UNIWERSYTET RZESZOWSKI

WYDZIAŁ NAUK ŚCISŁYCH I TECHNICZNYCH

INSTYTUT INFORMATYKI

Bohdan Kudrenko

134935

Informatyka

Dokumentacja projektu

System zarządzania uniwersytetem w technologii Java

Praca projektowa

Praca wykonana pod kierunkiem

mgr inż. Ewa Żesławska

Rzeszów 2025

Spis treści

1. Struktura projektowej pracy z programowania obiektowego JAVA

1.1. Opis założeń projektu

Celem projektu było stworzenie aplikacji desktopowej wspierającej zarządzanie podstawowymi danymi w systemie uniwersyteckim. Głównym problemem, który projekt stara się rozwiązać, jest brak scentralizowanego i prostego w obsłudze narzędzia do administrowania informacjami o studentach i wykładowcach. Aplikacja ma na celu usprawnienie pracy administracyjnej poprzez automatyzację operacji CRUD (Create, Read, Update, Delete) na danych osobowych. Jest to istotne dla zapewnienia spójności i aktualności danych w uczelnianej bazie.

1.2. Opis struktury projektu

1.2.1. Wykorzystane technologie

Projekt został zrealizowany z wykorzystaniem następujących technologii:

- **Język programowania:** Java (JDK 11 lub nowszy) [4].
- **Interfejs graficzny użytkownika (GUI):** Java Swing — standardowa biblioteka do tworzenia aplikacji okienkowych w Javie [5].
- **Baza danych:** SQLite — lekki, plikowy silnik bazy danych, który nie wymaga oddzielnego serwera [7].
- **Sterownik bazy danych:** JDBC (Java Database Connectivity) do komunikacji między aplikacją a bazą danych [6].

1.2.2. Struktura bazy danych

Baza danych składa się z dwóch głównych tabel: ‘students‘ i ‘lecturers‘. Poniżej przedstawiono schematy SQL użyte do ich utworzenia.

```sql
CREATE TABLE IF NOT EXISTS students (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    first_name TEXT NOT NULL,
    last_name TEXT NOT NULL,
    email TEXT,
    major TEXT,
    graduation_end DATE
);
```

Listing 1.1. Schemat SQL tabeli ‘students‘

Źródło: Opracowanie własne na podstawie pliku `DatabaseConnector.java`.

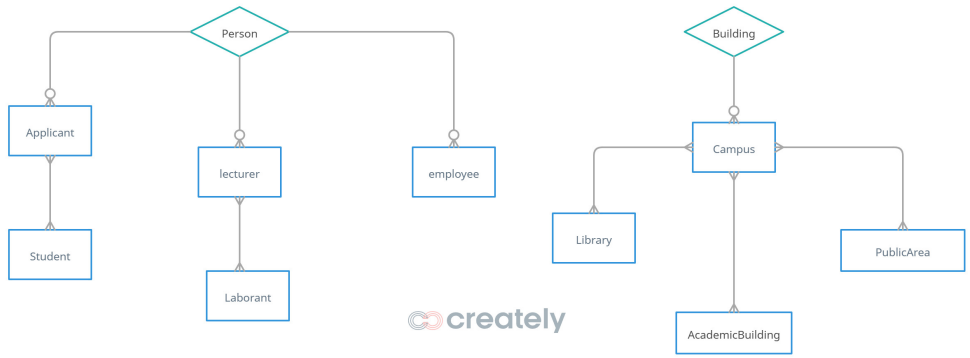
```sql
CREATE TABLE IF NOT EXISTS lecturers (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    first_name TEXT NOT NULL,
    last_name TEXT NOT NULL,
    email TEXT,
```

```
    position TEXT,
    academic_rank TEXT
);
```

Listing 1.2. Schemat SQL tabeli ‘lecturers‘

Źródło: Opracowanie własne na podstawie pliku `DatabaseConnector.java`.

Na rysunku 1.1 przedstawiono diagram ERD (Entity-Relationship Diagram) dla bazy danych.



Rysunek 1.1. Diagram ERD bazy danych
Źródło: Opracowanie własne.

1.2.3. Hierarchia klas

Projekt opiera się na zasadach programowania obiektowego [1]. Stworzono hierarchię klas, która odzwierciedla strukturę danych w systemie.

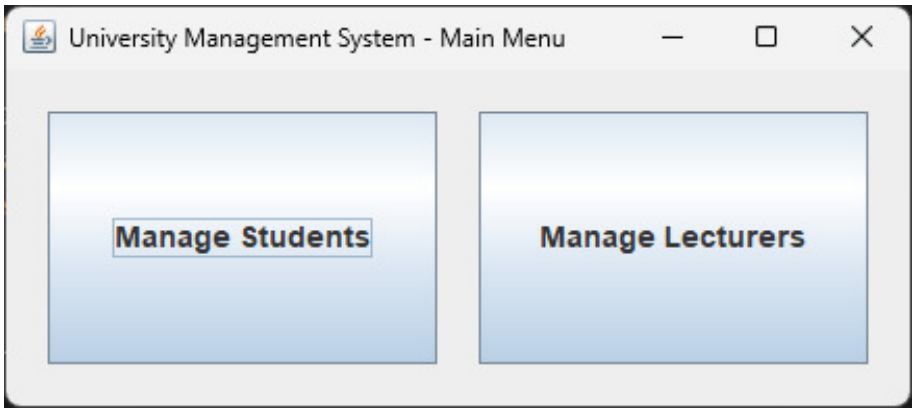
- **Person**: Abstrakcyjna klasa bazowa, która zawiera wspólne atrybuty dla wszystkich osób w systemie, takie jak ID, imię, nazwisko i email.
- **Student**: Klasa dziedzicząca po ‘Person‘, reprezentująca studenta. Dodatkowo zawiera pola ‘major‘ (kierunek studiów) i ‘graduationEnd‘ (data ukończenia studiów).
- **Employee**: Klasa dziedzicząca po ‘Person‘, reprezentująca pracownika. Dodano pole ‘position‘ (stanowisko).
- **Lecturer**: Klasa dziedzicząca po ‘Employee‘, reprezentująca wykładowcę. Rozszerza klasę pracownika o pole ‘academicRank‘ (stopień naukowy).
- **DAO (Data Access Object)**: W projekcie zastosowano wzorzec projektowy DAO [3, 2]. Interfejs ‘Dao<T>‘ definiuje podstawowe operacje bazodanowe, a jego implementacje ‘StudentDao‘ i ‘LecturerDao‘ zawierają logikę odpowiedzialną za komunikację z konkretnymi tabelami w bazie danych.

1.3. Grafik realizacji projektu

Realizacja projektu została zaplanowana i wykonana w określonych ramach czasowych. Podczas realizacji projektu napotkano pewne trudności, takie jak obsługa wyjątków SQL w kontekście interfejsu użytkownika oraz poprawne formatowanie daty między obiektami Javy a bazą danych. Problemy te zostały rozwiązane poprzez implementację globalnej obsługi wyjątków oraz zastosowanie klasy ‘LocalDate‘.

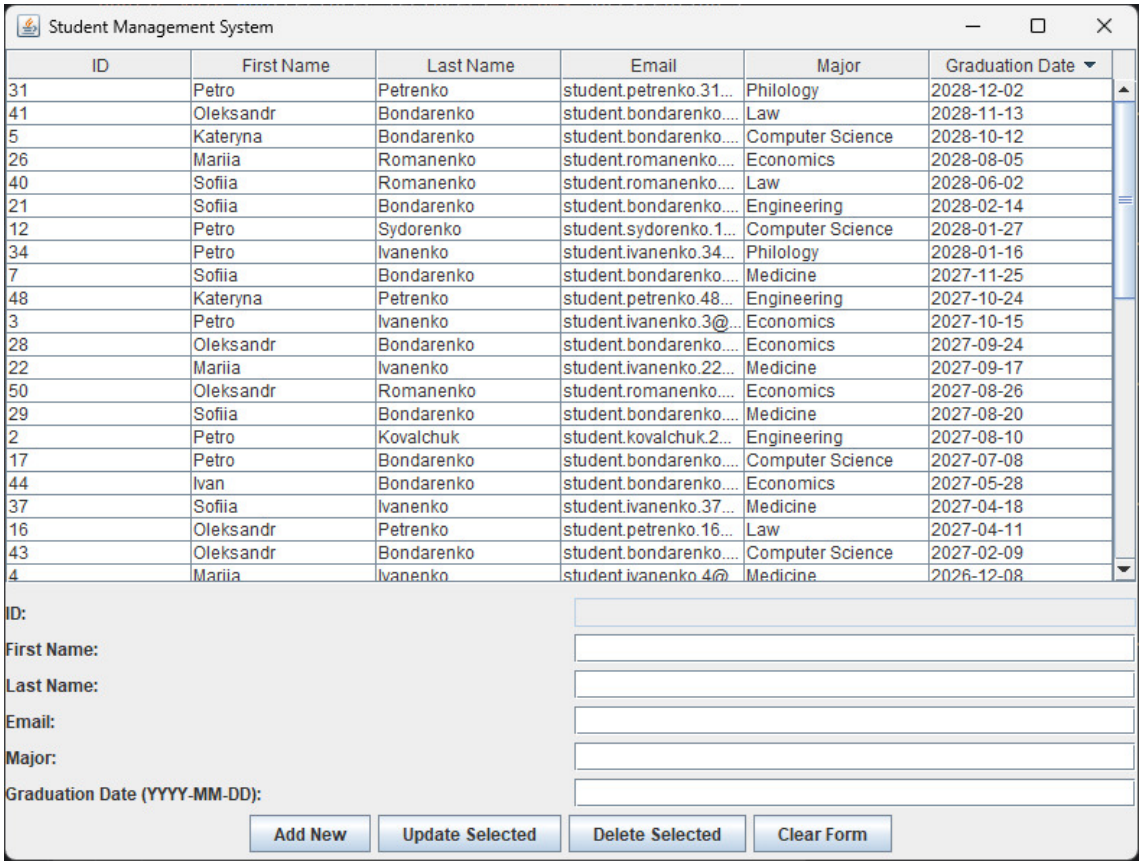
1.4. Prezentacja interfejsu użytkownika

Aplikacja posiada graficzny interfejs użytkownika (GUI) stworzony przy użyciu biblioteki Swing. Interfejs jest prosty i intuicyjny, co ułatwia zarządzanie danymi. Główne okno aplikacji (rys. 1.2) pozwala na wybór modułu do zarządzania — studentami lub wykładowcami.



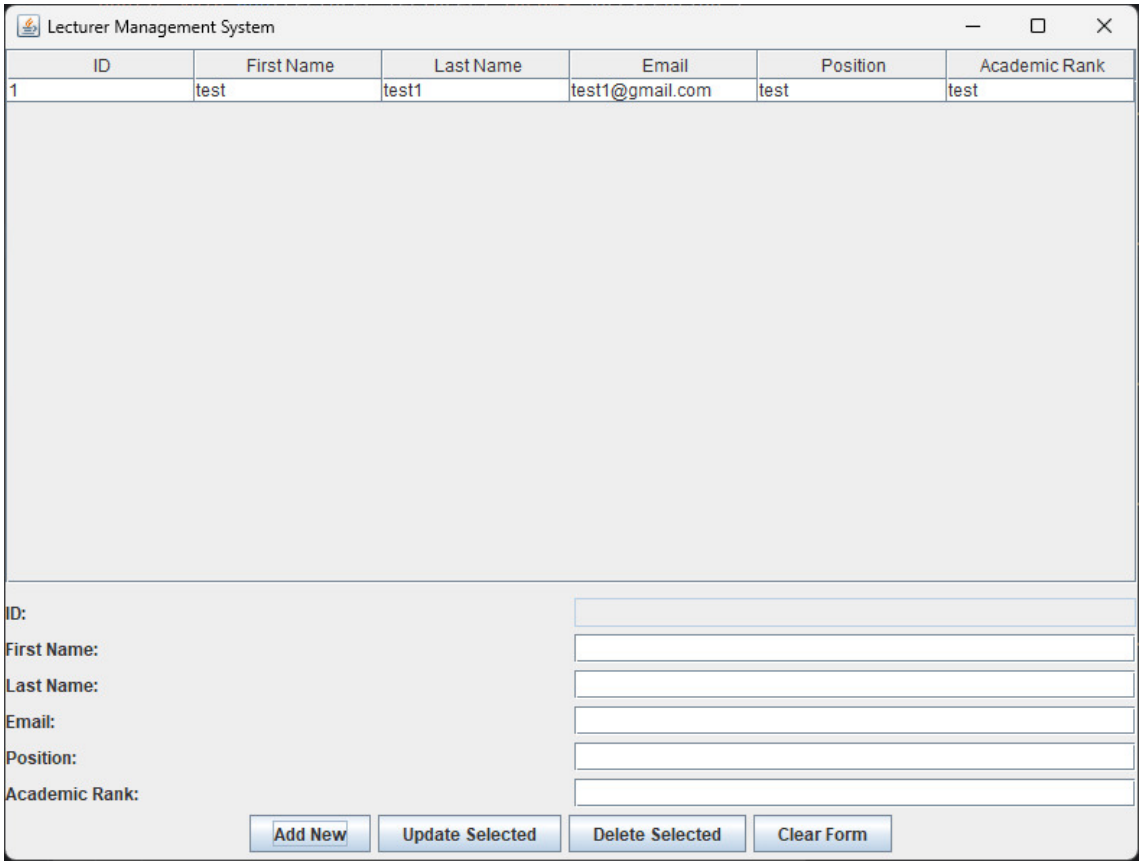
Rysunek 1.2. Główne okno aplikacji
Źródło: Zrzut ekranu z aplikacji (opracowanie własne).

Okno zarządzania studentami (rys. 1.3) zawiera tabelę z danymi, formularz do ich edycji i dodawania oraz przyciski do wykonywania operacji.



Rysunek 1.3. Interfejs do zarządzania studentami
Źródło: Zrzut ekranu z aplikacji (opracowanie własne).

Analogicznie, okno zarządzania wykładowcami (rys. 1.4) oferuje te same funkcjonalności dla danych wykładowców.



Rysunek 1.4. Interfejs do zarządzania wykładowcami
Źródło: Zrzut ekranu z aplikacji (opracowanie własne).

```
package com.manager.GUI;

import com.manager.dao.StudentDao;
import com.manager.university.Student;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.sql.SQLException;
import java.time.LocalDate;
import java.time.format.DateTimeParseException;
import java.util.List;

public class StudentManagementApp extends JFrame {

    private JTable studentTable;
    private DefaultTableModel tableModel;
    private JButton addButton, updateButton, deleteButton, clearButton;
    private JTextField idField, firstNameField, lastNameField, emailField, majorField,
    graduationEndField;
    private StudentDao studentDao;

    public StudentManagementApp() {
        studentDao = new StudentDao();
        setTitle("Student Management System");
```

```
25          setSize(800, 600);
26          setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
27          setLocationRelativeTo(null);
28          initUI();
29          refreshTableData();
30      }
31
32      private void initUI() {
33          tableModel = new DefaultTableModel(new Object[]{"ID", "First Name", "Last Name",
        "Email", "Major", "Graduation Date"}, 0);
34          studentTable = new JTable(tableModel);
35          studentTable.setAutoCreateRowSorter(true);
36          studentTable.getSelectionModel().addListSelectionListener(e -> {
37              if (!e.getValueIsAdjusting()) {
38                  fillFormFromSelectedRow();
39              }
40          });
41          JPanel formPanel = new JPanel(new GridLayout(6, 2, 5, 5));
42          idField = new JTextField();
43          idField.setEditable(false);
44          firstNameField = new JTextField();
45          lastNameField = new JTextField();
46          emailField = new JTextField();
47          majorField = new JTextField();
48          graduationEndField = new JTextField();
49          formPanel.add(new JLabel("ID:"));
50          formPanel.add(idField);
51          formPanel.add(new JLabel("First Name:"));
52          formPanel.add(firstNameField);
53          formPanel.add(new JLabel("Last Name:"));
54          formPanel.add(lastNameField);
55          formPanel.add(new JLabel("Email:"));
56          formPanel.add(emailField);
57          formPanel.add(new JLabel("Major:"));
58          formPanel.add(majorField);
59          formPanel.add(new JLabel("Graduation Date (YYYY-MM-DD):"));
60          formPanel.add(graduationEndField);
61          JPanel buttonPanel = new JPanel();
62          addButton = new JButton("Add New");
63          updateButton = new JButton("Update Selected");
64          deleteButton = new JButton("Delete Selected");
65          clearButton = new JButton("Clear Form");
66          buttonPanel.add(addButton);
67          buttonPanel.add(updateButton);
68          buttonPanel.add(deleteButton);
69          buttonPanel.add(clearButton);
70          JPanel controlPanel = new JPanel(new BorderLayout());
71          controlPanel.add(formPanel, BorderLayout.CENTER);
72          controlPanel.add(buttonPanel, BorderLayout.SOUTH);
73          setLayout(new BorderLayout(10, 10));
74          add(new JScrollPane(studentTable), BorderLayout.CENTER);
75          add(controlPanel, BorderLayout.SOUTH);
76          addButton.addActionListener(e -> addStudent());
77          updateButton.addActionListener(e -> updateStudent());
78          deleteButton.addActionListener(e -> deleteStudent());
79          clearButton.addActionListener(e -> clearForm());
80      }
81
82      private void refreshTableData() {
```

```
83          tableModel.setRowCount(0);
84          try {
85              List<Student> students = studentDao.getAll();
86              for (Student student : students) {
87                  tableModel.addRow(new Object[]{
88                          student.getId(),
89                          student.getFirstName(),
90                          student.getLastName(),
91                          student.getEmail(),
92                          student.getMajor(),
93                          student.getGraduationEnd()
94                  });
95              }
96          } catch (SQLException e) {
97              JOptionPane.showMessageDialog(this, "Error loading data from DB: " + e.
    getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
98          }
99      }
100
101     private void fillFormFromSelectedRow() {
102         int selectedRow = studentTable.getSelectedRow();
103         if (selectedRow != -1) {
104             idField.setText(tableModel.getValueAt(selectedRow, 0).toString());
105             firstNameField.setText(tableModel.getValueAt(selectedRow, 1).toString());
106             lastNameField.setText(tableModel.getValueAt(selectedRow, 2).toString());
107             emailField.setText(tableModel.getValueAt(selectedRow, 3).toString());
108             majorField.setText(tableModel.getValueAt(selectedRow, 4).toString());
109             graduationEndField.setText(tableModel.getValueAt(selectedRow, 5) != null ?
    tableModel.getValueAt(selectedRow, 5).toString() : "");
110         }
111     }
112
113     private void addStudent() {
114         try {
115             String firstName = firstNameField.getText();
116             String lastName = lastNameField.getText();
117             if (firstName.isEmpty() || lastName.isEmpty()) {
118                 JOptionPane.showMessageDialog(this, "First name and last name are
    required.", "Validation Error", JOptionPane.WARNING_MESSAGE);
119                 return;
120             }
121             Student student = new Student();
122             student.setFirstName(firstName);
123             student.setLastName(lastName);
124             student.setEmail(emailField.getText());
125             student.setMajor(majorField.getText());
126             student.setGraduationEnd(LocalDate.parse(graduationEndField.getText()));
127
128             studentDao.add(student);
129
130             refreshTableData();
131             clearForm();
132             JOptionPane.showMessageDialog(this, "Student added successfully!", "Success"
    , JOptionPane.INFORMATION_MESSAGE);
133
134         } catch (DateTimeParseException e) {
135             JOptionPane.showMessageDialog(this, "Invalid date format. Please use YYYY-MM
    -DD.", "Format Error", JOptionPane.ERROR_MESSAGE);
136         } catch (SQLException e) {
```

```
137             JOptionPane.showMessageDialog(this, "Database error during add operation: "
    + e.getMessage(), "DB Error", JOptionPane.ERROR_MESSAGE);
138         } catch (Exception e) {
139             JOptionPane.showMessageDialog(this, "An unknown error occurred: " + e.
    getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
140         }
141     }
142 
143     private void updateStudent() {
144         int selectedRow = studentTable.getSelectedRow();
145         if (selectedRow == -1) {
146             JOptionPane.showMessageDialog(this, "Please select a student to update.", "
    Error", JOptionPane.WARNING_MESSAGE);
147             return;
148         }
149         try {
150             Student student = new Student();
151             student.setId(Integer.parseInt(idField.getText()));
152             student.setFirstName(firstNameField.getText());
153             student.setLastName(lastNameField.getText());
154             student.setEmail(emailField.getText());
155             student.setMajor(majorField.getText());
156             student.setGraduationEnd(LocalDate.parse(graduationEndField.getText()));
157 
158             studentDao.update(student);
159             refreshTableData();
160             JOptionPane.showMessageDialog(this, "Student data updated successfully!", "
    Success", JOptionPane.INFORMATION_MESSAGE);
161         } catch (SQLException e) {
162             JOptionPane.showMessageDialog(this, "Database error during update operation:
     " + e.getMessage(), "DB Error", JOptionPane.ERROR_MESSAGE);
163         } catch (Exception e) {
164             JOptionPane.showMessageDialog(this, "An error occurred during update: " + e.
    getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
165         }
166     }
167 
168     private void deleteStudent() {
169         int selectedRow = studentTable.getSelectedRow();
170         if (selectedRow == -1) { return; }
171         int confirmation = JOptionPane.showConfirmDialog(this, "Are you sure?", "
    Confirmation", JOptionPane.YES_NO_OPTION);
172         if (confirmation == JOptionPane.YES_OPTION) {
173             try {
174                 int id = Integer.parseInt(idField.getText());
175                 studentDao.delete(id);
176                 refreshTableData();
177                 clearForm();
178                 JOptionPane.showMessageDialog(this, "Student deleted successfully!", "
    Success", JOptionPane.INFORMATION_MESSAGE);
179             } catch (SQLException e) {
180                 JOptionPane.showMessageDialog(this, "Database error during delete
    operation: " + e.getMessage(), "DB Error", JOptionPane.ERROR_MESSAGE);
181             }
182         }
183     }
184 
185     private void clearForm() {
186         idField.setText("");
```

```
187         firstNameField.setText("");
188         lastNameField.setText("");
189         emailField.setText("");
190         majorField.setText("");
191         graduationEndField.setText("");
192         studentTable.clearSelection();
193     }
194 
195     public static void main(String[] args) {
196         SwingUtilities.invokeLater(() -> {
197             com.manager.server.DatabaseConnector.initializeDatabase();
198             StudentManagementApp app = new StudentManagementApp();
199             app.setVisible(true);
200         });
201     }
202 }
```

Listing 1.3. Kod implementujący przycisk "Add New" w oknie zarządzania studentami.

Podsumowanie

W ramach niniejszej pracy zaprojektowano i zaimplementowano aplikację desktopową do zarządzania danymi studentów i wykładowców w systemie uniwersyteckim. Projekt został zrealizowany w technologii Java z wykorzystaniem biblioteki Swing do budowy interfejsu graficznego oraz bazy danych SQLite do przechowywania informacji. Główne cele pracy, takie jak stworzenie funkcjonalnego interfejsu CRUD (Create, Read, Update, Delete), implementacja warstwy dostępu do danych (DAO) oraz zaprojektowanie przejrzystej struktury klas, zostały w pełni osiągnięte. Aplikacja umożliwia dodawanie, edycję, usuwanie i przeglądanie danych w sposób intuicyjny dla użytkownika. Możliwe kierunki dalszego rozwoju projektu obejmują:

- Rozbudowę systemu o nowe moduły, np. zarządzanie kursami, ocenami czy planami zajęć.
- Implementację systemu uwierzytelniania użytkowników z różnymi poziomami uprawnień (np. administrator, pracownik dziekanatu).
- Migrację aplikacji do technologii webowej (np. z użyciem frameworka Spring Boot), co umożliwiłoby dostęp do systemu z dowolnego miejsca przez przeglądarkę internetową.
- Dodanie bardziej zaawansowanych funkcji, takich jak generowanie raportów i statystyk.

Spis rysunków

Listings

Bibliografia

[1] Joshua Bloch. *Effective Java*. Addison-Wesley Professional, 2018.

[2] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2002.

[3] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

[4] Cay S. Horstmann. *Core Java Volume I–Fundamentals*. Prentice Hall, 2019.

[5] Herbert Schildt. *Swing: A Beginner's Guide*. McGraw-Hill, 2007.

[6] Abraham Silberschatz, Henry F. Korth, S. Sudarshan. *Database System Concepts*. McGraw-Hill, 2011.

[7] SQLite. Sqlite home page. *https://www.sqlite.org/*, 2025. Accessed: 15-06-2025.

Załącznik nr 2 do Zarządzenia nr 228/2021 Rektora Uniwersytetu Rzeszowskiego z dnia 1 grudnia 2021 roku w sprawie ustalenia procedury antyplagiatowej w Uniwersytecie Rzeszowskim

OŚWIADCZENIE STUDENTA O SAMODZIELNOŚCI PRACY

.....Bohdan Kudrenko.....
Imię (imiona) i nazwisko studenta

Wydział Nauk Ścisłych i Technicznych

.....Informatyka.....
Nazwa kierunku

.....134935.....
Numer albumu

1. Oświadczam, że moja praca projektowa pt.: Przygotowanie dokumentacji do projektu w systemie LaTeX
   1) została przygotowana przeze mnie samodzielnie*,
   2) nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2021 r., poz. 1062) oraz dóbr osobistych chronionych prawem cywilnym,
   3) nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
   4) nie była podstawą otrzymania oceny z innego przedmiotu na uczelni wyższej ani mnie, ani innej osobie.
2. Jednocześnie wyrażam zgodę/~~nie wyrażam zgody~~** na udostępnienie mojej pracy projektowej do celów naukowo–badawczych z poszanowaniem przepisów ustawy o prawie autorskim i prawach pokrewnych.

Rzeszów 14.06.2025
(miejscowość, data)

Bohdan Kudrenko
(czytelny podpis studenta)

* Uwzględniając merytoryczny wkład prowadzącego przedmiot
** – niepotrzebne skreślić