6. Šesta laboratorijska vježba

6.1. DIZAJNIRANJE JAVAFX GRAFIČKOG SUČELJA

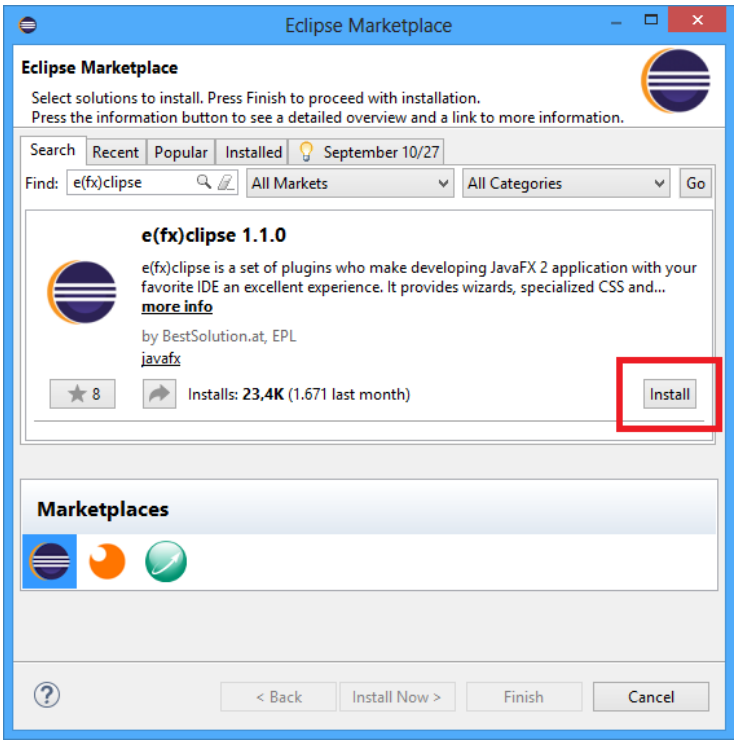
Svrha laboratorijske vježbe je upoznavanje načina dizajniranja grafičkog sučelja korištenjem JavaFX komponenti. Za dizajniranje je potrebno koristiti Scene Builder 2.0., a za razvoj aplikacije je potrebno koristiti Eclipse koji ima sve potrebne *pluginove* i koji je moguće preuzeti kao cjelinu s mrežnih stranica koje će biti navedene u nastavku vježbe. YouTube video koji prikazuje punu funkcionalnost aplikacije koju je potrebno implementirati moguće je vidjeti na sljedećoj poveznici: https://www.youtube.com/watch?v=oDr7w9G66pY.

6.2. ZADATAK

Za implementaciju vježbe potrebno je obaviti sljedeće korake i uvažiti primjedbe navedene na kraju dokumenta:

1. S mrežnih stranica http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html potrebno je preuzeti i instalirati razvojno okruženje Scene Builder (ako već nije instalirano na računalu).

2. Instalirati *plugin* za Eclipse pod nazivom "e(fx)clipse" na način da se odabere opcija u Eclipse IDE-u "Help->Eclipse Marketplace..." te u pretragu upiše „e(fx)clipse“. Nakon navedenog potrebno je instalirati „e(fx)clipse“ na tipku „install“ kao što je prikazano na sljedećoj slici.



Slika 1. Instalacija „e(fx)clipse“ *plugin-a*

3. Pomoću novog *plugin-a* za Eclipse potrebno je kreirati novi „JavaFX Project“ korištenjem opcije „File->New->Other->JavaFX->JavaFX Project“. Projekt je potrebno nazvati po prezimenu i rednom broju vježbe, npr. „Horvat-7“. Glavnu klasu koja se generira potrebno je promijeniti tako da izgleda kao sljedeći primjer:

```
public class JavaFXGlavna extends Application {

    private static BorderPane root;
    private Stage primaryStage;

    @Override
    public void start(Stage stage) {
           primaryStage = stage;
           try {
                  BorderPane rootPane = (BorderPane)
                     FXMLLoader.load(JavaFXGlavna.class
                                .getResource("../javafx/glavna.fxml"));
                  root = rootPane;
                  Scene scene = new Scene(root, 650, 250);
                  scene.getStylesheets().add(getClass().getResource(
                       "../glavna/application.css").toExternalForm());
                  primaryStage.setScene(scene);
                  primaryStage.show();
           } catch (Exception e) {
                  e.printStackTrace();
           }
    }

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

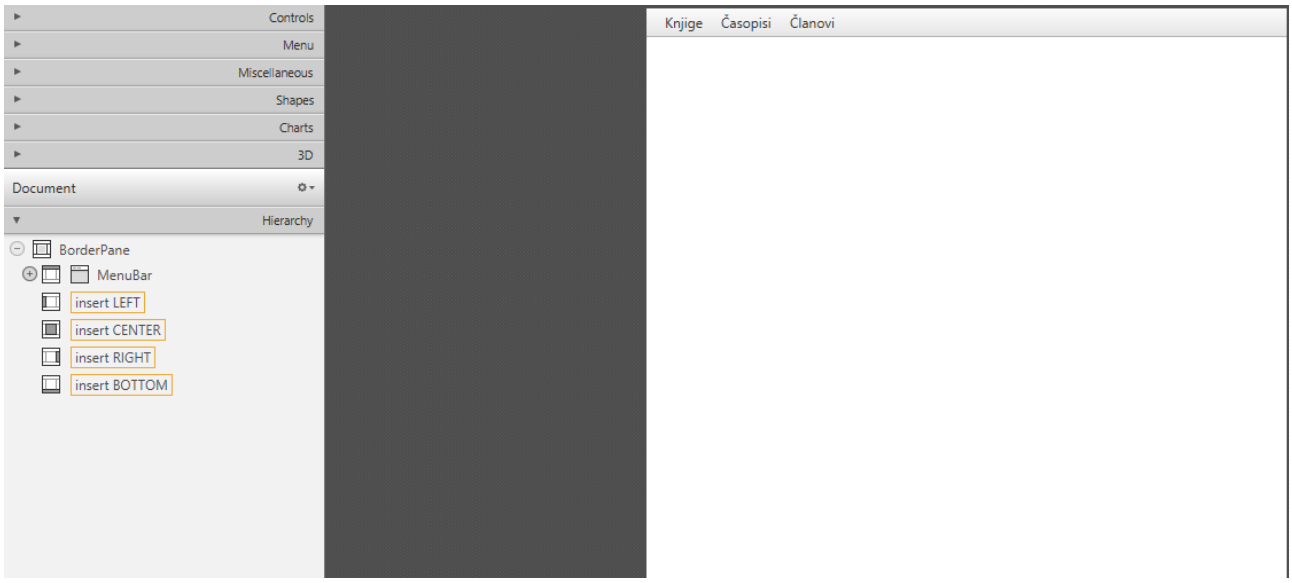
```
	public static void setCenterPane(BorderPane centerPane) {
		root.setCenter(centerPane);
	}
```

Preporučljivo je da se glavna klasa nalazi u paketu poput naziva „hr.tvz.java.vjezbe.glavna", kako bi se relativno od njega moglo pristupati ostalim resursima aplikacije. Na primjer, „../javafx/glavna.fxml" označava lokaciju datoteke „glavna.fxml" unutar paketa „hr.tvz.java.vjezbe.javafx".

4. Potrebno je kreirati „Maven" projekt slično kao u trećoj laboratorijskoj vježbi i kopirati „pom.xml" datoteku u projekt iz šeste laboratorijske vježbe u trenutni projekt.

5. Iz šeste vježbe je u sedmu potrebno kopirati paket s domenskim klasama, paket s enumeracijama, paket s iznimkama i tekstualne datoteke koje sadrže podatke o članovima, knjigama i časopisima.

6. Kreirati novi paket u kojem će se nalaziti FXML datoteke s kojima su definirani izgledi ekrana. Unutar tog paketa potrebno je kreirati prvu FXML datoteku pod nazivom „glavna.fxml" koja će sadržavati „kostur" ekrana aplikacije s izbornikom u kojem će se prikazivati ostale funkcionalnosti aplikacije. Datoteku je moguće kreirati korištenjem opcije „File->New->Other->JavaFX->New FXML Document". Nakon toga je potrebno označiti novu datoteku, kliknuti na desnu tipku miša i odabrati opciju „Open With SceneBuilder". Ekran treba dizajnirati tako da izgleda slično onome koji je prikazan na slici 2, korištenjem elemenata koji su prikazani u „Hierarchy" modulu. Prilikom dizajniranja grafičkog sučelja poželjno je koristiti „manual" naveden u četvrtoj napomeni na kraju dokumenta. Svaki od izbornika „Knjige", „Časopisi" i „Članovi" mora imati jedan „podizbornik" koji će pokretati pretraživanje entiteta (npr. „Pretraživanje knjiga").



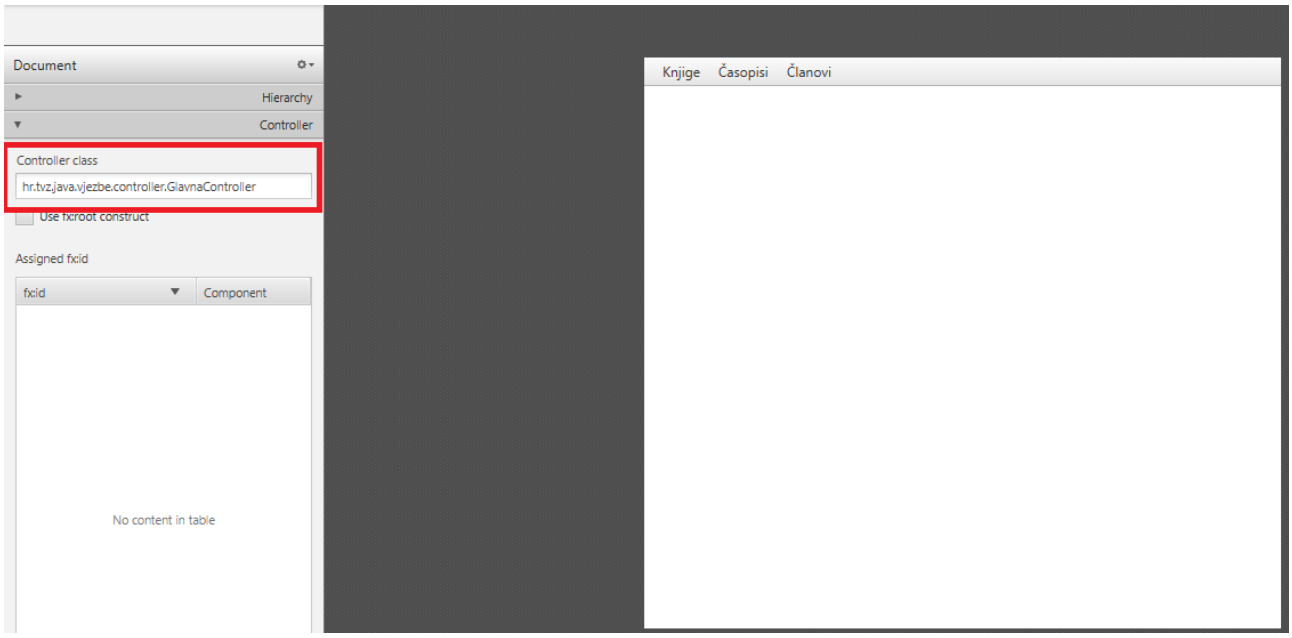
Slika 2. Izgled glavnog ekrana aplikacije

7. Kreirati novi paket pod nazivom npr. „hr.tvz.java.vjezbe.controller" u kojem će se nalaziti sve „Controller" klase koje će se koristiti unutar projekta. Na početku je unutar tog paketa

potrebno kreirati klasu pod nazivom „GlavnaController“ koja će sadržavati metode koje će se pozivati iz izbornika s ekrana prikazanog na slici 2. Primjer izgleda tog „Controllera“ s jednom metodom za prikaz ekrana koji će sadržavati tablicu s knjigama je prikazan u nastavku:

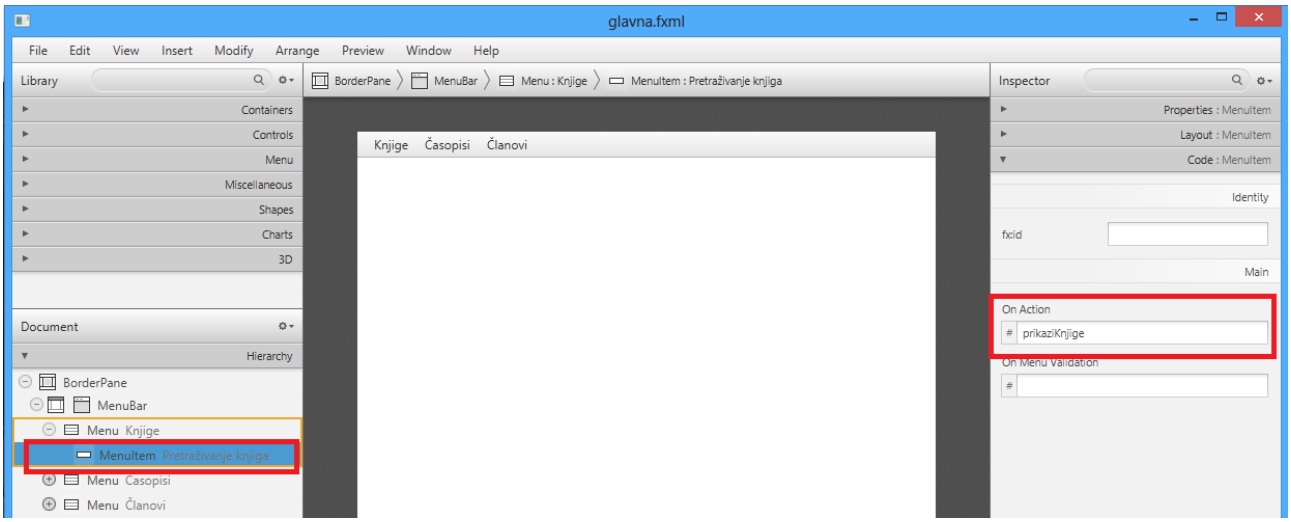
```
public class GlavnaController {

	public void prikaziKnjige() {
		try {
			BorderPane knjigePane = (BorderPane)
			   FXMLLoader.load(JavaFXGlavna.class
					.getResource("../javafx/knjige.fxml"));
			JavaFXGlavna.setCenterPane(knjigePane);
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

Nakon što je napisana ta metoda, korištenjem Scene Buildera je potrebno povezati komponentu na vrhu hijerarhije i upisati naziv klase koja označava klasu „GlavnaController“ kao na sljedećoj slici (unutar „Code“ modula):



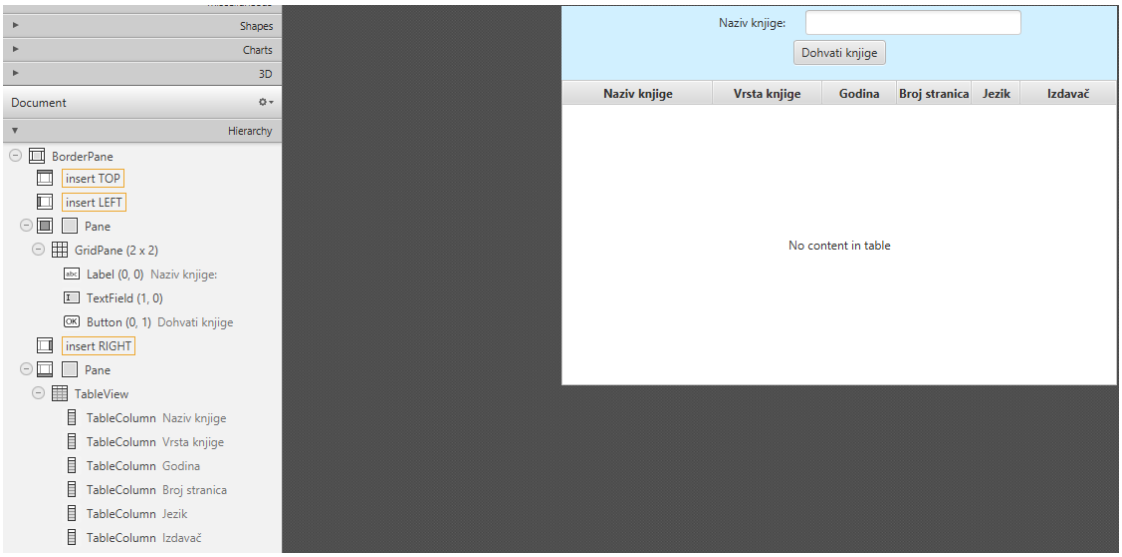
Slika 3. Konfiguriranje „Controller“ klase za ekran

Nakon povezivanja ekrana s „Controller“ klasom potrebno je povezati izbornik „Pretraživanje knjiga“ unutar glavnog izbornika „Knjige“. To je moguće napraviti tako da se odabere komponenta s izbornikom i unutar modula „Code“ u polje „On Action“ upiše naziv metode „prikaziKnjige“ iz „GlavnaController“ klase koju je potrebno pozvati prilikom odabira te opcije u izborniku (što je prikazano na slici 4).



Slika 4. Konfiguriranje metode „Controllera“ koja će se pozivati na odabir te opcije

8. Unutar paketa s FXML datotekama potrebno je kreirati novu datoteku pod nazivom „knjige.fxml“ koja će sadržavati komponente za filtriranje zapisa o knjigama i samu tablicu s knjigama.



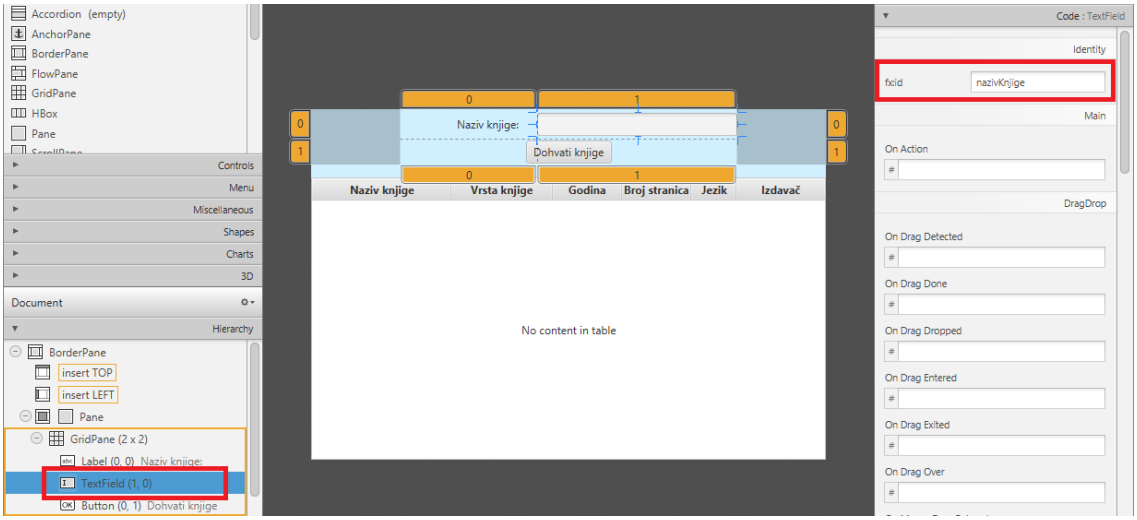
Slika 5. Izgled ekrana za prikaz i filtriranje podataka o knjigama

Preporuča se korištenje panela za svaki od segmenata ekrana radi lakšeg manipuliranja rasporedom komponenti na ekranu.

9. U paketu s „Controller“ klasama potrebno je kreirati novu klasu „KnjigeController“ koja će implementirati funkcionalnosti dohvaćanja podataka o knjigama iz tekstualnih datoteka i njihovo „filtriranje“ po nazivu. Osim toga je u tu klasu potrebno „preseliti“ cjelokupni kod koji čita tekstualnu datoteku „knjige.txt“, kreira objekte klase „Knjiga“ i dodaje ih u listu, koju je kasnije potrebno konvertirati u objekt klase „ObservableList“. Varijable i metode iz „KnjigeController“ je potrebno povezati s grafičkim sučeljem definiranim u sučelju „knjige.fxml“.

Prvo je potrebno „knjige.fxml“ povezati s nazivom „KnjigeController“ klase (kao u šestom koraku), a nakon toga gumb „Dohvati knjige“ povezati s metodom „prikaziSveKnjige“ (što je također prikazano u šestom koraku na primjeru izbornika za pretraživanje knjiga).

Sljedeće što je potrebno napraviti vezano je uz „mapiranje“ naziva varijabli označenih s „@FXML“ anotacijom s identifikatorima na grafičkom sučelju „knjige.fxml“ (slika 6):



Slika 6. Mapiranje naziva varijabli

Naziv varijable „nazivKnjige“ unutar klase „KnjigeController“ mora se podudarati s poljem „fx:id“ u „Code“ modulu unutar „Scene Buildera“. Na sličan način je potrebno povezati naziv tablice i kolona u tablici.

Unutar metode „initialize“ obavljena je inicijalizacija vrijednosti koje će se prikazivati u tablici u određenom stupcu iz određene varijable unutar klase „Knjiga“. Na primjer, vrijednost varijable „naziv“ iz klase „Knjiga“ prikazivati će se unutar stupca pod nazivom „nazivKnjigeColumn“. Na kraju je potrebno povezati gumb „Dohvati knjige“ s pozivom metode „prikaziSveKnjige“.

Prikaz moguće implementacije klase „KnjigeController“ prikazan je u nastavku:

```
public class KnjigeController {

    public KnjigeController() {
    }

    @FXML
    private TextField nazivKnjige;

    @FXML
    private TableView<Knjiga> knjigaTable;

    @FXML
    private TableColumn<Knjiga, String> nazivKnjigeColumn;

    @FXML
    private TableColumn<Knjiga, VrstaPublikacije> vrstaKnjigeColumn;

    @FXML
    private TableColumn<Knjiga, Integer> godinaIzdanjaKnjigeColumn;
```

```
    @FXML
    private TableColumn<Knjiga, Integer> brojStranicaKnjigeColumn;

    @FXML
    private TableColumn<Knjiga, Jezik> jezikKnjigeColumn;

    @FXML
    private TableColumn<Knjiga, String> nazivIzdavacaKnjigeColumn;

    @FXML
    public void initialize() {
          nazivKnjigeColumn.setCellValueFactory(
                      new PropertyValueFactory<Knjiga, String>("naziv"));
          vrstaKnjigeColumn.setCellValueFactory(
new PropertyValueFactory<Knjiga, VrstaPublikacije>("vrstaPublikacije"));
          godinaIzdanjaKnjigeColumn.setCellValueFactory(
new PropertyValueFactory<Knjiga, Integer>("godinaIzdanja"));
          brojStranicaKnjigeColumn.setCellValueFactory(
new PropertyValueFactory<Knjiga, Integer>("brojStranica"));
          jezikKnjigeColumn.setCellValueFactory(
new PropertyValueFactory<Knjiga, Jezik>("jezik"));
          nazivIzdavacaKnjigeColumn.setCellValueFactory(
new PropertyValueFactory<Knjiga, String>("izdavac"));
    }

    public void prikaziSveKnjige() {
          List<Knjiga> knjige = dohvatiKnjige();

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

          if (nazivKnjige.getText().isEmpty() == false) {
                filtriraneKnjige = knjige.stream().filter(p -> p.getNaziv()
                  .contains(nazivKnjige.getText()))
                .collect(Collectors.toList());
          } else {
                filtriraneKnjige = knjige;
          }
          ObservableList<Knjiga> listaKnjiga =
             FXCollections.observableArrayList(filtriraneKnjige);
          knjigaTable.setItems(listaKnjiga);
    }

    public List<Knjiga> dohvatiKnjige() {
          List<Knjiga> listaKnjiga = new ArrayList<Knjiga>();
          BufferedReader reader = null;
          String fileName = "knjige.txt";
          try {
                reader = new BufferedReader(new FileReader(new File(fileName)));
          } catch (FileNotFoundException e) {
                e.printStackTrace();
          }

          Knjiga knjiga = null;
          while (true) {
                try {
                      knjiga = ucitajKnjigu(reader);
                } catch (IOException e) {
                      e.printStackTrace();
```

```
                }

                if(knjiga == null) {
                        break;
                }
                listaKnjiga.add(knjiga);
        }
        return listaKnjiga;
    }

    private final Knjiga ucitajKnjigu(BufferedReader reader) throws IOException {
        String nazivKnjige = reader.readLine();

        if(nazivKnjige == null) {
                return null;
        }

        Jezik odabraniJezik = null;
        Integer uneseniJezik = Integer.parseInt(reader.readLine());

        for(Jezik jezik : Jezik.values()) {
                if(uneseniJezik == jezik.getKod()) {
                        odabraniJezik = jezik;
                        break;
                }
        }

        Izdavac noviIzdavac = unesiIzdavaca(reader);
        String godinaIzdavanjaString = reader.readLine();
        Integer godinaIzdavanja = Integer.parseInt(godinaIzdavanjaString);

        VrstaPublikacije vrstaPublikacije = null;

        String vrstaPublikacijeUnos = reader.readLine();

        if(VrstaPublikacije.ELEKTRONICKA.getKod() ==
             Integer.parseInt(vrstaPublikacijeUnos)) {
              vrstaPublikacije = VrstaPublikacije.ELEKTRONICKA;
        }
        else if (VrstaPublikacije.PAPIRNATA.getKod() ==
                    Integer.parseInt(vrstaPublikacijeUnos)){
              vrstaPublikacije = VrstaPublikacije.PAPIRNATA;
        }

        String brojStranicaString = reader.readLine();
        Integer brojStranica = Integer.parseInt(brojStranicaString);
        Knjiga novaKnjiga = new Knjiga(nazivKnjige, odabraniJezik, noviIzdavac,
godinaIzdavanja, vrstaPublikacije, brojStranica);

        return novaKnjiga;
    }

    private final Izdavac unesiIzdavaca(BufferedReader reader) throws IOException {
        String nazivIzdavaca = reader.readLine();
        String drzavaIzdavaca = reader.readLine();
        Izdavac noviIzdavac = new Izdavac(nazivIzdavaca, drzavaIzdavaca);
        return noviIzdavac;
    }
}
```

10. Na sličan način kao što je prikazano za knjige, potrebno je implementirati dohvat i filtriranje podataka o časopisima.

11. Na sličan način kao što je prikazano za knjige, potrebno je implementirati dohvat i filtriranje podataka o članovima.

12. Slično kao u trećoj laboratorijskoj vježbi unutar „pom.xml“ datoteke potrebno je kreirati novu ovisnost (Engl. *Dependency*) za dijaloge koji će biti korišteni u ovoj laboratorijskoj vježbi. Više o „controlsfx“ biblioteci moguće je naći među napomenama ove laboratorijske vježbe. Ovisnost koju je potrebno dodati slijedi u nastavku:

```
<dependency>
	<groupId>org.controlsfx</groupId>
	<artifactId>controlsfx</artifactId>
	<version>8.20.8</version>
</dependency>
```

13. Postojeće metode za dohvat podataka koje su se do sada nalazile u „Controller“ klasama potrebno je premjestiti u vanjsku klasu imena „Datoteke“ unutar novog paketa „hr.tvz.java.vjezbe.baza“. Metode je potrebno kreirati statičkima i povezati s postojećim metodama kako bi iste ispravno obavljale dohvat podataka.

14. Unutar klase „Datoteke“ potrebno je kreirati nove metode za pohranu liste knjiga, liste časopisa i liste članova u datoteke. Datoteku je svaki puta potrebno preplaviti novim podacima iz liste. Potrebno je obratiti pozornost da zapisivanje radi kao i čitanje kako ne bi došlo do pogreške prilikom čitanja iz datoteka.

15. Unutar svake od enumeracija „Jezik“ i „VrstaPublikacije“ potrebno je kreirati nove statičke metode naziva „vrijednosti“ koje će vratiti listu stringova jezika odnosno vrste publikacije kako bi se iste što jednostavnije implementirale za prikaz u „ComboBox“ kontroli. Metoda „getValues()“ unutar svake enumeracije vraća niz vrijednosti enumeracije.

16. Potrebno je „css“ dokument aplikacije dopuniti s dva nova svojstva koja su dana u nastavku, a koristit će se prilikom prikazivanja grešaka na kontrolama.

```
.text-field.error { -fx-border-color: red;  -fx-focus-color: red;}
.combo-box.error { -fx-border-color: red;  -fx-focus-color: red;}
```

17. Potrebno je kreirati novi paket naziva „hr.tvz.java.vjezbe.controller.base“ unutar kojeg je potrebno kreirati novu apstraktnu klasu „UrediBase“ koja će služiti kao bazna klasa za „controller“ klase koje će biti kreirane u ovoj laboratorijskoj vježbi.

Unutar klase potrebno je kreirati dvije privatne metode koje će se koristiti prilikom promjene dizajna kontrole pomoću „css“ dokumenta kreiranog u prethodnom koraku. Ovisno o tome je li kontrola ispravno popunjena ili nije, potrebno je dodavati i uklanjati „css“. Primjer implementacije te dvije metode dan je u nastavku:

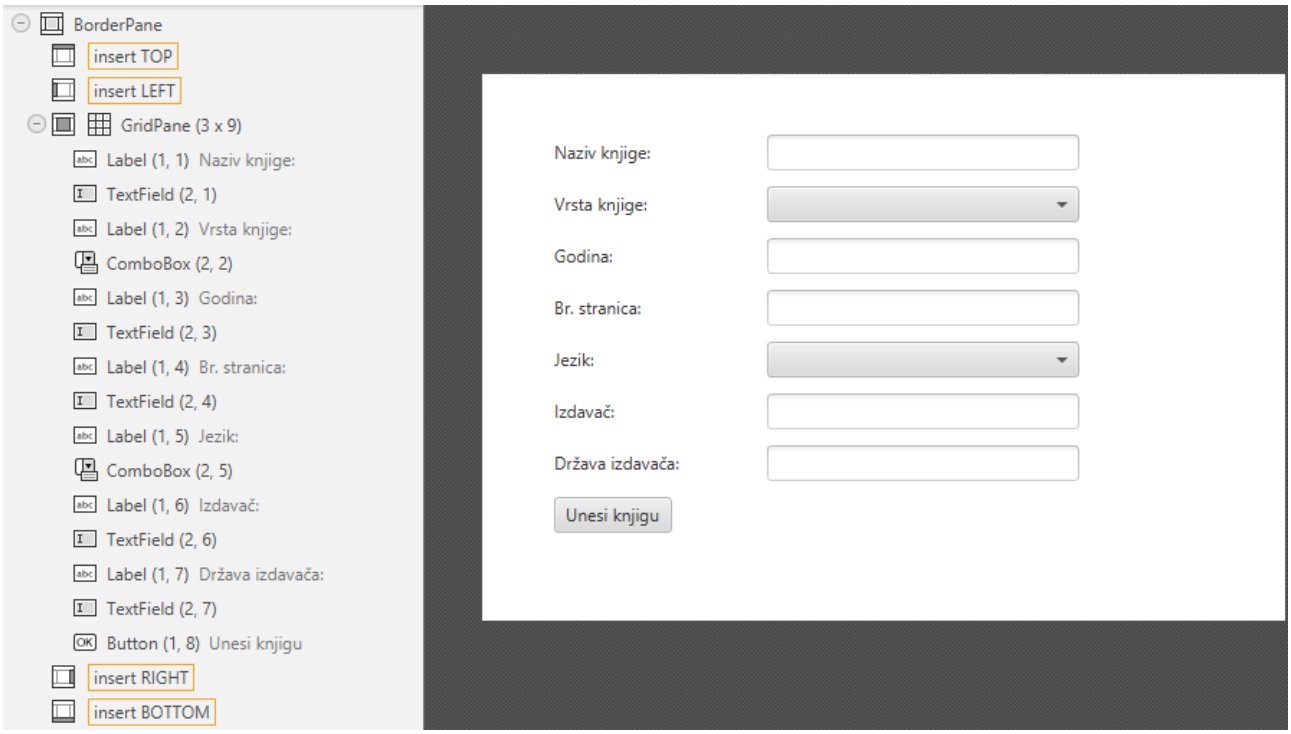
```
private void prikaziGresku(Control ctl, String message){
	ctl.getStyleClass().add("error");
	ctl.setTooltip(new Tooltip(message));
}
private void ukloniGresku(Control ctl){
	ctl.getStyleClass().remove("error");
	ctl.setTooltip(null);
}
```

Unutar klase potrebo je kreirati dvije „protected“ metode koje će testirati vrijednosti unutar kontrola. Jedna metoda mora provjeravati sadrži li kontrola vrijednost, a druga mora provjeravati sadrži li kontrola vrijednost (pozivom date metode) i sadrži li samo isključivo brojeve. Primjer kontrole koja provjerava vrijednost dan je u nastavku:

```
protected boolean validirajVrijednost(Control ctl){
	ukloniGresku(ctl);
	if(ctl instanceof TextField){
		if(((TextField)ctl).getText() != null &&
!((TextField)ctl).getText().equals(""))
			return true;
	}else if(ctl instanceof ComboBox<?>){
		if(((ComboBox<?>)ctl).getValue() != null)
			return true;
	}
	prikaziGresku(ctl, "Potrebno je unijeti vrijednost!");
	return false;
}
```

Prilikom provjeravanja sadrži li kontrola samo brojeve potrebno je voditi računa o tome da će se raditi samo o „TextField“ tipu kontrole i potrebno je rješenje napraviti što elegantnijim primjerice uz korištenje regularnih izraza (Engl. *Regular expressions*) koji su ugrađeni u Javu.

18. Unutar paketa „hr.tvz.java.vjezbe.javafx“ potrebno je kreirati novu FXML datoteku s nazivom „knjigaUredi.fxml“ koja će služiti dodavanju nove knjige u datoteku i promjeni postojeće. Izgled datoteke pomoću „Scene builder“ alata vidljiv je na slici 1.



Slika 7. Izgled datoteke „knjigaUredi.fxml“

19. Unutar paketa „hr.tvz.java.vjezbe.controller“ potrebno je kreirati „Controller“ klasu za uređivanje knjiga s nazivom „KnjigaUrediController“ i povezati je s „knjigaUredi.fxml“ datotekom. Prikaz moguće implementacije „Controller“ klase „KnjigaUrediController“ dan je u nastavku:

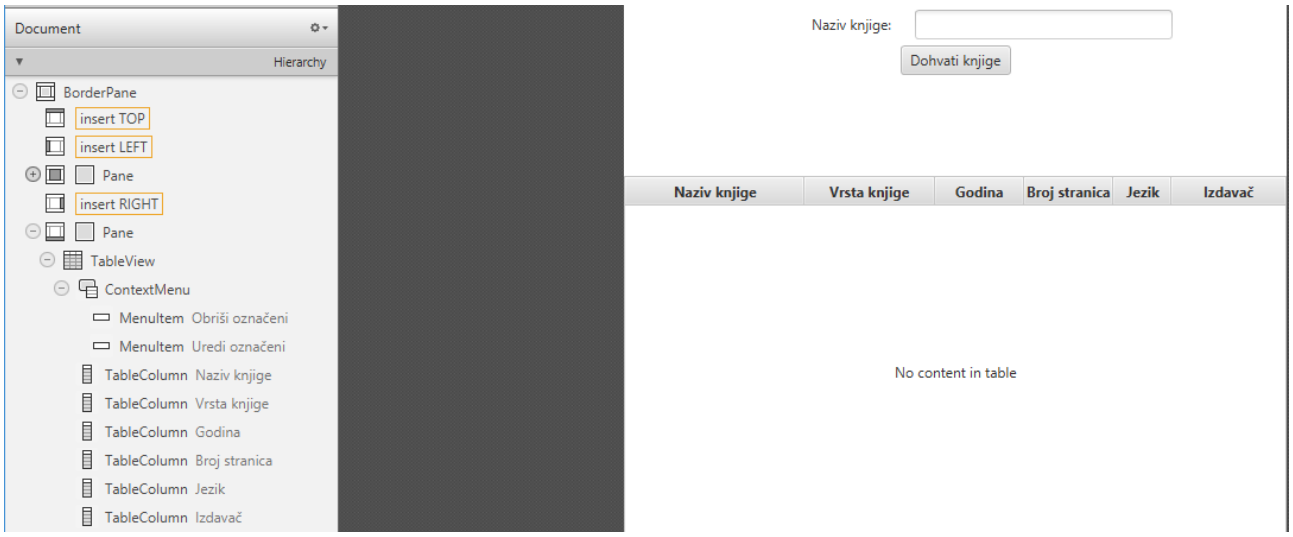
```
package hr.tvz.java.vjezbe.controller;
import hr.tvz.java.vjezbe.baza.Datoteke;
import hr.tvz.java.vjezbe.controller.base.UrediBase;
import hr.tvz.java.vjezbe.entitet.Izdavac;
import hr.tvz.java.vjezbe.entitet.Knjiga;
import hr.tvz.java.vjezbe.enumeracija.Jezik;
import hr.tvz.java.vjezbe.enumeracija.VrstaPublikacije;
import java.math.BigDecimal;
import java.util.List;
import org.controlsfx.dialog.Dialogs;
import javafx.fxml.FXML;
import javafx.scene.control.ComboBox;
import javafx.scene.control.TextField;

@SuppressWarnings("deprecation")
public class KnjigaUrediController extends UrediBase {
	@FXML
	private TextField nazivKnjiga;
	@FXML
	private ComboBox<String> vrstaKnjige;
	@FXML
	private TextField godinaKnjige;
	@FXML
	private TextField brStranicaKnjige;
	@FXML
```

```
	private ComboBox<String> jezikKnjige;
	@FXML
	private TextField nazivIzdavaca;
	@FXML
	private TextField drzavaIzdavaca;
	private boolean isEdit;
	private Knjiga zaPrikaz;
	private List<Knjiga> knjige;
	public KnjigaUrediController() {}

	public void urediParametri(List<Knjiga> knjige, Knjiga zaPrikaz) {
		this.zaPrikaz = zaPrikaz;
		this.knjige = knjige;
		isEdit = true;
		nazivKnjiga.setText(zaPrikaz.getNaziv());
		vrstaKnjige.setValue(zaPrikaz.getVrsta().toString());
		godinaKnjige.setText(zaPrikaz.getGodina() + "");
		brStranicaKnjige.setText(zaPrikaz.getBrStranica() + "");
		jezikKnjige.setValue(zaPrikaz.getJezik().toString());
		nazivIzdavaca.setText(zaPrikaz.getIzdavac().getNaziv());
		drzavaIzdavaca.setText(zaPrikaz.getIzdavac().getDrzava());
	}
	@FXML
	public void initialize() {
		jezikKnjige.getItems().addAll(Jezik.vrijednosti());
		vrstaKnjige.getItems().addAll(VrstaPublikacije.vrijednosti());
	}
	@FXML
	private void unesiKnjigu() {
		List<Knjiga> knjige = null;

		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;
		}
		if (isEdit) {
			knjige = this.knjige;
			knjige.remove(zaPrikaz);
		} else
			knjige = Datoteke.dohvatiKnjige();

		Jezik jezik = Jezik.valueOf(jezikKnjige.getValue());
		float cijenaStranice = (jezik == Jezik.HRVATSKI) ? 0.45f : 0.75f;
		knjige.add(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)));

		Datoteke.unesiKnjige(knjige);
```

```
            Dialogs.create().title("Informacija")
                        .message("Knjiga je uspješno unesena").showInformation();
      }
}
```

20. Unutar „glavna.fxml“ datoteke unutar menija „Knjige“ potrebno je dodati novu opciju „Nova knjiga“ koja će pozivati „knjigaUredi.fxml“ datoteku slično kao u točki sedam iz prošle laboratorijske vježbe.

21. Unutar datoteke „knjige.fxml“ potrebno je na „TableView“ komponentu dodati „ContextMenu“ komponentu s opcijama brisanja i izmjene. Komponentu „ContextMenu“ najjednostavnije je dodati na način da se ona samo „odvuče“ na „TableView“ komponentu. Nakon navedenog, izgled bi trebao biti kao na slici 2.



Slika 8. Izgled „ContextMenu“ komponente

22. Opcije izbornika iz prošlog koraka (brisanje i promjenu podataka) potrebno je implementirati u „Controller“ klasi „KnjigeController“ na način koji je pokazan u nastavku:

```
public void obrisi() {
      Knjiga c = knjigaTable.getSelectionModel().getSelectedItem();
      knjigaTable.getItems().remove(c);
      Datoteke.unesiKnjige(knjigaTable.getItems());
}
public void uredi() {
      try {
            FXMLLoader l = new FXMLLoader(getClass().getResource(
                        "../javafx/knjigaUredi.fxml"));
            BorderPane root = (BorderPane)l.load();
            KnjigaUrediController cont = l
                        .<KnjigaUrediController> getController();
            cont.urediParametri(knjigaTable.getItems(), knjigaTable
                        .getSelectionModel().getSelectedItem());
            JavaFXGlavna.setCenterPane(root);
      } catch (IOException e) {
            e.printStackTrace();
```

```
        }
}
```

23. Na sličan način kao što je prikazano za knjige, potrebno je implementirati dodavanje, promjenu i brisanje podataka o časopisima i članovima.

NAPOMENE:

1. Sve detalje u uputama koji nisu definirani moguće je proizvoljno definirati.

2. Nakon promjene parametara unutar Scene Buildera poželjno je spremiti njegove promjene, te izvršiti naredbu „Refresh“ nad projektom unutar Eclipsea kako bi se promjene ažurirale.

3. Tijekom razvoja poželjno je koristiti upute „manuala“ s mrežnih stranica: http://code.makery.ch/java/javafx-8-tutorial-intro/.

4. Za korištenje Scene Buildera poželjno je koristiti „manual“ sa sljedećih mrežnih stranica: https://docs.oracle.com/javase/8/scene-builder-2/get-started-tutorial/jfxsb-get_started.htm#JSBGS101

5. Aplikaciju je po želji moguće ukrasiti korištenjem CSS atributa kako je prikazano na predavanjima.