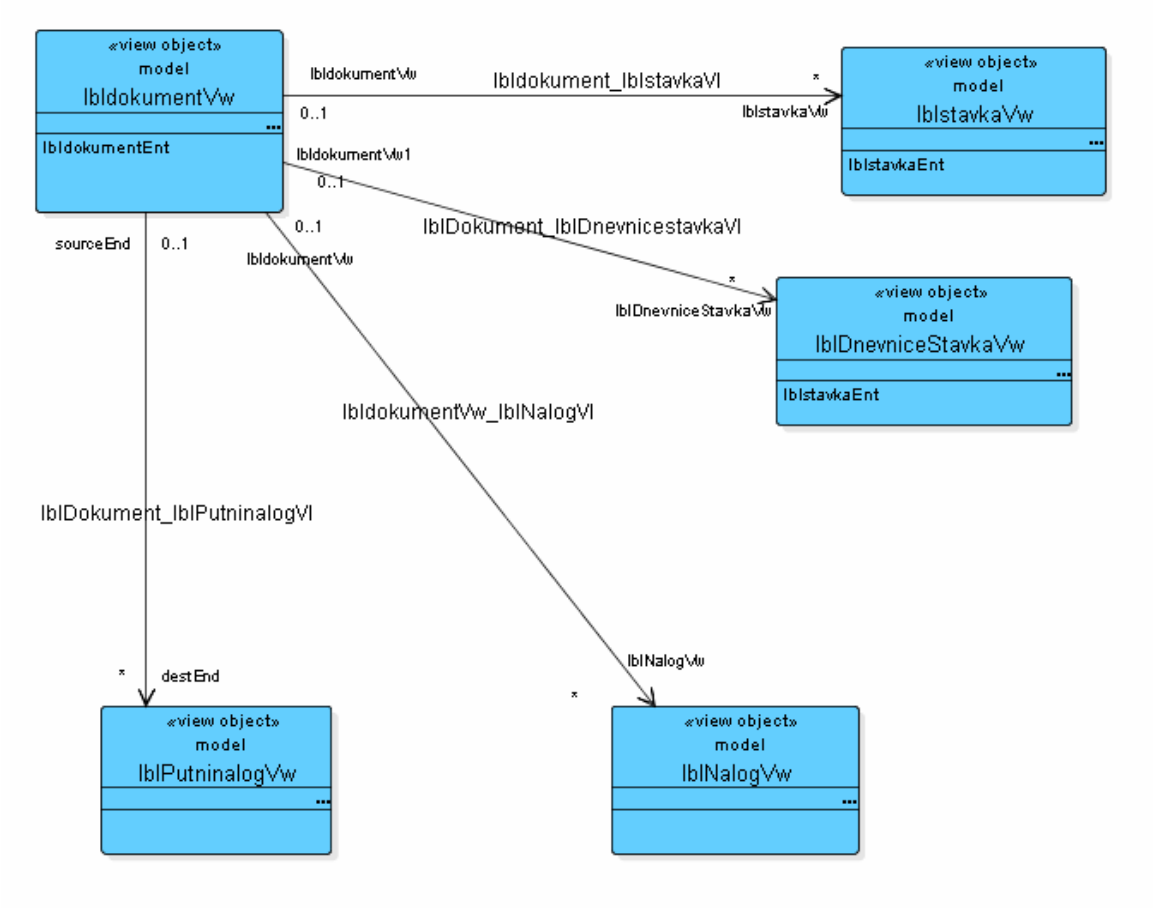
3.1.3 ADF model podataka

Dijagram koji predstavlja ADF model podataka za ceo modul „Inostrana blagajna“ prikazan je na slici 2.12. Pogled *IbldokumentVw*, odnosi se na entitet *IbldokumentEnt* koji predstavlja tabelu ibldokument. Pogledi su uglavnom grupisani u veze glavni-podređeni (*master-detail*) i povezani su pogled-vezama (npr. *Ibldokument_IblstavkaVl* je aktivna veza koja spaja glavni pogled *IbldokumentVw* i njegov podređeni pogled *IblstavkaVw*).

Pogledi kao što su *IblPutninalogVw* i *IblNalogVw* se ne odnose na entitet, već su zasnovani samo na SQL izrazu.

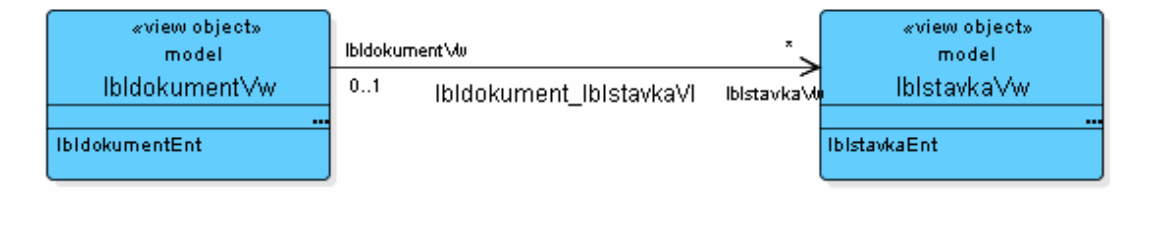
Ovaj dijagram predstavlja grafičku prezentaciju entiteta, pogleda, veza između njih i njihovih nasleđenih struktura.

Dijagram poslovne logike modula „Inostrana blagajna“ prikazan je na slici 2.1 sa početka poglavlja.



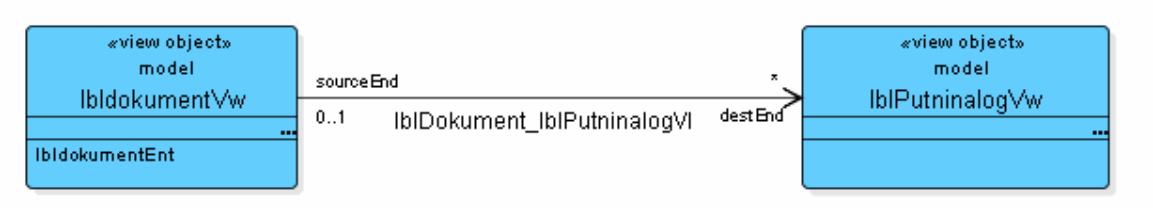
Slika 2.12 - ADF model za modul „Inostrana blagajna”

Na sledećem dijagramu prikazana je veza zaglavlja i kartice „Obračun troškova”:



Slika 2.13 – ADF model, Kartica Obračun troškova

Dijagram koji prikazuje vezu zaglavlja i kartice „Nalozi za plaćanje“ dat je na slici 2.14.
Pogled *IblPutninalogVw* zasniva se samo na SQL upitu i ne odnosi se na neki entitet.
Veza pogleda (*View Link*) je aktivna veza između pogled objekata.
Na dijagramu su označeni izvorni (*source*) i ciljni (*destination*) pogled.



Slika 2.14 – ADF model, Kartica Nalozi za plaćanje

3.1.4 Fizički model podataka

Fizički model podataka predstavlja bazni model tj. korišćene tabele u bazi.
Pogledi, koji se koriste za aplikciju, mogu da se zasnivaju na entitetima. Entiteti odgovaraju tabelama u bazi. Ukoliko se pogled zasniva samo na upitu, onda takav SQL pogled ne može da se koristi za ubacivanje, menjanje i brisanje podataka.

U sledećoj tabeli definisane su veze između ADF modela i fizičkog modela baze podataka. Za poglede, koji nisu kreirani nad fizičkom tabelom, prikazan je odgovarajući upit koji čini njihovu bazu.

Veza modela:

Rb	Entitet Objekti / Pogledi zasnovani na upitu	
1.	IbldokumentEnt	IBLDOKUMENT
2.	IblstavkaEnt	IBLSTAVKA
3.	IblPutninalogVw	Upit

Upit – IblPutninalogVw

```
SELECT ibl.vdid vdid,  ibl.dokid dokid, ibl.datod datod, ibl.datdo
datdo, ibl.valid valid , ibl.izn izn, val.skr skr, val.naziv naziv,
ibl.vdgen vdgen, ibl.dokgen dokgen, ibl.datdok datdok,
ibl.status status
FROM ibldokument ibl, valuta val
WHERE ibl.vdid = 805 and ibl.valid = val.valid(+)
 and ibl.status like 'OVEREN'
```

// vdid (identifikator vrste dokumenta) za nalog deviznoj blagajni je 805

Izgledi tabela ibldokument i iblstavka dati su u poglavlju 1. „Izgled baze podataka“, slika 2.5.

3.1.5 Dijagram sekvenci

U narednoj tabeli dat je pregled korišćenih listi vrednosti, SQL upiti koji se za njih koriste, kao i pregled nekih značajnijih metoda, čije su definicije date u prilogu.

Naziv	Putni Nalog

Aplikacioni modul: IbldokumentAm
Pogled: IbldokumentVw
IblstavkaVw

SQL Pogled: IblPutninalogVw

LISTE VREDNOSTI	

OSOBA	Izbor osoba vrši se pomoću liste vrednosti {Osoba} koja se kreira iz tabele „Osoba".
	`Select sifra, ime, prezime`
	`From osoba`

INOMESTO	Izbor mesta vrši se pomoću liste vrednosti {Inomesto} koja se kreira iz tabele „Inomesto”.
	`Select inomid, naziv, drzid`
	`From inomesto`

VALUTA	Izbor valute vrši se pomoću liste vrednosti {Valuta} koja se kreira iz tabele „Valuta".
	`Select valid, naziv, skr`
	`From valuta`

DRZAVA	Izbor države vrši se pomoću liste vrednosti {Drzava} koja se kreira iz tabele „Drzava".
	`Select drzid, naziv, valid`
	`From drzava`

VRSTAUSLUGETR	Izbor vrste troška vrši se pomoću liste vrednosti {Vrstauslugetr} koja se kreira iz tabele „Vrstausluge".
	`Select sifra, vuid, naziv`
	`From vrstausluge`

OPERACIJE

Metoda	Definisana na	Komentar

overa()->formirajNalogBlagajni()	IbldokumentAmImpl.java	Prilog 1.1
razovera()->razoveraPutnogNaloga()	IbldokumentAmImpl.java	Prilog 1.2

Dozvoljene operacije

Dozvoljene su standardne operacije: unos, brisanje, postavljanje i izvršavanje upita. Operacija brisanja je dozvoljena nad dokumentom koji ima status „EVIDENTIRAN“.

Overa i razovera dokumenta

Operacija overe se vrši pritiskom na odgovarajuće dugme iz standardnog toolbar-a koji se nalazi na vrhu ekrana. Sa overom dokumenta automatski se popunjava polje „Status“ (dobija vrednost „Overen”) i polje „Overio” koje dobija podatke o radniku koji je izvršio overu. Sa overom dokumenta zabranjuje se promena bilo kog polja na dokumentu.

Overom putnog naloga formira se dokument „Nalog blagajni novca” za isplatu, gde se automatski prenose odgovarajući podaci sa postojećeg putnog naloga i dokument dobija status „GENERISAN“. To se radi tako što metoda overa(), definisana na aplikacionom modulu, IbldokumentAmImpl.java, poziva metodu formirajNalogBlagajni() iz iste klase, koja kupi potrebne podatke sa dokumenta „Putni nalog“ i vrši ubacivanje tih podataka u tabelu ibldokument koja predstavlja dokument „Nalog blagajni novca” (identifikator vrste dokumenta, polje vdid, ima vrednost 805). Ovo ubacivanje se vrši pozivom metode:
insertNalogBlagajni(this.getDBTransaction(),
InoBlagajnaKonstante.VDID_NALOG_DEVIZNOJ_BLAGAJNI_NOVCA, datdok,
"I", vdgen, dokgen, valid, izn, izndin,osobaId, rdnuneo, "GENERISAN", zadatak, prevoz, opis);
Ova metoda nalazi se u klasi BlagajnaSqlExecutor.java, koja izvršava SQL upit za ubacivanje u tabelu, koji je definisan u metodi iz klase ControlSql.java.
Definicija metode formirajNalogBlagajni(), kao i njenih pomoćnih metoda data je u **prilogu 1.1**

Moguća je i inverzna operacija tj. operacija **razovere** koja ima suprotne efekte. Da bi se sprovela neka promena na overenom dokumentu, isti mora prethodno da se razoveri. Operacija razovere definisana je u istoj klasi *IbldokumentAmImpl.java* i poziva metodu kojom se proverava da li postoji već overen odgovarajući putni račun ili nalog blagajni (čija polja vdgen, dokgen odgovaraju poljima vdid, dokid iz putnog naloga). Ako takva dokumenta ne postoje, onda se vrši brisanje odgovarajućih evidentiranih dokumenata (putnog računa i naloga blagajni). Briše se iz tabele ibldokument sa odgovarajućim uslovom. Definicija ove metode data je u **prilogu 1.2**.
Razoverom dokument dobija status „Evidentiran“. Na razoverenom dokumentu moguće je vršiti izmene polja.

Ograničenje1: Ne može da se razoveri dokument putni nalog, ako već postoji overen putni račun ili overen nalog blagajni novca.

Ograničenje2: Moraju da postoje odgovarajući podaci u tabelama kurs i kursnoj listi za određeni dan i banku.

3.2 Putni račun

Putni račun je dokument koji se kreira na osnovu putnog naloga. Formiranje ovog dokumenta nije obavezno. Služi za evidentiranje putnog obračuna po povratku sa službenog puta. U daljem tekstu biće opisana poslovna logika aplikacije, kao i polja koje ona sadrži. Takođe će biti prikazan ADF model i fizički model podataka.

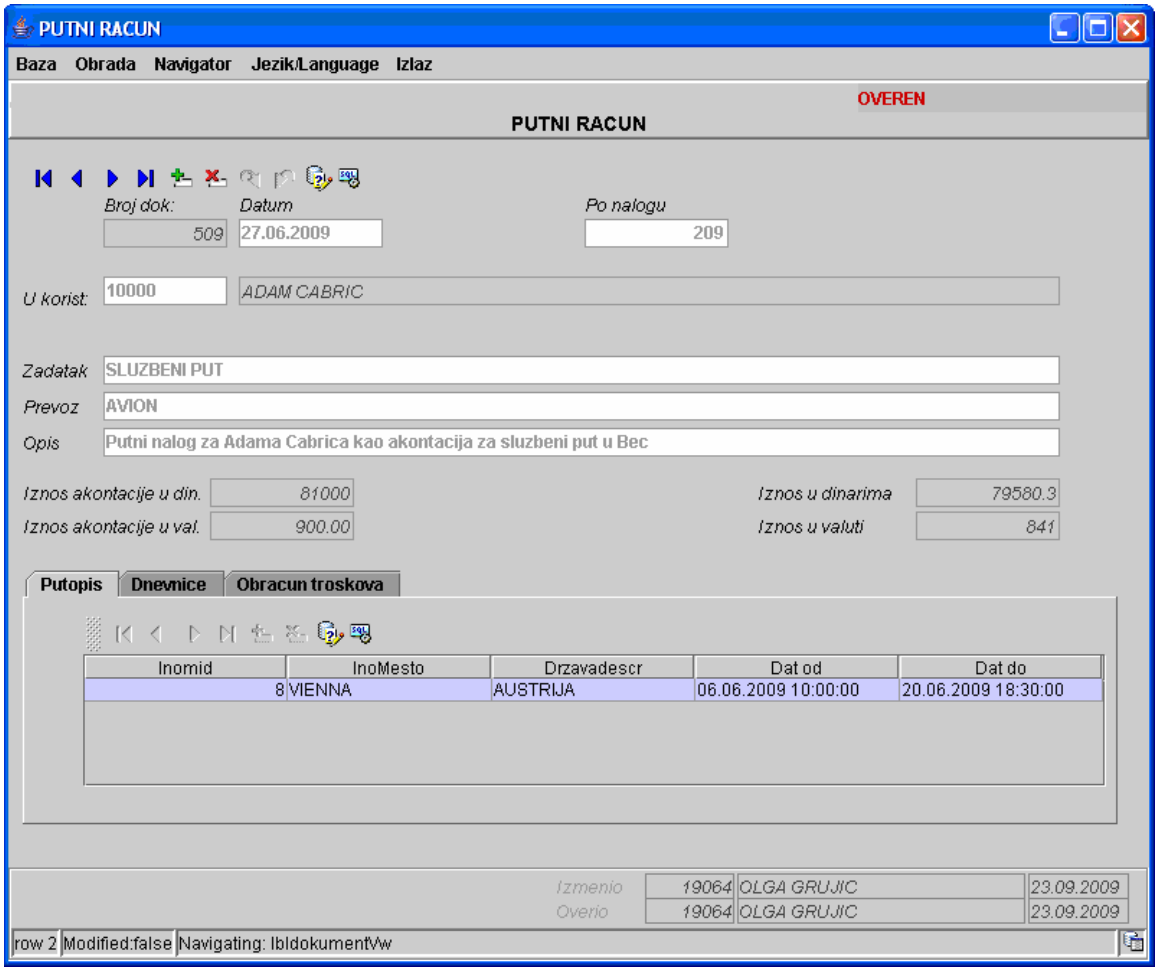
3.2.1 Opis poslovne logike i realan slučaj korišćenja

Faktura / račun u smislu zakona je:

1. račun ili drugi dokument sa ulogom računa, koji izdaje obaveznik PDV-a drugim obaveznicima PDV-a, kao i drugim licima, za svaki promet dobara i usluga, a koji se izvrši u republici;
2. račun koji obaveznik PDV-a izdaje kad naplati naknadu ili deo naknade, pre nego što izvrši promet dobara i usluga – avansno plaćanje;
3. konačni račun koji obaveznik PDV-a izdaje posle izdatog računa kod avansnog plaćanja (u kome odbija avansno plaćanje u kojem je sadržan PDV).

Putni račun je dokument koji služi za evidentiranje putnog obračuna po povratku sa službenog puta. Formira se na osnovu putnog naloga. Overom ovog dokumenta formira se „Nalog deviznoj blagajni“ i to za isplatu, ako je radnik potrošio više nego što mu je dato (tj. ako je iznos sa putnog računa veći od odgovarajućeg iznosa sa putnog naloga i tada se pored prenetih podataka sa odgovarajućeg putnog naloga, unosi broj računa i banka radnika u koju se vrši isplata) ili za uplatu, ako je iznos sa putnog računa manji od odgovarajućeg iznosa sa putnog naloga. U oba slučaja iznos na formiranom nalogu će odgovarati apsolutnoj razlici iznosa sa ručana i iznosa sa putnog naloga. Inače, ako su iznosi na računu i putnom nalogu isti onda se ne formira „Nalog deviznoj blagajni“ pri overi putnog računa.

Obrada putnog računa se vrši preko ekranske forme prikazane na slici 2.15.



Slika 2.15 - Ekranska maska Putni račun (zaglavlje i kartica „Putopis“)

3.2.2 Opis polja aplikacije

U ovom delu dati su opisi polja koja se nalze na formi „Putni račun“. Na ovoj formi pored zaglavlja, definisane su i tri kartice: „Putopis”, „Dnevnice”, „Obračun troškova”.

Zaglavlje

Polja sa standardnog toolbar-a, kao i polja: u korist, zadatak, prevoz, opis su objašnjena u poglavlju 3.1.2, pa se ovde neće ponavljati njihov opis.
U sledećoj tabeli su opisi polja koja se pojavljuju na zaglavlju forme.

Naziv	Putni račun - zaglavlje
Broj dok	Redni broj dokumenta (ibl.ibldokument.dokid). (U bazi, identifikator vrste dokumenta, polje vdid iz tabele ibldokument, za „Putni račun“je 802.)
Datum	Unosi se datum računa. Ovo polje je obavezno, inače ne može da se izvši overa dokumenta i unos stavki. (ibl.ibldokument.datdok)
Po nalogu	Upisuje se broj naloga za službeni put (npr. upiše se redni broj putnog naloga, pa se na osnovu njega automatski ispisuje osoba upućena na put, kao i zadatak putovanja, prevozno sredstvo i opis). (ibl.ibldokument.dokgen)
Iznos akontacije u din	Prenosi se iznos u dinarima sa odgovarajućeg putnog naloga. (ibl.ibldokument.izndinfakt)
Iznos akontacije u valuti	Prenosi se iznos u valuti sa putnog naloga (iznnalogval)
Iznos u dinarima	Ovo polje se popunjava posle unosa podataka na kartici „Obračun troškova”. Predstavlja sumu iznosa polja „Izndin” sa te kartice.
Iznos u valuti	Ovo polje se popunjava posle unosa podataka na kartici „Obračun troškova”. Predstavlja sumu iznosa polja „Iznval” sa te kartice.

Kartica Putopis

U ovu karticu se unosi mesto i period putovanja (sa tačnim vremenom polaska i povratka sa puta). Na osnovu perioda putovanja računa se broj dnevnica.
Izgled kartice „Putopis“ dat je na prethodnoj slici.

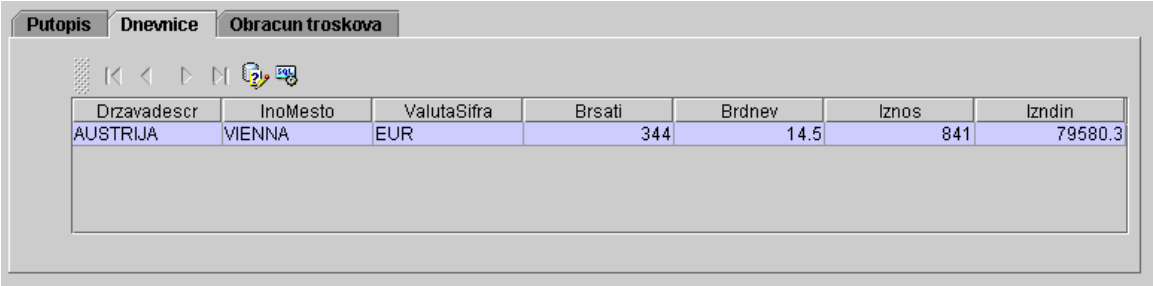
Naziv	Putni račun – putopis
Inomid	Unosi se identifikator mesta (korišćenje liste vrednosti) na osnovu kojeg se ispisuje odgovarajući grad i država. (ibl.iblstavka.inomid)
InoMesto	Ispisuje se odgovarajući grad (sif.inomesto.naziv).
Drzavadescr	Ispisuje se odgovarajuća država (sif.drzava.naziv).
Dat od	Unosi se datum i vreme polaska (ibl.iblstavka.datod).
Dat do	Unosi se datum i vreme povratka (ibl.iblstavka.datdo).

Kada se unese novi red u karticu „Putopis”, automatski se dodaje novi red u karticu „Dnevnice“ (slika 2.16) i u karticu „Obračun troškova” (slika 2.17) sa odgovarajućom valutom i iznosima, kao i vrstom troška.

Pri postavljanju datuma povratka radnika sa puta (metoda setDatdo() čiji je opis dat u **prilogu 2.1** zajedno sa definicijom njenih pomoćnih metoda), izračunava se ukupan broj sati koji je zaposleni proveo na puta (na osnovu datuma polaska i povratka sa puta, pri čemu se unosi pun format datuma sa vremenom). Ta vrednost se postavlja u polje „Br. sati“ na kartici „Dnevnice“, zatim se na osnovu izračunatih broja sati , izračunava i postavlja broj dnevnica, kao i ukupan iznos tih dnevnica u valuti i dinarima. Iznos dnevnica se računa tako što se upitom iz baze dobija iznos jedne dnevnice (iz tabele Dnevnica) za određeno mesto i državu, pa se taj iznos množi sa izračunatim brojem dnevnica i dobija se ukupan iznos dnevnica u valuti. Na osnovu njega se dobija iznos u dinarima (gleda se srednji kurs Narodne banke, tabele kurs i kursnalista, na odgovarajući datum koji odgovara datumu sa putnog računa).

Kartica Dnevnice

Kartica „Dnevice” se automatski popunjava na osnovu zaglavlja i kartice „Putopis”.
Izgled ove kartice prikazan je na slici 2.16:



Slika 2.16 - Putni račun, Kartica Denvnice

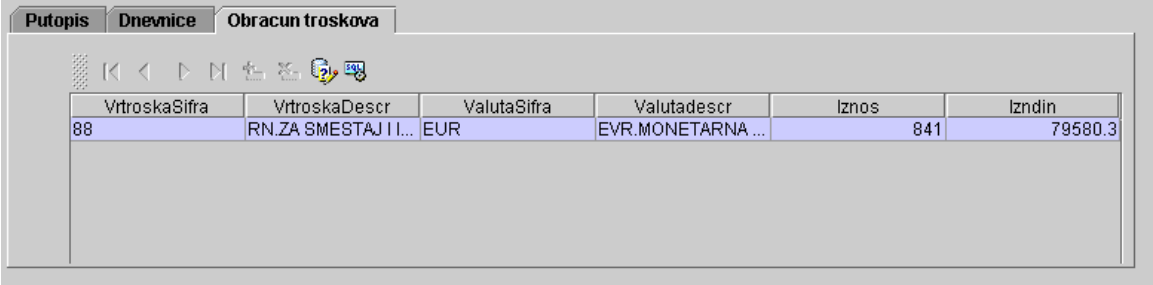
Neka polja sa ove kartice su već prethodno opisana, tako da se ovde neće ponavljati.

Naziv	Putni račun – dnevnice
Brsati	Na osnovu unetog perioda puta, polja „Dat od” i „Dat do” sa taba Putopis, izračunava se i ispisuje se ukupan broj sati, broj dnevnica (količnik broja sati sa brojem 24), iznos u valuti (proizvod broja dnevnica i vrednosti polja „Iznos” iz šifarnika Denevnice za dato mesto) i iznos u dinarima. (ibl.iblstavka.brsati)
Brdnev	Ispisuje se broj dnevnica (količnik broja sati sa brojem 24). (ibl.iblstavka.brdnev)

Kartica Obračun troškova

Polja na kartici „Obračun troškova" se automatski popunjavaju na osnovu kartica „Putopis" i „Dnevnice". Ukoliko je potrebno mogu se dodavati i novi redovi. U tom slučaju, iznosi sa ovih stavki se sabiraju i krajnje sume će biti prikazane na zaglavlju u poljima „Iznos u dinarima", „Iznos u valuti".

Izgled kartice „Obračun troškova":



Slika 2.17 - Putni račun, Tab Obračun troškova

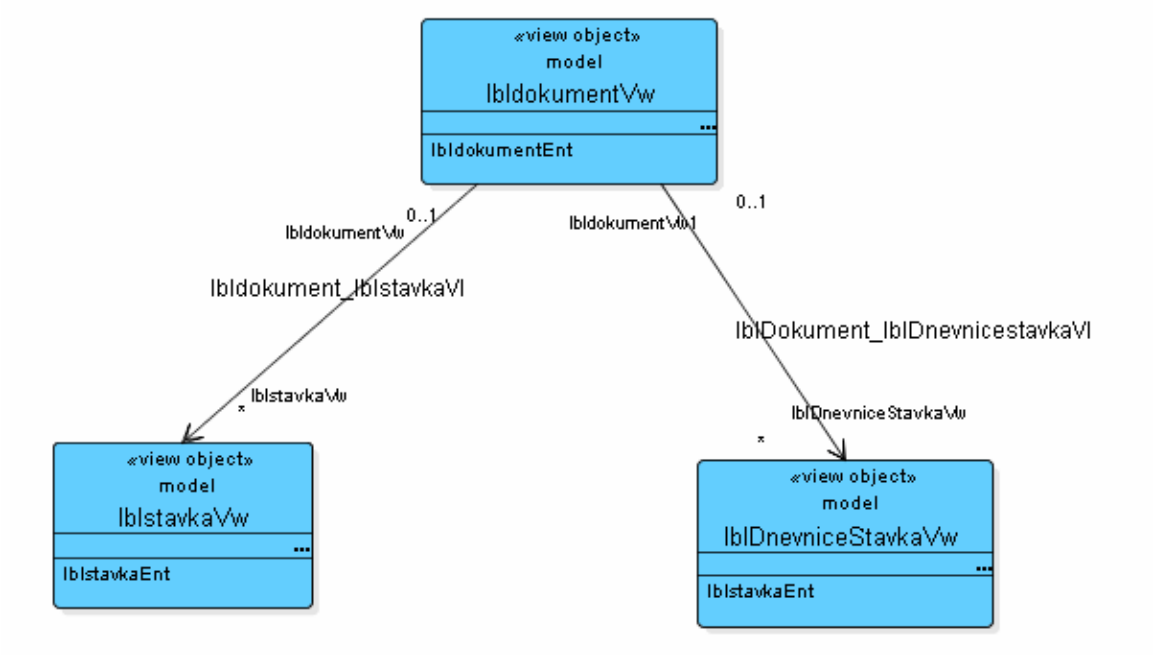
Poja valuta za iznose (*ValutaSifra*), kao i iznosi u valuti i u dinarima (*Iznos*, *Izndin),* prenose se sa kartice „Dnevnice".

Naziv	Putni račun – obračun troškova
VrtroskaSifra	Unosi se šifra troška i ispisuje se opis vrste troška (moguće korišćenje liste vrednosti – podaci se kupe iz tabele vrsta usluge). (ibl.iblstavka.vrtroska)
VrtroskaDescr	Po unosu šifre troška ispisuje se njegov naziv (sif.vrstausluge.naziv).

Poja sa standardnog toolbar-a, opisana su u poglavlju 3.1.2 „Opis polja aplikacije - Putni nalog".

3.2.3 ADF model

Sledeći dijagram predstavlja ADF model podataka za aplikaciju „Putni račun". Dijagram prikazuje poglede koji se koriste, entitete na koji se oni odnose i pogled-veze kojima su spojeni.



Slika 2.18 - ADF model aplikacije Putni račun

3.2.4 Fizički model podataka

U narednoj tabeli definisana su preslikavanja između ADF modela i fizičkog modela baze podataka.

Veza modela		
Rb	Entiteti / Pogledi zasnovani na upitu	
1.	IbldokumentEnt	IBLDOKUMENT
2.	IblstavkaEnt	IBLSTAVKA
3.	IblDnevniceStavkaVw	IBLSTAVKA

Izgled baznih tabela (fizički model baze) dat je u poglavlju 1. „Izgled baze podataka” (slika 2.5).

3.2.5 Dijagram sekvenci

Liste vrednosti koje se koriste su iste kao i za prethodno opisani dokument „Putni nalog“ (poglavlje 3.1.5), tako da se ovde neće ponovo navoditi. U tabeli je dat pregled nekih važnijih metoda. Definicije ovih metoda mogu se videti u prilogu.

Naziv	Putni račun	
Aplikacioni modul: IbldokumentAm **Pogled:** IbldokumentVw IblstavkaVw IblDnevniceStavkaVw		
OPERACIJE		
Metoda	Definisana na	Komentar
overa()-> overaPutnogRacuna()	IbldokumentAmImpl. java	Prilog 2.2
razovera()-> razoveraPutnogRacuna()	IbldokumentAmImpl. java	Vrši se provera postojanja odgovarajućeg overenog naloga blagajni novca, ako takav ne postoji briše se generisani nalog deviznoj blagajni novca.
setDokgen() (na zaglavlju)	IbldokumentEntImpl. java	Vrši se provera statusa putnog računa i ako ima status „Generisan“, onda se vrši prenošenje i postavljanje polja sa odgovarajućeg putnog naloga, inače javlja se poruka da već postoji overeni putni račun za nalog i da ga nije moguće ponovo kreirati.
setDatdo() (na tabu Putopis)	IblstavkaEntImpl.java	Prilog 2.1
setBrsati() (tab Dnevnice)	IblstavkaEntImpl.java	Prilog 2.1

Dozvoljene operacije

Standardne operacije su: unos, brisanje, postavljanje i izvršavanje upita.

Overa i razovera dokumenta

Overa: operacija overa(), definisana u klasi IbldokumentAmImpl, poziva metodu overaPutnogRacuna() iz iste klase. (Definicija ove metode kao i njenih pomoćnih metoda data je u **prilogu 2.2**). Putni račun se generiše na osnovu putnog naloga.

Sa odgovarajućeg putnog naloga uzima se iznos u dinarima, iznos u valuti i datum. Porede se iznosi u dinarima sa računa i sa naloga i izračunava se njihova razlika (ako je iznos sa računa veći od iznosa sa putnog naloga, vrši se isplata za tu razliku, a ako je iznos manji vrši se uplata). Iznos razlike u dinarima se konvertuje u iznos u odgovarajućoj valuti na datum koji odgovara datumu dokumenta sa putnog računa, na kurs odgovarajuće banke (uzima se srednji kurs Narodne banke). Vrši se ubacivanje odgovarajućeg iznosa u formirani dokument „Nalog deviznoj blagajni novca“, u zavisnosti da li je u pitanju uplata ili isplata. Menja se status u „OVEREN“ i popunjava se polje „Overio“.

Za razoveru postoje određena ograničenja.

Ograničenje1: Ne može da se razoveri putni račun ako postoji već overen odgovarajući nalog blagajni novca. Inače, briše se odgovarajući generisani nalog deviznoj blagajni novca.

Ograničenje2: Moraju da postoje odgovarajući podaci u tabelama kurs i kursnoj listi za određeni dan i banku.

Ograničenje3: Mora da postoje podaci u šifarniku „Denvnice“ za određeno mesto i period („Dat od“ i „Dat do“ polja sa taba „Putopis“ treba da su u odgovarajućem opsegu koji odgovara opsegu perioda iz šifarnika Dnevince za odgovarjuće mesto)

3.3 Nalog deviznoj blagajni

Nalog deviznoj blagajni je dokument koji se automatski formira na osnovu odgovarajućeg putnog naloga, kao i na osnovu putnog računa ako takav postoji.

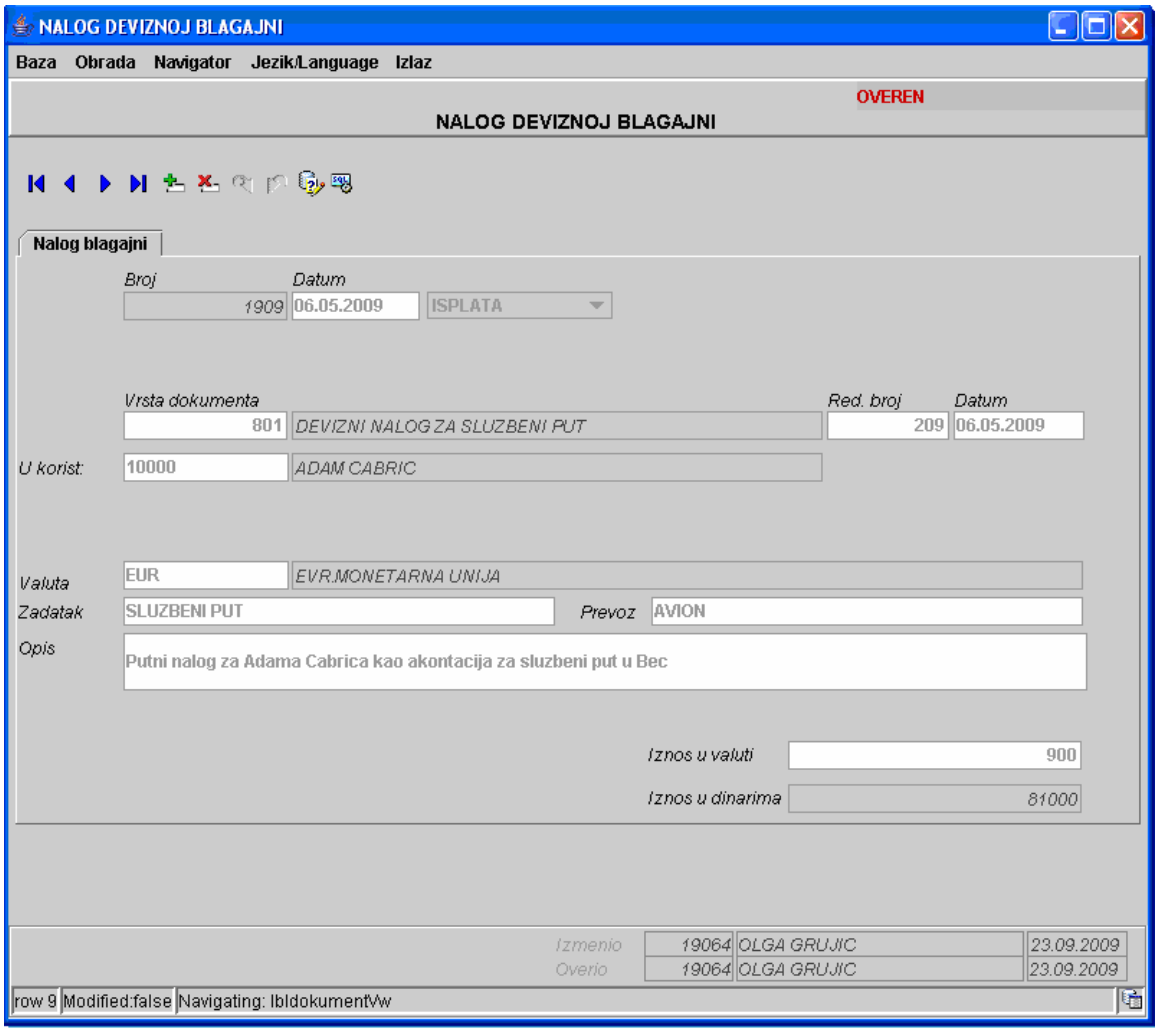
U daljem tekstu dat je izgled aplikacije, opis poslovne logika i opis polja koje aplikacija sadrži. Takođe će biti prikazan ADF model i fizički model podataka.

3.3.1 Opis poslovne logike i realni slučaj korišćenja

Pri overi Putnog naloga, automatski se formira nalog deviznoj blagajni za isplatu (svi podaci odgovaraju podacima sa odgovarajućeg putnog naloga). Na ovom primeru overava se nalog blagajni za isplatu, kojim se suma sa putnog naloga isplaćuje radniku (predstavlja izlaz iz blagajne).

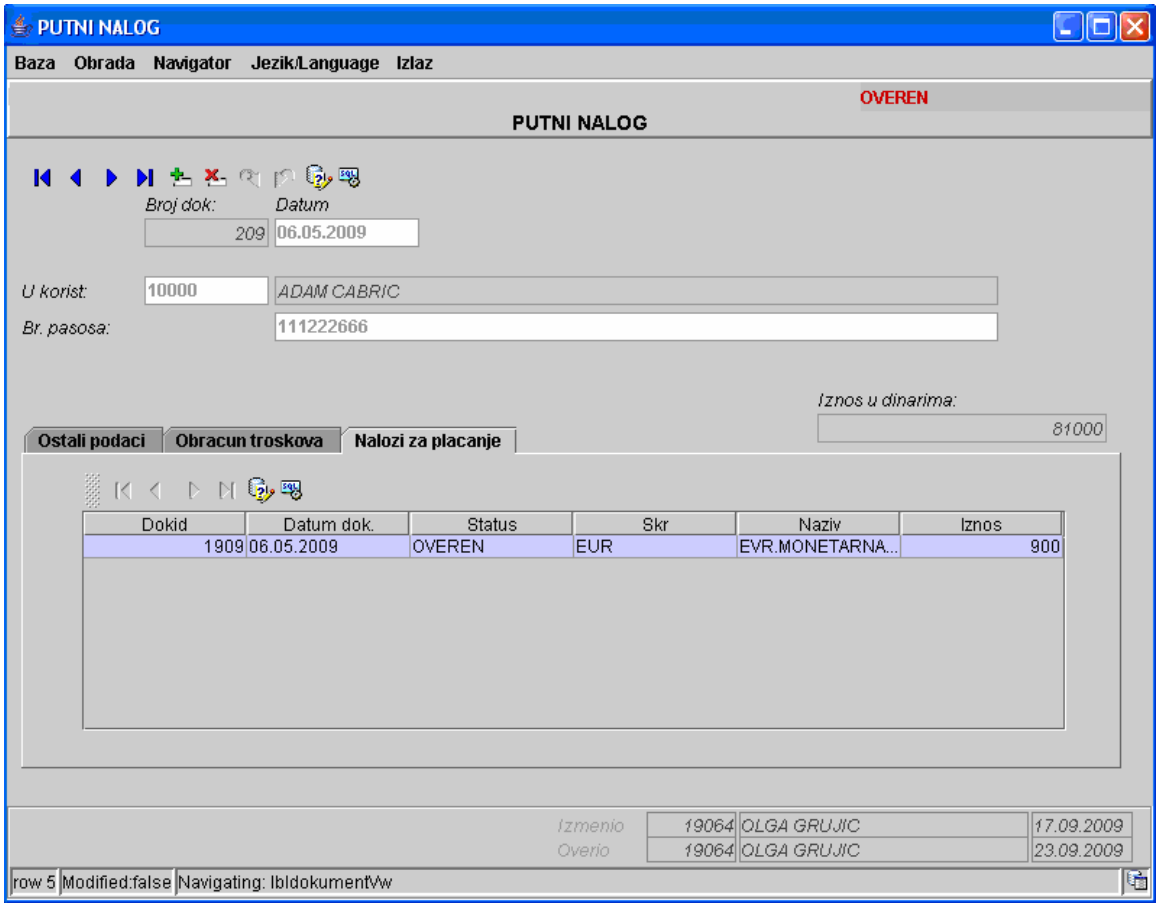
Pri overi dokumenta Putni račun, automatski se formira jedan nalog deviznoj blagajni za iznos koji odgovara apsolutnoj razlici iznosa sa odgovarajućeg putnog naloga i računa, i to za isplatu, ako je iznos sa računa veći od iznosa sa naloga (pri čemu se pojavljuju i polja „banka“, u kojoj ranik ima račun, kao i račun radnika na koji mu se uplaćuje iznos) ili za uplatu, ukoliko je taj iznos manji, inače, ako su isti iznosi, nalog se ne formira.

Izgled maske dat je na sledećoj slici (ovo je nalog koji se formira na osnovu putnog naloga):



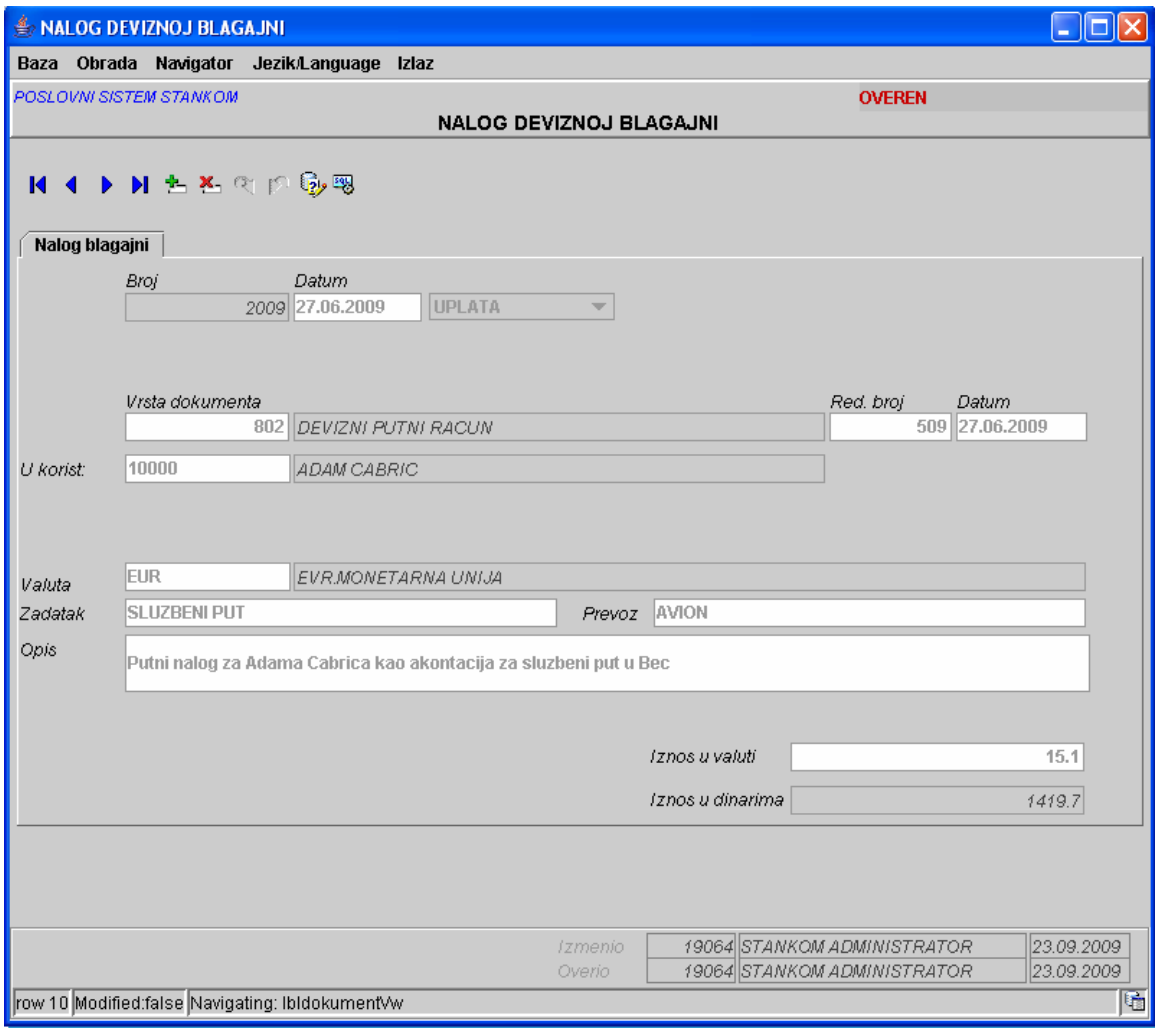
Slika 2.19 - Ekranska maska za obradu naloga blagajni na osnovu „Putnog naloga”

Pri overi „Naloga deviznoj blagajni“ na osnovu putnog naloga, popunjavaju se polja na tabu „Nalozi za plaćanje“ na odgovarajućem putnom nalogu, koji predstavlja evidenciju overenih naloga blagajni, slika 2.20:



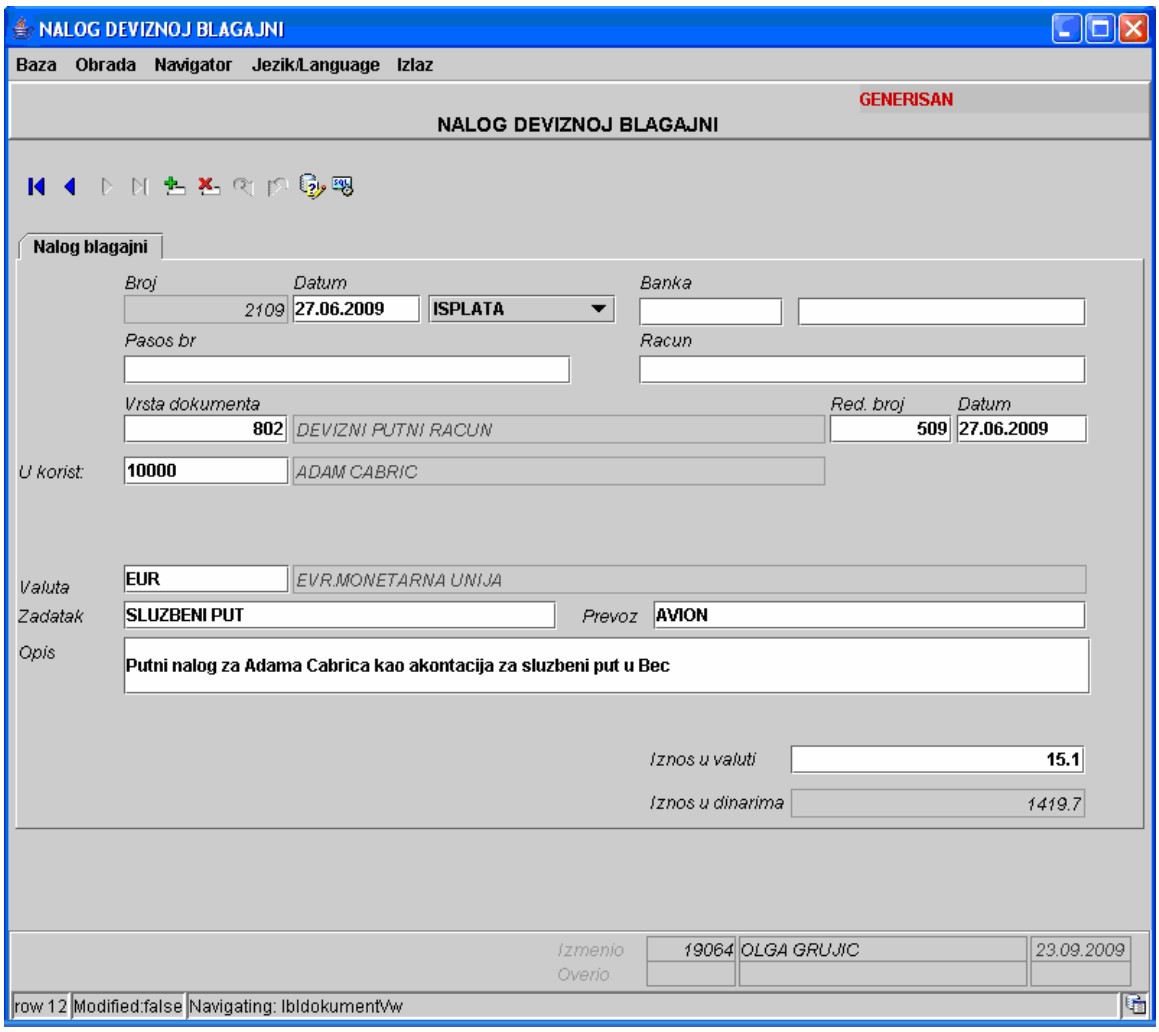
Slika 2.20 - Putni nalog, Kartica Nalozi za plaćanje

Na narednoj slici dat je izgled maske „Naloga deviznoj blagajni“ po putnom računu i to za uplatu (na ovom primeru je iznos sa računa manji od iznosa sa putnog naloga):



Slika 2.21 - Ekranska maska za obradu naloga blagajni (uplata) na osnovu putnog računa

U slučaju da je odgovarajući iznos sa računa veći od iznosa sa naloga izgled maske za isplatu bi bio kao na slici 2.22.



Slika 2.22 - Ekranska maska za obradu naloga blagajni (isplata) na osnovu putnog računa

3.3.2 Opis polja aplikacije

Pored unosa datuma naloga blagajni, koje je obavezno polje (promena nastala overom ovog naloga se evidentira u dnevniku novca od datuma koji stoji u ovom polju), ostala polja se prenose sa odgovarajućeg putnog naloga ili putnog računa. (Ova polja su opisana u prethodnim poglavljima, tako da se ovde neće ponavljati).

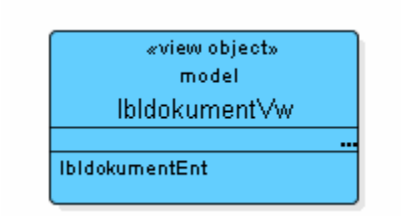
Polje, kojim se definiše da li se radi o uplati ili isplati iz blagajne, predstavlja obavezno polje, čije su dozvoljene vrednosti „Uplata" i „Isplata".

Ukoliko se radi o uplati na osnovu putnog naloga (ulaz u blagajnu, uzima se novac iz banke) ili isplati po putnom računu, onda se na formi pojavljuju polja banka (šifra banke u kojoj zaposleni ima račun), broj pasoša i račun (broj tekućeg računa na koji treba da se uplati/isplati novac).

Poja sa standardnog toolbar-a, opisana su u poglavlju 3.1.2 „Opis polja aplikacije - Putni nalog".

3.3.3 ADF model podataka

ADF model se sastoji samo od pogleda IbldokumentVw zasnivanog na entitetu IbldokumentEnt koji odgovara tabeli ibldokument.



3.3.4 Fizički model podataka

U narednoj tabeli definisane su veze između ADF modela i fizičkog modela baze podataka.

Veza modela		
Rb	Entitet/Pogled zasnovan na upitu	
1.	IbldokumentEnt	IBLDOKUMENT

Izgled tabele ibldokument dat je u poglavlju 1. „Izgled baze podataka“, slika 2.5.

3.3.5 Dijagram sekvenci

Liste vrednosti koje se koriste su opisane u poglavlju 3.1.5 (Dijagram sekvenci, „Putni nalog“), pa se ovde neće ponovljati.

Naziv	Nalog blagajni novca	
Aplikacioni modul: IbldokumentAm **Pogled:** IbldokumentVw		
OPERACIJE		
Metoda	Definisana na	Komentar
overa()-> overaNalogaDeviznojBlagajni()	IbldokumentAmImpl. java	Prilog 3

razovera-> razoveraNalogaBlagajniNovca()	IbldokumentAmImpl. java	Provera statusa odgovarajućeg dnevnika novca

Dozvoljene operacije

Standardne operacije su: unos, brisanje, postavljanje i izvršavanje upita.

Overa i razovera dokumenta

Metoda overa() (*IbldokumentAmImpl*) poziva metodu overaNalogaDeviznojBlagajni(), koja vrši proveru postojanja overenih dokumenata i ubacivanje naloga u dnevnik.
Definicija ove metode, kao i njenih pomoćnih metoda data je u **prilogu 3**.
Ukoliko ne postoji overen odgovarajući putni nalog ili putni račun, ne može da se overi ovaj dokument. Inače, ubacuje se nalog u dnevnik blagajni na odgovarajući datum (vrši se ubacivanje reda u tabelu ibldokument sa odgovarajućim vrednostima), sa ograničenjem da ne mogu da se ubacuju novi nalozi ako je odgovarajući dnevnik overen.
Menja se status dokumenta i popunjava se pojle „Overio“.
Inverzna operacija je operacija **razovere**.

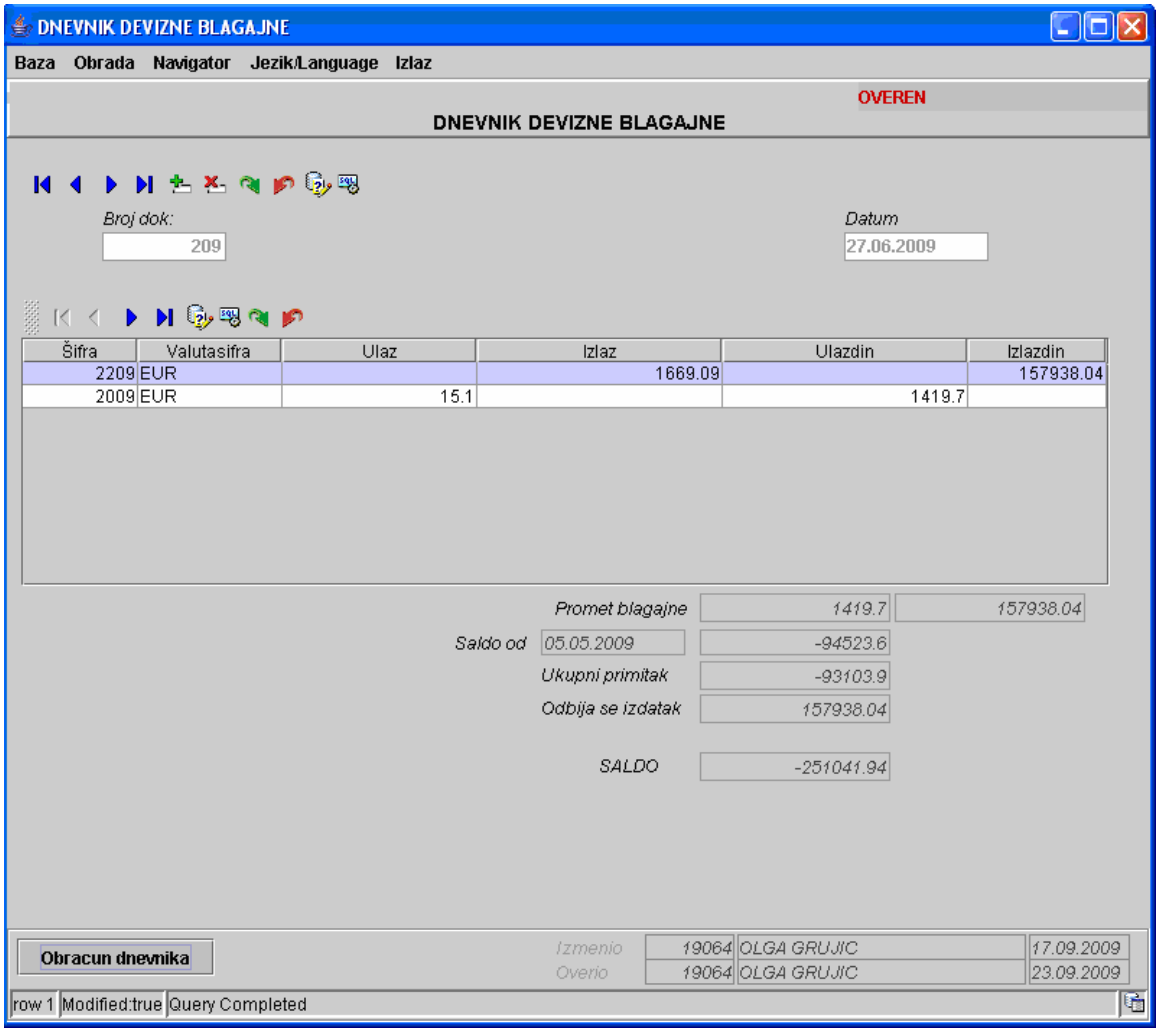
***Ogranič**enje1:* Ne može da se razoveri nalog blagajni, ako postoji već overen odgovarajući dnevnik blagajne novca (dnevnik sa datumom koji odgovara datumu sa naloga).

3.4 Dnevnik devizne blagajne

Dnevnik devizne blagajne je poslednji dokument koji se formira. Sadrži odgovarajuće naloge deviznoj blagajni. Na ovom dokumentu se vidi i ukupan ulaz i izlaz iz blagajne (na dnevnom nivou), kao i iznos salda.
U ovom poglavlju je dat opis poslovne logike aplikacije, njen izgled i opis polja, kao i ADF i fizički model podataka.

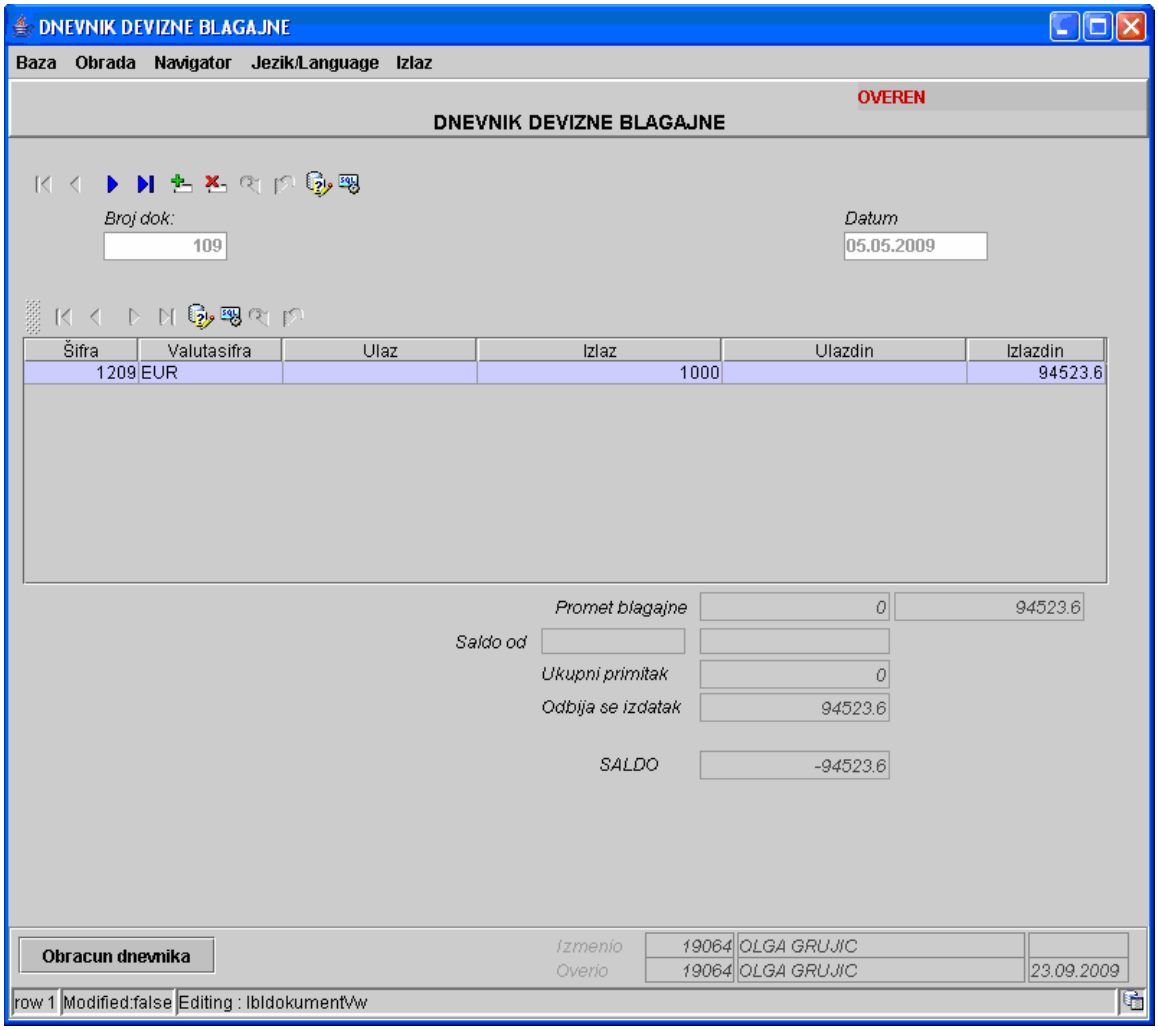
3.4.1 Opis poslovne logike i realan slučaj korišćenja

Dnevnik devizne blagajne je dokument koji prati dnevne promene u blagajni novca.
Po unosu datuma dokumenta, stavke dnevnika se automatski prenose sa naloga deviznoj blagajni koji ima isti datum.
Aplikacija za obradu dnevnika blagajne ima izgled kao na slici 2.23.



Slika 2.23 - Ekranska maska za dnevnik devizne blagane

Na sledećoj slici je prikazan prethodni dnevnik, tj. dnevnik sa datumom koji prvi prethodi datumu na slici 2.23:



Slika 2.24 - Ekranska maska za prvi prethodni dnevnik devizne blagane

3.4.2 Opis polja aplikacije

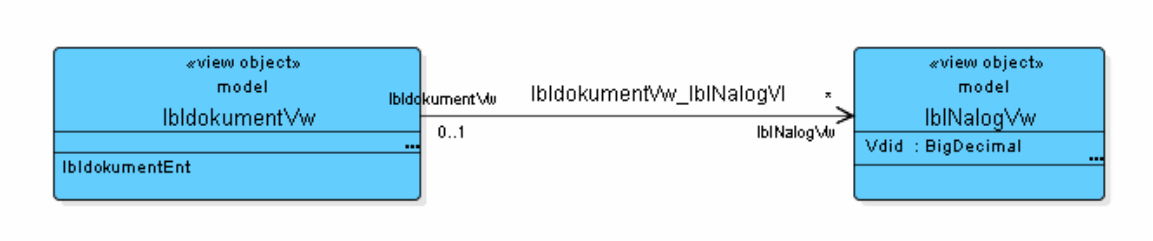
Datum dnevnika blagajne je obavezno polje, bez koga dokument ne bi mogao da se overi. Kad se unese određeni datum, koji odgovara nalogu deviznoj blagajni, automatski se prenose podaci sa naloga: broj naloga, valuta, uplata i/ili isplata u odgovarajućoj valuti, kao i odgovarajući iznosi u dinarima.

Polja koja se izračunavaju kada se klikne na dugme za obračun dnevnika su sledeća: promet blagajne (ukupna uplata u dinarima po nalozima za taj dan, tj. ulaz u blagajnu, i ukupna isplata u dinarima po nalozima za taj dan, tj. izlaz iz blagajne), prethodni saldo, ukupan primitak (ukupan ulaz blagajne novca za taj dan), izdatak (ukupan izlaz blagajne za taj dan) i ukupan saldo.

Poja sa standardnog toolbar-a, opisana su u poglavlju 3.1.2 „Opis polja aplikacije - Putni nalog“.

3.4.3 ADF model

Dijagram koji prikazuje ADF model podataka za aplikaciju „Dnevnik devizne blagajne”:



Slika 2.25 – ADF model za aplikaciju „Dnevnik devizne blagajne“

3.4.4 Fizički model podataka

Tabela mapiranja između ADF modela i fizičkog modela baze podataka:

Veza modela		
Rb	Entitet/ Pogled zasnovan na pogledu	
1.	IbldokumentEnt	IBLOKUMENT
2.	IblNalogVw	Upit

Izgled tabele ibldokument dat je u poglavlju 1. „Izgled baze podataka“, slika 2.5.
Pogled IblNalogVw zasnovan je samo na sql upitu i ne odnosi se na entitet.

Upit - IblNalogVw

```
SELECT IbldokumentEnt.VDID,
       IbldokumentEnt.DOKID,
       IbldokumentEnt.DATDOK,
       IbldokumentEnt.STATUS,
       IbldokumentEnt.PPID_B,
       IbldokumentEnt.VDGEN,
       IbldokumentEnt.DOKGEN,
       IbldokumentEnt.DATOD,
       IbldokumentEnt.DATDO,
       IbldokumentEnt.DRZID,
       IbldokumentEnt.INOMID,
       IbldokumentEnt.ZADATAK,
       IbldokumentEnt.PREVOZ,
       IbldokumentEnt.OPIS,
       IbldokumentEnt.IZN,
       IbldokumentEnt.IZNDIN,
       IbldokumentEnt.VALID,
```

```
        IbldokumentEnt.ZR_PPID,
        IbldokumentEnt.PASOSBR,
        IbldokumentEnt.IZNDINFAKT,
        IbldokumentEnt.KODUI,
        IbldokumentEnt.ULAZ,
        IbldokumentEnt.ULAZDIN,
        IbldokumentEnt.IZLAZ,
        IbldokumentEnt.PRIMUK,
        IbldokumentEnt.IZDUK,
        IbldokumentEnt.SALDO,
        IbldokumentEnt.IZLAZDIN,
        V.SKR ValutaSifra
FROM IBLDOKUMENT IbldokumentEnt, VALUTA V
WHERE IbldokumentEnt.VDID = 805 and  // nalog blagajni
 IbldokumentEnt.VALID = V.VALID and
 IbldokumentEnt.STATUS like 'OVEREN'
```

Izgeled baznih tabela i njihove povezanosti, prikazani su u prethodnim poglavljima (slika 2.5).

3.4.5 Dijagram sekvenci

Liste vrednosti koje se koriste u zaglavlju dokumenta su iste kao za „Putni nalog“ (poglavlje 3.1.5), tako da se ovde neće ponovo navoditi.

Naziv	Dnevnik blagajne novca

Aplikacioni modul: IbldokumentAm
Pogled: IbldokumentVw
IblNalogVw

SQL Pogled: IblNalogVw

OPERACIJE		
Metoda	Definisana na	Komentar
obracunDnevnika()	IbldokumentAmImpl. java	Prilog 4
razovera-> razoveraDnevnikaNovca()	IbldokumentAmImpl. java	Vrši se provera postojanja overenog dnevnika sa kasnijim datumom i ako takav dokument postoji nije moguće vršiti razoveru

Dozvoljene operacije

Standardne operacije su: unos, brisanje, postavljanje i izvršavanje upita.

Overa i razovera dokumenta

Overa: Vrši se provera postojanja dnevnika sa kasnijim datumom i ako takav postoji javlja se poruka o grešci.
Određuje se datum poslednjeg overenog dnevnika (prvog prethodnog dnevnika), kao i njegov saldo i te vrednosti se postavljaju (*set* metodama). Postavlja se primitak kao suma ulaza sa stavke i vrednosti izračunatog salda poslednjeg dnevnika, zatim izdatak, koji je razlika salda prethodnog dnevnika i ukupnog izlaza sa stavki. Polje „Saldo“ je razlika izračunatih polja „primitak“ i „izdatak“. Ta polja se izračunavaju i postavljaju, klikom na dugme **„Obračun dnevnika“** ili direktno overom dokumenta.
(Metoda obracunDnevnika() poziva se u klasi ActionObracunaj.java, ali takođe i u metodi overa()). Definicija metode obracunDnevnika(), kao i njenih pomoćnih metoda data je u **prilogu 4.**

Razoverom dokument dobija status „Evidentiran“. Na razoverenom dokumentu moguće je vršiti izmene polja.

***Ograničenje**:* Ne može da se overi i razoveri dnevnik, ako već postoji overen dnevnik sa kasnijim datumom. Ne može da se vrši overa dnevnika ako datum nije unešen.

IV ZAKLJUČAK

Na rast preduzeća i uspešnost poslovanja u velikoj meri utiče optimizacija poslovnih procesa. Ključni faktor uspeha su potpuna rešenja koja zauzimaju važno mesto u obnovi poslovnih procesa, čiji je cilj efikasnije poslovanje i veća produktivnost.

ERP sistemi su rešenja koja preduzećima omogućavaju brz odziv i rad sa kupcima, dobavljačima i poslovnim partnerima na globalnom tržištu.

Prednosti ERP sisema su sledeće:

- Poboljšava pristup informacijama
- Poboljšava tokove procesa i njihovu efikasnost
- Omogućava prilagođavanje potrebama
- Omogućava reinženjering poslovnih procesa i njihovo unapređenje
- Eliminisanje skupih i nefleksibilnih postojećih rešenja

Modul „Inostrana blagajna“, kao deo ERP sistema, može biti korisno softversko rešenje svima onima koje zanima poslovanje kako manjih preduzeća tako i većih kompanija.

Oracle ADF olakšava razvoj Java EE aplikacija omogućavajući: izbor pristupa razvoju, tehnologije i ciljne platforme. Kombinovanje napredne arhitekture Oracle ADF-a sa vizuelnim razvojnim okruženjem Oracle JDeveloper 10g, predstavlja dobro rešenje, kako za početnike, tako i za iskusne programere koji žele da povećaju svoju produktivnost u razvoju Java aplikacija.

V LITERATURA

1. Building J2EE Applications with Oracle ADF, Steve Muench, Oracle ADF Development Team
2. JDBC 4.0 and Oracle JDeveloper for J2EE Development, Deepak Vohra
3. ADF Business Components J2EE Design Pattern Catalog, Steve Muench, ADF Development Team
4. Oracle JDeveloper 10g : Empowering J2EE Development, Harshad Oak
5. Java 2 Enterprise Edition 1.4 (J2EE 1.4) Bible, James McGovern
6. http://www.oracle.com/technology
7. http://www.otn.oracle.com/product/jdev

VI PRILOZI

1. Prilog 1

1.1 Metoda za formiranje dokumenta Nalog blagajni novca

Ova metoda se poziva pri overi dokumenta (ovde putnog naloga).
Definisana je na aplikacionom modulu, u klasi IbldokumentAmImpl.java

```
public void formirajNalogBlagajni()
{
      IbldokumentVwRowImpl nal =
           (IbldokumentVwRowImpl)this.getIbldokumentVw().getCurrentRow();
       if(nal == null) throwBseException("Morate kreirati validan putni nalog!");
            // uzima se kao važeći srednji kurs Narodne banke
      BigDecimal ppid = InoBlagajnaKonstante.KURSNALISTA_PPID;
      Date datov = nal.getDatdok();
      BigDecimal vdgen = nal.getVdid();
      BigDecimal dokgen = nal.getDokid();
      BigDecimal valid = nal.getValid();
      String pasos = nal.getPasosbr();
      BigDecimal izn =   nal.getIzn();
      if(izn == null) throwBseException("Nemate podatke u kursnoj listi za dan:
                                              "+datov+" i banku: "+ppid);
      String zadatak = nal.getZadatak();
      String opis = nal.getOpis();
      String prevoz = nal.getPrevoz();
      BigDecimal osobaId = nal.getRdnid();
      BigDecimal izndin = nal.getIzndin();
      if(izndin == null) throwBseException("Nemate podatke u kursnoj listi za dan:
                                                 "+datov+" i banku: "+ppid);
      BigDecimal rdnuneo = this.getParamRdnid();
      Date datum = nal.getDatdok();

            /* Nalog za uplatu – ulaz u blagajnu */
      BlagajnaSqlExecutor.insertNalogBlagajniSP(this.getDBTransaction(),
        InoBlagajnaKonstante.VDID_NALOG_DEVIZNOJ_BLAGAJNI_NOVCA, datov,
                    "U", vdgen, dokgen, valid, izn, izndin,osobaId, rdnuneo, "GENERISAN",
                     zadatak, prevoz, opis,  pasos);

           /* Nalog za isplatu (izlaz), kojim se predaje novac randiku koji
              ide na službeni put
           */
       BlagajnaSqlExecutor.insertNalogBlagajni(this.getDBTransaction(),
        InoBlagajnaKonstante.VDID_NALOG_DEVIZNOJ_BLAGAJNI_NOVCA, datov,
           "I", vdgen, dokgen, valid, izn, izndin,osobaId, rdnuneo, "GENERISAN",
```

```
            zadatak, prevoz, opis);
  }
```

1.2 Metoda za razoveru dokumenta Putni nalog

Metoda je definisana u klasi IbldokumentAmImpl.java

```
private void razoveraPutnogNaloga(IbldokumentVwRowImpl red)
{
   String status =
   BlagajnaSqlExecutor.GetStastusGenerisanogNalogaBlagajni(this.getDBTransaction(),
                                                                red.getVdid(), red.getDokid());
   if((status != null) && (status.equals(InoBlagajnaKonstante.STATUS_OVEREN)))
               throwBseException("Ne mozete razoveriti ovaj dokument. Imate overen nalog
                                        blagajni novca.");
   status =
   BlagajnaSqlExecutor.GetStastusGenerisanogPutnogRacuna(this.getDBTransaction(),
                                                               red.getVdid(), red.getDokid());
   if((status != null) && (status.equals(InoBlagajnaKonstante.STATUS_OVEREN)))
               throwBseException("Ne mozete razoveriti ovaj dokument. Imate overen putni
                                        racun.");
                /* Ako takva dokumenta ne postoje, onda se vrši brisanje odgovarajućih
                   evidentiranih dokumenta (putnog računa i naloga blagajni). Vrši se brisanje
                   iz tabele ibldokument sa odgovarajućim uslovom
                */
    BlagajnaSqlExecutor.deleteGenerisaniPutniRacun(this.getDBTransaction(),
                                                          red.getVdid(), red.getDokid());
    BlagajnaSqlExecutor.deleteGenerisaniNalog(this.getDBTransaction(), red.getVdid(),
                                                    red.getDokid());
 }

/* metoda u klasi BlagajnaSqlExecutor.java */

public static String GetStastusGenerisanogNalogaBlagajni(
                  DBTransaction dbTransaction,BigDecimal vdgen, BigDecimal dokgen)
{
      String upit = ControlSql.createSqlGetStatusNaloga(
         InoBlagajnaKonstante.VDID_NALOG_DEVIZNOJ_BLAGAJNI_NOVCA,
         vdgen, dokgen);
      return (String)getUpit("default", dbTransaction).exeSingle(upit);
}

// u klasi ControlSql.java
public static String createSqlGetStatusNaloga(BigDecimal vdid,BigDecimal vdgen,
                                                    BigDecimal dokgen)
{
       String query = " select status from ibldokument where vdid = "+vdid+" and
                           vdgen="+vdgen+" and dokgen="+dokgen;
       return query;

 }
```

2. Prilog 2

2.1 Metoda koja postavlja datum

Metoda setDatdo() definisana je na entitetu, u klasi IblstavakaEntImpl.java
Date su i definicije metoda koje ova metoda poziva i drugih pomoćnih metoda.

```
public void setDatdo(Timestamp value)
  {

    proveriDatume(this.getDatod(), value);
    postaviBrojSati(this.getDatod(), value);
    setAttributeInternal(DATDO, value);

  }

private void postaviBrojSati(Timestamp datOd, Timestamp datDo)
{
     if((datOd != null) && (datDo != null))
    {
        BigDecimal brSati = CommandIzracunajBrojSati.execute(datOd, datDo);
        setBrsati(brSati);
     }
}

// Ova metoda je definisana u klasi CommandIzracunajBrojSati.java
public static BigDecimal execute(Timestamp datumOd, Timestamp datumDo)
{

    Calendar c = GregorianCalendar.getInstance();
    c.setTime(datumOd);
    Calendar c1 = GregorianCalendar.getInstance();
    c.setTime(datumOd);
    long datumDoLong = datumDo.getTime();
    long datumOdLong = datumOd.getTime();
    long millSec = datumDoLong - datumOdLong;
    long hours =  millSec/3600000;
    return new BigDecimal(hours);

}

// definisana na IblstavkaEntImpl
  public void setBrsati(BigDecimal value)
  {

    BigDecimal brDnevnica = CommandIzracunajBrojDnevnica.execute(value);
    setAttributeInternal(BRDNEV, brDnevnica);

     Date datdok = getMasterDate();
     BigDecimal ppid = InoBlagajnaKonstante.KURSNALISTA_PPID;
```

```
    BigDecimal inom = getInomid();
    System.out.println("inom "+inom);
    BigDecimal drz = getDrzid();

         // izračunava se iznos dnevnice za određenu državu i mesto
        // kolona izn se kupi iz tabele dnevnica uz određeni uslov
    BigDecimal IznosD =
       BlagajnaSqlExecutor.selectIznosDnevnice(this.getDBTransaction(), inom, drz);
    if(IznosD == null) throwBseException("Nemate postavljenu vrednost za dnevnice!");

                // dobijanje valute za drzavu i mesto
                // kolona valid se kupi iz tabele dnevnica
    BigDecimal ValutaD =
       BlagajnaSqlExecutor.selectValidDnevnice(this.getDBTransaction(), inom, drz);
    BigDecimal iznosDnevnica = (IznosD.multiply(brDnevnica));

       // S-funkcije služe za pristup bilo kom atributu na osnovu primarnog ili alternativnog ključa,
       // tako što iz generisanog reda vadimo šta nam je potrebno;
       // u ovom slučaju, pristupamo koloni skr iz tabele valuta
     BseSVector bv = getSSif().rowValuta(ValutaD, this.getDBTransaction());
     String val = (String)bv.nadjiElId(getSSif().VALUTA_SKR);
     setValutaSifra(val);

     setAttributeInternal(IZN, iznosDnevnica);
               //na osnovu valute, banke i odgovarajuceg datuma (datum sa dokumenta) vrši
              //se izračunvanje iznosa
     BigDecimal iznDin = izracunajIznosUDinarima(iznosDnevnica);
     setAttributeInternal(IZNDIN, iznDin);
     setAttributeInternal(BRSATI, value);

  }

//CommandIzracunajBrojDnevica.java
public static BigDecimal execute(BigDecimal brSati)
{
    if((brSati == null) || (brSati.intValue() == 0)) return new BigDecimal("0");
    int brsati = brSati.intValue();
    double brDnevnica = brsati/24;
    int ostatak = brsati % 24;

    if((ostatak >= 8) && (ostatak < 12)) brDnevnica=brDnevnica+0.5;
    if((ostatak >= 12)) brDnevnica=brDnevnica+1;
    BigDecimal brDnev = new BigDecimal(brDnevnica);

    return brDnev;
  }

private BigDecimal izracunajIznosUDinarima(BigDecimal value)
{
       BigDecimal valid = this.getValid();
       BigDecimal ppid = InoBlagajnaKonstante.KURSNALISTA_PPID;
       if(valid == null) throwBseException("Morate uneti valutu!");
```

```
                //dobijanje datuma sa zaglavlja
        Date datdok = this.getMasterDate();
        if(datdok == null) throwBseException("Morati postaviti datum dokumenta!");
        BigDecimal iznDin = ValutaConvertor.execute(this.getDBTransaction(),valid, null,
                                                    value, datdok, ppid);
         if(iznDin == null) throwBseException("Nemate podatke u kursnoj listi za dan:
                                              "+datdok+" i banku: "+ppid);
         else
            iznDin = iznDin.setScale(2, BigDecimal.ROUND_UP);

         return iznDin;
  }

// metoda koja vrši konverziju iz jedne valute u drugu;
// definisana je u klasi ValutaConvertor.java
public static BigDecimal execute(DBTransaction transaction, BigDecimal validFrom,
                    BigDecimal validTo, BigDecimal izn, Date datValid, BigDecimal ppid)
{

    String query = "select klid from kursnalista where dat = to_date('"+datValid+"', 'yyyy-
                                                                    MM-dd') and ppid = "+ppid;
    if((izn == null)) return null;
    BigDecimal rez = null;
    try
    {
      PreparedStatement ps = transaction.createPreparedStatement(query, 1);
      ResultSet rs = ps.executeQuery();
      while(rs.next()){
        BigDecimal klid = rs.getBigDecimal(1);
        rez = convert(transaction, validFrom, validTo, izn, klid);
      }
      rs.close();
      ps.close();
      return rez;
    }
    catch (SQLException e)
    {
      e.printStackTrace();
      return null;
    }
  }

private static BigDecimal convert(DBTransaction transaction, BigDecimal validFrom,
                                      BigDecimal validTo, BigDecimal izn, BigDecimal klid)
  {
       if(validFrom == null)
     {
        BigDecimal kurs = getSrednjiKurs(transaction, validTo, klid);
                //select srednji from kurs where valid = "+valid+" and  klid = "+klid;
        if(kurs == null) return null;
         BigDecimal rez = izn.divide(kurs,2).setScale(2, BigDecimal.ROUND_UP);
         return rez;
      }
```

```
    BigDecimal kurs = getSrednjiKurs(transaction, validFrom, klid);
    BigDecimal value = null;
    if(kurs != null) value = kurs.multiply(izn);
    if(validTo == null) return value;
    kurs = getSrednjiKurs(transaction, validTo, klid);
    if(kurs != null) kurs = value.divide(kurs,BigDecimal.ROUND_UP).setScale(2);
    return kurs;
 }
```

2.2 Metoda overe putnog računa

Ova metoda definisana je u klasi IbldokumentAmImpl.java

```
private void overaPutnogRacuna()
{
    // uzima se kurs Narodne banke
  BigDecimal ppid = InoBlagajnaKonstante.KURSNALISTA_PPID;
  IbldokumentVwRowImpl  red =
              (IbldokumentVwRowImpl)   this.getIbldokumentVw().getCurrentRow();
  BigDecimal vdidNaloga =
              InoBlagajnaKonstante.VDID_NALOG_DEVIZNOJ_BLAGAJNI_NOVCA;
  BigDecimal dokidRacuna = red.getDokid();
  BigDecimal vdgen = InoBlagajnaKonstante.VDID_PUTNI_NALOG;
  BigDecimal vdgenb = InoBlagajnaKonstante.VDID_PUTNI_RACUN;
  BigDecimal dokgen = red.getDokgen();
  BigDecimal iznRacuna = red.getIzndin();
  Date datov = red.getDatdok();
  BigDecimal valid =
        BlagajnaSqlExecutor.GetValutuPutnogRacuna(this.getDBTransaction(), dokgen);
  BigDecimal iznRacunaDev = red.getIzn();

  String zadatak = red.getZadatak();
  String opis = red.getOpis();
  String prevoz = red.getPrevoz();
  BigDecimal pozicijaid = red.getPozid();
  BigDecimal osobaId = red.getRdnid();
  BigDecimal rdnuneo = red.getRdnuneo();
  String status = InoBlagajnaKonstante.STATUS_GENERISAN;
  BigDecimal iznNaloga = red.getIzndinfakt();
  Date  dat = BlagajnaSqlExecutor.selectDatumDokumenta(this.getDBTransaction(),
                                                       vdgen, dokgen);

              // Iznos u valuti za putni nalog
  BigDecimal iznNalogaDev =   ValutaConvertor.execute(this.getDBTransaction(),null,
                                                      valid, iznNaloga, dat, ppid);

  BigDecimal iznosD = null;
  String UI = "U";
  String UI_2 = "I";
  if ((iznRacuna.compareTo(iznNaloga)) == 1)
   {
```

```
        UI ="I";
        UI_2="U";
         iznosD = iznRacuna.subtract(iznNaloga);
     }
    else if ((iznRacuna.compareTo(iznNaloga)) == -1)
     {
         iznosD = iznNaloga.subtract(iznRacuna);
     }
    else if ((iznRacuna.compareTo(iznNaloga)) == 0)
    {
         iznosD = null;
         return;
     }

    BigDecimal izn =  ValutaConvertor.execute(this.getDBTransaction(),null, valid,
                                                    iznosD, datov, ppid);

    BlagajnaSqlExecutor.insertNalogBlagajni(this.getDBTransaction(), vdidNaloga,
                                                   pozicijaid, datov,UI,vdgenb, dokgen,
                        valid,izn,iznosD, osobaId, rdnuneo, status, zadatak, prevoz, opis);

    BlagajnaSqlExecutor.insertNalogBlagajni(this.getDBTransaction(),vdidNaloga,
                pozicijaid, datov, UI_2,vdgenb, dokgen, valid,izn,iznosD, osobaId, rdnuneo,
                status, zadatak, prevoz, opis);

//  KURSNA RAZLIKA

    BigDecimal iznosRazlike = null;
    Date datumNaloga =
   BlagajnaSqlExecutor.selectDatumDokumenta(this.getDBTransaction(), vdgen, dokgen);

    if (iznNalogaDev.compareTo(iznRacunaDev) == -1)
      {

      BigDecimal iznN = ValutaConvertor.execute(this.getDBTransaction(),valid, null,
                                                    iznNalogaDev, datumNaloga, ppid);
      BigDecimal iznR = ValutaConvertor.execute(this.getDBTransaction(),valid, null,
                                                    iznNalogaDev, datov, ppid);
      iznosRazlike = iznR.subtract(iznN);
      }
   else if (iznNalogaDev.compareTo(iznRacunaDev) >= 0)
    {
     BigDecimal iznN = ValutaConvertor.execute(this.getDBTransaction(),valid, null,
                                                    iznRacunaDev, datumNaloga, ppid);
      BigDecimal iznR = ValutaConvertor.execute(this.getDBTransaction(),valid, null,
                                                    iznRacunaDev, datov, ppid);
      iznosRazlike = iznR.subtract(iznN);

     }

    red.setSaldo(iznosRazlike.setScale(2,BigDecimal.ROUND_HALF_UP));

    // ispisivanje upozorenja
```

```
  String statuspn =
     BlagajnaSqlExecutor.GetStastusNalogaZaPut(this.getDBTransaction(), vdgen, dokgen);
  if((statuspn != null) &&
     !(statuspn.equals(InoBlagajnaKonstante.STATUS_OVEREN)))
   ThrowBseException("Ne mozete overiti ovaj dokument. Nemate overen putni nalog.");

 }  // kraj metode overaPutnogRacuna()
```

3. Prilog 3

Metoda za overu naloga deviznoj blagajni

Metoda za overu naloga deviznoj blagajni definisana je u klasi IbldokumentAmImpl. Date su i definicije njenih pomoćnih metoda.

```
private void overaNalogaDeviznojBlagajni()
 {

            // obezbeđivanje da nalog deviznoj blagajni može da se overi samo ako je putni
           // nalog (ili putni račun) na osnovu koga se isti  generiše overen
  IbldokumentVwRowImpl red =
         (IbldokumentVwRowImpl) this.getIbldokumentVw().getCurrentRow();
  String status = BlagajnaSqlExecutor.GetStastusNalogaZaPut(this.getDBTransaction(),
                                                            red.getVdgen(), red.getDokgen());
  if((status != null) && !(status.equals(InoBlagajnaKonstante.STATUS_OVEREN)))
                    throwBseException("Ne mozete overiti ovaj dokument. Nemate overen
                                              odgovarajuci nalog (putni nalog ili putni racun).");

  upisNalogaDeviznojBlagajni();
  ubaciNalogUDnevnik();
 }

public void upisNalogaDeviznojBlagajni()
{

  IbldokumentVwRowImpl red =
                  (IbldokumentVwRowImpl) this.getIbldokumentVw().getCurrentRow();
  BigDecimal vdidNaloga = red.getVdid();
  BigDecimal dokidNaloga = red.getDokid();
  String UI = red.getKodui();

   if (UI.equals("U"))
    {
      red.setUlaz(red.getIzn());
      red.setUlazdin(red.getIzndin());
    }
   else if (UI.equals("I"))
     {
       red.setIzlaz(red.getIzn());
       red.setIzlazdin(red.getIzndin());
```

```
    }
 }


public void ubaciNalogUDnevnik()
{

      IbldokumentVwRowImpl red =
              (IbldokumentVwRowImpl) this.getIbldokumentVw().getCurrentRow();
      Date datDok = red.getDatdok();
      BigDecimal vdidDnevnika =
             InoBlagajnaKonstante.VDID_DNEVNIK_DEVIZNE_BLAGAJNE_NOVCA;
      String stastusDnevnik =
             BlagajnaSqlExecutor.selectStatusDanasnjegDnevnika(this.getDBTransaction(),
                                                                    vdidDnevnika, datDok);
      if((stastusDnevnik == null))
       {
         kreirajDnevnik(red);
       }
      else if(stastusDnevnik.equals(InoBlagajnaKonstante.STATUS_OVEREN))
       {
          throwBseException("Dnevnik devizne blagajne za: "+red.getDatdok()+" je overen
                                    i ne mozete ubacivati nove naloge!");
      }
 }

private void kreirajDnevnik(IbldokumentVwRowImpl red)
{
       Date datumDok = red.getDatdok();
       BigDecimal vdidDnevnika =
              InoBlagajnaKonstante.VDID_DNEVNIK_DEVIZNE_BLAGAJNE_NOVCA;
       BigDecimal radnikUneo = red.getRdnid();
       BlagajnaSqlExecutor.kreirajNoviDnevnikBlagajne(this.getDBTransaction(),
                                                          vdidDnevnika, datumDok, radnikUneo);
 }

   // u klasi BlagajnaSqlExecutor.java
public static void kreirajNoviDnevnikBlagajne (DBTransaction dbTransaction, BigDecimal vdid,
                                                 Date datdok, BigDecimal radnuneo)
{
      String upit = ControlSql.createSqlKreirajNoviDnevnikBlagajne (vdid, datdok, radnuneo);
      getUpit ("default", dbTransaction).executeInsert (upit);
 }

   // u klasi ControlSql.java
public static String createSqlKreirajNoviDnevnikBlagajne(BigDecimal vdid,Date datdok,
                                                          BigDecimal radnikuneo)
{

        String query = " INSERT INTO IBLDOKUMENT(VDID,DATDOK,RDNUNEO,
                           STATUS,IZN,IZNDIN)"+
                         " VALUES ("+vdid+","+"to_date('"+datdok+"','yyyy-MM-dd'),"
```

```
+radnikuneo+",'"+InoBlagajnaKonstante.STATUS_EVIDENTIRAN+"',"+0+","+0+")";

        return query;
 }
```

4. Prilog 4

Obračun dnevnika

```
public void obracunDnevnika()
{
  IbldokumentVwRowImpl row =
                        (IbldokumentVwRowImpl)this.getIbldokumentVw().getCurrentRow();
  Date datdok = row.getDatdok();
  BigDecimal vdid =
              InoBlagajnaKonstante.VDID_DNEVNIK_DEVIZNE_BLAGAJNE_NOVCA;

  BigDecimal ulaz = izracunajUkupanUlaz(row, "Ulazdin");
  row.setUlazdin(ulaz);
  BigDecimal izlaz = izracunajUkupanUlaz(row, "Izlazdin");
  row.setIzlazdin(izlaz);

  Date datpre=
      BlagajnaSqlExecutor.vratiDatumZadnjegDnevnika(this.getDBTransaction(), datdok,
                                                                    vdid);
  row.setDatDokPre(datpre);
  BigDecimal dokidpre = null;
  BigDecimal saldopre = InoBlagajnaKonstante.createBigDecimalNula();

  if (datpre != null)
  {
     dokidpre  =
                BlagajnaSqlExecutor.selectIdDanasnjegDnevnika(this.getDBTransaction(),
                                                                    vdid, datpre);
     saldopre =
                 BlagajnaSqlExecutor.vratiSaldoZadnjegDnevnika(this.getDBTransaction(),
                                                                    dokidpre, vdid);
     row.setSaldoDescr(saldopre);
   }

  if (saldopre == null)
     {
        saldopre.equals(InoBlagajnaKonstante.createBigDecimalNula());}
        BigDecimal primetak = ulaz.add(saldopre);
        row.setPrimuk(primetak);
        row.setIzduk(izlaz);
        BigDecimal saldo = primetak.subtract(izlaz);
        row.setSaldo(saldo);
     }
}
```