Java aplikacija za osnovne operacije nad tablicom u bazi podataka

Đurić, Dario

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Stručni studij

JAVA APLIKACIJA ZA OSNOVNE OPERACIJE NAD TABLICOM U BAZI PODATAKA

Završni rad

Dario Đurić

Osijek, 2019.

Sadržaj

1. UVOD

Ideja završnog rada bila je izrada programskog rješenja za rad s tablicama unutar baze podataka. Postoje četiri osnovne operacije kod baza podataka, a to su dodaj, obriši, učitaj i uredi. Te četiri operacije obuhvaćaju se engleskim nazivom *crud*, gdje svako slovo predstavlja naziv operacije na engleskom jeziku. Ovom aplikacijom olakšava se cjelokupan rad s tablicom u bazi podataka. Unutar aplikacije, svaki red iz tablice će biti ispisan odvojeno.

Ovaj završni rad uz Java aplikaciju sadrži četiri glavna poglavlja, te popis literature, sažetak i sažetak na engleskom jeziku. Nakon uvoda u temu i kratkog objašnjenja zadatka u prvom poglavlju, u drugom poglavlju teorijski su opisane tehnologije korištene kroz završni rad. U trećem poglavlju opisana je izrada aplikacije, te su objašnjene funkcije koje aplikacija koristi. Također, predočen je programski kod za neke od funkcija. Zadnje poglavlje odnosno zaključak sadrži kratak osvrt na cijeli rad.

1.1. Zadatak završnog rada

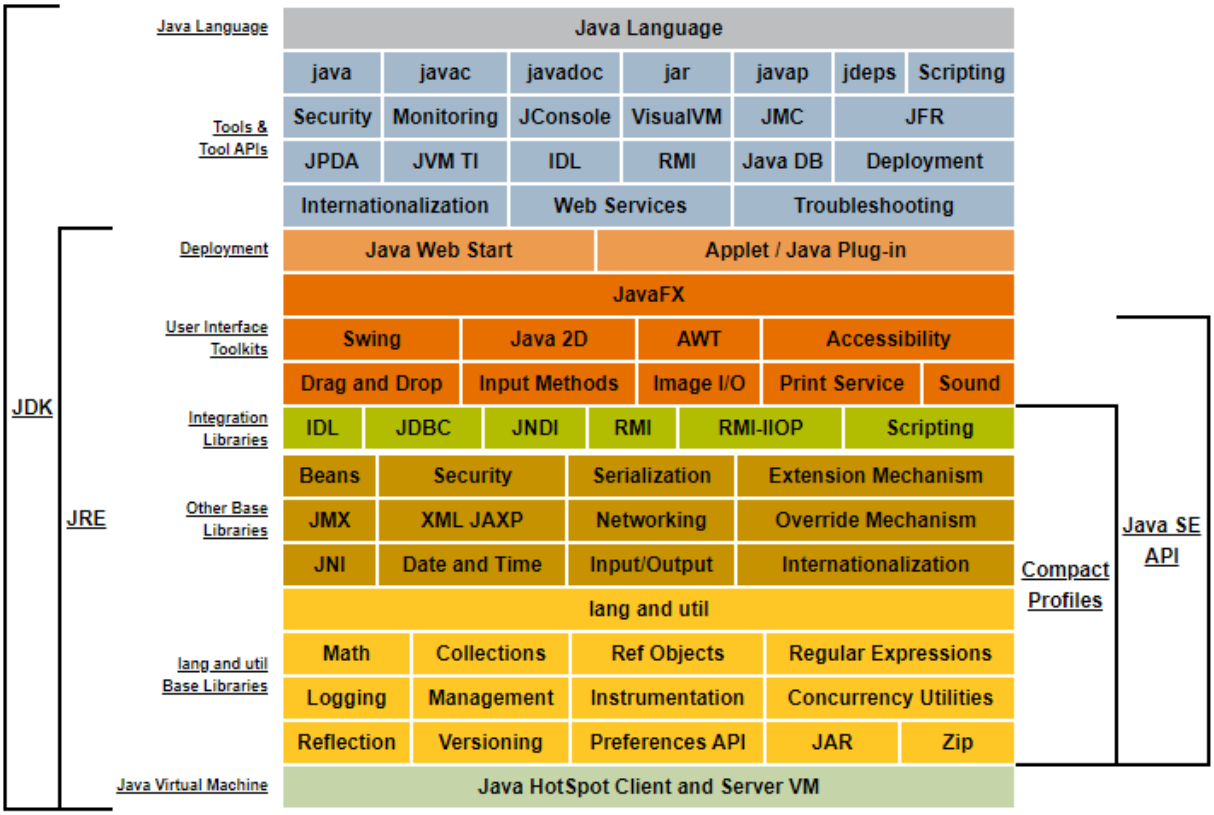
Zadatak ovog završnog rada je izrada programskog rješenja za osnovne operacije nad tablicom u bazi podataka. Među osnovne operacije spadaju dodaj, obriši, učitaj i uredi. Programski jezik u kojem je osmišljeno rješenje je Java, dok je baza podataka s tablicama u MySQL-u.

2. OPIS KORIŠTENIH TEHNOLOGIJA

2.1 Programski jezik Java

Java predstavlja objektno orijentirani programski jezik razvijen od strane „Sun Microsystems“ 1991.godine. Objavljen je 1995.godine, a nicijalni naziv programskog jezika bio je „Oak“. Inspiriran je C programskim jezikom. Velika prednost Java programskog jezika je neovisnost o arhitekturi. Jednom napisani Java kod može se izvšavati na bilo kojem operativnom sustavu koji ima instaliran Java virtualni stroj (engl. *Java Virtual Machine*). *Java Virtual Machine* predstavlja virtualni stroj kojim se izvršava Java kod. Prije izvršavanja koda na *Java Virtual Machine*, Java *compiler* prevodi izvorni kod u binarni (engl. *bytecode*). Java je i danas jedan od najkorištenijih programskih jezika, čiji se broj korisnika procjenjuje na oko 10 milijuna.

Zbog svoje sigurnosti i pouzdanosti koje pruža *Java Virtual Machine*, programski jezik Java koristi se kao osnovni jezik za programiranje na Googleovom mobilnom operacijskom sustavu Android.



Slika 2.1. Opis Java konceptualnog dijagrama, prema [1]

2.2 JavaFX

JavaFx predstavlja softversku platformu za izradu aplikacija sa grafičkim sučeljem (engl. *GUI applications*) koje se mogu koristiti na različitim uređajima. U prosincu 2008.godine tvrtka „Sun Microsystems“ objavljuje *framework* baziran na stilskim jezicima CSS i FXML koji je predstavljao nasljednika programskog jezika Swing.

JavaFX trenutno je član OpenJDK pod nazivom OpenJFX. Oracle želi ubrzati razvoj JavaFX platforme.

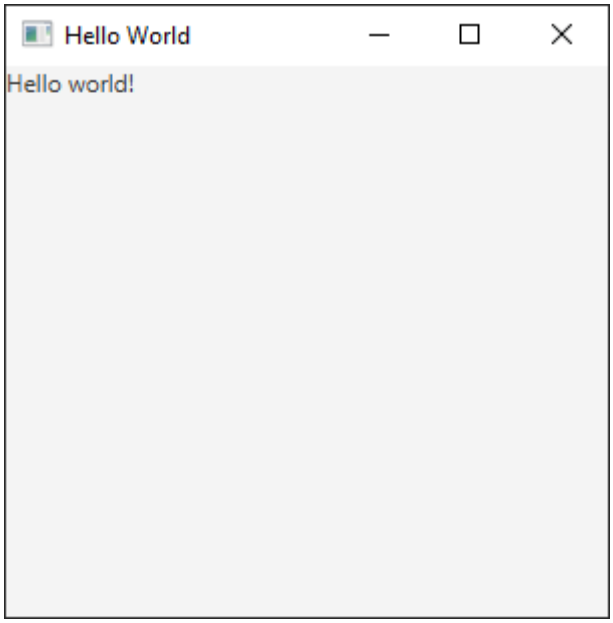
Slika 2.2. Logo programske platforme JavaFX, prema [2]

2.3 FXML

FXML predstavlja označiteljski jezik (engl. *user inteface markup language*) baziran na XML-u (engl. *Extensible Markup Language*). Razvila ga je tvrtka „Oracle“ 2011. godine. Nudi korisna rješenja za grafički prikaz Java koda zato što se sve napisano u FXML-u može direktno iskoristiti koristeći JavaFX platformu. Prednost korištenja FXML-a je odvajanje izgleda korisničkog sučelja i funkcionalnosti aplikacije. Dodatnu prednost pruža alat „Scene Builder“ koji omogućava grafičko dizajniranje izgleda aplikacije. Napisani kod ima ekstenziju .fxml i potrebno ga je učitati u Main klasi projekta. Na slici 2.3. prikazan je primjer koda napisanog koristeći FXML. Kod ispisuje jednostavan tekst na ekran što je prikazano na slici 2.4.

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml"
      fx:controller="sample.Controller">

    <Label text="Hello world!"/>

</VBox>
```

Slika 2.3. Primjer FXML koda



Slika 2.4. Ispis teksta preko jednostavnog FXML koda

2.4 MySQL

MySQL besplatan je sustav otvorenog tipa (engl. *open source*) za upravljanje bazama podataka. Koristi SQL (engl. *Structured Query Language*), te je jedan od relacijskih sustava za upravljanje bazama podataka. Relacijske baze pokazale su se kao najbolje za skladištenje i pretraživanje velike količine podataka. Unutar baze stvaraju se tablice koje se sadrže stupce i redove. Osnovni element koji se pohranjuje u bazu podataka naziva se entitet. Entitet predstavlja neku osobu, objekt ili samo podatak o kojem se pohranjuju pojedinosti. Te pojedinosti zovu se atributi, a poistovjećuju se sa stupcima.

Važan dio baza podataka jesu relacije. Postoje tri vrste relacija: jedan prema jedan, jedan prema više ili više prema jedan, te više prema više. Ovisno o potrebi, prilikom modeliranja baze uzima se relacija koja je potrebna.

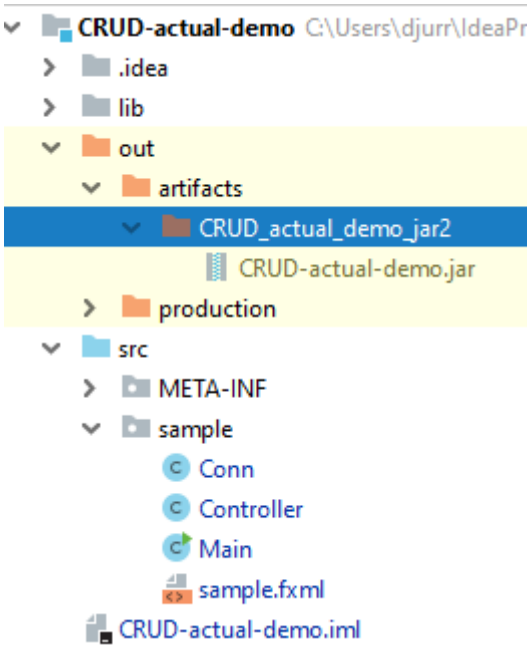
Za potrebe završnog rada korišten je besplatni sustav otvorenog tipa phpMyAdmin za izradu MySQL baze podataka. Na slici 2.5. prikazan je izgled jedne od tablica unutar baze.

id	ime	prezime	email
1	pero	peric	keskerec@gmail.com
2	pero	peric	djurraa@gmail.com
3	zaposlenik	zap	zap@amfka.com
4	Dzontra	Volta	dz.volt@ferit.hr
5	bruc	vlasic	cnbru@snk.hr
6	Gazda	Gazdinovic	gazda@firma.hr
7	rodhad	rodhadinovic	rodhad@exit.com
8	last	employee	last.employee@ferit.hr
9	test	testinovic	test@gmail.com

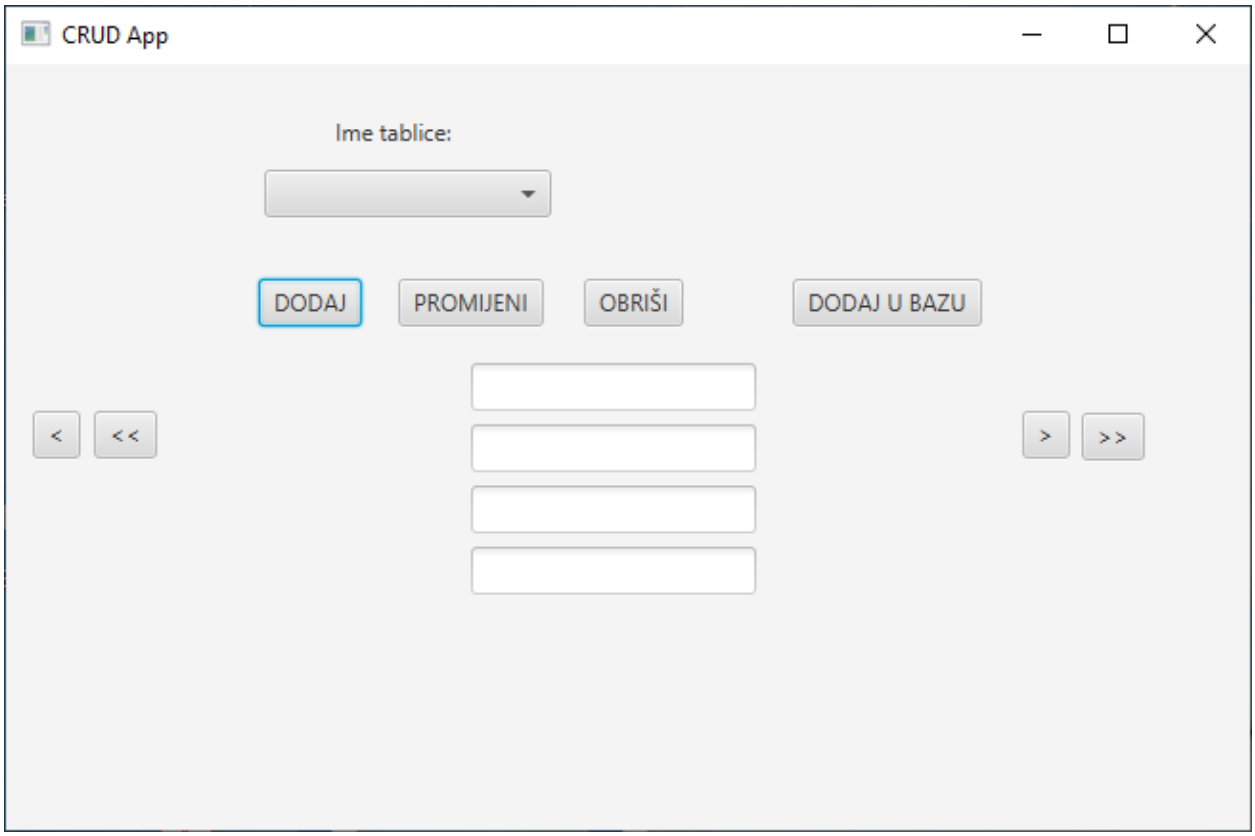
Slika 2.5. Tablica „zaposlenici“

3. PROGRAMSKO RJEŠENJE

Aplikacija za osnovne operacije nad bazom podataka strukturirana je po slici 3.1. Osnovne operacije su dodaj, obriši, učitaj i uredi. Slikom 3.2. prikazan je početni zaslon aplikacije. Nakon svake odrađene akcije u aplikaciji, u donjem dijelu pojavit će se etiketa (engl. *Label*) koja ispisuje posljednju akciju.



Slika 3.1. Struktura aplikacije



Slika 3.2. Početni zaslon aplikacije

Kod aplikacije podijeljen je u tri java klase i jednu fxml datoteku. Java klase su Conn, Controller i Main, dok je fxml datoteka sample.fxml. U klasi Conn nalazi se uspostava konekcije sa bazom podataka. Unutar klase korištena su 2 jdbc (engl. *java database connectivity*) *drivera* koji predstavljaju aplikacijsko programsko sučelje (engl. *application programming interface – API*). Ta dva *drivera* pružaju metode za izvršavanje upita na bazu podataka. Na slici 3.3. prikazan je kod klase Conn.

```
public class Conn {


    public Connection getConnection()
    {
        String userName="root";
        String password="";

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");

            Connection connection = DriverManager.getConnection( url: "jdbc:mysql://localhost/website?useUnicode=true&useJDBC" +
                    "CompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=CET", userName, password);

            return connection;

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

        return null;
    }
}
```

Slika 3.3. Klasa Conn

Klasa Controller je klasa u kojoj je napisan glavni tok podataka aplikacije. U toj klasi napisane su metode koje većinom komuniciraju sa bazom podataka, ali i klase koje poboljšavaju korisničko iskustvo (engl. *user expirience – UX*). Klasa Controller sadrži metode koje predstavljaju osnovne operacije nad bazom podataka. Na slici 3.4. prikazan je kod metode read, koja učitava tablicu koju korisnik odabere klikom na *ComboBox* „cbTableName“.

```
public void read(ActionEvent actionEvent) throws Exception {
    String tableName = cbTableName.getValue();
    currentID = 1;
    col1Textfield.setDisable(true);
    buttonDisable();
    setBlank();
    getLastID();

    List<String> rowList = getRows();

    col1Label.setText(rowList.get(0));
    col2Label.setText(rowList.get(1));
    col3Label.setText(rowList.get(2));
    col4Label.setText(rowList.get(3));

    PreparedStatement ps = connection.prepareStatement( sql: "SELECT * FROM " + tableName
            + " WHERE id=" + currentID + ";");
    ResultSet rs = ps.executeQuery();

    while (rs.next()) {
        col1Textfield.setText(rs.getString( columnIndex: 1));
        col2Textfield.setText(rs.getString( columnIndex: 2));
        col3Textfield.setText(rs.getString( columnIndex: 3));
        col4Textfield.setText(rs.getString( columnIndex: 4));
    }
    currentIDLabel.setText(col1Textfield.getText() + "/" + getLastID());
    actionLabel.setText("Tablica " + tableName + " učitana.");
}
```

Slika 3.4. Metoda read()

Operacija dodaj podijeljena je u dvije metode, „add“ i „addToDB“ što se vidi na slici 3.5. Metoda „addToDB“ poveća trenutni ID za jedan i omogući unos teksta za ostale atribute. Metoda „add“ sprema unesene atribute u tablicu.

```
public void addToDB(ActionEvent actionEvent) throws Exception {
    String tableName = cbTableName.getValue();
    int id = getLastID();
    id++;
    PreparedStatement preparedStatement = connection.prepareStatement(
            sql: "INSERT INTO " + tableName + " VALUES (?, ?, ?, ?);");
    preparedStatement.setInt( parameterIndex: 1, id);
    preparedStatement.setString( parameterIndex: 2, col2Textfield.getText());
    preparedStatement.setString( parameterIndex: 3, col3Textfield.getText());
    preparedStatement.setString( parameterIndex: 4, col4Textfield.getText());
    preparedStatement.executeUpdate();
    actionLabel.setText("Novi ID dodan");
    col1Textfield.setDisable(false);
    addToDbButton.setDisable(true);
    currentID = getLastID();
    currentIDLabel.setText(currentID + "/" + getLastID());
    buttonDisable();
}

public void add(ActionEvent actionEvent) throws Exception {
    setBlank();
    int lastID = getLastID();
    lastID++;
    col1Textfield.setText(Integer.toString(lastID));
    col1Textfield.setDisable(true);
    addToDbButton.setDisable(false);
}
```

Slika 3.5. Metode add() i addToDB()

Posljednje dvije osnovne metode „delete“ i „update“ prikazane su slikom 3.6.

```
public void update(ActionEvent actionEvent) throws Exception {
    String tableName = cbTableName.getValue();
    String sql = "UPDATE " + tableName + " SET " + col2Label.getText() + " = '" + col2Textfield.getText() + "', " +
            col3Label.getText() + " = '" + col3Textfield.getText() +
            "', " + col4Label.getText() + " = '" + col4Textfield.getText() + "' WHERE id= " + currentID + ";";

    PreparedStatement preparedStatement = connection.prepareStatement(sql);
    preparedStatement.executeUpdate();
    actionLabel.setText("Promjena na IDu " + currentID);
}

public void delete(ActionEvent actionEvent) throws Exception {
    String tableName = cbTableName.getValue();
    PreparedStatement preparedStatement = connection.prepareStatement( sql: "DELETE FROM " + tableName +
            " WHERE id=" + currentID + ";");
    preparedStatement.executeUpdate();
    setBlank();
    actionLabel.setText("ID " + currentID + " obrisan");
}
```

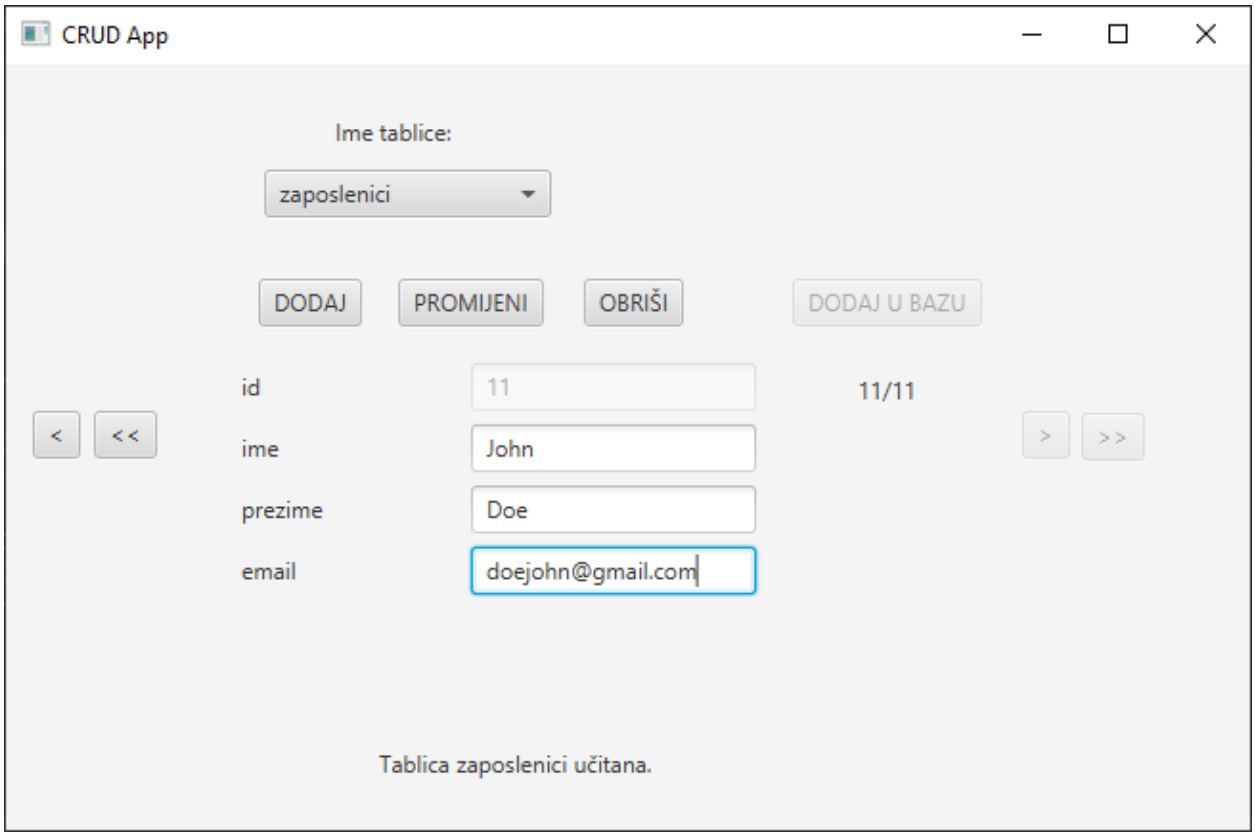
Slika 3.6. Metode update() i delete()

Klasa Main kreira prozor (engl. *window*) i učitava kod napisan u datoteci sample.fxml. Sadržaj datoteke sample.fxml nalazi se na slici 3.7. Za izradu ove aplikacije koristio se AnchorPnae koji omogućava ručno određivanje koordinata za svaki element aplikacije kao što su tipka (engl. *Button*), etiketa (engl. *Label*) i tekstualno polje (engl. *TextField*).

```
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="620.0" xmlns
    <children>
        <Button fx:id="addButton" layoutX="132.0" layoutY="112.0" mnemonicParsing="false" onAction="#add" text="DODAJ" />
        <Button fx:id="updateButton" layoutX="205.0" layoutY="112.0" mnemonicParsing="false" onAction="#update" text="PROMIJENI" />
        <Button fx:id="deleteButton" layoutX="302.0" layoutY="112.0" mnemonicParsing="false" onAction="#delete" text="OBRIŠI" />
        <Button fx:id="previousButton" layoutX="14.0" layoutY="181.0" mnemonicParsing="false" onAction="#previous" text="&lt;" />
        <Button fx:id="previous5Button" layoutX="46.0" layoutY="181.0" mnemonicParsing="false" onAction="#previous5" text="&lt;&lt;" />
        <Button fx:id="next5Button" layoutX="562.0" layoutY="182.0" mnemonicParsing="false" onAction="#next5" text="&gt;&gt;" />
        <Button fx:id="nextButton" layoutX="531.0" layoutY="181.0" mnemonicParsing="false" onAction="#next" text="&gt;" />
        <Button fx:id="addToDbButton" layoutX="411.0" layoutY="112.0" mnemonicParsing="false" onAction="#addToDB" text="DODAJ U BAZU" />

        <Label fx:id="tableNameLabel" layoutX="172.0" layoutY="25.0" prefHeight="20.0" prefWidth="76.0" text="Ime tablice:" />
        <Label fx:id="col1Label" layoutX="123.0" layoutY="156.0" prefHeight="25.0" prefWidth="120.0" text="" />
        <Label fx:id="col2Label" layoutX="123.0" layoutY="188.0" prefHeight="25.0" prefWidth="120.0" text="" />
        <Label fx:id="col3Label" layoutX="123.0" layoutY="220.0" prefHeight="25.0" prefWidth="120.0" text="" />
        <Label fx:id="col4Label" layoutX="123.0" layoutY="252.0" prefHeight="25.0" prefWidth="120.0" text="" />

        <Label fx:id="currentIDLabel" layoutX="445.0" layoutY="160.0" prefHeight="20.0" prefWidth="40.0" text="" />
        <Label fx:id="actionLabel" layoutX="195.0" layoutY="355.0" prefHeight="20.0" prefWidth="150.0" />
        <TextField fx:id="col1Textfield" layoutX="243.0" layoutY="156.0" />
        <TextField fx:id="col2Textfield" layoutX="243.0" layoutY="188.0" />
        <TextField fx:id="col3Textfield" layoutX="243.0" layoutY="220.0" />
        <TextField fx:id="col4Textfield" layoutX="243.0" layoutY="252.0" />


        <ComboBox fx:id="cbTableName" layoutX="135.0" layoutY="55.0" onAction="#read" prefWidth="150.0" />


    </children>
</AnchorPane>
```

Slika 3.7. Datoteka sample.fxml

Kretanje kroz aplikaciju veoma je jednostavno. Nakon učitavanja tablice, otvori se entitet čiji je ID broj 1. Tipke s lijeve i desne strane omogućavaju kretanje po entitetima naprijed i natrag, te za pet unaprijed i za pet unatrag. Kada je broj entiteta koji je preostao do kraja manji od pet, tipka za pet unaprijed se blokira. Isto tako, kada se dođe do posljednjeg entiteta u tablici, tipka za pomak unaprijed se blokira. Slikom 3.8. prikazan je zaslon aplikacije na pretposljednjem entitetu u tablici.



Slika 3.8. Pretposljednji entitet u tablici „zaposlenici“

4. ZAKLJUČAK

Baze podataka imaju svoju uporabu u svim dijelovima društva, od malih baza podataka obiteljskih poljorpivrednih poduzeća, preko faukltetskih baza podataka s tablicama s podacima svih studenata, profesora, kolegija, do ogrmonih državnih baza podataka.

Izradom završnog rada omogućen je još jedan način za prikazivanje tih podataka. Koristeći programski jezik Java i programsko okruženje JavaFX napravljena je aplikacija koja omogućava pojedinačni prikaz svakog entiteta iz određene tablice unutar baze podataka. Aplikacija pruža pregledno kretanje po svim entitetima koje olakšava rad s tablicom i štedi vrijeme.

Aplikacija je strukturirana tako da je moguća dodatna nadogradnja i implementacija novih metoda koje bi pružile još bolje korisničko iskustvo prilikom rada.

LITERATURA

[1] Oracle, dostupno na:
https://www.oracle.com/technetwork/es/java/javase/tech/index.html,
(rujan 2019.)

[2] Wikipedija, JavaFX, dostupno na:
https://en.wikipedia.org/wiki/JavaFX
(rujan 2019.)

[3] Wikipedija, Sun Microsystems, dostupno na:
https://en.wikipedia.org/wiki/Sun_Microsystems
(rujan 2019.)

[4] Wikipedija, Java (programming language), dostupno na:
https://en.wikipedia.org/wiki/Java_(programming_language)
(rujan 2019.)

[5] Wikipedija, FXML, dostupno na:
https://en.wikipedia.org/wiki/FXML
(rujan 2019.)

[6] Wikipedija, MySQL, dostupno na:
https://en.wikipedia.org/wiki/MySQL
(rujan 2019.)

[7] Scottish Qualifications Authority, dostupno na:
https://www.sqa.org.uk/e-learning/MDBS01CD/page_01.htm
(rujan 2019.)

SAŽETAK

Cilj ovog završnog rada bio je razviti Java aplikaciju za četiri osnovne operacije pri radu s tablicom u bazi podataka. Te četiri operacije su dodaj, obriši, uredi i učitaj. Korisnik unosi ime tablice iz baze podataka, te sam navigira kroz pojedine entitete u tablici. Pri izradi aplikacije korišteno je programko orkuženje JavaFX kako bi se dobilo grafičko sučelje. Aplikacija je jednostavna za korištenje i otvorena za proširenje. Uz tekst završnog rada prikazana je struktura i kod, te slike konačnog izgleda aplikacije kako bi se što preciznije prikazalo sve objašnjeno.

Ključne riječi: Java, baza podataka, tablica, aplikacija

JAVA APPLICATION FOR BASIC OPERATIONS ON TABLE IN DATABASE

ABSTRACT

In this bachelors thesis the goal was to develop an Java application for four basic operations on table in database. Those four operations are create, read, update and delete. User inputs name of the table in database and is able to navigate through entities. JavaFX was used during the development so we would get a graphic interface. Application is simple to use and is open for upgrades. Through the thesis there is shown text, structure, code, and images of final look of this application so it would most precisely describe everything written.

Keyworsd: Java, database, table, application

ŽIVOTOPIS

Dario Đurić rođen je 9. kolovoza 1996. godine u Slavonskom Brodu. Odrastao je u Donjoj Mahali u Bosni i Hercegovini. Školovanje započinje 2003. godine upisivanjem Osnovne škole Orašje u Orašju. Uz osnovnu školu završio je i osnovnu glazbenu školu u Orašju. Nakon završene osnovne škole upisuje Opću gimnaziju u Županji. Godine 2015. upisuje Fakultet elektrotehnike, računarstva, i informacijskih tehnologija u Osijeku. Tijekom 2018. godine završio je Backend developer akademiju koju je uz FERIT provodila tvrka UHP Digital.

PRILOZI

Na CD-u prilažem cijeli rad u .docx i .pdf formatu, aplikaciju u .jar formatu, te bazu podataka u .sql formatu