8. Osma laboratorijska vježba

8.1. 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 sedmoj 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=PT0IX2Wpo5c.

8.2. 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 sedme laboratorijske vježbe i preimenovati ga naziv s indeksom „8", npr. „Horvat-8".

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, mje-
secIzdanja) VALUES ('Bug', 2014, 2, 200, 5);
INSERT INTO RAZVOJ.CASOPIS (naziv, godinaIzdanja, vrstaPublikacije, brojStranica, mje-
secIzdanja) 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 novi paket i unutar njega klasu koja će komunicirati s bazom podataka. Prilikom spajanja na bazu podataka moraju se koristiti podaci definirani unutar „properties" datoteke iz četvrtog koraka. Napisati privatne metode koje će se koristiti za spajanje na bazu podataka i zatvaranje veze s njom.

6. 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 sedmoj laboratorijskoj vježbi. Primjer implementacije te metode prikazan je u nastavku:

```
public static List<Knjiga> dohvatiKnjige() throws Exception {
	Connection veza = connectToDatabase();

	List<Knjiga> listaKnjiga = new ArrayList<Knjiga>();

	String queryString = "SELECT * FROM RAZVOJ.KNJIGA";
```

```
      PreparedStatement preparedStatement =
         veza.prepareStatement(queryString);

      ResultSet rs = preparedStatement.executeQuery();

      while (rs.next()) {
            int id = rs.getInt("id");
            String naziv = rs.getString("naziv");
            Integer godinaIzdanja = rs.getInt("godinaIzdanja");
            Integer vrstaPublikacijeId = rs.getInt("vrstaPublikacije");
            VrstaPublikacije vrstaPublikacije = null;
            for(VrstaPublikacije temp : VrstaPublikacije.values()) {
                  if(vrstaPublikacijeId == temp.getKod()) {
                        vrstaPublikacije = temp;
                  }
            }
            Integer brojStranica = rs.getInt("brojStranica");
            Integer jezikId = rs.getInt("jezik");
            Jezik jezik = null;
            for(Jezik temp : Jezik.values()) {
                  if(jezikId == temp.getKod()) {
                        jezik = temp;
                  }
            }
            Integer izdavacId = rs.getInt("izdavac");
            Izdavac izdavac = dohvatiIzdavaca(izdavacId);

            Knjiga knjiga = new Knjiga(id, naziv, jezik, izdavac,
                        godinaIzdanja, vrstaPublikacije, brojStranica);

            listaKnjiga.add(knjiga);
      }

      closeConnectionToDatabase(veza);

      return listaKnjiga;
}
```

Metoda „dohvatiIzdavaca" može izgledati ovako:

```
private static Izdavac dohvatiIzdavaca(Integer id) throws Exception {
      Connection veza = connectToDatabase();

      String queryString = "SELECT * FROM RAZVOJ.IZDAVAC WHERE ID = ?";

      PreparedStatement preparedStatement =
            veza.prepareStatement(queryString);
      preparedStatement.setInt(1, id);
      ResultSet rs = preparedStatement.executeQuery();
```

```
      Izdavac izdavac = null;

      while (rs.next()) {
            int izdavacId = rs.getInt("id");
            String naziv = rs.getString("naziv");
            String drzava = rs.getString("drzava");
            izdavac = new Izdavac(izdavacId, naziv, drzava);
      }

      closeConnectionToDatabase(veza);

      return izdavac;
}
```

7. Slično kao i za knjige u šestom koraku, dohvat podataka iz baze je potrebno implementirati i za časopise i za članove.

8. 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
SQLException, IOException {
      Connection veza = connectToDatabase();

      String queryString = null;

      queryString =
"INSERT INTO RAZVOJ.POSUDBA_KNJIGA (clan, knjiga, datumPosudbe) VALUES
(?, ?, ?)";

      PreparedStatement preparedStatement =
         veza.prepareStatement(queryString);
      preparedStatement.setInt(1, posudba.getClan().getId());
      preparedStatement.setInt(2,
         ((Knjiga)posudba.getPublikacija()).getId());

      preparedStatement.setDate(3,
          convertToSQLDate(posudba.getDatumPosudbe()));
      preparedStatement.executeUpdate();

      closeConnectionToDatabase(veza);
}
```

Metoda za konverziju datuma može izgledati ovako:

```
private static java.sql.Date convertToSQLDate(final DateTime p_date) {
      Calendar calendar = Calendar.getInstance();
      calendar.setTime(p_date.toDate());
      java.sql.Date datum = new java.sql.Date(
         calendar.getTimeInMillis());
```

```
      return datum;
}
```

9. Slično kao u osmom koraku potrebno je napisati 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.

10. 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();
          }
      }
   }
});
```

11. Opciju „Članovi" koja dohvaća listu članova potrebno je maknuti iz glavnog izbornika, jer tamo više nije potrebna. 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" 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 (event.getClickCount() > 1) {
           Clan clan = (Clan) clanTable.getSelectionModel()
                                       .getSelectedItem();
          if (publikacija instanceof Knjiga) {
             Posudba<Knjiga> posudba = new Posudba<>(clan,
                          (Knjiga) publikacija, new DateTime());
             try {
                BazaPodataka.spremiPosudbuKnjige(posudba);
             } catch (Exception e) {
                e.printStackTrace();
             }
          } else {
             Posudba<Casopis> posudba = new Posudba<>(clan,
                          (Casopis) publikacija, new DateTime());
             try {
                BazaPodataka.spremiPosudbuCasopisa(posudba);
             } catch (Exception e) {
                e.printStackTrace();
             }

          }
          stage.close();
     }
   }
});
```

12. 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".

13. 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).

14. „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 knjiga 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) {
             SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy.");
             return new ReadOnlyObjectWrapper<String>(
               sdf.format(data.getValue().getDatumPosudbe().toDate()));
         }
     });
```

NAPOMENE:

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

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