Sustav podrške za računovodstvene servise “Dobby”

Dizajn specifikacija

PA1

11.05.2017.

Marko Nikolić
Stručnjak za dizajn

Namijenjeno:
MAZNI d.o.o.

Revizije

Datum	Opis	Autor	Komentari
11.05.2017.	Prva radna verzija	Marko Nikolić	Class dijagram, okvir dokumenta

Suglasnost i odobrenje dizajna specifikacija

Svi potpisani članovi tima suglasni su s konačnom verzijom specifikacije zahtjeva:

Potpis	Ime i prezime	Kvalifikacija	Datum
	Marko Nikolić	Stručnjak za dizajn	
	Sandra Dobrić	Stručnjak za specifikaciju	
	Anna-Maria Mihel	Projekt manager	

Sadržaj

1. Uvod

Za prikaz dizajna je korišten objektno orijentirani model, konkretno CRC metodologija za izradu modela, te u konačnici prikaz modela dijagramom klasa i dijagramom slijeda izrađenih pomoću StarUML-a. Za izradu CRC kartona pratiili su se zahtjevi iz prethodno napravljene specifikacije zahtjeva, te savjeti i prijedlozi cjelokupnog tima kako ih rasporediti u pojedine klase.

1.1 Kome je dokument namijenjen

Ciljna publika konačne verzije dizajna specifikacija su:

1) Ines Krnjus – stručnjak za testiranje, zadužena za detaljno i temeljito testiranje svih navedenih funkcionalnosti softwarea u skladu sa specifikacijom
2) Nemanja Šimpraga – stručnjak za kvalitetu, zadužen za praćenje razvoja softwarea i usporedbu sa njegovom specifikacijom
3) Matej Burić – stručnjak za verzioniranje

Primjerak konačne verzije dizajna specifikacija također će biti dostavljen i:

1) Anna-Maria Mihel – project manager tvrtke MAZNI d.o.o.

1.2 Sažetak faze dizajna

Prvotno je svaki član tima dao svoje prijedloge o načinu na koji ćemo podijeliti funkcionalnosti i zahtjeve iz specifikacije zahtjeva na cjeline. Izradili smo CRC kartone u obliku prikazanom na *slici 1.1.* koja se nalazi ispod, ali pošto je svaki član zapravo imao neku svoju sliku kako bi to sve trebalo izgledati, ih je trebalo povezati u nešto smisleno. To je učinjeno od strane mene uz savjete stručnjaka za specifikaciju, Sandre Dobrić.

IME KLASE:
PODKLASA:
NADKLASA:
ODGOVORNOSTI:
SURADNICI:

Slika 1.1.

Nakon što smo imali CRC kartone sam pomoću alata StarUML napravio dijagram klasa uz daljnje savjetovanje sa stručnjakom za specifikaciju, kako bi sve bilo objedinjeno.

1.3 Definicije i kratice

Korisnik – zaposlenik računovodstvenog servisa.

Admin – šef računovodstvenog servisa, ujedno može biti i korisnik sustava.

Partner – svako poduzeće klijent računovodstvenog servisa.

1.4 Reference

1.5 Pregled

U ostaku dokumenta će biti prikazana matrica specifičnih zahtjeva i dijagram klasa.

2. Dizajn arhitekture sustava

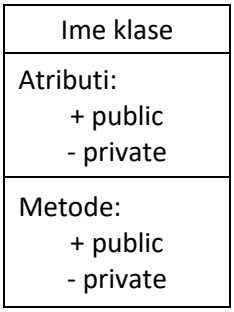
2.1 Matrica praćenja zahtjeva

REDNI BROJ ZAHTJEVA U SPECIFIKACIJI IZ LISTE SPECIFIČNIH ZAHTJEVA	POGLAVLJE U DETALJNOM PREGLEDU KOMPONENTI
3.1.1.	2.2.5.
3.1.2.	2.2.6.
3.1.3.	2.2.11.
3.1.4.	2.2.9. 2.2.10.
3.1.5.	2.2.8.
3.1.6.	2.2.7.
3.1.7.	2.2.4.
3.1.8.	2.2.3.

2.2 Detaljan pregled komponenti

2.2.1. Legenda znakova

- Na *slici 2.1.* je prikazan predložak po kojem je izrađen dijagram klasa, te što koji znak predstavlja unutar pojedine ćelije.
- Linije koje izlaze iz svake klase u dijagramu predstavljaju poveznice sa drugim klasama i sam tip poveznice.
    - Crvene linije predstavljaju poveznicu sa pomoćnom klasom za komunikaciju sa bazom podataka.
    - Plave linije predstavljaju poveznicu prema klasi *Izbornik*.
    - Zelene linije predstavljaju poveznicu prema pomoćnoj klasi za ispis.
    - Ljubičasta linija predstavlja poveznicu između početne klase Prijava i Izbornika.
    - Narančasta linija predstavlja poveznicu u izboru poslovnih knjiga.

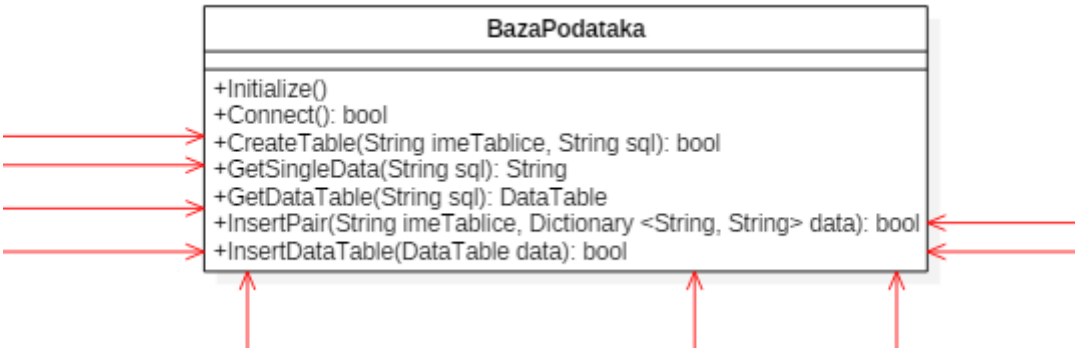


Slika 2.1.

2.2.2. Baza podataka

Pošto se većina rada sustava svodi na komunikaciju sa bazom podataka što se vidi na *slici 2.2.*, te je jako bitno da bude unčikovito smišljena je posebna klasa koja bi služila svim ostalim klasama upravo u tu svrhu. Bit će inicijalizirana jednom pri pokretanju programskog sustava te će sve ostale klase dobiti referencu na objekt kada im to bude potrebno. Na taj način nećemo stvarati nepotreban broj objekata i konekcija prema bazi.

Kao što se vidi na *slici 2.2.* klasa nema nikakvih atributa iz razloga što su nam za rad te klase potrebne samo privremene varijable, i na taj način ne zauzimamo nepotrebne resurse. Jedna od takvih varijabli je i *connection string* prema kojem će klasa dobiti uvid gdje se program nalazi i gdje bi u odnosu na to trebala biti i sama baza podataka, to se odvija u metodi *Intialize()* koju je potrebno pozvati prije svih ostalih metoda klase. Metodom *Connect()* ,koja vraća „boolean“ vrijednost ovisno o statusu, se povezujemo sa bazom podataka, a u slučaju prvog pokretanja programa dok još nemamo bazu podataka u toj funkciji se kreira i inicijalizira sve potrebne tablica za početak rada pomoću metode *CreateTable()* koja vraća „true“ vrijednost ako je kreirano uspješno. Ostale metode služe za dohvaćanje ili upisivanje različitih tipova podataka. Pri upisivanju također dobivamo povratnu „boolean“ vrijednost da li je uspješno ili neuspješno uneseno u bazu podataka. Sve su metode public da im možemo jednostavno pristupiti iz ostalih klasa.



Slika 2.2.

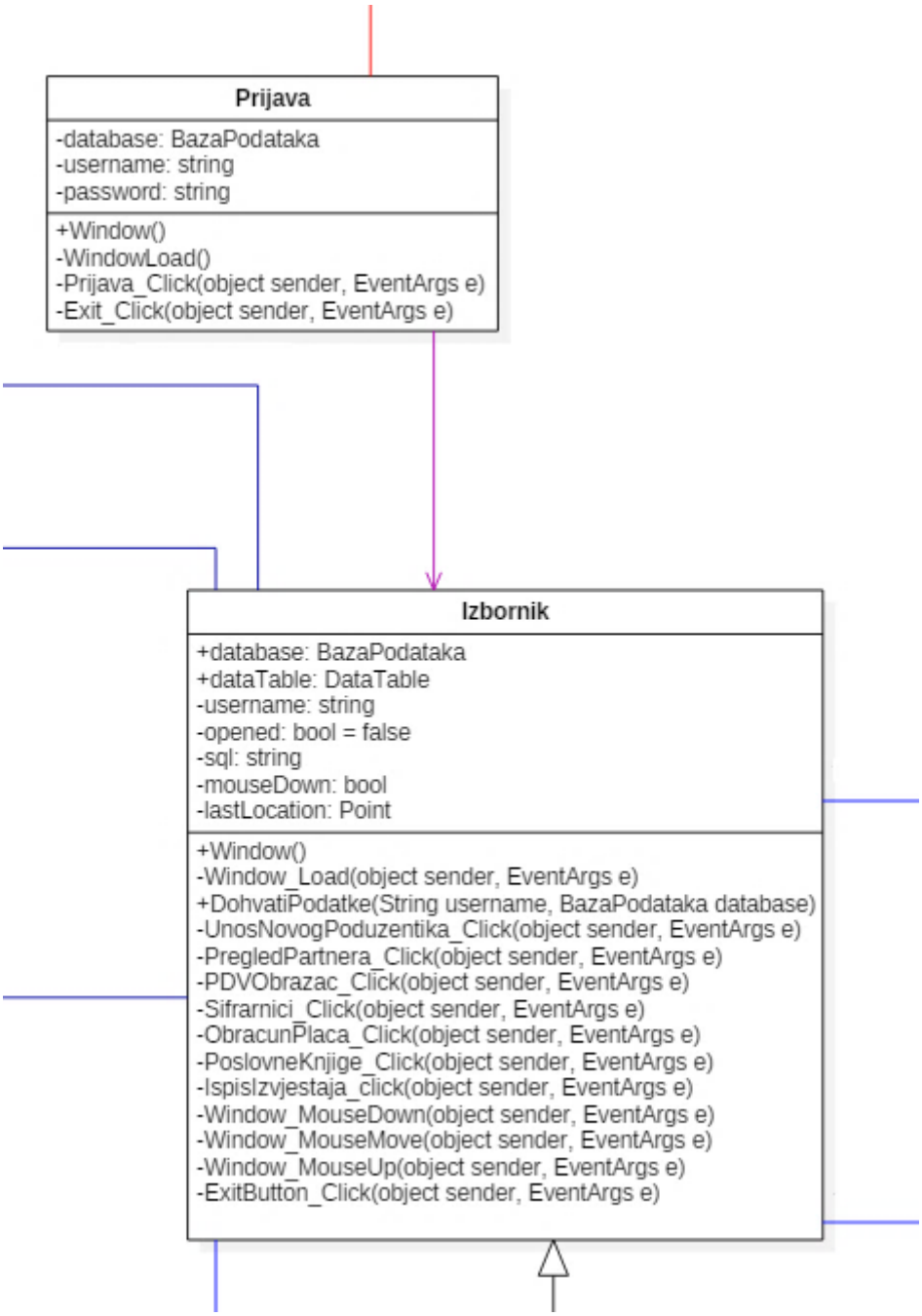
2.2.3. Izbornik i Prijava

Pri pokretanju programa prvo dolazi prozor, koji je zasebna klasa *Prijava*, za prijavu u sustav, te uspješnom prijavom dobivamo centralnu klasu *Izbornik* koji nadalje služi za otvaranje posebnih funkcionalnosti programskog sustava kao što je prikazano na *slici 2.3.*. Klasa *Prijava* također inicijalizira klasu *BazaPodataka* i prenosi referencu na klasu *Izbornik* koja nadalje to prenosi svim ostalim klasama kojima je potrebno. Prethodno navedene dostupne funkcionalnosti su također realizirane posebnim klasama što nam omogućuje lakšu implementaciju, predstavlja i inkrementalni model koji smo odabrali kao model rada, te omogoćuje jednostavniju eventualnu buduću nadogradnju sustava novim funkcionalnostima. Dovoljno je dodati vezu iz klase Izbornik na novu klasu. Sam izgled klase *Izbornik* ovisi o tome jeli osoba koja se prijavljuje u sustav obican korisnik ili admin. U slučaju prijave običnog korisnika otvara se *Izbornik*, a ako se prijavio admin postoji klasa *IzbornikAdmin* koja nasljeđuje *Izbornik* i sve njegove atribute i metode, te dodaje svoje vlastite dostupne samo adminu sustava.

Metode *Window()* i *WindowLoad()* su temeljne funkcije za inicijalizaciju prozora i GUI elemenata kao što su buttoni, tekstualna polja i slično. Te metode se provlače kroz svaku klasu pošto se sve funkcije programskog sustava otvaraju u zasebnom prozoru od Izbornika. Takav pristup nam daje veću mogućnost prilagođavanju svakog prozora sadržaju koji treba prikazati, dodavanje novih funkcionalnosti, odnosno prozora, i pri samom pokretanju programa treba se puno manje GUI elemenata učitati i ne trošimo bespotrebno resurse.

Klasa *Prijava* instancira novi objekt BazaPodataka. Taj objekt će dalje proslijediti *Izborniku* ako dođe do uspješne prijave. Varijable username i password su esencijalno 2 textboxa pa nam te varijable nisu potrebne jer možemo direktno dobiti podatke iz textboxa. Pri inicijalizaciji prozora pozivaju se i metode iz klase *BazaPodataka* za inicijalizaciju i spajanje sa bazom, *Initalize()* i *Connect()*. Metode *Prijava_Click()* i *Exit_Click()* se pokreću kada korisnik klikne na određeni button. Metoda *Prijava_Click()* šalje upit u bazu podataka da li postoji ta kombinacija username-a i password-a i ako postoji daje pristup izborniku, ako ne dobivamo upozorenje da smo unijeli krive podatke. Prije samog prikazivanja Izbornika pozivamo moramo ga instancirati pa proslijediti referencu na objekt BazaPodataka i dati username za prikaz pozdravne poruke. Podatke proslijeđujemo preko public metode *DohvatiPodatke* koja se nalazi u klasi *Izbornik.*

U klasi *Izbornik* se nalazi više atributa nego u ijednoj klasi do sad. Varijabla opened tipa boolean je uvedena zbog zahtjeva iz specifikacije da se ne može izaći iz aplikacije ako je bilo koji drugi prozor otvoren. Pri otvaranju bilo kojeg prozora varijabla se postavlja na vrijednost „true“ što se provjerava u svim metodama, tako da onemogućuje i da se otvori beskonačno mnogo prozora. Varijable mouseDown i lastLocation omogućuju da se prozor pomiče po ekranu klikom i micanjem miša. Metode se sastoje od skupa *event listenera* koji reagiraju na pojedini klik na buttone i instanciraju novu klasu, pozivaju metodu *DohvatiPodatke()* iz te klase i tek onda prikazuju prozor te klase, ali naravno to se sve događa ako varijabla opened ima vrijednost „false“, ako je vrijednost „true“ otvara se jednostavan messagebox koji obavještava korisnika da je neki prozor otvoren. Klikom na *ExitButton* se izlazi iz aplikacije.



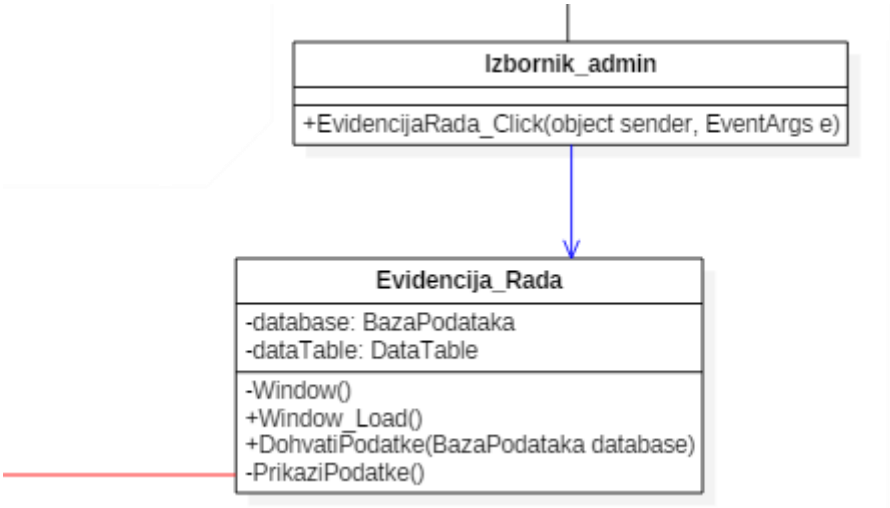
Slika 2.3.

2.2.4. Izbornik za admina i evidencija rada

Za sada jedina dodatna mogućnost koju admin posjeduje je pristup evidenciji rada. Navedena klasa daje uvid u to kako sustav posluje i pri tome povlači podatke iz baze podataka, prikazano na *slici 2.4.*. Ovakav pristup nam omogućuje lakše dodavanje mogućnosti adminu sustava u budućnosti ako do toga dođe.

Pri pokretanju prozora se poziva i metoda *PrikaziPodatke()* koja šalje upit bazi podataka i organizira podatke u tablice, za to koristi varijablu dataTable u koju se vrlo jednostavno mogu spremiti podaci iz baze podataka metodom *GetDataTable()* koja zahtjeva string kao parametar, ali nije potrebno taj string

čuvati cijelo vrijeme u memoriji jer se podaci dohvaćaju samo jedno pri otvaranju prozora i dovoljna je privremena varijabla unutar metode *PrikaziPodatke()*.

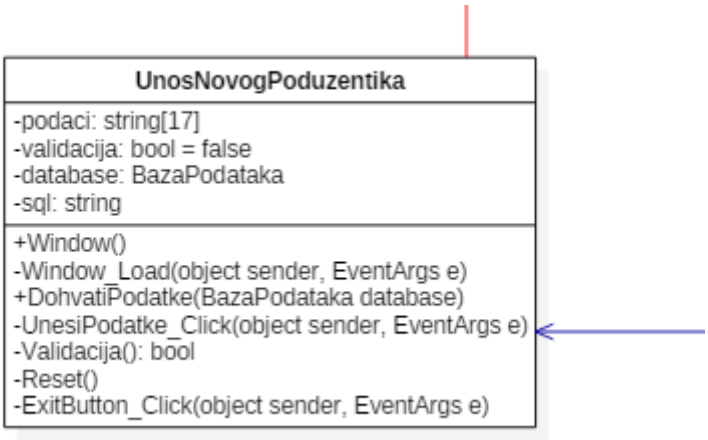


Slika 2.4.

2.2.5. Unos novog partnera

Jedna od bitnijih funkcija iz razloga što ako nismo unijeli nijednog poduzetnika nemamo sa čime raditi u svim ostalim funkcijama pa ju je potrebno implementirati među prvima, odmah nakon klase za komunikaciju sa bazom podataka i klase Izbornik iz koje ćemo pristupati ovoj klasi, što se vidi na *slici 2.5.*. Također treba provjeravati unesene podatke što radi vlastitom metodom.

U atributima vidimo varijablu podaci koja je polje od 17 stringova. Ta varijabla zapravo predstavlja GUI elemente kao što su *textboxovi* i padajući izbornici pa će se direktno u njima provjeravati za svaki pojedinu stavku koju treba unijeti da li odgovara onome što je navedeno da mora, ili ne smije sadržavati. Tako na primjer polje za unos OIB-a ne smije imati više od 11 znamenaka i moraju biti samo brojevi. Taj konkretan slučaj će se riješiti na način da ograničimo u textboxu prostor koliko znakova možemo unijeti i jednako tako namjestiti da se ne mogu unositi slova nego samo brojke. Na kraju se pritiskom na *button* Unesi poziva metoda *Validacija()* koja vraća bool vrijednost „true“ ako je sve ispravno uneseno i ako su sva polja popunjena i omogućuje daljnje napredovanje metode *UnesiPodatke_Click()* koja podatke šalje u bazu preko metode iz klase *BazaPodataka* metodom *InsertPair()* pošto su podaci poredani u parove (ime textboxa, vrijednost unutar) i spremaju se u Dictionary<string, string> tip varijable. Metoda *Reset()* se poziva nakon uspješnog unosa ili ako korisnik želi resetirati sva polja tokom unosa i u toj modi se svi textboxovi postavljaju na početne vrijednosti što je uglavnom prazno polje. Jedino polje koje se neće moći uređivati je polje sa šifrom partnera koje će dobiti vrijednost automatski i koja će biti prva iduća slobodna vrijednost prema bazi podataka.

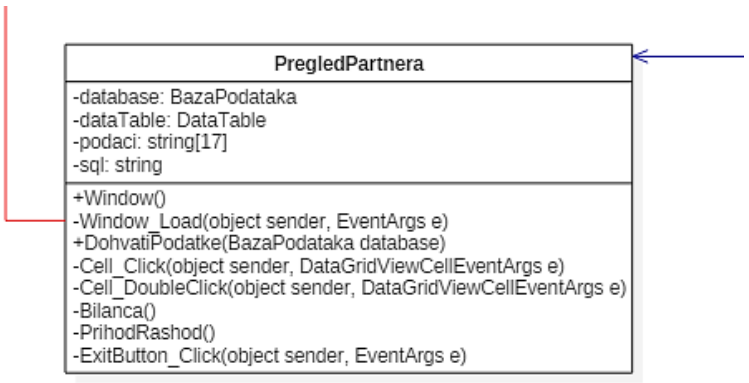


Slika 2.5

2.2.6. Pregled partnera

Daje mogućnost pregleda partnera abecedno u listi, te pretraživanje istih. Klikom na partnera daje dodatni pregled te dupli klik omogućava uređivanje podataka koji se nalaze u bazi podataka.

Cijela lista partnera će biti spremljena u varijablu dataTable tipa DataTable u koje ćemo upisati podatke pozivom metode *GetDataTable()* iz klase *BazaPodataka*. Također u sql komandi će biti dodan „ORDER BY name“ da bi dobili sortiranu listu po abecedi. Klikom na određenu ćeliju korisnik će dobiti više informacija o partneru koji se nalazi u tom redu, a duplim klikom omogućuje mu se i uređivanje tih podataka. Evente i funkcionalnosti za to obavljaju metode *Cell_Click()* i *Cell_DoubleClick()*. Pozivom metode *Bilanca()* prikazat će se tablica sa dva stupca, u lijevom vrijednost pasive, a u desnom vrijednost aktive. Također pozivom metode *PrihodRashod()* šalje se upit u bazu podataka i dobiva se zbrojena vrijednost iz knjiga primitaka i izdataka, odnosno prihod i rashod, te se dobivene vrijednost oduzimaju i sustav daje informaciju u messageboxu da li partner posluje sa dobitkom ili gubitkom.

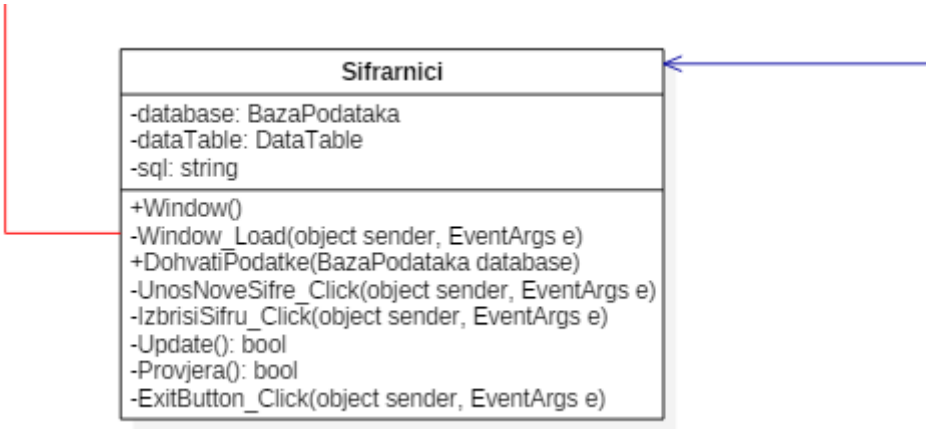


Slika 2.6.

2.2.7. Šifrarnici

Unos, uređivanje i brisanje šifrarnika. U slučaju uređivanja provjerava i ažurira podatke u svim ostalim tablicama, odnosno knjigama u bazi podataka, a brisanje je dozvoljeno samo ako nijedna druga tablica ne sadrži podatke određenog šifrarnika. Za rad je potrebna konekcija sa bazom podataka. Također podaci i šifrarnika su nužni za rad u klasama koje slijede.

Pri prvom pokretanju programa i kreiranju baze podataka također se kreira tablica šifrarnika te se definiraju tipovi PDV-a. Kad unosimo novog partnera njegova se šifra također automatski unosi u šifrarnik što se može vidjeti unutar pregleda šifrarnika prikazanog u tabličnom obliku. U slučaju da želimo ažurirati neku šifru ona se mijenja na svim mjestima gdje je unesena metodom *Update()* i ta nam metoda vraća bool vrijednost ovisno o tome da li je uspjelo ili ne. Kada brišemo neku šifru poziva se i metoda *Provjera()* koja vraća bool vrijednost da li je ta šifra u upotrebi ili ne, i ako nije tek onda se briše iz šifrarnika.

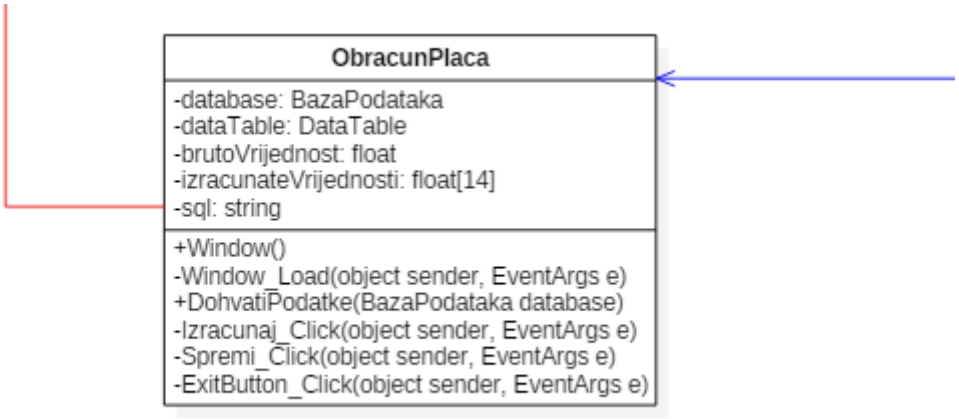


Slika 2.7.

2.2.8. Obračun Plaća

Omogućuje korisniku da unese iznos bruto plaće i daje mu tablični prikaz svih podataka izračuna. Također korisnik može obračun spremiti u bazu podataka što naravno zahtjeva i referencu na bazu podataka.

Klikom na button Izračunaj i ako je vrijednost bruto plaće unesena vrši se računanje podataka i spremaju se u float polje i na kraju se prikazuju u tabličnom obliku. U slučaju da korisnik nije ništa upisao u polje za unos bruto plaće dobiva upozorenje u obliku textboxa da je potrebno unijetu tu vrijednost. Također ne može unositi ništa osim znamenki i znaka „.“ za razdvajanje dekadske i decimalne znamenke.

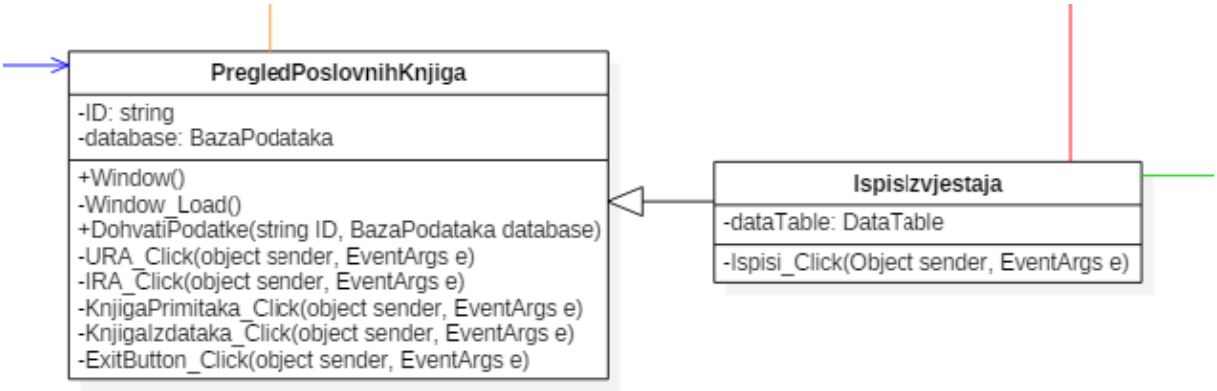


Slika 2.8.

2.2.9. Pregled poslovnih knjiga i Ispis izvještaja

Ovisno o tome što je korisnik izabrao u *Izborniku* otvara se *PregledPoslovnihKnjiga* ili podklasa *IspisIzvjestaja* koja dodaje mogućnost ispisa preko pomoćne klase za ispis bez otvaranja knjiga. Postojeće knjige su poredane u listi te ih tamo i korisnik odabire za ispis. Ako korisnik želi uređivati postojeću knjigu onda ulazi u *PregledPoslovnihKnjiga* i tamo odabire knjigu u koju želi unositi podatke te ga to ovisno o odabranoj knjizi vodi dalje u novi prozor, odnosno otvara se nova klasa.

U ovom prozoru imamo popis svih knjiga koje se nalaze u bazi podataka određenog partnera prikazane u listi. Uređivanje se vrši u drugoj klasi i drugom prozoru koji se instancira i pokreće klikom na odabranu knjigu i klikom na button za tu knjigu. Ako nije nijedna knjiga odabrana stvara se nova knjiga odabranog tipa. Varijabla ID predstavlja partnera čije knjige pregledavamo.



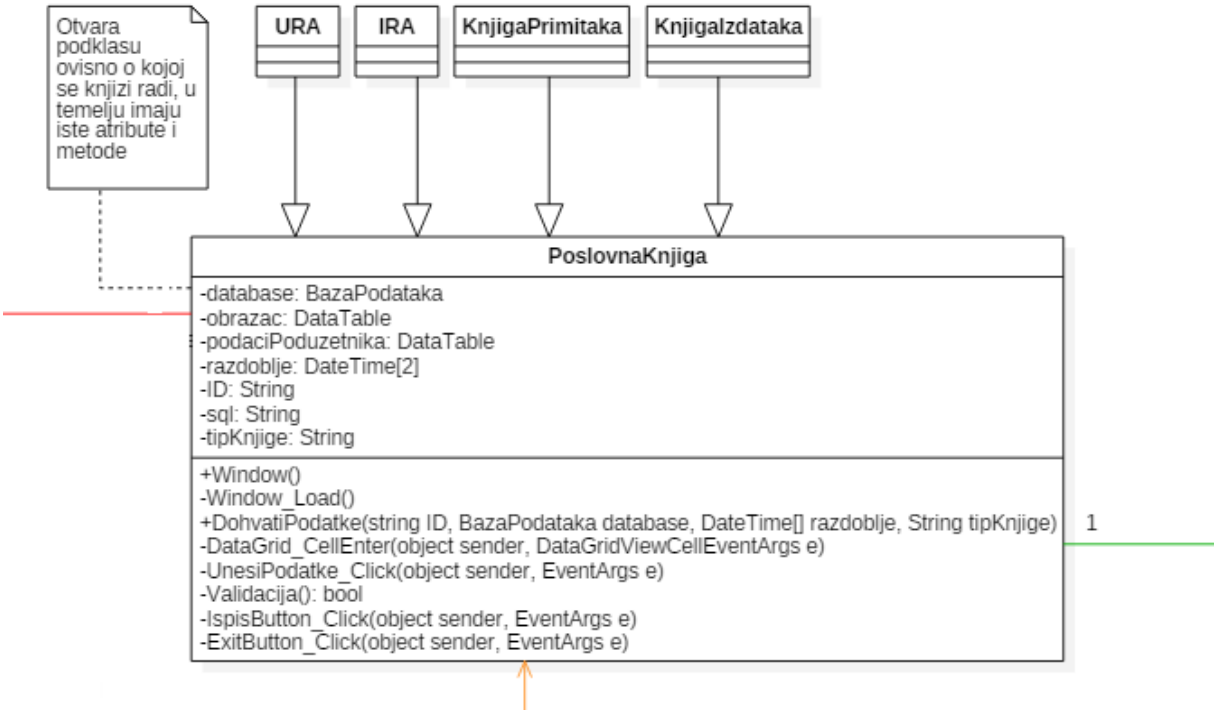
Slika 2.9.

2.2.10. Poslovna knjiga (URA, IRA, Knjiga primitaka/izdataka)

PoslovnaKnjiga predstavlja predložak za sve ostale knjige. Otvara se u novom prozoru i omogućava korisniku pregled, dodavanje i izmjenu podataka. Ovisno o kojoj knjizi se radi sama tablica u koju se unosi izgleda drugačije, te naravno razlikuje se i mjesto u bazi odakle se povlače i gdje se spremaju

podaci. Svaka knjiga ima i pristup pomoćnoj klasi *Ispis* preko koje ispisuju podatke. Uz to imamo i metodu *Validacija()* koja projevarava da li su uneseni podaci ispravni.

Ako postoje podaci prikazuju se u tabličnom obliku i u toj tablici se mogu uređivati, i ti podaci se direktno spremaju u varijablu obrazac tipa DataTable i preko metode *UnesiPodatke()* validiraju i spremaju u bazu podataka pomoću klase *BazaPodataka* i njene metode *InsertDataTable()*.

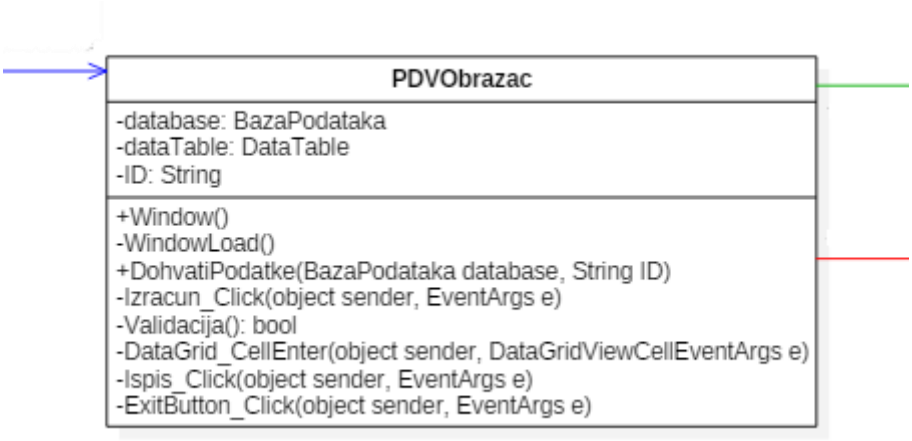


Slika 2.10.

2.2.11. PDV Obrazac

Ovoj klasi za normalno funkcioniranje su potrebni podaci iz knjiga URA i IRA koji se nalaze u bazi podataka pa zato ovu klasu treba implementirati tek nakon što su klasa *PoslovneKnjige* i njene podklase završene. Uz automatsko povlačenje podataka iz baze, također korisniku daje konačan izračun u tabličnom obliku i omogućuje spremanje obrasca u bazu podataka i ako to želi ispis.

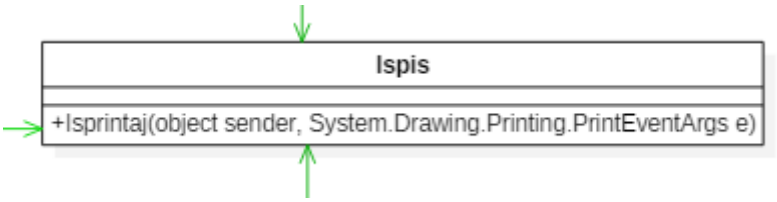
Da ne bi došlo do pogreške pri unosu imena partnera i ostalih podataka u padajućem izborniku su ponuđeni svi partneri koji se nalaze u sustavu, odnosno u bazi podataka. Također u varijabli ID je spremljeno i poslano iz izbornika ime knjigovođe koji ispunjava PDV obrazac. Klikom na button Izračun poziva se i metoda *Validacija* koja vraća bool vrijednost ovisno da li su sva polja ispunjena ili je neko polje ostalo prazno. Button Ispis poziva instancira klasu *Ispis* preko koje se tablica može i ispisati pozivanjem metode Isprintaj.



Slika 2.11.

2.2.12. Ispis

Ova pomoćna klasa pruža funkcije potrebne za ispis podataka, i poziva se samo kada je to potrebno. Zbog toga se može implementirati među zadnjima, ili zbog potreba testiranja za vrijeme implementacije poslovnih knjiga.



Slika 2.12.