



Bohdan Kudrenko

134935

Informatyka

**Przygotowanie dokumentacji do projektu w systemie LATEX**  
**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.....	3
1.1. Opis założeń projektu .....	3
1.2. Opis struktury projektu .....	3
1.2.1. Wykorzystane technologie .....	3
1.2.2. Struktura bazy danych .....	3
1.2.3. Hierarchia klas .....	4
1.3. Grafik realizacji projektu .....	5
1.4. Prezentacja interfejsu użytkownika .....	5
Podsumowanie .....	13
Spis rysunków .....	14
Spis listingów .....	15
Oświadczenie studenta o samodzielności pracy.....	16

# 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).
- Interfejs graficzny użytkownika (GUI): Java Swing—standardowa biblioteka do tworzenia aplikacji okienkowych w Javie.
- Baza danych: SQLite —lekki, plikowy silnik bazy danych, który nie wymaga oddzielnego serwera.
- Sterownik bazy danych: JDBC (Java Database Connectivity) do komunikacji między aplikacją a bazą danych.

### 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.

```
1 CREATE TABLE IF NOT EXISTS students (  
2     id INTEGER PRIMARY KEY AUTOINCREMENT,  
3     first_name TEXT NOT NULL,  
4     last_name TEXT NOT NULL,  
5     email TEXT,  
6     major TEXT,  
7     graduation_end DATE  
8 );
```

#### Listing 1.1. Schemat SQL tabeli 'students'

Źródło: Opracowanie własne na podstawie pliku DatabaseConnector.java.

```

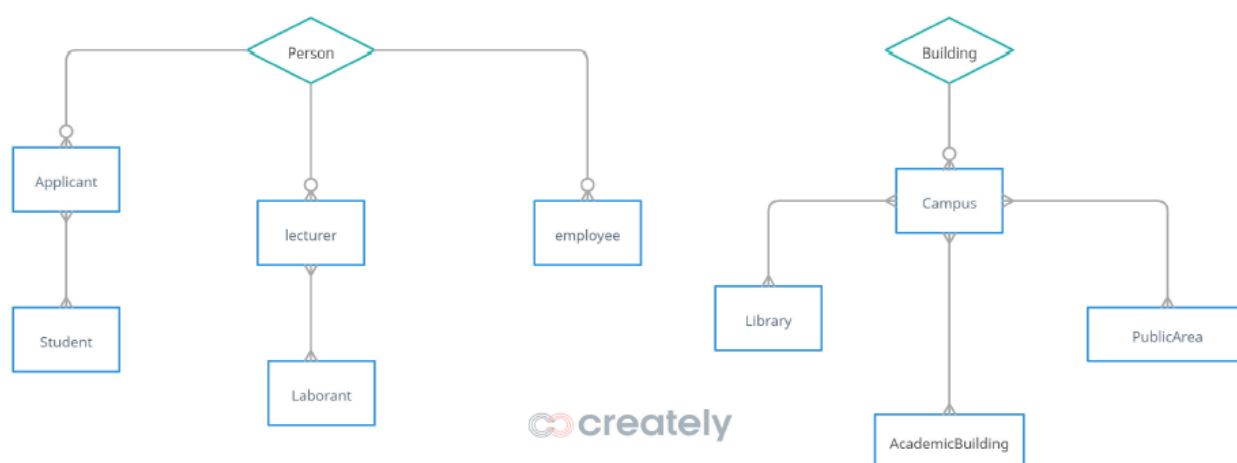
1 CREATE TABLE IF NOT EXISTS lecturers (
2     id INTEGER PRIMARY KEY AUTOINCREMENT,
3     first_name TEXT NOT NULL,
4     last_name TEXT NOT NULL,
5     email TEXT,
6     position TEXT,
7     academic_rank TEXT
8 );

```

### 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. 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 wzorec projektowy DAO. 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. Grafika realizacji projektu

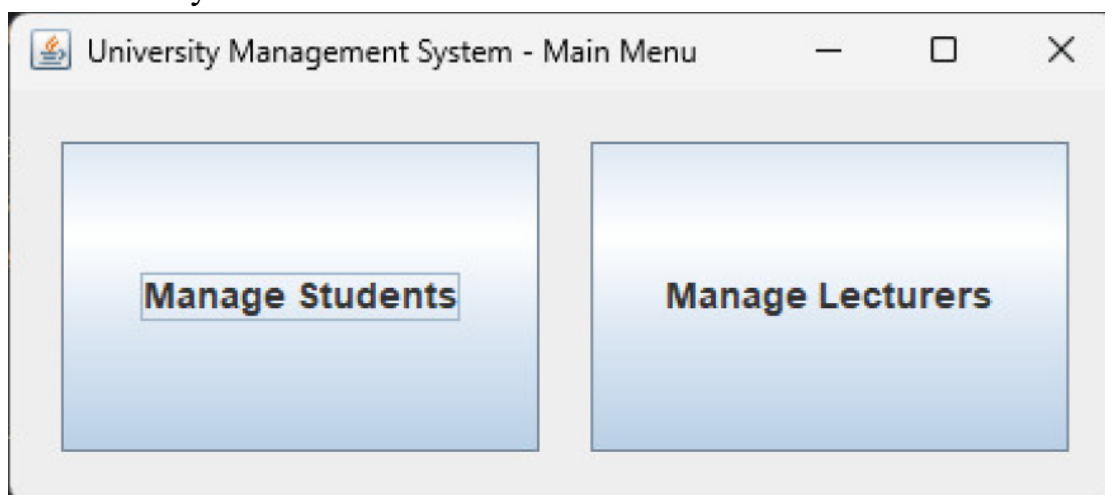
Realizacja projektu została zaplanowana i wykonana w określonych ramach czasowych.

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 ‘Local-Date’.

### 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.

The screenshot displays a window titled "Student Management System". It contains a table with the following columns: ID, First Name, Last Name, Email, Major, and Graduation Date. Below the table is a form with input fields for each column, labeled "ID:", "First Name:", "Last Name:", "Email:", "Major:", and "Graduation Date (YYYY-MM-DD):". At the bottom of the form are four buttons: "Add New", "Update Selected", "Delete Selected", and "Clear Form".

ID	First Name	Last Name	Email	Major	Graduation Date
31	Petro	Petrenko	student.petrenko.31...	Philology	2028-12-02
41	Oleksandr	Bondarenko	student.bondarenko....	Law	2028-11-13
5	Kateryna	Bondarenko	student.bondarenko....	Computer Science	2028-10-12
26	Mariia	Romanenko	student.romanenko....	Economics	2028-08-05
40	Sofiia	Romanenko	student.romanenko....	Law	2028-06-02
21	Sofiia	Bondarenko	student.bondarenko....	Engineering	2028-02-14
12	Petro	Sydorenko	student.sydorenko.1...	Computer Science	2028-01-27
34	Petro	Ivanenko	student.ivanenko.34...	Philology	2028-01-16
7	Sofiia	Bondarenko	student.bondarenko....	Medicine	2027-11-25
48	Kateryna	Petrenko	student.petrenko.48...	Engineering	2027-10-24
3	Petro	Ivanenko	student.ivanenko.3@...	Economics	2027-10-15
28	Oleksandr	Bondarenko	student.bondarenko....	Economics	2027-09-24
22	Mariia	Ivanenko	student.ivanenko.22...	Medicine	2027-09-17
50	Oleksandr	Romanenko	student.romanenko....	Economics	2027-08-26
29	Sofiia	Bondarenko	student.bondarenko....	Medicine	2027-08-20
2	Petro	Kovalchuk	student.kovalchuk.2...	Engineering	2027-08-10
17	Petro	Bondarenko	student.bondarenko....	Computer Science	2027-07-08
44	Ivan	Bondarenko	student.bondarenko....	Economics	2027-05-28
37	Sofiia	Ivanenko	student.ivanenko.37...	Medicine	2027-04-18
16	Oleksandr	Petrenko	student.petrenko.16...	Law	2027-04-11
43	Oleksandr	Bondarenko	student.bondarenko....	Computer Science	2027-02-09
4	Mariia	Ivanenko	student.ivanenko.4@...	Medicine	2026-12-08

Form fields:

ID:

First Name:

Last Name:

Email:

Major:

Graduation Date (YYYY-MM-DD):

Buttons: Add New, Update Selected, Delete Selected, Clear Form

**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.

ID	First Name	Last Name	Email	Position	Academic Rank
1	test	test1	test1@gmail.com	test	test

ID:

First Name:

Last Name:

Email:

Position:

Academic Rank:

**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");
    setSize(800, 600);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null);
    initUI();
    refreshTableData();
}

private void initUI() {
    tableModel = new DefaultTableModel(new Object[]{"ID", "First Name", "Last
Name", "Email", "Major", "Graduation Date"}, 0);
    studentTable = new JTable(tableModel);
    studentTable.setAutoCreateRowSorter(true);
    studentTable.getSelectionModel().addListSelectionListener(e -> {
        if (!e.getValueIsAdjusting()) {
            fillFormFromSelectedRow();
        }
    });
    JPanel formPanel = new JPanel(new GridLayout(6, 2, 5, 5));
    idField = new JTextField();
    idField.setEditable(false);
    firstNameField = new JTextField();
    lastNameField = new JTextField();
    emailField = new JTextField();
    majorField = new JTextField();
    graduationEndField = new JTextField();
    formPanel.add(new JLabel("ID:"));
    formPanel.add(idField);
    formPanel.add(new JLabel("First Name:"));
    formPanel.add(firstNameField);
    formPanel.add(new JLabel("Last Name:"));
    formPanel.add(lastNameField);
    formPanel.add(new JLabel("Email:"));
    formPanel.add(emailField);
    formPanel.add(new JLabel("Major:"));
    formPanel.add(majorField);
    formPanel.add(new JLabel("Graduation Date (YYYY-MM-DD):"));
    formPanel.add(graduationEndField);
    JPanel buttonPanel = new JPanel();
    addButton = new JButton("Add New");
    updateButton = new JButton("Update Selected");
    deleteButton = new JButton("Delete Selected");
    clearButton = new JButton("Clear Form");
}
```



```
        buttonPanel.add(addButton);
        buttonPanel.add(updateButton);
        buttonPanel.add(deleteButton);
        buttonPanel.add(clearButton);
        JPanel controlPanel = new JPanel(new BorderLayout());
        controlPanel.add(formPanel, BorderLayout.CENTER);
        controlPanel.add(buttonPanel, BorderLayout.SOUTH);
        setLayout(new BorderLayout(10, 10));
        add(new JScrollPane(studentTable), BorderLayout.CENTER);
        add(controlPanel, BorderLayout.SOUTH);
        addButton.addActionListener(e -> addStudent());
        updateButton.addActionListener(e -> updateStudent());
        deleteButton.addActionListener(e -> deleteStudent());
        clearButton.addActionListener(e -> clearForm());
    }

    private void refreshTableData() {
        tableModel.setRowCount(0);
        try {
            List<Student> students = studentDao.getAll();
            for (Student student : students) {
                tableModel.addRow(new Object[]{
                    student.getId(),
                    student.getFirstName(),
                    student.getLastName(),
                    student.getEmail(),
                    student.getMajor(),
                    student.getGraduationEnd()
                });
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Error loading data from DB: " +
                e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }

    private void fillFormFromSelectedRow() {
        int selectedRow = studentTable.getSelectedRow();
        if (selectedRow != -1) {
            idField.setText(tableModel.getValueAt(selectedRow, 0).toString());
            firstNameField.setText(tableModel.getValueAt(selectedRow, 1).toString());
            lastNameField.setText(tableModel.getValueAt(selectedRow, 2).toString());
            emailField.setText(tableModel.getValueAt(selectedRow, 3).toString());
            majorField.setText(tableModel.getValueAt(selectedRow, 4).toString());
            graduationEndField.setText(tableModel.getValueAt(selectedRow, 5) != null ?
                tableModel.getValueAt(selectedRow, 5).toString() : "");
        }
    }
}
```

```
private void addStudent() {
    try {
        String firstName = firstNameField.getText();
        String lastName = lastNameField.getText();
        if (firstName.isEmpty() || lastName.isEmpty()) {
            JOptionPane.showMessageDialog(this, "First name and last name are
required.", "Validation Error", JOptionPane.WARNING_MESSAGE);
            return;
        }
        Student student = new Student();
        student.setFirstName(firstName);
        student.setLastName(lastName);
        student.setEmail(emailField.getText());
        student.setMajor(majorField.getText());
        student.setGraduationEnd(LocalDate.parse(graduationEndField.getText()));

        studentDao.add(student);

        refreshTableData();
        clearForm();
        JOptionPane.showMessageDialog(this, "Student added successfully!",
"Success", JOptionPane.INFORMATION_MESSAGE);

    } catch (DateTimeParseException e) {
        JOptionPane.showMessageDialog(this, "Invalid date format. Please use YYYY-
MM-DD.", "Format Error", JOptionPane.ERROR_MESSAGE);
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database error during add operation:
" + e.getMessage(), "DB Error", JOptionPane.ERROR_MESSAGE);
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "An unknown error occurred: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
}

private void updateStudent() {
    int selectedRow = studentTable.getSelectedRow();
    if (selectedRow == -1) {
        JOptionPane.showMessageDialog(this, "Please select a student to update.",
"Error", JOptionPane.WARNING_MESSAGE);
        return;
    }
    try {
        Student student = new Student();
        student.setId(Integer.parseInt(idField.getText()));
        student.setFirstName(firstNameField.getText());
        student.setLastName(lastNameField.getText());
        student.setEmail(emailField.getText());
        student.setMajor(majorField.getText());
    }
```

```
        student.setGraduationEnd(LocalDate.parse(graduationEndField.getText()));

        studentDao.update(student);
        refreshTableData();
        JOptionPane.showMessageDialog(this, "Student data updated successfully!",
"Success", JOptionPane.INFORMATION_MESSAGE);
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database error during update
operation: " + e.getMessage(), "DB Error", JOptionPane.ERROR_MESSAGE);
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "An error occurred during update: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
}

private void deleteStudent() {
    int selectedRow = studentTable.getSelectedRow();
    if (selectedRow == -1) { return; }
    int confirmation = JOptionPane.showConfirmDialog(this, "Are you sure?",
"Confirmation", JOptionPane.YES_NO_OPTION);
    if (confirmation == JOptionPane.YES_OPTION) {
        try {
            int id = Integer.parseInt(idField.getText());
            studentDao.delete(id);
            refreshTableData();
            clearForm();
            JOptionPane.showMessageDialog(this, "Student deleted successfully!",
"Success", JOptionPane.INFORMATION_MESSAGE);
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Database error during delete
operation: " + e.getMessage(), "DB Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}

private void clearForm() {
    idField.setText("");
    firstNameField.setText("");
    lastNameField.setText("");
    emailField.setText("");
    majorField.setText("");
    graduationEndField.setText("");
    studentTable.clearSelection();
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        com.manager.server.DatabaseConnector.initializeDatabase();
        StudentManagementApp app = new StudentManagementApp();
    });
}
```

```
        app.setVisible(true);  
    });  
}  
}
```

**Listing 1.3.** Fragment kodu implementującego 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**

1.1 Diagram ERD bazy danych . . . . .	4
1.2 Główne okno aplikacji . . . . .	5
1.3 Interfejs do zarządzania studentami . . . . .	6
1.4 Interfejs do zarządzania wykładowcami . . . . .	7

# Listings

1.1 Schemat SQL tabeli 'students' .....	3
1.2 Schemat SQL tabeli 'lecturers' .....	4
1.3 Fragment kodu implementującego przycisk "Add New" w oknie zarządzania studentami. ....	12

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 L<sup>A</sup>T<sub>E</sub>X<sup>X</sup>
  - 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