GUI aplikacija za vođenje teretane

Lneniček, Mihael

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:277932

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-18**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Stručni studij

GUI APLIKACIJA ZA VOĐENJE TERETANE

Završni rad

Mihael Lneniček

Osijek, 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1S: Obrazac za imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

Osijek, 21.09.2020.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

Ime i prezime studenta:	Mihael Lneniček
Studij, smjer:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	AI4564, 24.09.2019.
OIB studenta:	66554638255
Mentor:	Robert Šojo
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Marina Peko
Član Povjerenstva 1:	Robert Šojo
Član Povjerenstva 2:	dr.sc. Ivana Hartmann-Tolić
Naslov završnog rada:	GUI aplikacija za vođenje teretane
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada	Kreiranje GUI aplikacije za vođenje teretane kojom se voditelju teretane i treneru omogućava kompletan uvid u podatke o članovima teretane. Voditelj treba biti glavni administrator kojemu je omogućeno kreiranje trenera i članova, uvid u posjećenost, status članova, članarine, statistika članova na mjesečnoj bazi, zauzeće trenera i kontroli svojih radnih sati. Trener treba imati uvid u status članova, članarine, dogovorene termine, te kontrola radnih sati. Student može odabrati samostalno tehnologije koje će koristiti u procesu kreiranja aplikacije. Opisati tehnologije u procesu izrade aplikacije, opisati pojedine dijelove aplikacije, te kreirati potpuno funkcionalnu aplikaciju.
Prijedlog ocjene pismenog dijela ispita (završnog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	21.09.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 04.10.2020.

Ime i prezime studenta:	Mihael Lneniček
Studij:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	AI4564, 24.09.2019.
Turnitin podudaranje [%]:	12

Ovom izjavom izjavljujem da je rad pod nazivom **: GUI aplikacija za vođenje teretane**

izrađen pod vodstvom mentora Robert Šojo

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD 1
1.1. Zadatak završnog rada 1
2. KORIŠTENE TEHNOLOGIJE ZA IZRADU APLIKACIJE 2
2.1. Java 2
2.2. JavaFx 3
2.3. SQL 3
2.4. Eclipse IDE 5
3. FUNKCIONALNOST APLIKACIJE 6
3.1. Baza podataka 6
3.2. Funkcionalnost prijave 8
3.3. Funkcionalnost voditelja 11
3.4. Funkcionalnost trenera 17
4. RAD S APLIKACIJOM 20
4.1. Prijava 20
4.2. Voditelj 21
4.3. Trener 27
5. ZAKLJUČAK 29
LITERATURA 30
SAŽETAK 31
ABSTRACT 32
ŽIVOTOPIS 33
PRILOZI 34

1. UVOD

U ovom radu pokazat će se rad GUI (engl. *Graphical User Interface*) aplikacije za vođenje teretane. Svrha aplikacije je da voditelju teretane i treneru da kompletan uvid o podacima članova teretane. U aplikaciji najveće ovlasti ima voditelj teretane te je on glavni administrator. Omogućeno mu je kreiranje trenera te članova teretane, ima uvid u posjećenost teretane, status članova, članarine, statistiku članova na mjesečnoj bazi, zauzeća trenera te u kontrolu svojih radnih sati. Treneru omogućava uvid u status članova, članarine, dogovorene termine te kontrolu svojih radnih sati.

Tehnologije koje su se koristile za izradu ove aplikacije opisane su u drugom poglavlju. U trećem poglavlju opisane su funkcionalnosti aplikacije. U četvrtom poglavlju objašnjeno je kako se koristi s aplikacijom.

1.1. Zadatak završnog rada

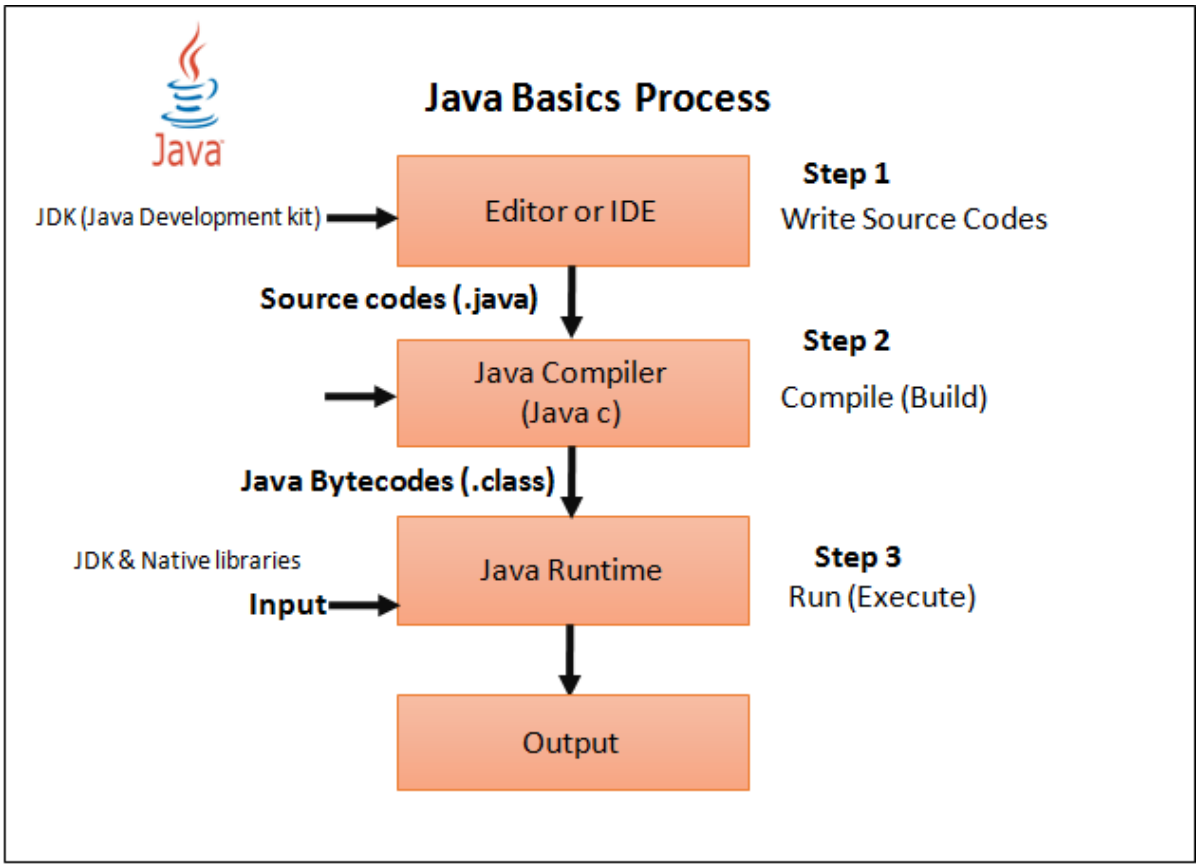
Zadatak završnog rada je kreiranje JavaFx GUI aplikacije koja će olakšati rad u vođenju teretane voditelju teretane te treneru. Aplikacija omogućava voditelju teretane kreiranje trenera i člana te sve informacije o trenerima i članovima teretane. Treneru omogućava uvid u kontrolu svojih radnih sati, zakazanim terminima te statuse članova.

2. KORIŠTENE TEHNOLOGIJE ZA IZRADU APLIKACIJE

U ovom poglavlju opisane su sve tehnologije koje su korištene za izradu aplikacije.

2.1. Java

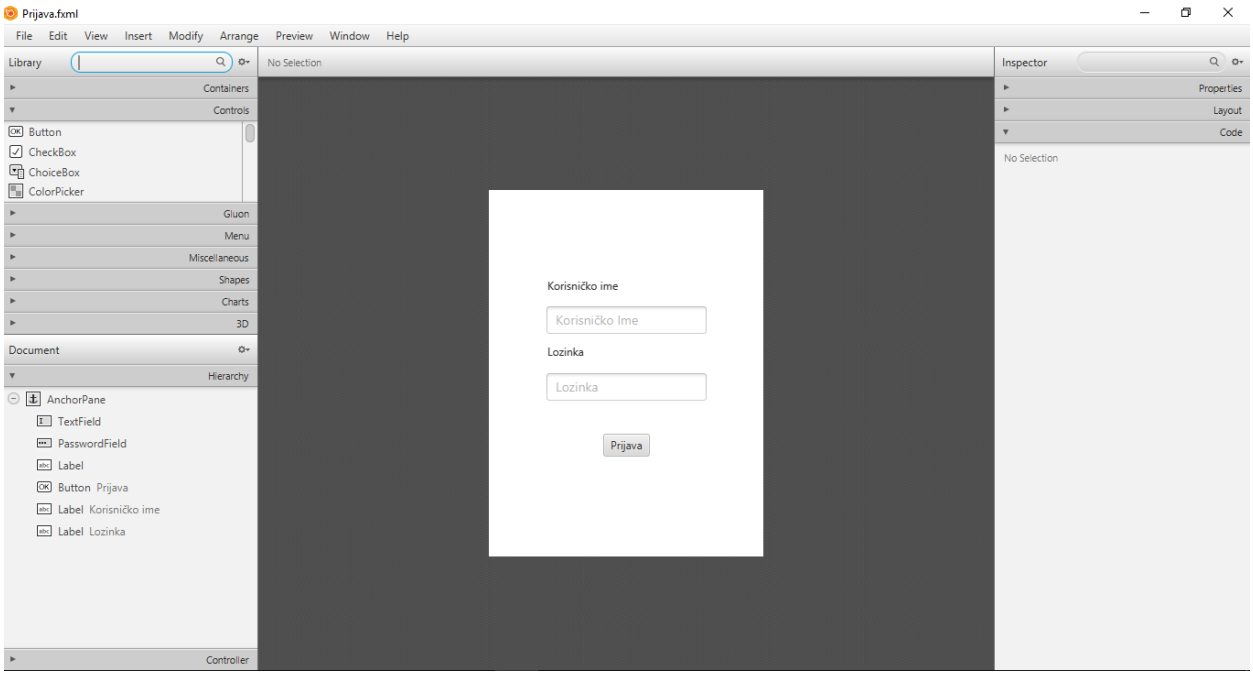
Programski jezik Java nastala je 1995. godine, razvio ju je James Gosling u tvrtki Sun Microsystems. Java je sada u vlasništvu Oracle Corporation koja ju je otkupila 27. Siječnja 2010. Već dugi niz godina je jedan od najpopularnijih programskih jezika. Java je objektno-orijentirani programski jezik čija je namjena izvršavanje koda na više platforma. Java ima sličnu sintaksu programskom jeziku C++ koji je također objektno-orijentiran, ali su drugačiji jezici. Npr. programski jezik C++ programerima omogućuje implementaciju preopterećenja operatora dok kod Java programskog jezika to nije moguće. Aplikacije napisane u programskom jeziku Java mogu se pokrenuti na bilo kojem računalu, bez obzira na konfiguraciju računala te koji imaju instaliran JVM (engl. *Java Virtual Maschine*) u operacijskom sustavu. JVM za rad koristi bajt kod, koji se kompilira iz izvornog koda [1]. Proces stvaranja izvršnog programa prikazan je na slici 2.1 [2].



Slika 2.1. *Java osnovni proces.*

2.2. JavaFx

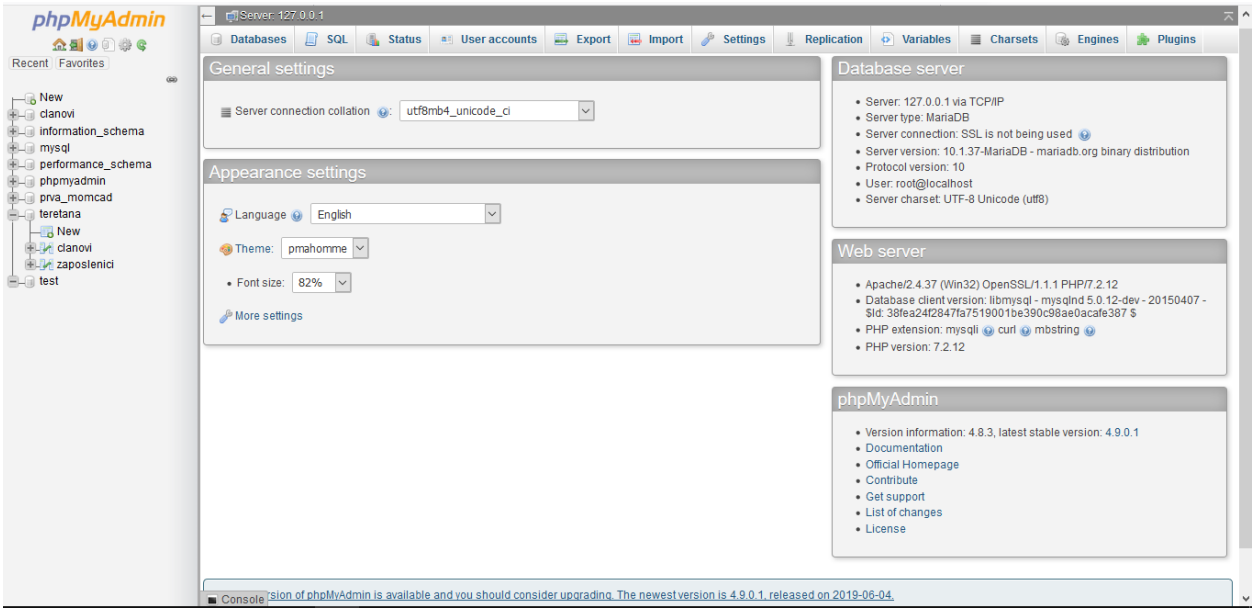
JavaFx je dodatak za programski jezik Java koji služi za kreiranje aplikacije s grafičkim korisničkim sučeljem. *JavaFx* aplikacija pomoću JVM-a može se pokrenuti na bilo kojoj platformi. Zamjena je za *Swing* koji se koristio za razvoj grafičkog sučelja programskog jezika Jave. Za sastavljanje GUI-a koristi se FXML. FXML je jezik koji je baziran na XML-u (engl. *eXtensible Markup Language*) a omogućuje kreiranje grafičkog sučelja koji koristi Java komponente bez pisanja koda [3]. Program *SceneBuilder* je grafičko sučelje kojim se lakše kreiraju i pozicioniraju prezentacijski elementi. Sučelje programa *SceneBuilder* prikazano je na slici 2.2.



Slika 2.2. *Sučelje programa SceneBuilder.*

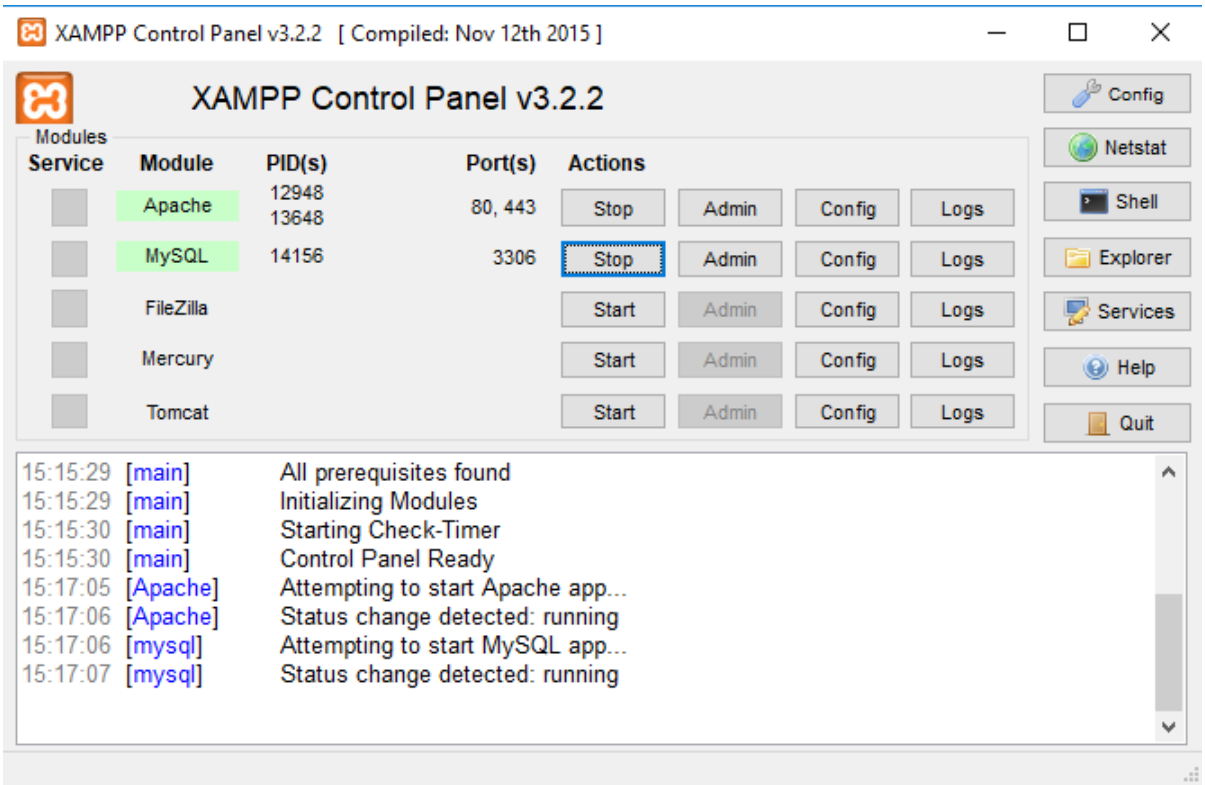
2.3. SQL

SQL (engl. *Structured Query Language*) je standardni programski jezik za relacijske baze podataka. SQL je 1987. godine postao standard međunarodne organizacije za standardizaciju (engl. *International Organization for Standardization;* kratica ISO). SQL omogućuje izvršavati upite (engl. *Query*) u bazu podataka. S upitima mogu se kreirati baze podataka, dodavati nove tablice u bazu podataka, odabrati, dodavati, brisati ili ažurirati neki podatak iz tablice podataka [4]. PhpMyAdmin je softverski alat kojim se upravlja s bazom podataka. Sučelje alata phpMyAdmin prikazano je na slici 2.3.



Slika 2.3. *Sučelje alata phpMyAdmin.*

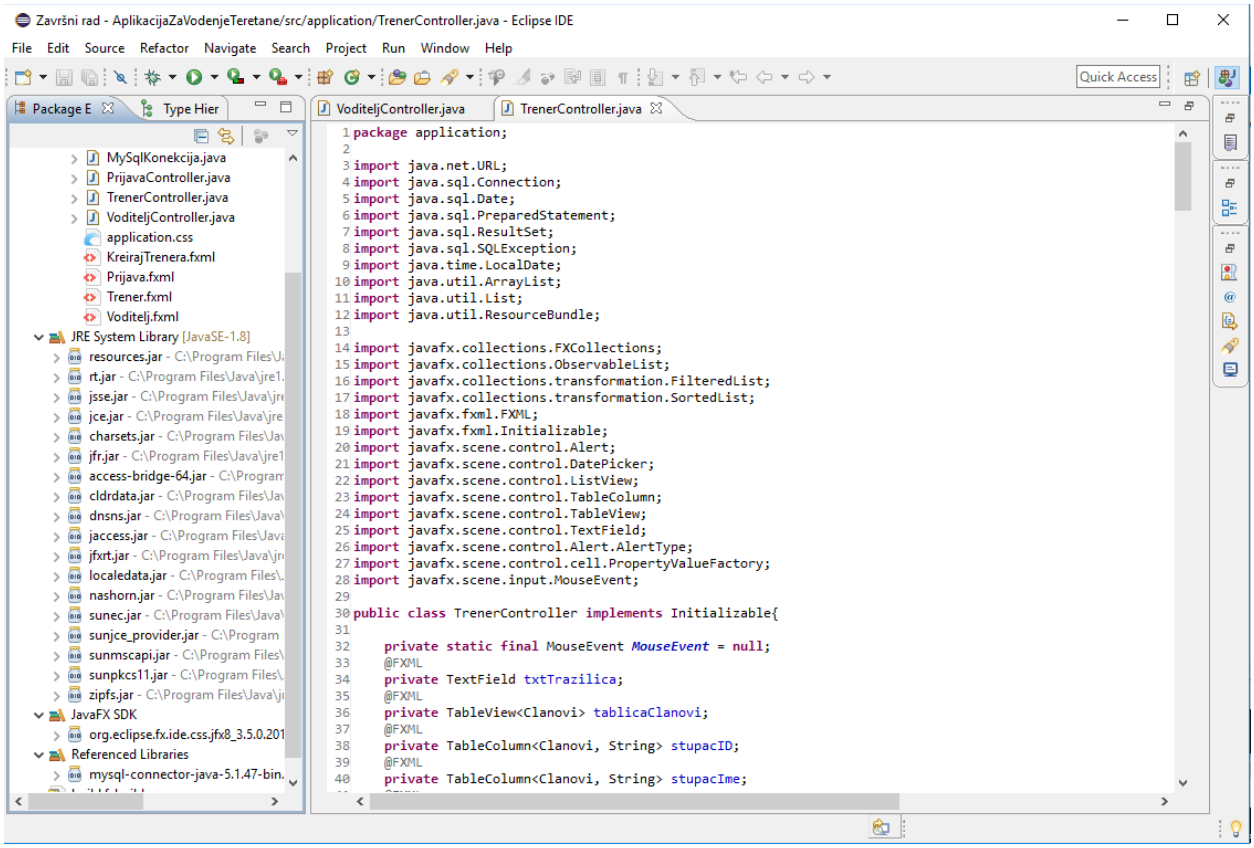
Za izradu lokalnog servera korišten je serverski paket za izradu servera XAMPP. Sučelje XAMPP-a prikazano je na slici 2.4.



Slika 2.4. *Sučelje XAMPP-a.*

2.4. Eclipse IDE

Eclipse IDE je platforma koja je većim dijelom napisana u Javi. Omogućuje razvijanje aplikacija u različitim programskim jezicima, kao što su Java, C, C++, Python, PHP i mnogi drugi. Već dugo godina je najpopularnije Java integrirano razvojno okruženje. Razvio ga je IBM 2001. godine, cilj IBM-ovog projekta bio je zasjeniti Microsoftov Visual Studio pa je zbog toga i dobio takav naziv. Uz *Eclipse IDE* za razvoj Java aplikacija neki od poznatih alata su *NetBeans* i *IntelliJ IDEA* [5]. Za izradu aplikacije korištena je inačica Eclipse Java 2019-03. Sučelje inačice Eclipse Java 2019-03 prikazano je na slici 2.5.



Slika 2.5. *Sučelje inačice Eclipse Java 2019-03.*

3. FUNKCIONALNOST APLIKACIJE

U ovom poglavlju opisana je struktura baze podataka te funkcionalnost funkcija u aplikaciji.

3.1. Baza podataka

Baza podataka sastoji se od pet tablica: *„clanovi“, „dolasciuteretanu“, „radnisati“, „termini“ i „zaposlenici“*. Tablica *„clanovi“* sadrži ime i prezime člana, identifikacijski broj, godine, visinu, težinu, spol, broj dolazaka u tjednu, datum kada je plaćena članarina te ako je aktivan ili ne. Primarni ključ u tablici je *„id“*. Na slici 3.1. prikazana je struktura tablice *„clanovi“*.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**id**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
☐	2	**ime**	varchar(100)	utf8_general_ci		No	*None*			Change Drop More
☐	3	**prezime**	varchar(100)	utf32_general_ci		No	*None*			Change Drop More
☐	4	**godine**	smallint(6)			No	*None*			Change Drop More
☐	5	**visina**	float			No	*None*			Change Drop More
☐	6	**tezina**	float			No	*None*			Change Drop More
☐	7	**spol**	varchar(50)	utf8_general_ci		No	*None*			Change Drop More
☐	8	**dolasci**	varchar(50)	utf8_general_ci		No	*None*			Change Drop More
☐	9	**datum**	text	utf8_general_ci		No	*None*			Change Drop More
☐	10	**aktivan**	varchar(50)	utf8_general_ci		No	*None*			Change Drop More

Slika 3.1. *Struktura tablice „clanovi“.*

Tablica *„dolasciuteretanu“* sadrži identifikator dolaska, identifikator člana koji je došao u teretanu, datum dolaska u teretanu. Primarni ključ tablice je *„idDolaska“* a strani ključ je *„id“*, koji je primarni ključ tablice *„clanovi“*. Na slici 3.2. prikazana je struktura tablice *„dolasciuteretanu“*.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**idDolaska**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
☐	2	**id**	int(11)			No	*None*			Change Drop More
☐	3	**datum**	date			No	*None*			Change Drop More

S1ika 3.2. *Struktura tablice „dolasciuteretanu“.*

Tablica *„zaposlenici“* sadrži identifikator, ime i prezime zaposlenika, korisničko ime i lozinku koji su potrebni za prijavu u aplikaciju te vrijednost koja govori je li zaposlenik voditelj te mu tako

pruža veće mogućnosti u aplikaciji. Primarni ključ u tablici je *„id“*. Na slici 3.3. prikazana je struktura tablice *„zaposlenici“*.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
☐	2	ime	varchar(100)	utf8_general_ci		No	None			Change Drop More
☐	3	prezime	varchar(100)	utf8_general_ci		No	None			Change Drop More
☐	4	korisnickolme	varchar(100)	utf8_general_ci		No	None			Change Drop More
☐	5	lozinka	varchar(100)	utf8_general_ci		No	None			Change Drop More
☐	6	voditelj	tinyint(1)			No	None			Change Drop More

Slika 3.3. *Struktura tablice „zaposlenici“.*

Tablica *„radnisati“* sadrži identifikator, identifikator od zaposlenika za kojeg se odnose ti radni sati te radno vrijeme zaposlenika za svaki dan u tjednu. Primarni ključ tablice je *„id“* dok je *„idZaposlenika“* strani ključ (primarni ključ u tablici *„zaposlenici“* pod nazivom *„id“).* Na slici 3.4. prikazana je struktura tablice *„radnisati“*.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
☐	2	idZaposlenika	int(11)			No	None			Change Drop More
☐	3	pon	varchar(50)	utf8_general_ci		No	None			Change Drop More
☐	4	uto	varchar(50)	utf8_general_ci		No	None			Change Drop More
☐	5	sri	varchar(50)	utf8_general_ci		No	None			Change Drop More
☐	6	cet	varchar(50)	utf8_general_ci		No	None			Change Drop More
☐	7	pet	varchar(50)	utf8_general_ci		No	None			Change Drop More
☐	8	sub	varchar(50)	utf8_general_ci		No	None			Change Drop More
☐	9	ned	varchar(50)	utf8_general_ci		No	None			Change Drop More

Slika. 3.4. *Struktura tablice „radnisati“.*

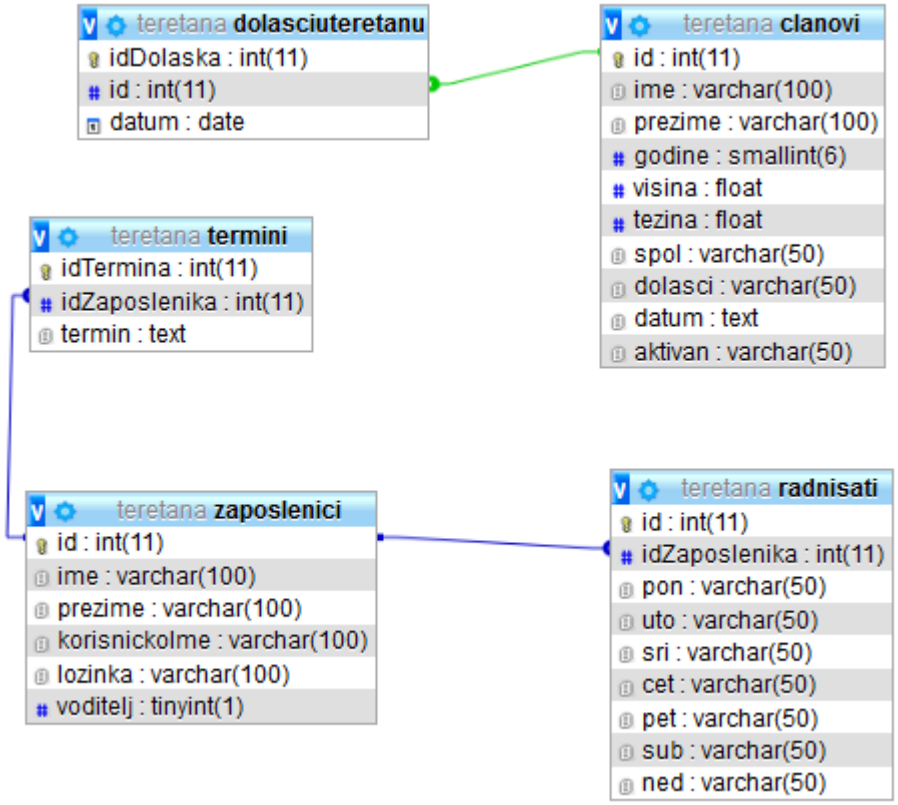
Tablica *„termini“* sadrži identifikator, identifikator od zaposlenika za kojeg se odnose ti termini te zakazane termine trenera. Primarni ključ je *„idTermina“* a „*idZaposlenika*„ je strani ključ (primarni ključ u tablici *„zaposlenici“* pod nazivom *„id“*). Na slici 3.5. prikazana je struktura tablice *„termini“*.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	idTermina	int(11)			No	None		AUTO_INCREMENT	Change Drop More
☐	2	idZaposlenika	int(11)			No	None			Change Drop More
☐	3	termin	text	utf8_general_ci		No	None			Change Drop More

Slika 3.5. *Struktura tablice „termini“.*

Slike od 3.1. do 3.5. prikazuju primarne ključeve koji imaju svojstva „AUTO_INCREMENT“. To znači da će se prilikom unosa u bazu automatski povećati vrijednost

identifikacijskog broja za jedan u odnosu na prijašnji. To pomaže korisnicima aplikacije jer ne moraju pamtiti koje su bile posljednje vrijednosti identifikatora te tako smanjuju mogućnost ljudske pogreške. Na slici 3.6. prikazan je E-R dijagram baze podataka.



Slika 3.6. *E-R dijagram baze podataka.*

3.2. Funkcionalnost prijave

Kako bi koristili bazu podataka koja je opisana u prošlom potpoglavlju aplikaciju je potrebno spojiti s bazom podataka. Na slici 3.7. prikazano je povezivanje aplikacije s bazom podataka.

```
package application;
import java.sql.*;

public class MySqlKonekcija {
    public static Connection Konektor() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/teretana?"
                    + "useUnicode=yes&characterEncoding=UTF-8", "root", "");
            return connection;
        } catch (Exception e) {
            return null;
        }
    }
}
```

Slika 3.7. *Programski kôd „MySqlKonekcija.java“ klase.*

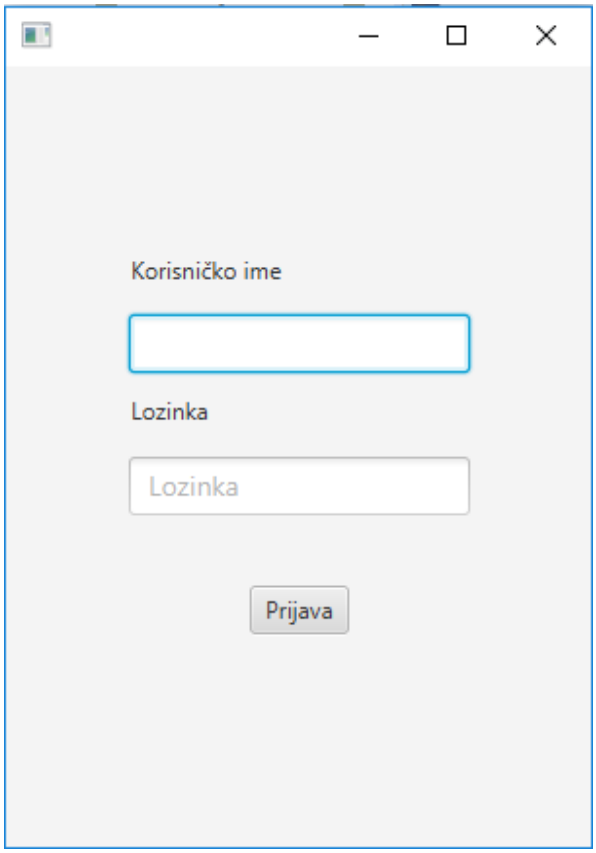
Metoda *„getConnection“* omogućava otvaranje veze s bazom podataka koja s metodom *„forName“* uspostavlja vezu s JDBC (engl. *Java DataBase Connectivity*) upravljačkim programom. Metoda *„DriverManager.getConnection“* šalje relativni put od baze podataka koja se koristi, koja se u ovom slučaju nalazi u korijenskom direktoriju (engl. *root*). Za pokretanje aplikacije koristi se *„Main.java“* klasa. Na slici 3.8. prikazan je programski kôd *„Main.java“* klase.

```
package application;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.fxml.FXMLLoader;

public class Main extends Application {
    @Override
    public void start(Stage primaryStage) {
        try {
            Parent root = FXMLLoader.load(getClass().getResource("/application/Prijava.fxml"));
            Scene scene = new Scene(root);
            scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch(Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Slika 3.8. *Programski kôd „Main.java“ klase*

Za pokretanje aplikacije koristi se preopterećena metoda *„start()“* u kojoj se pomoću metode *„FXMLLoader.load“* učitava dodani fxml dokument koji se sprema kao objekt u referencu

klase *„Parent“*. Pomoću reference klase *„Parent“* postavlja se scena te nakon scene i pozornica aplikacije. Na slici 3.9. prikazan je učitani fxml dokument nakon izvršavanja metode *„start()“*.



Slika 3.9. *Sučelje prijave u aplikaciju*

Za prijavu se koriste podaci iz baze podataka. Slika 3.10. prikazuje programski kôd za dobivanje podataka iz baze koji su potrebni za prijavljivanje zaposlenika.

```
public boolean Prijavljen(String korisnik, String lozinka, boolean voditelj) throws SQLException {
    PreparedStatement preparedStatement = null;
    ResultSet resultSet = null;
    String query = "SELECT * FROM zaposlenici WHERE korisnickoIme = ? AND lozinka = ? AND voditelj = ?";
    try {
        preparedStatement = konekcija.prepareStatement(query);
        preparedStatement.setString(1, korisnik);
        preparedStatement.setString(2, lozinka);
        preparedStatement.setBoolean(3, voditelj);
        resultSet = preparedStatement.executeQuery();
        if(resultSet.next()) {
            return true;
        }else {
            return false;
        }
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }finally {
        preparedStatement.close();
        resultSet.close();
    }
}
```

Slika 3.10. *Programski kôd metode „Prijavljen()“.*

SQL upitom u bazu dohvaćamo sve podatke iz tablice *„zaposlenici“*. Rezultat upita sprema se u varijablu *„resultSet“*. Ti dobiveni podaci se provjeravaju s podacima koje unese zaposlenik. Na slici 3.11. prikazan je programski kôd prijave u aplikaciju.

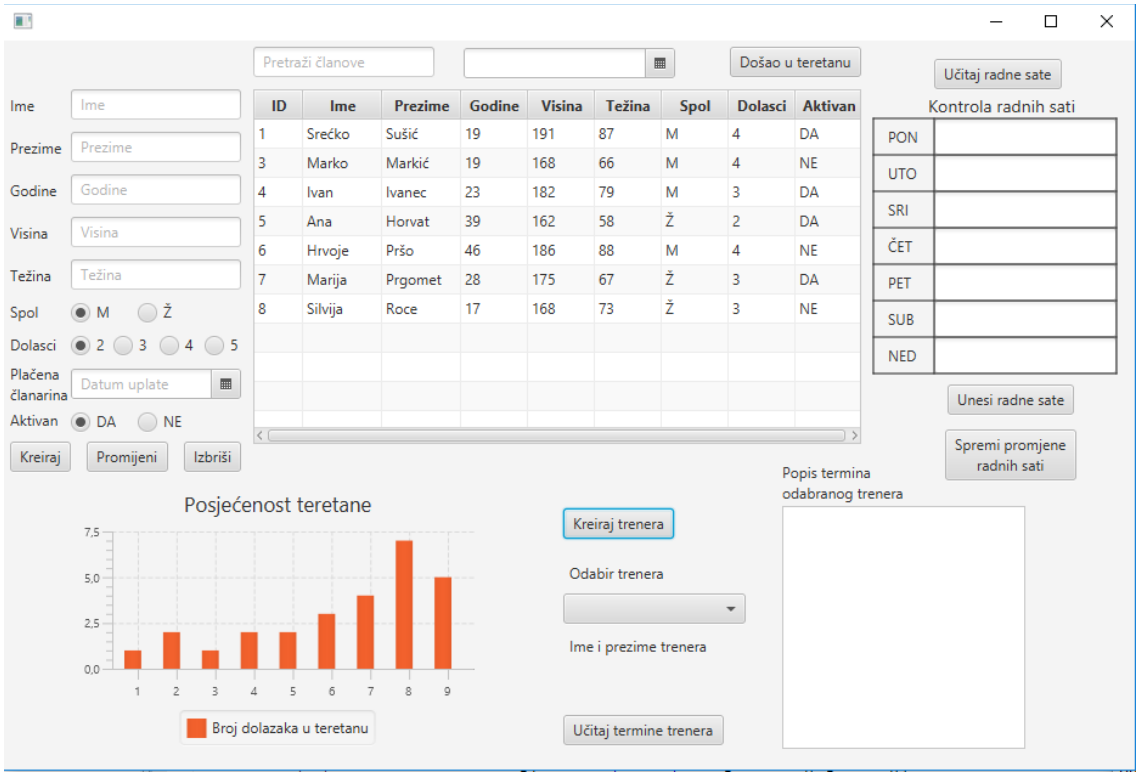
```
public void Prijava(ActionEvent event) {
    try {
        if(modelPrijave.Prijavljen(txtKorisnickoIme.getText(), txtLozinka.getText(), voditelj)) {
            ((Node)event.getSource()).getScene().getWindow().hide();
            Stage primaryStage = new Stage();
            FXMLLoader loader = new FXMLLoader();
            Pane root = loader.load(getClass().getResource("/application/Voditelj.fxml").openStream());
            Scene scene = new Scene(root);
            scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
            primaryStage.setScene(scene);
            primaryStage.show();
        }else if((modelPrijave.Prijavljen(txtKorisnickoIme.getText(), txtLozinka.getText(), false))) {
            ((Node)event.getSource()).getScene().getWindow().hide();
            Stage primaryStage = new Stage();
            FXMLLoader loader = new FXMLLoader();
            Pane root = loader.load(getClass().getResource("/application/Trener.fxml").openStream());
            TrenerController trenerController = (TrenerController)loader.getController();
            trenerController.DajId(modelPrijave.IdTrenera(txtKorisnickoIme.getText(), txtLozinka.getText()));
            Scene scene = new Scene(root);
            scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
            primaryStage.setScene(scene);
            primaryStage.show();
        }else {
            lblStatus.setText("Neispravno korisničko ime ili lozinka");
        }
    } catch (SQLException e) {
        lblStatus.setText("Neispravno korisničko ime ili lozinka");
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Slika 3.11. *Programski kôd metode „Prijava“.*

Provjeravaju se uneseni podaci za korisničko ime i lozinku, te se provjerava vrijednost za voditelja. Ako se korisničko ime i lozinka koji su uneseni nalaze u bazi te ako je vrijednost istinita (engl. *true*) zatvara se obrazac za prijavu te se učitava dio aplikacije koji je namijenjen za voditelja. A ako se uneseni podaci za korisničko ime i lozinku nalaze u bazi te ako je vrijednost lažna (engl. *false*) zatvara se obrazac za prijavu te se učitava dio aplikacije za trenere. Za krivo unesene podatke ispisuje se *„Neispravno korisničko ime ili lozinka“*.

3.3. Funkcionalnost voditelja

Na slici 3.11. prikazano je sučelje aplikacije kada je prijavljen voditelj teretane.



Slika 3.11 *Sučelje aplikacije za voditelja.*

Na slici 3.12. i 3.13. prikazano je kako se unosi novi član u bazu podataka.

```
public void DodavanjeClana() throws SQLException {
    PreparedStatement preparedStatement = null;
    String ime = txtIme.getText();
    String prezime = txtPrezime.getText();
    String godine = txtGodine.getText();
    String visina = txtVisina.getText();
    String tezina = txtTezina.getText();
    String spol = "";
    if(rbMusko.isSelected()) {
        spol = rbMusko.getText();
    }
    else {
        spol = rbZensko.getText();
    }
    String dolasci = "";
    if(rb2.isSelected()) {
        dolasci = rb2.getText();
    }
    if(rb3.isSelected()) {
        dolasci = rb3.getText();
    }
    if(rb4.isSelected()) {
        dolasci = rb4.getText();
    }
    if(rb5.isSelected()) {
        dolasci = rb5.getText();
    }
    String aktivan = "";
    if(rbDa.isSelected()) {
        aktivan = rbDa.getText();
    }
    else {
        aktivan = rbNe.getText();
    }
```

Slika 3.12. *Programski kôd „DodavanjeClana()“ - prvi dio.*

```
    if(ProvjeraPolja()) {
    try {
        String query = "INSERT INTO clanovi (ime, prezime, godine, visina, tezina, spol, dolasci, datum, aktivan) "
                + "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ? )";
        preparedStatement = MySqlKonekcija.Konektor().prepareStatement(query);
        preparedStatement.setString(1, ime);
        preparedStatement.setString(2, prezime);
        preparedStatement.setString(3, godine);
        preparedStatement.setString(4, visina);
        preparedStatement.setString(5, tezina);
        preparedStatement.setString(6, spol);
        preparedStatement.setString(7, dolasci);
        preparedStatement.setString(8, this.dpDatumUplate.getEditor().getText());
        preparedStatement.setString(9, aktivan);
        OcistiPolja();

    } catch (SQLException e) {

        e.printStackTrace();
    }
    finally {
        preparedStatement.execute();
        preparedStatement.close();
    }
    OsvjeziTablicu();
    }

}
```

Slika 3.13. *Programski kôd „DodavanjeClana()“ - drugi dio.*

Voditelj teretane unosi potrebne podatke za unos novog člana, ako neki podaci nisu unijeti poziva se metoda *„ProvjeraPolja()“* koja upozorava korisnika koji je podatak zaboravio unijeti. A ako su uneseni svi podaci koji su potrebni za unos novog člana, SQL upitom podaci se dodaju u bazu podataka. Nakon dodanog novog člana poziva se metoda *„OcistiPolja“* koja briše sve podatke koji su uneseni te postavlja gumbove za odabir na početnu vrijednost. Slika 3.14. prikazuje programski kôd metode *„OcistiPolja()“*. Nakon unosa novog člana u bazu poziva se metoda *„OsvjeziTablicu“* koja osvježava tablicu u kojoj se nalaze podaci od članova teretane. Slika 3.15. prikazuje programski kôd metode *„OsvjeziTablicu“*.

```
public void OcistiPolja() {
    txtId.clear();
    txtIme.clear();
    txtPrezime.clear();
    txtGodine.clear();
    txtVisina.clear();
    txtTezina.clear();
    rbMusko.setSelected(true);
    rbZensko.setSelected(false);
    rb2.setSelected(true);
    rb3.setSelected(false);
    rb4.setSelected(false);
    rb5.setSelected(false);
    dpDatumUplate.setValue(null);
    dpDatumUplate.getEditor().setText(null);
    rbDa.setSelected(true);
    rbNe.setSelected(false);
}
```

Slika 3.14. *Programski kôd metode „OcistiPolja()“.*

```
public void OsvjeziTablicu() {
    podaci.clear();
    try {
        Connection konekcija = MySqlKonekcija.Konektor();
        String query = "SELECT * FROM clanovi";
        ResultSet resultSet=konekcija.createStatement().executeQuery(query);
        while(resultSet.next()) {
            podaci.add(new Clanovi(
                resultSet.getString(1),
                resultSet.getString(2),
                resultSet.getString(3),
                resultSet.getString(4),
                resultSet.getString(5),
                resultSet.getString(6),
                resultSet.getString(7),
                resultSet.getString(8),
                resultSet.getString(10)
                ));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    stupacID.setCellValueFactory(new PropertyValueFactory<>("id"));
    stupacIme.setCellValueFactory(new PropertyValueFactory<>("ime"));
    stupacPrezime.setCellValueFactory(new PropertyValueFactory<>("prezime"));
    stupacGodine.setCellValueFactory(new PropertyValueFactory<>("godine"));
    stupacVisina.setCellValueFactory(new PropertyValueFactory<>("visina"));
    stupacTezina.setCellValueFactory(new PropertyValueFactory<>("tezina"));
    stupacSpol.setCellValueFactory(new PropertyValueFactory<>("spol"));
    stupacDolasci.setCellValueFactory(new PropertyValueFactory<>("dolasci"));
    stupacAktivan.setCellValueFactory(new PropertyValueFactory<>("aktivan"));

    tablicaClanovi.setItems(podaci);
```

Slika 3.15. *Programski kôd metode „OsvjeziTablicu()“.*

Metoda *„OsvjeziTablicu()“* briše sve podatke koji se nalaze u tablici te SQL upitom dohvaća sve podatke koji se nalaze u bazi podataka u tablici *„clanovi“*. Nakon što dohvati sve podatke iz baze, postavlja podatke u tablicu po stupcima gdje koji podatak pripada. U tablici s podacima o članovima moguće je odabiranje člana. Slika 3.16. prikazuje programski kôd metode za odabiranje željenog člana. Odabranom članu teretane moguće je promijeniti trenutne informacije, na slici 3.17. prikazano je kako se mijenjaju informacije o članu teretane. Također odabranog člana moguće je i izbrisati, na slici 3.18. prikazano je kako se briše član.

```
public void getSelected(MouseEvent event)throws SQLException {
    try {
        Clanovi clanovi = (Clanovi)tablicaClanovi.getSelectionModel().getSelectedItem();
        String query = "SELECT * FROM clanovi WHERE id=?";
        PreparedStatement preparedStatement = MySqlKonekcija.Konektor().prepareStatement(query);
        preparedStatement.setString(1, clanovi.getId());
        ResultSet resultSet = preparedStatement.executeQuery();
        while(resultSet.next()) {
            txtId.setText(resultSet.getString(1));
            txtIme.setText(resultSet.getString(2));
            txtPrezime.setText(resultSet.getString(3));
            txtGodine.setText(resultSet.getString(4));
            txtVisina.setText(resultSet.getString(5));
            txtTezina.setText(resultSet.getString(6));
            if("M".equals(resultSet.getString(7))) {
                rbMusko.setSelected(true);
            }else if("Ž".equals(resultSet.getString(7))) {
                rbZensko.setSelected(true);
            }else {
                rbMusko.setSelected(false);
                rbZensko.setSelected(false);
            }
            if("2".equals(resultSet.getString(8))) {
                rb2.setSelected(true);
            }else if("3".equals(resultSet.getString(8))) {
                rb3.setSelected(true);
            }else if("4".equals(resultSet.getString(8))) {
                rb4.setSelected(true);
            }else if("5".equals(resultSet.getString(8))) {
                rb5.setSelected(true);
            }else {
                rb2.setSelected(false);
                rb3.setSelected(false);
                rb4.setSelected(false);
                rb5.setSelected(false);
            }
            this.dpDatumUplate.getEditor().setText(resultSet.getString(9));
            if("DA".equals(resultSet.getString(10))) {
                rbDa.setSelected(true);
            }else if("NE".equals(resultSet.getString(10))) {
                rbNe.setSelected(true);
            }else {
                rbDa.setSelected(false);
                rbNe.setSelected(false);
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

}
```

Slika 3.16. *Programski kôd metode „getSelected()“.*

Klikom mišem na željenog člana, ispisuju se svi podaci od tog člana koji se dobivaju SQL upitom, prepoznavanje koje podatke treba ispisati radi se pomoću identifikatora člana.

```
public void Promijeni() throws SQLException{
    PreparedStatement preparedStatement = null;
    String spol = "";
    if(rbMusko.isSelected()) {
        spol = rbMusko.getText();
    }
    else {
        spol = rbZensko.getText();
    }
    String dolasci = "";
    if(rb2.isSelected()) {
        dolasci = rb2.getText();
    }
    if(rb3.isSelected()) {
        dolasci = rb3.getText();
    }
    if(rb4.isSelected()) {
        dolasci = rb4.getText();
    }
    if(rb5.isSelected()) {
        dolasci = rb5.getText();
    }
    String aktivan = "";
    if(rbDa.isSelected()) {
        aktivan = rbDa.getText();
    }
    else {
        aktivan = rbNe.getText();
    }
    try {
        String query = "UPDATE clanovi SET id=?, ime=?, prezime=?, godine=?, visina=?, tezina=?, "
                + "spol=?, dolasci=?, datum=?, aktivan=? WHERE id='"+txtId.getText()+"'";
        preparedStatement = MySqlKonekcija.Konektor().prepareStatement(query);
        preparedStatement.setString(1, txtId.getText());
        preparedStatement.setString(2, txtIme.getText());
        preparedStatement.setString(3, txtPrezime.getText());
        preparedStatement.setString(4, txtGodine.getText());
        preparedStatement.setString(5, txtVisina.getText());
        preparedStatement.setString(6, txtTezina.getText());
        preparedStatement.setString(7, spol);
        preparedStatement.setString(8, dolasci);
        preparedStatement.setString(9, ((TextField)dpDatumUplate.getEditor()).getText());
        preparedStatement.setString(10, aktivan);
        preparedStatement.execute();
        OcistiPolja();
    } catch(Exception e) {
        e.printStackTrace();
    }
    OsvjeziTablicu();
}
```

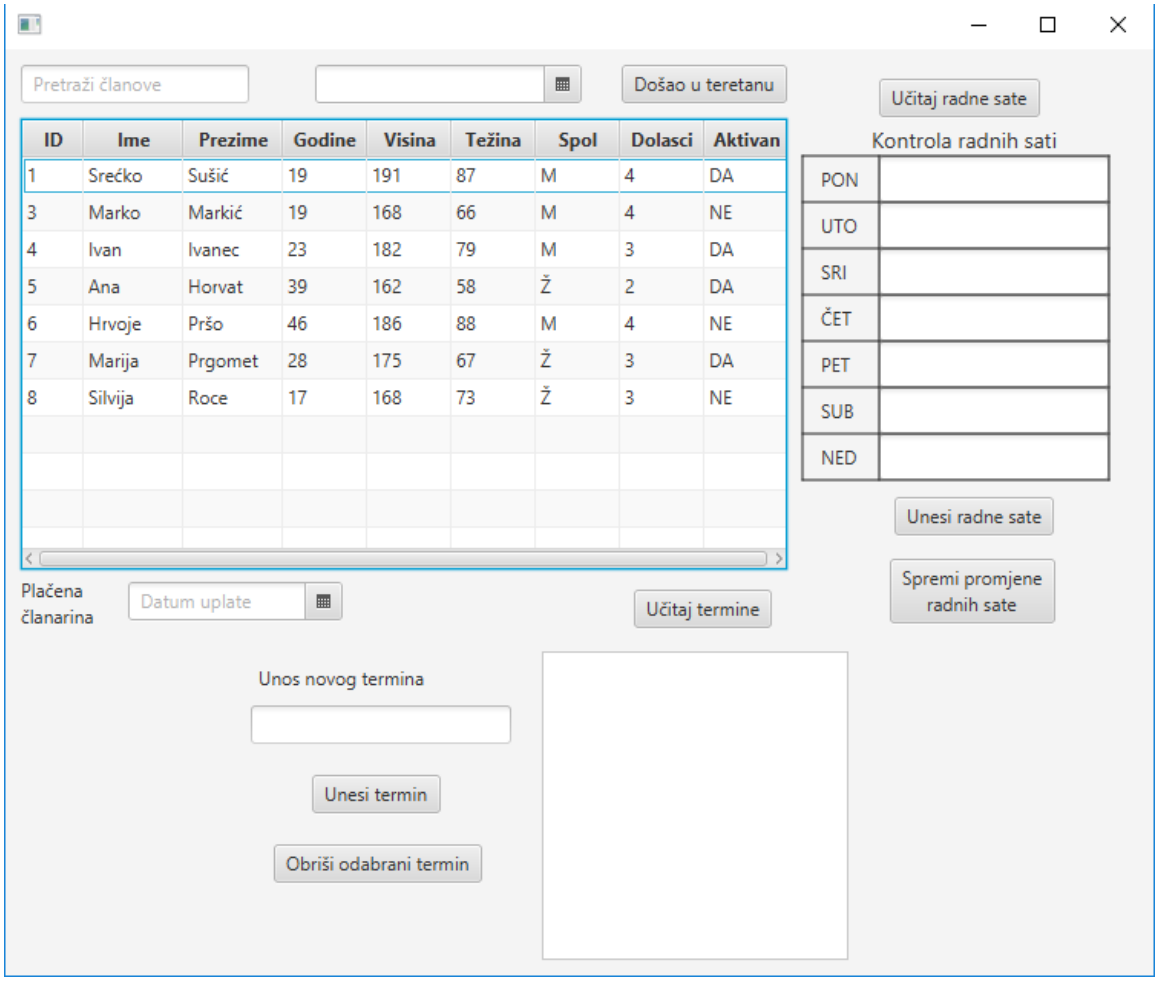
Slika 3.17. *Programski kôd metode „Promijeni()“.*

```
public void Izbrisi() throws SQLException{
    String query = "DELETE FROM clanovi WHERE id=?";
    try {
        PreparedStatement preparedStatement = MySqlKonekcija.Konektor().prepareStatement(query);
        preparedStatement.setString(1, txtId.getText());
        preparedStatement.execute();
        OcistiPolja();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    OsvjeziTablicu();
}
```

Slika 3.18. *Programski kôd metode „Izbrisi()“.*

3.4. Funkcionalnost trenera

Na slici 3.19. prikazano je sučelje aplikacije kada je prijavljen trener.



Slika 3.19. *Sučelje aplikacije za trenera*

Na slici 3.20. prikazan je programski kôd metode za unos novog termina.

```
public void DodajTermin() throws SQLException {
    PreparedStatement preparedStatement = null;
    if(ProvjeraPolja()) {
    try {
        String query = "INSERT into termini (idZaposlenika, termin) VALUES((SELECT id FROM zaposlenici "
                + "WHERE id = '"+txtIdZaposlenika.getText()+"'), ?)";
        preparedStatement = MySqlKonekcija.Konektor().prepareStatement(query);
        preparedStatement.setString(1, txtTermin.getText());
        OcistiPolja();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    finally {
        preparedStatement.execute();
        preparedStatement.close();
    }
    }
}
```

Slika 3.20. *Programski kôd metode „DodajTermin()“.*

Za unos novog termina u kojem nije ništa napisano poziva se metoda *„ProvjeraPolja()“*. Na slici 3.21. prikazan je programski kôd metode *„ProvjeraPolja()“*. Pri unosu termina u bazu dodaju se termin i identifikator zaposlenika koji je trenutno prijavljen u aplikaciju.

```
private boolean ProvjeraPolja() {
    if(txtTermin.getText().isEmpty()) {
        Alert alert = new Alert(AlertType.WARNING);
        alert.setTitle("Provjera polja");
        alert.setHeaderText(null);
        alert.setContentText("Unesite termin");
        alert.showAndWait();
        return false;
    }
    return true;
}
```

Slika 3.21. *Programski kôd metode „ProvjeraPolja“.*

Metoda ne dopušta unos u bazu dok se ne unese termin. Slika 3.22. prikazuje kako se ispisuju termini iz baze.

```
public void Termini() {
    List<String> termin = new ArrayList<>();
    try {
        String query = "SELECT termin FROM termini WHERE idZaposlenika = '"+txtIdZaposlenika.getText()+"'";
        PreparedStatement preparedStatement = MySqlKonekcija.Konektor().prepareStatement(query);
        ResultSet resultSet = preparedStatement.executeQuery();
        while(resultSet.next()) {
            termin.add(resultSet.getString("termin"));
        }
        ObservableList<String> termini = FXCollections.observableArrayList(termin);
        lvTermini.setItems(termini);
        preparedStatement.close();
        resultSet.close();
    } catch (SQLException e) {

        e.printStackTrace();
    }
}
```

Slika 3.22. *Programski kôd metode „Termini()“.*

Termini od zaposlenika koji je trenutno prijavljen u aplikaciju dodaju se u listu te se prikazuju kao lista podataka. Slika 3.23. prikazuje programski kôd metode za odabir termina.

```
public void OdabraniTermin() {
    try {
        String query = "SELECT * FROM termini WHERE termin = ?";
        PreparedStatement preparedStatement = MySqlKonekcija.Konektor().prepareStatement(query);
        preparedStatement.setString(1, (String)lvTermini.getSelectionModel().getSelectedItem());
        ResultSet resultSet = preparedStatement.executeQuery();
        while(resultSet.next()) {
            txtIdTermina.setText(resultSet.getString("idTermina"));

        }
        preparedStatement.close();
        resultSet.close();
    } catch (SQLException e) {

        e.printStackTrace();
    }
}
```

Slika 3.23. *Programski kôd metode „OdabraniTermin()“.*

Odabrani termin moguće je izbrisati, za brisanje termina koristi se metoda za brisanje termina, na slici 3.24 prikazan je programski kôd metode za brisanje termina.

```
public void IzbrisiOdabraniTermin() throws SQLException{
    String query = "DELETE FROM termini WHERE idTermina=?";
    try {
        PreparedStatement preparedStatement = MySqlKonekcija.Konektor().prepareStatement(query);
        preparedStatement.setString(1, txtIdTermina.getText());
        preparedStatement.execute();
        OcistiPolja();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Slika 3.24. *Programski kôd metode „IzbrisiOdabraniTermin()“.*

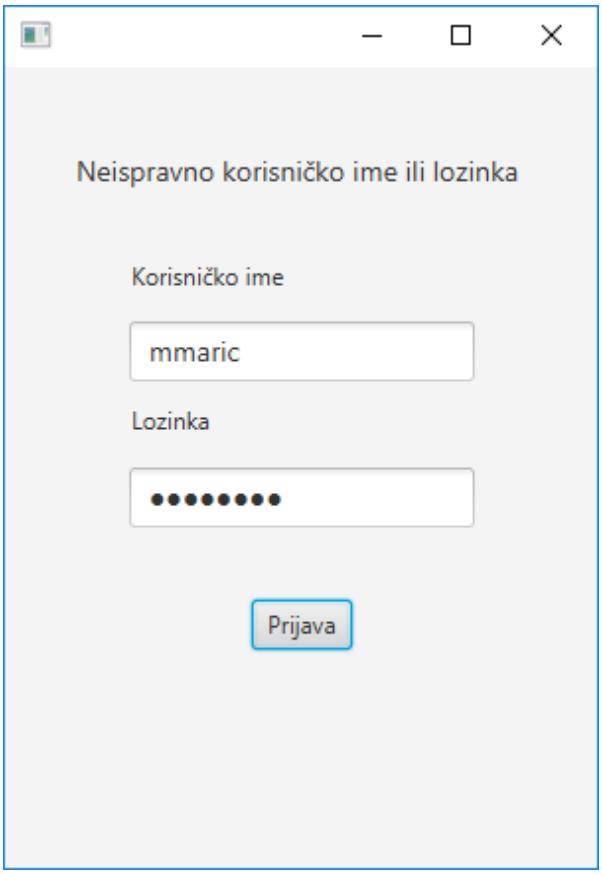
.

4. RAD S APLIKACIJOM

U ovom poglavlju biti će objašnjeno kako se koristi aplikacija.

4.1. Prijava

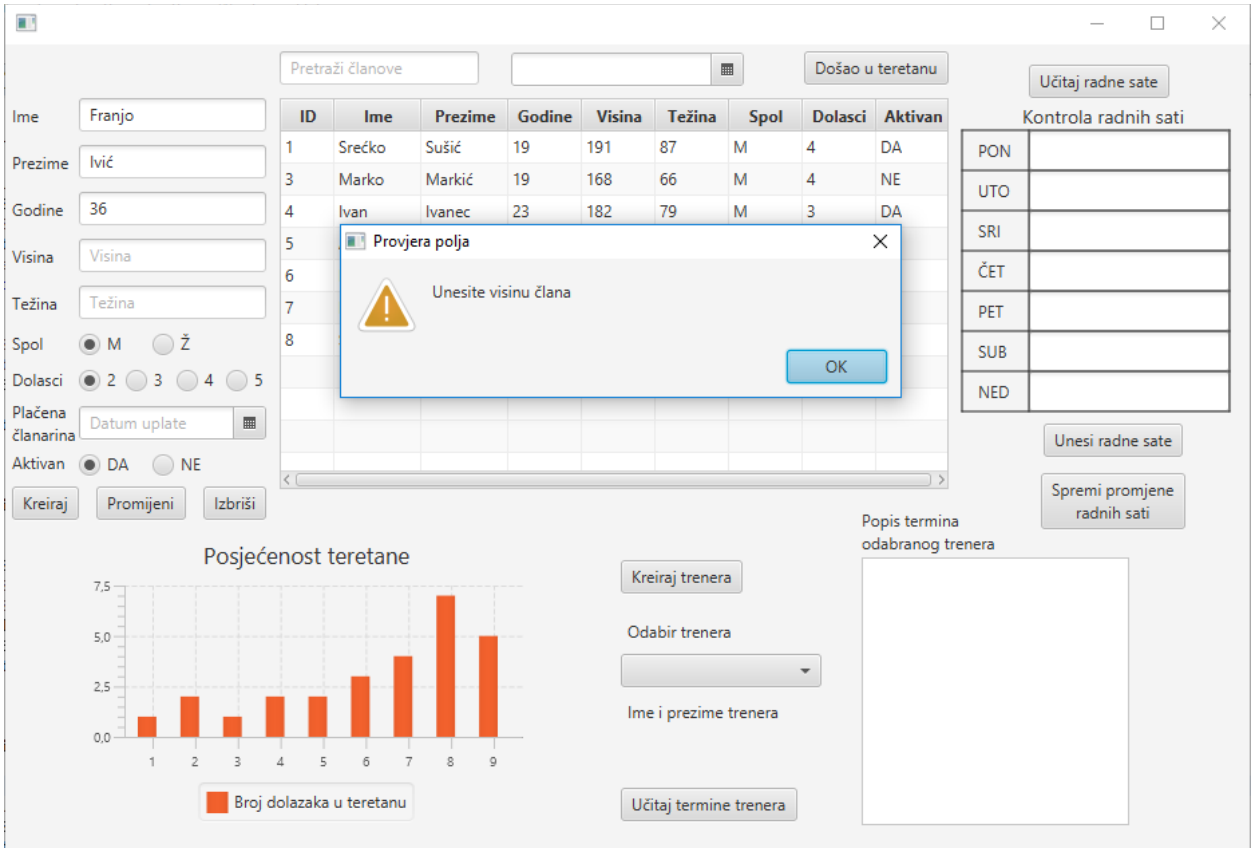
Za prijavu u aplikaciju potrebno je unijeti dva podatka, a to su korisničko ime zaposlenika i lozinka. Nakon unošenja podataka pritiskom tipke *„Prijava“* zaposlenik teretane se prijavljuje u aplikaciju. Izgled aplikacije za voditelja nakon prijave prikazan je na slici 4.2. Izgled aplikacije za trenera nakon prijave prikazan je na slici 4.9. Za krivo unesene podatke za korisničko ime ili lozinku ispisuje se poruka koja je prikazana na slici 4.1.



Slika 4.1. *Sučelje prijave u aplikaciju pri krivom unosu.*

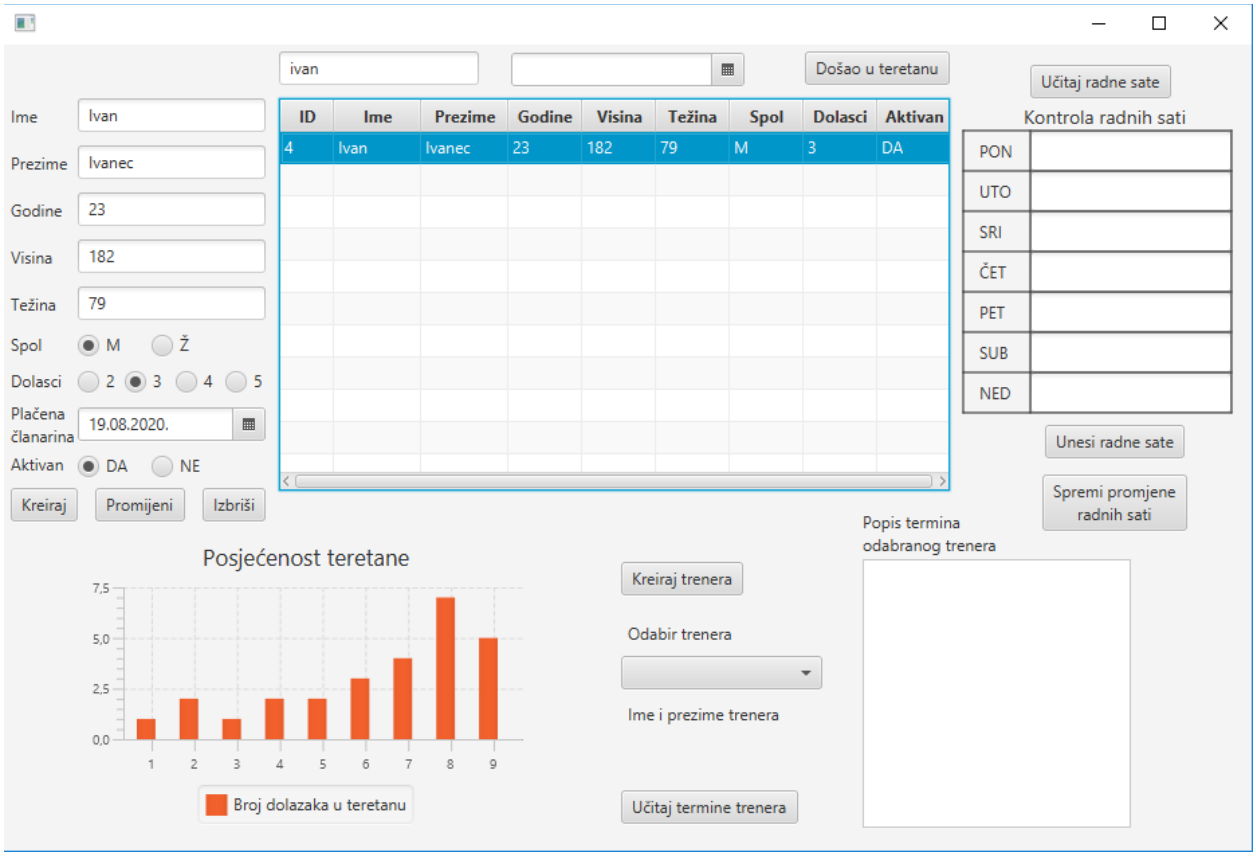
4.2. Voditelj

Prema slici 4.2. voditelju teretane omogućeno je kreiranje člana, kreiranje člana se izvodi tako da se upisuju podaci o članu koji su potrebni za unos člana. Ako se prilikom unosa podataka zaboravi unijeti neki podatak dobiva se upozorenje da taj podatak nije unesen. Slika 4.2. prikazuje unošenje člana s upozorenjem da nije unesen neki podatak.



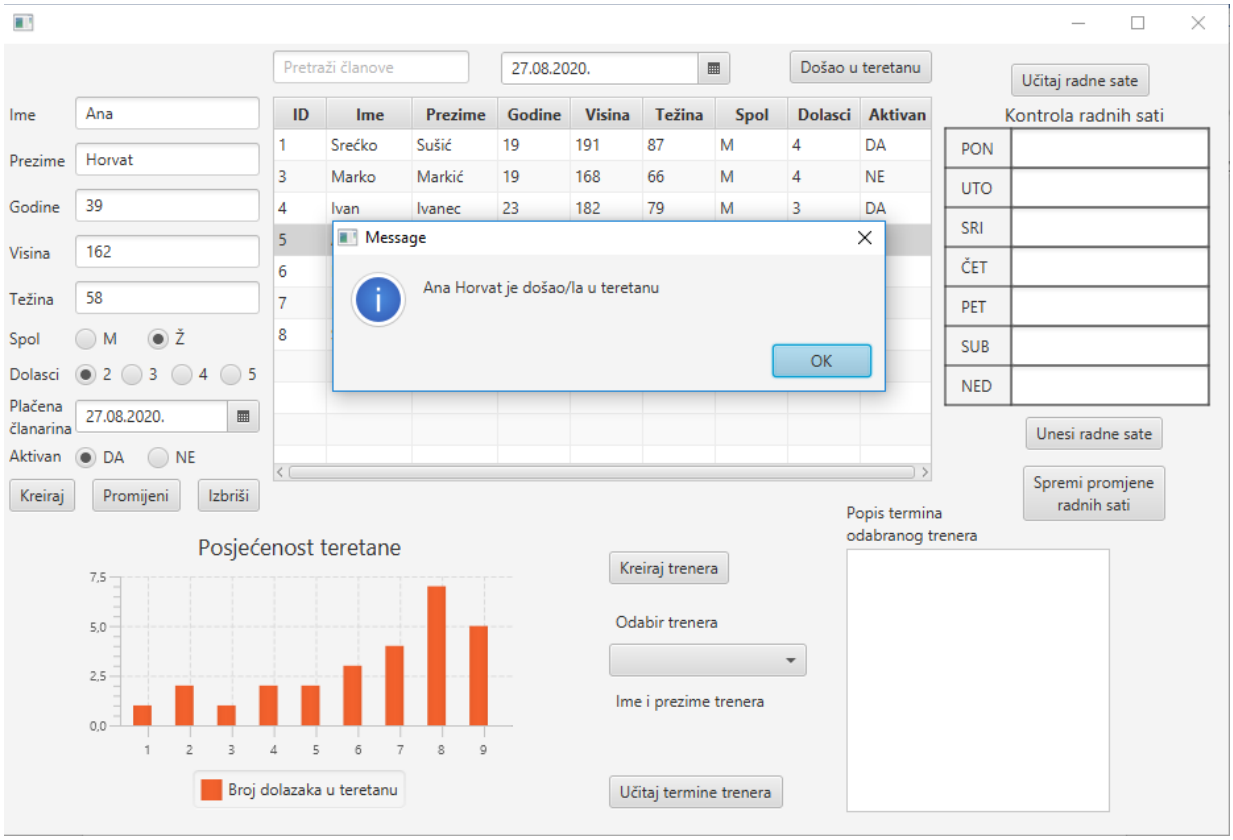
Slika 4.2. *Sučelje aplikacije pri unosu člana.*

Pritiskom tipke *„Kreiraj“* dodaju se novi članovi, sve dok nisu uneseni svi podaci koji su potrebni za unos člana dobiva se upozorenje da nije unesen taj podatak. Članove je moguće pretraživati po imenu i prezimenu s čime se smanjuje odabir na manji broj članova. Brisanjem podataka iz tražilice dobiva se opet cijeli popis članova. Odabranom članu teretane moguće je promijeniti podatke ili se može izbrisati. Slika 4.3. prikazuje sučelje aplikacije kada je odabran član s pretraživanjem. Nakon što su promijenjeni podaci, tipkom *„Promijeni“* spremaju se promjene i promijenjeni podaci se prikazuju u tablici. Za brisanje odabranog člana koristimo tipku *„Izbriši“* s kojom se briše iz baze te se više ne prikazuje u tablici.



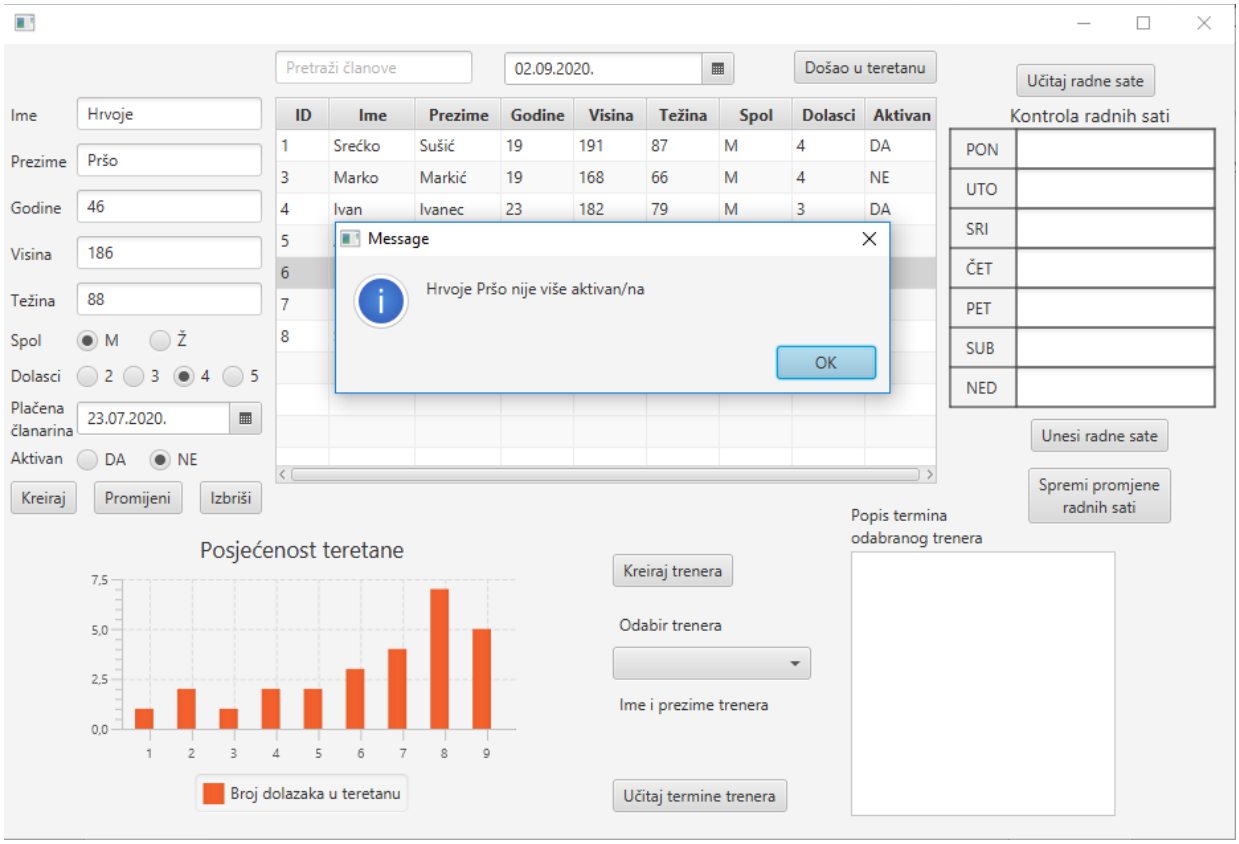
Slika 4.3. *Sučelje aplikacije s odabranim članom*.

Za odabranog člana moguće je unijeti dolazak u teretanu, odabire se datum kada je član došao u teretanu i pritiskom tipke *„Došao u teretanu“* sprema se dolazak u bazu. Slika 4.4. prikazuje dodavanje dolaska člana u teretanu.



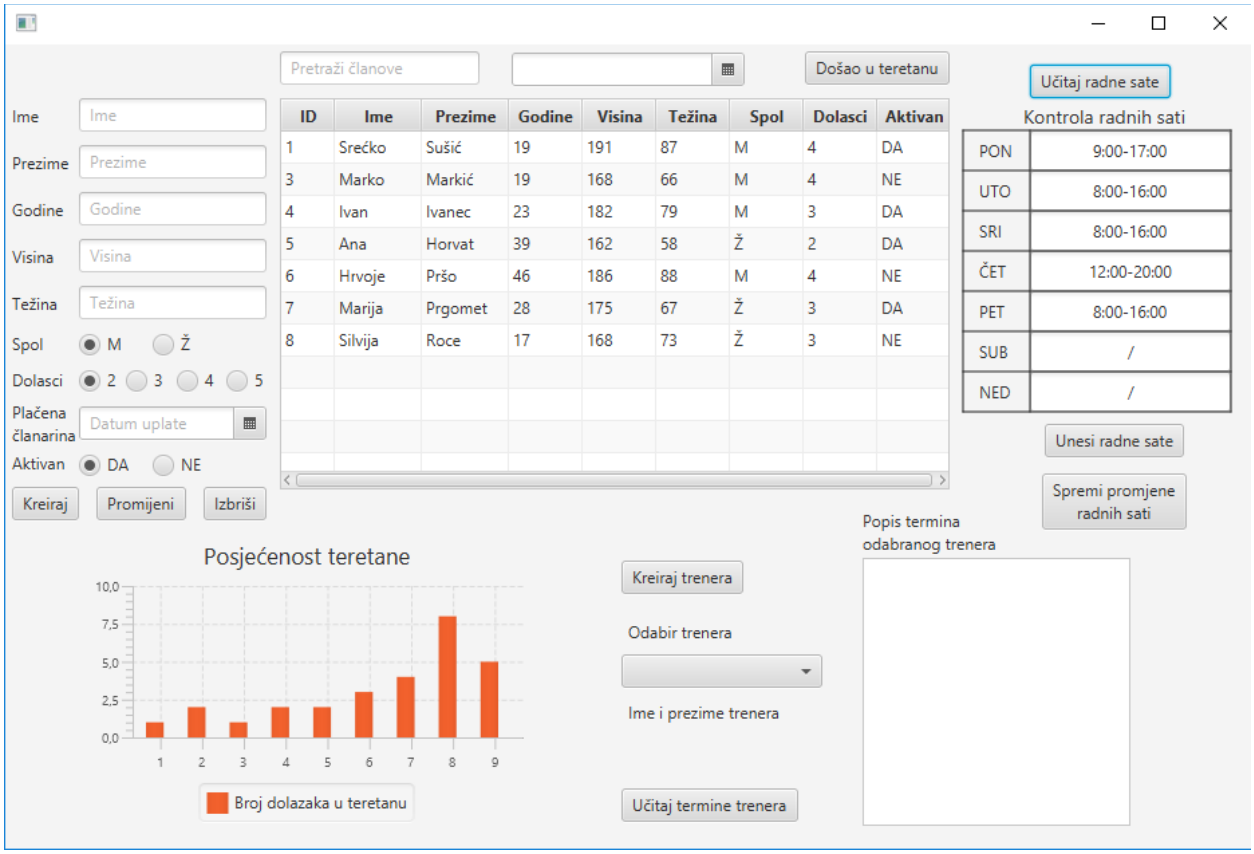
Slika 4.4 *Sučelje aplikacije pri unosu dolaska člana u teretanu.*

Za člana koji više nije aktivan ne dopušta se unos dolaska u teretanu, to prikazuje slika 4.5. Broj dolazaka u teretanu može se vidjeti u donjem lijevom kutu.



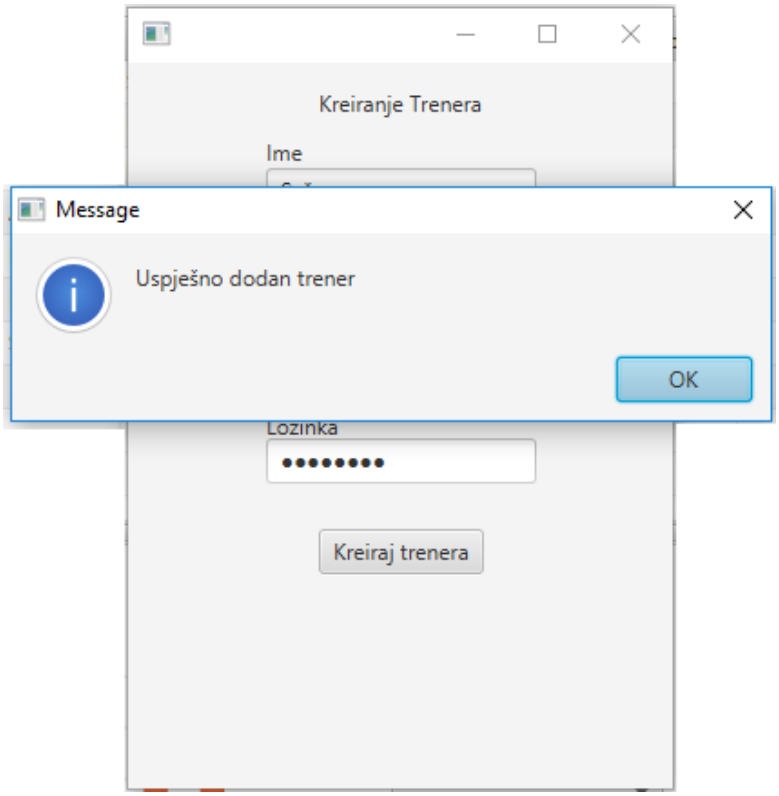
Slika 4.5 *Sučelje aplikacije pri unosu dolaska člana u teretanu ako član više nije aktivan.*

Pritiskom tipke *„Učitaj radne sate“* ispisuje se radno vrijeme po danima iz baze. Slika 4.6. prikazuje učitane radne sate. Ako zaposlenik nije upisao radne sate, upisuje radno vrijeme za svaki dan u tjednu te pritiskom tipke *„Unesi radne sate“* dodaje radne sate u bazu. Učitani radni sati mogu se promijeniti tako da se prepravi radno vrijeme za dan u kojem je potrebno promijeniti radno vrijeme i tipkom *„Spremi promjene radnih sati“* sprema se novo radno vrijeme. Kada se ponovno učitava radno vrijeme ispisuje se ono koje je zadnje uneseno.

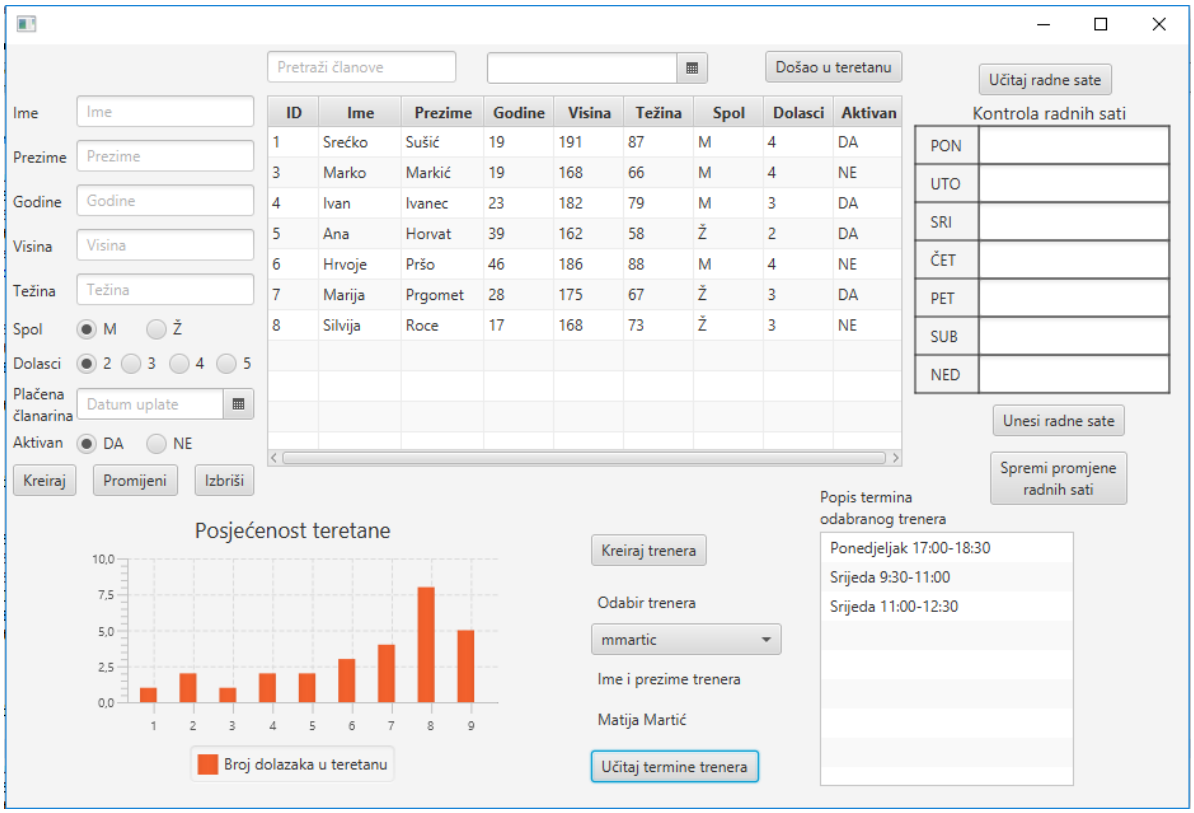


Slika 4.6. *Sučelje aplikacije kada su učitani radni sati*

Za kreiranje novog trenera otvara se novi prozor. Za kreiranje trenera potrebno je unesti ime, prezime, korisničko ime i lozinku. Slika 4.7. prikazuje uspješno dodavanje novog trenera. Ako je prilikom unosa podataka zaboravljeno unijeti neki podatak dobiva se upozorenje kao što je prikazano na slici 4.2 pri dodavanju novog člana. Za svakog trenera teretane moguće je vidjeti koje ima zakazane termine. Odabiranjem trenera iz popisa svih trenera teretane te pritiskom tipke *„Učitaj termine trenera“* ispisuju se termini koje taj odabrani trener ima zakazane. Slika 4.8. prikazuje ispis zakazanih termina odabranog trenera.



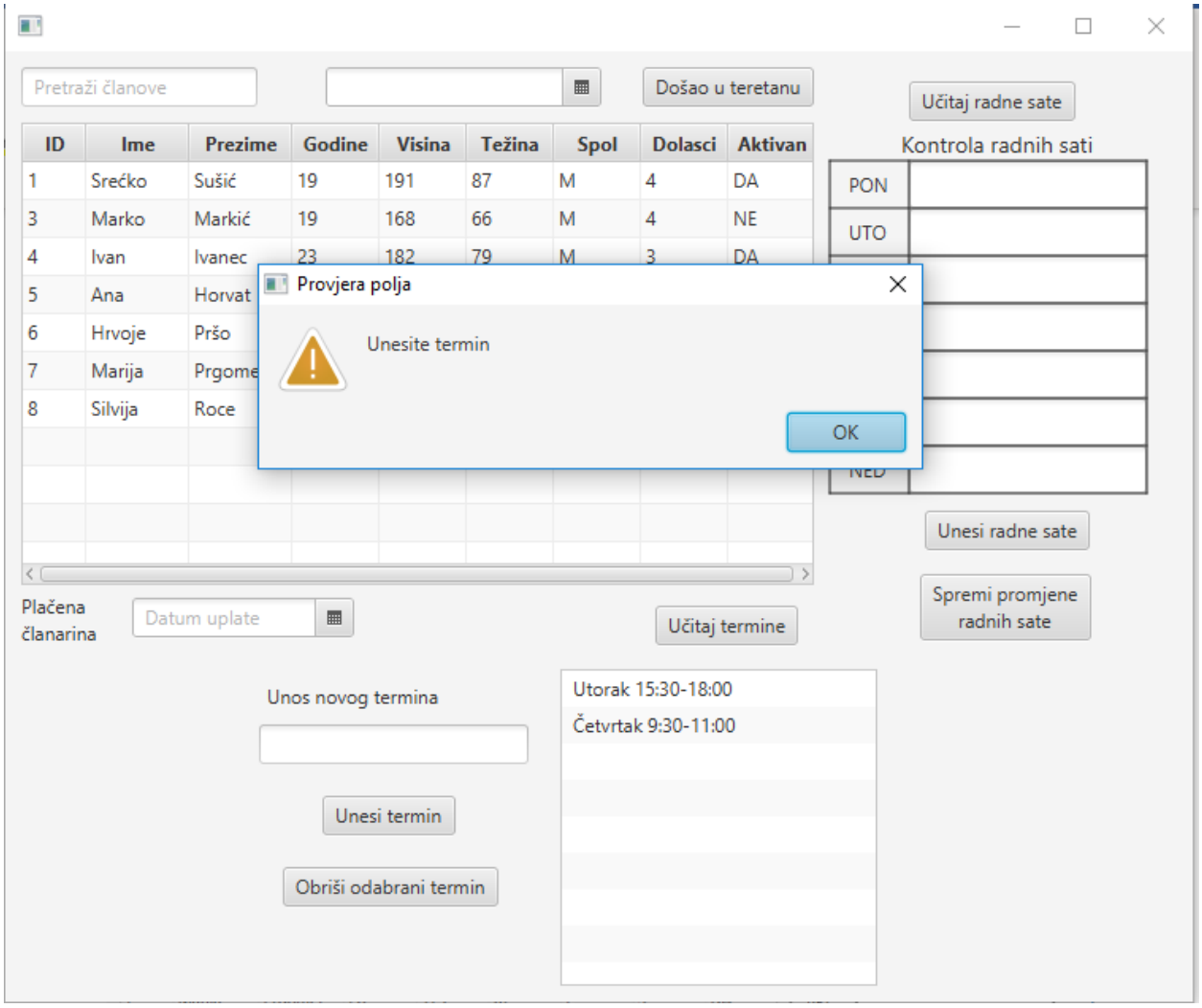
Slika 4.7. *Sučelje aplikacije za uspješno dodavanje novog trenera.*



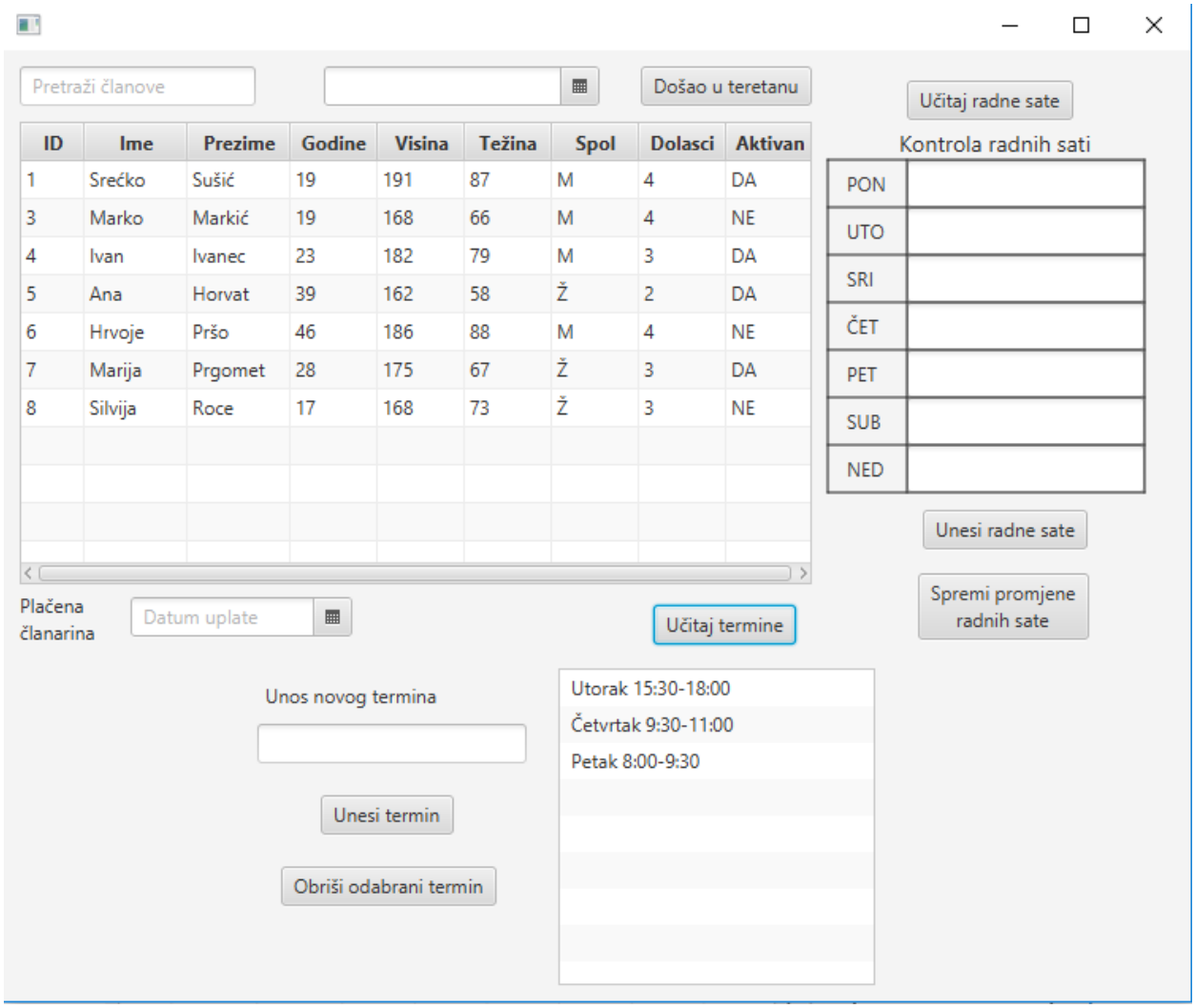
Slika 4.8. *Sučelje aplikacije za ispis zakazanih termina odabranog trenera*

4.3. Trener

Trener teretane kao i voditelj može unijeti dolazak člana u teretanu kao što je prikazano na slici 4.4. i na slici 4.5., kontrolirati radne sate kao što je prikazano na slici 4.6. te može dodavati i brisati zakazane termine. Nakon što se unese željeni termin pritiskom tipke *„Unesi termin“*, termin se dodaje u bazu. Ako za dodavanje novog termina nije upisan termin dobiva se upozorenje da unesemo termin. Slika 4.9. prikazuje unos termina s upozorenjem. Pritiskom tipke „Učitaj termine“ ispisuju se svi termini koji se nalaze u bazi za prijavljenog trenera. Termin se može izbrisati tako da se odabere termin koji se želi obrisati iz popisa te pritiskom *tipke „Obriši odabrani termin“* briše se iz baze. Pritiskom tipke *„Učitaj termine“* taj termin se više ne nalazi na popisu. Slika 4.10. prikazuje termine prijavljenog trenera u aplikaciju nakon što je uspješno dodan novi termin.



Slika 4.9. *Sučelje aplikacije pri unosu termina*



Slika 4.10. *Sučelje aplikacije s učitanim popisom termina trenera*

5. ZAKLJUČAK

Zadatak završnog rada je napraviti JavaFx GUI aplikaciju za vođenje teretane. Aplikacija je pisana u Eclipse razvojnom okruženju te je za rad s bazom podataka korišten softverski alat phpMyAdmin. Aplikacija je potpuno sigurna jer prije bilo kakvog korištenja aplikacije potrebno se prijaviti u aplikaciju. Aplikacija omogućava voditelju teretane unos članova i trenera teretane te uvid u status članova. Za svakog člana može unijeti dolazak u teretanu te ima uvid u posjećenost teretane po mjesecima. Za svakog trenera može provjeriti kada ima zakazane termine te u kontrolu svojih radnih sati. Treneru omogućava uvid u status članova, dodavanje dolaska člana u teretanu, dodavanje termina te uvid u zakazane termine te također kontrolu radnih sati. Za rad s aplikacijom potrebno je imati instaliran XAMPP te pristup internetu kako bi mogli pristupiti bazi podataka. Aplikacija je napisana u programskom jeziku Java te ju je zbog toga moguće pokrenuti na bilo kojem operacijskom sustavu koji ima instaliran JVM.

Aplikacija bi se mogla proširiti tako da se dolazak člana u teretanu evidentira pomoću RFID kartica te bi tako olakšali rad zaposlenicima teretane jer ne bi morali unositi dolazak za svakog člana koji dođe u teretanu.

LITERATURA

[1] Java; https://www.sciencedirect.com/topics/computer-science/java-programming-language [2.9..2020.]

[2] Java osnovni proces; https://www.javaassignmenthelp.com/Java-Programming-Help [3.9..2020.]

[3] JavaFx; http://tutorials.jenkov.com/javafx/index.html [3.9.2020.]

[4] Sql; https://www.w3schools.com/sql/sql_intro.asp [4.9.2020.]

[5] Eclipse IDE; https://www.infoworld.com/article/3114167/choosing-your-java-ide.html [4.9.2020.]

SAŽETAK

Tema: GUI aplikacija za vođenje teretane

GUI aplikaciju za vođenje teretane moguće je pokrenuti na bilo kojem operacijskom sustavu na kojemu je instaliran JVM. Aplikacija olakšava vođenje teretane tako što nudi sve informacije koje su potrebne voditelju i trenerima teretane za što lakše upravljanje s teretanom na jednom mjestu. Svi podaci o članovima teretane se nalaze u bazi podataka, o podacima iz baze podataka se brine voditelj teretane jer je on administrator aplikacije.

Ključne riječi: baza, evidencija, JavaFx, SceneBuilder, teretana

ABSTRACT

Title: GUI application for gym management

GUI application for gym management can be ran on any operating system on which the JVM is installed. The application facilitates gym management by displaying all necessary data to gym manager and trainers in one place. All data about gym members is stored in the database which is managed by gym manager who is also the applications administrator.

Keywords: database, gym, JavaFx, record, SceneBuilder

ŽIVOTOPIS

Mihael Lneniček rođen je 31. kolovoza 1995. godine u Bjelovaru. Osnovnu školu završio je u Češkoj osnovnoj školi Jan Amos Komensky u Daruvaru 2010. godine. Te iste godine upisuje srednju školu u Tehničkoj školi Daruvar koju završava 2014. godine te stječe zvanje tehničar za računalstvo. Akademske godine 2016./2017. upisuje preddiplomski stručni studij Elektrotehnike, smjer informatika na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Zna se služiti s bazom podataka te programskim jezicima kao što su Java, C, C++.

PRILOZI

Na DVD-u koji je priložen uz završni rad nalazi se program, programski kôd te word i pdf dokumenti.