Programsko rješenje blagajne u Javi

Božić, Ema

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTHENIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLGIJA

Diplomski studij

PROGRAMSKO RJEŠENJE BLAGAJNE U JAVA-i

Diplomski rad

EMA BOŽIĆ

Osijek, 2016.

SADRŽAJ

1. UVOD

Java je najpopularnija razvojna platforma koja ima široku primjenu u razvoju informacijskih sustava. Koristi se za razvoj širokog spektra programskih rješenja, od poslužiteljskih aplikacija, desktop aplikacija, web aplikacija do aplikacija za mobilne uređaje, pa čak i pametne kartice. Java platforma dostupna je za većinu današnjih operacijskih sustava (Windows, Unis, Linux, Mac) i potpuno je prenosiva između njih. [1]

Aplikacija „Salon za pse“ zamišljena je kao pomoć pri vođenju salona za uređivanje kućnih ljubimaca, tako što olakšava organizaciju te upravljanje podacima o zaposlenicima, korisnicima salona, uslugama koje su u ponudi i slično. Osim opcija dodavanja novih, promjene postojećih te brisanja zaposlenika/korisnika/usluga/pasa, omogućeno je i kreiranje računa. Odabirom opcije „Nova posjeta“ otvara se forma u kojoj se odabire korisnik, zaposlenik i datum te se u tablicu mogu dodati obavljene usluge i količina istih. Nakon unosa podataka klikom na gumb „Ispis računa“ kreira se PDF dokument, tj. račun, koji se pohranjuje lokalno na računalu.

Također se mogu pogledati i statistički podaci o najčešće korištenim uslugama prikazani tortnim grafikonom.

1.1 Zadatak diplomskog rada

Objasniti način rada blagajne. Modelirati bazu podataka koja može podržati aplikaciju. U programskom jeziku Java/J++ izraditi funkcionalnosti blagajne. Opisati način izrade aplikacije. Testirati rad aplikacije.

2. KORIŠTENE TEHNOLOGIJE

Aplikacija je rađena u razvojnom okruženju *NetbeansIDE* (inačica 8.1) uz *Java development Kit* (inačica 8.0). Pisana je Java programskim jezikom (razvojni okvir *Swing*). Temeljena je na MVC (engl. *Model – View – Controller*) modelu. Baza aplikacije je SQL (engl. *Structured Query Language*) relacijska baza podataka (verzija 5.6.24).

2.1 PROGRAMSKI JEZIK JAVA

Java je objektno orijentirani programski jezik temeljen na C++ jeziku, razvijen u tvrtci *Sun Microsystems* 1995. godine. Najveća prednost, obzirom na ostale programske jezike, je što se programi pisani u Java-i mogu izvoditi na svim operacijskim sustavima koji posjeduju JVM (engl. *Java Virtual Machine*) bez ikakvih prilagođavanja. Java kôd prevodi se u u tzv. *bytecode* kojeg pokreće operacijski sustav, drugi program ili uređaji pomoću Java interpretera. *Bytecode* nije izvršni kôd, već visokooptimizirani skup instrukcija dizajniran za izvršavanje unutar Java izvršnog sustava koji predstavlja interpreter za *bytecode*. Još jedna u nizu značajki programskog jezika Java je podrška za višenitno programiranje. [2]

Broj korisnika koji se kreće od 7, do preko 10 milijuna dokazuje kako je Java jedan od najkorištenijih programskih jezika.

2.1.1 Objektna orijentiranost Java-e

Iako je temeljena na C++ strukturi, koja dozvoljava korištenje i strukturnih principa, u Java-i su obavezni objektno orijentirani principi, to jest sve predstavlja objekt, a sav izvorni kôd piše se unutar klasa (engl. *class*), čak i glavna aplikacija. [3]

Klasa (engl. *class*) je osnovica modularnosti i strukture objektno-orijentiranog programa. Primjerice, u ovoj aplikaciji, jedna od klasa je *Zaposlenik* te sadrži osnovne informacije koje mora sadržavati instanca te klase, konkretno: ime, prezime, OIB i plaću. Instanca klase naziva se objekt (engl. *object*) i sukladno ovom primjeru to može biti nekakav zaposlenik, primjerice Zara Marić.

Osim navedenog, Java se temelji na četiri principa: enkapsulacija (engl. *encapsulation*) koja predstavlja koncept po kojem je informacija u klasi zaštićena od direktnog pristupa i jedini način promjene informacija je kroz utemeljene metode, nasljeđivanje (engl. *inheritance*) kao koncept po kojem se može definirati klasna hijerarhija, apstrakcija (engl. *abstraction*) koja obuhvaća pojednostavljivanje karakteristika objekta, gdje se zanemaruju detalji i uzimaju se samo zajedničke

karakteristike klase te polimorfizam (engl. *polymorphism*) kao karakteristika klase čije ponašanje se mijenja ovisno o implementaciji. [3]

2.1.2 Razvojno okruženje NetBeansIDE

NetBeansIDE[1] je razvojno okruženje pisano u Java-i koje omogućava razvoj aplikacije kao skupa programskih komponenata nazvanih moduli (engl. *modules*). Zamišljeno je kako bi olakšalo programiranje za Java *Swing* razvojni okvir, prvenstveno za desktop aplikacije. Glavne značajke okruženja su: mogućnost upravljanja korisničkim sučeljem (izbornici, alatne trake, itd.), upravljanje korisničkim postavkama, upravljanje resursima (spremanje i učitavanje podataka), upravljanje prozorima (engl. *frame*) te čarobnjak koji omogućava razvijanje aplikacije korak po korak. [4]

2.1.3 Java Development Kit

Java Development Kit je okruženje za razvijanje i pokretanje Java aplikacija koje obuhvaća i JRE (engl. *Java Runtime Environment*). Neka razvojna okruženja već imaju ugrađene ove dodatke. Isti su obavezni kako bih mogli pisati i pokretati Java aplikacije. [5]

2.1.4 Model View Controller

Model-View-Controller je obrazac arhitekture softvera. Princip obrasca je odvajanje pojedinih dijelova aplikacije u komponente ovisno o njihovoj namjeni. Model (engl. *model*) se sastoji od poslovnih pravila te funkcija ugrađenih u poslovnu logiku (engl. *business logic*). Pogled na prikaz podataka definiran je u komponenti *View*, a odnosi se na tablice, dijagrame, obrasce, itd. Prikaz podataka može biti ostvaren kroz više pogleda (engl. *views*). Upravitelj (engl. *controller*) prima ulazne podatke (engl. *input data*) te ih pretvara u naloge modelu ili pogledu. Ovakvim načinom pisanja aplikacije olakšan je nezavisan razvoj, preglednost, testiranje te održavanje aplikacije. [6]

2.2 RELACIJSKA BAZA PODATAKA

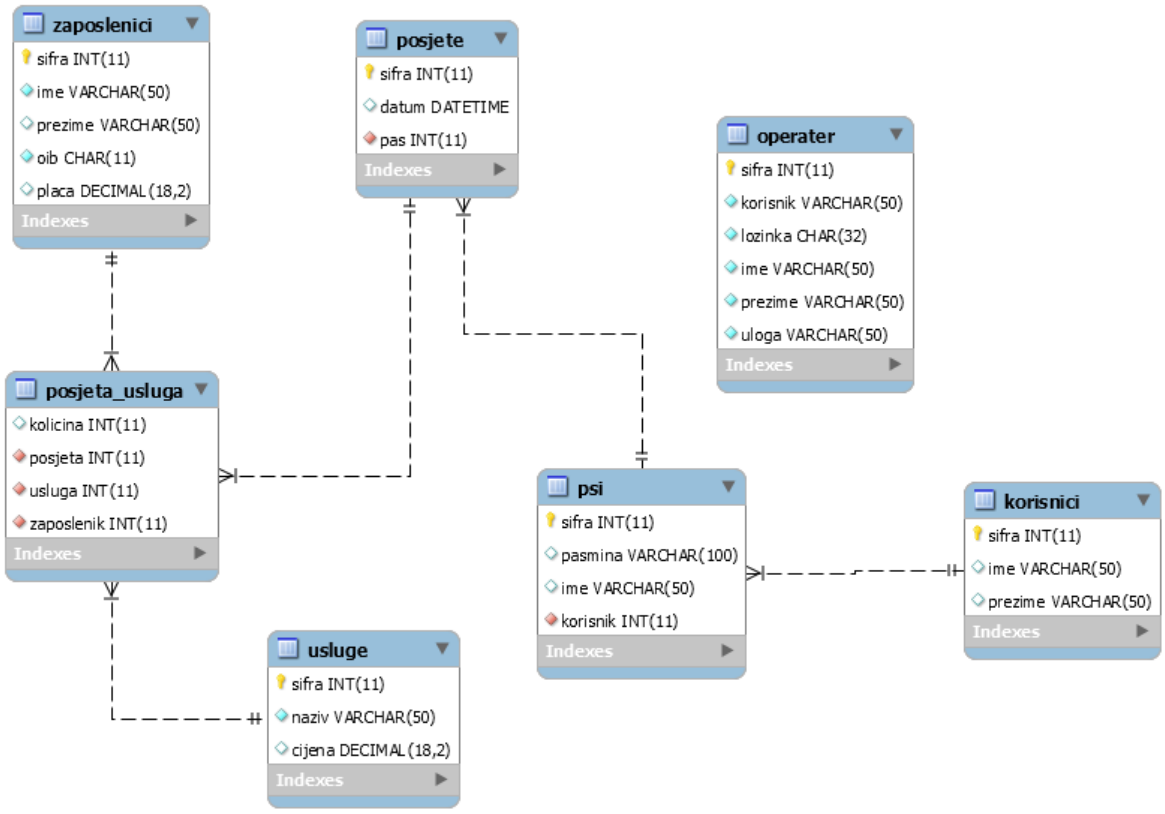
Baza podataka koja podržava ovu aplikaciju je relacijska baza podataka pisana SQL[2] jezikom. SQL je računalni jezik za izradu, pretraživanje, ažuriranje i brisanje podataka unutar relacijskih baza te je stardardiziran preko ANSI i ISO standarda. Naredbe SQL-a omogućavaju: pretragu i

[1] IDE – Integrated Development Environment
[2] SQL – Structured Query Language

grupiranje podataka, manipulaciju podacima, kontrolu transakcije, definiranje podataka, kontrola podataka i slično. [7]

2.2.1 ER dijagram

ER dijagram (engl. *Entity Relationship diagram*) je grafički prikaz informacija o odnosima između objekata u tablicama. Na slici 2.1. prikazan je dijagram relacijske baze podataka korištene u aplikaciji. [8]



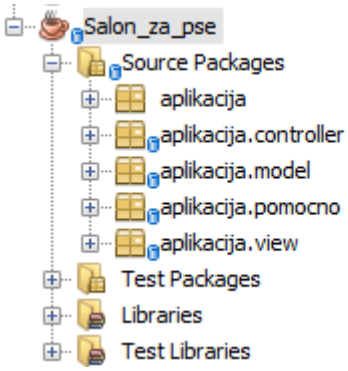
Sl.2.1. *ER dijagram relacijske baze podataka*

Sastoji se od sedam tablica, od kojih tablice *korisnici*, *zaposlenici* te *usluge* nemaju niti jedan strani ključ. Tablica *posjeta_usluga* je međutablica koja olakšava izvedbu veze više na više (engl. *many to many*) u relacijskoj bazi te osim podatka o količini sadrži strane ključeve tablica *usluge*, *zaposlenici* te *posjete*. Svaka tablica definirana je jedinstvenim identifikatorom[3] tj. cjelobrojnom vrijednošću kojom je moguće razlikovati podatke. Tablica *operater* dodana je naknadno kako bi omogućila postavljanje korisničkog imena i zaporke te uloge pojedinog zaposlenika.

[3] Engl. identifier (ID)

3. STRUKTURA APLIKACIJE

Aplikacija „Salon za pse“ zasnovana je na spomenutoj relacijskoj bazi podataka. Struktura aplikacije podijeljena je prema MVC obrascu tako da svaka komponenta čini zasebni paket (engl. *package*) u projektu.



Sl.3.1. *MVC paketi aplikacije*

Na slici 3.1. prikazan je svaki od paketa. Paket *aplikacija* sadrži Java klasu (engl. *class*) *Salon_za_pse.java* koja sadrži *main* metodu te pokreće aplikaciju. Paket *aplikacija.controller* sastoji se od upravitelja (engl. *controller*) za svaku pojedinu tablicu iz baze, točnije za svaki model. Analogno navedenom, *aplikacija.model* te *aplikacija.view* sadrže modele i poglede za pojedine entitete.

3.1 PAKET MODELA

Paket modela sastoji se od devet klasa. Prva je *Entitet.java* koja sadrži varijablu jedinstvenog identifikatora (engl. *ID*) te metode za dohvaćanje i postavljanje iste. *Entitet* klasu nasljeđuju preostale klase te će tako svaka od njih imati svoj jedinstveni identifikator. Slijede modeli za svaku tablicu iz baze u koje će podaci biti pohranjeni te ćemo im se moći pristupati preko metoda za dohvaćanje (engl. *getter*), odnosno metoda za postavljanje (engl. *setter*).

Primjerice klasa *Zaposlenici.java* sadrži slijedeće parametre:

- *private String ime*;
- *private String prezime*;
- *private String OIB*;
- *private BigDecimal placa*;

Klasa *Zaposlenici* također nasljeđuje klasu *Entitet.java* što implicira postojanje još jednog parametra – jedinstvenog identifikatora. Svaki od tih parametara može se dohvaćati i postavljati. Analogno ovom modelu napravljeni su i modeli za tablice *Operater* i *Korisnici*, ovisno o parametrima koji ih opisuju.

Preostali modeli razlikuju se zbog stranog ključa kojim su tablice međusobno povezane. Primjerice klasa *Psi.java* posjeduje strani ključ na tablicu *Korisnici* čime je definirano da pas ima vlasnika. Zbog toga je najprije definirana klasa *Korisnici.java* koja posjeduje parametre o korisniku te nakon toga klasa *Psi.java*. Parametri psa tada izgledaju ovako:

- *private String pasmina*;
- *private String ime*;
- *private Korisnici vlasnik*;
- *private Posjete posjeta*;

Parametar *vlasnik* instanca je klase *Korisnici.java* te omogućava pohranu podataka o vlasniku pojedinog psa. Analogno je i za parametar *posjeta*.

3.2 PAKET UPRAVITELJA

Upravitelj (engl. *controller*) prima ulazne vrijednosti i priprema ih za modele. Stoga za svaki model, postoji i upravitelj. Za klasu *Zaposlenici.java* postoji upravitelj klasa naziva *obradaZaposlenika.java* u paketu *aplikacija.controller* koja komunicira s relacijskom bazom podataka. Sadrži četiri metode koje ovisno o navedenom SQL upitu vraćaju određeni rezultat. Moguće je kreirati novog zaposlenika, ažurirati ili obrisati postojećeg te dohvatiti listu svih zaposlenika u bazi sa njihovim podacima. Za modele poput *Psi.java* potrebno je samo promijeniti SQL upit, točnije napraviti povezivanje tablica pomoću naredbe *inner join*.

3.3 PAKET POGLEDA

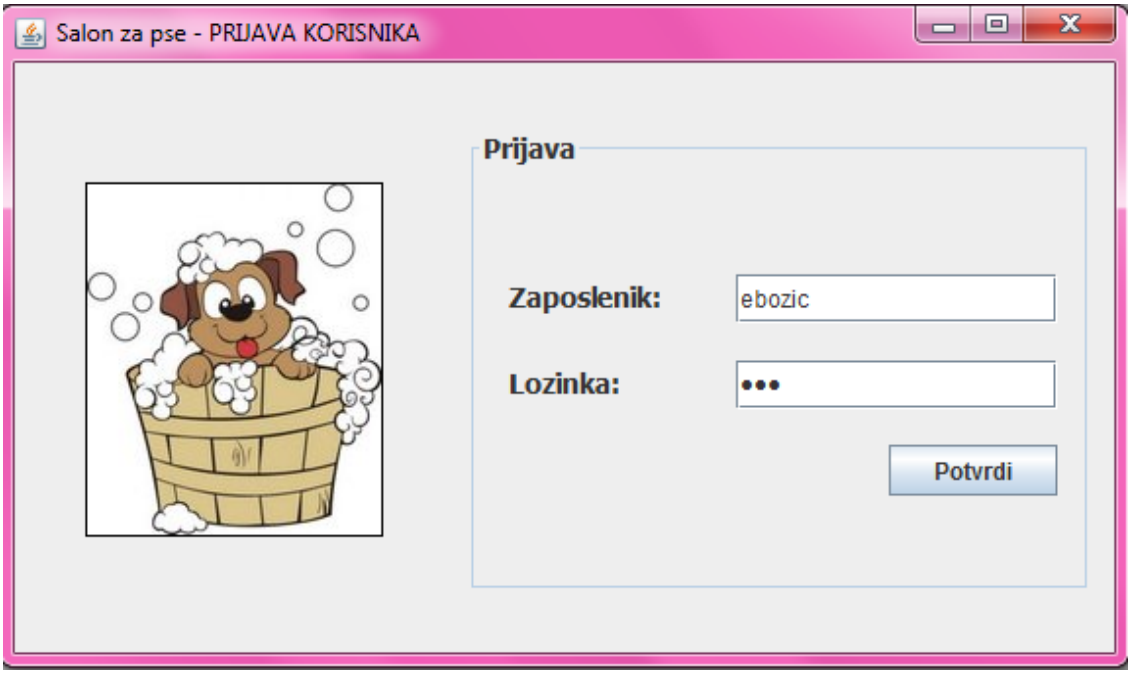
U aplikaciji ne mora nužno biti onoliko pogleda koliko je modela. „Salon za pse“ ima šest pogleda: *prijava korisnika, izbornik, prozor zaposlenici, prozor psi* (obuhvaća i prikaz korisnika), *prozor usluge* te *prozor posjeta* (koji je ujedno kreiranje računa).

4. IZGLED APLIKACIJE

Aplikacija se sastoji od šest Java okvira (engl. *jframe*). Nakon prijave zaposlenika omogućeno je u izborniku odabrati opciju *zaposlenici*, *psi*, *nova posjeta*, *statistike* ili *usluge*. Svaka od njih nudi određene mogućnosti.

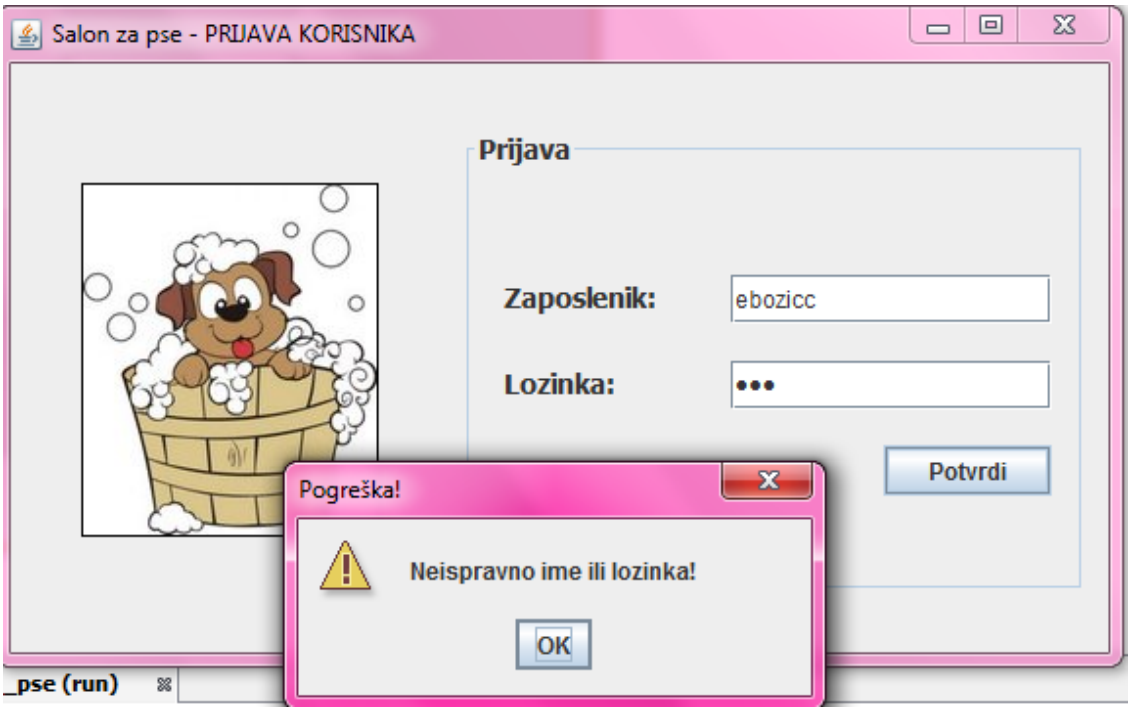
4.1 PRIJAVA KORISNIKA

Prijava korisnika prvi je prozor koji se otvara pri pokretanju aplikacije te je prikazan slici 4.1. Potrebno je upisati ime operatera te zaporku. Navedeni parametri definirani su u bazi podataka. Klikom na gumb „Potvrdi“, ukoliko su uneseni podaci točni, otvara se izbornik.



Sl.4.1. *Prijava korisnika*

Na slici 4.2. prikazan je primjer kada operater unese krivo korisničko ime ili zaporku. U tom slučaju pritisak na gumb „Potvrdi“ aktivirati će prozor za grešku. Klikom na „OK“ moguće je ponoviti unos.



Sl.4.2. *Netočna prijava korisnika*

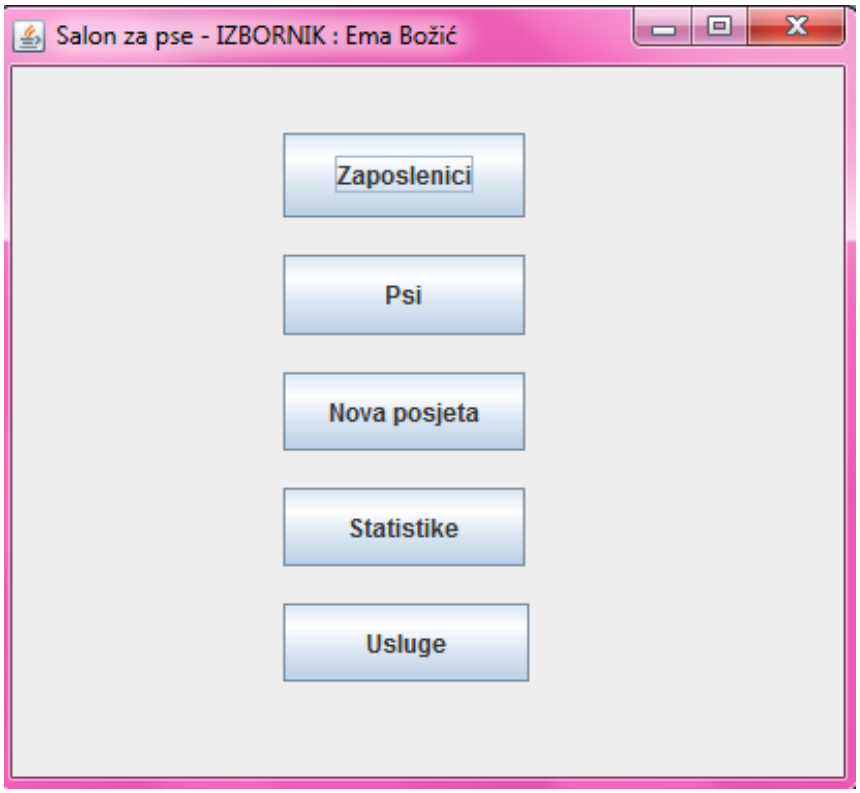
Upravitelj za prijavu operatera prikazan je na slici 4.3.

```
public Operater getZaposlenik(String korisnik, char[] lozinka){
    Operater z = null;
    try {


        veza=Baza.dohvatiVezu();
        izraz = veza.prepareStatement("select * from operater where korisnik=? and lozinka=md5(?)");
        izraz.setString(1, korisnik);
        izraz.setString(2, new String(lozinka));
        rs = izraz.executeQuery();


        while(rs.next ()){
            z = new Operater();
            z.setSifra(rs.getInt("sifra"));
            z.setLozinka(rs.getString("lozinka"));
            z.setKorisnik(rs.getString("korisnik"));
            z.setIme(rs.getString("ime"));
            z.setPrezime(rs.getString("prezime"));
            z.setUloga(rs.getString("uloga"));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return z;
}
```

Sl.4.3. *Upravitelj za prijavu*

4.2 IZBORNIK

Kada se zaposlenik prijavi otvara se izbornik prikazan na slici 4.4. Na alatnoj traci prikazano je ime i prezime prijavljenog operatera.



Sl.4.4. *Izbornik*

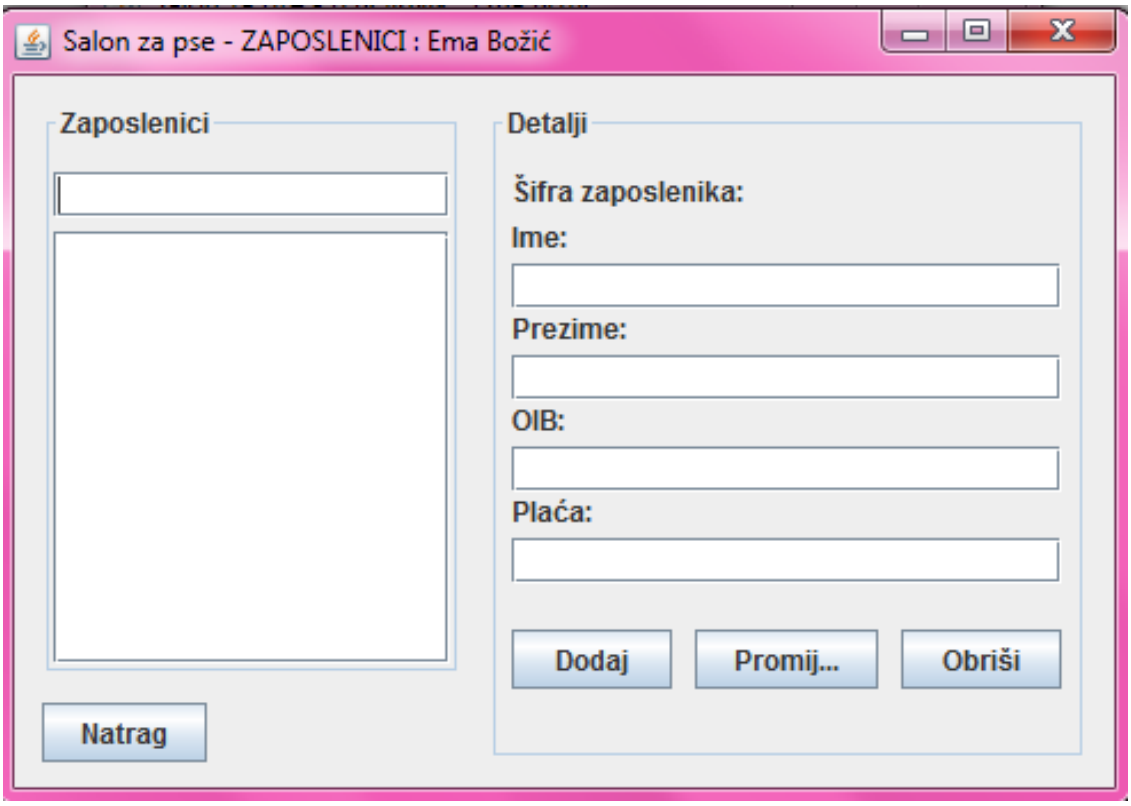
Pritiskom na željeni gumb otvara se pripadajući prozor ovisno kako je definirano u pogledu izbornika, slika 4.5. prikazuje da će se odabirom gumba „Zaposlenici“ otvoriti prozor zaposlenici.

```
private void btnZaposleniciActionPerformed(java.awt.event.ActionEvent evt) {
    new prozorZaposlenici(operater).setVisible(true);
}
```

Sl.4.5. *Pogled izbornik*

4.3 PROZOR ZAPOSLENICI

Klikom na gumb „Zaposlenici“ u izborniku otvara se prozor na slici 4.6.

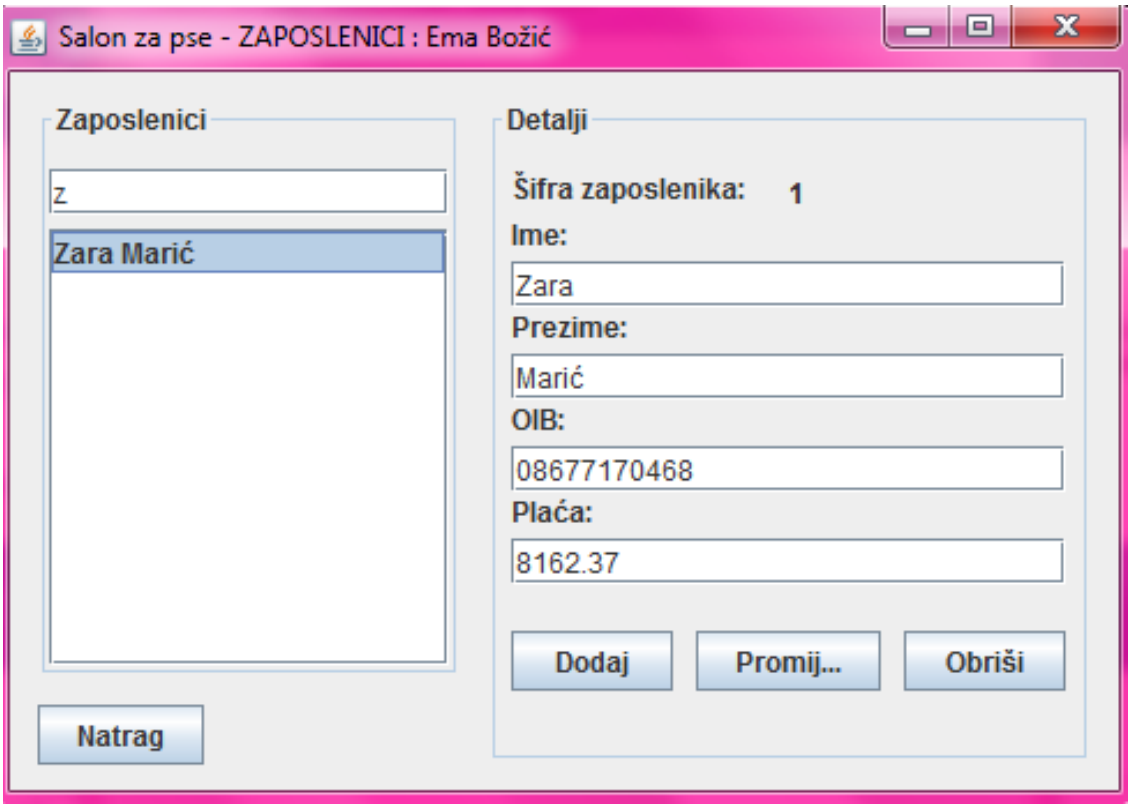


Sl.4.6. *Prozor zaposlenici*

Na lijevoj strani prozora omogućeno je pretraživanje zaposlenika u bazi podataka unosom imena ili prezimena u polje za pretragu. Odabirom željenog zaposlenika na desnoj strani prozora prikazuju se dodatne informacije (slika 4.7.).

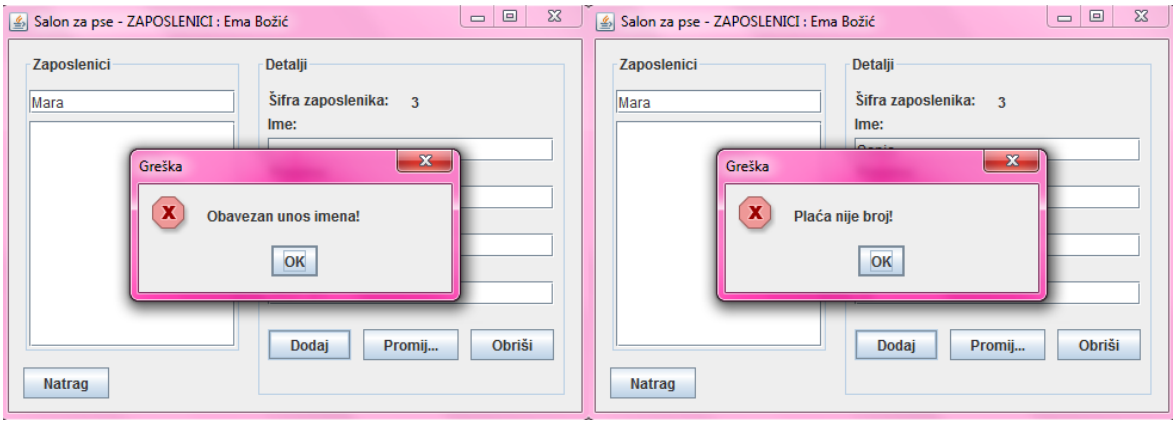
```
private void lstPopisValueChanged(javax.swing.event.ListSelectionEvent evt) {
    if(evt.getValueIsAdjusting()){
        return;
    }
    z = (Zaposlenici) lstPopis.getSelectedValue();

    if(z==null)
    {
        return;
    }

    lblSifra.setText(String.valueOf(z.getSifra()));
    txtIme.setText(z.getIme());
    txtPrezime.setText(z.getPrezime());
    txtOib.setText(z.getOib());
    txtPlaca.setText(String.valueOf(z.getPlaca()));
}
```

Sl.4.7. *Prikaz detalja*



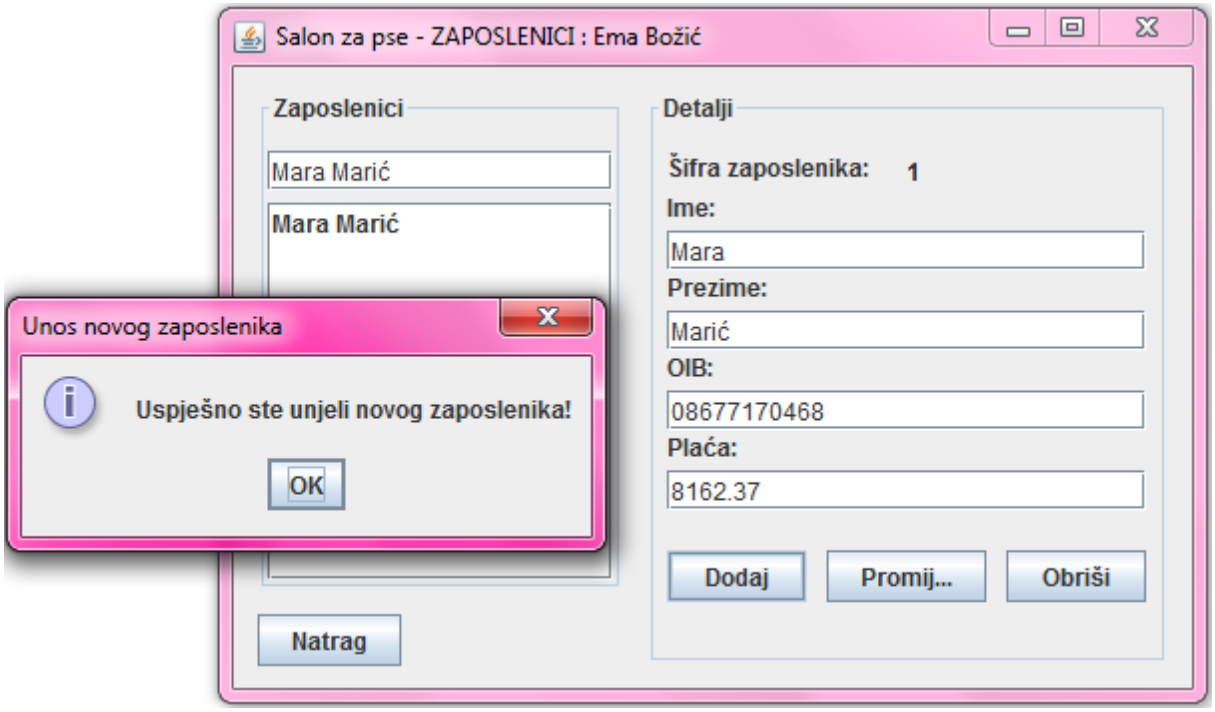
Sl.4.8. *Prikaz detalja odabranog zaposlenika*

Na slici 4.8. odabrana je zaposlenica Zara Marić te su prikazani njezini podaci. Tri gumba na dnu istog omogućavaju dodavanje novog zaposlenika, promjenu postojećih podataka i brisanje.



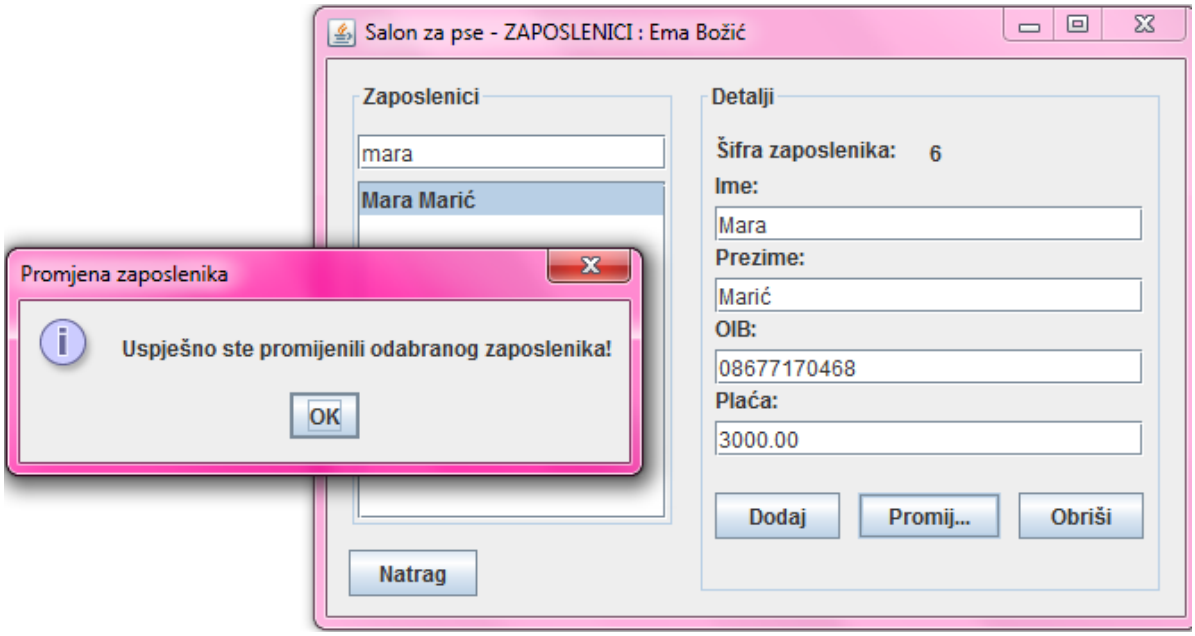
Sl.4.9. *Prikaz kontrole unosa*

Također su definirane kontrole kako bi se osiguralo popunjavanje svih podataka u onom formatu u kojem je potrebno da bi bili uneseni u bazu (slika 4.9.).



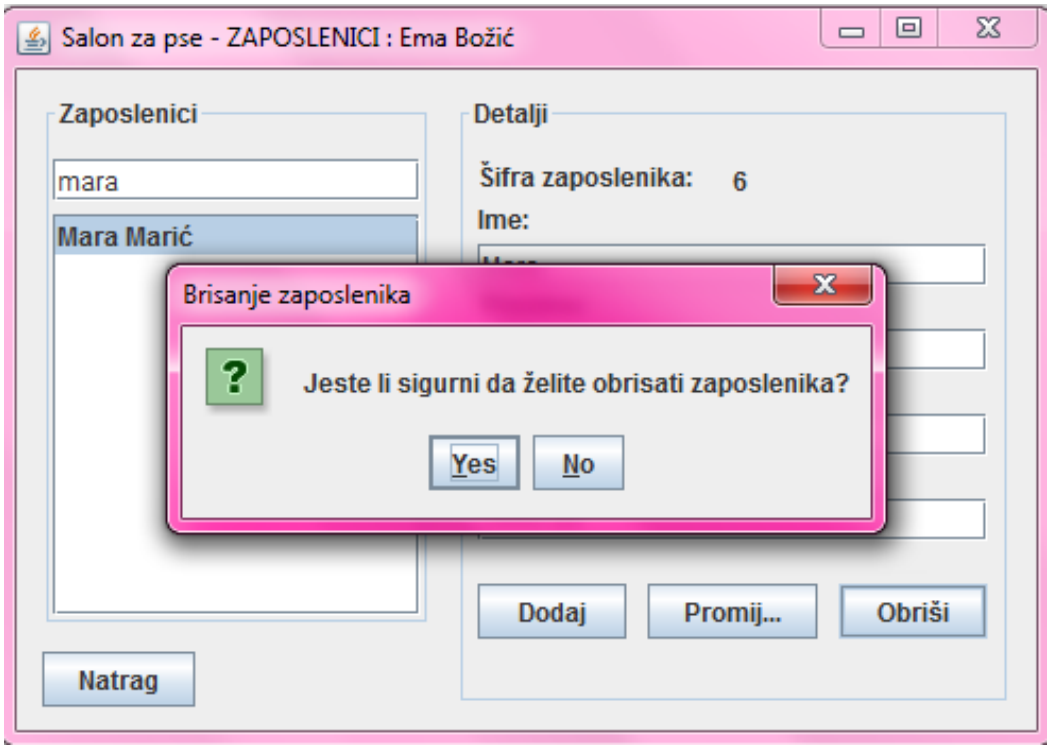
Sl.4.10. *Dodavanje novog zaposlenika*

Slika 4.10. prikazuje dodavanje novog zaposlenika u bazu. Potrebno je popuniti sve tražene podatke te klikom na gumb „Dodaj“ spremiti novog zaposlenika u bazu.

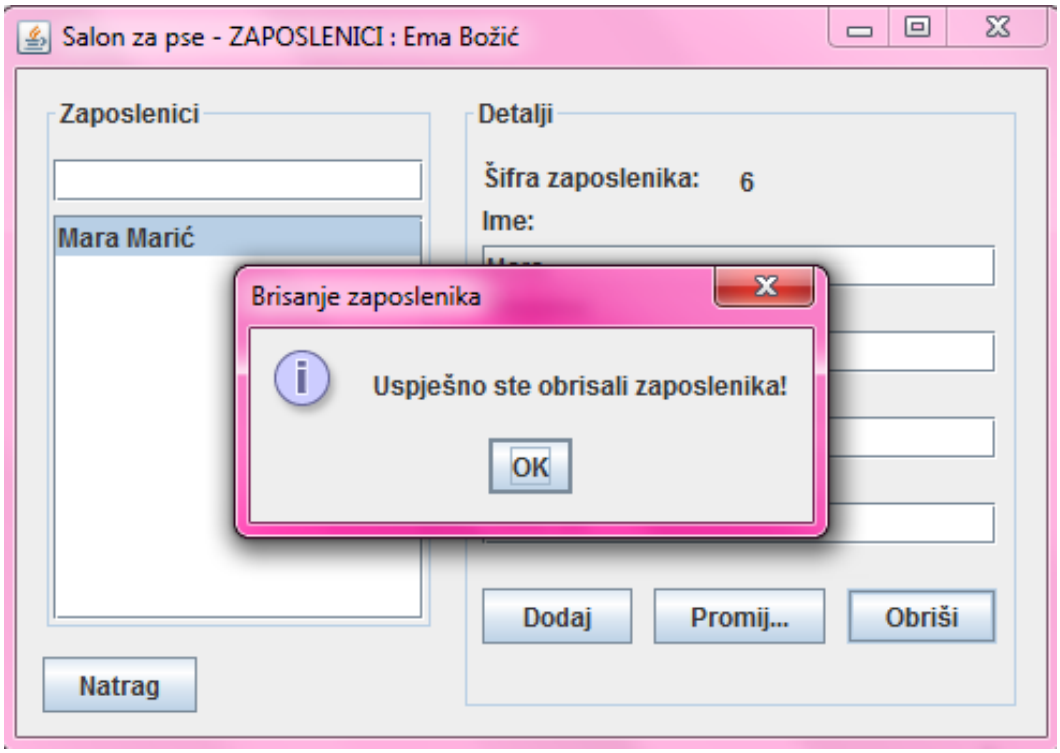


Sl.4.11. *Ažuriranje podataka zaposlenika*

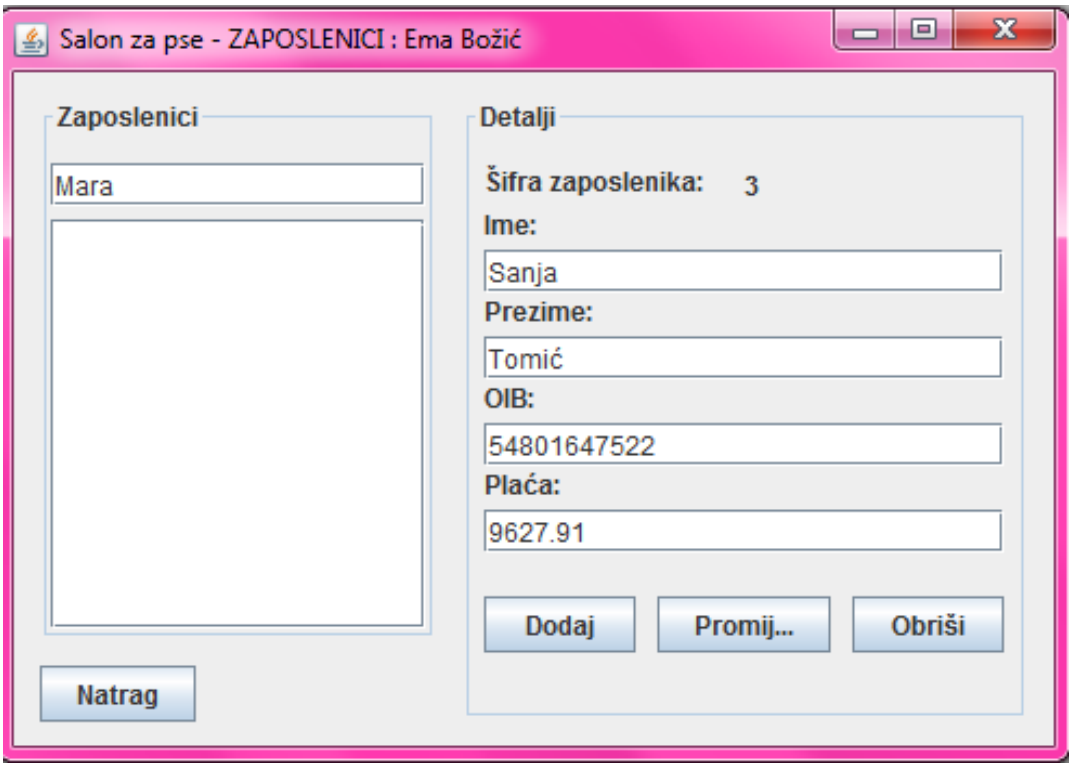
Operater također može ažurirati informacije o zaposlenicima, psima, korisnicima i ostalim entitetima kako je prikazano na slici 4.11. U ovom slučaju promijenjena je vrijednost plaće.



Sl.4.12. *Brisanje zaposlenika*



Sl.4.13. *Brisanje zaposlenika*



Sl.4.14. *Brisanje zaposlenika*

Brisanje zaposlenika ostvaruje se kroz par koraka. Nakon odabira zaposlenika kojeg je potrebno obrisati te klika na gumb „Brisanje“ najprije se javlja prozor sa porukom kako je prikazano na slici 4.12. Klikom na „Yes“ zaposlenik je obrisan (slika 4.13.). Slika 4.14. je prikaz pokušaja traženja obrisanog zaposlenika, no vidljivo je kako nije pronađen. Klikom na gumb „Natrag“ ponovno se otvara izbornik.

```
public int createZaposlenik(Zaposlenici z){

    int vrati=0;
    try{
        veza=Baza.dohvatiVezu();
        izraz=veza.prepareStatement
    ("insert into zaposlenici values (null, ?, ?, ?, ?)", Statement.RETURN_GENERATED_KEYS);
        izraz.setString(1,z.getIme());
        izraz.setString(2,z.getPrezime());
        izraz.setString(3,z.getOib());
        izraz.setBigDecimal(4, z.getPlaca());
        izraz.executeUpdate();

        rs=izraz.getGeneratedKeys();
        rs.next();

        vrati=rs.getInt(1);


        } catch (Exception e) {
        e.printStackTrace();
    }

    return vrati;
}
```

Sl.4.15. *Metoda za kreiranje zaposlenika*

Svaka od tri opcije omogućena je u upravitelju (engl. *controller*) odgovarajućim SQL upitom. Primjer za kreiranje zaposlenika prikazan je na slici 4.15.

```
public int updateZaposlenik (Zaposlenici z){
    int vrati=0;
    try {
        veza=Baza.dohvatiVezu();
        izraz = veza.prepareStatement
        (" update zaposlenici set " + " ime=?, prezime=?, " + " oib=?, placa=? where sifra=? ");
        izraz.setString(1, z.getIme());
        izraz.setString(2, z.getPrezime());
        izraz.setString(3, z.getOib());
        izraz.setBigDecimal(4, z.getPlaca());
        izraz.setInt(5, z.getSifra());

        vrati=izraz.executeUpdate();

    } catch (Exception e) {
    }

    return vrati;
}
```

Sl.4.16. *Metoda za ažuriranje zaposlenika*

Ažuriranje podataka o zaposleniku omogućeno je SQL naredbom „UPDATE“ preko jedinstvenog identifikatora (slika 4.16.).

```
public int deleteZaposlenik (Zaposlenici z){
    int vrati=0;
    try {
        veza=Baza.dohvatiVezu();
        izraz = veza.prepareStatement(" delete from zaposlenici " + " where sifra=? ");
        izraz.setInt(1, z.getSifra());
        vrati=izraz.executeUpdate();

    } catch (Exception e) {
        return -1;
    }


    return vrati;
}
```

Sl.4.17. *Metoda za brisanje zaposlenika*

Metoda za brisanje zaposlenika prikazana je na slici 4.17. gdje se koristi SQL naredba „DELETE“ te jedinstveni identifikator kako bi definirali o kojem se zaposleniku radi.

```
public List<Zaposlenici> getZaposlenici (String uvjet){
      List<Zaposlenici> l = new ArrayList<>();
    try {
        veza=Baza.dohvatiVezu();

        izraz = veza.prepareStatement("select * from zaposlenici where concat (ime,' ', prezime) like ?");
        izraz.setString(1, "%"  + uvjet + "%");
        rs = izraz.executeQuery();

        Zaposlenici z;
        while(rs.next())
        {
            z = new Zaposlenici();
            z.setSifra(rs.getInt("sifra"));
            z.setIme(rs.getString("ime"));
            z.setPrezime(rs.getString("prezime"));
            z.setOib(rs.getString("oib"));
            z.setPlaca(rs.getBigDecimal("placa"));
            l.add(z);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
return l;
}
}
```

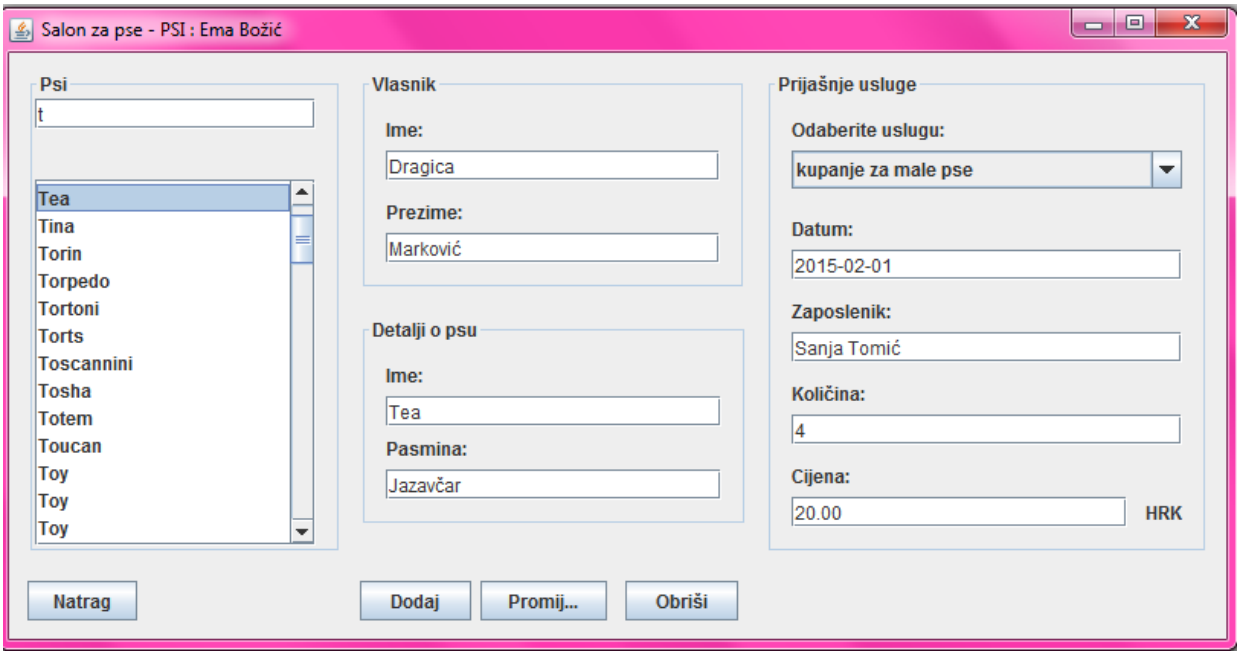
Sl.4.17. *Metoda za dohvaćanje zaposlenika*

Rad sa podacima zaposlenika omogućen je metodom *getZaposlenik* koja kao rezultat vraća listu tipa *Zaposlenici*.

Analogno je napravljeno pregledavanje, kreiranje, ažuriranje i brisanje za tablicu *Usluge*, sa odgovarajućim vrijednostima.

4.4 PROZOR PSI

Klikom na gumb „Psi“ u izborniku otvara se prozor koji omogućava pregled svi pasa u bazi podataka salona, njihovih vlasnika te usluga koje su obavili.



Sl.4.18. *Prozor psi*

Na lijevoj strani, kao i kod prozora zaposlenika, postavljena je tražilica kojom je moguće odabrati željenog psa. Odabirom istog dobiveni su podaci o vlasniku, o samom psu te o prijašnjim uslugama koje su obavljene nad tim psom. Gumbi za dodavanje, promjenu i brisanje rade analogno kao i kod zaposlenika. Za dodavanje novih usluga ili promjenu postojeće u izborniku odabrati „Usluge“.

Tablica psi ima dva vanjska ključa, *posjete* i *korisnik*, te dohvaćanje i brisanje podataka u upravitelju izgleda nešto drugačije kako je prikazano na slikama 4.19. i 4.20.

```
public List<Psi> getPsi (String uvjet){
        List<Psi> l = new ArrayList<>();
        try {
            veza=Baza.dohvatiVezu();
            izraz = veza.prepareStatement(" select b.sifra as sifravlasnika, b.ime as imevlasnika, "
                    + " b.prezime as prezimevlasnika, "
                    + " a.ime as imepsa, a.pasmina, a.sifra as sifrapsa" + " from psi a "
                    + " inner join korisnici b on a.korisnik=b.sifra "
                    + " where a.ime like ? order by a.ime ");
            izraz.setString(1, "%" + uvjet + "%");
            rs = izraz.executeQuery();
            Psi p;
            Korisnici k;
            while(rs.next())
            {
                p = new Psi();
                p.setSifra(rs.getInt("sifrapsa"));
                p.setIme(rs.getString("imepsa"));
                p.setPasmina(rs.getString("pasmina"));
                k= new Korisnici();
                k.setSifra(rs.getInt("sifravlasnika"));
                k.setIme(rs.getString("imevlasnika"));
                k.setPrezime(rs.getString("prezimevlasnika"));
                p.setVlasnik(k);
                l.add(p);       }
      } catch (Exception e) {e.printStackTrace();}
        return l;}
```

Sl.4.19. *Metoda za dohvaćanje podataka o psima*

```
public int deletePsi(Psi p){
    int vrati = 0;
    try {
        veza.setAutoCommit(false);
        izraz = veza.prepareStatement("delete from posjete where pas=?");
        izraz.setInt(1, p.getSifra());
        izraz.executeUpdate();
        izraz = veza.prepareStatement("delete from psi where sifra=?");
                izraz.setInt(1, p.getSifra());
        vrati = izraz.executeUpdate();
        veza.commit();
    } catch (SQLException es) {
      return -1;
    }

    return vrati;

}
```

Sl.4.20. *Metoda za brisanje podataka o psima*

4.5 PROZOR STATISTIKE

Prozor statistike otvara tortni grafikon (engl. *pie chart*) koji daje uvid u najčešće korištene usluge u salonu (slika 4.21.).

Sl.4.21. *Tortni grafikon usluga*

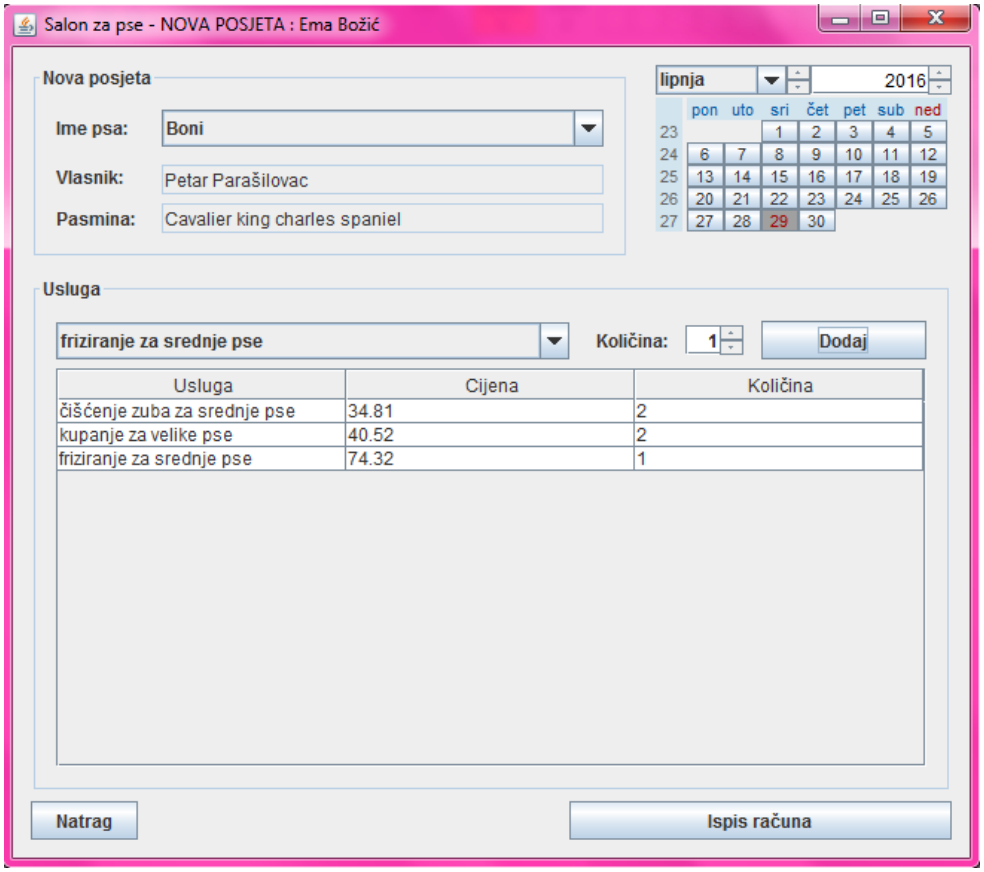
Tortni grafikon napravljen je uz pomoć biblioteke (engl. *library*) *jfreechart*. Različite boje dijelova tortnog grafikona prikazuju zastupljenost različitih usluga. Imena usluga navedena su pokraj grafikona. Postavke grafikona prikazane su na slici 4.22.

```
private void btnStatistikeActionPerformed(java.awt.event.ActionEvent evt) {
    DefaultPieDataset pieDataset = new DefaultPieDataset();

    pieDataset.setValue("Kupanje za male pse", ou.getKolicinaUsluge1());
    pieDataset.setValue("Kupanje za srednje pse", ou.getKolicinaUsluge2());
    pieDataset.setValue("Kupanje za velike pse", ou.getKolicinaUsluge3());
    pieDataset.setValue("Čišćenje zuba za male pse", ou.getKolicinaUsluge4());
    pieDataset.setValue("Čišćenje zuba za srednje pse", ou.getKolicinaUsluge5());
    pieDataset.setValue("Čišćenje zuba za velike pse", ou.getKolicinaUsluge6());
    pieDataset.setValue("Friziranje za male pse", ou.getKolicinaUsluge7());
    pieDataset.setValue("Friziranje za srednje pse", ou.getKolicinaUsluge8());
    pieDataset.setValue("Friziranje za velike pse", ou.getKolicinaUsluge9());
    JFreeChart chart = ChartFactory.createPieChart("Količina pojedinih usluga", pieDataset, true, true, true);
    PiePlot P = (PiePlot) chart.getPlot();
    ChartFrame frame = new ChartFrame("Tortni grafikon", chart);
    frame.setVisible(true);
    frame.setBackground(Color.WHITE);
    frame.setSize(450, 500);
    P.setBackgroundPaint(Color.WHITE);
}
```

Sl.4.22. *Postavke tortnog grafikona*

4.6 PROZOR NOVA POSJETA

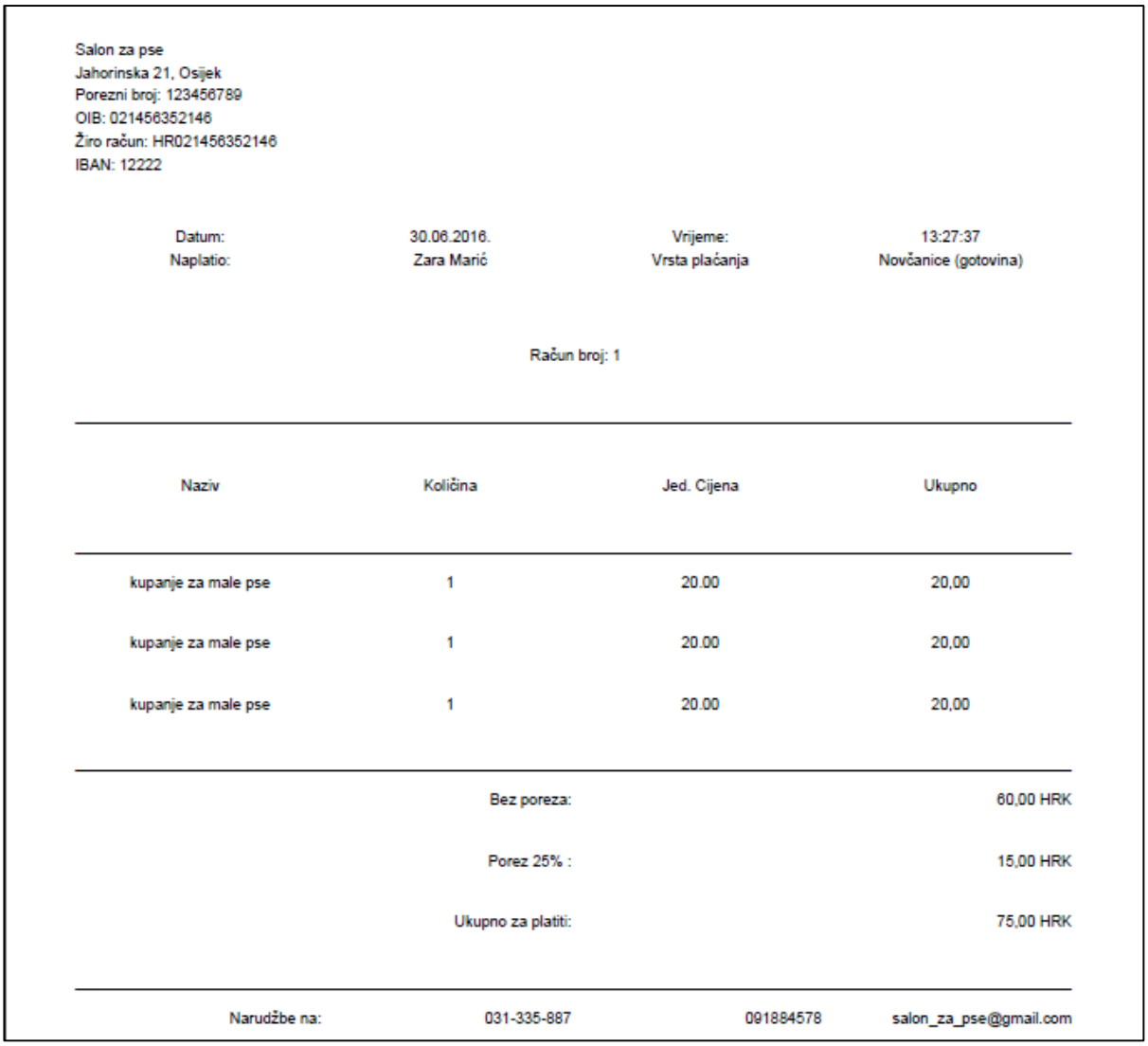
Klikom na gumb „Nova posjeta“ otvara se prozor za kreiranje računa prikazan na slici 4.23. Omogućen je odabir psa te dodavanje obavljenih usluga u tablicu.



Sl.4.23. *Kreiranje računa*

Padajući izbornik sadrži popis svih pasa u bazi te omogućava odabir željenog. Nakon odabira popune se podaci o vlasniku i pasmini psa. U desnom gornjem kutu nalazi se kalendar koji je kreiran koristeći biblioteku *jcalendar* te je postavljen na trenutni datum. U panelu *Usluga* nalazi se padajući izbornik sa popisom svih usluga, kućica za odabir količine te gumb „Dodaj“. Klikom na isti popunjava se tablica pomoću koje će se kreirati račun. Tablica sadrži podatke o nazivu usluge, cijeni te odabranoj količini. Gumb „Ispis računa“ omogućava prikaz kreiranog računa u PDF[4] formatu.

[4] PDF - Portable Document Format



Salon za pse
Jahorinska 21, Osijek
Porezni broj: 123456789
OIB: 021456352146
Žiro račun: HR021456352146
IBAN: 12222

Datum:	30.06.2016.	Vrijeme:	13:27:37
Naplatio:	Zara Marić	Vrsta plaćanja	Novčanice (gotovina)

Račun broj: 1

Naziv	Količina	Jed. Cijena	Ukupno
kupanje za male pse	1	20.00	20,00
kupanje za male pse	1	20.00	20,00
kupanje za male pse	1	20.00	20,00

Bez poreza:	60,00 HRK
Porez 25% :	15,00 HRK
Ukupno za platiti:	75,00 HRK

Narudžbe na:	031-335-887	091884578	salon_za_pse@gmail.com

Sl.4.24. *Račun u PDF formatu*

PDF dokument uzima u obzir odabranog blagajnika, odabrane usluge i količine na računu te preko metoda *getCijenaUkupna*, *getPorez* te *getCijenaPorez* računa potrebne vrijednosti. Za generiranje PDF-a korištena je biblioteka *itextpdf*. PDF račun sprema se lokalno na računalu.

4.7 KONTROLA UNOSA

Kontrola unosa odrađena je kroz nekoliko provjera kako bi se osiguralo da će unos u bazu biti uspješan. Potrebno je popuniti sva tražena polja te unijeti pravilne vrijednosti.

```
if(txtIme.getText().trim().length()==0){
        JOptionPane.showMessageDialog(
                getRootPane(), //prozor koji ga zove
                "Obavezan unos imena!", //prikazani tekst
                "Greška",//naslov
                JOptionPane.ERROR_MESSAGE //vrsta poruke
        );
        txtIme.requestFocus();
        return false;
    }
        z.setIme(txtIme.getText());
```

Sl.4.25. *Kontrola unesenog imena*

Na slici 4.25. prikazan je dio programa koji provjerava pravilan unos imena, ukoliko nije uneseno pojavljuje se prozor „Greška“ sa porukom „Obavezan unos imena!“.

```
        if(txtPrezime.getText().trim().length()==0){
        JOptionPane.showMessageDialog(
                getRootPane(), //prozor koji ga zove
                "Obavezan unos prezimena!", //prikazani tekst
                "Greška",//naslov
                JOptionPane.ERROR_MESSAGE //vrsta poruke
        );
        txtPrezime.requestFocus();
        return false;
        }
        z.setPrezime(txtPrezime.getText());
```

Sl.4.26. *Kontrola unesenog prezimena*

Kontrola prezimena odrađena je na sličan način kao i kontrola imena, ukoliko je unesena vrijednost jednaka nuli pojavljuje se prozor *Greška* (slika 4.26.).

```
    String regex = "\\d+";
    String data = txtOib.getText();

if (data.matches(regex)){
       z.setOib(txtOib.getText());}
else{
            JOptionPane.showMessageDialog(
            getRootPane(), //prozor koji ga zove
            "Oib nije broj!", //prikazani tekst
            "Greška",//naslov
            JOptionPane.ERROR_MESSAGE //vrsta poruke
            );
         txtOib.requestFocus();
         return false;
    }
```

Sl.4.27. *Kontrola unesenog OIB-a*

OIB se provjerava koristeći regularni izraz (engl. *regex*). Regularni izraz je izraz kojim se definira uzorak koji se koristi za pretraživanje teksta. Konkretno za OIB potrebno je dozvoliti isključivo unos brojeva, stoga je regularni izraz definiran kao *String* vrijednosti „\\d+“ kako je prikazano na slici 4.27. Ukoliko se unese nekakav drugi znak javlja se prozor *Greška* sa porukom „Oib nije broj!“. [9]

```
if(txtPlaca.getText().trim().length()>0){
    try {
            z.setPlaca(new BigDecimal(txtPlaca.getText()));
    } catch (Exception e) {
        JOptionPane.showMessageDialog(
        getRootPane(), //prozor koji ga zove
        "Plaća nije broj!", //prikazani tekst
        "Greška",//naslov
        JOptionPane.ERROR_MESSAGE //vrsta poruke
);
txtPlaca.requestFocus();
txtPlaca.selectAll();
return false;
    }
}
```

Sl.4.28. *Kontrola unesenog iznosa plaće*

Kontrola unosa plaće, ukoliko je duljina unesene vrijednosti veća od nule, pokušava postaviti tu vrijednost u bazu. Ukoliko ne uspije, izvršiti će se *catch* blok te ispisati poruka „Plaća nije broj!“ jer je vrijednost iste definirana kao *BigDecimal* čime je odmah osigurano da ne može primiti druge znakove osim brojeva. (slika 4.28.)

4.8 SPAJANJE NA BAZU

Kako bi kôd bio pregledniji spajanje na bazu izrađeno je u jednoj klasi koja se poziva preko metoda u upraviteljima.

```
public class Baza {
    private static Connection veza;
    private static Baza baza =null;
    protected Baza(){
        try {
            Class.forName("com.mysql.jdbc.Driver");
            veza = DriverManager.getConnection(
                    "jdbc:mysql://localhost/salon_za_pse?useUnicode=true&characterEncoding=utf-8",
                    "edunova",
                    "edunova");

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static Connection dohvatiVezu(){
        if(baza==null){
            baza = new Baza();
        }
        return veza;
    }
    public static void zatvoriVezu(){
         try {
             veza.close();
         } catch (SQLException ex) {
             Logger.getLogger(Baza.class.getName()).log(Level.SEVERE, null, ex);
         }}
```

Sl.4.29. *Spajanje na bazu*

Na slici 4.29. nalazi se spomenuta klasa koja definira spajanje na bazu pomoću biblioteke *mysql-connector-java*. Također su definirane i dvije metode, *dohvatiVezu* i *zatvoriVezu*, koje se pozivaju u upraviteljima ili gdje god je potrebno, kako je prikazano na slikama 4.30. te 4.31.

```
    private Connection veza;
    private PreparedStatement izraz;
    private ResultSet rs;
```

Sl.4.30. *Pozivanje baze u upravitelju*

```
            veza=Baza.dohvatiVezu();
```

Sl.4.31. *Pozivanje baze u upravitelju – otvaranje konekcije*

5. ZAKLJUČAK

Izrada Java desktop aplikacije, kao primjer blagajne salona za pse, tema je ovog diplomskog rada. Za potrebe rada izrađena je SQL relacijska baza podataka koja se sastoji od sedam međusobno povezanih tablica. Aplikacija je temeljena na navedenoj bazi te omogućava pregledavanje, ažuriranje, promjenu i brisanje svih podataka. Sve navedene funkcije prikazane su na zanimljiv način, neke od njih i kroz više pogleda (engl. view). Izrada aplikacije bazirana je na model-view-controller obrascu čime je olakšana preglednost projekta. Aplikacija nudi korisniku lako baratanje svim potrebnim podacima te izradu računa pri novoj posjeti. Račun se može spremiti kao PDF dokument.

LITERATURA

[1] Java – tehnološka platforma, URL: http://www.igea.hr/java (25. lipnja 2016.)

[2] Java, URL: http://www.algebra.hr/edukacija/java/ (25. lipnja 2016.)

[3] Uvod u Java programiranje, URL: http://www.linuxzasve.com/uvod-u-java-programiranje-prvi-dio (26. lipnja 2016.)

[4] NetBeansIDE, URL: https://netbeans.org/features/index.html (26. lipnja 2016.)

[5] Java Development Kit, URL: http://java.com/en/download/faq/develop.xml (26. lipnja 2016.)

[6] Model-View-Controlles, URL: http://www.tutorialspoint.com/design_pattern/mvc_pattern.htm (28. lipnja 2016.)

[7] SQL syntax, URL: http://www.w3schools.com/sql/sql_syntax.asp (28. lipnja 2016.)

[8] ER dijagrami, URL: http://www.vps.ns.ac.rs/Materijal/mat5457.pdf (28. lipnja 2016.)

[9] Regularni izrazi, URL: http://www.java.hr/node/181 (29. lipnja 2016.)

SAŽETAK

Diplomski rad teme „Programsko rješenje blagajne u Java-i“ na konkretnom primjeru prikazuje moguće rješenje za aplikaciju koja olakšava vođenje salona za uređivanje kućnih ljubimaca. Ista omogućava rukovanje podacima o zaposlenicima, korisnicima salona, uslugama. Također sadrži grafički prikaz statistika o najčešće korištenim uslugama. Kreiranje računa, kao glavna funkcionalnost, omogućena je popunjavanjem forme sa svim potrebnim podacima, od strane zaposlenika, na temelju kojih se, lokalno na računalu, kreira PDF dokument računa.

Ključne riječi: Java, desktop aplikacija, baza podataka, Swing

ABSTRACT

CASH REGISTER JAVA APPLICATION

The thesis covers solution for cashbox in Java on an exact example of management of a spa for pets. This solution allows handling data about employees, customers and saloon services. It also contains a graphical representation of statistics on the most frequently used ones. The main functionality of this application is creating a receipt with the necessary data which employees fill via entry form. Bill is later saved as a PDF document locally on the computer.

Keywords: Java, desktop application, database, Swing framework

ŽIVOTOPIS

Ema Božić rođena je 05. siječnja 1993. godine u Osijeku. Osnovnu školu „Ivana Filipovića“ završila je 2007. godine u Osijeku. Srednju školu, „III. gimnaziju“, pohađala je u Osijeku i završila 2011. godine. Po završetku srednje škole upisuje preddiplomski studij računarstva na Elektrotehničkom fakultetu u Osijeku koji uspješno završava 2014. godine. Trenutno završava drugu godinu diplomskog studija na istom fakultetu.

PRILOG CD

Na priloženom CD-u nalazi se diplomski rad u *docx* i *pdf* formatu te aplikacija.