7. Sedma laboratorijska vježba

7.1. SPAJANJE JAVAFX APLIKACIJE NA H2 BAZU PODATAKA

Svrha laboratorijske vježbe je spajanje JavaFX aplikacije na H2 bazu podataka, dohvaćanje i pohranjivanje podataka u nju. Umjesto datoteka koje su se koristile u šestoj laboratorijskoj vježbi, podatke je potrebno preuzeti iz kreirane i popunjene baze podataka. Za korištenje H2 baze podataka potrebno je instalirati Data Tools Platform (DTP) dodatak za Eclipse, te definirati *driver* za H2 bazu podataka unutar Eclipsea. Sve funkcionalnosti vježbe koje je potrebno implementirati moguće je vidjeti na sljedećem YouTube videu: https://www.youtube.com/watch?v=dfqQVi8YSLQ.

7.2. ZADATAK

Za implementaciju vježbe potrebno je obaviti sljedeće korake:

1. S mrežnih stranica http://www.eclipse.org/datatools/downloads.php preuzeti i instalirati DTP dodatak za Eclipse prema uputama u predavanjima.

2. Kopirati projekt iz šeste laboratorijske vježbe i preimenovati ga naziv s indeksom „7", npr. „Horvat-7".

3. Unutar razvojnog okruženja Eclipse kreirati instancu H2 baze podataka s proizvoljnim nazivom, te korisničkim imenom i lozinkom za pristup bazi. Nakon toga u toj bazi podataka izvršiti sljedeću SQL skriptu (podatke u tablici je moguće proizvoljno definirati):

```
CREATE SCHEMA RAZVOJ;
CREATE TABLE RAZVOJ.VRSTA_PUBLIKACIJE
(
id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
naziv VARCHAR(20) NOT NULL,
PRIMARY KEY (id)
);

INSERT INTO RAZVOJ.VRSTA_PUBLIKACIJE (naziv) VALUES ('ELEKTRONICKA');
INSERT INTO RAZVOJ.VRSTA_PUBLIKACIJE (naziv) VALUES ('PAPIRNATA');

CREATE TABLE RAZVOJ.JEZIK
(
id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
naziv VARCHAR(20) NOT NULL,
```

```
PRIMARY KEY (id)
);

INSERT INTO RAZVOJ.JEZIK (naziv) VALUES ('HRVATSKI');
INSERT INTO RAZVOJ.JEZIK (naziv) VALUES ('ENGLESKI');
INSERT INTO RAZVOJ.JEZIK (naziv) VALUES ('NJEMACKI');
INSERT INTO RAZVOJ.JEZIK (naziv) VALUES ('FRANCUSKI');
INSERT INTO RAZVOJ.JEZIK (naziv) VALUES ('TALIJANSKI');
INSERT INTO RAZVOJ.JEZIK (naziv) VALUES ('RUSKI');
INSERT INTO RAZVOJ.JEZIK (naziv) VALUES ('KINESKI');

CREATE TABLE RAZVOJ.IZDAVAC
(
id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
naziv VARCHAR(50) NOT NULL,
drzava VARCHAR(20) NOT NULL,
PRIMARY KEY (id)
);

INSERT INTO RAZVOJ.IZDAVAC (naziv, drzava) VALUES ('Školska knjiga', 'Hrvatska');
INSERT INTO RAZVOJ.IZDAVAC (naziv, drzava) VALUES ('Packt Publishing', 'SAD');

CREATE TABLE RAZVOJ.KNJIGA
(
id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
naziv VARCHAR(50) NOT NULL,
godinaIzdanja INT NOT NULL,
vrstaPublikacije INT NOT NULL,
brojStranica INT NOT NULL,
jezik INT NOT NULL,
izdavac INT NOT NULL,
PRIMARY KEY (id),
FOREIGN KEY (vrstaPublikacije) REFERENCES RAZVOJ.VRSTA_PUBLIKACIJE(id),
FOREIGN KEY (jezik) REFERENCES RAZVOJ.JEZIK(id),
FOREIGN KEY (izdavac) REFERENCES RAZVOJ.IZDAVAC(id)
);

INSERT INTO RAZVOJ.KNJIGA (naziv, godinaIzdanja, vrstaPublikacije, brojStranica, jezik,
izdavac) VALUES ('Programiranje u Javi', 2014, 1, 400, 1, 1);
INSERT INTO RAZVOJ.KNJIGA (naziv, godinaIzdanja, vrstaPublikacije, brojStranica, jezik,
izdavac) VALUES ('Java web applications', 2014, 2, 600, 2, 2);

CREATE TABLE RAZVOJ.CASOPIS
(
id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
naziv VARCHAR(20) NOT NULL,
godinaIzdanja INT NOT NULL,
vrstaPublikacije INT NOT NULL,
brojStranica INT NOT NULL,
mjesecIzdanja INT NOT NULL,
PRIMARY KEY (id),
FOREIGN KEY (vrstaPublikacije) REFERENCES RAZVOJ.VRSTA_PUBLIKACIJE(id)
);

INSERT INTO RAZVOJ.CASOPIS (naziv, godinaIzdanja, vrstaPublikacije, brojStranica,
mjesecIzdanja) VALUES ('Bug', 2014, 2, 200, 5);
INSERT INTO RAZVOJ.CASOPIS (naziv, godinaIzdanja, vrstaPublikacije, brojStranica,
mjesecIzdanja) VALUES ('Java Magazine', 2014, 2, 100, 5);
```

```
CREATE TABLE RAZVOJ.CLAN
(
id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
oib CHAR(11) NOT NULL,
ime VARCHAR(50) NOT NULL,
prezime VARCHAR(50) NOT NULL,
PRIMARY KEY (id)
);

INSERT INTO RAZVOJ.CLAN (oib, ime, prezime) VALUES ('12334534512', 'Pero', 'Perić');
INSERT INTO RAZVOJ.CLAN (oib, ime, prezime) VALUES ('44332211221', 'Ivo', 'Ivić');

CREATE TABLE RAZVOJ.POSUDBA_KNJIGA
(
id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
clan INT NOT NULL,
knjiga INT NOT NULL,
datumPosudbe DATE NOT NULL,
PRIMARY KEY (id),
FOREIGN KEY (clan) REFERENCES RAZVOJ.CLAN(id),
FOREIGN KEY (knjiga) REFERENCES RAZVOJ.KNJIGA(id)
);

CREATE TABLE RAZVOJ.POSUDBA_CASOPISA
(
id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
clan INT NOT NULL,
casopis INT NOT NULL,
datumPosudbe DATE NOT NULL,
PRIMARY KEY (id),
FOREIGN KEY (clan) REFERENCES RAZVOJ.CLAN(id),
FOREIGN KEY (casopis) REFERENCES RAZVOJ.CASOPIS(id)
);
```

4. Unutar projekta iz drugog koraka kreirati tekstualnu „properties“ datoteku koja će sadržavati URL baze i pristupne podatke za nju:

```
#Osnovni podaci za spajanje na bazu podataka
bazaPodatakaUrl = jdbc:h2:tcp://localhost/~/Knjiznica

#Podaci za pristupanje bazi podataka
korisnickoIme = student
lozinka = student
```

5. U projektu iz drugog koraka potrebno je kreirati novu klasu unutar paketa „hr.tvz.java.vjezbe.baza“ koja će komunicirati s bazom podataka. Prilikom spajanja na bazu podataka moraju se koristiti podaci definirani unutar „properties“ datoteke iz četvrtog koraka. Napisati privatne statičke metode koje će se koristiti za spajanje na bazu podataka i zatvaranje veze s njom.

6. Unutar paketa „hr.tvz.java.vjezbe.controller.base“ potrebno je kreirati novu klasu „DialogHelpers“ i unutar iste statičku metodu koja ne vraća niti prima parametre, a na poziv prikazuje poruku o greški prilikom pristupanja bazi podataka. Detalje o izradi dijaloga moguće je pronaći u prošloj laboratorijskoj vježbi.

7. Unutar klase iz petog koraka napisati javnu statičku metodu koja će dohvaćati sve zapise o knjigama iz baze podataka i vraćati listu knjiga. Osim toga je u sve klase koje se koriste unutar knjige potrebno dodati još jedan cjelobrojni parametar „id“ u koji će se spremati vrijednosti primarnog ključa iz baze podataka. Osim toga je potrebno proširiti konstruktor tako da prima i sprema taj podatak, te odgovarajuće „getter“ i „setter“ metode. Tu metodu je potrebno pozvati iz odgovarajućeg „controllera“ koji prikazuje podatke o knjigama na ekranu, umjesto čitanja podataka iz datoteke, kao što je to bilo implementirano u šestoj laboratorijskoj vježbi. Primjer implementacije te metode prikazan je u nastavku:

```
public static List<Knjiga> dohvatiKnjige() throws Exception {
	Connection connection = DatabaseUtils.connectToDatabase();
	List<Knjiga> listaKnjiga = new ArrayList<>();
	String queryString = "SELECT * FROM RAZVOJ.KNJIGA";
	PreparedStatement preparedStatement = connection
			.prepareStatement(queryString);
	ResultSet resultSet = preparedStatement.executeQuery();
	while (resultSet.next()) {
		int id = resultSet.getInt("id");
		String naziv = resultSet.getString("naziv");
		Integer godinaIzdanja = resultSet.getInt("godinaIzdanja");
		Integer vrstaPublikacijeId = resultSet.getInt("vrstaPublikacije");
		VrstaPublikacije vrstaPublikacije = null;
		for (VrstaPublikacije temp : VrstaPublikacije.values()) {
			if (vrstaPublikacijeId == temp.getKod()) {
				vrstaPublikacije = temp;
			}
		}
		Integer brojStranica = resultSet.getInt("brojStranica");
		Integer jezikId = resultSet.getInt("jezik");
		Jezik jezik = null;
		for (Jezik temp : Jezik.values()) {
			if (jezikId == temp.getKod()) {
				jezik = temp;
			}
		}
		Integer izdavacId = resultSet.getInt("izdavac");
		Izdavac izdavac = dohvatiIzdavaca(izdavacId);
		float cijenaStranice = (jezik == Jezik.HRVATSKI) ? 0.45f : 0.75f;
		Knjiga k = new Knjiga(id, naziv, jezik, izdavac, godinaIzdanja,
				brojStranica, vrstaPublikacije,
				BigDecimal.valueOf(cijenaStranice));
		listaKnjiga.add(k);
	}

	DatabaseUtils.closeConnectionToDatabase(connection);
	return listaKnjiga;
}
```

Metoda „dohvatiIzdavaca“ može izgledati ovako:

```
  private static Izdavac dohvatiIzdavaca(Integer izdavacId) throws Exception {
	Connection connection = DatabaseUtils.connectToDatabase();
	String queryString = "SELECT * FROM RAZVOJ.IZDAVAC WHERE ID = ?";
	PreparedStatement preparedStatement = connection
```

```
                .prepareStatement(queryString);
        preparedStatement.setInt(1, izdavacId);
        ResultSet rs = preparedStatement.executeQuery();
        Izdavac izdavac = null;
        while (rs.next()) {
                int id = rs.getInt("id");
                String naziv = rs.getString("naziv");
                String drzava = rs.getString("drzava");
                izdavac = new Izdavac(id, naziv, drzava);
        }
        DatabaseUtils.closeConnectionToDatabase(connection);
        return izdavac;
}
```

8. Slično kao i za knjige u sedmom koraku, dohvat podataka iz baze je potrebno implementirati i za časopise i za članove. Detalje oko SQL naredbi moguće je naći u napomeni ove laboratorijske vježbe.

9. Unutar klase iz petog koraka potrebno je kreirati metodu za unos nove knjige. Metodu je moguće implementirati na sljedeći način:

```
public static void spremiKnjigu(Knjiga knjiga) throws Exception {
        Connection connection = DatabaseUtils.connectToDatabase();
        connection.setAutoCommit(false);

        PreparedStatement insertIzdavac = connection
                .prepareStatement("INSERT INTO RAZVOJ.IZDAVAC (naziv, drzava) VALUES
(?, ?)");
        insertIzdavac.setString(1, knjiga.getIzdavac().getNaziv());
        insertIzdavac.setString(2, knjiga.getIzdavac().getDrzava());
        insertIzdavac.executeUpdate();
        //dohvat generiranog ključa zbog potrebe za unosom knjige koja je vezana na
        //izdavača
        ResultSet generatedKeys = insertIzdavac.getGeneratedKeys();
        if (generatedKeys.next()) {
                knjiga.getIzdavac().setId(generatedKeys.getInt(1));
        }
        PreparedStatement insertKnjiga = connection
                .prepareStatement("INSERT INTO RAZVOJ.KNJIGA (naziv, godinaizdanja,
vrstapublikacije, brojstranica, jezik, izdavac) VALUES (?, ?, ?, ?, ?, ?)");
        insertKnjiga.setString(1, knjiga.getNaziv());
        insertKnjiga.setInt(2, knjiga.getGodina());
        insertKnjiga.setInt(3, knjiga.getVrsta().getKod());
        insertKnjiga.setInt(4, knjiga.getBrStranica());
        insertKnjiga.setInt(5, knjiga.getJezik().getKod());
        insertKnjiga.setInt(6, knjiga.getIzdavac().getId());
        insertKnjiga.executeUpdate();
        connection.commit();
        connection.setAutoCommit(true);
        DatabaseUtils.closeConnectionToDatabase(connection);
}
```

Ova metoda radi pomoću transakcija jer je unutar unosa knjige potrebno unijeti i izdavača što znači da je simultano potrebno izvršiti dva upita.

10. Slično kao i za knjige u devetom koraku, unos podataka u bazu je potrebno implementirati i za časopise i za članove koje su ujedno i jednostavnije jer nemaju transakcije.

11. Unutar klase iz petog koraka potrebno je kreirati metodu za promjenu postojeće knjige. Metodu je moguće implementirati na sljedeći način:

```
public static void promijeniKnjigu(Knjiga knjiga) throws Exception {
	Connection connection = DatabaseUtils.connectToDatabase();
	connection.setAutoCommit(false);
	PreparedStatement insertIzdavac = connection
			.prepareStatement("UPDATE RAZVOJ.IZDAVAC SET naziv = ?, drzava = ?
WHERE ID = ?");
	insertIzdavac.setString(1, knjiga.getIzdavac().getNaziv());
	insertIzdavac.setString(2, knjiga.getIzdavac().getDrzava());
	insertIzdavac.setInt(3, knjiga.getIzdavac().getId());
	insertIzdavac.executeUpdate();
	PreparedStatement insertKnjiga = connection
			.prepareStatement("UPDATE RAZVOJ.KNJIGA SET naziv = ?, godinaizdanja
= ?, vrstapublikacije = ?, brojstranica = ?, jezik = ?, izdavac = ? WHERE ID = ?");
		insertKnjiga.setString(1, knjiga.getNaziv());
	insertKnjiga.setInt(2, knjiga.getGodina());
	insertKnjiga.setInt(3, knjiga.getVrsta().getKod());
	insertKnjiga.setInt(4, knjiga.getBrStranica());
	insertKnjiga.setInt(5, knjiga.getJezik().getKod());
	insertKnjiga.setInt(6, knjiga.getIzdavac().getId());
	insertKnjiga.setInt(7, knjiga.getId());
	insertKnjiga.executeUpdate();
	connection.commit();
	connection.setAutoCommit(true);
	DatabaseUtils.closeConnectionToDatabase(connection);
}
```

Ova metoda također posjeduje transakcije iz razloga što se simultano mijenjaju izdavač i knjiga.

12. Slično kao u jedanaestom koraku metode za promjenu podataka u bazi potrebno je implementirati za časopise i članove koje su ujedno i jednostavnije jer nemaju transakcije.

13. Umjesto kako su do sada funkcionirali unos i promjena podataka potrebno ih je implementirati po logici da se samo nova knjiga unosi, odnosno postojeća mijenja i metoda ne prima cijelu listu kako je bilo u prošloj laboratorijskoj vježbi. Varijablu „knjige“ potrebno je obrisati, a unos knjige promijeniti. Moguća implementacija unosa knjige nalazi se u nastavku:

```
  @FXML
  private void unesiKnjigu() {
      if (!(validirajVrijednost(nazivIzdavaca)
                  & validirajVrijednost(nazivKnjiga)
                  & validirajVrijednost(drzavaIzdavaca)
                  & validirajVrijednost(vrstaKnjige)
                  & validirajVrijednost(jezikKnjige)
                  & validirajBroj(godinaKnjige) & validirajBroj(brStranicaKnjige))) {
```

```
			Dialogs.create().title("Greška")
					.message("Podaci nisu u ispravnom formatu!").showError();
			return;
		}
		Jezik jezik = Jezik.valueOf(jezikKnjige.getValue());
		float cijenaStranice = (jezik == Jezik.HRVATSKI) ? 0.45f : 0.75f;
		Knjiga knjiga = new Knjiga(nazivKnjiga.getText(), jezik, new Izdavac(
				nazivIzdavaca.getText(), drzavaIzdavaca.getText()),
				Integer.parseInt(godinaKnjige.getText()),
				Integer.parseInt(brStranicaKnjige.getText()),
				VrstaPublikacije.valueOf(vrstaKnjige.getValue()),
				BigDecimal.valueOf(cijenaStranice));
		try {
			if (isEdit) {
				knjiga.setId(zaPrikaz.getId());
				knjiga.getIzdavac().setId(zaPrikaz.getIzdavac().getId());
				BazaPodataka.promijeniKnjigu(knjiga);

			} else
				BazaPodataka.spremiKnjigu(knjiga);
		} catch (Exception e) {
			DialogHelper.DatabaseError();
			e.printStackTrace();
			return;
		}
		Dialogs.create().title("Informacija")
				.message("Knjiga je uspješno unesena").showInformation();
}
```

14. U klasi iz petog koraka potrebno je implementirati metodu za spremanje podataka o posudbi knjige u tablicu „POSUDBA_KNJIGA". To je moguće napraviti na sljedeći način:

```
public static void spremiPosudbuKnjige(Posudba<Knjiga> posudba)
		throws Exception {
   Connection connection = DatabaseUtils.connectToDatabase();
   String queryString = null;
   queryString = "INSERT INTO RAZVOJ.POSUDBA_KNJIGA (clan, knjiga, datumPosudbe) VALUES
(?, ?, ?)";
   PreparedStatement preparedStatement = connection
                   .prepareStatement(queryString);
   preparedStatement.setInt(1, posudba.getClan().getId());
   preparedStatement.setInt(2, ((Knjiga) posudba.getPublikacija()).getId());
   preparedStatement.setDate(3, Date.valueOf(posudba.getDatum().toLocalDate()));
   preparedStatement.executeUpdate();
   DatabaseUtils.closeConnectionToDatabase(connection);
}
```

15. Slično kao u četrnaestom koraku potrebno je kreirati i metodu za spremanje podataka o posudbi časopisa u tablicu „POSUDBA_CASOPISA". Osim toga je potrebno napisati metode za dohvaćanje listi posudba knjiga i časopisa.

16. Ekran za pregled popisa knjiga doraditi na način da se dvostrukim klikom miša na odabranu knjigu otvara dodatni prozor pomoću kojeg korisnik odabire člana koji posuđuje odabranu knjigu. Nakon što korisnik odabere i knjigu, aplikacija treba zapisati podatak o

posudbi knjige u bazu podataka. To je moguće implementirati unutar metode „initialize" tako da se za tablicu koja prikazuje podatke o knjigama ugradi „EventHandler" koji će predavati informaciju o odabranoj knjizi klasi „ClanoviController" (dorade te klase su opisane u sljedećem koraku), te prikazati dijalog za odabir člana koji posuđuje knjigu:

```
knjigaTable.setOnMouseClicked(new EventHandler<MouseEvent>(){
      @Override
      public void handle(MouseEvent event) {
       if (event.getClickCount()>1) {
          Knjiga knjiga = (Knjiga)
           knjigaTable.getSelectionModel().getSelectedItem();
        try {
           FXMLLoader fxmlLoader = new FXMLLoader();
           URL location =
            ClanoviController.class.getResource("../javafx/clanovi.fxml");
            fxmlLoader.setLocation(location);
           fxmlLoader.setBuilderFactory(new JavaFXBuilderFactory());
           Parent root = (Parent)fxmlLoader.load(location.openStream());
           ClanoviController controller =
             (ClanoviController)fxmlLoader.getController();
           controller.setPublikacija(knjiga);
           Stage stage = new Stage();
           stage.setTitle("Odabir člana za posudbu knjige " +
             knjiga.getNaziv());
           stage.setScene(new Scene(root, 650, 250));
           stage.show();
             controller.setStage(stage);
         } catch (IOException e) {
           e.printStackTrace();
         }
      }
   }
});
```

17. Klasu „ClanoviController" potrebno je doraditi tako da mu se dodaju još dvije privatne varijable koje označavaju objekt koji predstavlja publikaciju i objekt klase „Stage", te generiraju pripadajuće „getter" i „setter" metode:

```
private Publikacija publikacija;
private Stage stage;
```

Također je unutar metode „initialize" „Controller" klase za rad s članovima potrebno implementirati „EventHandler" sličan onome za tablicu s popisom knjiga, koji će dohvatiti odabranog korisnika i obaviti spremanje podataka o posudbama u bazu pozivom metode definirane u osmom koraku:

```
clanTable.setOnMouseClicked(new EventHandler<MouseEvent>() {
      @Override
      public void handle(MouseEvent event) {
            if (publikacija == null)
                  return;
            if (event.getClickCount() > 1) {
                  Clan clan = (Clan) clanTable.getSelectionModel()
```

```
                                .getSelectedItem();
                        if (publikacija instanceof Knjiga) {
                                Posudba<Knjiga> posudba = new Posudba<>(clan,
                                                (Knjiga) publikacija, LocalDateTime.now());
                                try {
                                        BazaPodataka.spremiPosudbuKnjige(posudba);
                                } catch (Exception e) {
                                        e.printStackTrace();
                                        DialogHelper.DatabaseError();
                                }
                        } else {
                                Posudba<Casopis> posudba = new Posudba<>(clan,
                                                (Casopis) publikacija, LocalDateTime.now());
                                try {
                                        BazaPodataka.spremiPosudbuCasopisa(posudba);
                                } catch (Exception e) {
                                        e.printStackTrace();
                                        DialogHelper.DatabaseError();
                                }
                        }
                        Dialogs.create().title("Informacija")
                                        .message("Posudba je uspješno kreirana.")
                                        .showInformation();
                        stage.close();
                }
        }
});
```

18. Slično kao što je implementirana funkcionalnost za spremanje podataka o posudbi knjige, potrebno je implementirati i funkcionalnost za spremanje podataka o posudbama časopisa, pri čemu su potrebne ekvivalentne preinake klase „CasopisiController“ kao što je to bio slučaj kod klase „KnjigeController“.

19. Glavni izbornik je potrebno proširiti novim funkcionalnostima pomoću kojih će se ispisivati popis podataka o posudbama knjiga i popis podataka o posudbama časopisa. Nakon odabira tih opcija moraju se prikazati ekrani koji omogućavaju dohvat podataka o posudbama knjiga i časopisa, te mogućnosti „filtriranja“ zapisa po nazivima knjiga ili časopisa. Za te ekrane je potrebno kreirati „FXML“ datoteke, „Controller“ klase i metode za prikaz novih ekrana u glavnom „Controlleru“ (koji sadrži i metode za prikaz tablica s knjigama i časopisima).

20. „Controller“ klase koje služe za dohvaćanje podataka o posudbama knjiga i časopisa moraju izgledati vrlo slično ekvivalentnim „Controller“ klasama za dohvat knjiga i časopisa, samo što imaju dorađenu metodu „initialize“ za prikaz „ugnježđenih“ podataka (npr. naziva knjige unutar objekt klase „Posudba“). Primjer implementacije metode „initialize“ u slučaju „Controller“ klase za dohvat podataka o posudbama časopisa izgleda ovako:

```
@FXML
public void initialize() {
  nazivCasopisaColumn.setCellValueFactory(
  new Callback<CellDataFeatures<Posudba<Casopis>, String>,
  ObservableValue<String>>() {
```

```
    @Override
    public ObservableValue<String> call(
      CellDataFeatures<Posudba<Casopis>, String> data) {
        return new ReadOnlyObjectWrapper<String>(
         data.getValue().getPublikacija().getNaziv());
    }
  });

  prezimeKorisnikaColumn.setCellValueFactory(
    new Callback<CellDataFeatures<Posudba<Casopis>, String>,
      ObservableValue<String>>() {
        @Override
        public ObservableValue<String> call(
          CellDataFeatures<Posudba<Casopis>, String> data) {
             return new ReadOnlyObjectWrapper<String>(
             data.getValue().getClan().getPrezime());
    }
  });

  imeKorisnikaColumn.setCellValueFactory(
    new Callback<CellDataFeatures<Posudba<Casopis>, String>,
      ObservableValue<String>>() {
    @Override
    public ObservableValue<String> call(
      CellDataFeatures<Posudba<Casopis>, String> data) {
        return new ReadOnlyObjectWrapper<String>(
         data.getValue().getClan().getIme());
    }
  });

  datumPosudbeColumn.setCellValueFactory(
    new Callback<CellDataFeatures<Posudba<Casopis>, String>,
       ObservableValue<String>>() {
           @Override
           public ObservableValue<String> call(
           CellDataFeatures<Posudba<Casopis>, String> data) {
             DateTimeFormatter format = DateTimeFormatter
                    .ofPattern("dd.MM.yyyy.");
             return new ReadOnlyObjectWrapper<String>(format
                    .format(param.getValue().getDatum()));      }
    });
```

NAPOMENE:

1) Na laboratorijskim vježbama je moguće da baza podataka koja unutar URL-a sadržava znak „~“ neće funkcionirati normalno pa je preporuka umjesto npr. „jdbc:h2:tcp://localhost/~/Knjiznica“ koristiti URL „jdbc:h2:tcp://localhost/Knjiznica“.

2) Iz razloga što je kolegij baze podataka sljedeći semestar, u nastavku se nalazi popis nužnih SQL naredbi za rješavanje ove laboratorijske vježbe:

Dohvat svih časopisa:

```
SELECT * FROM RAZVOJ.CASOPIS
```

Dohvat svih članova:

```
SELECT * FROM RAZVOJ.CLAN
```

Pohranjivanje časopisa:
```
INSERT INTO RAZVOJ.CASOPIS (naziv, godinaizdanja, vrstapublikacije,
brojstranica, mjesecizdanja) VALUES (?, ?, ?, ?, ?)
```

Pohranjivanje člana:
```
INSERT INTO RAZVOJ.CLAN (ime, prezime, oib) VALUES (?, ?, ?)
```

Promjena časopisa:
```
UPDATE RAZVOJ.CASOPIS SET naziv = ?, godinaizdanja = ?, vrstapublikacije = ?,
brojstranica = ?, mjesecizdanja = ? WHERE ID = ?
```

Promjena člana:
```
UPDATE RAZVOJ.CLAN SET ime = ?, prezime = ?, oib = ? WHERE ID = ?
```

Brisanje časopisa:
```
DELETE FROM RAZVOJ.CASOPIS WHERE id = ?
```

Brisanje člana:
```
DELETE FROM RAZVOJ.CLAN WHERE id = ?
```

Pohranjivanje posudbe:
```
INSERT INTO RAZVOJ.POSUDBA_CASOPISA (clan, casopis, datumPosudbe) VALUES (?, ?,
?)
```

Dohvat posudbi časopisa:
```
SELECT * FROM RAZVOJ.POSUDBA_CASOPISA
```

Dohvat članova po ID-u:
```
SELECT * FROM RAZVOJ.CLAN WHERE id = ?
```

Dohvat časopisa po ID-u:
```
SELECT * FROM RAZVOJ.CASOPIS WHERE id = ?
```

Dohvat posudbi časopisa:
```
SELECT * FROM RAZVOJ.POSUDBA_CASOPISA
```

Dohvat časopisa po ID-u:
```
SELECT * FROM RAZVOJ.CASOPIS WHERE id = ?
```

Dohvat knjige po ID-u:
```
SELECT * FROM RAZVOJ.KNJIGA WHERE id = ?
```

Dohvat izdavača po ID-u:
```
SELECT * FROM RAZVOJ.IZDAVAC WHERE ID = ?
```

SQL jezik nije osjetljiv na mala i velika slova, te je u potpunosti svejedno hoće li biti upit napisan velikim ili malim slovima.

3) Svi detalji koji nisu definirani mogu se proizvoljno definirati.