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).
- 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.

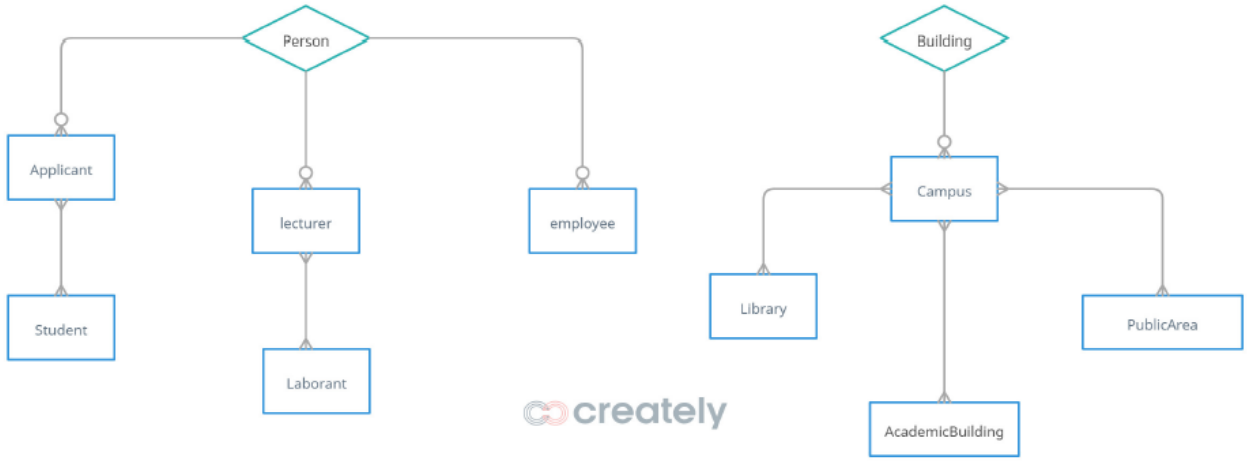
```
1 CREATE TABLE IF NOT EXISTS students (
2         id INTEGER PRIMARY KEY AUTOINCREMENT,
3         first_name TEXT NOT NULL,
4         last_name TEXT NOT NULL,
5         email TEXT,
6         major TEXT,
7         graduation_end DATE
8 );
```

Listing 1.1. Schemat SQL tabeli ‘students‘
Źródło: Opracowanie własne na podstawie pliku DatabaseConnector.java.

```
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. 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. Interfejs ‘Dao<T>‘ definiuje podstawowe operacje bazodanowe, a jego implementacje ‘StudentDao‘ I ‘LecturerDao‘ zawierają logikę odpowiedzialną za komunikację z konkretnymi tabelami w bazie danych.

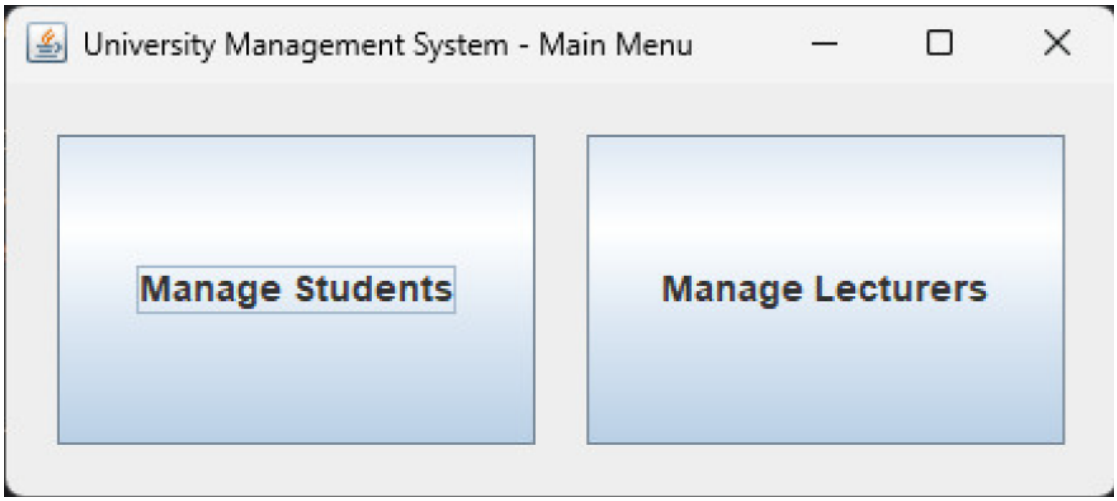
1.3. Grafik realizacji projektu i prezentacja interfejsu użytkownika

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‘.

Aplikacja posiada graficzny interfejs użytkownika (GUI) stworzony przy użyciu biblioteki Swing. Interfejs jest prosty i intuicyjny, co ułatwia zarządzanie danymi.

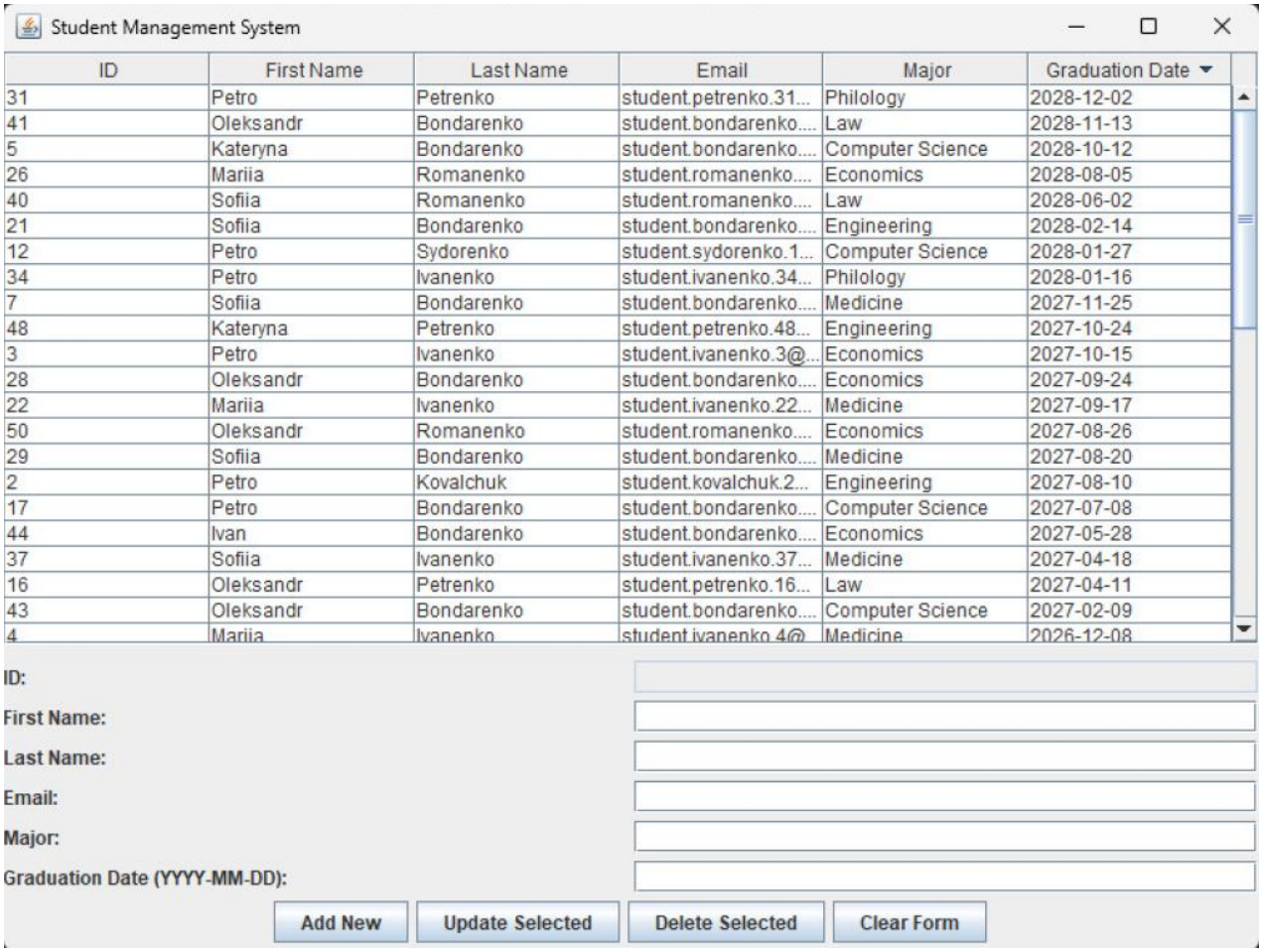
Główne okno aplikacji (rys.1.2) pozwala na wybór modułu do zarządzania — studentami lub wykładowcami.



Rysunek 1.2. Główne okno aplikacji

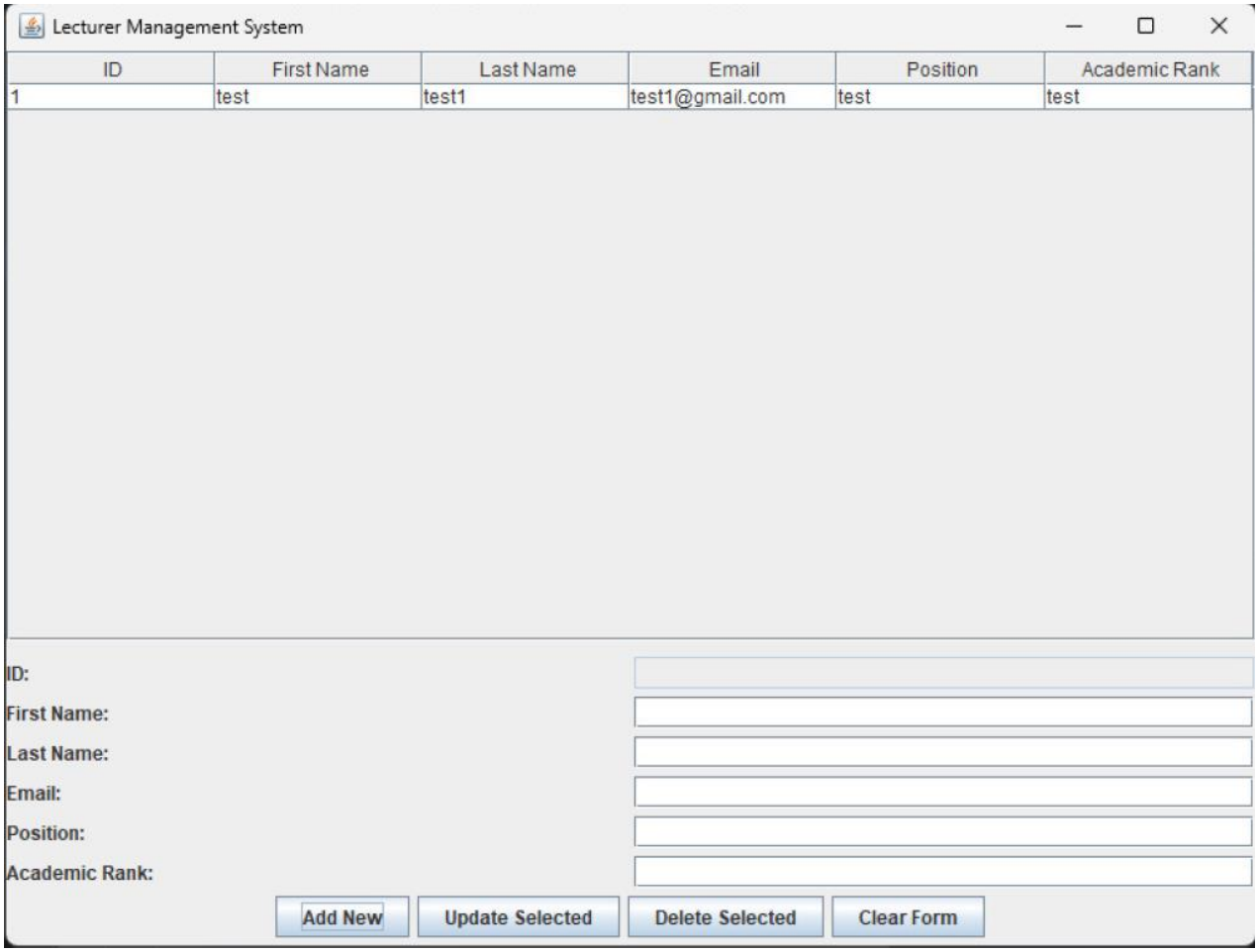
Źródło: Zrzut ekranu z aplikacji (opracowanie własne).

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");
        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. admi- nistrator, 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

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