Objektno orjentisano programiranje:

Programska rješenja u Javi

[Prilagođeno od]

Korištene tehnologije

- Programski jezik Java
  - Razvojno okruženje (NetBeans IDE, Eclipse)
  - Java Development Kit (okruženje za razvijanje i pokretanje Java aplikacija koje obuhvata i JRE - engl. Java Runtime Environment).
  - Neka razvojna okruženja već imaju ugrađene ove dodatke koji su neophodni kako bismo mogli pisati i pokretati Java aplikacije.
  - Model-View-Controller (MVC) je obrazac arhitekture softvera sa odvajanjem pojedinih dijelova aplikacije u komponente zavisno o njihovoj namjeni

Korištene tehnologije

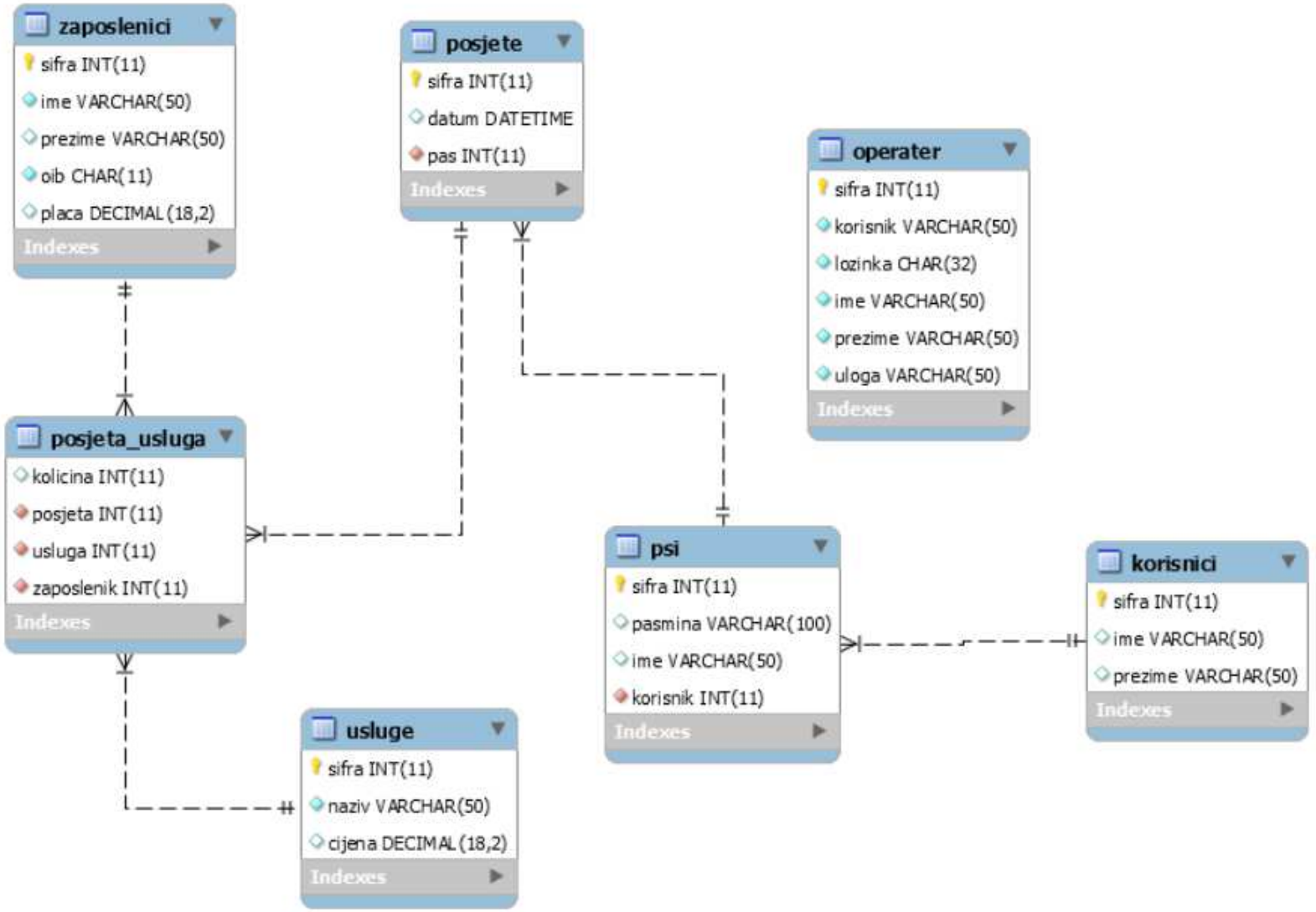
- **Model-View-Controller** (MVC) je obrazac arhitekture softvera sa odvajanjem pojedinih dijelova aplikacije u komponente zavisno o njihovoj namjeni:
  - **Model** (engl. model) sadrži poslovna pravila te funkcije ugrađe u poslovnu logiku (engl. business logic).
  - Prikaz podataka opisan je u komponenti **View**, a odnosi se na tablice, dijagrame, obrasce, itd. Prikaz podataka može biti ostvaren kroz više pogleda (engl. views).
  - Upravljač (engl. C**ontroller**) prima ulazne podatke (engl. input data) te ih pretvara u naloge modelu ili pogledu.
- Ovakvim pisanjem aplikacije olakšan je razvoj, preglednost, testiranje te održavanje aplikacije.

Korištene tehnologije

- Relaciona baza podataka
  - Baza podataka za aplikaciju – relaciona baza podataka pisana SQL jezikom (SQL je jezik za izradu, pretraživanje, ažuriranje i brisanje podataka unutar relacionih baza zasnovan na ANSI i ISO standardima). SQL naredbe omogućavaju: pretragu i grupisanje podataka, manipulaciju podacima, kontrolu transakcije, definisanje podataka, kontrola podataka i slično.
  - ER dijagram (engl. Entity Relationship diagram) je grafički prikaz informacija o odnosima između objekata u tabelama.

Korištene tehnologije

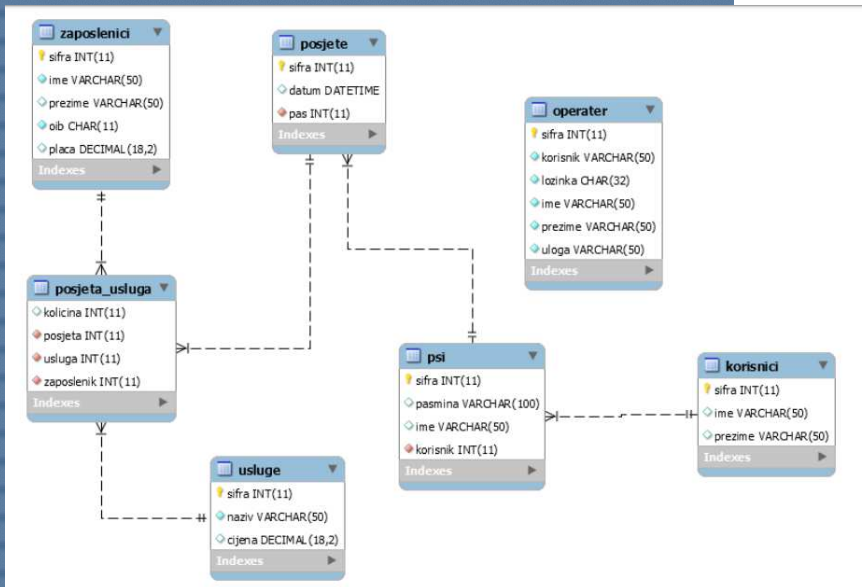
- ER dijagram



Sl.2.1. *ER dijagram relacijske baze podataka*

Korištene tehnologije

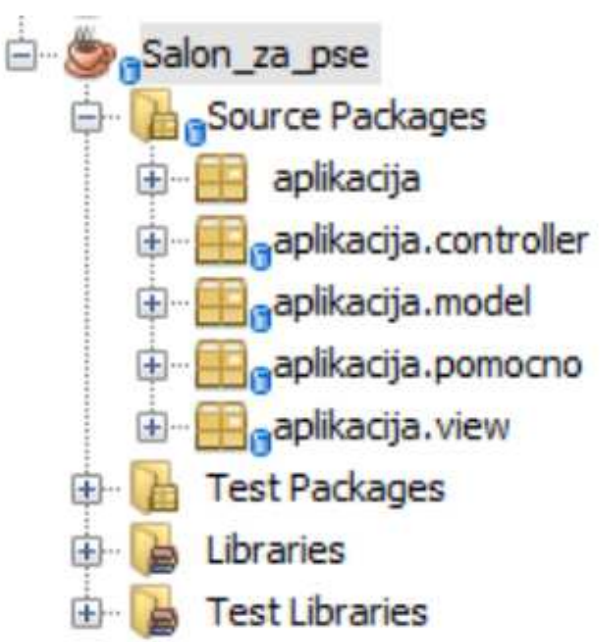
- BAZA se sastoji od sedam tabela (tabele korisnici, zaposlenici te usluge nemaju strani ključ).
- Tabela posjeta_usluga je međutablica koja olakšava izvedbu veze više na više (engl. Many to many) u relacijskoj bazi te
- Svaka tablica definirana je jedinstvenim identifikatorom ID (cjelobrojna vrijednost kojom je moguće razlikovati podatke.
- Tabela operater dodana je naknadno kako bi omogućila postavljanje korisničkog imena i lozinke te uloge pojedinog zaposlenika



Sl.2.1. *ER dijagram relacijske baze podataka*

Korištene tehnologije

- Struktura aplikacije (primjer)
  - Aplikacija zasnovana relacijskoj bazi podataka.
  - Struktura aplikacije podijeljena je prema MVC obrascu tako da svaka komponenta čini zasebni paket (engl. package) u projektu.
  - Paket aplikacija sadrži Java klasu (engl. class) *`Salon_za_pse.java`* koja sadrži `main` metodu te pokreće aplikaciju.
  - Paket `aplikacija.controller` sastoji se od upravljača (engl. controller) za svaku pojedinu tabelu iz baze, tačnije za svaki model.
  - `aplikacija.model` te `aplikacija.view` sadrže modele i poglede za pojedine entitete



Sl.3.1. *MVC paketi aplikacije*

Korištene tehnologije

- PAKET MODELA
  - Paket modela sastoji se (u primjeru) od devet klasa .
    - 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 tabelu iz baze, podacima će se moći pristupati preko metoda za dohvat (engl. getter) ili postavljanje (engl. setter).

Korištene tehnologije

- PAKET UPRAVLJAČA
  - Upravljač (engl. controller) prima ulazne vrijednosti i priprema ih za modele. Za svaki model, postoji i upravljač.
  - Za klasu `Zaposlenici.java` postoji upravljač klasa pod imenom `obradaZaposlenika.java` u paketu `aplikacija.controller` koja komunicira sa bazom.
  - Sadrži četiri metode koje u zavisnosti od navedenog SQL upita vraćaju određeni rezultat (kreiranje novog zaposlenika, ažuriranje ili brisanje postojećeg te listu svih zaposlenih).

Korištene tehnologije

- PAKET POGLEDA
- U aplikaciji ne mora biti onoliko pogleda koliko je modela.

Korištene tehnologije

- Konekcija na bazu:

```
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);
         }}
```

Korištene tehnologije

- SQL UPITI
- Autentikacija/prijava – najčešće SQL komanda ima oblik (upravljač):

```
select * from operater where ime_korisnika = ? and lozinka = ?
```

voditi računa u *SQL injection* napadu!

```
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;
```

Korištene tehnologije

- SQL UPITI
- Kreiranje korisnika:

```
insert into zaposlenici values (?, ?, ?, ?)
```

```
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;
}
```

Korištene tehnologije

- SQL UPITI
- Izmjena podataka o korisniku:

```
update zaposlenici set ime=? Where ID= ?
```

```
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;
}
```

Korištene tehnologije

- SQL UPITI
- Brisanje korisnika:

```
delete from zaposlenici where ID= ?
```

```
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;
}
```

Korištene tehnologije

- SQL UPITI
- Dohvati podatke o korisniku:

```
select * from operater where ime_korisnika like ? and prezime like ?
```

```
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;
```

Korištene tehnologije

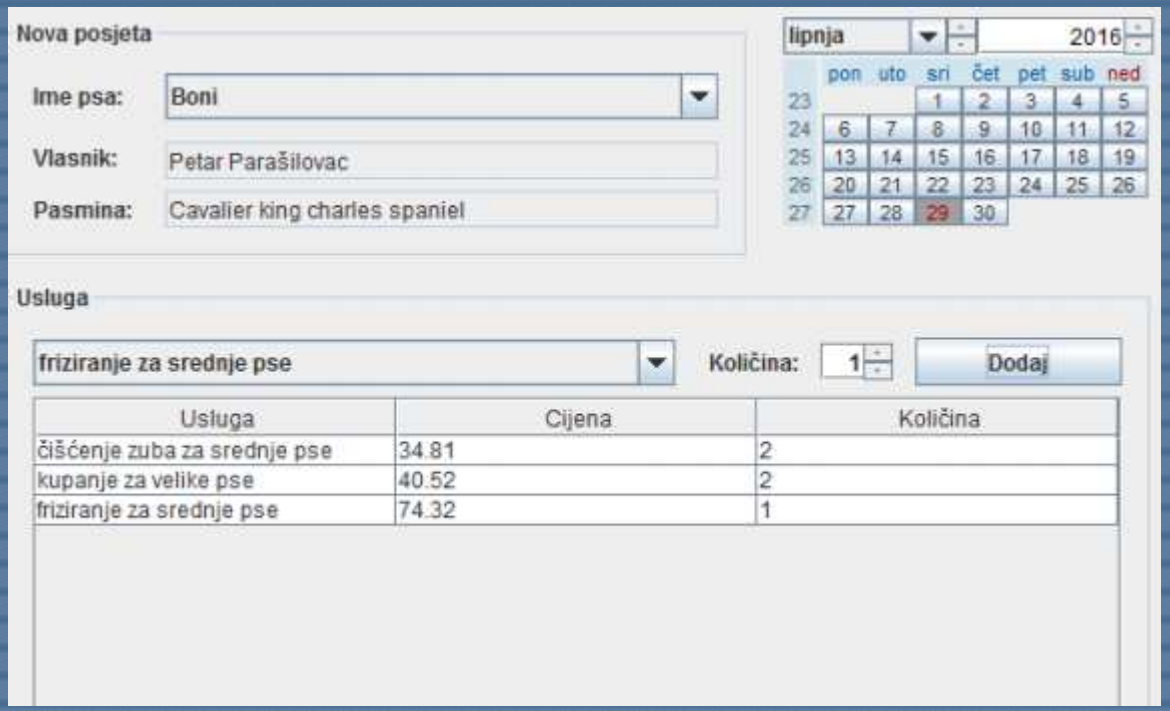
- PROZOR STATISTIKE
- Neke podatke trebamo prikazati kao *pie chart* – biblioteka `jfreechart`


```
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", pieDat
    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);
}
```

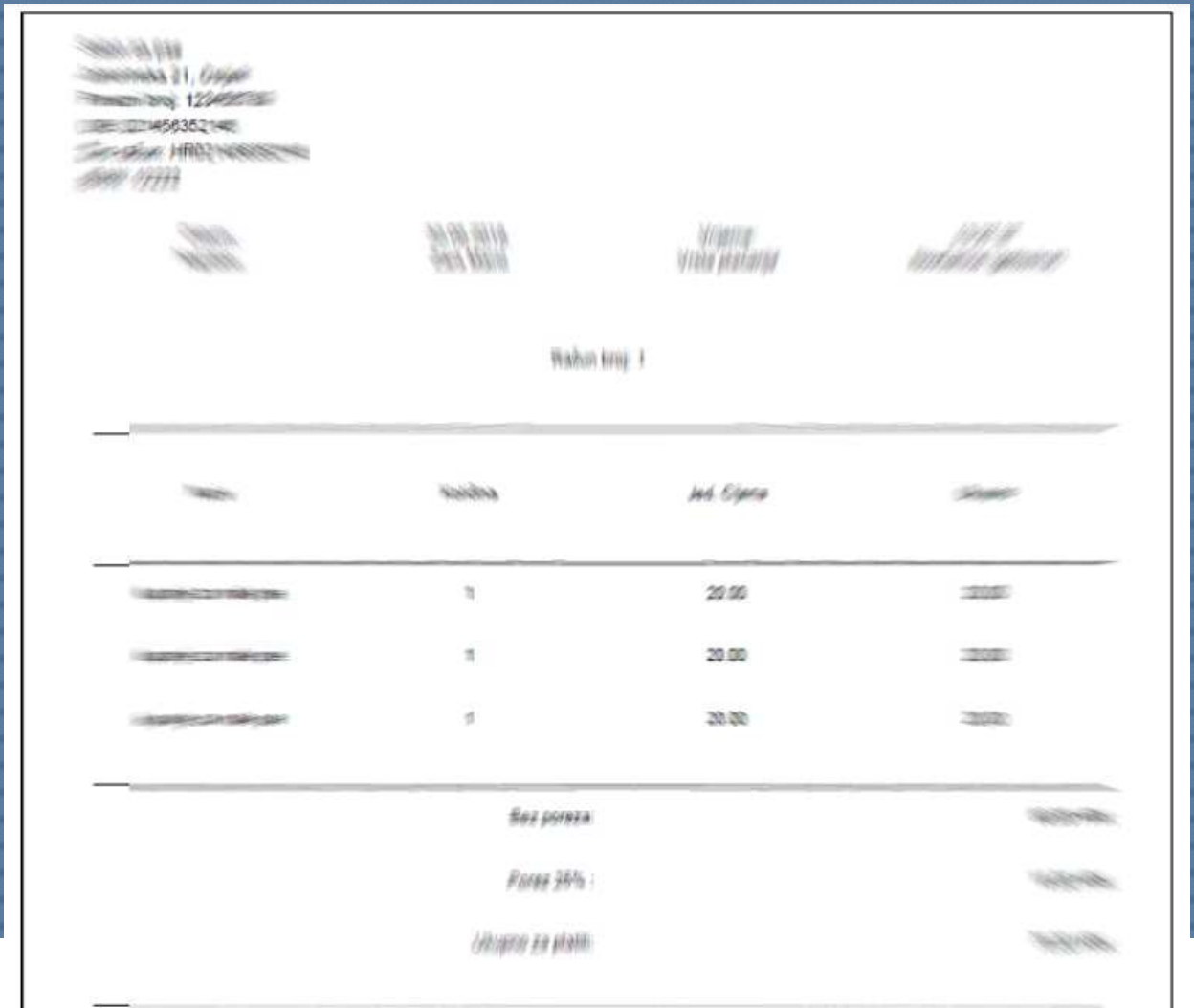
Korištene tehnologije

- PROZOR KALENDAR
- U slučaju da nam treba kalendar - `jcalendar`



Korištene tehnologije

- PDF IZVJEŠTAJ
- Za generisanje PDF-a (na lokalni računar) korištena je biblioteka `itextpdf`



Korištene tehnologije

- KONTROLA UNOSA
- Kontrola da se u bazu mogu ubaciti samo isporavne vrijednosti
- Ime/Prezime:

```
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());
```

Korištene tehnologije

- KONTROLA UNOSA
- Regularni izrazi - *regex*: - izraz kojim se definiše uzorak koji se koristi za pretraživanje teksta (npr. OIB/JMBG potrebno je dozvoliti isključivo unos brojeva pa je regularni izraz definsan kao String vrijednosti „\\d+“

```
    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;
    }
```

Korištene tehnologije

- KONTROLA UNOSA
- Kontrola unosa numeričke vrijednosti/valute: ukoliko je dužina vrijednosti veća od nule, postavi vrijednost u bazu.
- Ukoliko ne uspije, izvršiti će se catch blok jer je vrijednost iste definisana kao BigDecimal

```
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;
    }
}
```