UNIVERZITET ZA POSLOVNI INŽENJERING I MENADŽMENT BANJA LUKA

REDOVNE STUDIJE
SMIJER: INFORMACIONE TEHNOLOGIJE

Seminarski rad iz predmeta:

OBJEKTNO ORJENTISANO PROGRAMIRANJE

Tema:

IZRADA DESKTOP APLIKACIJE „STUDENTSKA SLUŽBA“ U JAVA PROGRAMSKOM JEZIKU

Profesor:

Saša Salapura

Student:

Bunčić Dejan
Br. Indeksa 1-134/18

Banja Luka Jun 2019.

SADRŽAJ

1. UVOD

Programski jezik Java je jedan od najpopularnijih razvojnih platformi koja ima široku primjenu u razvoju informacionih sistema. Koristi se za razvoj širokog spektra programskih rješenja, od serverskih aplikacija, desktop aplikacija, web aplikacija do aplikacija za mobilne uređaje. Java je dostupna za većinu operativnih sistema (Windows, Linux, Mac itd.) , i zahvaljujući Java virtalnoj mašini program pisan u Javi se može pokrenuti na svim operativnim sistemima.

Aplikacija „Studentska služba“ je zamišljena kao kompletan IS jednog univerziteta, a u seminarskom radu će biti opisan način izrade kao i osnovne funkcionalnosti za dva dijela aplikacije a to su prijava u samu aplikaciju i rad sa osnovnim podacima studenata.

2. KORIŠTENE TEHNOLOGIJE

Aplikacija je razvijana u razvojnom okruženju NetbeansIDE (verzija 8.2) uz Java development Kit (verzija 1.8u211). Pisana je Java programskim jezikom i korišene su Swing gui komponente. Baza za potrebe aplikacije je mySQL (engl. Structured Query Language) verzija 8.

Slika 1. Java i MySql

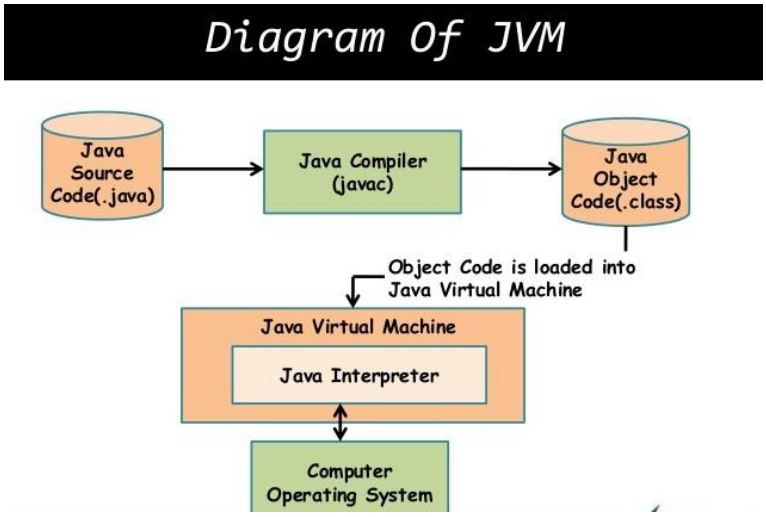
2.1. PROGRAMSKI JEZIK JAVA

Java je objektno orjentisan programski jezik koji su razvili *James Gosling, Patrick Naughton* i drugi inženjeri u kompaniji *Sun Microsystems*. Razvoj je počeo 1991, kao dio projekta *Green,* a objavljen je u novembru 1995.Kompanija *Sun* posjeduje trademark na ime Java, ali samo dev kit je moguće bez plaćanja skinuti sa Sun internet sajta.Velika prednost u odnosu na većinu dotadašnjih programskih jezika je to što se programi pisani u Javi mogu izvoditi bez prilagođavanja na svim operativnim sistemima za koje postoji JVM *(Java Virtual Machine),* dok je klasične programe pisane na primjer u C-u potrebno prilagođavati platformi (Operativnom sistemu) na kome se pokreću.

Zbog toga i i zbog bogate grupe klasa za rad s mrežnim komunikacijama u jednom trenutku je Java bila najbolji izbor za široku lepezu mogućih aplikacija. *Microsoft* je zato razvio svoj C# i .NET platformu kao odgovor na open source alternative.

Java je jedan od najkorištenijih programskih jezika. Procjene o broju korisnika kreću se od 7 do preko 10 miliona. Iako inspirisan jezikom C, Java pruža bolji stepen sigurnosti i pouzdanosti zahvaljujući VM-u i hermetički zatvorenom okolišu u kome svaki program funkcioniše: na Javi se brže razvija program s manje grešaka.

Upravo zbog toga je popularna za razvoj programa na mobilnim telefonima i kod finansijskih kompanija. Javlja se kao osnovni jezik za programiranje Googleovog sistema *Android*.



Slika 2. Princip izvršavanja java aplikacije

2.2. INTEGRISANO RAZVOJNO OKRUŽENJE NET BEANS

NetBeans je integrisano razvojno okruženje (IDE) prvenstveno namenjeno razvoju Java aplikacija, ali isto tako pruža dosta dodatnih mogućnosti koje mu omogućavaju da se jednako efikasno može koristiti za razvoj računarskih programa i u ostalim programskim jezicima kao što su C, C++, PHP, Fortran, Pajton, Rubi i drugi. NetBeans jednako dobro radi na različitim platformama kao što su Windows, Linux, MacOS. Podražava različite tehnologije i alate koji poboljšavaju razvojni proces aplikacije.

Slika 3. NetBeans IDE

2.3. MYSQL BAZA PODATAKA

Baza podataka koja je korištena za skladištenje podataka je *MySQL*. *MySQL* je besplatan, open source. Uz *PostgreSQL MySQL* je čest izbor baze za projekte otvorenog koda, te distribuiše kao sastavni dio serverskih Linux distribucija, no također postoje verzije i za ostale operativne sisteme poput Mac OS-a, Windows-a itd.

Ranije u svom razvoju, *MySQL* baza podataka suočila se s raznim protivnicima *MySQL* sistema organizovanja podataka jer su joj nedostajale neke osnovne funcije definisane SQL standardom. Naime, *MySQL* baza je optimizovana kako bi bila brza nauštrb funkcionalnosti. Nasuprot tome, vrlo je stabilna i ima dobro dokumentisane module i ekstenzije te podršku od brojnih programskih jezika: PHP, Java , Perl, Python...

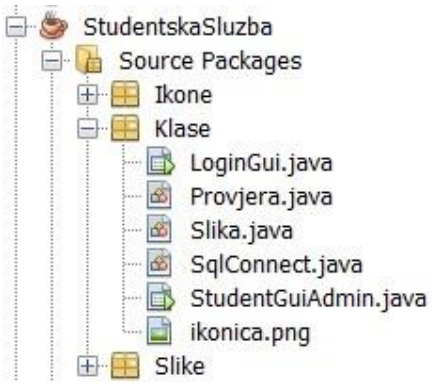
MySQL baze su relacionog tipa, koji se pokazao kao najbolji način skladištenja i pretraživanja velikih količina podataka i u suštini predstavljaju osnovu svakog informacionog sistema, tj. temelj svakog poslovnog subjekta koji svoje poslovanje bazira na dostupnosti kvalitetnih i brzih informacija.

Slika 4. MySql baza

3. STRUKTURA APLIKACIJE

Aplikacija „Studentska Služba“ se bazira na pomenutoj *MySQL* bazi podataka, a sastoji se od tri paketa , od kojeg su dva korištena za ikonice i dodatne slike a treća za klase.

U paketu *„Klase“* se nalazi pet Java klasa , od kojeg su dvije GUI klase **LoginGui** i **StudentGuiAdmin** u kojoj se sadrži main metod, i tri klase od kojih **SqlConnect** služi za konektovanje na bazu i za ostale funkcije nad bazom podataka. Klasa **Slika** sadrži funkcije za za manipulaciju sa slikama studenata , a funkcija **Provjera** služi za provjeru ispravnosti unosa podatak.



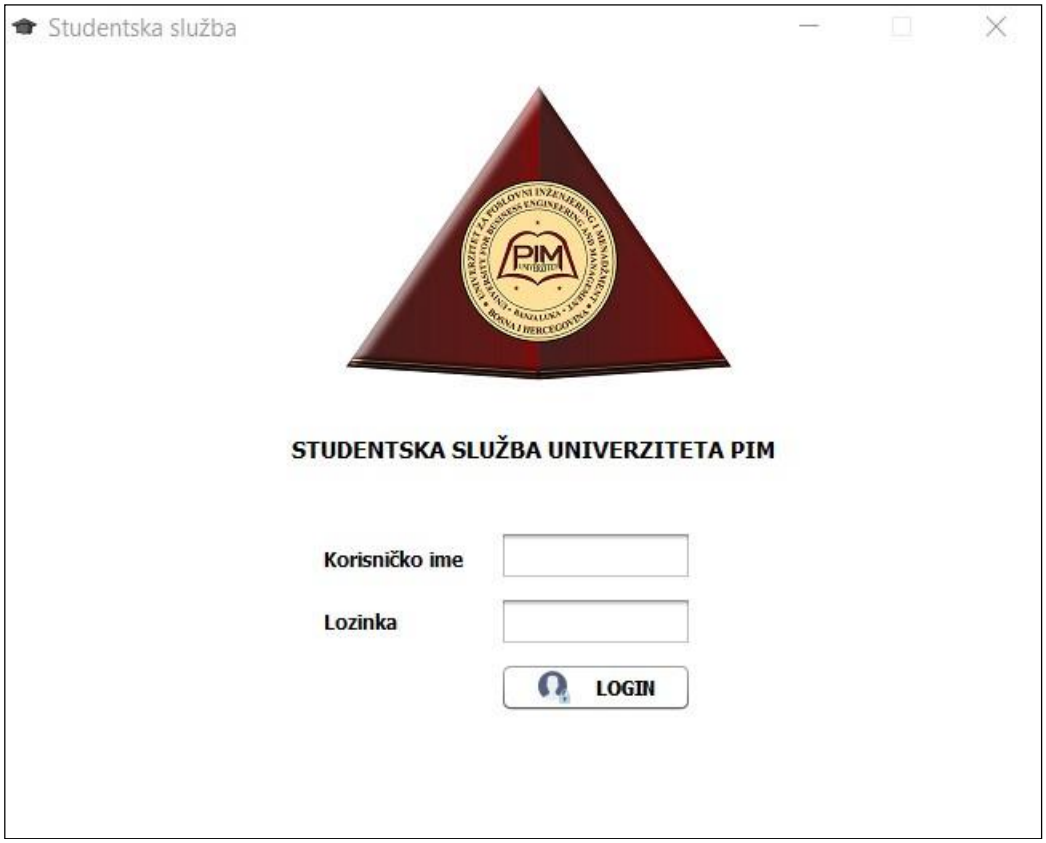
Slika 5. Struktura aplikacije

4. IZGLED APLIKACIJE

Aplikacija se sastoji od dva Java okvira (engl. jFrame). Prvi jFrame **LoginGui** služi za logovanje u aplikaciju a drugi jFrame **StudentGuiAdmin** služi za manipulaciju sa podacima studenata, unos, brisanje, pretragu, štampu i prepravljanje podataka i sastoji se od 35 Swing komponenti.

4.1. PRIJAVA U APLIKACIJU

Prvi je prozor koji se prikaže kada pokrenete aplikaciju je GUI za login korisnika i prikazan je na slici broj 6. Od korisnika se traži da upišu korisničko ime i lozinku. Navedeni parametri definisani su u bazi podataka. Klikom na dugme „LOGIN“, ukoliko su unešeni podaci tačni, otvara se prozor za rad sa podacima studenata.



Slika 6. Prijava korisnika

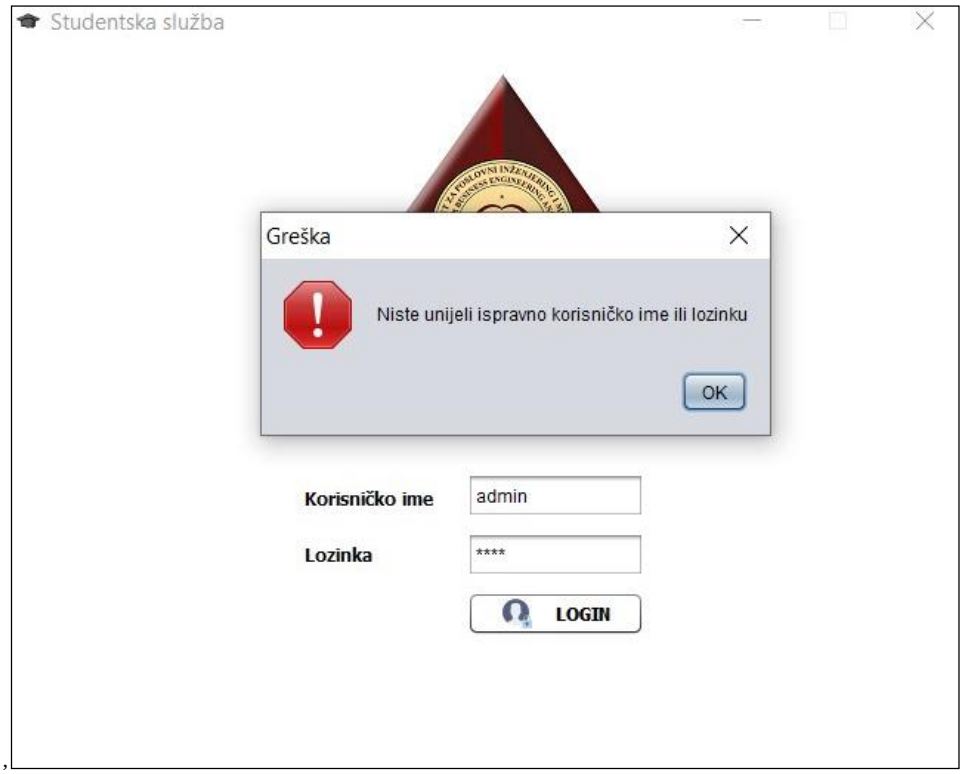
Na slici broj 7 je prikazan primjer greške ako korisnik ne unese korisničko ime ili lozinku , klikom na OK korisnik ima mogućnost da ponovi unos. Na slici br. 8 je prikazan kod koji provjerava unos , tj. ako su polja prazna program ispiše grešku , a ako su popunjena poziva se funkcija za login.

Slika 7. Greška prilikom prijave

```
private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {

    if(txtUsername.getText().isEmpty()|| txtPassword.getText().isEmpty()){


        JOptionPane.showMessageDialog(null, "Niste unijeli korisničko ime ili lozinku", "GREŠKA", JOptionPane.ERROR_MESSAGE);

        return;
    }


     if(this.sql.loginKorisnika(this.txtUsername.getText(), this.txtPassword.getText())){
        zatvori();
     }
```

Slika 8. Kod za provjeru unosa korisničkog imena i lozinke

Ako korisnik unese pogrešno korisničko ime ili lozinku program prikaže grešku (slika br. 9) , a ako su korisničko ime i lozinka ispravni poziva se funkcija **loginKorisnika** koja kreira novi objekat **StudentGuiAdmin** i isti se prikazuje a zatvara se **LoginGui**.



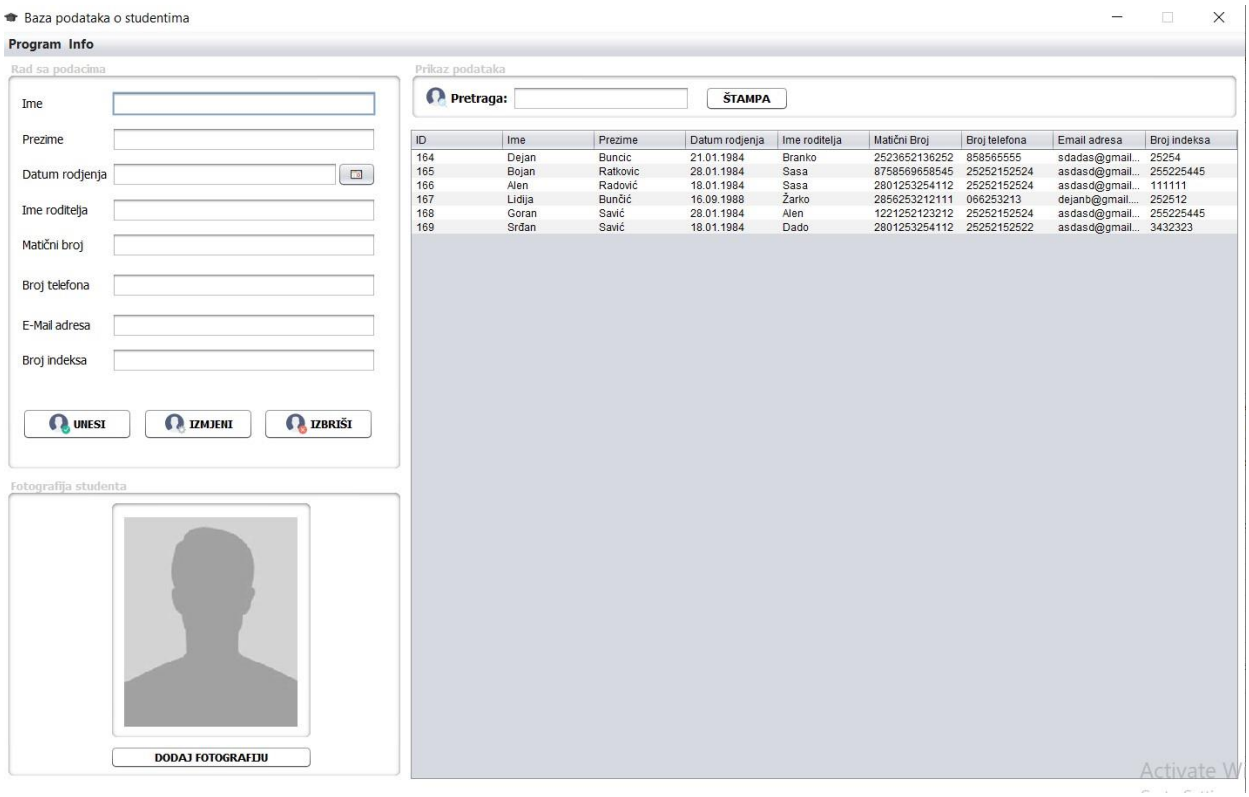
Slika 9. Greška prilikom unosa pogrešnog korisničkog imena ili lozinke

```
public boolean loginKorisnika(String username, String password) { //provjera da li ima korisnika u bazi

    try {

        ps = konekcija.prepareStatement("SELECT * FROM korisnik where korisnickoIme = '" + username + "'  AND lozinka = '" + password + "'");

        rs = ps.executeQuery();
        if (rs.next()) {
            StudentGuiAdmin sg = new StudentGuiAdmin();
            sg.setVisible(true);
            sg.pack();
            sg.setLocationRelativeTo(null);
            rs.close();
            ps.close();
            return true;




        } else {
            JOptionPane.showMessageDialog(null, "Niste unijeli ispravno korisničko ime ili lozinku", "Greška", JOptionPane.ERROR_MESSAGE);
        }

    } catch (Exception e) {

        System.out.println(e);

    }

    return false;
```

Slika 10. Kod za login korisnika

4.2. FORMA ZA ADMINISTRACIJU PODATAKA

Nakon uspješne prijave (u ovom slučaju prijave administratora) otvara se prozor za rad sa podacima studenata (slika br. 11). Administrator može da unese novog studenta u bazu , izmjeni podatke, izbriše studenta iz baze i da vrši pretragu.



Slika 11. Administracija podataka

4.3. UNOS PODATAKA / KREIRANJE NOVOG STUDENTA

Nakon popunjavanja svih polja i učitavanja fotografije klikom na dugme **„DODAJ FOTOGRAFIJU“** klikom na dugme „**UNESI**“ vrši se ubacivanje podataka u bazu , ali prije toga se vrši provjera ispravnosti unešenih podataka (Slika 15) . Za provjeru se koristi klasa **Provjera** koja u sebi ima funkcije za provjeru ispravnosti i kontrolu unešenih podataka (Slika 12).

```
public static final Pattern VALIDNOST_EMAIL_ADRESE =
            Pattern.compile("^[A-Z0-9._%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,6}$", Pattern.CASE_INSENSITIVE);


   public static boolean provjeraJmbg(String jmbg) {

       if (jmbg.length() != 13 ) {

           return true;
       }

       if(!Character.isDigit(jmbg.charAt(0))|| !Character.isDigit(jmbg.charAt(1))|| !Character.isDigit(jmbg.charAt(2))||
            !Character.isDigit(jmbg.charAt(3))|| !Character.isDigit(jmbg.charAt(4))|| !Character.isDigit(jmbg.charAt(5))
            || !Character.isDigit(jmbg.charAt(6))|| !Character.isDigit(jmbg.charAt(7))|| !Character.isDigit(jmbg.charAt(8))
            || !Character.isDigit(jmbg.charAt(9))|| !Character.isDigit(jmbg.charAt(10))|| !Character.isDigit(jmbg.charAt(11))
            || !Character.isDigit(jmbg.charAt(12))){
           return true;
       }

       return false;
}

   public static boolean provjeraEmailAdrese(String email) {
       Matcher matcher = VALIDNOST_EMAIL_ADRESE .matcher(email);
       return matcher.find();
       }


   public static boolean provjeraUnesenihPolja(String provjera){

       if(provjera.length()<3 || Character.isDigit(provjera.charAt(0))){
           return true;
       }

       return false;
```

Slika 12. Klasa Provjera

```
private void jButtonDodajFotografijuActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser jfc = new JFileChooser();
    FileNameExtensionFilter fnf = new FileNameExtensionFilter("JPG file", "jpg");
    jfc.setFileFilter(fnf);
    int rezultat = jfc.showSaveDialog(null);
    if (rezultat == JFileChooser.APPROVE_OPTION) {

        File novaSlika = jfc.getSelectedFile();
        String putanja = novaSlika.getAbsolutePath();

        jLabelSlika.setIcon(Slika.obradiSliku(putanja, null));
        putanjaSlike = putanja;
    } else {
        System.out.println("Niste odabrali sliku");
    }
```

Slika 13. Kod za odabir fotografije

```
public static ImageIcon obradiSliku(String putanjaSlike, byte[] bajtoviSlike) {

    ImageIcon slika = null;

    if (putanjaSlike != null) {

        slika = new ImageIcon(putanjaSlike);
    } else {

        slika = new ImageIcon(bajtoviSlike);
    }

    Image ikonica = slika.getImage();
    Image slika2 = ikonica.getScaledInstance(200, 240, Image.SCALE_SMOOTH);

    ImageIcon ikonicaFinal = new ImageIcon(slika2);

    return ikonicaFinal;

}
```

Slika 14. Funkcija za obradu slike

```
private void btnUnesiActionPerformed(java.awt.event.ActionEvent evt) {
    if (Provjera.provjeraUnesenihPolja(txtIme.getText())) {
        JOptionPane.showMessageDialog(null, "Obavezan unos imena", "Greška", JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (Provjera.provjeraUnesenihPolja(txtPrezime.getText())) {
        JOptionPane.showMessageDialog(null, "Niste unijeli ispravno Prezime", "Greška", JOptionPane.ERROR_MESSAGE);
        return;
    }

    Date datum = jDatum.getDate();

    if (datum == null) { //prvojera da li je unijet datum rodjenja
        JOptionPane.showMessageDialog(null, "Niste unijeli datum rodjenja");
        return;
    }
    if (Provjera.provjeraUnesenihPolja(txtImeRoditelja.getText())) {
        JOptionPane.showMessageDialog(null, "Niste unijeli ispravno Ime roditelja ", "Greška", JOptionPane.ERROR_MESSAGE);
        return;
    }

    if (Provjera.provjeraJmbg(txtMaticniBroj.getText())) {
        JOptionPane.showMessageDialog(null, "Neispravno unešen matični broj ", "Greška", JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (txtBrojTelefona.getText().length() < 9 || !Character.isDigit(txtBrojTelefona.getText().charAt(0))) {
        JOptionPane.showMessageDialog(null, "Niste ispravno unijeli broj telefona ", "Greška", JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (!Provjera.provjeraEmailAdrese(txtEmailAdresa.getText())) {
        JOptionPane.showMessageDialog(null, "Niste unijeli ispravno Email adresu  ", "Greška", JOptionPane.ERROR_MESSAGE);
        return;
    }
```

Slika 15. Provjera unosa prije upisa u bazu

```
SimpleDateFormat Date_Format = new SimpleDateFormat("dd.MM.yyyy"); //formatiranje datuma 28.01.1984
String datumRodjenja = Date_Format.format(datum);
Connection kon = sql.konekcijaNaBazu();
PreparedStatement ps;
InputStream slika = new FileInputStream(new File(putanjaSlike));

try {
    ps = kon.prepareStatement("insert into student (ime,prezime,datumRodjenja,imeRoditelja,jmbg,telefon,email,brojIndeksa,slika)"
            + "values(?,?,?,?,?,?,?,?,?)");

    ps.setString(1, txtIme.getText());
    ps.setString(2, txtPrezime.getText());
    ps.setString(3, datumRodjenja);
    ps.setString(4, txtImeRoditelja.getText());
    ps.setString(5, txtMaticniBroj.getText());
    ps.setString(6, txtBrojTelefona.getText());
    ps.setString(7, txtEmailAdresa.getText());
    ps.setString(8, txtBrojIndeksa.getText());
    ps.setBlob(9, slika);

    ps.executeUpdate();

    ps.close();

} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "GREŠKA");
    Logger.getLogger(StudentGuiAdmin.class.getName()).log(Level.SEVERE, null, ex);
}

this.sql.prikaziIzBazePodatke(tblStudent);
resetujPolja();
```

Slika 16. Zapisivanje podataka u bazu

Nakon što se izvrše sve provjere ispravnosti unosa , podaci se upisuju u bazu (Slika 16) , a zatim se prikazuju osvježena polja u tabeli pozivom funkcije **prikaziIzBazePodataka** (Slika 17) a zatim funkcija **resetujPolja** (slika 18) za postavljanje svih polja za unos na zadane vrijednosti .

```
public void prikaziIzBazePodatke(JTable jtb) {

    try {
        int rows = 0;
        int rowIndex = 0;

        String query = "select * from student";
        ps = konekcija.prepareStatement(query);
        rs = ps.executeQuery();

        if (rs.next()) {
            rs.last();
            rows = rs.getRow();
            rs.beforeFirst();

        }
        String[][] data = new String[rows][9];

        while (rs.next()) {
            data[rowIndex][0] = rs.getInt(1) + "";
            data[rowIndex][1] = rs.getString(2);
            data[rowIndex][2] = rs.getString(3);
            data[rowIndex][3] = rs.getString(4);
            data[rowIndex][4] = rs.getString(5);
            data[rowIndex][5] = rs.getString(6);
            data[rowIndex][6] = rs.getString(7);
            data[rowIndex][7] = rs.getString(8);
            data[rowIndex][8] = rs.getString(9);

            rowIndex++;
        }
        String[] kolone = {"ID", "Ime", "Prezime", "Datum rodjenja", "Ime roditelja", "Matični Broj", "Broj telefona", "Email adresa", "Broj indeksa"};
        DefaultTableModel model = new DefaultTableModel(data, kolone);
        jtb.setModel(model);

    } catch (Exception ex) {
        System.out.println(ex);
    }

}
```

Slika 17. Funkcija za prikaz iz baze

```
private void resetujPolja() {

    this.txtIme.setText(null);
    this.txtPrezime.setText(null);
    this.jDatum.setDate(null);
    this.txtImeRoditelja.setText(null);
    this.txtMaticniBroj.setText(null);
    this.txtBrojTelefona.setText(null);
    this.txtEmailAdresa.setText(null);
    this.txtBrojIndeksa.setText(null);
    this.jLabelSlika.setIcon(new ImageIcon(getClass().getResource("/Slike/noimg.jpg")));

}
```

Slika 18. Funkcija za resetovanje polja

4.4. IZMJENA PODATAKA

Klikom na dugme **„IZMJENI“** vrši se poziv funkcije ***izmjeni*** koja vrši provjeru i uzima vrijednosti iz polja a zatim mijenja postojeće podatake od studenta (Slika 19 , 20 , 21).

```
public void izmjeni(String txtIme, String txtPrezime, String txtImeRoditelja, String txtMaticniBroj, String txtBrojTelefona,
        String txtEmailAdresa,String txtBrojIndeksa ,int studentid){

    if (Provjera.provjeraUnesenihPolja(txtIme)) {
        JOptionPane.showMessageDialog(null, "Niste unijeli ispravno Ime", "Greška", JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (Provjera.provjeraUnesenihPolja(txtPrezime)) {
        JOptionPane.showMessageDialog(null, "Niste unijeli ispravno Prezime", "Greška", JOptionPane.ERROR_MESSAGE);
        return;
    }

     if (Provjera.provjeraUnesenihPolja(txtImeRoditelja)) {
        JOptionPane.showMessageDialog(null, "Niste unijeli ispravno Ime roditelja ", "Greška", JOptionPane.ERROR_MESSAGE);
        return;
    }

    if (Provjera.provjeraJmbg(txtMaticniBroj)) {
        JOptionPane.showMessageDialog(null, "Neispravno unešen matični broj ", "Greška", JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (txtBrojTelefona.length() < 9 || !Character.isDigit(txtBrojTelefona.charAt(0))) {
        JOptionPane.showMessageDialog(null, "Niste ispravno unijeli broj telefona ", "Greška", JOptionPane.ERROR_MESSAGE);
        return;
    }
    if(!Provjera.provjeraEmailAdrese(txtEmailAdresa)){
        JOptionPane.showMessageDialog(null, "Niste unijeli ispravno Email adresu  ", "Greška", JOptionPane.ERROR_MESSAGE);
        return;
    }
```

Slika 19. Provjera unosa

```
    try {
    String query = "update student set ime='" + txtIme + "',prezime='" + txtPrezime + "',imeRoditelja='" + txtImeRoditelja + "'"
                    + ",jmbg='" + txtMaticniBroj + "',telefon='" + txtBrojTelefona + "',email='" + txtEmailAdresa + "',brojIndeksa='"
                    +txtBrojIndeksa+"' where student_id =" + studentid;

        st = konekcija.createStatement();
        int rezultat = st.executeUpdate(query);
        if (rezultat == 1) {

            JOptionPane.showMessageDialog(null, "Uspjesno ste izvršili izmjenu u bazi");

            st.close();
            rs.close();

        } else {

            JOptionPane.showMessageDialog(null, "GREŠKA","Greška",JOptionPane.ERROR_MESSAGE);
        }

    } catch (Exception e) {


    }

}
```

Slika 20. Izmjena podataka

```
private void btnIzmjeniActionPerformed(java.awt.event.ActionEvent evt) {

    try {

        studentid = Integer.parseInt(tblStudent.getValueAt(tblStudent.getSelectedRow(), 0).toString());

        this.sql.izmjeni(this.txtIme.getText(), this.txtPrezime.getText(),
                this.txtImeRoditelja.getText(), this.txtMaticniBroj.getText(),
                this.txtBrojTelefona.getText(), this.txtEmailAdresa.getText(), this.txtBrojIndeksa.getText(), studentid);

        this.sql.prikaziIzBazePodatke(tblStudent);

        resetujPolja();
    } catch (Exception e) {

        JOptionPane.showMessageDialog(null, "Greska , niste odabralli studenta ");
    }
```

Slika 21. Poziv funkcije za izmjenu podataka

4.5. BRISANJE PODATAKA IZ BAZE

Brisanje se vrši tako što se u tabeli klikne na studenta kojeg se želi obrisati a zatim se klikne na dugme **„BRIŠI“** vrši brisanje odabranog studenta iz baze (Slika 22, 23, 24).

```
private void tblStudentMouseClicked(java.awt.event.MouseEvent evt) {

    studentid = Integer.parseInt(tblStudent.getValueAt(tblStudent.getSelectedRow(), 0).toString());

    if (studentid == 0) {
        JOptionPane.showMessageDialog(null, "Greška , niste odabrali studenta");

    }

    try {
        this.sql.prikaziPodateUPoljima(txtIme, txtPrezime, jDatum, txtImeRoditelja, txtMaticniBroj, txtBrojTelefona, txtEmailAdresa, txtBrojIndeksa, jLabelSlika, studentid);

    } catch (IOException | ParseException ex) {
        Logger.getLogger(StudentGuiAdmin.class.getName()).log(Level.SEVERE, null, ex);
    }
```

Slika 22. Odabir studenta

```
public void izbrisi(String id) {

    if(id == null ){

        return;
    }

    try {

        String query = "delete from student where student_id = '" + id + "' ";
        st = konekcija.createStatement();
        int rezultat = st.executeUpdate(query);
        if (rezultat == 1) {

            JOptionPane.showMessageDialog(null, "Uspjesno ste izvršili brisanje iz baze");
        } else {

            JOptionPane.showMessageDialog(null, "Neuspješno brisanje", "GREŠKA",JOptionPane.ERROR_MESSAGE);
        }

    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, e);
    }
```

Slika 23. Funkcija za brisanje

```
private void btnIzbrisiActionPerformed(java.awt.event.ActionEvent evt) {

    if (this.tblStudent.getSelectedRow() != -1) {
        int daLiZeliteIzbrisati = JOptionPane.showConfirmDialog(null, "Da li ste sigurni?", "BRISANJE", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);
        if (daLiZeliteIzbrisati == 0) {
            String id = String.valueOf(this.tblStudent.getValueAt(this.tblStudent.getSelectedRow(), 0));
            this.sql.izbrisi(id);
            this.sql.prikaziIzBazePodatke(tblStudent);
            resetujPolja();
}
    }
}
```

Slika 24.Poziv funkcije za brisanje studenta

4.6. PRETRAGA I ŠTAMPA

Pretraga se vrši tako što se kursor postavi u polje za pretragu a zatim se kuca ime ili prezime a kao rezultat se vraćaju polja koja sadrže te karaktere i prikazuju se u tabeli , pretraga se vrši pomoću funkcije **pretragaPoImenuIPrezimenu** (Slika 26) , a štampa podataka željenog studenta se vrši pritiskom na dugme **ŠTAMPA**.

```
private void txtPretraga1KeyReleased(java.awt.event.KeyEvent evt) {
    this.sql.pretragaPoImenuIPrezimenu(tblStudent, this.txtPretraga1.getText(), this.txtPretraga1.getText());
}
```

Slika 25. Polje pretraga

```
try {

    String query = "select * from student where ime like '%" + trazenoIme + "%' or prezime like '%" + trazenoPrezime + "%'";

    st = konekcija.createStatement();
    rs = st.executeQuery(query);
    int brojac = 0;
    while (rs.next()) {

        brojac++;

    }
    Object nazivKolona[] = {" ID ", "Ime", "Prezime", "Datum rodjenja", "Ime roditelja", "Matični broj", "Broj telefona", "Email adresa", "Broj indeksa"};
    Object podaci[][] = new Object[brojac][9];
    rs = st.executeQuery(query);
    int i = 0;
    while (rs.next()) {
        podaci[i][0] = rs.getInt("student_id");
        podaci[i][1] = rs.getString("ime");
        podaci[i][2] = rs.getString("prezime");
        podaci[i][3] = rs.getString("datumRodjenja");
        podaci[i][4] = rs.getString("imeRoditelja");
        podaci[i][5] = rs.getString("jmbg");
        podaci[i][6] = rs.getString("telefon");
        podaci[i][7] = rs.getString("email");
        podaci[i][8] = rs.getString("brojIndeksa");
        i++;

    }

    jtb.setModel(new javax.swing.table.DefaultTableModel(podaci, nazivKolona));

} catch (Exception e) {
```

Slika 26. Funkcija za pretragu

```
private void jButtonStampajActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        MessageFormat header  = new MessageFormat("----- PODACI O STUDENTIMA -----");
        MessageFormat footer = new MessageFormat("Univerzitet PIM");
        tblStudent.print(JTable.PrintMode.FIT_WIDTH,header,footer);
    } catch (PrinterException ex) {
        Logger.getLogger(StudentGuiAdmin.class.getName()).log(Level.SEVERE, null, ex);
    }


}
```

Slika 27. Kod za štampu

ZAKLJUČAK

Izrada Java desktop aplikacije „Studentska služba“ je tema ovog seminarskog rada. Za potrebe rada izrađena je SQL baza podataka koja se sastoji od dvije tabele. Aplikacija je temeljena na navedenoj bazi te omogućava pregledavanje, ažuriranje, promjenu i brisanje podataka kao i štampu podataka o studentu. Sve navedene funkcije prikazane su na zanimljiv način. Aplikacija nudi korisniku lako baratanje potrebnim podacima te jednostavno kreiranje novih studenata .

LITERATURA

1. wikipedia.org
2. youtube.com
3. java.com
4. pluralsight.com
5. scribd.com
6. google.com
7. salapura.com/java