

UNIwersYTET RZESZOWSKI
WYDZIAŁ NAUK ŚCISŁYCH I TECHNICZNYCH
INSTYTUT INFORMATYKI



Mykhailo Kleban
134922

Informatyka

System rezerwacji sal/podział godzin

Praca projektowa

Praca wykonana pod kierunkiem
dr inż. Ewa Żesławska

Rzeszów 2025

Spis treści

1. Streszczenie	2
1.1. Streszczenie w języku polskim	2
1.2. Abstract in English	2
2. Opis projektu	3
2.1. Cel i przeznaczenie	3
2.2. Główne funkcjonalności	3
2.3. Typy zajęć	3
2.4. Zastosowane technologie	3
3. Implementacja systemu	4
3.1. Struktura aplikacji	4
4. Harmonogram realizacji projektu	5
5. Opis interfejsu użytkownika	6
5.1. Baza danych	12
5.2. Diagram Baza Danych	12
5.3. Interfejs użytkownika	13
5.4. Walidacja i błędy	13
5.5. Szczegółowy opis komponentów GUI	13
5.5.1. Formularz Dodawania Zajęć (DodajZajeciaPanel)	13
5.5.2. Formularz Edytowania Zajęć (EdytujZajeciaPanel)	13
5.5.3. Formularz Logowania (Login)	14
5.5.4. Panel Sekretariatu (SekretariatPanel)	14
5.6. Diagram komponentów GUI i zależności	15
6. Testowanie systemu	16
6.1. Testy logowania	16
6.2. Testy dodawania zajęć	16
6.3. Testy filtrowania danych	17
6.4. Testy edycji i usuwania zajęć	17
6.5. Testy walidacji danych	18
7. Podsumowanie i wnioski	20
7.1. Osiągnięte rezultaty	20
7.2. Wnioski	20
7.3. Linki do repozytoriów GitHub	21
Bibliografia	22
Spis rysunków	23
Spis listingów	24

1. Streszczenie

1.1. Streszczenie w języku polskim

Celem niniejszego projektu było stworzenie aplikacji wspomagającej proces **rezerwacji sal** oraz zarządzania **harmonogramem zajęć** na uczelni. Projekt został zrealizowany w języku **Java** z wykorzystaniem **graficznego interfejsu użytkownika** (Swing), połączenia z relacyjną bazą danych **PostgreSQL** za pomocą **JDBC** oraz wzorca **DAO** do komunikacji z bazą.

Aplikacja umożliwia wykonywanie operacji **dodawania, edytowania, filtrowania i usuwania zajęć** dla różnych typów (**wykład, laboratorium, projekt**). Uwzględnia różnice w obsłudze **grup** w zależności od typu zajęć oraz zawiera **walidację danych wejściowych** i odpowiednie **komunikaty błędów**.

Projekt pozwolił na rozwinięcie umiejętności z zakresu **programowania obiektowego**, pracy z **bazami danych** oraz tworzenia aplikacji desktopowych z wykorzystaniem bibliotek **GUI**.

1.2. Abstract in English

The aim of this project was to create an application that supports the process of **room reservation** and **class schedule management** at a university. The project was implemented in **Java** using a **graphical user interface** (Swing), connection to a relational database **PostgreSQL** via **JDBC**, and the **DAO design pattern** for data access.

The application allows **adding, editing, filtering, and deleting classes** of various types (**lecture, laboratory, project**). It handles **group selection logic** depending on the type of class and includes **input validation** with appropriate **error messages**.

The project enabled the development of skills in **object-oriented programming**, working with **databases**, and creating desktop applications using **GUI libraries**.

2. Opis projektu

2.1. Cel i przeznaczenie

Projekt **System rezerwacji sal / podział godzin** został zaprojektowany z myślą o ułatwieniu zarządzania harmonogramem zajęć akademickich. Głównym celem systemu jest wsparcie administracji uczelni w planowaniu i koordynowaniu zajęć dydaktycznych w sposób zautomatyzowany i intuicyjny.

2.2. Główne funkcjonalności

System umożliwia użytkownikowi wykonywanie następujących operacji:

- dodawanie zajęć wraz z informacjami: dzień tygodnia, godzina, typ zajęć, kierunek, przedmiot, prowadzący, sala, grupa;
- edytowanie oraz usuwanie wcześniej dodanych zajęć;
- filtrowanie zajęć według sali, grupy i typu zajęć;
- walidację danych przy wprowadzaniu (np. sprawdzanie konfliktów sal i grup);
- obsługę wyjątków i prezentację komunikatów błędów.

2.3. Typy zajęć

System rozróżnia trzy typy zajęć, które różnią się liczbą przypisanych grup:

- **Wykład (Wykład)** – przeznaczony dla wszystkich grup;
- **Projekt (ćwiczenia)** – przeznaczony dla dwóch konkretnych grup (np. A i B);
- **Laboratorium** – przeznaczone dla jednej grupy.

2.4. Zastosowane technologie

Do realizacji projektu wykorzystano następujące technologie:

- język **Java**, którego zasady zostały zaczerpnięte z literatury [2];
- biblioteka **Swing** do budowy graficznego interfejsu użytkownika;
- baza danych **PostgreSQL**;
- interfejs **JDBC** do komunikacji z bazą danych;
- wzorzec projektowy **DAO**, zgodnie z podejściem przedstawionym w [1].

3. Implementacja systemu

System został zaimplementowany w języku **Java**, z wykorzystaniem biblioteki **Swing** do budowy graficznego interfejsu użytkownika oraz technologii **JDBC** do komunikacji z bazą danych **PostgreSQL**.

3.1. Struktura aplikacji

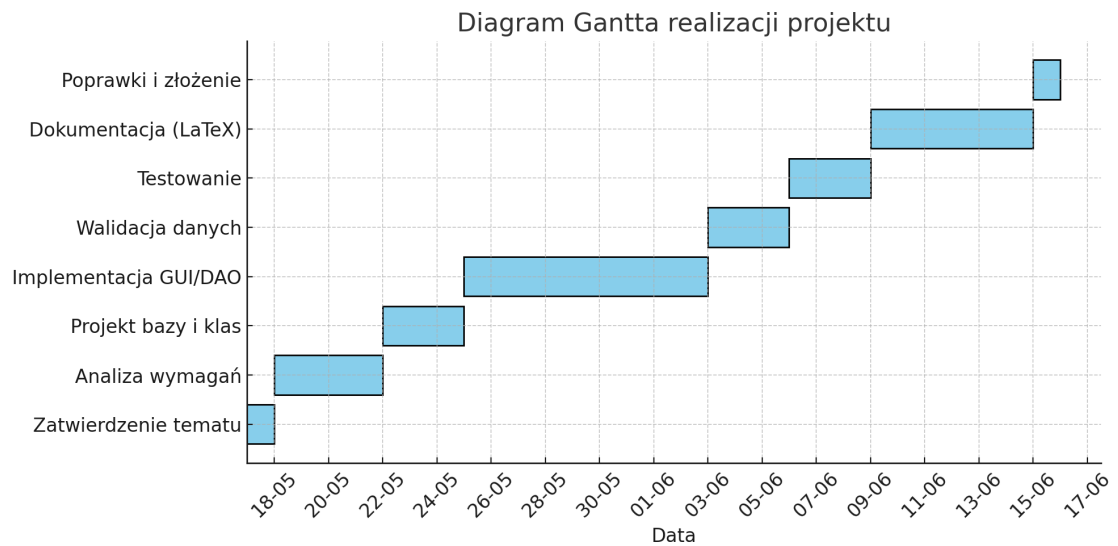
Struktura projektu została logicznie podzielona na pakiety zgodnie z zasadami dobrej organizacji kodu. W folderze `src` znajdują się wszystkie elementy źródłowe aplikacji, zorganizowane w następujący sposób:

- `dao` – klasy odpowiedzialne za dostęp do danych:
 - `LoginDAO` – obsługa uwierzytelniania użytkownika;
 - `ZajeciaDAO` – operacje CRUD na tabeli zajęć.
- `database` – logika połączenia z bazą danych:
 - `DatabaseConnection` – klasa łącząca aplikację z PostgreSQL.
- `DodajZajeciaPanel` – komponent GUI odpowiedzialny za dodawanie nowych zajęć:
 - `DodajZajeciaPanel.java/.form` – panel formularza oraz jego widok.
- `EdytujZajeciaPanel` – komponent GUI służący do edycji zajęć:
 - `EdytujZajeciaPanel.java/.form` – logika i widok edycji.
- `Login` – komponent odpowiedzialny za ekran logowania:
 - `Login.java/.form` – widok i obsługa logowania.
- `model` – klasy reprezentujące dane biznesowe:
 - `Zajecia` – klasa bazowa reprezentująca ogólne zajęcia;
 - `Wyklad`, `Projekt`, `Laboratorium` – klasy dziedziczące;
 - `PlanZajec` – klasa pomocnicza reprezentująca pojedynczy wpis w planie.
- `SekretariatPanel` – główny interfejs do zarządzania zajęciami:
 - `SekretariatPanel.java/.form` – widok panelu oraz jego logika.
- `resource` – folder przechowujący zasoby zewnętrzne (np. ikony lub grafiki).
- `Main.java` – klasa uruchamiająca aplikację.

Takie rozdzielenie pozwala na lepszą czytelność kodu, łatwiejsze zarządzanie komponentami oraz zgodność z zasadami programowania obiektowego.

4. Harmonogram realizacji projektu

W poniższej tabeli oraz na wykresie Gantta przedstawiono plan realizacji projektu wraz z zakładanym czasem trwania każdego etapu.



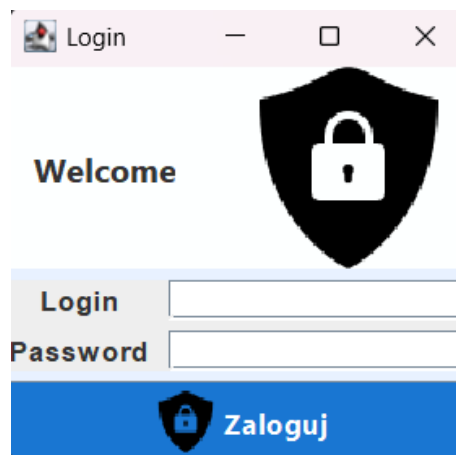
Rys. 1. Harmonogram realizacji projektu w formie wykresu Gantta

5. Opis interfejsu użytkownika

Panel logowania

Jak pokazano na Rys. 2, panel logowania umożliwia autoryzację użytkownika przed uzyskaniem dostępu do głównego interfejsu systemu. Formularz składa się z dwóch pól: `Login` oraz `Password`, oraz przycisku **Zaloguj**.

W przypadku poprawnych danych, użytkownik zostaje przekierowany do głównego panelu systemu. Jeśli jednak dane logowania są nieprawidłowe, system wyświetla stosowny komunikat błędu za pomocą okna typu `JOptionPane`. Panel zapewnia podstawową walidację pustych pól i poprawności danych logowania.



Rys. 2. Panel logowania do systemu

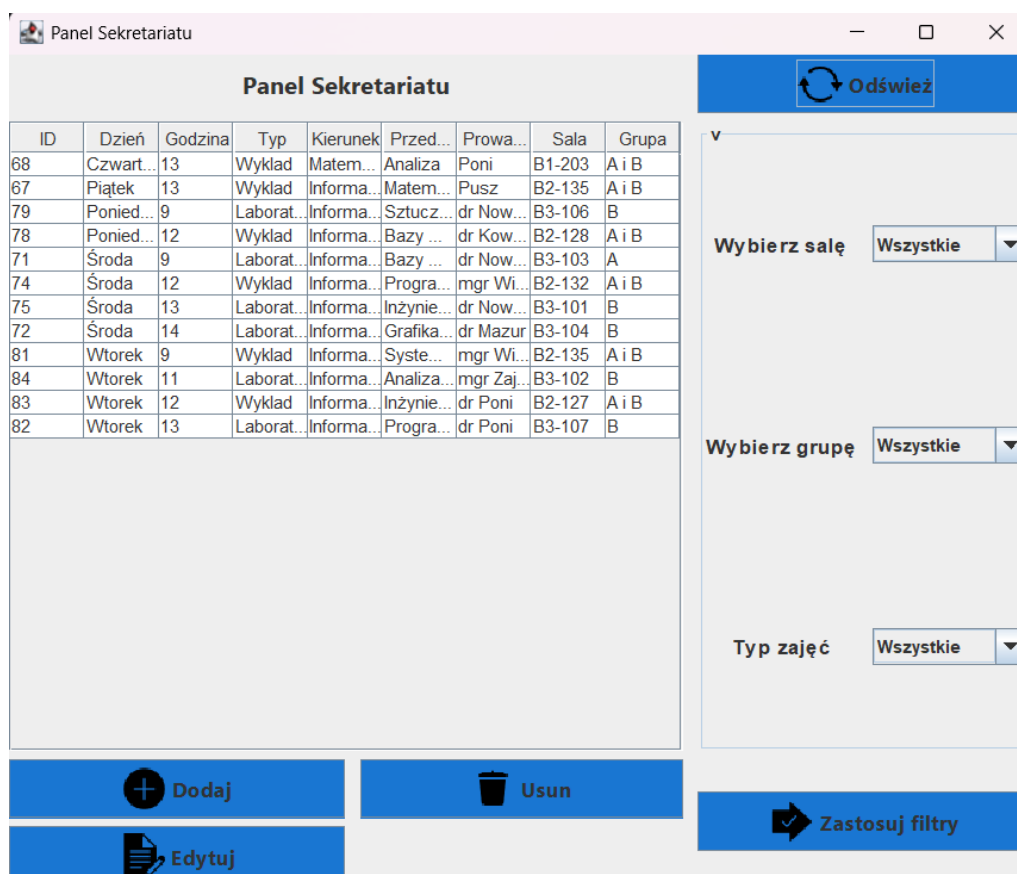
Główny panel sekretariatu

Na Rys. 3 widoczny jest główny panel sekretariatu. Panel ten jest centralnym miejscem zarządzania danymi w systemie. Wyświetla on tabelę ze wszystkimi zajęciami zapisanymi w bazie danych.

Użytkownik ma do dyspozycji przyciski:

- **Dodaj** – otwiera formularz umożliwiający dodanie nowych zajęć,
- **Edytuj** – pozwala zmodyfikować zaznaczony rekord,
- **Usuń** – usuwa wybrane zajęcia z bazy,
- **Zastosuj filtry** – uruchamia filtrację danych,
- **Odśwież** – ładuje wszystkie dane ponownie.

Filtrowanie zajęć możliwe jest przez rozwijane listy, w których użytkownik może wybrać: salę, grupę lub typ zajęć. Dane prezentowane są w tabeli `JTable`, umieszczonej w kontenerze `JScrollPane`.

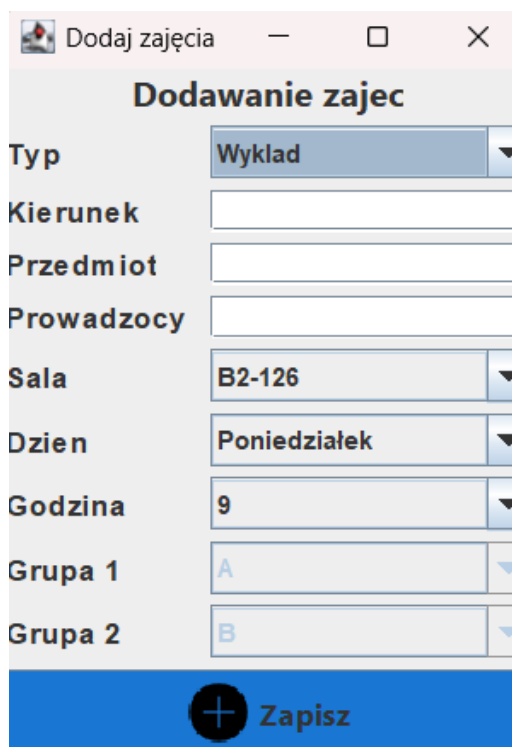


Rys. 3. Główny panel sekretariatu z listą zajęć i filtrowaniem

Formularz dodawania zajęć typu Wykład

Rysunek 4 przedstawia panel umożliwiający dodawanie zajęć typu **Wykład**. Formularz ten zawiera pola tekstowe do uzupełnienia takich informacji jak kierunek studiów, nazwa przedmiotu, prowadzący, numer sali, dzień tygodnia oraz godzina.

Pola Grupa 1 i Grupa 2 są zablokowane, ponieważ wykład dotyczy wszystkich studentów danego kierunku niezależnie od przypisania do grupy.

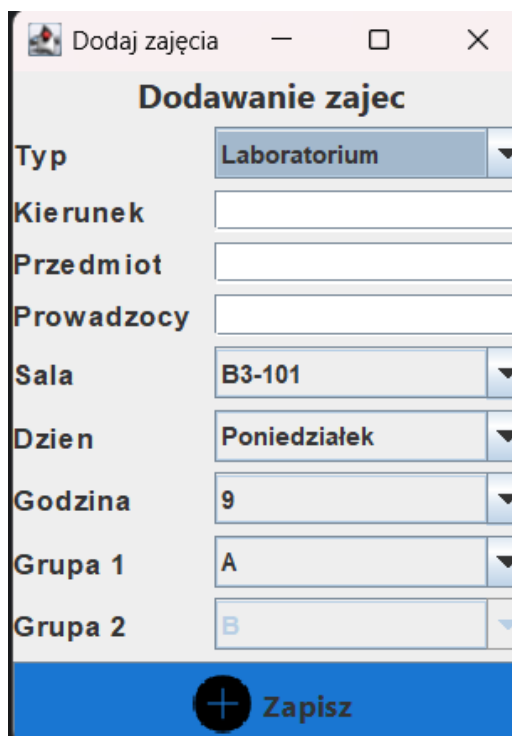


Rys. 4. Formularz dodawania zajęć typu Wykład

Formularz dodawania zajęć typu Laboratorium

Na Rys. 5 zaprezentowano formularz służący do dodawania zajęć typu **Laboratorium**. W tym przypadku możliwe jest przypisanie jednej grupy do zajęć – użytkownik wybiera ją w polu Grupa 1, natomiast Grupa 2 pozostaje nieaktywne.

Formularz ten dostosowuje swoją funkcjonalność dynamicznie w zależności od wybranego typu zajęć. Przycisk **Zapisz** umożliwia dodanie danych po przejściu poprawnej walidacji.



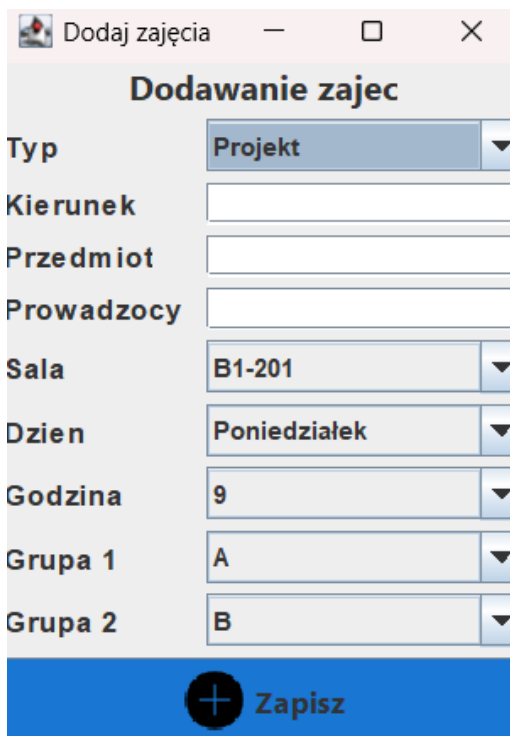
The image shows a software window titled "Dodaj zajęcia" (Add classes). Inside, the title "Dodawanie zajęć" (Adding classes) is centered. The form contains several fields: "Typ" (Type) is a dropdown menu set to "Laboratorium"; "Kierunek" (Direction) is an empty text field; "Przedmiot" (Subject) is an empty text field; "Prowadzocy" (Instructors) is an empty text field; "Sala" (Room) is a dropdown menu set to "B3-101"; "Dzień" (Day) is a dropdown menu set to "Poniedziałek" (Monday); "Godzina" (Hour) is a dropdown menu set to "9"; "Grupa 1" (Group 1) is a dropdown menu set to "A"; and "Grupa 2" (Group 2) is a dropdown menu set to "B". At the bottom of the form is a blue button with a white plus icon and the text "Zapisz" (Save).

Rys. 5. Formularz dodawania zajęć typu Laboratorium

Formularz dodawania zajęć typu Projekt

Jak przedstawiono na Rys. 6, formularz ten umożliwia przypisanie dwóch grup do wspólnych zajęć typu **Projekt**. Dane te są zapisywane zarówno w tabeli *zajecia*, jak i w powiązanej tabeli *projekty*.

Użytkownik musi podać wszystkie wymagane informacje: kierunek, przedmiot, prowadzący, sala, dzień i godzina zajęć, oraz przypisać obie grupy projektowe. System weryfikuje poprawność danych oraz sprawdza potencjalne konflikty terminów przed zapisaniem rekordu.



The image shows a software window titled "Dodaj zajęcia" (Add classes). Inside, the title "Dodawanie zajec" is displayed. The form contains the following fields and values:

- Typ:** Projekt (dropdown menu)
- Kierunek:** (empty text field)
- Przedmiot:** (empty text field)
- Prowadzocy:** (empty text field)
- Sala:** B1-201 (dropdown menu)
- Dzien:** Poniedziałek (dropdown menu)
- Godzina:** 9 (dropdown menu)
- Grupa 1:** A (dropdown menu)
- Grupa 2:** B (dropdown menu)

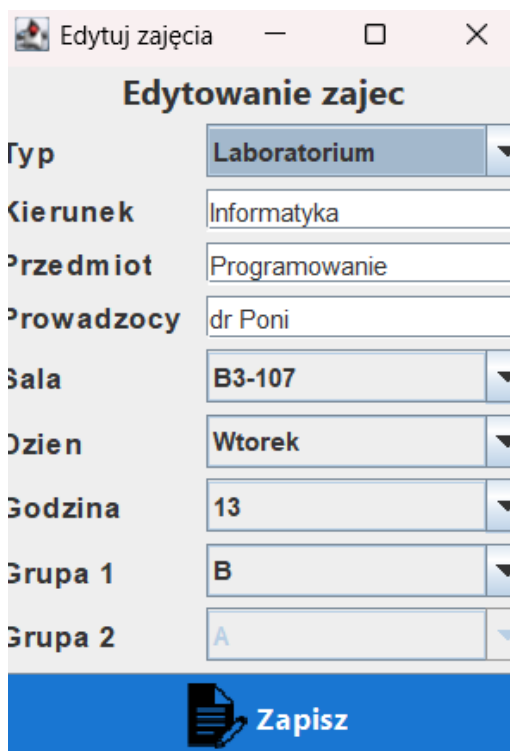
At the bottom of the form is a large blue button with a white plus icon and the text "Zapisz" (Save).

Rys. 6. Formularz dodawania zajęć typu Projekt


Formularz edycji zajęć

Rys. 7 prezentuje panel edycji zajęć. Użytkownik może zaznaczyć rekord z tabeli głównego panelu i przejść do jego modyfikacji. Wszystkie dane z wybranego wiersza są automatycznie załadowane do formularza.

Po zakończeniu edycji wystarczy kliknąć przycisk **Zapisz**, aby zapisać zmiany w bazie danych. Formularz umożliwia edycję dowolnego typu zajęć — również typu Projekt i Laboratorium, przy czym zachowane są te same zasady walidacji co podczas dodawania.



Typ	Laboratorium
Kierunek	Informatyka
Przedmiot	Programowanie
Prowadzocy	dr Poni
Sala	B3-107
Dzień	Wtorek
Godzina	13
Grupa 1	B
Grupa 2	A

 **Zapisz**

Rys. 7. Formularz edycji istniejących zajęć

5.1. Baza danych

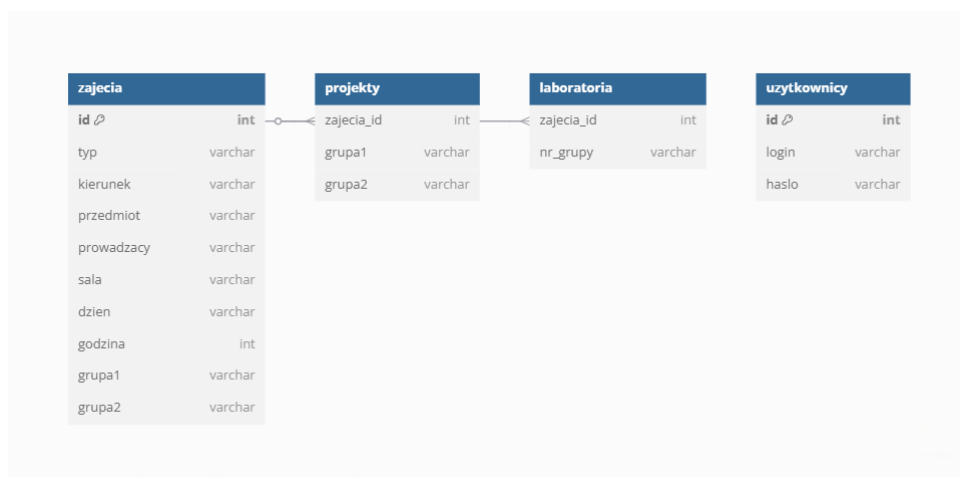
Dane przechowywane są w relacyjnej bazie danych **PostgreSQL**, w ramach schematu `public` bazy `javabase`. System korzysta z następujących tabel:

- **zajecia** – główna tabela przechowująca dane o wszystkich zajęciach:
 - `id`, `typ`, `kierunek`, `przedmiot`, `prowadzacy`, `sala`, `dzien`, `godzina`, `grupa1`, `grupa2`.
- **projekty** – zawiera przypisanie dwóch grup do zajęć typu Projekt:
 - `zajecia_id`, `grupa1`, `grupa2`.
- **laboratoria** – przechowuje numer grupy przypisanej do zajęć typu Laboratorium:
 - `zajecia_id`, `nr_grupy`.
- **uzytkownicy** – tabela logowania przechowująca dane uwierzytelniające użytkowników:
 - `id`, `login`, `haslo`.

Tabele są powiązane logicznie przez kolumnę `zajecia_id`, a dane zabezpieczone są poprzez ograniczenia integralności oraz indeksy. Struktura została zaprojektowana tak, aby umożliwiać wygodne wykonywanie operacji CRUD i filtrowania danych według sali, grupy i typu zajęć.

5.2. Diagram Baza Danych

Na poniższym diagramie (Rys. 8) przedstawiono strukturę relacyjnej bazy danych systemu. Schemat zawiera powiązania między tabelą `zajecia`, która stanowi centralny element, a tabelami powiązanymi `projekty` oraz `laboratoria`, które przechowują szczegóły zależne od typu zajęć. Tabela `uzytkownicy` umożliwia kontrolę dostępu do systemu.



Rys. 8. Diagram relacyjny bazy danych systemu

5.3. Interfejs użytkownika

Interfejs został wykonany w technologii **Swing**. Główne okna aplikacji to:

- **Ekran logowania** – umożliwia dostęp tylko zalogowanym użytkownikom;
- **Panel sekretariatu** – pozwala na przeglądanie i zarządzanie zajęciami;
- **Formularz dodawania zajęć** – umożliwia wprowadzenie nowych zajęć;
- **Formularz edycji zajęć** – pozwala na modyfikację istniejących rekordów.

5.4. Walidacja i błędy

System zawiera zabezpieczenia:

- Sprawdzanie dostępności sali w wybranym dniu i godzinie;
- Sprawdzanie, czy dana grupa nie ma już zajęć w tym czasie;
- Obsługa wyjątków SQL i wyświetlanie komunikatów błędów użytkownikowi.

5.5. Szczegółowy opis komponentów GUI

5.5.1. Formularz Dodawania Zajęć (DodajZajeciaPanel)

- **Cel:** Dodawanie nowych rekordów zajęć do bazy danych.
- **Elementy:**
 - `JComboBox` – typ zajęć (np. Wykład, Laboratorium, Projekt);
 - `TextField` – kierunek, przedmiot, prowadzący, sala, godzina, grupa1, grupa2;
 - `JComboBox` – dzień tygodnia;
 - `Button` – „Zapisz” — zapisuje dane do bazy danych.
- **Uwagi:** Interfejs zawiera etykiety (`JLabel`) przypisane do każdego pola. Formularz obsługuje walidację przed zapisem zajęć.

5.5.2. Formularz Edytowania Zajęć (EdytujZajeciaPanel)

- **Cel:** Edytowanie istniejących danych zajęć.
- **Elementy:** Te same co w `DodajZajeciaPanel`, jednak służą do aktualizacji danych:
 - Pola są wstępnie wypełnione danymi z wybranego rekordu;
 - `Button` – „Zapisz” — aktualizuje dane w bazie.
- **Uwagi:** Pola są edytowalne i automatycznie uzupełniane na podstawie danych wybranych z tabeli.

5.5.3. Formularz Logowania (Login)

- **Cel:** Autoryzacja użytkownika w systemie.
- **Elementy:**
 - `TextField` – login;
 - `PasswordField` – hasło;
 - `Button` – „Zaloguj” — weryfikuje dane logowania.
- **Uwagi:** Prosty, nowoczesny wygląd z ikoną kłódki; Zabezpieczenie dostępu do aplikacji.

5.5.4. Panel Sekretariatu (SekretariatPanel)

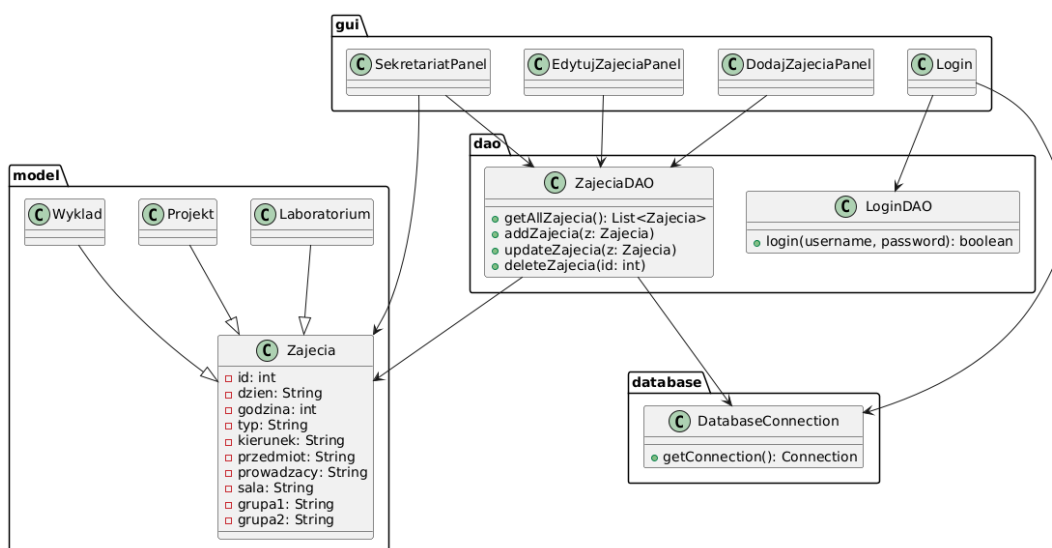
- **Cel:** Zarządzanie zajęciami (CRUD).
- **Elementy:**
 - `JTable` – wyświetlanie listy zajęć;
 - `JComboBox` – filtrowanie po sali, grupie, typie zajęć;
 - `Button` – „Dodaj”, „Usuń”, „Edytuj”, „Zastosuj filtry”, „Odśwież”;
 - `JScrollPane` – przewijana tabela;
 - `JLabel` – etykiety opisowe.
- **Uwagi:** Obsługuje filtrowanie i pełną obsługę CRUD; główny ekran zarządzania.

5.6. Diagram komponentów GUI i zależności

Diagram przedstawiony na Rys. 9 ilustruje architekturę aplikacji oraz przepływ zależności pomiędzy jej głównymi komponentami. Aplikacja została zaprojektowana zgodnie z podejściem warstwowym (ang. layered architecture), które oddziela logikę GUI, model danych, logikę dostępu do danych (DAO) oraz warstwę połączenia z bazą danych.

- Komponenty pakietu `gui` odpowiadają za graficzny interfejs użytkownika: `SekretariatPanel`, `DodajZajeciaPanel`, `EdytujZajeciaPanel`, `Login`.
- Komponenty modelu (`Zajecia`, `Wyklad`, `Projekt`, `Laboratorium`) reprezentują dane przekazywane między warstwami.
- Pakiet `dao` zawiera klasy `ZajeciaDAO` oraz `LoginDAO`, odpowiedzialne za dostęp do bazy danych poprzez odpowiednie metody (np. `getAllZajecia()`, `addZajecia()`, `login()`).
- Warstwa `database` odpowiada za ustanowienie połączenia z bazą danych — klasę `DatabaseConnection`.

Zależności między komponentami są jednokierunkowe, co sprzyja testowalności i rozszerzalności systemu. GUI komunikuje się wyłącznie z DAO, które obsługuje logikę zapisu i odczytu danych z relacyjnej bazy danych PostgreSQL.



Rys. 9. Zależności pomiędzy komponentami GUI, modelem danych i warstwą DAO

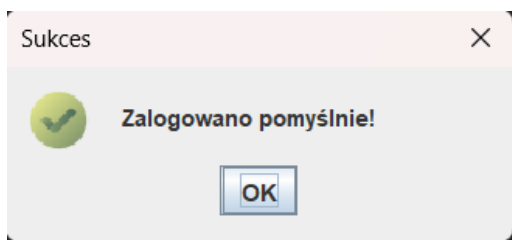
6. Testowanie systemu

System został przetestowany manualnie poprzez interfejs graficzny oraz poprzez analizę zapytań SQL wykonywanych przez klasę `ZajeciaDAO`. Testy przeprowadzono w środowisku lokalnym z bazą danych PostgreSQL.

6.1. Testy logowania

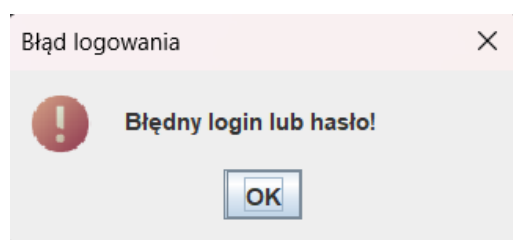
Przetestowano poprawność działania formularza logowania.

Poprawne logowanie – użytkownik uzyskuje dostęp do systemu



Rys. 10. Logowanie zakończone sukcesem

Niepoprawne logowanie – system wyświetla komunikat błędu



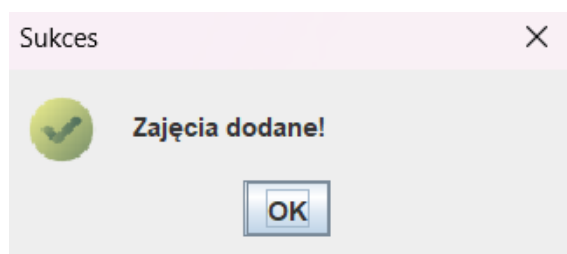
Rys. 11. Błąd logowania – nieprawidłowe dane

6.2. Testy dodawania zajęć

Przetestowano dodawanie różnych typów zajęć:

- Dodanie zajęć typu `Projekt` – poprawne wstawienie danych do bazy;
- Sprawdzenie walidacji grupy i sali – przy próbie konfliktu system wyświetla stosowny komunikat;
- Dodanie zajęć tylko dla jednej grupy (np. `grupaA`) działa prawidłowo.

Pomyślne dodanie zajęć – system wyświetla komunikat potwierdzający



Rys. 12. Potwierdzenie dodania zajęć

6.3. Testy filtrowania danych

Sprawdzono poprawność działania filtrów:

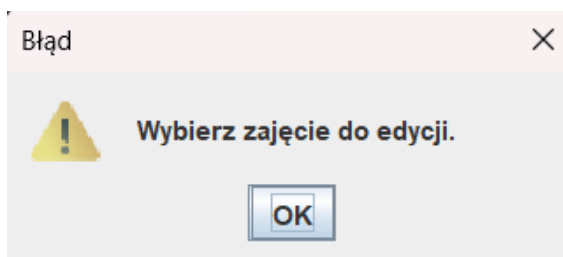
- Filtrowanie po dniu i grupie – poprawne ograniczenie wyników;
- Filtrowanie po typie zajęć i przedmiocie – wyświetlane są tylko pasujące rekordy;
- Filtrowanie nieistniejących danych – system poprawnie wyświetla pustą tabelę bez błędów.

6.4. Testy edycji i usuwania zajęć

Przetestowano operacje modyfikacji i usuwania danych:

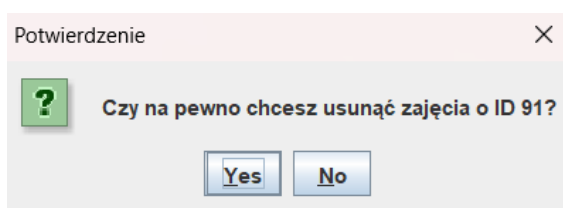
- Edycja sali i prowadzącego – zmiany są natychmiast widoczne w tabeli GUI;
- Usunięcie zajęć z bazy powoduje ich zniknięcie z interfejsu;
- Obsługa wyjątków SQL przy próbie edycji nieistniejącego rekordu działa prawidłowo.

Brak zaznaczonego wiersza – system przypomina o konieczności wyboru przed edycją



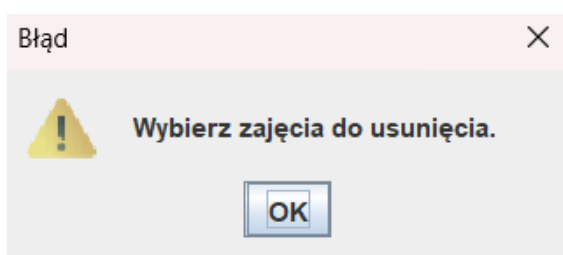
Rys. 13. Brak zaznaczonego rekordu do edycji

Podczas usuwania rekordów użytkownik otrzymuje pytanie potwierdzające



Rys. 14. Potwierdzenie usunięcia zajęć

Brak zaznaczonego wiersza – system informuje o konieczności wyboru przed usunięciem



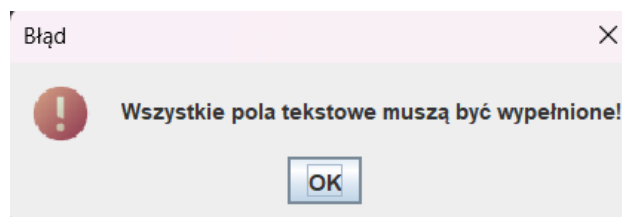
Rys. 15. Brak zaznaczonego rekordu do usunięcia

6.5. Testy walidacji danych

Sprawdzono poprawność działania walidacji formularza:

- Puste pola – system wyświetla komunikat błędu i nie zapisuje danych;
- Nieprawidłowe dane (np. litery w polu godzina) – poprawna obsługa błędu i komunikat;
- Próba dodania zajęć do już zajętej sali – system blokuje takie dodanie;
- Konflikt grupy – użytkownik otrzymuje odpowiedni alert.

Użytkownik nie uzupełnił pól tekstowych — system wyświetla komunikat



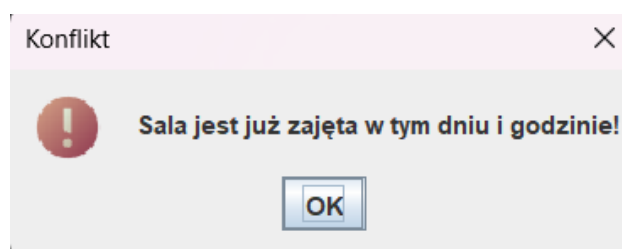
Rys. 16. Puste pola formularza – komunikat błędu

Opis: Ten fragment kodu sprawdza, czy pola formularza nie są puste. Jeśli są – zostaje wyświetlony komunikat, a operacja zostaje przerwana.

Listing 1. Walidacja pustych pól

```
if (kierunek.isEmpty() || przedmiot.isEmpty() || prowadzacy.isEmpty()) {  
    JOptionPane.showMessageDialog(null, "Wszystkie pola tekstowe muszą być wypełnione!",  
        "Błąd", JOptionPane.WARNING_MESSAGE, warningIcon);  
    return;  
}
```

Próba dodania zajęć do już zajętej sali — pojawia się komunikat o konflikcie



Rys. 17. Błąd – sala już zajęta w tym terminie

Opis: Fragment ten sprawdza, czy sala jest dostępna o wybranej porze. W razie konfliktu, informuje użytkownika i przerywa operację.

Listing 2. Sprawdzanie dostępności sali

```
if (dao.czySalaZajeta(sala, dzien, godzina, null)) {  
    JOptionPane.showMessageDialog(null, "Sala jest już zajęta w tym dniu i godzinie!",  
        "Konflikt", JOptionPane.WARNING_MESSAGE, warningIcon);  
    return;  
}
```

Opis: Ten kod odpowiada za informowanie użytkownika o konflikcie w harmonogramie danej grupy. Obsługuje przypadek, gdy grupa ma już zajęcia w wybranym terminie.

Listing 3. Sprawdzenie konfliktu grupy

```
JOptionPane.showMessageDialog(null, "Wybrana grupa ma już zajęcia w tym dniu i godzinie!",  
    "Konflikt", JOptionPane.WARNING_MESSAGE, warningIcon);  
return;
```

7. Podsumowanie i wnioski

Celem niniejszego projektu było zaprojektowanie i zaimplementowanie funkcjonalnego systemu rezerwacji sal oraz zarządzania harmonogramem zajęć dla instytucji edukacyjnej, przy wykorzystaniu języka Java, biblioteki Swing do tworzenia graficznego interfejsu użytkownika, oraz bazy danych PostgreSQL jako backendu do przechowywania danych. Projekt miał na celu połączenie teorii poznanej w trakcie studiów z praktycznym zastosowaniem w formie kompletnej aplikacji.

7.1. Osiągnięte rezultaty

W wyniku realizacji projektu udało się z powodzeniem zaimplementować następujące funkcjonalności:

- Stworzono graficzny interfejs użytkownika umożliwiający intuicyjne zarządzanie danymi o zajęciach;
- Zaimplementowano możliwość dodawania, edytowania oraz usuwania informacji o zajęciach bezpośrednio z poziomu GUI;
- Wprowadzono zaawansowane filtrowanie według sali, grupy oraz typu zajęć;
- Zapewniono walidację danych wejściowych oraz obsługę najczęstszych wyjątków logicznych i technicznych;
- Stworzono strukturę kodu zgodną z podejściem obiektowym, z wyodrębnieniem warstwy dostępu do danych (DAO), modeli danych oraz interfejsów graficznych;
- Połączono aplikację z relacyjną bazą danych PostgreSQL za pomocą technologii JDBC.

Projekt został wykonany zgodnie z założeniami oraz przyjętym harmonogramem. Aplikacja przeszła serię testów funkcjonalnych, które potwierdziły poprawność działania wszystkich podstawowych funkcji systemu.

7.2. Wnioski

W trakcie realizacji projektu zdobyto cenne doświadczenie w zakresie projektowania i implementacji aplikacji desktopowych. W szczególności pogłębiono umiejętności w następujących obszarach:

- stosowanie zasad programowania obiektowego w praktycznych projektach;
- projektowanie i realizacja graficznych interfejsów użytkownika z użyciem Java Swing;
- budowa aplikacji z dostępem do relacyjnej bazy danych z użyciem JDBC oraz języka SQL;
- stosowanie walidacji danych oraz obsługi wyjątków na poziomie interfejsu i logiki aplikacyjnej;
- organizacja kodu w warstwach oraz stosowanie wzorców projektowych takich jak DAO;
- testowanie i debugowanie aplikacji desktopowych.

System spełnia wszystkie założenia funkcjonalne i stanowi solidną podstawę do dalszego rozwoju. W przyszłości możliwe jest rozszerzenie systemu o dodatkowe funkcjonalności, takie jak:

- mechanizm rejestracji i logowania użytkowników z różnymi poziomami dostępu;
- eksport danych do formatu PDF lub CSV;
- integracja z zewnętrznymi systemami kalendarzowymi (np. Google Calendar);
- rozwinięcie systemu o obsługę powiadomień e-mail dla prowadzących i studentów;
- przekształcenie projektu w aplikację webową z użyciem nowoczesnych frameworków.

Zrealizowany projekt potwierdza możliwość tworzenia złożonych systemów informatycznych nawet w ramach ograniczonego czasu i zasobów, pod warunkiem zastosowania właściwych praktyk inżynierii oprogramowania.

7.3. Linki do repozytoriów GitHub

Źródła projektu oraz dokumentacji dostępne są w publicznych repozytoriach GitHub:

- Link do projektu: https://github.com/yerzan/FINALY_PROJECT
- Link do dokumentacji \LaTeX : https://github.com/yerzan/Latex_java

Bibliografia

- [1] Michael Blaha and James Rumbaugh. *Object-Oriented Modeling and Design with UML*. Pearson, 2005.
- [2] Cay S. Horstmann. *Core Java Volume I – Fundamentals*. Pearson Education, 12th edition, 2022.

Spis rysunków

1	Harmonogram realizacji projektu w formie wykresu Gantta	5
2	Panel logowania do systemu	6
3	Główny panel sekretariatu z listą zajęć i filtrowaniem	7
4	Formularz dodawania zajęć typu Wykład	8
5	Formularz dodawania zajęć typu Laboratorium	9
6	Formularz dodawania zajęć typu Projekt	10
7	Formularz edycji istniejących zajęć	11
8	Diagram relacyjny bazy danych systemu	12
9	Zależności pomiędzy komponentami GUI, modelem danych i warstwą DAO	15
10	Logowanie zakończone sukcesem	16
11	Błąd logowania – nieprawidłowe dane	16
12	Potwierdzenie dodania zajęć	16
13	Brak zaznaczonego rekordu do edycji	17
14	Potwierdzenie usunięcia zajęć	17
15	Brak zaznaczonego rekordu do usunięcia	17
16	Puste pola formularza – komunikat błędu	18
17	Błąd – sala już zajęta w tym terminie	18

Spis listingów

1	Walidacja pustych pól	18
2	Sprawdzanie dostępności sali	18
3	Sprawdzenie konfliktu grupy	19

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

.....Mykhailo Kleban.....
Imię (imiona) i nazwisko studenta

Wydział Nauk Ścisłych i Technicznych

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

.....134922.....
Numer albumu

1. Oświadczam, że moja praca projektowa pt.: System rezerwacji sal/podział godzin
 - 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 na udostępnienie mojej pracy projektowej do celów naukowo-badawczych z poszanowaniem przepisów ustawy o prawie autorskim i prawach pokrewnych.

Rzeszów, 15.06.2025
(miejscowość, data)

Mykhailo Kleban
(czytelny podpis studenta)