

Instrukcja tworzenia CRUDa JavaFX

PAWEŁ JADANOWSKI



Pobieramy

IntelliJ IDEA Community Edition

<https://www.jetbrains.com/idea/download/download-thanks.html?code=IIC>

Scene Buildera

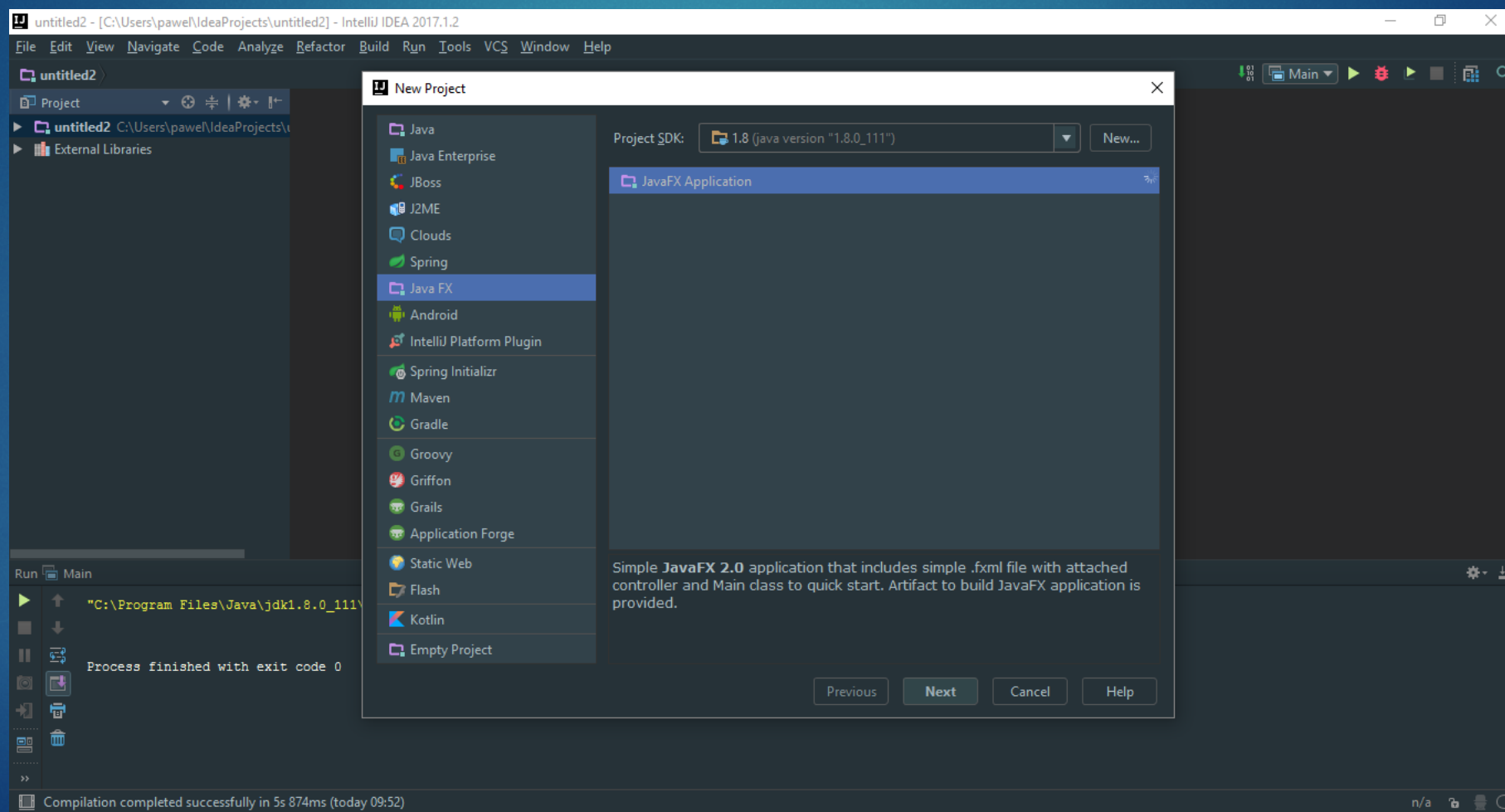
<http://gluonhq.com/products/scene-builder/>

JDBC SQLite Driver

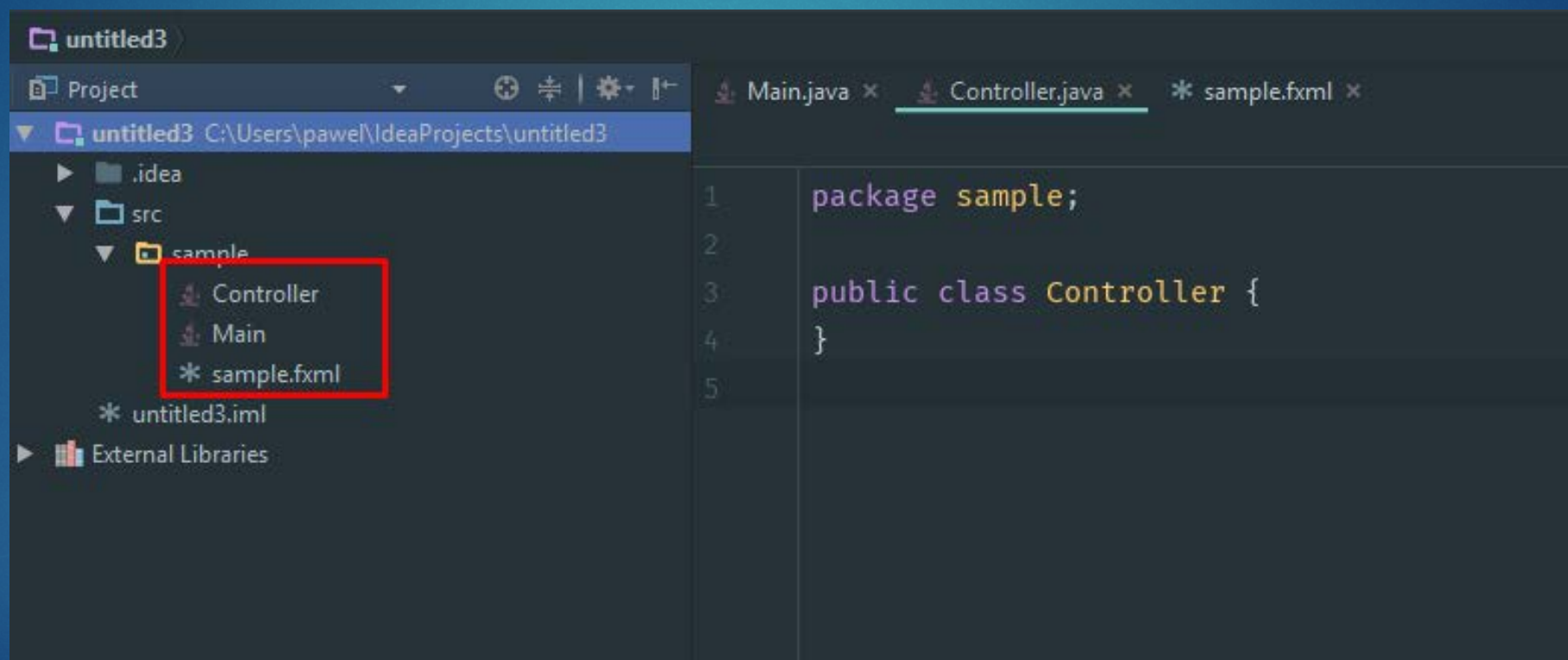
<https://bitbucket.org/xerial/sqlite-jdbc/downloads/>

Tworzenie nowego projektu JavaFX

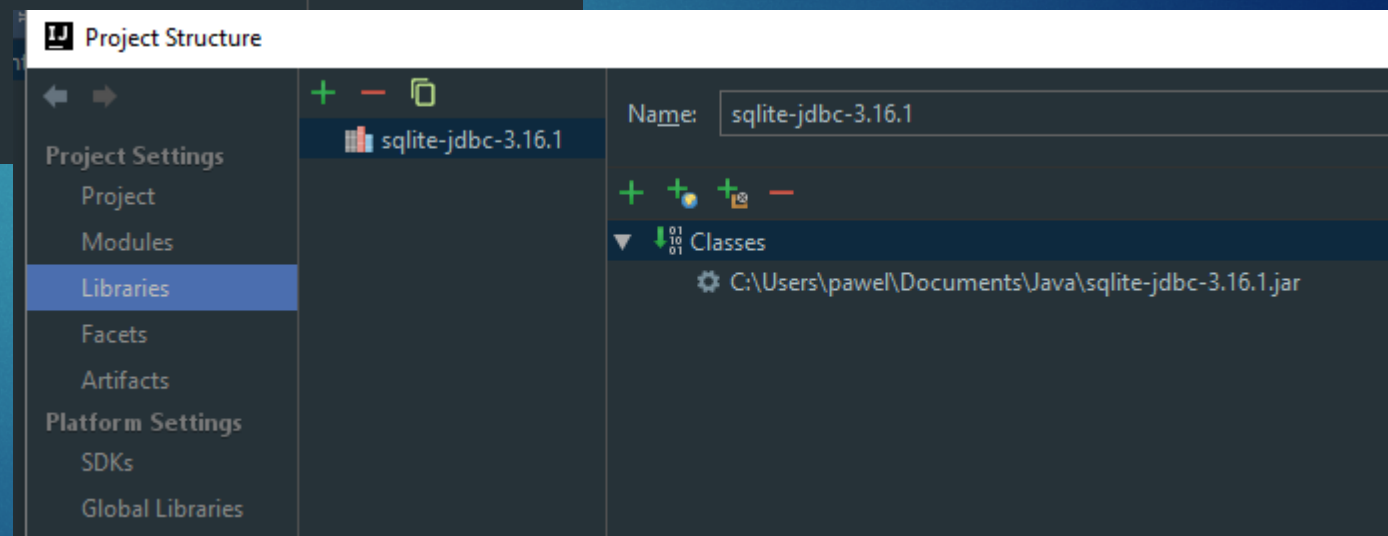
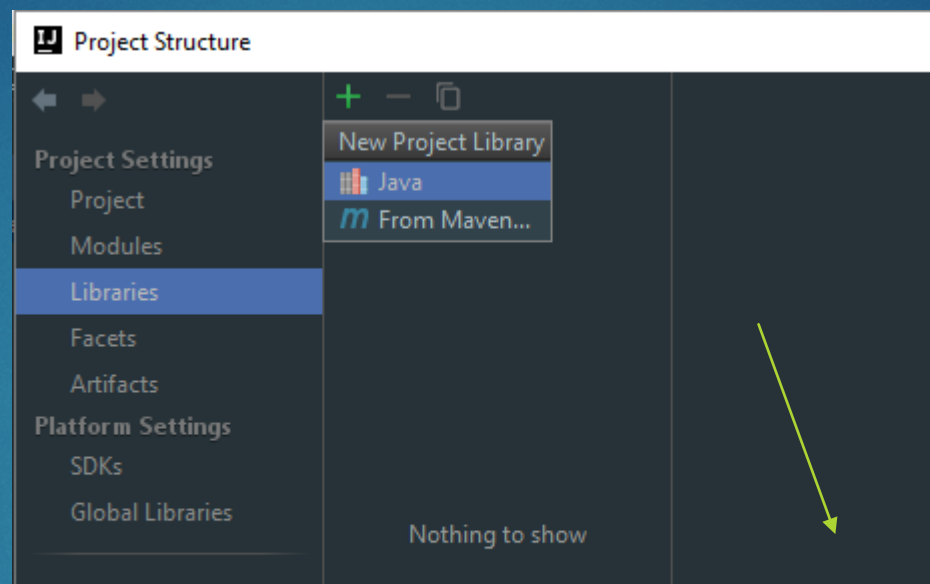
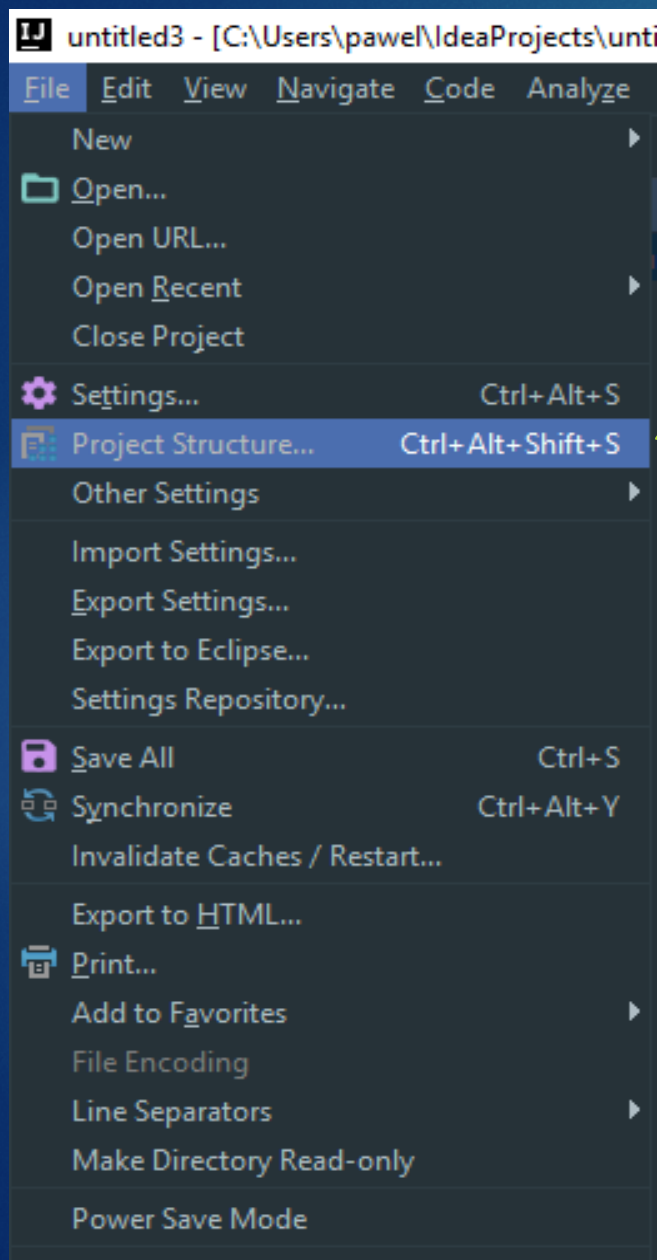
W IntelliJ Idea wybieramy
File → New → Project



Automatycznie zostają wygenerowane 3 pliki: Controller, Main, sample.fxml



W celu połączenia do bazy danych potrzebujemy podpiąć sterownik



Tworzymy klasę z połączeniem do bazy

```
dbConnection.java x
1  package sample;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.SQLException;
6
7  public class dbConnection {
8
9      private static final String SQLITECONN = "jdbc:sqlite:school.sqlite";
10
11  @ public static Connection getConnection() throws SQLException {
12
13      try {
14          Class.forName("org.sqlite.JDBC");
15          return DriverManager.getConnection(SQLITECONN);
16      } catch (ClassNotFoundException e) {
17          e.printStackTrace();
18      }
19      return null;
20  }
21 }
```


Podpinamy Scene buildera do programu

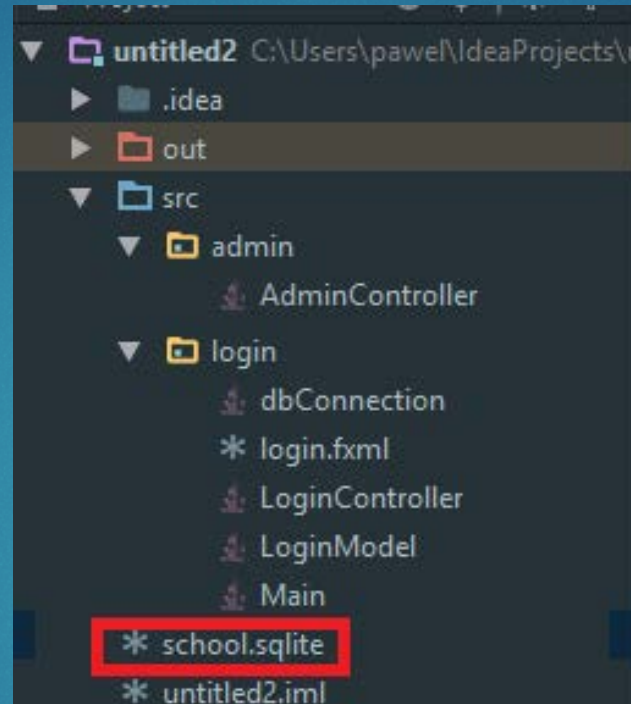
Po kliknięciu prawym na pliku *.fxml wybieramy
„Open in SceneBuilder”

Przy pierwszej próbie będziemy musieli podpiąć Scene Builder’a

Znajduje się on odpowiednio

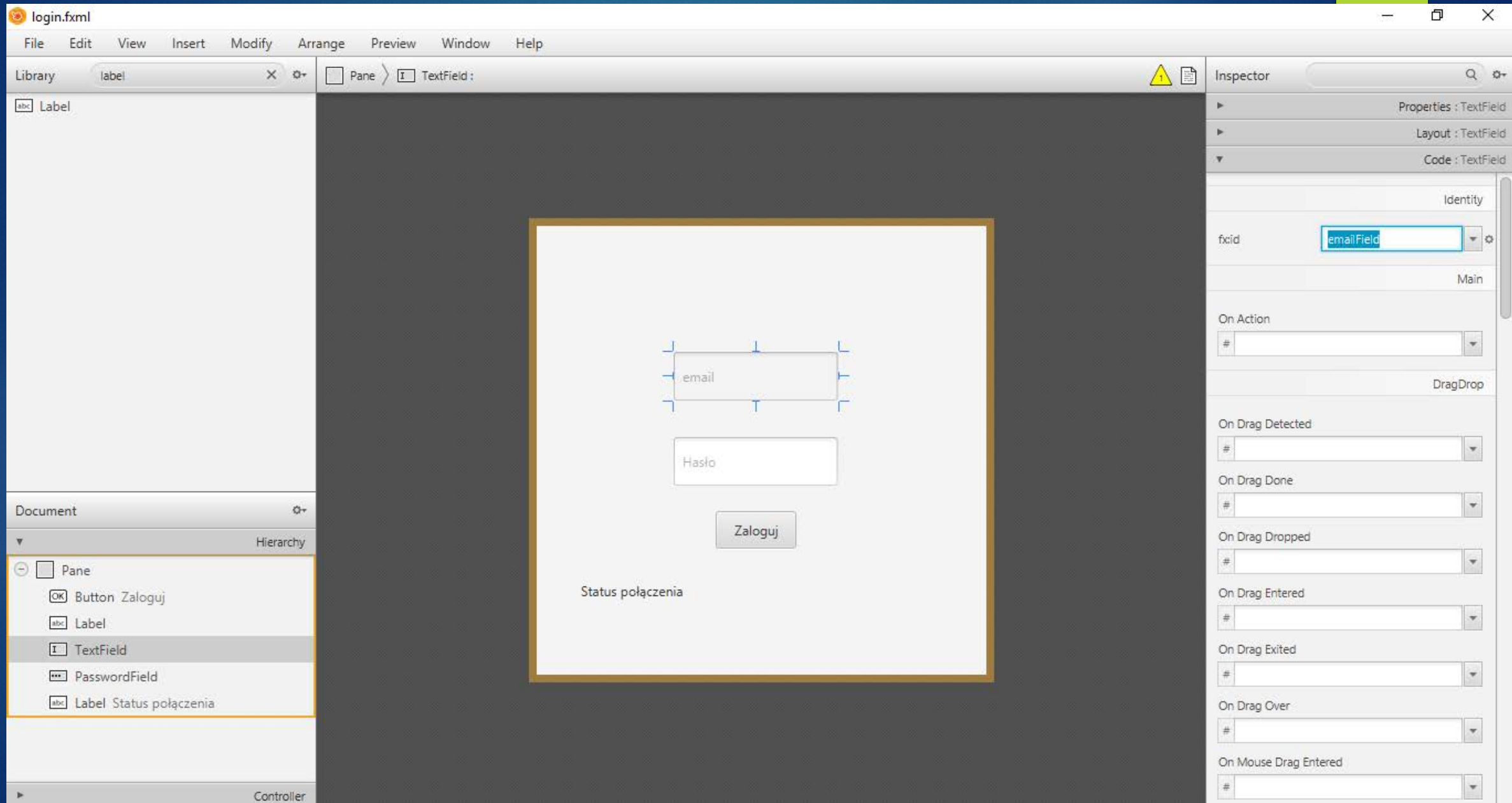
`C:\Users\Paweł\AppData\Local\SceneBuilder\SceneBuilder.exe`

Ostatecznie zmieniając nazwy na bardziej wymowne niż „sample” nasza struktura projektu wygląda tak

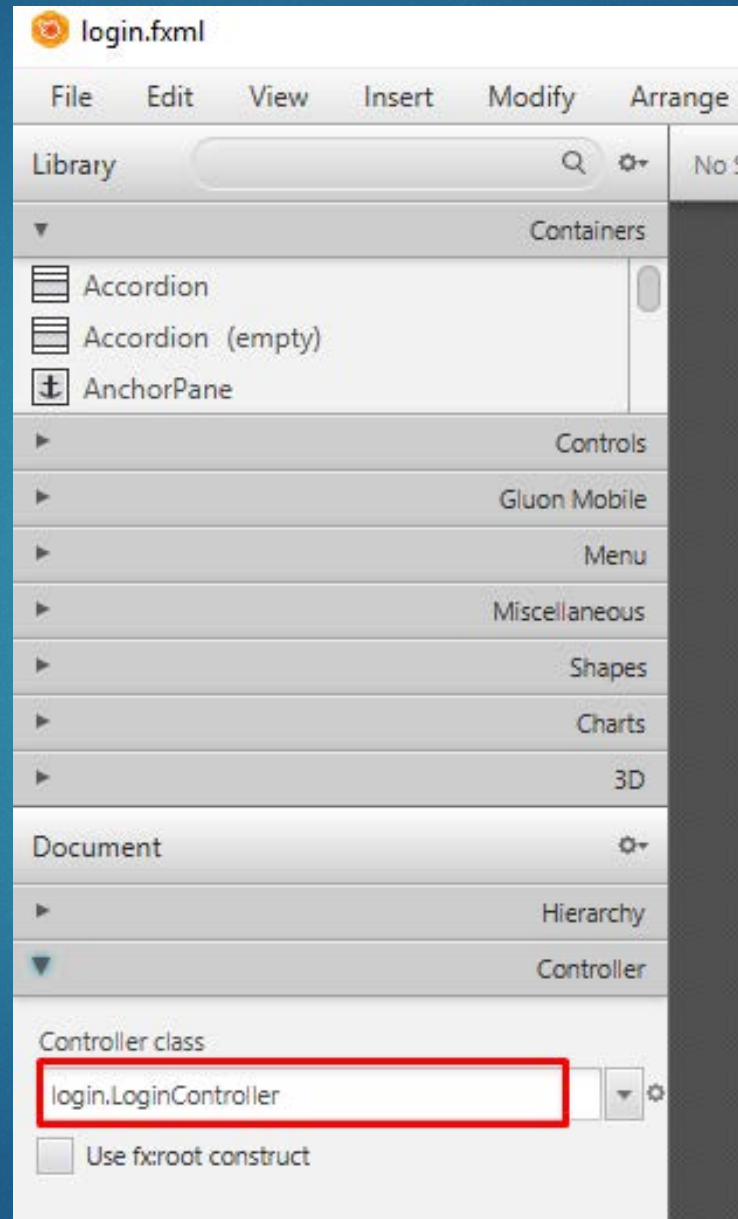


Warto zwrócić uwagę na to że plik z bazą danych został również dorzucony do projektu

Następnie tworzymy layout oraz każdemu elementowi przypisujemy unikalne fx:id



Podpinamy kontroler do pliku fxml



Następnie tworzymy klasę LoginModel



Tworzymy klasę LoginModel i piszemy poniższy kod

File - C:\Users\pawe\IdeaProjects\untitled2\src\sample>LoginModel.java

```
1 package sample;
2
3 import java.sql.Connection;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7
8 public class LoginModel {
9
10     Connection connection;
11
12     public LoginModel(){
13         try {
14             this.connection = dbConnection.getConnection();
15         } catch (SQLException e){
16             e.printStackTrace();
17         }
18
19         if (this.connection == null){
20             System.exit(1);
21         }
22     }
23
24     public boolean isDatabaseConnected(){
25         return this.connection != null;
26     }
27
28     public boolean checkLoginData(String user, String pass) throws Exception
29     {
30         PreparedStatement pr = null;
31         ResultSet result = null;
32
33         String sql = "SELECT * FROM students where email=? and password = ? ";
34
35         try {
36             pr = this.connection.prepareStatement(sql);
37             pr.setString(1, user);
38             pr.setString(2, pass);
39
40             result = pr.executeQuery();
41
42             if (result.next()){
43                 return true;
44             } else return false;
45         }
46         catch (SQLException e){
47             return false;
48         }
49         finally {
50             pr.close();
51             result.close();
52         }
53     }
54
55 }
56
57 }
```


oraz tworzymy klasę LoginController



```

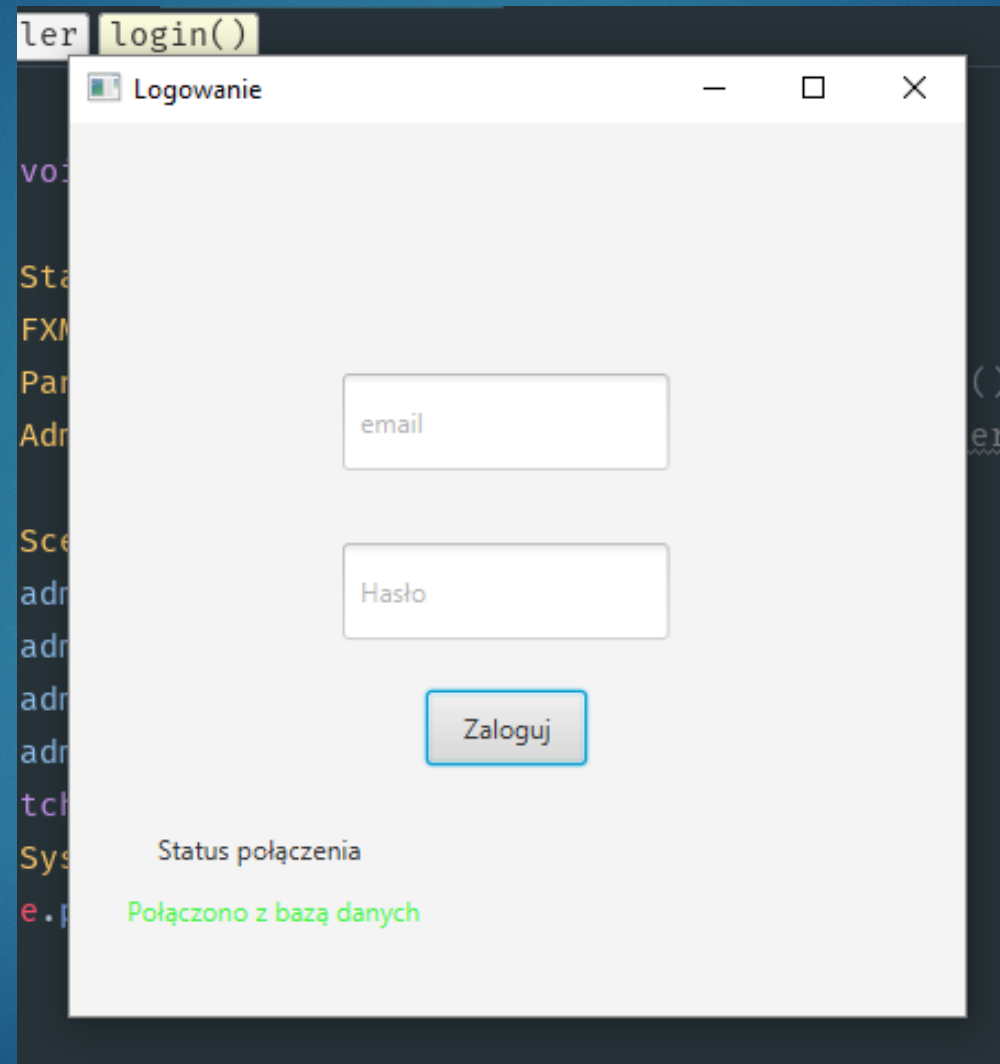
1 package login;
2
3 import admin.AdminController;
4 import javafx.event.ActionEvent;
5 import javafx.fxml.FXML;
6 import javafx.fxml.FXMLLoader;
7 import javafx.fxml.Initializable;
8 import javafx.scene.Scene;
9 import javafx.scene.control.*;
10 import javafx.scene.layout.Pane;
11 import javafx.scene.paint.Color;
12 import javafx.stage.Stage;
13
14 import java.io.IOException;
15 import java.net.URL;
16 import java.util.ResourceBundle;
17
18 public class LoginController implements Initializable {
19
20
21     LoginModel loginModel = new LoginModel();
22     @FXML
23     private Label dbStatus, loginStatusLbl;
24     @FXML
25     private TextField emailField;
26     @FXML
27     private PasswordField passwordField;
28     @FXML
29     private Button loginButton;
30
31     public void initialize(URL url, ResourceBundle rb) {
32         if (this.loginModel.isDatabaseConnected()) {
33             this.dbStatus.setText("Połączono z bazą danych");
34             this.dbStatus.setTextFill(Color.web("#42f445"));
35         } else {
36             this.dbStatus.setText("Brak połączenia z bazą danych");
37             this.dbStatus.setTextFill(Color.web("#f44141"));
38         }
39     }
40
41 }
42
43
44 @FXML
45 public void login(ActionEvent event) {
46     try {
47         if (this.loginModel.checkLoginData(
48             this.emailField.getText(),
49             this.passwordField.getText()
50         )) {
51             Stage stage = (Stage) this.loginButton.getScene().getWindow();
52             stage.close();
53
54             adminLogin();
55
56         } else {
57             this.loginStatusLbl.setText("Błędne dane");
58         }
59     } catch (Exception e) {
60         e.printStackTrace();
61     }
62
63
64
65 }

```



```
66
67     private void adminLogin() {
68         try{
69             Stage adminStage = new Stage();
70             FXMLLoader adminLoader = new FXMLLoader();
71             Pane adminroot = (Pane)adminLoader.load(getClass().getResource("/
admin/adminFXML.fxml").openStream());
72             AdminController adminController = (AdminController)adminLoader.
getController();
73
74             Scene scene = new Scene(adminroot);
75             adminStage.setScene(scene);
76             adminStage.setTitle("Panel Admina");
77             adminStage.setResizable(false);
78             adminStage.show();
79         } catch (IOException e) {
80             System.out.println("AdminLogin() ex");
81             e.printStackTrace();
82         }
83     }
84 }
85
86 }
87
```

Pozostało już uruchomić program



The image shows a screenshot of a web browser window with a tab labeled 'login()'. In the foreground, a modal dialog box titled 'Logowanie' (Login) is displayed. The dialog has a white background and a standard Windows-style title bar with minimize, maximize, and close buttons. It contains two text input fields: the first is labeled 'email' and the second is labeled 'Hasło' (Password). Below these fields is a button labeled 'Zaloguj' (Login). At the bottom of the dialog, there is a status section titled 'Status połączenia' (Connection status) which displays the message 'Połączono z bazą danych' (Connected to the database) in green text.

W pakiecie admin powinniśmy mieć teraz następujące klasy

- AdminController
- adminFXML.fxml

- Oraz w zależności od rodzaju przechowywanych tabel w bazie klasę np. StudentData dla tabeli ze studentami

Zajmijmy się najpierw wypełnieniem klasy StudentData

Tworzymy w niej pola odpowiadające rodzajom kolumn w danej tabeli. Przykładowy plik

```
1 package admin;
2
3
4 import javafx.beans.property.SimpleIntegerProperty;
5 import javafx.beans.property.SimpleStringProperty;
6
7 public class StudentData {
8
9     private final SimpleIntegerProperty ID;
10    private final SimpleStringProperty firstName;
11    private final SimpleStringProperty lastName;
12    private final SimpleStringProperty birthDay;
13    private final SimpleStringProperty email;
14    private final SimpleStringProperty password;
15
16    public StudentData(int id, String firstname, String lastname, String dob,
17String email, String pass) {
18        this.ID = new SimpleIntegerProperty(id);
19        this.firstName = new SimpleStringProperty(firstname);
20        this.lastName = new SimpleStringProperty(lastname);
21        this.birthDay = new SimpleStringProperty(dob);
22        this.email = new SimpleStringProperty(email);
23        this.password = new SimpleStringProperty(pass);
24    }
25
26    public int getID() {
27        return ID.get();
28    }
29
30    public SimpleIntegerProperty IDProperty() {
31        return ID;
32    }
33
34    public void setID(int ID) {
35        this.ID.set(ID);
36    }
37
38    public String getFirstName() {
39        return firstName.get();
40    }
41
42    public SimpleStringProperty firstNameProperty() {
43        return firstName;
44    }
45
46    public void setFirstName(String firstName) {
47        this.firstName.set(firstName);
48    }
49
50    public String getLastName() {
51        return lastName.get();
52    }
53
54    public SimpleStringProperty lastNameProperty() {
55        return lastName;
56    }
57
58    public void setLastName(String lastName) {
59        this.lastName.set(lastName);
60    }
61
62    public String getBirthDay() {
63        return birthDay.get();
64    }
```



```
65     public SimpleStringProperty birthDayProperty() {
66         return birthDay;
67     }
68
69     public void setBirthDay(String birthDay) {
70         this.birthDay.set(birthDay);
71     }
72
73     public String getEmail() {
74         return email.get();
75     }
76
77     public SimpleStringProperty emailProperty() {
78         return email;
79     }
80
81     public void setEmail(String email) {
82         this.email.set(email);
83     }
84
85     public String getPassword() {
86         return password.get();
87     }
88
89     public SimpleStringProperty passwordProperty() {
90         return password;
91     }
92
93     public void setPassword(String password) {
94         this.password.set(password);
95     }
96 }
97
```

Następnie edytujemy plik adminFXML w Scene Builderze i tworzymy layout

Ważną rzeczą jest to aby do każdego elementu przypisać unikalne fx:id

Dotyczy to pól tekstowych, tabeli, kolumn tabeli, przycisków etc

The screenshot displays the JavaFX Scene Builder interface. The central canvas shows a window titled "Zarządzanie uczniami" with a tabbed interface. The "uczniowie" tab is active, showing a form with fields for ID, Imię, Nazwisko, Email, and Hasło, along with buttons for "Dodaj", "Wyczyść formularz", "Edytuj", "Usuń", and "Załaduj liste studentów". A table is also present, currently empty, with columns for ID, Imię, Nazwisko, Data urodzenia, and Email. The "ID" column header is highlighted with a red box.

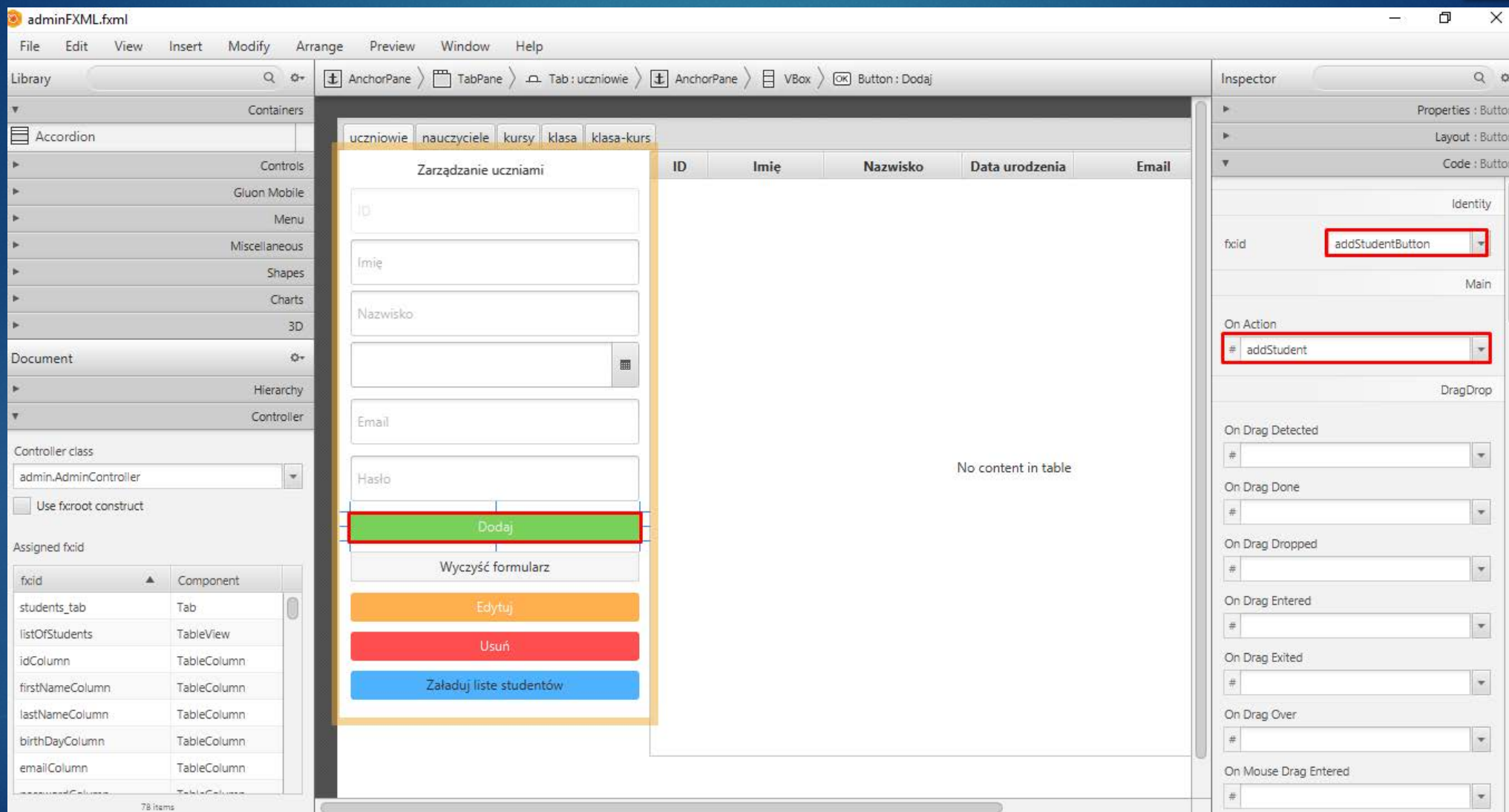
On the left, the "Library" pane shows various UI components. The "Assigned fx:id" table lists the following components:

fx:id	Component
students_tab	Tab
listOfStudents	TableView
idColumn	TableColumn
firstNameColumn	TableColumn
lastNameColumn	TableColumn
birthDayColumn	TableColumn
emailColumn	TableColumn

The "idColumn" entry is highlighted with a blue background and a red border.

On the right, the "Inspector" pane shows the properties for the selected "TableColumn". The "Identity" section shows "fx:id" set to "idColumn", which is also highlighted with a red box. The "Edit" section shows fields for "On Edit Start", "On Edit Commit", and "On Edit Cancel", each with a dropdown menu.

Pozostaje nam przypisać jeszcze do odpowiednich przycisków odpowiednie funkcje oraz napisać je w kontrolerze



Przykładowy AdminController =>

```

1 package admin;
2
3 import javafx.collections.FXCollections;
4 import javafx.collections.ListChangeListener;
5 import javafx.collections.ObservableList;
6 import javafx.event.ActionEvent;
7 import javafx.fxml.FXML;
8 import javafx.fxml.Initializable;
9 import javafx.scene.control.*;
10 import javafx.scene.control.cell.PropertyValueFactory;
11 import login.dbConnection;
12
13 import java.net.URL;
14 import java.sql.Connection;
15 import java.sql.PreparedStatement;
16 import java.sql.ResultSet;
17 import java.sql.SQLException;
18 import java.time.LocalDate;
19 import java.time.format.DateTimeFormatter;
20 import java.time.format.DateTimeParseException;
21 import java.util.Optional;
22 import java.util.ResourceBundle;
23
24 public class AdminController implements Initializable {
25
26     // dla studentow
27     @FXML
28     private TextField idField, firstNameField, lastNameField, emailField;
29     @FXML
30     private DatePicker birthDayField;
31     @FXML
32     private PasswordField passwordField;
33     @FXML
34     private TableView<StudentData> listOfStudents;
35
36     @FXML
37     private TableColumn<StudentData, String> idColumn, firstNameColumn, lastNameColumn;
38     @FXML
39     private TableColumn<StudentData, String> emailColumn, birthDayColumn, passwordColumn;
40
41     @FXML
42     private Button listAllStudentsBtn;
43
44     @FXML
45     Tab students_tab;
46
47     @FXML
48     private ObservableList<StudentData> studentList;
49
50
51     private dbConnection db;
52
53
54     @Override
55     public void initialize(URL location, ResourceBundle resources) {
56         this.db = new dbConnection();
57
58         ObservableList<StudentData> selectedRow = listOfStudents.getSelectionModel().
59         getSelectedItem();
60         selectedRow.addListener(new ListChangeListener() {
61             @Override
62             public void onChanged(Change c) throws NullPointerException {
63                 try {
64                     StudentData selectedR = listOfStudents.getSelectionModel().
65                     getSelectedItem();
66
67                     // zanim ustawi poprawne ID znajdz w tabeli
68                     String ssid = findProperID(); // napisz funkcję która pobiera id
69                     studenta np z ukrytego pola
70                     idField.setText(ssid);
71
72                     firstNameField.setText(selectedR.getFirstName());
73                     lastNameField.setText(selectedR.getLastName());
74

```



```

71
72         // ustawianie pola daty
73         DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.
yyyy");
74         try {
75             birthDayField.setValue(LocalDate.parse(selectedR.getBirthDay(),
formatter));
76         } catch (DateTimeParseException | NullPointerException e) {
77             birthDayField.setValue(null);
78         }
79
80         passwordField.setText(selectedR.getPassword());
81         emailField.setText(selectedR.getEmail());
82
83         } catch (NullPointerException e) {
84             idField.setText(null);
85             firstNameField.setText(null);
86             lastNameField.setText(null);
87             birthDayField.setValue(null);
88             passwordField.setText(null);
89             emailField.setText(null);
90         }
91     }
92 }
93 });
94
95 }
96
97
98 private int idCounter = 1;
99
100 @FXML
101 public void loadStudentData(ActionEvent actionEvent) {
102     String query = "SELECT * from students;";
103     try {
104
105         Connection conn = dbConnection.getConnection();
106         this.studentList = FXCollections.observableArrayList();
107
108         ResultSet rs = conn.createStatement().executeQuery(query);
109
110         while (rs.next()) {
111             this.studentList.add(new StudentData(
112                 idCounter++,
113                 rs.getString(2),
114                 rs.getString(3),
115                 rs.getString(4),
116                 rs.getString(5),
117                 rs.getString(6)
118             ));
119         }
120     } catch (SQLException er) {
121         er.printStackTrace();
122     }
123
124     this.idColumn.setCellValueFactory(new PropertyValueFactory<StudentData, String>(
"ID"));
125     this.firstNameColumn.setCellValueFactory(new PropertyValueFactory<StudentData,
String>("firstName"));
126     this.lastNameColumn.setCellValueFactory(new PropertyValueFactory<StudentData,
String>("lastName"));
127     this.birthDayColumn.setCellValueFactory(new PropertyValueFactory<StudentData,
String>("birthDay"));
128     this.emailColumn.setCellValueFactory(new PropertyValueFactory<StudentData,
String>("email"));
129     this.passwordColumn.setCellValueFactory(new PropertyValueFactory<StudentData,
String>("password"));
130
131     this.listOfStudents.setItems(null);
132     this.listOfStudents.setItems(this.studentList);
133
134     idCounter = 1;
135 }

```

```

136
137     @FXML
138     public void addStudent(ActionEvent actionEvent) {
139         String insertQuery = "insert into students(firstname, lastname, birthDay, email
140         , password) " +
141         "values (?, ?, ?, ?, ?)";
142         try {
143             Connection conn = dbConnection.getConnection();
144             PreparedStatement stm = conn.prepareStatement(insertQuery);
145
146             // walidacja danych
147             // funkcja containsDigit oraz isEmail do napisania własnoręcznie
148             if (containsDigit(this.firstNameField.getText())){
149                 Alert alert = new Alert(Alert.AlertType.ERROR);
150                 alert.setTitle("Niewłaściwe dane");
151                 alert.setHeaderText("Dane nie mogą zawierać cyfr");
152                 alert.showAndWait();
153             }
154
155             if (!isEmail(this.emailField.getText())){
156                 Alert alert = new Alert(Alert.AlertType.ERROR);
157                 alert.setTitle("Niewłaściwe dane");
158                 alert.setHeaderText("Email się nie zgadza");
159                 alert.showAndWait();
160             }
161
162             // można też pobrać id studenta z ukrytego pola
163             stm.setString(1, String.valueOf(this.idField.getText()));
164             stm.setString(2, String.valueOf(this.firstNameField.getText(
165             )));
166             stm.setString(3, String.valueOf(this.lastNameField.getText(
167             )));
168             stm.setString(4, String.valueOf(this.birthDayField.getEditor
169             ().getText()));
170             stm.setString(5, String.valueOf(this.emailField.getText()));
171             stm.setString(6, String.valueOf(this.passwordField.getText(
172             )));
173
174             stm.execute();
175             conn.close();
176
177         } catch (SQLException | NullPointerException e) {
178             e.printStackTrace();
179         }
180     }
181
182     @FXML
183     public void deleteStudent(ActionEvent event) {
184         StudentData getSelectedRow = listOfStudents.getSelectionModel().getSelectedItem(
185         );
186         try {
187             Connection conn = dbConnection.getConnection();
188
189             if (!getSelectedRow.toString().equals("")) {
190
191                 String fn = getSelectedRow.getFirstName();
192                 String ln = getSelectedRow.getLastName();
193                 String sID = null;
194
195                 PreparedStatement ps = conn.prepareStatement("SELECT student_id FROM
196                 students where firstName = ? AND lastName = ? ;");
197
198                 ps.setString(1, fn);
199                 ps.setString(2, ln);
200
201                 ResultSet res = ps.executeQuery();
202
203                 while (res.next()) {
204                     sID = res.getString(1);
205                 }
206                 System.out.println(sID);
207             }
208         }
209     }

```



```

202         String deleteQuery = "delete from students where student_id=" + idField.
        getText() + ";";
203
204         Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
205         alert.setTitle("Potwierdzenie usunięcia");
206         alert.setHeaderText("Chcesz usunąć studenta?");
207
208         alert.setContentText("'" + getSelectedRow.getFirstName() + " " +
        getSelectedRow.getLastName());
209
210         Optional<ButtonType> result = alert.showAndWait();
211         if (result.get() == ButtonType.OK) {
212             PreparedStatement stm = conn.prepareStatement(deleteQuery);
213             stm.execute();
214
215             listAllStudentsBtn.fire();
216
217         } else {
218             // ... user chose CANCEL or closed the dialog
219         }
220     }
221
222
223
224     conn.close();
225
226     } catch (SQLException e) {
227         e.printStackTrace();
228     }
229 }
230
231
232 @FXML
233 public void editStudent(ActionEvent event) {
234
235     String updateQuery = "update students set firstname=?, lastname=?, birthDay
    =? , email=?, password=?" +
    "where student_id = ?";
236
237     try {
238         Connection conn = dbConnection.getConnection();
239         PreparedStatement stm = conn.prepareStatement(updateQuery);
240
241         stm.setString(1, String.valueOf(AdminController.this.firstNameField.getText(
242 )))
243         stm.setString(2, String.valueOf(AdminController.this.lastNameField.getText(
244 )))
245         stm.setString(3, String.valueOf(AdminController.this.birthDayField.getEditor
    ().getText()));
246         stm.setString(4, String.valueOf(AdminController.this.emailField.getText()));
247         stm.setString(5, String.valueOf(AdminController.this.passwordField.getText(
248 )))
249         stm.setString(6, String.valueOf(AdminController.this.idField.getText()));
250
251         stm.execute();
252
253         conn.close();
254
255     } catch (SQLException | NullPointerException e) {
256         e.printStackTrace();
257     }
258 }
259
260
261 }
262

```

Oczywiście tak zrealizowany program ma parę „niedociągnięć” ale poprawę funkcjonalności zostawię już osobistym gustom ;)

Ważne, że tak zrealizowane zadanie „u mnie działa” ;)

Powodzenia