

Politechnika Wrocławska  
Wydział Elektroniki  
Kierunek Automatyka i Robotyka

---

BAZY DANYCH (PROJEKT)  
TEMAT 9. ELEKTRONICZNY INDEKS WYŻSZEJ UCZELNI.

---

*Autorzy:*  
MICHAŁ PROŚBA  
PATRYK WIECZOREK

*Prowadzący:*  
Dr. inż. Roman Ptak  
Mgr. inż. Norbert Kozłowski

*Termin:*  
Środa 9:15

25 maja 2021

# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>3</b>
1.1	Cel projektu . . . . .	3
1.2	Harmonogram prac . . . . .	3
1.3	Wstępne wymagania klienta . . . . .	3
<b>2</b>	<b>Analiza wymagań</b>	<b>3</b>
2.1	Wymagania funkcjonalne . . . . .	3
2.1.1	Niezałogowany użytkownik . . . . .	3
2.1.2	Student . . . . .	3
2.1.3	Pracownik dziekanatu . . . . .	4
2.1.4	Prowadzący . . . . .	4
2.1.5	Administrator . . . . .	4
2.2	Wymagania niefunkcjonalne . . . . .	4
2.2.1	Rozmiar bazy danych . . . . .	4
2.2.2	Wykorzystane technologie . . . . .	4
2.2.3	Bezpieczeństwo systemu . . . . .	5
<b>3</b>	<b>Projekt systemu</b>	<b>5</b>
3.1	Projekt bazy danych . . . . .	5
3.1.1	Analiza rzeczywistości i uproszczony model konceptualny . . . . .	5
3.1.2	Model logiczny i normalizacja . . . . .	6
3.2	Model fizyczny i ograniczenia integralności danych . . . . .	7
3.3	Sekwencje . . . . .	7
3.4	Widoki . . . . .	7
3.5	Procedury składowe . . . . .	10
3.6	Wyzwalacze . . . . .	14
3.7	Mechanizmy bezpieczeństwa na poziomie bazy danych . . . . .	15
3.8	Projekt aplikacji użytkownika . . . . .	16
3.8.1	Architektora aplikacji i diagramy projektowe . . . . .	16
3.9	Interfejs graficzny . . . . .	17
3.10	Metoda podłączenia do bazy danych . . . . .	17
<b>4</b>	<b>Tworzenie bazy danych</b>	<b>17</b>
4.1	Tworzenie tabel, relacji oraz definicja ograniczeń w bazie danych . . . . .	17
4.2	Generacja kodu SQL . . . . .	19
4.3	Implementacja widoków . . . . .	24
4.3.1	Widok - "dane_studenta_view" . . . . .	24
4.3.2	Widok - "dane_uzytkownika_view" . . . . .	24
4.3.3	Widok - "prowadzone_kursy_prowadzacego" . . . . .	25
4.3.4	Widok - "grupy_kursu_prowadzacego" . . . . .	25
4.3.5	Widok - "studenci_prowadzacego" . . . . .	26
4.3.6	Widok - "oceny_studenta" . . . . .	26
4.3.7	Widok - "plan_zajec_studenta" . . . . .	27
4.3.8	Widok - "lista_uzytkownikow" . . . . .	27
4.3.9	Widok - "przeglądanie_grup" . . . . .	28
4.4	Implementacja procedur składowanych . . . . .	28
4.4.1	Procedura składowana - "wyswietl_dane_studenta" . . . . .	28
4.4.2	Procedura składowana - "wyswietl_dane_uzytkownika" . . . . .	28
4.4.3	Procedura składowana - "wyswietl_grupy_kursu_prowadzacego" . . . . .	29
4.4.4	Procedura składowana - "wyswietl_oceny_studenta" . . . . .	29
4.4.5	Procedura składowana - "wyswietl_plan_zajec_studenta" . . . . .	29
4.4.6	Procedura składowana - "wyswietl_prowadzone_kursy_prowadzacego" . . . . .	29
4.4.7	Procedura składowana - "wyswietl_studentow_prowadzacego" . . . . .	29
4.4.8	Procedura składowana - "dodaj_grupe_zajeciowa" . . . . .	30
4.4.9	Procedura składowana - "dodaj_kierunek" . . . . .	30
4.4.10	Procedura składowana - "dodaj_kurs" . . . . .	30
4.4.11	Procedura składowana - "dodaj_uzytkownika" . . . . .	30

4.4.12	Procedura składowana - "zapisz_studenta"	33
4.4.13	Procedura składowana - "odrzuc_reklamacje"	33
4.4.14	Procedura składowana - "popraw_ocene"	34
4.4.15	Procedura składowana - "reklamuj_ocene"	34
4.4.16	Procedura składowana - "usun_uzytkownika"	34
4.4.17	Procedura składowana - "wprowadz_ocene"	35
4.4.18	Procedura składowana - "zatwierdz_ocene"	35
4.4.19	Procedura składowana - "zmien_imie"	35
4.4.20	Procedura składowana - "zmien_kod_pocztowy"	35
4.4.21	Procedura składowana - "zmien_kraj_zamieszkania"	36
4.4.22	Procedura składowana - "zmien_miasto"	36
4.4.23	Procedura składowana - "zmien_nazwisko"	36
4.4.24	Procedura składowana - "zmien_numer_domu"	36
4.4.25	Procedura składowana - "zmien_numer_kontaktowy"	36
4.4.26	Procedura składowana - "zmien_numer_lokalu"	37
4.4.27	Procedura składowana - "zmien_plec"	37
4.4.28	Procedura składowana - "zmien_ulice"	37
4.5	Model bazy danych z tabelami, widokami oraz procedurami składowymi	37
4.6	Implementacja zabezpieczeń	38
4.7	Testy procedur	40
4.8	Testowanie widoków	46
4.9	Testy wydajnościowe	49
<b>5</b>	<b>Implementacja i testy aplikacji</b>	<b>52</b>
5.1	Instrukcja użytkownika aplikacji	52
<b>6</b>	<b>Testy aplikacji</b>	<b>67</b>
6.1	Test wprowadzenia błędnego hasła	67
6.2	Test zmiany danych osobowych użytkownika	67
6.3	Test wprowadzania oceny przez prowadzącego	68
6.4	Test zatwierdzenia oceny przez studenta	69
6.5	Dodanie studenta przez pracownika dziekanatu	69
6.6	Dodanie użytkownika przez administratora	70
<b>7</b>	<b>Dodatkowe mechanizmy</b>	<b>72</b>
7.1	Wersjonowanie bazy danych	72
7.2	Disaster recovery	73
7.3	Skalowanie bazy danych	74
7.4	Replikacja bazy danych	74
7.5	Zabezpieczenie przed SQL injection	74
7.6	Strategia aktualizacji bazy danych	75
<b>8</b>	<b>Podsumowanie i wnioski</b>	<b>75</b>

# 1 Wstęp

## 1.1 Cel projektu

Celem projektu jest stworzenie aplikacji desktopowej wraz z bazą danych w postaci elektronicznego indeksu wyższej uczelni. Aplikacja, będzie pozwalała na obsługę użytkowników takich jak: studenci, pracownicy dziekanatu, prowadzący oraz administratorzy. System ma pozwolić na swobodny dostęp do funkcjonalności adekwatnej dla danej grupy użytkowników.

## 1.2 Harmonogram prac

- Przygotowanie i uzasadnienie potrzeby realizacji. Analiza problemu.
- Faza projektowa
- Implementacja i walidacja
- Wdrożenie aplikacji
- Prezentacja na forum publicznym i zaliczenie całości.

## 1.3 Wstępne wymagania klienta

Politechnika zleciła stworzenie elektronicznego indeksu wyższej uczelni. System aktualnie używany na Politechnice nie spełnia wymagań prowadzących oraz studentów, konieczna więc jest implementacja nowego rozwiązania. Przy projekcie wymagane jest, aby następujące kryteria zostały zachowane:

- Zaprojektowanie aplikacji desktopowej, która będzie obsługiwała indeks elektroniczny uczelni
- Aplikacja musi być kompatybilna z systemem Windows
- Każdy użytkownik ma swoje indywidualne konto
- Aplikacja umożliwia rozgraniczenie funkcji poszczególnych użytkowników (studenci, pracownicy dziekanatu, prowadzący, administratorzy)
- Aplikacja ma umożliwiać filtrację wyświetlanych danych (odnośnie np. studentów, ocen, zajęć dydaktycznych)
- Aplikacja desktopowa ma pozwolić na przejrzystą prezentację danych
- Informacje mają być przechowywane w lokalnej bazie danych
- Aplikacja powinna być lokalna (brak dostępu za pośrednictwem internetu)
- Studenci mają tylko możliwość odczytu bazy danych i modyfikacji parametrów swojego konta
- Pracownicy dziekanatu i prowadzący mają ograniczony dostęp do zmian w bazie danych
- Administratorzy mają pełne uprawnienia w systemie

# 2 Analiza wymagań

## 2.1 Wymagania funkcjonalne

### 2.1.1 Niezalogowany użytkownik

- Możliwość zalogowania się

### 2.1.2 Student

- Możliwość wyświetlenia, planu zajęć aktualnego studenta
- Możliwość wyświetlenia aktualnych prowadzących z danego przedmiotu
- Możliwość weryfikowania ocen z poszczególnych przedmiotów
- Edycja parametrów swojego konta

### **2.1.3 Pracownik dziekanatu**

- Dodawanie i usuwanie studentów z listy studentów
- Edycja parametrów kont studentów
- Edycja parametrów swojego konta

### **2.1.4 Prowadzący**

- Dostęp do prowadzonych przedmiotów
- Dostęp do listy studentów uczęszczających do danego prowadzącego na dany przedmiot
- Wprowadzanie ocen
- Edycja parametrów swojego konta

### **2.1.5 Administrator**

- Dodawanie i usuwanie studentów, pracowników dziekanatu oraz prowadzących
- Edycja parametrów danej grupy użytkowników
- Edycja parametrów swojego konta

## **2.2 Wymagania niefunkcjonalne**

### **2.2.1 Rozmiar bazy danych**

- 1 - 2 administratorów
- 5 - 10 pracowników dziekanatu
- 10 - 100 prowadzących
- 1000 - 5000 studentów
- 1 - 5 kierunków studiów
- 100 - 500 kursów
- 1000 - 5000 grup zajęciowych
- 1000000 - 5000000 ocen

### **2.2.2 Wykorzystane technologie**

- System operacyjny: Windows.
- Języki programowania: SQL, C#.
- System zarządzania bazą danych: MySQL.
- Program do modelowania baz danych: MySQL Workbench
- Rodzaj aplikacji: Desktopowa
- Praca w sieci lokalnej: Tak
- Interfejs graficzny: Tak
- Dostęp do internetu: Nie

### 2.2.3 Bezpieczeństwo systemu

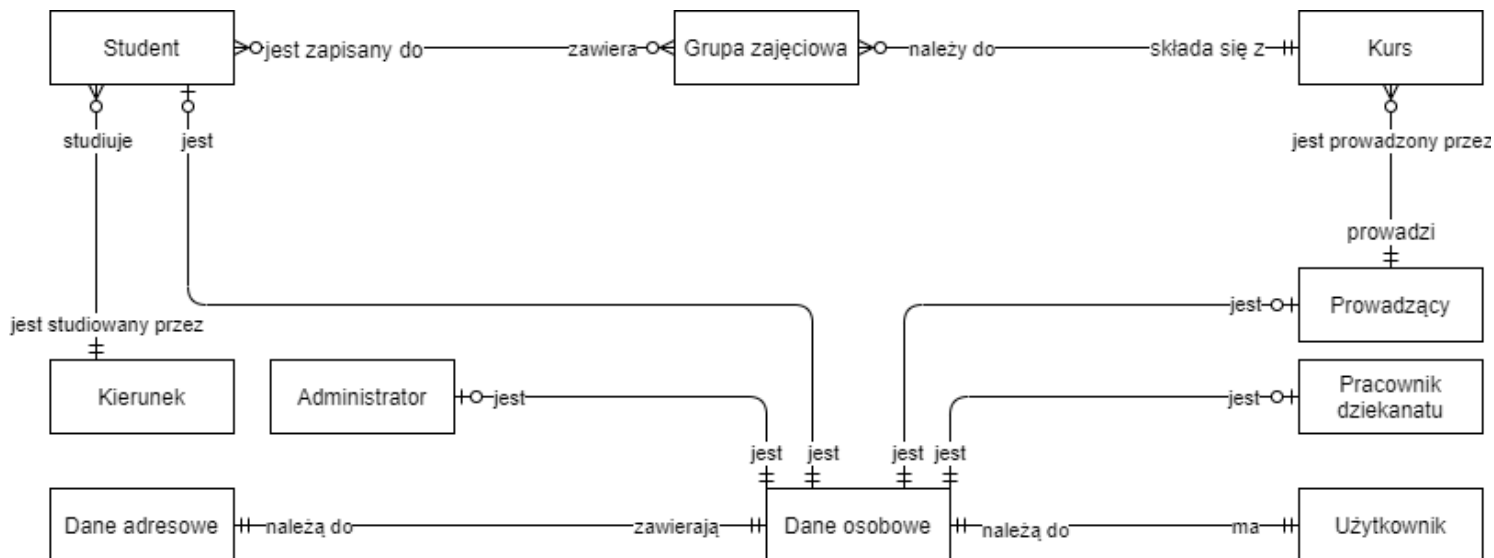
- System upoważnia do dostępu tylko zarejestrowanych użytkowników
- Logowanie za pomocą loginu i hasła
- Użytkownicy zostaną podzieleni na grupy: student, pracownik dziekanatu, prowadzący, administrator, z adekwatnymi uprawnieniami

## 3 Projekt systemu

### 3.1 Projekt bazy danych

#### 3.1.1 Analiza rzeczywistości i uproszczony model konceptualny

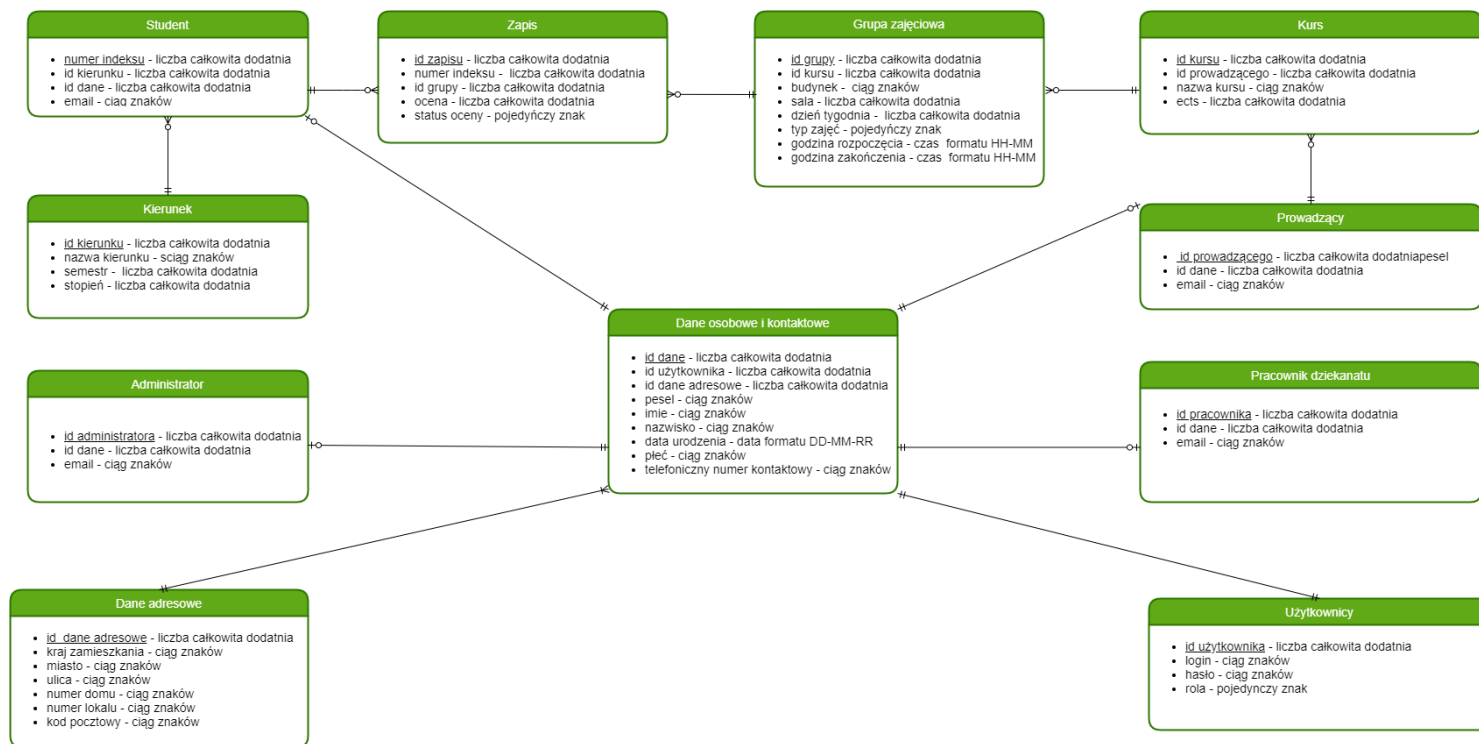
Projekt zakłada aplikację desktopową połączoną z bazą danych, umożliwiającą obsługę użytkowników Politechniki. Użytkownicy podzieleni są na grupy. Każda grupa posiada swoje uprawnienia. Grupy takie jak pracownicy dziekanatu oraz administratorzy, posiadają dodatkowe funkcjonalności związane z edycją parametrów kont, czy też dodawaniem i usuwaniem użytkowników innych grup. W tym punkcie funkcjonalności te zostały pominięte. W modelu konceptualnym zostały wyodrębnione encje, niezbędne do implementacji systemu. Każdy student, prowadzący, pracownik dziekanatu oraz administrator jest równocześnie użytkownikiem, który posiada unikatowe dane. Osoba przynależąca do jednej z grup może równocześnie przynależać do innej grupy. Przykładowo pracownik dziekanatu może być jednocześnie administratorem. Dla uproszczenia przyjmujemy, że jeden student może studiować tylko jeden kierunek. Natomiast kierunek może być studiowany przez wielu lub żadnego studenta, ponieważ przywidujemy opcję, w której powstał kierunek i żaden student nie jest jeszcze do niego zapisany. Student dodatkowo może być zapisany do wielu lub żadnej grupy zajęciowej oraz tak samo grupa zajęciowa zawiera wielu lub żadnego studenta. Grupa zajęciowa składa się tylko z jednego kursu. Natomiast kurs należy do wielu lub żadnej grupy zajęciowej. Kurs jest prowadzony przez jednego prowadzącego, natomiast prowadzący może prowadzić wiele kursów lub ani jednego.



Rysunek 1: Model konceptualny

### 3.1.2 Model logiczny i normalizacja

W modelu logicznym poszczególnym encję przyporządkowaliśmy atrybuty oraz określiliśmy w jaki sposób będziemy je zapamiętywać. Zostało podane jakie typy wartości będą potrzebne do ich zapamiętania. Nazwy typów zmiennych nie zostały określone dokładnie, ponieważ zależą od języka, w którym baza danych zostanie zaimplementowana. W modelu logicznym została także dokonana normalizacja, czyli pozbyliśmy się nadmiarowych informacji. Przykładowo kurs zawiera tylko id prowadzącego zamiast zawierać informację o imieniu i nazwisku prowadzącego, ponieważ posiadając id prowadzącego możemy pozyskać informacje o jego imieniu i nazwisku. Dodatkowo związki wielu do wielu jak w przypadku studenta oraz grupy zajęciowej zostały rozdzielone relacjami z tabelami pośredniczącymi. Każdy użytkownik posiada atrybuty takie jak login, hasło oraz rola pozwalająca rozróżnić do jakiej grupy przynależy użytkownik. Dodatkowo w tabeli użytkownicy został dodany atrybut klucza obcego id dane, pozwalający na przyporządkowanie każdemu użytkownikowi jego unikalnych danych osobowych. Zakładamy, że dane osobowe każdego z użytkowników składają się z tych samych atrybutów, takich jak pesel, imię, nazwisko, data urodzenia, płeć, adres, numer kontaktowy. Encje studenta, administratora, prowadzącego posiadają atrybut id użytkownika dzięki czemu jesteśmy w stanie sprawdzić, którym użytkownikiem jest w encji użytkownicy. Encja student posiada dodatkowy atrybut id kierunku pozwalający zweryfikować kierunek, semestr oraz stopień studiów, ponieważ w encji kierunek znajdują się właśnie takie atrybuty. Encja przynależności łączy studenta z grupą zajęciową dlatego zawiera atrybuty takie jak numer indeksu oraz id grupy. Dodatkowo w tej encji znajdują się atrybuty ocena oraz status oceny. Encja grup zajęciowych posiada atrybuty związane z samą grupą zajęciową takie jak budynek, sala, dzień tygodnia, typ zajęć, godzina rozpoczęcia oraz godzina zakończenia. Dodatkowo podany jest atrybut id kursu pozwalający na weryfikację do jakiego kursu należy grupa zajęciowa. Encja kurs posiada atrybuty nazwa kursu, ects oraz id prowadzącego pozwalające na weryfikację, jaki prowadzący prowadzi dany kurs.

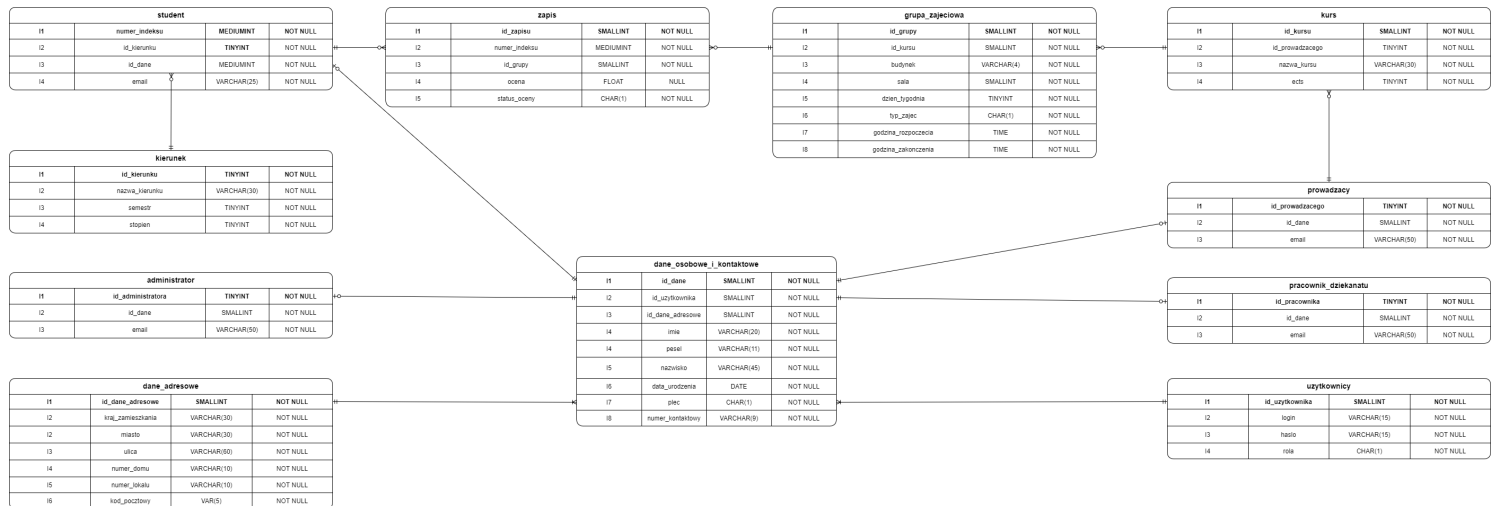


Rysunek 2: Model logiczny

## 3.2 Model fizyczny i ograniczenia integralności danych

W modelu fizycznym określiliśmy dokładne typy zmiennych, uwzględniając zmienne występujące w języku SQL, ponieważ bazę danych będziemy implementować za pomocą MySQL Workbench. Dodatkowo w encjach zostały wyodrębnione klucze główne (Primary Key) oraz umieszczone klucze pomocnicze (Foreign Key). Baza jest stosunkowo duża przez co zostały wprowadzone pewne ograniczenia:

- wszystkie dane typu całkowitego takie jak TINYINT, SMALLINT oraz MEDIUMINT są liczbami nieujemnymi (unsigned)
- wszystkie dane typu tekstowego są ograniczone do pewnej maksymalnej ilości znaków jakie mogą przechowywać



Rysunek 3: Model fizyczny

## 3.3 Sekwencje

- Sekwencje w przypadku encji student rozpoczynamy od wartości 240000, kolejne numery indeksu są zrealizowane jako AUTOINKREMENT.
- Sekwencja dla pozostałych kluczy głównych (Primary Key) została zrealizowana jako AUTOINKREMENT.

## 3.4 Widoki

- Nazwa widoku: dane.studenta\_view
  - Użyte tabele: kierunek, student, dane\_osobowe\_i\_kontaktowe
  - Tabela wynikowa: 5 kolumny:
    - dane\_osobowe\_i\_kontaktowe.id\_uzytkownika
    - kierunek.nazwa\_kierunku
    - kierunek.nazwa\_stopien
    - kierunek.nazwa\_semestr
    - student.numer\_albumu
  - Tabela zawiera informacje o danych studenta



- – Nazwa widoku: dane\_uzytkownika\_view
- Użyte tabele: uzytkownik, dane\_osobowe, dane\_adresowe
- Tabela wynikowa: 11 kolumnny:
  1. dane\_osobowe.id\_uzytkownika
  2. dane\_osobowe.imie
  3. dane\_osobowe.nazwisko
  4. dane\_osobowe.pesel
  5. dane\_osobowe.date\_urodzenia
  6. dane\_osobowe.plec
  7. dane\_osobowe.numer\_kontaktowy
  8. dane\_adresowe.kraj\_zamieszkania
  9. dane\_adresowe.miasto
  10. dane\_adresowe.ulica
  11. dane\_adresowe.numer\_domu
  12. dane\_adresowe.numer\_lokalu
- Tabela zawiera informacje o danych użytkownika
  
- – Nazwa widoku: prowadzone\_kursy\_prowadzacego\_view
- Użyte tabele: dane\_osobowe\_i\_kontaktowe, prowadzacy, kurs
- Tabela wynikowa: 2 kolumnny:
  1. dane\_osobowe\_id\_kontaktowe.id\_uzytkownika
  2. kurs.nazwa\_kursu
  3. kurs.ects
- Tabela zawiera informacje o prowadzonych kursach prowadzącego
  
- – Nazwa widoku: grupy\_kursu\_prowadzacego\_view
- Użyte tabele: grupa\_zajeciowa, kurs, prowadzacy, dane\_osobowe\_i\_kontaktowe
- Tabela wynikowa: 6 kolumnny:
  1. dane\_osobowe\_i\_kontaktowe.id\_uzytkownika
  2. grupa\_zajeciowa.id\_grupy
  3. grupa\_zajeciowa.budynek
  4. grupa\_zajeciowa.sala
  5. grupa\_zajeciowa.dzien\_tygodnia
  6. grupa\_zajeciowa.typ\_zajec
  7. grupa\_zajeciowa.godzina\_rozpoczecia
  8. grupa\_zajeciowa.godzina\_zakonczenia
- Tabela zawiera informacje o grupach kursów prowadzonych przez prowadzącego
  
- – Nazwa widoku: studenci\_prowadzacego\_view
- Użyte tabele: dane\_osobowe\_i\_kontaktowe, prowadzacy, kurs, grupa\_zajeciowa, zapis, student
- Tabela wynikowa: 7 kolumnny:
  1. dane\_osobowe\_i\_kontaktowe.id\_uzytkownika
  2. student.numer\_indeksu
  3. student.id\_dane
  4. kurs.nazwa\_kursu
  5. grupa\_zajeciowa.id\_grupy
  6. grupa\_zajeciowa.typ\_zajec
  7. zapis.ocena
  8. zapis.status\_ocen
  9. Zapis.ocena

- Tabela zawiera informacje o planie lekcji studenta
- – Nazwa widoku: `oceny_studenta_view`
- Użyte tabele: `dane_osobowe_i_kontaktowe`, `prowadzacy`, `kurs`, `grupa_zajeciowa`, `zapis`, `student`
- Tabela wynikowa: 6 kolumnny:
  1. `student.id_dane`
  2. `dane_osobowe_i_kontaktowe.imie`
  3. `dane_osobowe_i_kontaktowe.nazwisko`
  4. `kurs.nazwa.kursu`
  5. `kurs.ects`
  6. `grupa_zajeciowa.id_grupy`
  7. `zapis.ocena`
  8. `zapis.status_oceny`
- Tabela zawiera informacje o ocenach studenta
- – Nazwa widoku: `plan_zajec_studenta_view`
- Użyte tabele: `dane_osobowe_i_kontaktowe`, `prowadzacy`, `kurs`, `grupa_zajeciowa`, `zapis`, `student`
- Tabela wynikowa: 10 kolumnny:
  1. `student.id_dane`
  2. `dane_osobowe_i_kontaktowe.imie`
  3. `dane_osobowe_i_kontaktowe.nazwisko`
  4. `kurs.nazwa.kursu`
  5. `kurs.ects`
  6. `grupa_zajeciowa.id_grupy`
  7. `grupa_zajeciowa.budynek`
  8. `grupa_zajeciowa.sala`
  9. `grupa_zajeciowa.dzien_tygodnia`
  10. `grupa_zajeciowa.typ_zajec`
  11. `grupa_zajeciowa.godzina_rozpoczecia`
  12. `grupa_zajeciowa.godzina_zakonczenia`
- Tabela zawiera informacje o planie lekcji studenta
- – Nazwa widoku: `lista_uzytkownikow_view`
- Użyte tabele: `Uzytkownicy`, `Dane_osobowe`
- Tabela wynikowa: 4 kolumnny:
  1. `uzytkownicy.id_uzytkownika`
  2. `dane_osobowe_i_kontaktowe.imie`
  3. `dane_osobowe_i_kontaktowe.nazwisko`
  4. `uzytkownicy.rola`
- Tabela zawiera informacje o wszystkich użytkownikach

- – Nazwa widoku: `przeglądanie_grup_view`
- Użyte tabele: `dane_osobowe.i.kontaktowe`, `przewodzący`, `kurs`, `grupa_zajeciowa`, `zapis`
- Tabela wynikowa: 9 kolumnny:
  1. `dane_osobowe.imie`
  2. `dane_osobowe.nazwisko`
  3. `kurs.nazwa_kursu`
  4. `grupa_zajeciowa.id_grupy`
  5. `grupa_zajeciowa.budynek`
  6. `grupa_zajeciowa.sala`
  7. `grupa_zajeciowa.dzien_tygodnia`
  8. `grupa_zajeciowa.typ_zajec`
  9. `grupa_zajeciowa.godzina_rozpoczecia`
  10. `grupa_zajeciowa.godzina_zakonczenia`
- Tabela zawiera informacje o planie lekcji studenta

### 3.5 Procedury składowe

#### ■ `dodaj_uzytkownika`

- Parametry:
  1. `pesel`
  2. `imie`
  3. `nazwisko`
  4. `data_urodzenia`
  5. `plec`
  6. `numer_kontaktowy`
  7. `kraj_zamieszkania`
  8. `miasto`
  9. `ulica`
  10. `numer_domu`
  11. `numer_lokalu`
  12. `kod_pocztowy`
  13. `login`
  14. `haslo`
  15. `rola`
  16. `kierunek`
  17. `semestr`
  18. `stopien`
  19. `id_uzytkownika` (wywołującego procedure)
- Do tabel `Uzytkownicy`, `Dane_adresowe`, `Dane_osobowe` dodajemy podane parametry, w zależności od podanej roli dodajemy też nową pozycję w jednej z tabel: `Student`, `Administrator`, `Pracownik_dziekanatu` lub `Prowadzący`.

#### ■ `usun_uzytkownika`

- Parametry:
  1. `id_uzytkownika` (którego chcemy usunąć)
  2. `id_uzytkownika` (wywołującego procedure)
- Korzystając z `id_uzytkownika` znajdujemy wszystkie informacje dotyczące danego użytkownika, a następnie je usuwamy.

#### ■ wprowadz\_ocene

- Parametry:
  1. numer\_indeksu
  2. id\_grupy
  3. ocena
  4. id\_uzytkownika (wywołującego procedure)
- Korzystając z podanych wprowadzamy ocenę końcową.

#### ■ zatwierdz\_ocene

- Parametry:
  1. id\_zapisu
  2. id\_uzytkownika (wywołującego procedure)
- Zatwierdzamy ocenę, która odpowiada danemu id\_zapis.

#### ■ reklamuj\_ocene

- Parametry:
  1. id\_zapisu
  2. id\_uzytkownika (wywołującego procedure)
- Reklamujemy ocenę, która odpowiada danemu id\_zapis.

#### ■ odrzuc\_reklamacje

- Parametry:
  1. numer\_indeksu
  2. id\_grupy
  3. id\_uzytkownika (wywołującego procedure)
- Odrzucamy reklamację, która odpowiada danemu id\_zapis.

#### ■ popraw\_ocene

- Parametry:
  1. numer\_indeksu
  2. id\_grupy
  3. ocena
  4. id\_uzytkownika (wywołującego procedure)
- Korzystając z podanych parametrów poprawiamy ocenę przypisaną do danego id\_zapisu uznając w ten sposób reklamację studenta.

#### ■ dodaj\_grupe\_zajeciowa

- Parametry:
  1. id\_kursu
  2. budynek
  3. sala
  4. dzien\_tygodnia
  5. typ\_zajec
  6. godzina\_rozpoczecia
  7. godzina\_zakonczenia
  8. id\_uzytkownika (wywołującego procedure)
- Korzystając z podanych parametrów dodajemy grupę zajęciową.

#### ■ dodaj\_kierunek

- Parametry:
  1. nazwa
  2. semestr
  3. stopien
  4. id\_uzytkownika (wywołującego procedure)
- Korzystając z podanych parametrów dodajemy kierunek.

#### ■ dodaj\_kurs

- Parametry:
  1. id\_prowadzacego
  2. nazwa
  3. punkty\_ects
  4. id\_uzytkownika (wywołującego procedure)
- Korzystając z podanych parametrów dodajemy kurs.

#### ■ zapisz\_studenta

- Parametry:
  1. numer\_indeksu
  2. grupa
  3. id\_uzytkownika (wywołującego procedure)
- Korzystając z podanych parametrów zapisujemy studenta do danej grupy zajeciovej.

#### ■ zmien\_imie

- Parametry:
  1. imie
  2. id\_uzytkownika (wywołującego procedure)
- Korzystając z podanych parametrów zmieniamy imie bieżącego użytkownika.

#### ■ zmien\_kod\_pocztowy

- Parametry:
  1. kod\_pocztowy
  2. id\_uzytkownika (wywołującego procedure)
- Korzystając z podanych parametrów zmieniamy kod pocztowy bieżącego użytkownika.

#### ■ zmien\_kraj\_zamieszkania

- Parametry:
  1. kraj\_zamieszkania
  2. id\_uzytkownika (wywołującego procedure)
- Korzystając z podanych parametrów zmieniamy kraj zamieszkania bieżącego użytkownika.

#### ■ zmien\_miasto

- Parametry:
  1. miasto
  2. id\_uzytkownika (wywołującego procedure)
- Korzystając z podanych parametrów zmieniamy miasto bieżącego użytkownika.

- `zmien_ulice`
  - Parametry:
    1. `ulica`
    2. `id_uzytkownika` (wywołującego procedure)
  - Korzystając z podanych parametrów zmieniamy ulicę bieżącego użytkownika.
- `zmien_nazwisko`
  - Parametry:
    1. `nazwisko`
    2. `id_uzytkownika` (wywołującego procedure)
  - Korzystając z podanych parametrów zmieniamy nazwisko bieżącego użytkownika.
- `zmien_numer_domu`
  - Parametry:
    1. `numer_domu`
    2. `id_uzytkownika` (wywołującego procedure)
  - Korzystając z podanych parametrów zmieniamy numer domu bieżącego użytkownika.
- `zmien_numer_kontaktowy`
  - Parametry:
    1. `numer_kontaktowy`
    2. `id_uzytkownika` (wywołującego procedure)
  - Korzystając z podanych parametrów zmieniamy numer kontaktowy bieżącego użytkownika.
- `zmien_numer_lokalu`
  - Parametry:
    1. `numer_lokalu`
    2. `id_uzytkownika` (wywołującego procedure)
  - Korzystając z podanych parametrów zmieniamy numer lokalu bieżącego użytkownika.
- `zmien_plec`
  - Parametry:
    1. `plec`
    2. `id_uzytkownika` (wywołującego procedure)
  - Korzystając z podanych parametrów zmieniamy płeć bieżącego użytkownika.
- `wyswietl_dane_studenta`
  - Parametry:
    1. `id_uzytkownika`
  - Korzystając z podanych parametrów wyświetlamy dane studenta
- `wyswietl_dane_uzytkownika`
  - Parametry:
    1. `id_uzytkownika`
  - Korzystając z podanych parametrów wyświetlamy dane użytkownika

- `wyswietl_grupy_kursu_prowadzacego`
  - Parametry:
    1. `id_uzytkownika`
  - Korzystając z podanych parametrów wyświetlamy grupy kursu prowadzącego
- `wyswietl_oceny_studenta`
  - Parametry:
    1. `id_uzytkownika`
  - Korzystając z podanych parametrów wyświetlamy oceny studenta
- `wyswietl_plan_zajec_studenta`
  - Parametry:
    1. `id_uzytkownika`
  - Korzystając z podanych parametrów wyświetlamy plan zajęć studenta
- `wyswietl_prowadzone_kursy_prowadzacego`
  - Parametry:
    1. `id_uzytkownika`
  - Korzystając z podanych parametrów wyświetlamy prowadzone kursy przez prowadzącego
- `wyswietl_studentow_prowadzacego`
  - Parametry:
    1. `id_uzytkownika`
  - Korzystając z podanych parametrów wyświetlamy studentów jakich uczy dany prowadzący

### 3.6 Wyzwalacze

W naszej bazie danych nie będziemy wykorzystywać wyzwalaczy. Ich funkcjonalność pełnią procedury składowe.

### 3.7 Mechanizmy bezpieczeństwa na poziomie bazy danych

Rozgraniczenie uprawnień użytkowników odbywa się za pośrednictwem atrybutu "rola" w tabeli "Uzytkownicy". Poniżej znajdują się diagram z tabelami występującymi w projekcie określające uprawnienia odpowiednich grup użytkowników. Przyjmujemy następujące oznaczenia litera:

- A - uprawnienia administratora
- P - uprawnienia prowadzącego
- D - uprawnienia pracownika dziekanatu
- S - uprawnienia student

Dla każdego atrybutu są zdefiniowane prawa dostępu oznaczone jako:

- A - uprawnienia do odczytu (Read) i zapisu (Write)
- R - uprawnienia do odczytu (Read)
- W - uprawnienia do zapisu (Write)
- - - brak uprawnień do odczytu (Read) i zapisu (Write)

W tabelach takich jak Dane\_osobowe\_i\_kontaktowe oraz Dane\_adresowe użytkownicy mają dostęp do zapisu i odczytu tylko swoich danych.

Student					
	A	P	D	S	
numer_indeksu	-	-	-	-	
id_kierunku	-	-	-	-	
id_uzytkownika	-	-	-	-	
email	R	R	R	R	

Kierunek					
	A	P	D	S	
id_kierunku	-	-	-	-	
nazwa_kierunku	A	R	R	R	
semestr	R	R	R	R	
stopien	R	R	R	R	

Administrator					
	A	P	D	S	
id_administratora	-	-	-	-	
id_uzytkownika	-	-	-	-	
email	R	R	R	R	

Dane adresowe					
	A	P	D	S	
id_dane_adresowe	-	-	-	-	
kraj_zamieszkania	A	A	A	A	
miasto	A	A	A	A	
ulica	A	A	A	A	
numer_domu	A	A	A	A	
numer_lokalu	A	A	A	A	
kod_pocztowy	A	A	A	A	

Zapis					
	A	P	D	S	
id_zapisu	-	-	-	-	
numer_indeksu	-	-	-	-	
id_grupy	-	-	-	-	
ocena	A	A	R	R	
status_oceny	A	A	R	A	

Grupa_zajeciowa					
	A	P	D	S	
id_grupy	-	-	-	-	
id_kursu	-	-	-	-	
budynek	A	R	R	R	
sala	A	R	R	R	
dzien_tygodnia	A	R	R	R	
typ_zajec	A	R	R	R	
godzina_rozpoczecia	A	R	R	R	
godzina_zakonczenia	A	R	R	R	

Dane_osobowe_i_kontaktowe					
	A	P	D	S	
id_dane	-	-	-	-	
id_uzytkownika	-	-	-	-	
id_dane_adresowe	-	-	-	-	
imie	A	R	R	R	
pesel	A	R	R	R	
nazwisko	A	R	R	R	
data_urodzenia	A	R	R	R	
plec	A	R	R	R	
numer_kontaktowy	A	A	A	A	

Kurs					
	A	P	D	S	
id_kursu	-	-	-	-	
id_prowadzacego	-	-	-	-	
nazwa_kursu	A	R	R	R	
ects	A	R	R	R	

Prowadzacy					
	A	P	D	S	
id_prowadzacego	-	-	-	-	
id_uzytkownika	-	-	-	-	
email	R	R	R	R	

Pracownik_dziekanatu					
	A	P	D	S	
id_pracownika	-	-	-	-	
id_uzytkownika	-	-	-	-	
email	R	R	R	R	

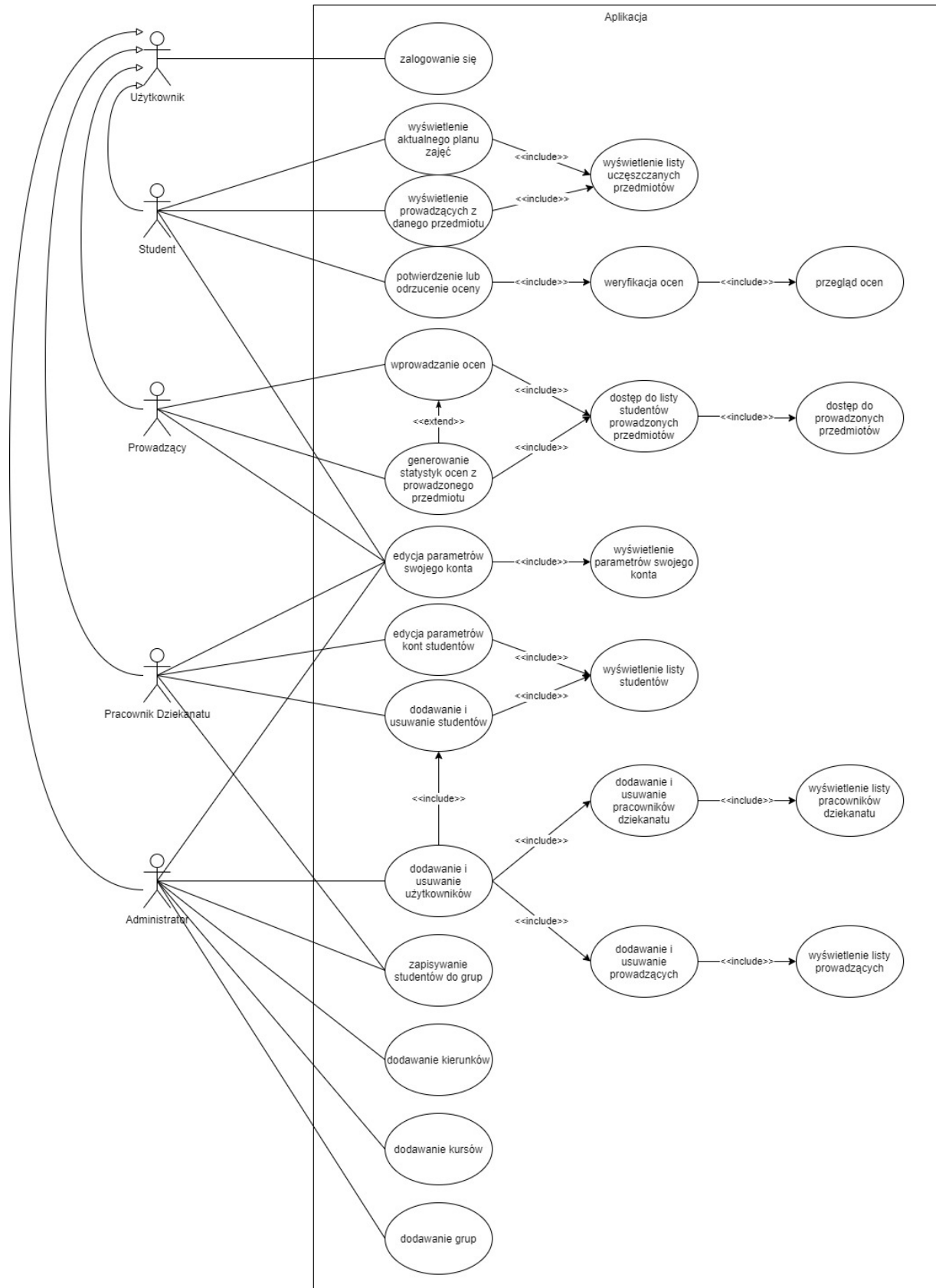
Uzytkownicy					
	A	P	D	S	
id_uzytkownika	-	-	-	-	
id_dane	-	-	-	-	
login	R	R	R	R	
haslo	-	-	-	-	
rola	R	R	R	R	

Rysunek 4: Diagram uprawnień



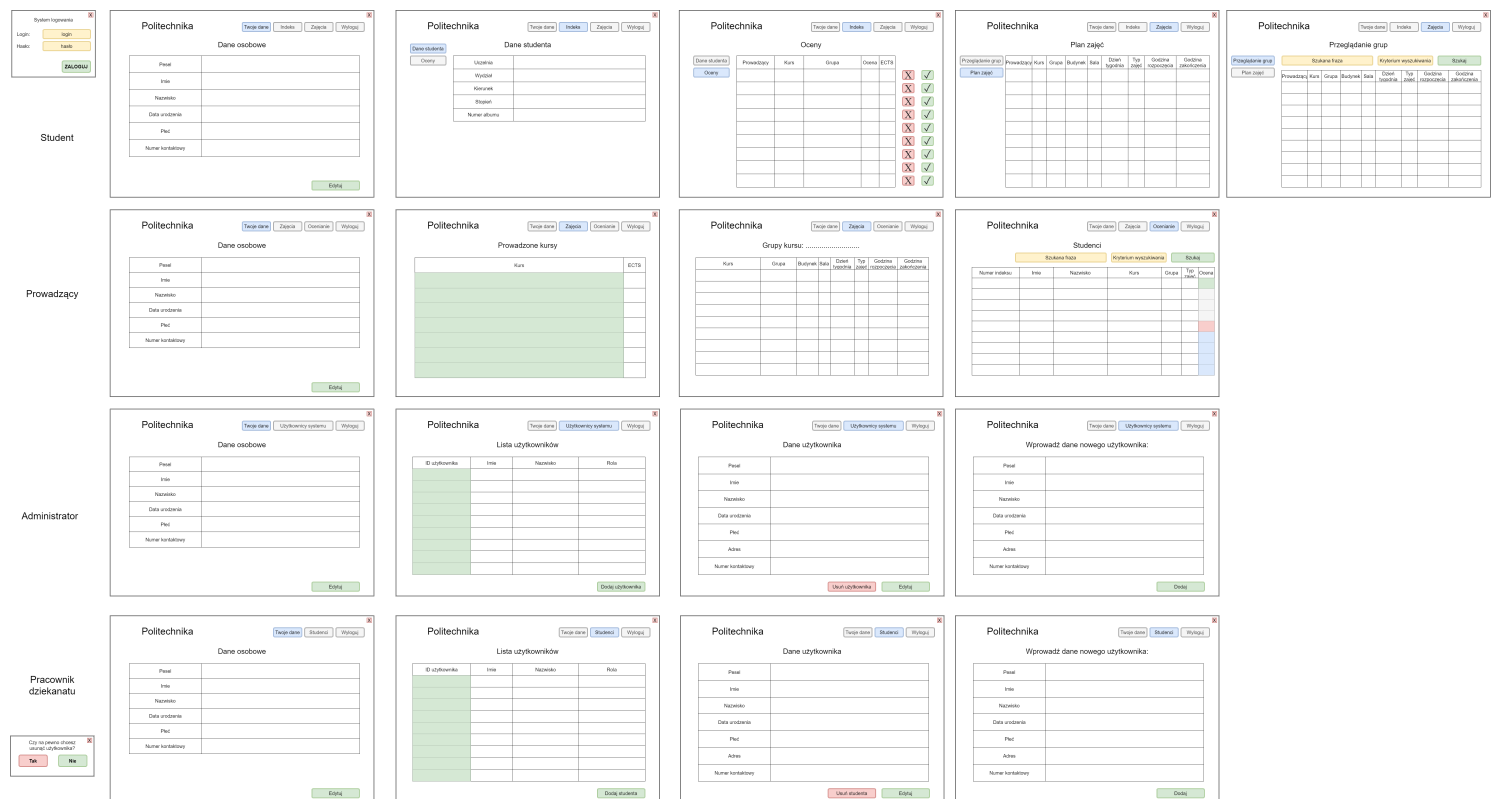
## 3.8 Projekt aplikacji użytkownika

### 3.8.1 Architektora aplikacji i diagramy projektowe



Rysunek 5: Diagram przypadków użycia

## 3.9 Interfejs graficzny



Rysunek 6: Projekt interfejsu graficznego

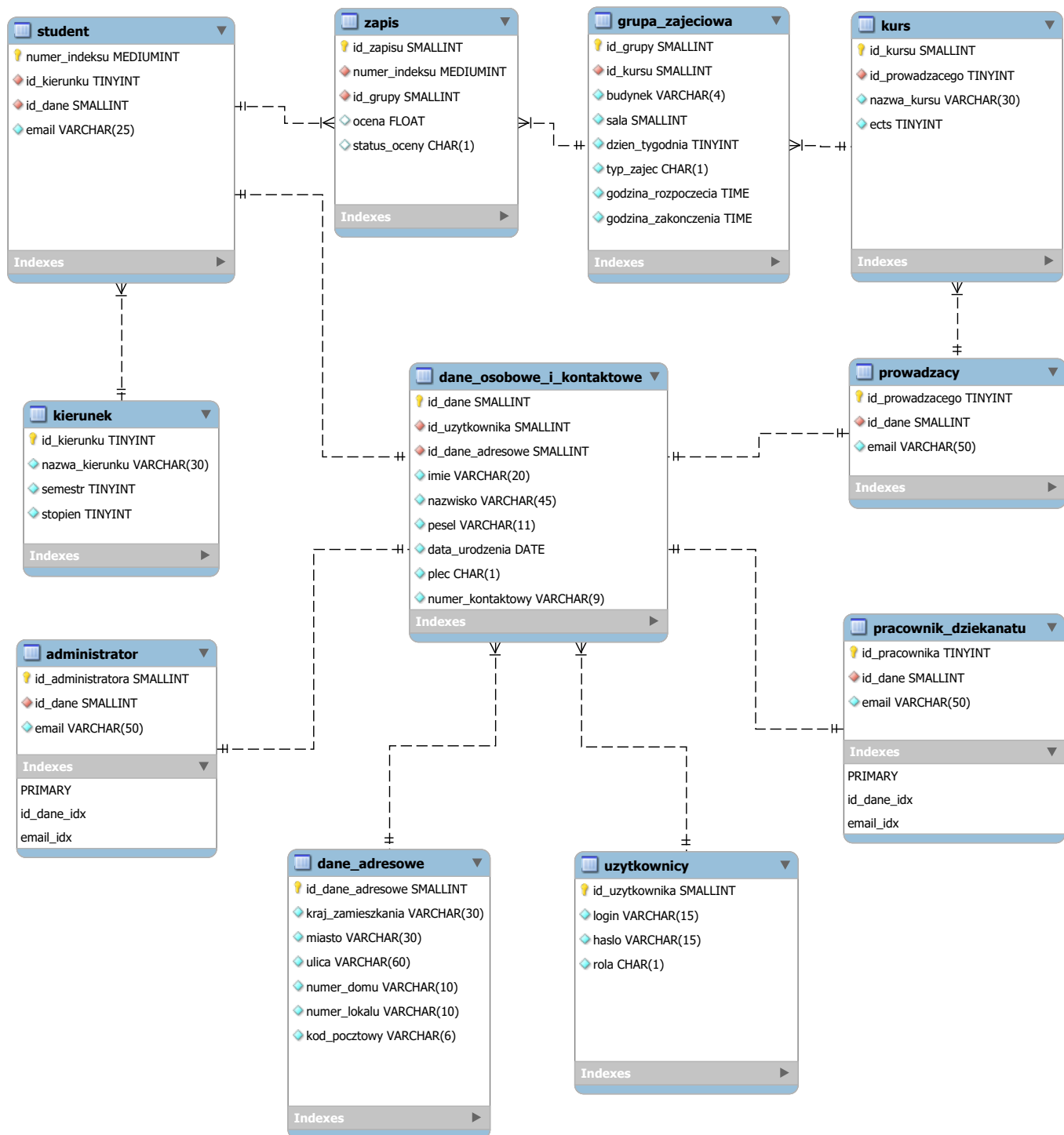
## 3.10 Metoda połączenia do bazy danych

Skorzystamy ze standardu ODBC aby łączyć się z bazą danych.

## 4 Tworzenie bazy danych

### 4.1 Tworzenie tabel, relacji oraz definicja ograniczeń w bazie danych

Baza danych została stworzona za pomocą narzędzia MySQL Workbench z silnikiem InnoDB. Tabele, relacje oraz definicje ograniczeń realizujemy przy pomocy narzędzia modelu w środowisku MySQL Workbench. Sekwencje realizujemy za pomocą autoinkrementacji. Wszystkie klucze główne inkrementują się od jedynek z wyjątkiem klucza głównego w tabeli student o nazwie numer\_indeksu, który inkrementuje się od 240000. Dodatkowo dla poszczególnych kolumn zostały zdefiniowane cechy takie jak zawartość kolumny przy inicjalizacji (NULL albo NOT NULL) oraz czy dana kolumna musi być unikatowa (UNIQUE). Poniżej znajduje się model naszej bazy danych.



## 4.2 Generacja kodu SQL

Na podstawie modelu został wygenerowany kod SQL, za pomocą polecenia "Forward Engineer".

```
-- MySQL Workbench Forward Engineering
```

```
SET @@OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @@OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @@OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,
NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
```

```
-- Schema elektroniczny_indeks
```

```
CREATE SCHEMA IF NOT EXISTS `elektroniczny_indeks` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci ;
USE `elektroniczny_indeks` ;
```

```
-- Table `elektroniczny_indeks`.`uzytkownicy`
```

```
CREATE TABLE IF NOT EXISTS `elektroniczny_indeks`.`uzytkownicy` (
  `id_uzytkownika` SMALLINT NOT NULL AUTO_INCREMENT,
  `login` VARCHAR(15) CHARACTER SET 'utf8' NOT NULL,
  `haslo` VARCHAR(15) CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NOT NULL,
  `rola` CHAR(1) CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NOT NULL,
  PRIMARY KEY (`id_uzytkownika`),
  INDEX `login_idx` (`login` ASC) INVISIBLE,
  INDEX `haslo_idx` (`haslo` ASC) INVISIBLE,
  INDEX `rola_idx` (`rola` ASC) VISIBLE,
  UNIQUE INDEX `login_UNIQUE` (`login` ASC) VISIBLE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

```
-- Table `elektroniczny_indeks`.`dane_adresowe`
```

```
CREATE TABLE IF NOT EXISTS `elektroniczny_indeks`.`dane_adresowe` (
  `id_dane_adresowe` SMALLINT NOT NULL AUTO_INCREMENT,
  `kraj_zamieszkania` VARCHAR(30) CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NOT NULL,
  `miasto` VARCHAR(30) CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NOT NULL,
  `ulica` VARCHAR(60) CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NOT NULL,
  `numer_domu` VARCHAR(10) CHARACTER SET 'utf8' NOT NULL,
  `numer_lokalu` VARCHAR(10) CHARACTER SET 'utf8' NOT NULL,
  `kod_pocztowy` VARCHAR(6) CHARACTER SET 'utf8' NOT NULL,
  PRIMARY KEY (`id_dane_adresowe`),
  INDEX `kraj_zamieszkania_idx` (`kraj_zamieszkania` ASC) INVISIBLE,
  INDEX `miasto_idx` (`miasto` ASC) INVISIBLE,
  INDEX `ulica` (`ulica` ASC) INVISIBLE,
  INDEX `numer_domu` (`numer_domu` ASC) INVISIBLE,
  INDEX `numer_lokalu` (`numer_lokalu` ASC) VISIBLE,
  INDEX `kod_pocztowy` (`kod_pocztowy` ASC) VISIBLE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

```
-- Table `elektroniczny_indeks`.`dane_osobowe_i_kontaktowe`
```

```

-----
CREATE TABLE IF NOT EXISTS `elektroniczny_indeks`.`dane_osobowe_i_kontaktowe` (
  `id_dane` SMALLINT NOT NULL AUTO_INCREMENT,
  `id_uzytkownika` SMALLINT NOT NULL,
  `id_dane_adresowe` SMALLINT NOT NULL,
  `imie` VARCHAR(20) CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NOT NULL,
  `nazwisko` VARCHAR(45) CHARACTER SET 'utf8' NOT NULL,
  `pesel` VARCHAR(11) CHARACTER SET 'utf8' NOT NULL,
  `data_urodzenia` DATE NOT NULL,
  `plec` CHAR(1) CHARACTER SET 'utf8' NOT NULL,
  `numer_kontaktowy` VARCHAR(9) CHARACTER SET 'utf8' NOT NULL,
  PRIMARY KEY (`id_dane`),
  INDEX `id_uzytkownika_idx` (`id_uzytkownika` ASC) VISIBLE,
  INDEX `id_dane_adresowe_idx` (`id_dane_adresowe` ASC) VISIBLE,
  INDEX `imie_idx` (`imie` ASC) VISIBLE,
  INDEX `nazwisko_idx` (`nazwisko` ASC) VISIBLE,
  INDEX `pesel_idx` (`pesel` ASC) INVISIBLE,
  INDEX `data_urodzenia_idx` (`data_urodzenia` ASC) INVISIBLE,
  INDEX `plec_idx` (`plec` ASC) INVISIBLE,
  INDEX `numer_kontaktowy_idx` (`numer_kontaktowy` ASC) INVISIBLE,
  UNIQUE INDEX `pesel_UNIQUE` (`pesel` ASC) VISIBLE,
  CONSTRAINT `id_uzytkownika`
    FOREIGN KEY (`id_uzytkownika`)
      REFERENCES `elektroniczny_indeks`.`uzytkownicy` (`id_uzytkownika`)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
  CONSTRAINT `id_dane_adresowe`
    FOREIGN KEY (`id_dane_adresowe`)
      REFERENCES `elektroniczny_indeks`.`dane_adresowe` (`id_dane_adresowe`)
        ON DELETE CASCADE
        ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

-----
-- Table `elektroniczny_indeks`.`administrator`
-----

```

```

CREATE TABLE IF NOT EXISTS `elektroniczny_indeks`.`administrator` (
  `id_administradora` SMALLINT NOT NULL AUTO_INCREMENT,
  `id_dane` SMALLINT NOT NULL,
  `email` VARCHAR(50) CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NOT NULL,
  PRIMARY KEY (`id_administradora`),
  INDEX `id_dane_idx` (`id_dane` ASC) INVISIBLE,
  INDEX `email_idx` (`email` ASC) INVISIBLE,
  CONSTRAINT `fk_uzytkownik_administrator`
    FOREIGN KEY (`id_dane`)
      REFERENCES `elektroniczny_indeks`.`dane_osobowe_i_kontaktowe` (`id_uzytkownika`)
        ON DELETE CASCADE
        ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

-----
-- Table `elektroniczny_indeks`.`prowadzacy`
-----

```

```

CREATE TABLE IF NOT EXISTS `elektroniczny_indeks`.`prowadzacy` (
  `id_prowadzacego` TINYINT NOT NULL AUTO_INCREMENT,
  `id_dane` SMALLINT NOT NULL,
  `email` VARCHAR(50) CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NOT NULL,
  PRIMARY KEY (`id_prowadzacego`),
  INDEX `id_dane_idx` (`id_dane` ASC) INVISIBLE,
  INDEX `email_idx` (`email` ASC) INVISIBLE,
  CONSTRAINT `fk_uzytkownik_prowadzacy`
    FOREIGN KEY (`id_dane`)
      REFERENCES `elektroniczny_indeks`.`dane_osobowe_i_kontaktowe` (`id_uzytkownika`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

-----
-- Table `elektroniczny_indeks`.`kurs`
-----

```

```

CREATE TABLE IF NOT EXISTS `elektroniczny_indeks`.`kurs` (
  `id_kursu` SMALLINT NOT NULL AUTO_INCREMENT,
  `id_prowadzacego` TINYINT NOT NULL,
  `nazwa_kursu` VARCHAR(30) CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NOT NULL,
  `ects` TINYINT NOT NULL,
  PRIMARY KEY (`id_kursu`),
  INDEX `id_prowadzacego_idx` (`id_prowadzacego` ASC) VISIBLE,
  INDEX `nazwa_kursu_idx` (`nazwa_kursu` ASC) INVISIBLE,
  INDEX `ects_idx` (`ects` ASC) VISIBLE,
  UNIQUE INDEX `nazwa_kursu_UNIQUE` (`nazwa_kursu` ASC) VISIBLE,
  CONSTRAINT `id_prowadzacego`
    FOREIGN KEY (`id_prowadzacego`)
      REFERENCES `elektroniczny_indeks`.`prowadzacy` (`id_prowadzacego`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

-----
-- Table `elektroniczny_indeks`.`grupa_zajeciowa`
-----

```

```

CREATE TABLE IF NOT EXISTS `elektroniczny_indeks`.`grupa_zajeciowa` (
  `id_grupy` SMALLINT NOT NULL AUTO_INCREMENT,
  `id_kursu` SMALLINT NOT NULL,
  `budynek` VARCHAR(4) CHARACTER SET 'utf8' NOT NULL,
  `sala` SMALLINT NOT NULL,
  `dzien_tygodnia` TINYINT NOT NULL,
  `typ_zajec` CHAR(1) CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NOT NULL,
  `godzina_roz poczeczia` TIME NOT NULL,
  `godzina_zakonczenia` TIME NOT NULL,
  PRIMARY KEY (`id_grupy`),
  INDEX `id_kursu_idx` (`id_kursu` ASC) VISIBLE,
  INDEX `budynek_idx` (`budynek` ASC) INVISIBLE,
  INDEX `sala_idx` (`sala` ASC) INVISIBLE,
  INDEX `dzien_tygodnia_idx` (`dzien_tygodnia` ASC) INVISIBLE,
  INDEX `typ_zajec_idx` (`typ_zajec` ASC) INVISIBLE,
  INDEX `godzina_roz poczeczia_idx` (`godzina_roz poczeczia` ASC) INVISIBLE,

```

```

INDEX `godzina_zakonczenia_idx` (`godzina_zakonczenia` ASC) VISIBLE,
CONSTRAINT `id_kursu`
  FOREIGN KEY (`id_kursu`)
  REFERENCES `elektroniczny_indeks`.`kurs` (`id_kursu`)
  ON DELETE CASCADE
  ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

-----
-- Table `elektroniczny_indeks`.`kierunek`
-----

```

```

CREATE TABLE IF NOT EXISTS `elektroniczny_indeks`.`kierunek` (
  `id_kierunku` TINYINT NOT NULL AUTO_INCREMENT,
  `nazwa_kierunku` VARCHAR(30) CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NOT NULL,
  `semestr` TINYINT NOT NULL,
  `stopien` TINYINT NOT NULL,
  PRIMARY KEY (`id_kierunku`),
  INDEX `nazwa_kierunku_idx` (`nazwa_kierunku` ASC) INVISIBLE,
  INDEX `semestr_idx` (`semestr` ASC) INVISIBLE,
  INDEX `stopien_idx` (`stopien` ASC) VISIBLE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

-----
-- Table `elektroniczny_indeks`.`pracownik_dziekanatu`
-----

```

```

CREATE TABLE IF NOT EXISTS `elektroniczny_indeks`.`pracownik_dziekanatu` (
  `id_pracownika` TINYINT NOT NULL AUTO_INCREMENT,
  `id_dane` SMALLINT NOT NULL,
  `email` VARCHAR(50) CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NOT NULL,
  PRIMARY KEY (`id_pracownika`),
  INDEX `id_dane_idx` (`id_dane` ASC) VISIBLE,
  INDEX `email_idx` (`email` ASC) INVISIBLE,
  CONSTRAINT `fk_uzytkownik_dziekanat`
    FOREIGN KEY (`id_dane`)
    REFERENCES `elektroniczny_indeks`.`dane_osobowe_i_kontaktowe` (`id_uzytkownika`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

-----
-- Table `elektroniczny_indeks`.`student`
-----

```

```

CREATE TABLE IF NOT EXISTS `elektroniczny_indeks`.`student` (
  `numer_indeksu` MEDIUMINT NOT NULL AUTO_INCREMENT,
  `id_kierunku` TINYINT NOT NULL,
  `id_dane` SMALLINT NOT NULL,
  `email` VARCHAR(25) CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NOT NULL,
  PRIMARY KEY (`numer_indeksu`),
  INDEX `id_kierunku_idx` (`id_kierunku` ASC) VISIBLE,
  INDEX `email_idx` (`email` ASC) INVISIBLE,

```

```

INDEX `fk_uzytkownik_student_idx` (`id_dane` ASC) VISIBLE,
CONSTRAINT `fk_uzytkownik_student`
  FOREIGN KEY (`id_dane`)
  REFERENCES `elektroniczny_indeks`.`dane_osobowe_i_kontaktowe` (`id_dane`)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
CONSTRAINT `id_kierunku`
  FOREIGN KEY (`id_kierunku`)
  REFERENCES `elektroniczny_indeks`.`kierunek` (`id_kierunku`)
  ON DELETE CASCADE
  ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT=240000
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

-----
-- Table `elektroniczny_indeks`.`zapis`
-----

```

```

CREATE TABLE IF NOT EXISTS `elektroniczny_indeks`.`zapis` (
  `id_zapisu` SMALLINT NOT NULL AUTO_INCREMENT,
  `numer_indeksu` MEDIUMINT NOT NULL,
  `id_grupy` SMALLINT NOT NULL,
  `ocena` FLOAT NULL DEFAULT NULL,
  `status_oceny` CHAR(1) CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NULL,
  PRIMARY KEY (`id_zapisu`),
  INDEX `numer_indeksu_idx` (`numer_indeksu` ASC) VISIBLE,
  INDEX `id_grupy_idx` (`id_grupy` ASC) VISIBLE,
  INDEX `ocena_idx` (`ocena` ASC) INVISIBLE,
  INDEX `status_oceny_idx` (`status_oceny` ASC) INVISIBLE,
  CONSTRAINT `numer_indeksu`
    FOREIGN KEY (`numer_indeksu`)
    REFERENCES `elektroniczny_indeks`.`student` (`numer_indeksu`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `id_grupy`
    FOREIGN KEY (`id_grupy`)
    REFERENCES `elektroniczny_indeks`.`grupa_zajeciowa` (`id_grupy`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```



## 4.3 Implementacja widoków

W następnym kroku zostały stworzone widoki za pomocą poleceń SQL. Poniżej znajdują się zaimplementowane widoki.

### 4.3.1 Widok - "dane\_studenta\_view"

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `dane_studenta_view` AS
  SELECT
    `dane_osobowe_i_kontaktowe`.`id_uzytkownika` AS `id_uzytkownika`,
    `kierunek`.`nazwa_kierunku` AS `nazwa_kierunku`,
    `kierunek`.`stopien` AS `stopien`,
    `kierunek`.`semestr` AS `semestr`,
    `student`.`numer_indeksu` AS `numer_indeksu`
  FROM
    ((`kierunek`
  JOIN `student`)
  JOIN `dane_osobowe_i_kontaktowe`)
  WHERE
    ((`kierunek`.`id_kierunku` = `student`.`id_kierunku`)
    AND (`dane_osobowe_i_kontaktowe`.`id_dane` = `student`.`id_dane`))
```

### 4.3.2 Widok - "dane\_uzytkownika\_view"

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `dane_uzytkownika_view` AS
  SELECT
    `dane_osobowe_i_kontaktowe`.`id_uzytkownika` AS `id_uzytkownika`,
    `dane_osobowe_i_kontaktowe`.`imie` AS `imie`,
    `dane_osobowe_i_kontaktowe`.`nazwisko` AS `nazwisko`,
    `dane_osobowe_i_kontaktowe`.`pesel` AS `pesel`,
    `dane_osobowe_i_kontaktowe`.`data_urodzenia` AS `data_urodzenia`,
    `dane_osobowe_i_kontaktowe`.`plec` AS `plec`,
    `dane_osobowe_i_kontaktowe`.`numer_kontaktowy` AS `numer_kontaktowy`,
    `dane_adresowe`.`kraj_zamieszkania` AS `kraj_zamieszkania`,
    `dane_adresowe`.`miasto` AS `miasto`,
    `dane_adresowe`.`ulica` AS `ulica`,
    `dane_adresowe`.`numer_domu` AS `numer_domu`,
    `dane_adresowe`.`numer_lokalu` AS `numer_lokalu`
  FROM
    (`dane_osobowe_i_kontaktowe`
  JOIN `dane_adresowe`)
  WHERE
    (`dane_adresowe`.`id_dane_adresowe` = `dane_osobowe_i_kontaktowe`.`id_dane_adresowe`)
```

#### 4.3.3 Widok - "prowadzone\_kursy\_prowadzacego"

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `prowadzone_kursy_prowadzacego_view` AS
  SELECT
    `dane_osobowe_i_kontaktowe`.`id_uzytkownika` AS `id_uzytkownika`,
    `kurs`.`nazwa_kursu` AS `nazwa_kursu`,
    `kurs`.`ects` AS `ects`
  FROM
    ((`dane_osobowe_i_kontaktowe`
  JOIN `prowadzacy`)
  JOIN `kurs`)
  WHERE
    ((`dane_osobowe_i_kontaktowe`.`id_dane` = `prowadzacy`.`id_dane`)
    AND (`prowadzacy`.`id_prowadzacego` = `kurs`.`id_prowadzacego`))
```

#### 4.3.4 Widok - "grupy\_kursu\_prowadzacego"

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `grupy_kursu_prowadzacego_view` AS
  SELECT
    `dane_osobowe_i_kontaktowe`.`id_uzytkownika` AS `id_uzytkownika`,
    `grupa_zajeciowa`.`id_grupy` AS `id_grupy`,
    `grupa_zajeciowa`.`budynek` AS `budynek`,
    `grupa_zajeciowa`.`sala` AS `sala`,
    `grupa_zajeciowa`.`dzien_tygodnia` AS `dzien_tygodnia`,
    `grupa_zajeciowa`.`typ_zajec` AS `typ_zajec`,
    `grupa_zajeciowa`.`godzina_rozpoczecia` AS `godzina_rozpoczecia`,
    `grupa_zajeciowa`.`godzina_zakonczenia` AS `godzina_zakonczenia`
  FROM
    (((`grupa_zajeciowa`
  JOIN `kurs`)
  JOIN `prowadzacy`)
  JOIN `dane_osobowe_i_kontaktowe`)
  WHERE
    ((`dane_osobowe_i_kontaktowe`.`id_dane` = `prowadzacy`.`id_dane`)
    AND (`prowadzacy`.`id_prowadzacego` = `kurs`.`id_prowadzacego`)
    AND (`kurs`.`id_kursu` = `grupa_zajeciowa`.`id_kursu`))
```

#### 4.3.5 Widok - "studenci\_prowadzacego"

```
CREATE
ALGORITHM = UNDEFINED
DEFINER = `root`@`localhost`
SQL SECURITY DEFINER
VIEW `studenci_prowadzacego_view` AS
SELECT
    `dane_osobowe_i_kontaktowe`.`id_uzytkownika` AS `id_uzytkownika`,
    `student`.`numer_indeksu` AS `numer_indeksu`,
    `student`.`id_dane` AS `id_dane`,
    `kurs`.`nazwa_kursu` AS `nazwa_kursu`,
    `grupa_zajeciowa`.`id_grupy` AS `id_grupy`,
    `grupa_zajeciowa`.`typ_zajec` AS `typ_zajec`,
    `zapis`.`ocena` AS `ocena`,
    `zapis`.`status_oceny` AS `status_oceny`
FROM
    (((((`dane_osobowe_i_kontaktowe`
    JOIN `prowadzacy`)
    JOIN `kurs`)
    JOIN `grupa_zajeciowa`)
    JOIN `zapis`)
    JOIN `student`)
WHERE
    ((`dane_osobowe_i_kontaktowe`.`id_dane` = `prowadzacy`.`id_dane`)
    AND (`prowadzacy`.`id_prowadzacego` = `kurs`.`id_prowadzacego`)
    AND (`kurs`.`id_kursu` = `grupa_zajeciowa`.`id_kursu`)
    AND (`zapis`.`id_grupy` = `grupa_zajeciowa`.`id_grupy`))
```

#### 4.3.6 Widok - "oceny\_studenta"

```
CREATE
ALGORITHM = UNDEFINED
DEFINER = `root`@`localhost`
SQL SECURITY DEFINER
VIEW `oceny_studenta_view` AS
SELECT
    `student`.`id_dane` AS `id_dane`,
    `dane_osobowe_i_kontaktowe`.`imie` AS `imie`,
    `dane_osobowe_i_kontaktowe`.`nazwisko` AS `nazwisko`,
    `kurs`.`nazwa_kursu` AS `nazwa_kursu`,
    `kurs`.`ects` AS `ects`,
    `grupa_zajeciowa`.`id_grupy` AS `grupa_zajeciowa`,
    `zapis`.`ocena` AS `ocena`,
    `zapis`.`status_oceny` AS `status_oceny`
FROM
    (((((`dane_osobowe_i_kontaktowe`
    JOIN `prowadzacy`)
    JOIN `kurs`)
    JOIN `grupa_zajeciowa`)
    JOIN `zapis`)
    JOIN `student`)
WHERE
    ((`student`.`numer_indeksu` = `zapis`.`numer_indeksu`)
    AND (`grupa_zajeciowa`.`id_grupy` = `zapis`.`id_grupy`)
    AND (`kurs`.`id_kursu` = `grupa_zajeciowa`.`id_kursu`)
    AND (`prowadzacy`.`id_prowadzacego` = `kurs`.`id_prowadzacego`)
    AND (`dane_osobowe_i_kontaktowe`.`id_dane` = `prowadzacy`.`id_dane`))
```

#### 4.3.7 Widok - "plan\_zajec\_studenta"

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `plan_zajec_studenta_view` AS
  SELECT
    `student`.`id_dane` AS `id_dane`,
    `dane_osobowe_i_kontaktowe`.`imie` AS `imie`,
    `dane_osobowe_i_kontaktowe`.`nazwisko` AS `nazwisko`,
    `kurs`.`nazwa_kursu` AS `nazwa_kursu`,
    `grupa_zajeciowa`.`id_grupy` AS `id_grupy`,
    `grupa_zajeciowa`.`budynek` AS `budynek`,
    `grupa_zajeciowa`.`sala` AS `sala`,
    `grupa_zajeciowa`.`dzien_tygodnia` AS `dzien_tygodnia`,
    `grupa_zajeciowa`.`typ_zajec` AS `typ_zajec`,
    `grupa_zajeciowa`.`godzina_rozpoczecia` AS `godzina_rozpoczecia`,
    `grupa_zajeciowa`.`godzina_zakonczenia` AS `godzina_zakonczenia`
  FROM
    ((((`dane_osobowe_i_kontaktowe`
    JOIN `prowadzacy`)
    JOIN `kurs`)
    JOIN `grupa_zajeciowa`)
    JOIN `zapis`)
    JOIN `student`)
  WHERE
    ((`dane_osobowe_i_kontaktowe`.`id_dane` = `prowadzacy`.`id_dane`)
    AND (`prowadzacy`.`id_prowadzacego` = `kurs`.`id_prowadzacego`)
    AND (`kurs`.`id_kursu` = `grupa_zajeciowa`.`id_kursu`)
    AND (`zapis`.`id_grupy` = `grupa_zajeciowa`.`id_grupy`)
    AND (`student`.`numer_indeksu` = `zapis`.`numer_indeksu`))
```

#### 4.3.8 Widok - "lista\_uzytkownikow"

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `lista_uzytkownikow_view` AS
  SELECT
    `uzytkownicy`.`id_uzytkownika` AS `id_uzytkownika`,
    `dane_osobowe_i_kontaktowe`.`imie` AS `imie`,
    `dane_osobowe_i_kontaktowe`.`nazwisko` AS `nazwisko`,
    `uzytkownicy`.`rola` AS `rola`
  FROM
    (`uzytkownicy`
    JOIN `dane_osobowe_i_kontaktowe`)
  WHERE
    (`uzytkownicy`.`id_uzytkownika` = `dane_osobowe_i_kontaktowe`.`id_uzytkownika`)
```

### 4.3.9 Widok - "przeglądanie\_grup"

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `przeglądanie_grup_view` AS
  SELECT
    `dane_osobowe_i_kontaktowe`.`imie` AS `imie`,
    `dane_osobowe_i_kontaktowe`.`nazwisko` AS `nazwisko`,
    `kurs`.`nazwa_kursu` AS `nazwa_kursu`,
    `grupa_zajeciowa`.`id_grupy` AS `id_grupy`,
    `grupa_zajeciowa`.`budynek` AS `budynek`,
    `grupa_zajeciowa`.`sala` AS `sala`,
    `grupa_zajeciowa`.`dzien_tygodnia` AS `dzien_tygodnia`,
    `grupa_zajeciowa`.`typ_zajec` AS `typ_zajec`,
    `grupa_zajeciowa`.`godzina_rozpoczecia` AS `godzina_rozpoczecia`,
    `grupa_zajeciowa`.`godzina_zakonczenia` AS `godzina_zakonczenia`
  FROM
    ((((`dane_osobowe_i_kontaktowe`
    JOIN `prowadzacy`)
    JOIN `kurs`)
    JOIN `grupa_zajeciowa`)
    JOIN `zapis`)
  WHERE
    ((`dane_osobowe_i_kontaktowe`.`id_dane` = `prowadzacy`.`id_dane`)
    AND (`prowadzacy`.`id_prowadzacego` = `kurs`.`id_prowadzacego`)
    AND (`kurs`.`id_kursu` = `grupa_zajeciowa`.`id_kursu`)
    AND (`zapis`.`id_grupy` = `grupa_zajeciowa`.`id_grupy`))
```

## 4.4 Implementacja procedur składowanych

Następnie zostały stworzone procedury składowe za pomocą poleceń SQL. Poniżej znajdują się zaimplementowane procedury składowe.

### 4.4.1 Procedura składowana - "wyswietl\_dane\_studenta"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `wyswietl_dane_studenta`(id_uzytkownika_n SMALLINT)
BEGIN
  SELECT nazwa_kierunku, stopien, semestr, numer_indeksu
  FROM dane_studenta_view
  WHERE id_uzytkownika = id_uzytkownika_n;
END
```

### 4.4.2 Procedura składowana - "wyswietl\_dane\_uzytkownika"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `wyswietl_dane_uzytkownika`(id_uzytkownika_n SMALLINT)
BEGIN
  SELECT imie, nazwisko, pesel, data_urodzenia, plec, numer_kontaktowy, kraj_zamieszkania, miasto,
  ulica, numer_domu, numer_lokalu
  FROM dane_uzytkownika_view
  WHERE id_uzytkownika = id_uzytkownika_n;
END
```

#### 4.4.3 Procedura składowana - "wyswietl\_grupy\_kursu\_prowadzacego"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `wyswietl_grupy_kursu_prowadzacego`(id_uzytkownika_n SMALLINT)
BEGIN
    SELECT id_grupy, budynek, sala, dzien_tygodnia, typ_zajec, godzina_rozpoczecia, godzina_zakonczenia
    FROM grupy_kursu_prowadzacego_view
    WHERE id_uzytkownika = id_uzytkownika_n;
END
```

#### 4.4.4 Procedura składowana - "wyswietl\_oceny\_studenta"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `wyswietl_oceny_studenta`(id_uzytkownika_n SMALLINT)
BEGIN
    SELECT imie, nazwisko, nazwa_kursu, ects, grupa_zajeciowa, ocena, status_oceny
    FROM oceny_studenta_view
    WHERE id_dane = (SELECT id_dane FROM dane_osobowe_i_kontaktowe WHERE
    id_uzytkownika = id_uzytkownika_n);
END
```

#### 4.4.5 Procedura składowana - "wyswietl\_plan\_zajec\_studenta"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `wyswietl_plan_zajec_studenta`(id_uzytkownika_n SMALLINT)
BEGIN
    SELECT imie, nazwisko, nazwa_kursu, id_grupy, budynek, sala, dzien_tygodnia, typ_zajec,
    godzina_rozpoczecia, godzina_zakonczenia
    FROM plan_zajec_studenta_view
    WHERE id_dane = (SELECT id_dane FROM dane_osobowe_i_kontaktowe WHERE
    id_uzytkownika = id_uzytkownika_n);
END
```

#### 4.4.6 Procedura składowana - "wyswietl\_prowadzone\_kursy\_prowadzacego"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `wyswietl_prowadzone_kursy_prowadzacego`(id_uzytkownika_n SMALLINT)
BEGIN
    SELECT nazwa_kursu, ects
    FROM prowadzone_kursy_prowadzacego_view
    WHERE id_uzytkownika = id_uzytkownika_n;
END
```

#### 4.4.7 Procedura składowana - "wyswietl\_studentow\_prowadzacego"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `wyswietl_studentow_prowadzacego`(id_uzytkownika_n SMALLINT)
BEGIN
    SELECT numer_indeksu, nazwa_kursu, id_grupy, typ_zajec, ocena, status_oceny
    FROM studenci_prowadzacego_view
    WHERE id_uzytkownika = id_uzytkownika_n;
END
```

#### 4.4.8 Procedura składowana - "dodaj\_grupe\_zajeciowa"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `dodaj_grupe_zajeciowa`(kurs smallint, budynek_n varchar(4),
sala_n smallint, dzien_tygodnia_n tinyint, typ_zajec_n char(1), godzina_roz poczenia_n time,
godzina_zakonczenia_n time, biezacy_uzytkownik smallint)
BEGIN
    declare rola_uzytkownika char(1);

    set rola_uzytkownika = (select rola from `uzytkownicy` where id_uzytkownika = biezacy_uzytkownik);

    if (rola_uzytkownika='a') then
        insert into `grupa_zajeciowa`(`id_kursu`, `budynek`, `sala`, `dzien_tygodnia`, `typ_zajec`,
`godzina_roz poczenia`, `godzina_zakonczenia`) values
        (kurs, budynek_n, sala_n, dzien_tygodnia_n, typ_zajec_n, godzina_roz poczenia_n,
        godzina_zakonczenia_n);
    end if;
END
```

#### 4.4.9 Procedura składowana - "dodaj\_kierunek"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `dodaj_kierunek`(nazwa varchar(30), semestr_n tinyint,
stopien_n tinyint, biezacy_uzytkownik smallint)
BEGIN
    declare rola_uzytkownika char(1);

    set rola_uzytkownika = (select rola from `uzytkownicy` where id_uzytkownika = biezacy_uzytkownik);

    if (rola_uzytkownika='a') then
        insert into `kierunek` (`nazwa_kierunku`, `semestr`, `stopien`) values
        (nazwa, semestr_n, stopien_n);
    end if;
END
```

#### 4.4.10 Procedura składowana - "dodaj\_kurs"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `dodaj_kurs`(prowadzacy tinyint, nazwa varchar(30),
punkty_ects tinyint, biezacy_uzytkownik smallint)
BEGIN
    declare rola_uzytkownika char(1);

    set rola_uzytkownika = (select rola from `uzytkownicy` where id_uzytkownika = biezacy_uzytkownik);

    if (rola_uzytkownika='a') then
        insert into `kurs`(`id_prowadzacego`, `nazwa_kursu`, `ects`) values
        (prowadzacy, nazwa, punkty_ects);
    end if;
END
```

#### 4.4.11 Procedura składowana - "dodaj\_uzytkownika"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `dodaj_uzytkownika`(pesel_n varchar(11), imie_n varchar(20),
nazwisko_n varchar(45), data_urodzenia_n date, plec_n char(1), numer_kontaktowy_n varchar(9),
kraj_zamieszkania_n varchar(30), miasto_n varchar(30), ulica_n varchar(30), numer_domu_n varchar(10),
numer_lokalu_n varchar(10), kod_pocztowy_n varchar(5), login_n varchar(15), haslo_n varchar(15),
rola_n char(1), kierunek varchar(30), ktory_semestr tinyint, ktory_stopien tinyint,
biezacy_uzytkownik smallint)
BEGIN
    declare rola_uzytkownika char(1);
    declare czy_istnieje bool;
    declare identyfikator mediumint;
```

```

declare identyfikator_danych mediumint;
declare imie_email varchar(20);
declare nazwisko_email varchar(45);

set rola_uzytkownika = (select rola from `uzytkownicy` where id_uzytkownika = biezacy_uzytkownik);
set czy_istnieje = true;

if (((rola_uzytkownika='a') or (rola_uzytkownika='d' and rola_n='s')) and
((select not exists(select id_dane from `dane_osobowe_i_kontaktowe` where
pesel=pesel_n)) and (select not exists(select id_uzytkownika from `uzytkownicy` where
login=login_n)))) then insert into `uzytkownicy`(`login`,`haslo`,`rola`) values
(login_n, haslo_n, rola_n);
    if (select not exists(select id_dane_adresowe from `dane_adresowe` where
kraj_zamieszkania=kraj_zamieszkania_n and miasto=miasto_n and ulica=ulica_n and
numer_domu=numer_domu_n and numer_lokalu=numer_lokalu_n and kod_pocztowy=kod_pocztowy_n))
then insert into `dane_adresowe`(`kraj_zamieszkania`,`miasto`,`ulica`,`numer_domu`,
`numer_lokalu`,`kod_pocztowy`) values (kraj_zamieszkania_n, miasto_n, ulica_n,
numer_domu_n, numer_lokalu_n, kod_pocztowy_n);
        end if;
insert into `dane_osobowe_i_kontaktowe`(`id_uzytkownika`,`id_dane_adresowe`,`imie`,`pesel`,
`nazwisko`,`data_urodzenia`,`plec`,`numer_kontaktowy`) values
((select id_uzytkownika from `uzytkownicy` where (login=login_n and haslo=haslo_n and
rola=rola_n)), (select id_dane_adresowe from `dane_adresowe` where
kraj_zamieszkania=kraj_zamieszkania_n and miasto=miasto_n and ulica=ulica_n and
numer_domu=numer_domu_n and numer_lokalu=numer_lokalu_n and kod_pocztowy=kod_pocztowy_n),
imie_n, pesel_n, nazwisko_n, data_urodzenia_n, plec_n, numer_kontaktowy_n);
if rola_n='s' then
    insert into `student`(`id_kierunku`,`id_dane`,`email`) values
    ((select id_kierunku from `kierunek` where (nazwa_kierunku=kierunek and
semestr=ktory_semestr and stopien=ktory_stopien)),
    (select id_dane from `dane_osobowe_i_kontaktowe` where
    (id_uzytkownika=(select id_uzytkownika from `uzytkownicy` where (login=login_n and
haslo=haslo_n and rola=rola_n)) and
    id_dane_adresowe=(select id_dane_adresowe from `dane_adresowe` where
kraj_zamieszkania=kraj_zamieszkania_n and miasto=miasto_n and ulica=ulica_n and
numer_domu=numer_domu_n and numer_lokalu=numer_lokalu_n and
kod_pocztowy=kod_pocztowy_n) and imie=imie_n and pesel=pesel_n and
nazwisko=nazwisko_n and data_urodzenia=data_urodzenia_n and
plec=plec_n and numer_kontaktowy=numer_kontaktowy_n)),
    '@student.po.edu.pl');
set identyfikator=(select numer_indeksu from student where
(id_kierunku=(select id_kierunku from `kierunek` where (nazwa_kierunku=kierunek and
semestr=ktory_semestr and stopien=ktory_stopien))and
id_dane=(select id_dane from `dane_osobowe_i_kontaktowe` where
(id_uzytkownika=(select id_uzytkownika from `uzytkownicy` where
(login=login_n and haslo=haslo_n and rola=rola_n)) and
    id_dane_adresowe=(select id_dane_adresowe from `dane_adresowe` where
kraj_zamieszkania=kraj_zamieszkania_n and miasto=miasto_n and ulica=ulica_n and
numer_domu=numer_domu_n and numer_lokalu=numer_lokalu_n and
kod_pocztowy=kod_pocztowy_n) and
imie=imie_n and pesel=pesel_n and nazwisko=nazwisko_n and
data_urodzenia=data_urodzenia_n and
plec=plec_n and numer_kontaktowy=numer_kontaktowy_n)))));
update `student`
set email = concat((select cast(identyfikator as char(6))), '@student.po.edu.pl')
where numer_indeksu = identyfikator;
elseif rola_n='p' then
    insert into `prowadzacy`(`id_dane`,`email`) values
    ((select id_dane from `dane_osobowe_i_kontaktowe` where

```



```

(id_uzytkownika=(select id_uzytkownika from `uzytkownicy` where
(login=login_n and haslo=haslo_n and rola=rola_n)) and
id_dane_adresowe=(select id_dane_adresowe from `dane_adresowe` where
kraj_zamieszkania=kraj_zamieszkania_n and miasto=miasto_n and ulica=ulica_n and
numer_domu=numer_domu_n and numer_lokalu=numer_lokalu_n and
kod_pocztowy=kod_pocztowy_n) and imie=imie_n and pesel=pesel_n and
nazwisko=nazwisko_n and data_urodzenia=data_urodzenia_n and
plec=plec_n and numer_kontaktowy=numer_kontaktowy_n)),
'@po.edu.pl');
set identyfikator_danych = (select id_dane from `dane_osobowe_i_kontaktowe` where
(id_uzytkownika=(select id_uzytkownika from `uzytkownicy` where (login=login_n and
haslo=haslo_n and rola=rola_n)) and id_dane_adresowe=(select id_dane_adresowe from
`dane_adresowe` where kraj_zamieszkania=kraj_zamieszkania_n and miasto=miasto_n and
ulica=ulica_n and numer_domu=numer_domu_n and numer_lokalu=numer_lokalu_n and
kod_pocztowy=kod_pocztowy_n) and imie=imie_n and pesel=pesel_n and
nazwisko=nazwisko_n and data_urodzenia=data_urodzenia_n and
plec=plec_n and numer_kontaktowy=numer_kontaktowy_n));
set identyfikator = (select id_prowadzacego from `prowadzacy` where
id_dane=identyfikator_danych);
set imie_email = (select imie from `dane_osobowe_i_kontaktowe` where
id_dane=identyfikator_danych);
set nazwisko_email = (select nazwisko from `dane_osobowe_i_kontaktowe` where
id_dane=identyfikator_danych);
update `prowadzacy`
    set email = concat(imie_email, '.', nazwisko_email, '@po.edu.pl')
    where id_prowadzacego = identyfikator;
elseif rola_n='d' then
    insert into `pracownik_dziekanatu`(`id_dane`,`email`) values
    ((select id_dane from `dane_osobowe_i_kontaktowe` where
    (id_uzytkownika=(select id_uzytkownika from `uzytkownicy` where
    (login=login_n and haslo=haslo_n and rola=rola_n)) and
    id_dane_adresowe=(select id_dane_adresowe from `dane_adresowe` where
    kraj_zamieszkania=kraj_zamieszkania_n and miasto=miasto_n and
    ulica=ulica_n and numer_domu=numer_domu_n and numer_lokalu=numer_lokalu_n and
    kod_pocztowy=kod_pocztowy_n) and imie=imie_n and pesel=pesel_n and
    nazwisko=nazwisko_n and data_urodzenia=data_urodzenia_n and
    plec=plec_n and numer_kontaktowy=numer_kontaktowy_n)),
    '@po.edu.pl');
set identyfikator_danych = (select id_dane from `dane_osobowe_i_kontaktowe` where
(id_uzytkownika=(select id_uzytkownika from `uzytkownicy` where
(login=login_n and haslo=haslo_n and rola=rola_n)) and
id_dane_adresowe=(select id_dane_adresowe from `dane_adresowe` where
kraj_zamieszkania=kraj_zamieszkania_n and miasto=miasto_n and
ulica=ulica_n and numer_domu=numer_domu_n and numer_lokalu=numer_lokalu_n and
kod_pocztowy=kod_pocztowy_n) and imie=imie_n and pesel=pesel_n and
nazwisko=nazwisko_n and data_urodzenia=data_urodzenia_n and
plec=plec_n and numer_kontaktowy=numer_kontaktowy_n));
set identyfikator = (select id_pracownika from `pracownik_dziekanatu` where
id_dane=identyfikator_danych);
set imie_email = (select imie from `dane_osobowe_i_kontaktowe` where
id_dane=identyfikator_danych);
set nazwisko_email = (select nazwisko from `dane_osobowe_i_kontaktowe` where
id_dane=identyfikator_danych);
update `pracownik_dziekanatu`
    set email = concat(imie_email, '.', nazwisko_email, '@po.edu.pl')
    where id_pracownika = identyfikator;
elseif rola_n='a' then
    insert into `administrator`(`id_dane`,`email`) values
    ((select id_dane from `dane_osobowe_i_kontaktowe` where

```

```

(id_uzytownika=(select id_uzytownika from `uzytkownicy` where
(login=login_n and haslo=haslo_n and rola=rola_n)) and
id_dane_adresowe=(select id_dane_adresowe from `dane_adresowe` where
kraj_zamieszkania=kraj_zamieszkania_n and miasto=miasto_n and
ulica=ulica_n and numer_domu=numer_domu_n and numer_lokalu=numer_lokalu_n and
kod_pocztowy=kod_pocztowy_n) and imie=imie_n and pesel=pesel_n and
nazwisko=nazwisko_n and data_urodzenia=data_urodzenia_n and
plec=plec_n and numer_kontaktowy=numer_kontaktowy_n)),
'@po.edu.pl');
set identyfikator_danych = (select id_dane from `dane_osobowe_i_kontaktowe` where
(id_uzytownika=(select id_uzytownika from `uzytkownicy` where (login=login_n and
haslo=haslo_n and rola=rola_n)) and id_dane_adresowe=(select id_dane_adresowe from
`dane_adresowe` where kraj_zamieszkania=kraj_zamieszkania_n and
miasto=miasto_n and ulica=ulica_n and numer_domu=numer_domu_n and
numer_lokalu=numer_lokalu_n and kod_pocztowy=kod_pocztowy_n) and
imie=imie_n and pesel=pesel_n and nazwisko=nazwisko_n and
data_urodzenia=data_urodzenia_n and
plec=plec_n and numer_kontaktowy=numer_kontaktowy_n));
set identyfikator = (select id_administratora from `administrator` where
id_dane=identyfikator_danych);
set imie_email = (select imie from `dane_osobowe_i_kontaktowe` where
id_dane=identyfikator_danych);
set nazwisko_email = (select nazwisko from `dane_osobowe_i_kontaktowe` where
id_dane=identyfikator_danych);
update `administrator`
    set email = concat(imie_email, '.', nazwisko_email, '@po.edu.pl')
    where id_administratora = identyfikator;
end if;
end if;
END

```

#### 4.4.12 Procedura składowana - "zapisz\_studenta"

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `zapisz_studenta`(student mediumint, grupa smallint,
biezacy_uzytkownik smallint)
BEGIN
    declare rola_uzytkownika char(1);

    set rola_uzytkownika = (select rola from `uzytkownicy` where id_uzytownika = biezacy_uzytkownik);

    if ((rola_uzytkownika='a') or (rola_uzytkownika='d')) then
        insert into `zapis`(`numer_indeksu`, `id_grupy`) values (student, grupa);
    end if;
END

```

#### 4.4.13 Procedura składowana - "odrzuć\_reklamacje"

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `odrzuć_reklamacje`(indeks_studenta mediumint,
grupa smallint, biezacy_uzytkownik smallint)
BEGIN
    declare rola_uzytkownika char(1);

    set rola_uzytkownika = (select rola from `uzytkownicy` where id_uzytownika = biezacy_uzytkownik);

    if ((rola_uzytkownika='a') or (rola_uzytkownika='p')) then
        update `zapis`
        set status_oceny = 'o'
        where numer_indeksu = indeks_studenta and id_grupy = grupa;
    end if;
END

```

#### 4.4.14 Procedura składowana - "popraw\_ocene"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `popraw_ocene`(nowa_ocena float, indeks_studenta mediumint, grupa smallint, biezacy_uzytkownik smallint)
BEGIN
    declare rola_uzytkownika char(1);

    set rola_uzytkownika = (select rola from `uzytkownicy` where id_uzytkownika = biezacy_uzytkownik);

    if ((rola_uzytkownika='a')or(rola_uzytkownika='p')) then
        update `zapis`
        set status_oceny = 'p',
        ocena = nowa_ocena
        where numer_indeksu = indeks_studenta and id_grupy = grupa;
    end if;
END
```

#### 4.4.15 Procedura składowana - "reklamuj\_ocene"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `reklamuj_ocene`(zapis smallint, biezacy_uzytkownik smallint)
BEGIN
    declare rola_uzytkownika char(1);

    set rola_uzytkownika = (select rola from `uzytkownicy` where id_uzytkownika = biezacy_uzytkownik);

    if ((rola_uzytkownika='a')or(rola_uzytkownika='s')) then
        update zapis
        set status_oceny = 'n'
        where id_zapisu = zapis;
    end if;
END
```

#### 4.4.16 Procedura składowana - "usun\_uzytkownika"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `usun_uzytkownika`(id_uzytkownika_usun smallint, biezacy_uzytkownik smallint)
BEGIN
    declare rola_uzytkownika char(1);
    declare rola_uzytkownika_usun char(1);
    declare mynumber smallint;

    set rola_uzytkownika = (select rola from `uzytkownicy` where id_uzytkownika = biezacy_uzytkownik);
    set rola_uzytkownika_usun = (select rola from `uzytkownicy` where id_uzytkownika = id_uzytkownika_usun);

    set mynumber = (select count(id_dane_adresowe) from `dane_osobowe_i_kontaktowe` where id_dane_adresowe=(select id_dane_adresowe from `dane_osobowe_i_kontaktowe` where id_uzytkownika=id_uzytkownika_usun));

    if ((rola_uzytkownika='a') or (rola_uzytkownika='d' and rola_uzytkownika_usun='s')) then
        if mynumber=1 then
            delete from `dane_adresowe` where id_dane_adresowe=(select id_dane_adresowe from `dane_osobowe_i_kontaktowe` where id_uzytkownika = id_uzytkownika_usun);
        end if;
        delete from `uzytkownicy` where id_uzytkownika = id_uzytkownika_usun;
    end if;
END
```

#### 4.4.17 Procedura składowana - "wprowadz\_ocene"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `wprowadz_ocene`(nowa_ocena float, indeks_studenta mediumint, grupa smallint, biezacy_uzytkownik smallint)
BEGIN
    declare rola_uzytkownika char(1);

    set rola_uzytkownika = (select rola from `uzytkownicy` where id_uzytkownika = biezacy_uzytkownik);

    if ((rola_uzytkownika='a')or(rola_uzytkownika='p')) then
        update `zapis`
        set ocena = nowa_ocena,
        status_oceny = 'w'
        where numer_indeksu = indeks_studenta and id_grupy = grupa;
    end if;
END
```

#### 4.4.18 Procedura składowana - "zatwierdz\_ocene"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `zatwierdz_ocene`(zapis smallint, biezacy_uzytkownik smallint)
BEGIN
    declare rola_uzytkownika char(1);

    set rola_uzytkownika = (select rola from `uzytkownicy` where id_uzytkownika = biezacy_uzytkownik);

    if ((rola_uzytkownika='a')or(rola_uzytkownika='s')) then
        update zapis
        set status_oceny = 't'
        where id_zapisu = zapis;
    end if;
END
```

#### 4.4.19 Procedura składowana - "zmien\_imie"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `zmien_imie`(imie_n varchar(20), biezacy_uzytkownik smallint)
BEGIN
    declare id_danych smallint;
    set id_danych = (select id_dane from `dane_osobowe_i_kontaktowe` where id_uzytkownika=biezacy_uzytkownik);
    update `dane_osobowe_i_kontaktowe`
    set imie = imie_n
    where id_dane = id_danych;
END
```

#### 4.4.20 Procedura składowana - "zmien\_kod\_pocztowy"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `zmien_kod_pocztowy`(kod_pocztowy_n varchar(10), biezacy_uzytkownik smallint)
BEGIN
    declare id_danych_adresowych smallint;
    set id_danych_adresowych = (select id_dane_adresowe from `dane_osobowe_i_kontaktowe` where id_uzytkownika=biezacy_uzytkownik);
    update `dane_adresowe`
    set kod_pocztowy = kod_pocztowy_n
    where id_dane_adresowe = id_danych_adresowych;
END
```

#### 4.4.21 Procedura składowana - "zmien\_kraj\_zamieszkania"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `zmien_kraj_zamieszkania`(kraj_zamieszkania_n varchar(30),
biezacy_uzytkownik smallint)
BEGIN
    declare id_danych_adresowych smallint;
    set id_danych_adresowych = (select id_dane_adresowe from `dane_osobowe_i_kontaktowe` where
id_uzytkownika=biezacy_uzytkownik);
    update `dane_adresowe`
        set kraj_zamieszkania = kraj_zamieszkania_n
        where id_dane_adresowe = id_danych_adresowych;
END
```

#### 4.4.22 Procedura składowana - "zmien\_miasto"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `zmien_miasto`(miasto_n varchar(30), biezacy_uzytkownik smallint)
BEGIN
    declare id_danych_adresowych smallint;
    set id_danych_adresowych = (select id_dane_adresowe from `dane_osobowe_i_kontaktowe` where
id_uzytkownika=biezacy_uzytkownik);
    update `dane_adresowe`
        set miasto = miasto_n
        where id_dane_adresowe = id_danych_adresowych;
END
```

#### 4.4.23 Procedura składowana - "zmien\_nazwisko"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `zmien_nazwisko`(nazwisko_n varchar(45),
biezacy_uzytkownik smallint)
BEGIN
    declare id_danych smallint;
    set id_danych = (select id_dane from `dane_osobowe_i_kontaktowe` where id_uzytkownika=biezacy_uzytkownik);
    update `dane_osobowe_i_kontaktowe`
        set nazwisko = nazwisko_n
        where id_dane = id_danych;
END
```

#### 4.4.24 Procedura składowana - "zmien\_numer\_domu"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `zmien_numer_domu`(numer_domu_n varchar(10),
biezacy_uzytkownik smallint)
BEGIN
    declare id_danych_adresowych smallint;
    set id_danych_adresowych = (select id_dane_adresowe from `dane_osobowe_i_kontaktowe` where
id_uzytkownika=biezacy_uzytkownik);
    update `dane_adresowe`
        set numer_domu = numer_domu_n
        where id_dane_adresowe = id_danych_adresowych;
END
```

#### 4.4.25 Procedura składowana - "zmien\_numer\_kontaktowy"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `zmien_numer_kontaktowy`(numer_kontaktowy_n varchar(9),
biezacy_uzytkownik smallint)
BEGIN
    declare id_danych smallint;
    set id_danych = (select id_dane from `dane_osobowe_i_kontaktowe` where id_uzytkownika=biezacy_uzytkownik);
    update `dane_osobowe_i_kontaktowe`
        set numer_kontaktowy = numer_kontaktowy_n
        where id_dane = id_danych;
END
```

#### 4.4.26 Procedura składowana - "zmien\_numer\_lokalu"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `zmien_numer_lokalu`(numer_lokalu_n varchar(10),  
biezacy_uzytkownik smallint)  
BEGIN  
    declare id_danych_adresowych smallint;  
    set id_danych_adresowych = (select id_dane_adresowe from `dane_osobowe_i_kontaktowe` where  
id_uzytkownika=biezacy_uzytkownik);  
    update `dane_adresowe`  
        set numer_lokalu = numer_lokalu_n  
        where id_dane_adresowe = id_danych_adresowych;  
END
```

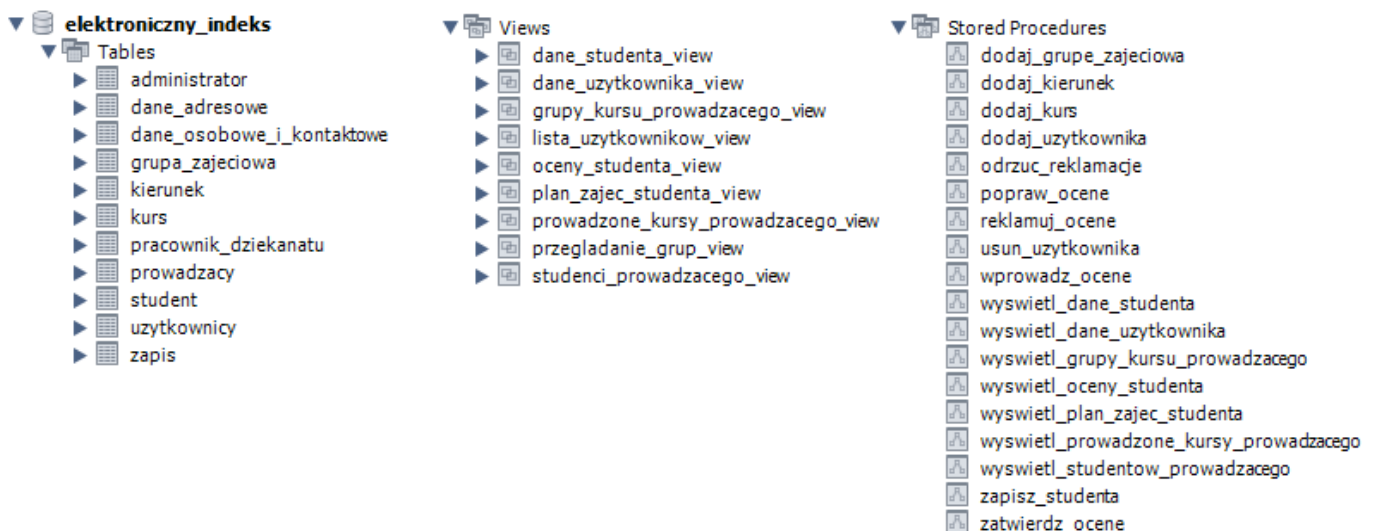
#### 4.4.27 Procedura składowana - "zmien\_plec"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `zmien_plec`(plec_n char(1), biezacy_uzytkownik smallint)  
BEGIN  
    declare id_danych smallint;  
    set id_danych = (select id_dane from `dane_osobowe_i_kontaktowe` where id_uzytkownika=biezacy_uzytkownik);  
    update `dane_osobowe_i_kontaktowe`  
        set plec = plec_n  
        where id_dane = id_danych;  
END
```

#### 4.4.28 Procedura składowana - "zmien\_ulice"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `zmien_ulice`(ulica_n varchar(30), biezacy_uzytkownik smallint)  
BEGIN  
    declare id_danych_adresowych smallint;  
    set id_danych_adresowych = (select id_dane_adresowe from `dane_osobowe_i_kontaktowe` where  
id_uzytkownika=biezacy_uzytkownik);  
    update `dane_adresowe`  
        set ulica = ulica_n  
        where id_dane_adresowe = id_danych_adresowych;  
END
```

### 4.5 Model bazy danych z tabelami, widokami oraz procedurami składowymi



Rysunek 7: Model bazy danych z tabelami, widokami oraz procedurami składowymi

## 4.6 Implementacja zabezpieczeń

Nasz projekt zawiera cztery role użytkowników: Administrator, Pracownik Dziekanatu, Prowadzący oraz Student. Każda rola ma pewien zakres możliwości oraz zestaw procedur z których może skorzystać. Przy każdej procedurze wprowadzającej zmiany w bazie danych, umieszczony został dodatkowy argument wywołania - id\_uzytkownika, który wywołuje daną procedurę. Na podstawie tego numeru id sprawdzamy role danego użytkownika, a na podstawie sprawdzonej roli określamy czy użytkownik o takich uprawnieniach powinien móc skorzystać z wywołanej procedury. Jeśli nie, procedura nie wprowadzi zmian w bazie danych. Dodatkowo w procedurach w których jest to konieczne sprawdzane jest jeszcze przed próbą dodania do bazy danych, czy dany rekord jest unikatowy. Poniżej kod procedury usun\_uzytkownika oraz test tej procedury. Test pokazuje, że będąc studentem o id\_uzytkownika = 3 nie jesteśmy w stanie usunąć administratora o id\_uzytkownika = 8.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `usun_uzytkownika`(id_uzytkownika_usun smallint,
biezacy_uzytkownik smallint)
BEGIN
    declare rola_uzytkownika char(1);
    declare rola_uzytkownika_usun char(1);
    declare mynumber smallint;

    set rola_uzytkownika = (select rola from `uzytkownicy` where id_uzytkownika = biezacy_uzytkownik);
    set rola_uzytkownika_usun = (select rola from `uzytkownicy` where id_uzytkownika = id_uzytkownika_usun);

    set mynumber = (select count(id_dane_adresowe) from `dane_osobowe_i_kontaktowe` where
id_dane_adresowe=(select id_dane_adresowe from `dane_osobowe_i_kontaktowe` where
id_uzytkownika=id_uzytkownika_usun));

    if ((rola_uzytkownika='a') or (rola_uzytkownika='d' and rola_uzytkownika_usun='s')) then
        if mynumber=1 then
            delete from `dane_adresowe` where id_dane_adresowe=(select id_dane_adresowe from
`dane_osobowe_i_kontaktowe` where id_uzytkownika = id_uzytkownika_usun);
        end if;
        delete from `uzytkownicy` where id_uzytkownika = id_uzytkownika_usun;
    end if;
END
```

```
select * from uzytkownicy;
```

	id_uzytkownika	login	haslo	rola
▶	1	admin	admin	a
	3	login1	haslo1	s
	4	login2	haslo2	p
	5	login3	haslo3	p
	6	login4	haslo4	d
	7	login5	haslo5	d
	8	login6	haslo6	a
	9	login7	haslo7	a
*	NULL	NULL	NULL	NULL

```
call usun_uzytkownika(8,3);
```

```
select * from uzytkownicy;
```

	id_uzytkownika	login	haslo	rola
▶	1	admin	admin	a
	3	login1	haslo1	s
	4	login2	haslo2	p
	5	login3	haslo3	p
	6	login4	haslo4	d
	7	login5	haslo5	d
	8	login6	haslo6	a
	9	login7	haslo7	a
*	NULL	NULL	NULL	NULL

Rysunek 8: Testy procedury w bazie danych związanej z zabezpieczeniami

Kolejnym testem zabezpieczeń, jest próba dodania użytkownika o identycznych danych co użytkownik już znajdujący się w bazie. Zaimplementowane zabezpieczenia sprawiają, że taka operacja jest niemożliwa, ze względu na identyczny pesel oraz login, które w naszej bazie muszą być unikatowe. Dzięki walidacji danych na samym początku procedury dodaj\_uzytkownika, mamy pewność, że do żadnej tabeli nie zostaną dodane nowe komórki, gdy jakkolwiek warunek nie jest spełniony.

```
select * from uzytkownicy;
```

	id_uzytkownika	login	haslo	rola
▶	1	admin	admin	a
	3	login1	haslo1	s
	4	login2	haslo2	p
	5	login3	haslo3	p
	6	login4	haslo4	d
	7	login5	haslo5	d
	8	login6	haslo6	a
	9	login7	haslo7	a
	10	login0	haslo0	s
*	NULL	NULL	NULL	NULL

```
call dodaj_uzytkownika('0000000000', 'ImieStudenta0', 'NazwiskoStudenta0', '2000-01-01', 'm', '0000000000', 'Kraj0', 'Miasto0', 'Ulica0', '0', '0a', '00000', 'login0', 'haslo0', 's', 'Automatyka', 1, 2, 1);
```

```
select * from uzytkownicy;
```

	id_uzytkownika	login	haslo	rola
▶	1	admin	admin	a
	3	login1	haslo1	s
	4	login2	haslo2	p
	5	login3	haslo3	p
	6	login4	haslo4	d
	7	login5	haslo5	d
	8	login6	haslo6	a
	9	login7	haslo7	a
	10	login0	haslo0	s
*	NULL	NULL	NULL	NULL

Rysunek 9: Testy procedury w bazie danych związanej z zabezpieczeniami



## 4.7 Testy procedur

Poniżej przedstawione zostały testy wszystkich zaimplementowanych przez nas procedur. Aby jednak móc którąkolwiek z procedur wywołać, musimy sobie ręcznie stworzyć użytkownika z uprawnieniami administratora. Taki użytkownik będzie mógł bez problemów dodać pozostałych użytkowników, kierunki, grupy, kursy oraz dane, co pozwoli nam przetestować działanie napisanego kodu. Warto jednak dodać, że będzie on istniał tylko w tabeli użytkownicy, to nam wystarczy, aby przygotować bazę do testów. Po dodaniu administratora rozpoczynamy więc od podstawowych procedur pozwalających na wprowadzanie pierwszych informacji do bazy danych. Przygotowujemy więc strukturę wydziału zaczynając od dodania odpowiednich kierunków, następnie tworzymy użytkowników każdej roli, tworzymy kursy i grupy zajęciowe oraz finalnie dodajemy studentów do konkretnych grup zajęciowych. Jak można zobaczyć na poniższych obrazkach, wszystkie testy dały pozytywny wynik, procedury dodające informacje do bazy danych funkcjonują poprawnie.

```
insert into `uzytkownicy`(`login`,`haslo`,`rola`) values ('admin','admin','a');
```

id_uzytkownika	login	haslo	rola
1	admin	admin	a

```
call dodaj_kierunek('Automatyka', 1, 2, 1);
call dodaj_kierunek('Automatyka', 3, 1, 1);
```

id_kierunku	nazwa_kierunku	semestr	stopien
1	Automatyka	1	2
2	Automatyka	3	1

```
call dodaj_uzytkownika('000000000000', 'ImieStudenta0', 'NazwiskoStudenta0', '2000-01-01', 'm', '000000000', 'Kraj0', 'Miasto0', 'Ulica0', '0', '0a', '00000', 'login0', 'haslo0', 's', 'Automatyka', 1, 2, 1);
call dodaj_uzytkownika('111111111111', 'ImieStudenta1', 'NazwiskoStudenta1', '2000-01-02', 'm', '1111111111', 'Kraj1', 'Miasto1', 'Ulica1', '1', '1a', '11111', 'login1', 'haslo1', 's', 'Automatyka', 3, 1, 1);
call dodaj_uzytkownika('222222222222', 'ImieProwadzacego2', 'NazwiskoProwadzacego2', '2000-01-03', 'm', '2222222222', 'Kraj2', 'Miasto2', 'Ulica2', '2', '2a', '22222', 'login2', 'haslo2', 'p', 'Automatyka', 1, 2, 1);
call dodaj_uzytkownika('333333333333', 'ImieProwadzacego3', 'NazwiskoProwadzacego3', '2000-01-04', 'm', '3333333333', 'Kraj3', 'Miasto3', 'Ulica3', '3', '3a', '33333', 'login3', 'haslo3', 'p', 'Automatyka', 3, 1, 1);
call dodaj_uzytkownika('444444444444', 'ImiePracownika4', 'NazwiskoPracownika4', '2444-01-05', 'm', '4444444444', 'Kraj4', 'Miasto4', 'Ulica4', '4', '4a', '44444', 'login4', 'haslo4', 'd', 'Automatyka', 1, 2, 1);
call dodaj_uzytkownika('555555555555', 'ImiePracownika5', 'NazwiskoPracownika5', '2000-01-06', 'm', '5555555555', 'Kraj5', 'Miasto5', 'Ulica5', '5', '5a', '55555', 'login5', 'haslo5', 'd', 'Automatyka', 3, 1, 1);
call dodaj_uzytkownika('666666666666', 'ImieAdmina6', 'NazwiskoAdmina6', '2666-01-07', 'm', '6666666666', 'Kraj6', 'Miasto6', 'Ulica6', '6', '6a', '66666', 'login6', 'haslo6', 'a', 'Automatyka', 1, 2, 1);
call dodaj_uzytkownika('777777777777', 'ImieAdmina7', 'NazwiskoAdmina7', '2600-01-08', 'm', '7777777777', 'Kraj7', 'Miasto7', 'Ulica7', '7', '7a', '77777', 'login7', 'haslo7', 'a', 'Automatyka', 3, 1, 1);
```

id_uzytkownika	login	haslo	rola	numer_indeksu	id_kierunku	id_dane	email
1	admin	admin	a	240000	1	1	240000@student.po.edu.pl
2	login0	haslo0	s	240001	2	2	240001@student.po.edu.pl
3	login1	haslo1	s				
4	login2	haslo2	p				
5	login3	haslo3	p				
6	login4	haslo4	d				
7	login5	haslo5	d				
8	login6	haslo6	a				
9	login7	haslo7	a				

id_pracownika	id_dane	email
1	5	ImiePracownika4.NazwiskoPracownika4@po.edu.pl
2	6	ImiePracownika5.NazwiskoPracownika5@po.edu.pl

id_prowadzacego	id_dane	email
1	3	ImieProwadzacego2.NazwiskoProwadzacego2@po.edu.pl
2	4	ImieProwadzacego3.NazwiskoProwadzacego3@po.edu.pl

id_administratora	id_dane	email
1	7	ImieAdmina6.NazwiskoAdmina6@po.edu.pl
2	8	ImieAdmina7.NazwiskoAdmina7@po.edu.pl

id_dane	id_uzytkownika	id_dane_adresowe	imie	nazwisko	pesel	data_urodzenia	plec	numer_kontaktowy
1	2	1	ImieStudenta0	NazwiskoStudenta0	000000000000	2000-01-01	m	0000000000
2	3	2	ImieStudenta1	NazwiskoStudenta1	111111111111	2000-01-02	m	1111111111
3	4	3	ImieProwadzacego2	NazwiskoProwadzacego2	222222222222	2000-01-03	m	2222222222
4	5	4	ImieProwadzacego3	NazwiskoProwadzacego3	333333333333	2000-01-04	m	3333333333
5	6	5	ImiePracownika4	NazwiskoPracownika4	444444444444	2444-01-05	m	4444444444
6	7	6	ImiePracownika5	NazwiskoPracownika5	555555555555	2000-01-06	m	5555555555
7	8	7	ImieAdmina6	NazwiskoAdmina6	666666666666	2666-01-07	m	6666666666
8	9	8	ImieAdmina7	NazwiskoAdmina7	777777777777	2600-01-08	m	7777777777

id_dane_adresowe	kraj_zamieszkania	miasto	ulica	numer_domu	numer_lokalu	kod_pocztowy
1	Kraj0	Miasto0	Ulica0	0	0a	00000
2	Kraj1	Miasto1	Ulica1	1	1a	11111
3	Kraj2	Miasto2	Ulica2	2	2a	22222
4	Kraj3	Miasto3	Ulica3	3	3a	33333
5	Kraj4	Miasto4	Ulica4	4	4a	44444
6	Kraj5	Miasto5	Ulica5	5	5a	55555
7	Kraj6	Miasto6	Ulica6	6	6a	66666
8	Kraj7	Miasto7	Ulica7	7	7a	77777

```
call dodaj_kurs(1, 'Kurs1', 5, 1);
call dodaj_kurs(1, 'Kurs2', 10, 1);
call dodaj_kurs(2, 'Kurs3', 5, 1);
call dodaj_kurs(2, 'Kurs4', 10, 1);
```

id_kursu	id_prowadzacego	nazwa_kursu	ects
1	1	Kurs1	5
2	1	Kurs2	10
3	2	Kurs3	5
4	2	Kurs4	10

```
call dodaj_grupe_zajeciowa(1, 'Bud1', 213, 1, 'W', '11:15:00', '13:00:00', 1);
call dodaj_grupe_zajeciowa(1, 'Bud2', 213, 1, 'W', '11:15:00', '13:00:00', 1);
call dodaj_grupe_zajeciowa(2, 'Bud3', 214, 2, 'L', '09:15:00', '11:00:00', 1);
call dodaj_grupe_zajeciowa(2, 'Bud4', 214, 2, 'L', '09:15:00', '11:00:00', 1);
call dodaj_grupe_zajeciowa(3, 'Bud1', 213, 1, 'W', '11:15:00', '13:00:00', 1);
call dodaj_grupe_zajeciowa(3, 'Bud2', 213, 1, 'W', '11:15:00', '13:00:00', 1);
call dodaj_grupe_zajeciowa(4, 'Bud3', 214, 2, 'L', '09:15:00', '11:00:00', 1);
call dodaj_grupe_zajeciowa(4, 'Bud4', 214, 2, 'L', '09:15:00', '11:00:00', 1);
```

id_grupy	id_kursu	budynek	sala	dzien_tygodnia	typ_zajec	godzina_rozpoczecia	godzina_zakonczenia
1	1	Bud1	213	1	W	11:15:00	13:00:00
2	1	Bud2	213	1	W	11:15:00	13:00:00
3	2	Bud3	214	2	L	09:15:00	11:00:00
4	2	Bud4	214	2	L	09:15:00	11:00:00
5	3	Bud1	213	1	W	11:15:00	13:00:00
6	3	Bud2	213	1	W	11:15:00	13:00:00
7	4	Bud3	214	2	L	09:15:00	11:00:00
8	4	Bud4	214	2	L	09:15:00	11:00:00

Rysunek 10: Testy procedur w bazie danych związanych z dodawaniem rekordów

```

call zapisz_studenta(240000, 1, 1);
call zapisz_studenta(240001, 1, 1);
call zapisz_studenta(240000, 2, 1);
call zapisz_studenta(240001, 2, 1);
call zapisz_studenta(240000, 3, 1);
call zapisz_studenta(240001, 3, 1);
call zapisz_studenta(240000, 4, 1);
call zapisz_studenta(240001, 4, 1);
call zapisz_studenta(240000, 5, 1);
call zapisz_studenta(240001, 5, 1);
call zapisz_studenta(240000, 6, 1);
call zapisz_studenta(240001, 6, 1);
call zapisz_studenta(240000, 7, 1);
call zapisz_studenta(240001, 7, 1);
call zapisz_studenta(240000, 8, 1);
call zapisz_studenta(240001, 8, 1);

```

	id_zapisu	numer_indeksu	id_grupy	ocena	status_oceny
▶	1	240000	1	NULL	NULL
	2	240001	1	NULL	NULL
	3	240000	2	NULL	NULL
	4	240001	2	NULL	NULL
	5	240000	3	NULL	NULL
	6	240001	3	NULL	NULL
	7	240000	4	NULL	NULL
	8	240001	4	NULL	NULL
	9	240000	5	NULL	NULL
	10	240001	5	NULL	NULL
	11	240000	6	NULL	NULL
	12	240001	6	NULL	NULL
	13	240000	7	NULL	NULL
	14	240001	7	NULL	NULL
	15	240000	8	NULL	NULL
	16	240001	8	NULL	NULL
•	NULL	NULL	NULL	NULL	NULL

Rysunek 11: Testy procedur w bazie danych związanych z dodawaniem rekordów

Kolejne testy dotyczyć będą procedur skupionych wokół ocen studentów. Poniżej zostały one sprawdzone z pozytywnym wynikiem. Umożliwiają one wprowadzanie, zatwierdzanie, reklamowanie i poprawianie ocen, jak też odrzucenie reklamacji.

```
select * from zapis where `id_grupy`=1 and `numer_indeksu`=240000;
```

	id_zapisu	numer_indeksu	id_grupy	ocena	status_oceny
▶	1	240000	1	NULL	NULL
*	NULL	NULL	NULL	NULL	NULL

```
call wprowadz_ocene(4.5, 240000, 1, 4);
```

```
select * from zapis where `id_grupy`=1 and `numer_indeksu`=240000;
```

	id_zapisu	numer_indeksu	id_grupy	ocena	status_oceny
▶	1	240000	1	4.5	w
*	NULL	NULL	NULL	NULL	NULL

Rysunek 12: Testy procedór w bazie danych związanych z wprowadzaniem ocen

```
select * from zapis where `id_grupy`=1 and `numer_indeksu`=240000;
```

	id_zapisu	numer_indeksu	id_grupy	ocena	status_oceny
▶	1	240000	1	4.5	w
*	NULL	NULL	NULL	NULL	NULL

```
call reklamuj_ocene(1, 2);
```

```
select * from zapis where `id_grupy`=1 and `numer_indeksu`=240000;
```

	id_zapisu	numer_indeksu	id_grupy	ocena	status_oceny
▶	1	240000	1	4.5	n
*	NULL	NULL	NULL	NULL	NULL

Rysunek 13: Testy procedór w bazie danych związanych z reklamowaniem ocen

```
select * from zapis where `id_grupy`=1 and `numer_indeksu`=240000;
```

	id_zapisu	numer_indeksu	id_grupy	ocena	status_oceny
▶	1	240000	1	4.5	n
*	NULL	NULL	NULL	NULL	NULL

```
call zatwierdz_ocene(1, 2);
```

```
select * from zapis where `id_grupy`=1 and `numer_indeksu`=240000;
```

	id_zapisu	numer_indeksu	id_grupy	ocena	status_oceny
▶	1	240000	1	4.5	t
*	NULL	NULL	NULL	NULL	NULL

Rysunek 14: Testy procedór w bazie danych związanych z zatwierdzaniem ocen

```
select * from zapis where `id_grupy`=1 and `numer_indeksu`=240000;
```

	id_zapisu	numer_indeksu	id_grupy	ocena	status_oceny
▶	1	240000	1	4.5	n
*	NULL	NULL	NULL	NULL	NULL

```
call odrzuc_reklamacje(240000, 1, 4);
```

```
select * from zapis where `id_grupy`=1 and `numer_indeksu`=240000;
```

	id_zapisu	numer_indeksu	id_grupy	ocena	status_oceny
▶	1	240000	1	4.5	o
*	NULL	NULL	NULL	NULL	NULL

Rysunek 15: Testy procedór w bazie danych związanych z odrzucaniem reklamacji

```
select * from zapis where `id_grupy`=1 and `numer_indeksu`=240000;
```

	id_zapisu	numer_indeksu	id_grupy	ocena	status_oceny
▶	1	240000	1	4.5	n
*	NULL	NULL	NULL	NULL	NULL

```
call popraw_ocene(5.0, 240000, 1, 4);
```

```
select * from zapis where `id_grupy`=1 and `numer_indeksu`=240000;
```

	id_zapisu	numer_indeksu	id_grupy	ocena	status_oceny
▶	1	240000	1	5	p
*	NULL	NULL	NULL	NULL	NULL

Rysunek 16: Testy procedór w bazie danych związanych z poprawianiem ocen

Ostatnia procedura ingerująca w zawartość bazy danych umożliwia usuwanie użytkowników. Testy tej procedury również przyniosły pozytywny efekt, co możemy zobaczyć na poniższym obrazku.

```
select * from uzytkownicy;
```

	id_uzytkownika	login	haslo	rola
▶	1	admin	admin	a
	2	login0	haslo0	s
	3	login1	haslo1	s
	4	login2	haslo2	p
	5	login3	haslo3	p
	6	login4	haslo4	d
	7	login5	haslo5	d
	8	login6	haslo6	a
	9	login7	haslo7	a
*	NULL	NULL	NULL	NULL

```
call usun_uzytkownika(2,1);
```

```
select * from uzytkownicy;
```

	id_uzytkownika	login	haslo	rola
▶	1	admin	admin	a
	3	login1	haslo1	s
	4	login2	haslo2	p
	5	login3	haslo3	p
	6	login4	haslo4	d
	7	login5	haslo5	d
	8	login6	haslo6	a
	9	login7	haslo7	a
*	NULL	NULL	NULL	NULL

Rysunek 17: Testy procedór w bazie danych związanych z usuwaniem uzytkownikow

Poniższe znajdują się testy wszystkich procedur związanych z wyświetlaniem widoków. Analizując wcześniej wprowadzone dane testowe i wyniki zwrócone przez procedury, można stwierdzić, że poniższe procedury działają poprawnie:

1. Procedura `wyswietl_dane_studenta` zwraca jeden rekord odpowiadający za wyświetlenie danych zalogowanego studenta.
2. Procedura `wyswietl_dane_uzytkownika` zwraca jeden rekord odpowiadający za wyświetlenie danych zalogowanego użytkownika.
3. Procedura `wyswietl_grupy_kursu_prowadzacego` zwraca rekordy odpowiadające za wyświetlenie prowadzonych grup kursu zalogowanego prowadzącego.
4. Procedura `wyswietl_oceny_studenta` zwraca rekordy odpowiadające za wyświetlenie ocen zalogowanego studenta.
5. Procedura `wyswietl_plan_zajec_studenta` zwraca rekordy odpowiadające za wyświetlenie planu zajęć zalogowanego studenta.
6. Procedura `wyswietl_grupy_kursu_prowadzacego` zwraca rekordy odpowiadające za wyświetlenie kursów prowadzącego przez zalogowanego prowadzącego.
7. Procedura `wyswietl_studentow_prowadzacego` zwraca rekordy odpowiadające za wyświetlenie studentów, których uczy zalogowany prowadzący.

1

call wyswietl\_dane\_studenta(2)

	nazwa_kierunku	stopien	semestr	numer_indeksu
►	Automatyka	2	1	240000

1

call wyswietl\_dane\_uzytkownika(5)

	imie	nazwisko	pesel	data_urodzenia	plec	numer_kontaktowy	kraj_zamieszkania	miasto	ulica	numer_domu	numer_lokalu
►	ImieProwadzacego3	NazwiskoProwadzacego3	3333333333	2000-01-04	m	333333333	Kraj3	Miasto3	Ulica3	3	3a

1

call wyswietl\_grupy\_kursu\_prowadzacego(4)

	id_grupy	budynek	sala	dzien_tygodnia	typ_zajec	godzina_rozpoczecia	godzina_zakonczenia
►	1	Bud1	213	1	W	11:15:00	13:00:00
	2	Bud2	213	1	W	11:15:00	13:00:00
	3	Bud3	214	2	L	09:15:00	11:00:00
	4	Bud4	214	2	L	09:15:00	11:00:00

1

call wyswietl\_oceny\_studenta(2)

	imie	▲ nazwisko	nazwa_kursu	ects	grupa_zajeciowa	ocena	status_oceny
►	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	5	1	NULL	NULL
	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	5	2	NULL	NULL
	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	10	3	NULL	NULL
	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	10	4	NULL	NULL
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	5	5	NULL	NULL
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	5	6	NULL	NULL
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	10	7	NULL	NULL
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	10	8	NULL	NULL

1

call wyswietl\_plan\_zajec\_studenta(2)

	imie	▲ nazwisko	nazwa_kursu	id_grupy	budynek	sala	dzien_tygodnia	typ_zajec	godzina_rozpoczecia	godzina_zakonczenia
►	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	1	Bud1	213	1	W	11:15:00	13:00:00
	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	2	Bud2	213	1	W	11:15:00	13:00:00
	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	3	Bud3	214	2	L	09:15:00	11:00:00
	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	4	Bud4	214	2	L	09:15:00	11:00:00
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	5	Bud1	213	1	W	11:15:00	13:00:00
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	6	Bud2	213	1	W	11:15:00	13:00:00
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	7	Bud3	214	2	L	09:15:00	11:00:00
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	8	Bud4	214	2	L	09:15:00	11:00:00

1

call wyswietl\_prowadzone\_kursy\_prowadzacego(4)

	nazwa_kursu	ects
►	Kurs2	10
	Kurs1	5

1

call wyswietl\_studentow\_prowadzacego(5)

	numer_indeksu	nazwa_kursu	id_grupy	typ_zajec	ocena	status_oceny
►	240000	Kurs3	5	W	NULL	NULL
	240000	Kurs3	5	W	NULL	NULL
	240000	Kurs3	6	W	NULL	NULL
	240000	Kurs3	6	W	NULL	NULL
	240000	Kurs4	7	L	NULL	NULL
	240000	Kurs4	7	L	NULL	NULL
	240000	Kurs4	8	L	NULL	NULL
	240000	Kurs4	8	L	NULL	NULL
	240001	Kurs3	5	W	NULL	NULL
	240001	Kurs3	5	W	NULL	NULL
	240001	Kurs3	6	W	NULL	NULL
	240001	Kurs3	6	W	NULL	NULL
	240001	Kurs4	7	L	NULL	NULL
	240001	Kurs4	7	L	NULL	NULL
	240001	Kurs4	8	L	NULL	NULL
	240001	Kurs4	8	L	NULL	NULL

Rysunek 18: Testy procedur w bazie danych związanych z widokami

## 4.8 Testowanie widoków

Poniżej znajdują się testy wszystkich zaimplementowanych widoków. Analizując wprowadzone wcześniej dane testowe i wyniki zwrócone przez widoki, można stwierdzić, że widoki działają poprawnie:

1. Widok `dane_studenta_view` zwraca listę danych wszystkich studentów. Widok ten będzie potrzebny do wybrania ze wszystkich danych danych aktualnie zalogowanego studenta.
2. Widok `dane_uzytkownika_view` zwraca listę danych wszystkich użytkowników. Widok ten będzie potrzebny do wybrania ze wszystkich danych danych aktualnie zalogowanego użytkownika.
3. Widok `grupy_kursu_prowadzacego_view` zwraca listę wszystkich grup\_kursu zapisanych do danego prowadzącego. Widok ten będzie potrzebny do wybrania ze wszystkich grup\_kursów tylko tych, którzy odpowiadają zalogowanemu prowadzącemu.
4. Widok `lista_uzytkownikow_view` zwraca listę wszystkich użytkowników dodanych do bazy danych.
5. Widok `oceny_studenta_view` zwraca listę wszystkich danych dotyczących ocen studentów. Widok ten będzie potrzebny do wybrania ze wszystkich danych dotyczących ocen tylko tych, którzy odpowiadają zalogowanemu studentowi.
6. Widok `plan_zajec_studenta_view` zwraca listę wszystkich danych dotyczących planu studentów. Widok ten będzie potrzebny do wybrania ze wszystkich danych dotyczących planu tylko tych, którzy odpowiadają zalogowanemu studentowi.
7. Widok `prowadzone_kursy_prowadzacego_view` zwraca listę wszystkich kursów zapisanych do danego prowadzącego. Widok ten będzie potrzebny do wybrania ze wszystkich kursów zapisanych do danego prowadzącego tylko tych, którzy odpowiadają zalogowanemu prowadzącemu.
8. Widok `przeglądanie_grup_view` zwraca listę wszystkich dostępnych grup w bazie danych.
9. Widok `studenci_prowadzacego_view` zwraca listę wszystkich studentów zapisanych do danego prowadzącego. Widok ten będzie potrzebny do wybrania ze wszystkich studentów tylko tych, którzy odpowiadają zalogowanemu prowadzącemu.

1 SELECT \* FROM dane\_studenta\_view

	id_uzytkownika	nazwa_kierunku	stopien	semestr	numer_indeksu
▶	2	Automatyka	2	1	240000
	3	Automatyka	1	3	240001

1 SELECT \* FROM dane\_uzytkownika\_view

	id_uzytkownika	imie	nazwisko	pesel	data_urodzenia	plec	numer_kontaktowy	kraj_zamieszkania	miasto	ulica	numer_domu	numer_lokalu
▶	2	ImieStudenta0	NazwiskoStudenta0	0000000000	2000-01-01	m	000000000	Kraj0	Miasto0	Ulica0	0	0a
	3	ImieStudenta1	NazwiskoStudenta1	1111111111	2000-01-02	m	111111111	Kraj1	Miasto1	Ulica1	1	1a
	4	ImieProwadzacego2	NazwiskoProwadzacego2	2222222222	2000-01-03	m	222222222	Kraj2	Miasto2	Ulica2	2	2a
	5	ImieProwadzacego3	NazwiskoProwadzacego3	3333333333	2000-01-04	m	333333333	Kraj3	Miasto3	Ulica3	3	3a
	6	ImiePracownika4	NazwiskoPracownika4	4444444444	2444-01-05	m	444444444	Kraj4	Miasto4	Ulica4	4	4a
	7	ImiePracownika5	NazwiskoPracownika5	5555555555	2000-01-06	m	555555555	Kraj5	Miasto5	Ulica5	5	5a
	8	ImieAdmina6	NazwiskoAdmina6	6666666666	2666-01-07	m	666666666	Kraj6	Miasto6	Ulica6	6	6a
	9	ImieAdmina7	NazwiskoAdmina7	7777777777	2600-01-08	m	777777777	Kraj7	Miasto7	Ulica7	7	7a

1 SELECT \* FROM grupy\_kursu\_prowadzacego\_view

	id_uzytkownika	id_grupy	budynek	sala	dzien_tygodnia	typ_zajec	godzina_rozpoczecia	godzina_zakonczenia
▶	4	1	Bud1	213	1	W	11:15:00	13:00:00
	4	2	Bud2	213	1	W	11:15:00	13:00:00
	4	3	Bud3	214	2	L	09:15:00	11:00:00
	4	4	Bud4	214	2	L	09:15:00	11:00:00
	5	5	Bud1	213	1	W	11:15:00	13:00:00
	5	6	Bud2	213	1	W	11:15:00	13:00:00
	5	7	Bud3	214	2	L	09:15:00	11:00:00
	5	8	Bud4	214	2	L	09:15:00	11:00:00

1 SELECT \* FROM lista\_uzytkownikow\_view

	id_uzytkownika	imie	nazwisko	rola
▶	2	ImieStudenta0	NazwiskoStudenta0	s
	3	ImieStudenta1	NazwiskoStudenta1	s
	4	ImieProwadzacego2	NazwiskoProwadzacego2	p
	5	ImieProwadzacego3	NazwiskoProwadzacego3	p
	6	ImiePracownika4	NazwiskoPracownika4	d
	7	ImiePracownika5	NazwiskoPracownika5	d
	8	ImieAdmina6	NazwiskoAdmina6	a
	9	ImieAdmina7	NazwiskoAdmina7	a

1 SELECT \* FROM oceny\_studenta\_view

	id_dane	imie	nazwisko	nazwa_kursu	ects	grupa_zajeciowa	ocena	status_oceny
▶	1	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	5	1	NULL	NULL
	1	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	5	2	NULL	NULL
	1	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	10	3	NULL	NULL
	1	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	10	4	NULL	NULL
	2	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	5	1	NULL	NULL
	2	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	5	2	NULL	NULL
	2	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	10	3	NULL	NULL
	2	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	10	4	NULL	NULL
	1	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	5	5	NULL	NULL
	1	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	5	6	NULL	NULL
	1	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	10	7	NULL	NULL
	1	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	10	8	NULL	NULL
	2	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	5	5	NULL	NULL
	2	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	5	6	NULL	NULL
	2	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	10	7	NULL	NULL
	2	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	10	8	NULL	NULL

1 SELECT \* FROM plan\_zajec\_studenta\_view

	id_dane	imie	nazwisko	nazwa_kursu	id_grupy	budynek	sala	dzien_tygodnia	typ_zajec	godzina_rozpoczecia	godzina_zakonczenia
▶	1	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	1	Bud1	213	1	W	11:15:00	13:00:00
	1	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	2	Bud2	213	1	W	11:15:00	13:00:00
	1	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	3	Bud3	214	2	L	09:15:00	11:00:00
	1	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	4	Bud4	214	2	L	09:15:00	11:00:00
	1	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	5	Bud1	213	1	W	11:15:00	13:00:00
	1	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	6	Bud2	213	1	W	11:15:00	13:00:00
	1	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	7	Bud3	214	2	L	09:15:00	11:00:00
	1	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	8	Bud4	214	2	L	09:15:00	11:00:00
	2	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	1	Bud1	213	1	W	11:15:00	13:00:00
	2	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	2	Bud2	213	1	W	11:15:00	13:00:00
	2	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	3	Bud3	214	2	L	09:15:00	11:00:00
	2	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	4	Bud4	214	2	L	09:15:00	11:00:00
	2	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	5	Bud1	213	1	W	11:15:00	13:00:00
	2	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	6	Bud2	213	1	W	11:15:00	13:00:00
	2	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	7	Bud3	214	2	L	09:15:00	11:00:00
	2	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	8	Bud4	214	2	L	09:15:00	11:00:00

1 SELECT \* FROM prowadzone\_kursy\_prowadzacego\_view

	id_uzytkownika	nazwa_kursu	ects
▶	4	Kurs1	5
	4	Kurs2	10
	5	Kurs3	5
	5	Kurs4	10

Rysunek 19: Testy widoków w bazie danych



1 SELECT \* FROM przegladanie\_grup\_view

	imie	nazwisko	nazwa_kursu	id_grupy	budynek	sala	dzien_tygodnia	typ_zajec	godzina_rozpoczecia	godzina_zakonczenia
▶	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	1	Bud1	213	1	W	11:15:00	13:00:00
	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	1	Bud1	213	1	W	11:15:00	13:00:00
	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	2	Bud2	213	1	W	11:15:00	13:00:00
	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs1	2	Bud2	213	1	W	11:15:00	13:00:00
	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	3	Bud3	214	2	L	09:15:00	11:00:00
	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	3	Bud3	214	2	L	09:15:00	11:00:00
	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	4	Bud4	214	2	L	09:15:00	11:00:00
	ImieProwadzacego2	NazwiskoProwadzacego2	Kurs2	4	Bud4	214	2	L	09:15:00	11:00:00
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	5	Bud1	213	1	W	11:15:00	13:00:00
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	5	Bud1	213	1	W	11:15:00	13:00:00
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	6	Bud2	213	1	W	11:15:00	13:00:00
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs3	6	Bud2	213	1	W	11:15:00	13:00:00
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	7	Bud3	214	2	L	09:15:00	11:00:00
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	7	Bud3	214	2	L	09:15:00	11:00:00
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	8	Bud4	214	2	L	09:15:00	11:00:00
	ImieProwadzacego3	NazwiskoProwadzacego3	Kurs4	8	Bud4	214	2	L	09:15:00	11:00:00

1 SELECT \* FROM studenci\_prowadzacego\_view

	id_uzytkownika	numer_indeksu	id_dane	nazwa_kursu	id_grupy	typ_zajec	ocena	status_oceny
▶	5	240000	1	Kurs3	5	W	NULL	NULL
	5	240000	1	Kurs3	5	W	NULL	NULL
	5	240000	1	Kurs3	6	W	NULL	NULL
	5	240000	1	Kurs3	6	W	NULL	NULL
	5	240000	1	Kurs4	7	L	NULL	NULL
	5	240000	1	Kurs4	7	L	NULL	NULL
	5	240000	1	Kurs4	8	L	NULL	NULL
	5	240000	1	Kurs4	8	L	NULL	NULL
	5	240001	2	Kurs3	5	W	NULL	NULL
	5	240001	2	Kurs3	5	W	NULL	NULL
	5	240001	2	Kurs3	6	W	NULL	NULL
	5	240001	2	Kurs3	6	W	NULL	NULL
	5	240001	2	Kurs4	7	L	NULL	NULL
	5	240001	2	Kurs4	7	L	NULL	NULL
	5	240001	2	Kurs4	8	L	NULL	NULL
	5	240001	2	Kurs4	8	L	NULL	NULL
	4	240000	1	Kurs1	1	W	NULL	NULL
	4	240000	1	Kurs1	1	W	NULL	NULL
	4	240000	1	Kurs1	2	W	NULL	NULL
	4	240000	1	Kurs1	2	W	NULL	NULL
	4	240000	1	Kurs2	3	L	NULL	NULL
	4	240000	1	Kurs2	3	L	NULL	NULL
	4	240000	1	Kurs2	4	L	NULL	NULL
	4	240000	1	Kurs2	4	L	NULL	NULL
	4	240001	2	Kurs1	1	W	NULL	NULL
	4	240001	2	Kurs1	1	W	NULL	NULL
	4	240001	2	Kurs1	2	W	NULL	NULL
	4	240001	2	Kurs1	2	W	NULL	NULL
	4	240001	2	Kurs2	3	L	NULL	NULL
	4	240001	2	Kurs2	3	L	NULL	NULL
	4	240001	2	Kurs2	4	L	NULL	NULL
	4	240001	2	Kurs2	4	L	NULL	NULL

Rysunek 20: Testy widoków w bazie danych

## 4.9 Testy wydajnościowe

Aby móc mówić o testach wydajnościowych naszej bazy danych, musieliśmy przygotować procedurę, która stworzy zadaną ilość danych testowych. Poniższa procedura dodaje określoną w wywołaniu liczbę użytkowników do naszej bazy danych. Warto zauważyć, że dodanie użytkownika, który jest studentem powoduje stworzenie rekordu w czterech tabelach ("uzytkownicy", "dane\_adresowe", "dane\_osobowe\_i\_kontaktowe" oraz "student"). Testy przeprowadziliśmy dla różnej liczby użytkowników, maksymalnie 7 tysięcy, co daje nam łącznie około 28 tysięcy rekordów we wszystkich tabelach łącznie. Ze względu na to, że nasz projekt przewiduje do 5112 użytkowników to nie było konieczności testów na większej ilości danych. Poniżej procedura, tabele i wykresy będące efektem testów wydajnościowych. Zważywszy na to, że dodawanie użytkowników czy też modyfikacja ich danych w większości przypadkach jest wykonywana indywidualnie, wynik testów można określić jako pozytywny.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `generuj_dane_testowe`(ilosc int)
BEGIN
    DECLARE counter INT DEFAULT 1;
    DECLARE pesel varchar(11);
    DECLARE imie varchar(20);
    DECLARE nazwisko varchar(45);
    DECLARE numer_kontaktowy varchar(9);
    DECLARE kraj_zamieszkania varchar(30);
    DECLARE miasto varchar(30);
    DECLARE ulica varchar(30);
    DECLARE numer_domu varchar(10);
    DECLARE numer_lokalu varchar(10);
    DECLARE kod_pocztowy varchar(5);
    DECLARE login varchar(15);
    DECLARE haslo varchar(15);

    insert into `uzytkownicy`(`login`,`haslo`,`rola`) values ('admin', 'admin', 'a');

    call dodaj_kierunek('Automatyka', 1, 2, 1);

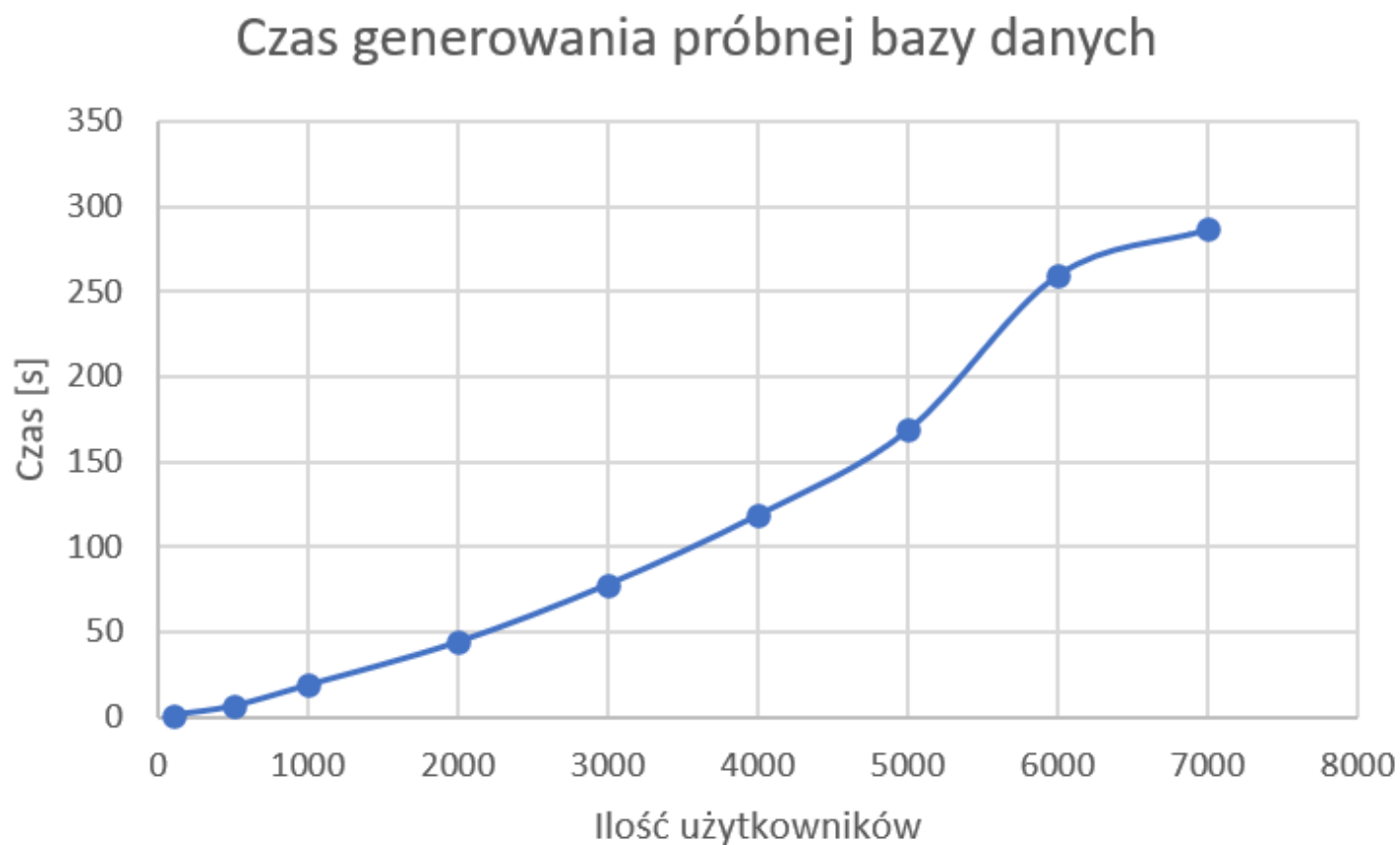
    WHILE counter <= ilosc DO
        SET pesel = concat('P', (select cast(counter as char(11))));
        SET imie = concat('I', (select cast(counter as char(20))));
        SET nazwisko = concat('N', (select cast(counter as char(45))));
        SET numer_kontaktowy = concat('NK', (select cast(counter as char(9))));
        SET kraj_zamieszkania = concat('KZ', (select cast(counter as char(30))));
        SET miasto = concat('M', (select cast(counter as char(30))));
        SET ulica = concat('U', (select cast(counter as char(30))));
        SET numer_domu = concat('ND', (select cast(counter as char(10))));
        SET numer_lokalu = concat('NL', (select cast(counter as char(10))));
        SET kod_pocztowy = concat('K', (select cast(counter as char(5))));
        SET login = concat('L', (select cast(counter as char(15))));
        SET haslo = concat('H', (select cast(counter as char(15))));

        call dodaj_uzytkownika(pesel, imie, nazwisko, '2000-01-01', 'm',
                                numer_kontaktowy, kraj_zamieszkania, miasto,
                                ulica, numer_domu, numer_lokalu, kod_pocztowy,
                                login, haslo, 's', 'Automatyka', 1, 2, 1);

        SET counter = counter + 1;
    END WHILE;
END
```

Ilość użytkowników	Czas generowania próbnej bazy danych
100	1,125
500	6,344
1000	18,672
2000	44,047
3000	77,89
4000	118,672
5000	168,609
6000	259,734
7000	286,484

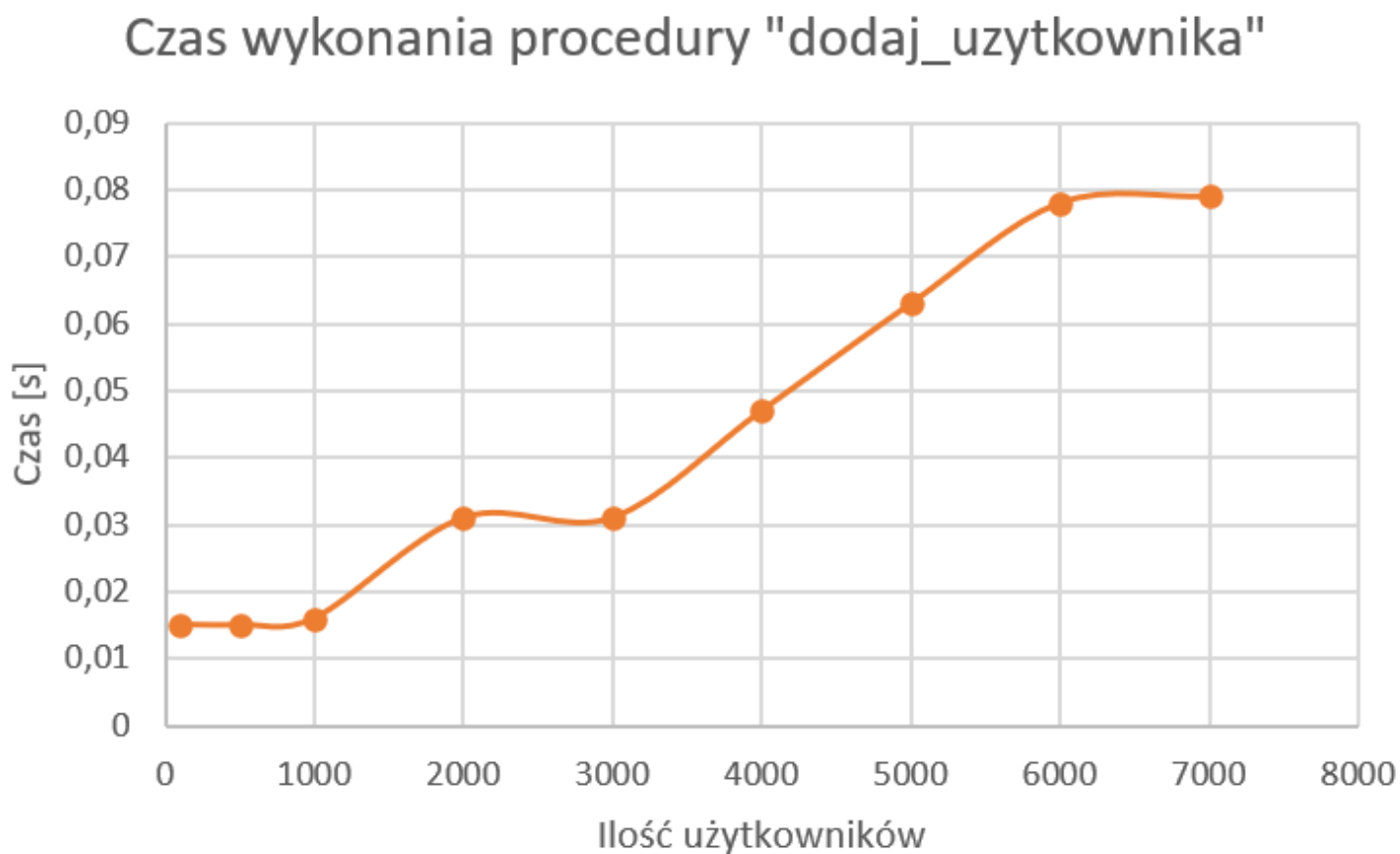
Rysunek 21: Tabela zależności czasu tworzenia próbnej bazy danych od ilości tworzonych użytkowników



Rysunek 22: Wykres zależności czasu tworzenia próbnej bazy danych od ilości tworzonych użytkowników

Ilość użytkowników	Czas wykonania procedury "dodaj_uzytkownika"
100	0,015
500	0,015
1000	0,016
2000	0,031
3000	0,031
4000	0,047
5000	0,063
6000	0,078
7000	0,079

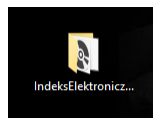
Rysunek 23: Tabela zależności czasu wykonania procedury "dodaj\_uzytkownika" od ilości istniejących użytkowników



Rysunek 24: Wykres zależności czasu wykonania procedury "dodaj\_uzytkownika" od ilości istniejących użytkowników

## 5 Implementacja i testy aplikacji

Na podstawie programu w języku C wygenerowaliśmy plik wykonywalny setup.exe. Do utworzenia takiego pliku wykorzystaliśmy narzędzie wbudowane w VisualStudio. Aby zainstalować naszą aplikację należy kliknąć dwukrotnie myszką w ikonkę setup.exe, wybrać ścieżkę, gdzie ma się plik zainstalować. Po instalacji możemy uruchomić program IndeksElektroniczny.



Rysunek 25: Folder z programem

Application Files	20.05.2021 18:34	Folder plików	
IndeksElektroniczny.application	20.05.2021 18:34	Application Manif...	6 KB
setup.exe	20.05.2021 18:34	Aplikacja	559 KB

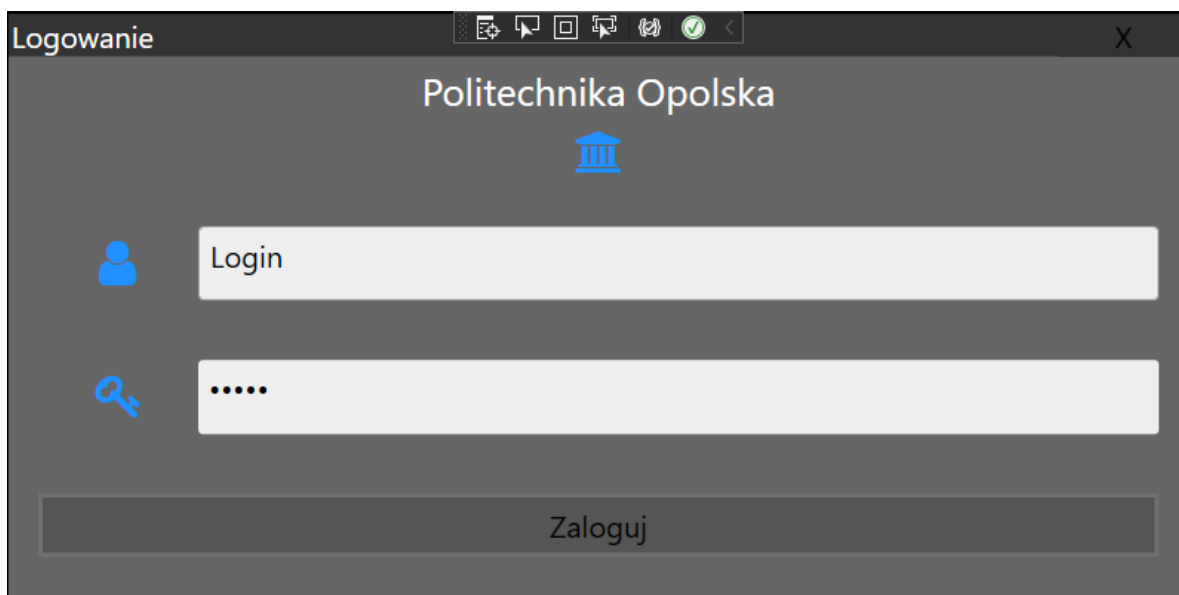
Rysunek 26: Pliki instalacyjne

### 5.1 Instrukcja użytkownika aplikacji

1. Student ma możliwość:

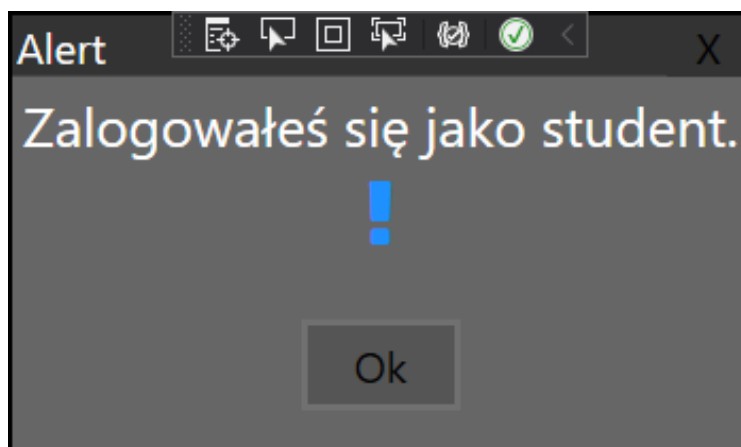
- Podglądu danych osobowych
- Edycji danych osobowych
- Podglądu danych studenta
- Podglądu i akceptowania ocen
- Podgląd grup zajęciowych
- Podgląd planu lekcji

2. Po uruchomieniu aplikacji pojawia się okno logowania:



Rysunek 27: Ekran logowania

### 3. Zalogowanie się jako student



Rysunek 28: Ekran komunikatu studenta

#### ■ Okno danych osobowych studenta

A web application window titled "Student" with a dark gray header. The header contains the logo of "Politechnika Opolska" and a navigation bar with tabs: "Twoje dane" (selected), "Indeks", "Zajęcia", and "Wyloguj". Below the header, the title "Dane osobowe" is centered. The form consists of a table with 12 rows, each representing a personal data field. The fields are: Pesel, Imię, Nazwisko, Data urodzenia, Płeć, Numer kontaktowy, Kraj zamieszkania, Miasto, Ulica, Numer domu, Numer lokalu, and Kod pocztowy. Each field has a corresponding value entered. At the bottom right of the form, there is a button labeled "Edytuj".

Dane osobowe	
Pesel	0000000001
Imię	Michał
Nazwisko	Prośba
Data urodzenia	02.03.1998
Płeć	m
Numer kontaktowy	987654321
Kraj zamieszkania	Polska
Miasto	Opole
Ulica	Domańskiego
Numer domu	69g
Numer lokalu	8
Kod pocztowy	45-819

Rysunek 29: Ekran danych osobowych studenta

## ■ Okno edycji danych osobowych studenta

Student

Politechnika Opolska

Twoje dane Indeks Zajęcia Wyloguj

Dane osobowe

Pesel	00000000001
Imię	Michał
Nazwisko	Prośba
Data urodzenia	02.03.1998
Płeć	m
Numer kontaktowy	987654321
Kraj zamieszkania	Polska
Miasto	Opole
Ulica	Domańskiego
Numer domu	69g
Numer lokalu	8
Kod pocztowy	45-819

Anuluj Zapisz zmiany

Rysunek 30: Ekran edycji danych osobowych studenta

## ■ Okno danych studenta

Student

Politechnika Opolska

Twoje dane Indeks Zajęcia Wyloguj

Dane Studenta

Dane Studenta	Nazwa kierunku	Automatyka
Oceny	Stopień	1
	Semestr	6
	Numer indeksu	240001

Rysunek 31: Ekran danych studenta

## ■ Okno indeksu studenta

Student

Politechnika Opolska

Twoje dane Indeks Zajęcia Wyloguj

Oceny

Dane Studenta	ID Zapisu	Imie	Nazwisko	Nazwa kursu	Ects	ID Grupy	Ocena	Status Oceny
Oceny	5	Krzysztof	Łopatka	Matematyka	2	5	0	E
	6	Krzysztof	Łopatka	Matematyka	2	6	0	E
	7	Krzysztof	Łopatka	Fizyka	3	7	0	E
	8	Krzysztof	Łopatka	Fizyka	3	8	0	E

Rysunek 32: Ekran edycji indeksu studenta

## ■ Okno przeglądania grup zajęciowych

Student

Politechnika Opolska

Twoje dane Indeks Zajęcia Wyloguj

Przeglądanie grup

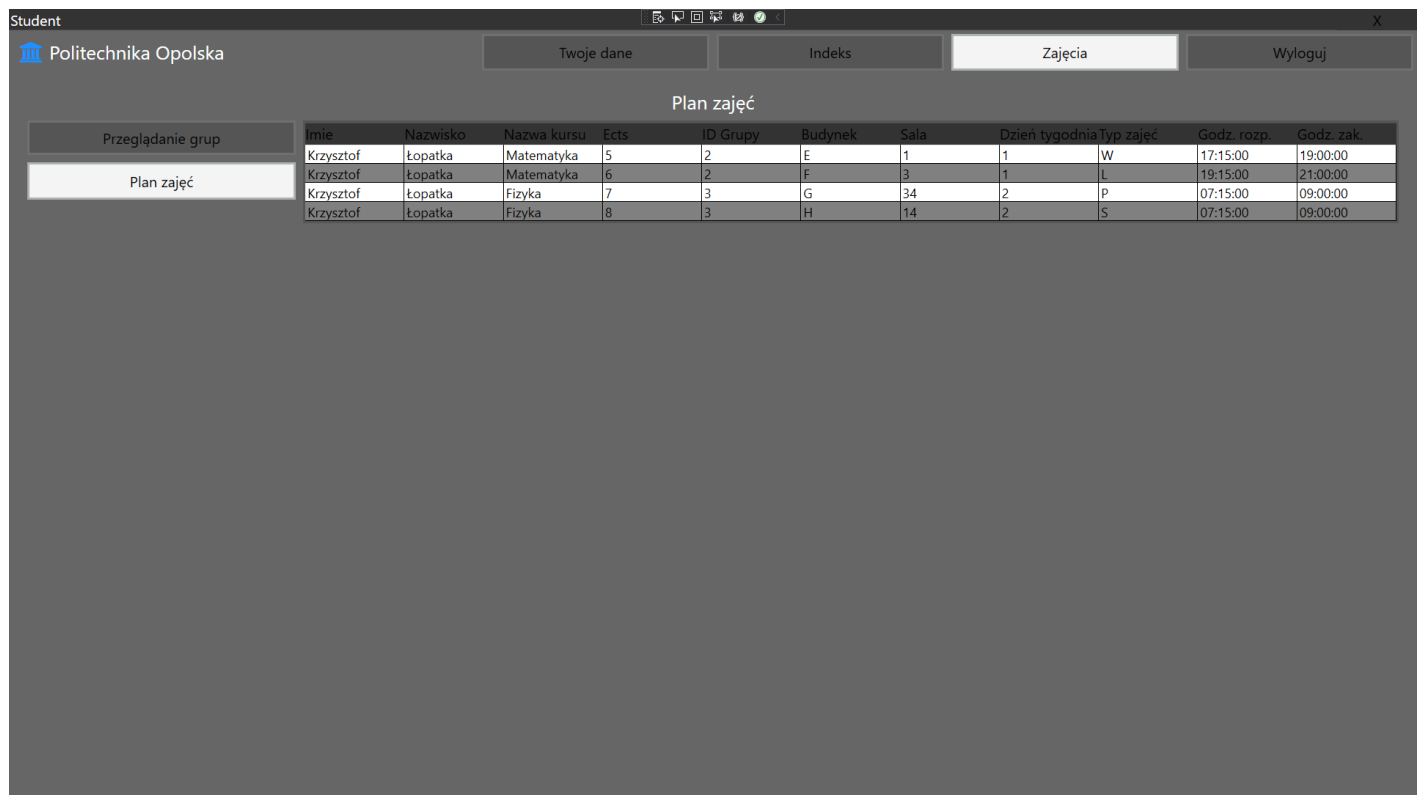
Przeglądanie grup Wszystkie Szukaj

Plan zajęć	Imie	Nazwisko	Nazwa kursu	Ects	ID Grupy	Budynek	Sala	Dzień tygodnia	Typ zajęć	Godz. rozp.	Godz. zak.
Plan zajęć	Krzysztof	Łopatka	SPD	5	1	A	21	1	W	11:15:00	13:00:00
	Krzysztof	Łopatka	SPD	5	2	B	23	1	L	13:15:00	15:00:00
	Krzysztof	Łopatka	Bazy Danych	10	3	C	14	2	P	09:15:00	11:00:00
	Krzysztof	Łopatka	Bazy Danych	10	4	D	2	2	S	15:15:00	17:00:00
	Krzysztof	Łopatka	Matematyka	2	5	E	1	1	W	17:15:00	19:00:00
	Krzysztof	Łopatka	Matematyka	2	6	F	3	1	L	19:15:00	21:00:00
	Krzysztof	Łopatka	Fizyka	3	7	G	34	2	P	07:15:00	09:00:00
	Krzysztof	Łopatka	Fizyka	3	8	H	14	2	S	07:15:00	09:00:00
	Karolina	Rogalik	Elektrotechnika	3	9	I	16	2	W	09:15:00	11:00:00
	Karolina	Rogalik	Elektrotechnika	3	10	J	25	2	L	13:15:00	15:00:00
	Karolina	Rogalik	SNIN	7	11	K	28	2	P	17:15:00	19:00:00
	Karolina	Rogalik	SNIN	7	12	L	19	2	S	09:15:00	11:00:00
	Karolina	Rogalik	Algebra	6	13	M	18	2	W	07:15:00	09:00:00
	Karolina	Rogalik	Algebra	6	14	N	12	2	L	13:15:00	15:00:00
	Karolina	Rogalik	Analiza Matema	8	15	O	11	2	P	17:15:00	19:00:00
	Karolina	Rogalik	Analiza Matema	8	16	P	22	2	S	19:15:00	21:00:00

Rysunek 33: Ekran przeglądania grup zajęciowych



## ■ Okno planu zajęć studenta



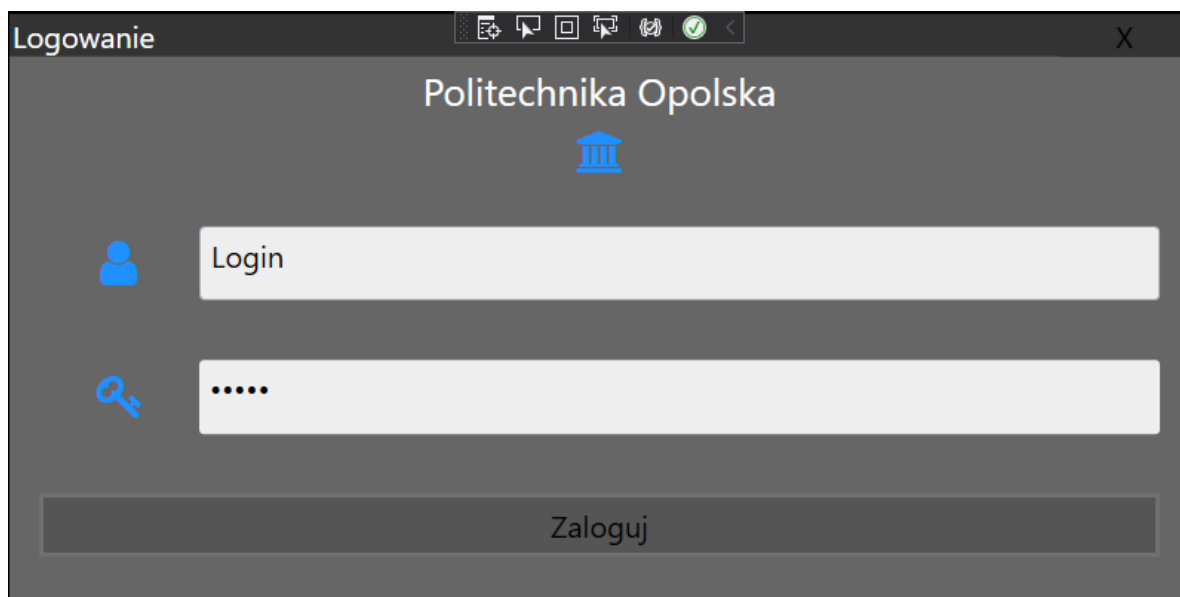
Imię	Nazwisko	Nazwa kursu	Ects	ID Grupy	Budynek	Sala	Dzień tygodnia	Typ zajęć	Godz. rozp.	Godz. zak.
Krzysztof	Łopatka	Matematyka	5	2	E	1	1	W	17:15:00	19:00:00
Krzysztof	Łopatka	Fizyka	7	3	G	34	2	P	07:15:00	09:00:00
Krzysztof	Łopatka	Fizyka	8	3	H	14	2	S	07:15:00	09:00:00

Rysunek 34: Ekran planu zajęć studenta

### 4. Prowadzący ma możliwość:

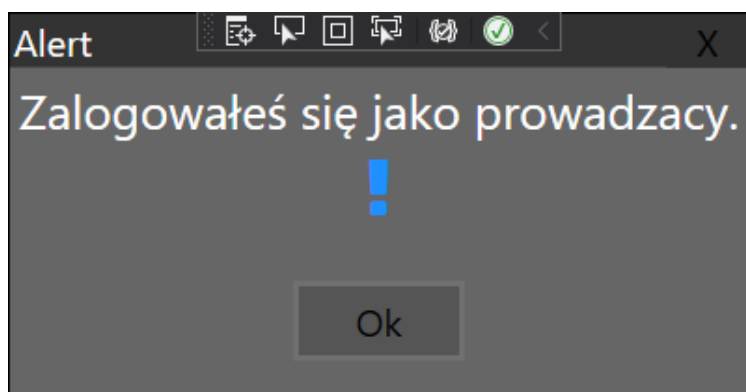
- Podglądu danych osobowych
- Edycji danych osobowych
- Podglądu prowadzonych kursów
- Podglądu prowadzonych grup kursów
- Podgląd prowadzonych studentów
- Edycji ocen studentów

5. Po uruchomieniu aplikacji pojawia się okno logowania:



Rysunek 35: Ekran logowania

6. Zalogowanie się jako prowadzący



Rysunek 36: Ekran komunikatu prowadzącego

## ■ Okno danych osobowych prowadzącego

Prowadzący

Politechnika Opolska

Twoje dane   Zajęcia   Ocenianie   Wyloguj

Dane osobowe

Pesel	1111111112
Imię	Karolina
Nazwisko	Rogalik
Data urodzenia	01.03.1970
Płeć	k
Numer kontaktowy	123321123
Kraj zamieszkania	Polska
Miasto	Wrocław
Ulica	Uroczą
Numer domu	27g
Numer lokalu	7
Kod pocztowy	49-076

Edytuj

Rysunek 37: Ekran danych osobowych prowadzącego

## ■ Okno edycji danych osobowych prowadzącego

Prowadzący

Politechnika Opolska

Twoje dane   Zajęcia   Ocenianie   Wyloguj

Dane osobowe

Pesel	1111111112
Imię	Karolina
Nazwisko	Rogalik
Data urodzenia	01.03.1970
Płeć	k
Numer kontaktowy	123321123
Kraj zamieszkania	Polska
Miasto	Wrocław
Ulica	Uroczą
Numer domu	27g
Numer lokalu	7
Kod pocztowy	49-076

Anuluj   Zapisz zmiany

Rysunek 38: Ekran edycji danych osobowych prowadzącego

■ Okno prowadzonych kursów prowadzącego

Prowadzący

Politechnika Opolska

Twoje dane

Zajecia

Ocenianie

Wyloguj

Prowadzone kursy

ID Kursu	Nazwa kursu	Ects	
5	Elektrotechnika	3	
6	SNIN	7	
7	Algebra	6	
8	Analiza Matematyczna	8	

Rysunek 39: Ekran prowadzonych kursów prowadzącego

■ Okno prowadzonych grup kursów osobowych prowadzącego

Prowadzący

Politechnika Opolska

Twoje dane

Zajęcia

Ocenianie

Wyloguj

Grupy kursu: Analiza Matematyczna

ID Grupy	Budynek	Sala	Dzień tygodnia	Typ zajęć	Godz. rozp.	Godz. zak.
15	O	11	2	P	17:15:00	19:00:00
16	P	22	2	S	19:15:00	21:00:00

Powrót

Rysunek 40: Ekran prowadzonych grup kursów prowadzącego

## ■ Okno studentów prowadzącego

Prowadzący

Politechnika Opolska

Twoje dane Zajecia **Ocenianie** Wyloguj

Ocenianie

Wszystkie Szukaj

Numer indeksu	Nazwa kursu	ID Grupy	Typ zajęć	Ocena	Status oceny
240002	Elektrotechnika	9	W	0	E
240002	Elektrotechnika	10	L	0	E
240002	SNIN	11	P	0	E
240002	SNIN	12	S	0	E
240003	Algebra	13	W	0	E
240003	Algebra	14	L	0	E
240003	Analiza Matematyczna	15	P	0	E
240003	Analiza Matematyczna	16	S	0	E

Rysunek 41: Ekran studentów prowadzącego

## ■ Okno studenta prowadzącego

Prowadzący

Politechnika Opolska

Twoje dane Zajecia Ocenianie Wyloguj

Student: 240000

Numer Indeksu	240000
Kurs	Elektrotechnika
Grupa	10
Typ zajęć	L
Ocena	0
Status oceny	E

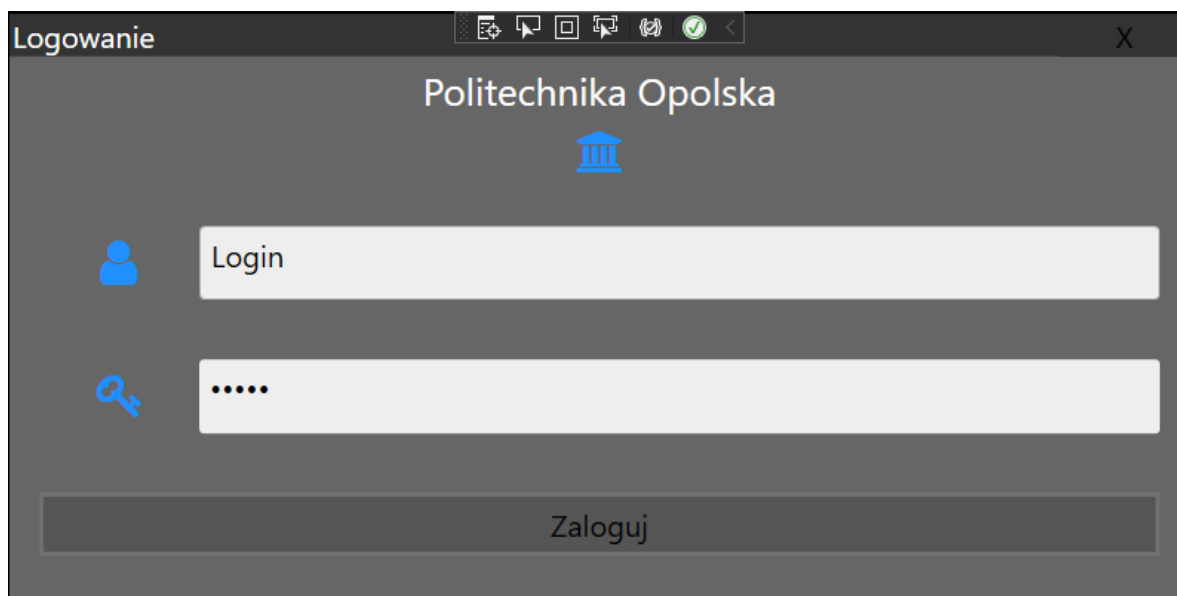
Powrót Edytuj

Rysunek 42: Ekran studentaw prowadzącego

7. Pracownik dziekanatu ma możliwość:

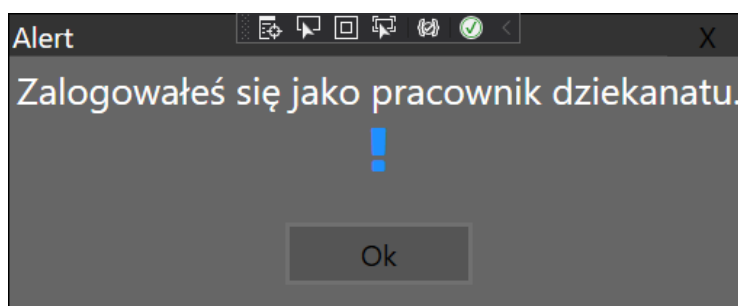
- Podglądu danych osobowych
- Edycji danych osobowych
- Podglądu danych studentów
- Dodanie
- Podgląd prowadzonych studentów
- Edycji ocen studentów

8. Po uruchomieniu aplikacji pojawia się okno logowania:



Rysunek 43: Ekran logowania

9. Zalogowanie się jako pracownik dziekanatu



Rysunek 44: Ekran komunikatu pracownika dziekanatu

## ■ Okno danych osobowych pracownika dziekanatu

The screenshot shows a web application window titled 'Pracownik dziekanatu'. At the top left is the logo of Politechnika Opolska. To the right of the logo are three tabs: 'Twoje dane' (selected), 'Studenci', and 'Wyloguj'. Below the tabs is the title 'Dane osobowe'. The form contains the following fields:

Pesel	2222222223
Imie	Monika
Nazwisko	Korzeniec
Data urodzenia	01.05.1978
Plec	m
Numer kontaktowy	456654456
Kraj zamieszkania	Polska
Miasto	Wroclaw
Ulica	Kolorowa
Numer domu	19b
Numer lokalu	4
Kod pocztowy	41-076

An 'Edytuj' button is located at the bottom right of the form.

Rysunek 45: Ekran danych osobowych pracownika dziekanatu

## ■ Okno edycji danych osobowych pracownika dziekanatu

This screenshot is identical to the previous one, but the 'Edytuj' button at the bottom right has been replaced by two buttons: 'Anuluj' and 'Zapisz zmiany'.

Rysunek 46: Ekran edycji danych osobowych pracownika dziekanatu

## ■ Okno listy studentów dziekanatu

Pracownik dziekanatu

Politechnika Opolska

Twoje dane Studenci Wyloguj

Lista studentów

ID użytkownika	Numer indeksu	Imię	Nazwisko	Pesel	Nazwa kierunku	Stopień	Semestr	Numer kontaktowy
2	240000	Patryk	Wieczorek	00000000000	Automatyka	1	6	123456789
3	240001	Michał	Prośba	00000000001	Automatyka	1	6	987654321
4	240002	Adam	Kowalski	00000000002	Elektronika	2	1	123654321
5	240003	Ania	Brzęczyśzykiewicz	00000000003	Telekomunikacja	2	3	987654321

Dodaj studenta

Rysunek 47: Ekran listy studentów dziekanatu

## ■ Okno studenta dziekanatu

Pracownik dziekanatu

Politechnika Opolska

Twoje dane Studenci Wyloguj

Dane studenta

Numer indeksu	240002
Pesel	00000000002
Imię	Adam
Nazwisko	Kowalski
Data urodzenia	24.01.1997 00:00:00
Płeć	m
Numer kontaktowy	123654321
Kraj zamieszkania	Polska
Miasto	Wrocław
Ulica	Dawida
Numer domu	13b
Numer lokalu	2
Kod pocztowy	43-718
Kierunek	Elektronika
Semestr	1
Stopień	2

Anuluj Usuń studenta Zapisz zmiany

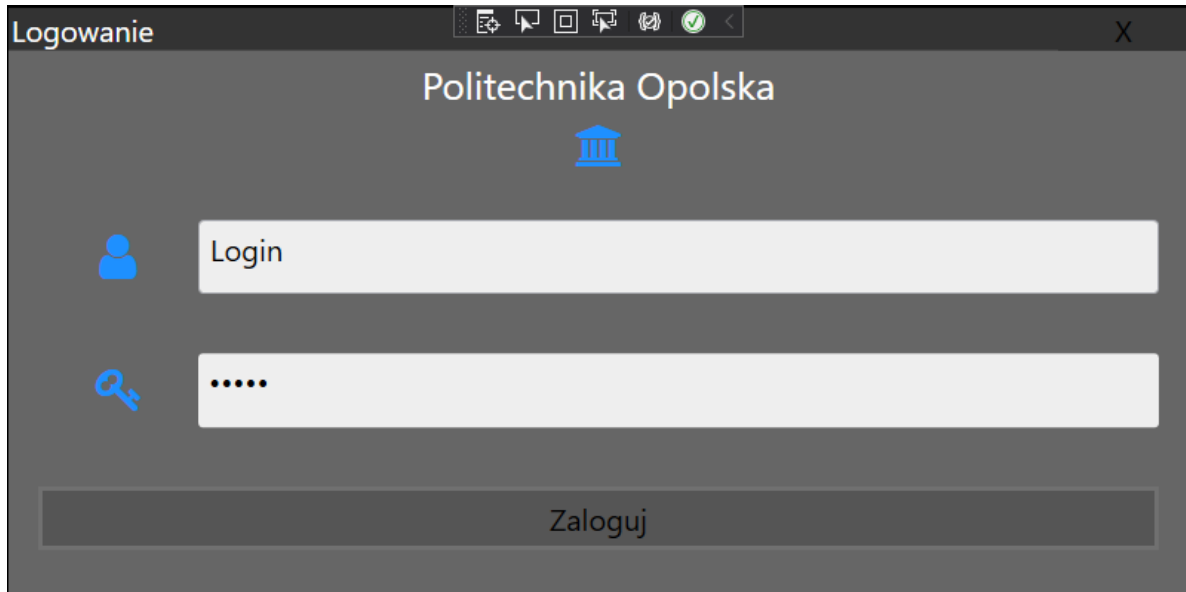
Rysunek 48: Ekran studenta dziekanatu



10. Administrator ma możliwość:

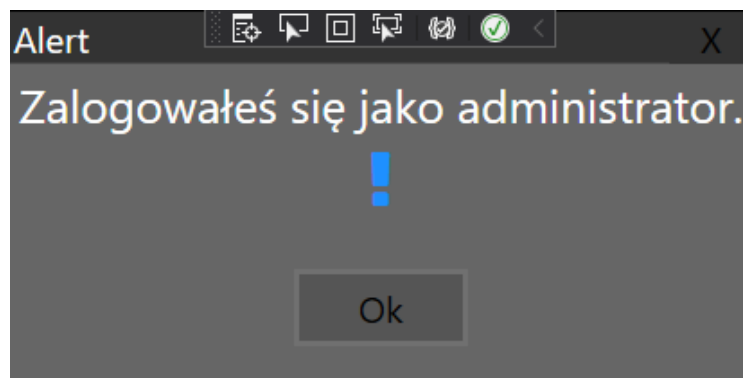
- Podglądu danych osobowych
- Edycji danych osobowych
- Podglądu użytkowników
- Dodanie użytkownika
- Podgląd danego użytkownika
- Edycji danych użytkownika

11. Po uruchomieniu aplikacji pojawia się okno logowania:



Rysunek 49: Ekran logowania

12. Zalogowanie się jako administrator



Rysunek 50: Ekran komunikatu administratora

## ■ Okno danych osobowych administratora

The screenshot shows a web application window titled 'Administrator'. The header bar includes the logo of 'Politechnika Opolska' and three navigation buttons: 'Twoje dane' (highlighted), 'Użytkownicy', and 'Wyloguj'. The main content area is titled 'Dane osobowe' and contains a form with the following fields:

Pesel	3333333333
Imie	Konrad
Nazwisko	Wolny
Data urodzenia	07.01.1966 00:00:00
Płeć	m
Numer kontaktowy	666666666
Kraj zamieszkania	Polska
Miasto	Wrocław
Ulica	Zielona
Numer domu	27b
Numer lokalu	4
Kod pocztowy	41-076

An 'Edytuj' button is located at the bottom right of the form area.

Rysunek 51: Ekran danych osobowych administratora

## ■ Okno edycji danych osobowych administratora

This screenshot shows the same 'Administrator' window, but with the 'Twoje dane' button highlighted, indicating the edit mode. The 'Dane osobowe' form fields are identical to the previous screenshot. At the bottom right, there are two buttons: 'Anuluj' and 'Zapisz zmiany'.

Rysunek 52: Ekran edycji administratora dziekanatu

## ■ Okno listy użytkowników administratora

Administrator

Politechnika Opolska

Twoje dane Użytkownicy Wyloguj

Lista Użytkowników

ID użytkownika	Imię	Nazwisko	Rola
10	Konrad	Wolny	a
8	Ryszard	Konradek	d
9	Monika	Korzeniec	d
6	Krzysztof	Łopatka	p
7	Karolina	Rogalik	p
2	Patryk	Wieczorek	s
3	Michał	Prośba	s
4	Adam	Kowalski	s
5	Ania	Brzęczyszczkiewicz	s

Dodaj Użytkownika

Rysunek 53: Ekran listy użytkowników administratora

## ■ Okno użytkownika administratora

Administrator

Politechnika Opolska

Twoje dane Użytkownicy Wyloguj

Dane użytkownika

Pesel	2222222223
Imię	Monika
Nazwisko	Korzeniec
Data urodzenia	01.05.1978
Płeć	m
Numer kontaktowy	456654456
Kraj zamieszkania	Polska
Miasto	Wrocław
Ulica	Kolorowa
Numer domu	19b
Numer lokalu	4
Kod pocztowy	41-076
Login	Monika60
Hasło	Monika60
Rola	d

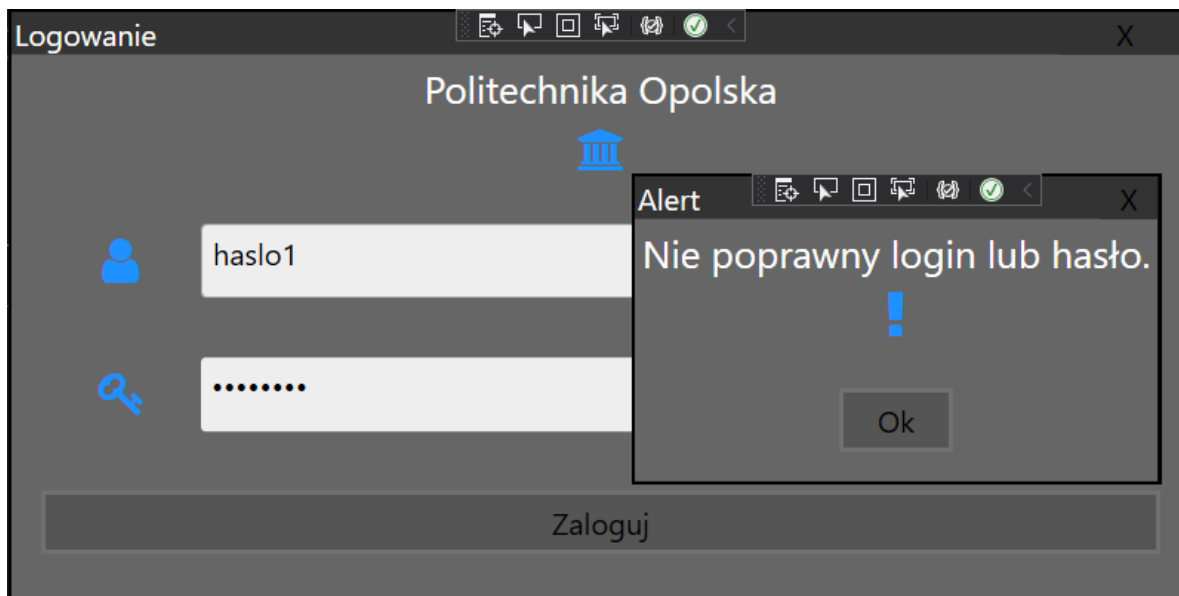
Anuluj Usuń użytkownika Zapisz Zmiany

Rysunek 54: Ekran użytkownika administratora

## 6 Testy aplikacji

### 6.1 Test wprowadzenia błędnego hasła

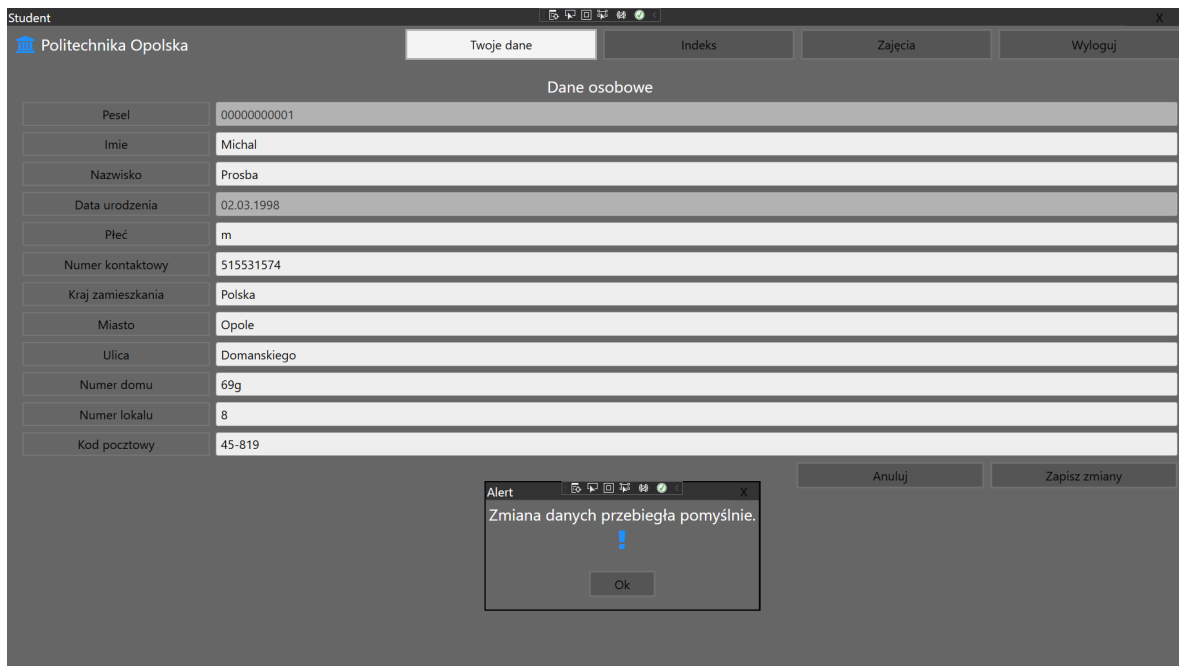
- Jeśli użytkownik wpisze błędny login lub hasło aplikacja wysyła alert o błędnym danych logowania



Rysunek 55: Alert błędne dane logowania

### 6.2 Test zmiany danych osobowych użytkownika

- Jeśli użytkownik spróbuje zmienić dane osobowe zgodnie z poprawnością danych to pojawia się alert o poprawnej zmianie danych. Dane te są zapisywane także poprawnie w bazie danych.



Rysunek 56: Alert zmiana danych przebiegła pomyślnie

- Jeśli użytkownik spróbuje zmienić dane osobowe niezgodnie z poprawnością danych to pojawia się alert o niepoprawnych danych. Dane te nie są zapisywane do bazy danych.

The screenshot shows the 'Student' application interface. At the top, there's a header with 'Politechnika Opolska' and navigation buttons: 'Twoje dane', 'Indeks', 'Zajęcia', and 'Wyloguj'. Below the header, the title 'Dane osobowe' is centered. A form contains the following fields and values:

Pesel	0000000001
Imię	Michał
Nazwisko	Prośba
Data urodzenia	02.03.1998
Płeć	m
Numer kontaktowy	5155315741
Kraj zamieszkania	Polska
Miasto	Opole
Ulica	Domanskiego
Numer domu	69g
Numer lokalu	8
Kod pocztowy	45-819

At the bottom right of the form are buttons 'Anuluj' and 'Zapisz zmiany'. An 'Alert' dialog box is open in the center, displaying the message: 'Numer kontaktowy - niepoprawna ilość znaków. Wymagana ilość znaków: 9'. The dialog has an 'Ok' button.

Rysunek 57: Alert niepoprawna ilość znaków

### 6.3 Test wprowadzania oceny przez prowadzącego

- Jeśli prowadzący spróbuje zmienić ocenę studenta zgodnie z poprawnością danych to pojawia się alert o poprawnej zmianie danych. Dane te są zapisywane, także poprawnie w bazie danych.

The screenshot shows the 'Prowadzący' application interface. At the top, there's a header with 'Politechnika Opolska' and navigation buttons: 'Twoje dane', 'Zajęcia', 'Ocenianie', and 'Wyloguj'. Below the header, the title 'Student: 240001' is centered. A form contains the following fields and values:

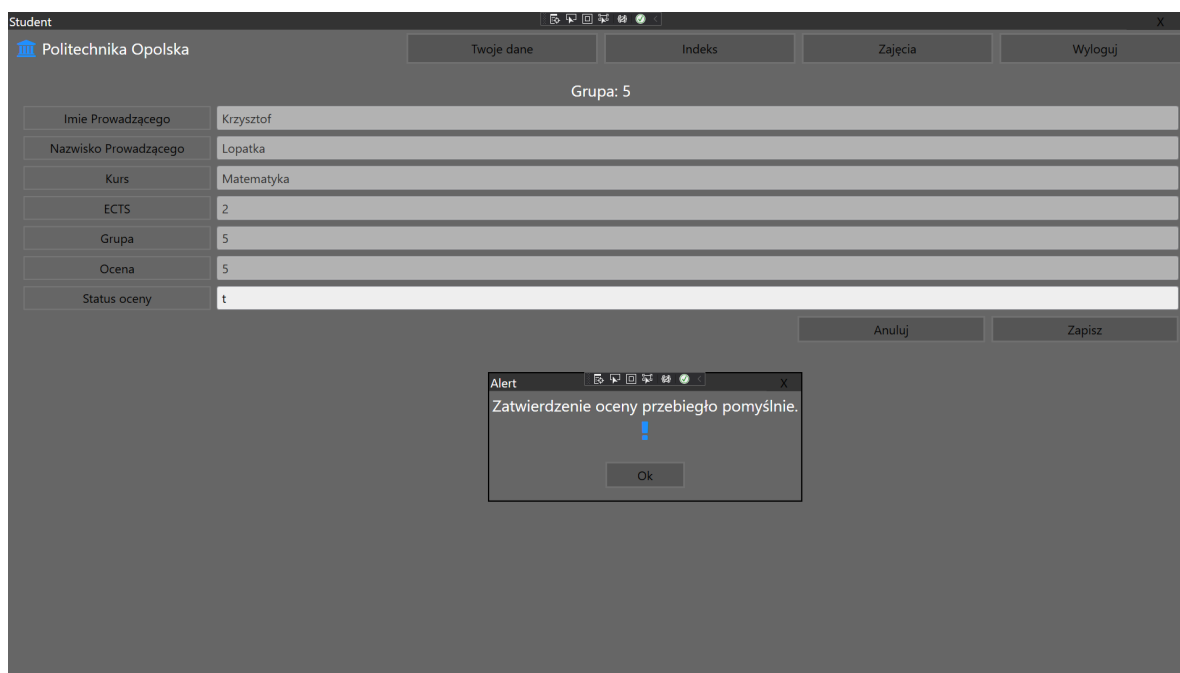
Numer Indeksu	240001
Kurs	Matematyka
Grupa	5
Typ zajęć	W
Ocena	5.0
Status oceny	w

At the bottom right of the form are buttons 'Anuluj' and 'Zapisz'. An 'Alert' dialog box is open in the center, displaying the message: 'Wprowadzenie oceny przebiegło pomyślnie.'. The dialog has an 'Ok' button.

Rysunek 58: Alert wprowadzenie oceny

## 6.4 Test zatwierdzenia oceny przez studenta

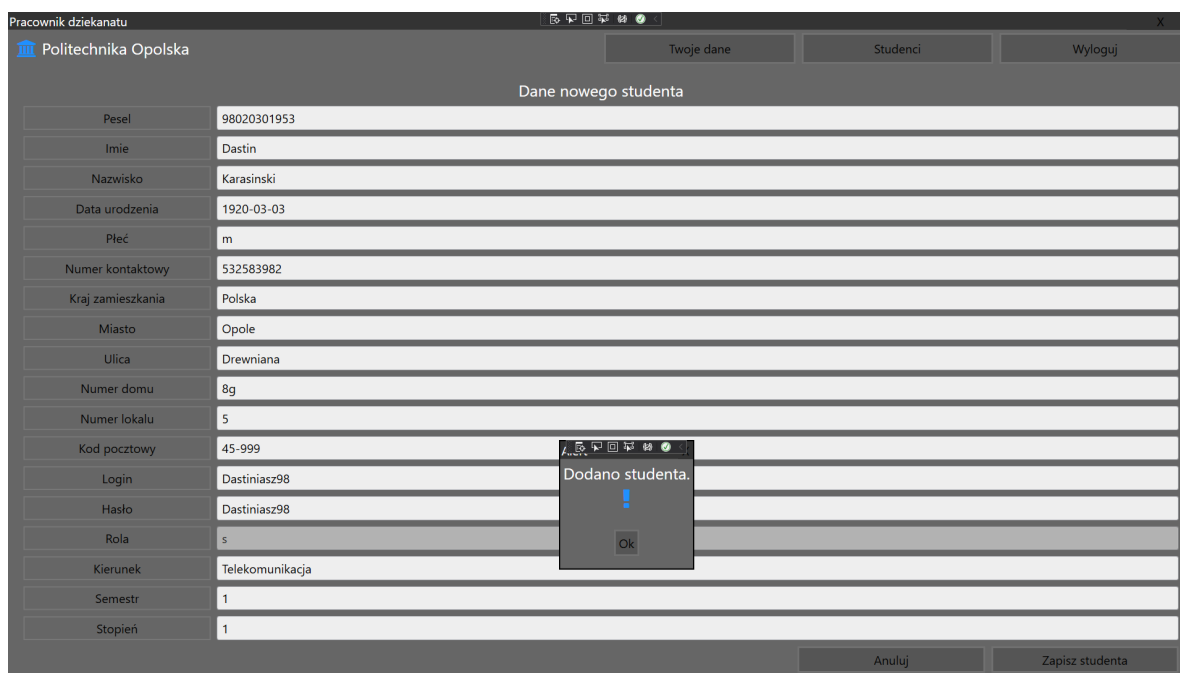
- Jeśli student spróbuje zatwierdzić ocenę zgodnie z poprawnością danych to pojawia się alert o poprawnej zmianie oceny. Dane te są zapisywane, także poprawnie w bazie danych.



Rysunek 59: Alert zatwierdzenie oceny

## 6.5 Dodanie studenta przez pracownika dziekanatu

- Jeśli pracownik dziekanatu spróbuje dodać użytkownika zgodnie z poprawnością danych to pojawia się alert o poprawnym dodaniu użytkownika. Dane te są zapisywane, także poprawnie w bazie danych.



Rysunek 60: Alert dodano studenta

- Jeśli pracownik dziekanatu spróbuje dodać użytkownika niezgodnie z poprawnością danych to pojawia się alert o niepoprawnym dodaniu użytkownika. Dane te nie są zapisywane w bazie danych.

The screenshot shows the 'Pracownik dziekanatu' application window. The title bar includes 'Politechnika Opolska', 'Twoje dane', 'Studenci', and 'Wyloguj'. The main content area is titled 'Dane nowego studenta' and contains a form with the following fields and values:

Label	Value
Pesel	98020301953
Imię	Dastin
Nazwisko	Karasinski
Data urodzenia	1920-03-03
Płeć	m
Numer kontaktowy	532583982
Kraj zamieszkania	Polska
Miasto	Opole
Ulica	Drewniana
Numer domu	8g
Numer lokalu	5
Kod pocztowy	45-999112
Login	Dastiniasz98
Hasło	Dastiniasz98
Rola	s
Kierunek	Telekomunikacja
Semestr	1
Stopień	1

An alert dialog box is overlaid on the form, displaying the message: 'Alert Kod pocztowy za długi. Maksymalna ilość znaków: 6'. The dialog has an 'Ok' button. At the bottom of the form, there are 'Anuluj' and 'Zapisz studenta' buttons.

Rysunek 61: Alert błędne dane

## 6.6 Dodanie użytkownika przez administratora

- Jeśli administrator spróbuje dodać użytkownika zgodnie z poprawnością danych to pojawia się alert o poprawnym dodaniu użytkownika. Dane te są zapisywane w bazie danych.

The screenshot shows the 'Administrator' application window. The title bar includes 'Politechnika Opolska', 'Twoje dane', 'Uzytkownicy', and 'Wyloguj'. The main content area is titled 'Dane nowego użytkownika' and contains a form with the following fields and values:

Label	Value
Pesel	98020301923
Imię	Ania
Nazwisko	Piernik
Data urodzenia	1989-03-01
Płeć	k
Numer kontaktowy	521654812
Kraj zamieszkania	Polska
Miasto	Wroclaw
Ulica	Walecznych
Numer domu	8d
Numer lokalu	6
Kod pocztowy	34-921
Login	Ania92
Hasło	Ania92
Rola	a
Kierunek	
Semestr	
Stopień	

An alert dialog box is overlaid on the form, displaying the message: 'Dodano użytkownika.'. The dialog has an 'Ok' button. At the bottom of the form, there are 'Anuluj' and 'Zapisz Użytkownika' buttons.

Rysunek 62: Alert dodano użytkownika

- Jeśli administrator spróbuje dodać użytkownika niezgodnie z poprawnością danych to pojawia się alert o niepoprawnym dodaniu użytkownika. Dane te są zapisywane w bazie danych.

The screenshot shows a web application window titled 'Administrator' with a sub-header 'Politechnika Opolska'. It features three tabs: 'Twoje dane', 'Użytkownicy', and 'Wyloguj'. The active tab is 'Użytkownicy', displaying the 'Dane nowego użytkownika' form. The form includes fields for Pesel, Imię, Nazwisko, Data urodzenia, Płeć, Numer kontaktowy, Kraj zamieszkania, Miasto, Ulica, Numer domu, Numer lokalu, Kod pocztowy, Login, Hasło, Rola, Kierunek, Semestr, and Stopień. An 'Alert' dialog box is open, showing the following text: 'Login powinien mieć odpowiedni format. Powinien zawierać dużą literę, małą literę i cyfrę. Na przykład: 'Login123''. The dialog has an 'Ok' button. At the bottom of the form are 'Anuluj' and 'Zapisz Użytkownika' buttons.

Pesel	98020301923
Imię	Ania
Nazwisko	Piernik
Data urodzenia	1989-03-01
Płeć	k
Numer kontaktowy	521654812
Kraj zamieszkania	Polska
Miasto	Wrocław
Ulica	Walecznych
Numer domu	8d
Numer lokalu	6
Kod pocztowy	34-921
Login	Ania
Hasło	Ania92
Rola	a
Kierunek	
Semestr	
Stopień	

Rysunek 63: Alert błędne dane



## 7 Dodatkowe mechanizmy

### 7.1 Wersjonowanie bazy danych

Wersjonowanie naszej bazy danych realizujemy za pomocą narzędzi Flyway oraz Spawn. Po pobraniu i instalacji obu programów, wymagana była ich odpowiednia konfiguracja.

- Pierwszym krokiem było stworzenie kontenera danych za pomocą programu Spawn.

```
C:\Users\patry>spawnctl create data-container --image mysql:flyway-getting-started --name elektroniczny_indeks --lifetime 2000h
Data container 'elektroniczny_indeks' created!
-> Server=instances.spawn.cc;Port=31282;User Id=root;Database=foobardb;Password=sCPvNtKk1zLwmfLc
```


Rysunek 64: Tworzenie kontenera danych

- Następnie wyświetlamy dane naszego kontenera, ponieważ za chwilę będą nam potrzebne.

```
C:\Users\patry>spawnctl get data-container elektroniczny_indeks -o yaml
id: 97816
imageid: 12976
name: elektroniczny_indeks
revision: rev.0
status: 2
engine: MySQL
engineversion: "5.7"
statusmessage: Running
connectionstring: Server=instances.spawn.cc;Port=31282;User Id=root;Database=foobardb;Password=sCPvNtKk1zLwmfLc
host: instances.spawn.cc
port: 31282
user: root
password: sCPvNtKk1zLwmfLc
createdat: 1621514257
expiresat: 1628714277
owner: ""
```

Rysunek 65: Wyświetlenie danych utworzonego kontenera

- Mając powyższe dane możemy skonfigurować narzędzie Flyway pod nasz kontener danych. Wprowadzamy dane do pliku o nazwie flyway.conf. Edytujemy poniższe linijki zgodnie z wygenerowanymi danymi z aplikacji Spawn. Wprowadzamy login, hasło, hosta, port, nazwę bazy danych oraz ścieżkę do folderu w którym będziemy umieszczać pliki z poleceniami SQL.

 flyway.conf	20.05.2021 14:45	Plik CONF	23 KB
---	------------------	-----------	-------

Rysunek 66: Plik konfiguracyjny Flyway

```
flyway.url=jdbc:mysql://instances.spawn.cc:31282/elektroniczny_indeks
```

Rysunek 67: Flyway url

```
flyway.user=root
```

Rysunek 68: Flyway user



`flyway.password=sCPvNtKk1zLwmfLc`

Rysunek 69: Flyway password

`flyway.locations=filesystem:C:\Users\patry\flyway-7.9.1\sql`

Rysunek 70: Flyway location

- Kolejnym krokiem jest wklejenie odpowiednio nazwanych plików z kodem SQL do podanej przez nas ścieżki.

 V1_Create_all_tables.sql	20.05.2021 14:35	SQL Text File	68 KB
 V2_Create_data.sql	20.05.2021 14:30	SQL Text File	9 KB

Rysunek 71: Pliki SQL

- Teraz korzystając z narzędzia Flyway można wywołać pliki SQL, które stworzą nam pierwszą wersję bazy danych oraz wygenerują dane testowe.

```
C:\Users\patry>flyway migrate
Flyway Community Edition 7.9.1 by Redgate
Database: jdbc:mysql://instances.spawn.cc:31282/foobardb (MySQL 5.7)
Successfully validated 2 migrations (execution time 00:00.497s)
Creating Schema History table 'foobardb`.`flyway_schema_history` ...
Current version of schema 'foobardb': << Empty Schema >>
Migrating schema 'foobardb' to version '1 - Create all tables'
WARNING: DB: Changing sql mode 'NO_AUTO_CREATE_USER' is deprecated. It will be removed in a future release. (SQL State: HY000 - Error Code: 3090)
WARNING: DB: Changing sql mode 'NO_AUTO_CREATE_USER' is deprecated. It will be removed in a future release. (SQL State: HY000 - Error Code: 3090)
Migrating schema 'foobardb' to version '2 - Create data'
Successfully applied 2 migrations to schema 'foobardb', now at version v2 (execution time 01:00.678s)
```

Rysunek 72: Tworzenie pierwszej wersji aplikacji

- Tym samym nasza baza jest gotowa do użytku. Ostatnim krokiem jest odpowiednia konfiguracja połączenia w naszej aplikacji dostępowej.

```
this.connectionString = @"Server = instances.spawn.cc; Port = 31282; User Id = root; Database = elektroniczny_indeks; Password = sCPvNtKk1zLwmfLc; Pooling=True;";
```

Rysunek 73: Konfiguracja połączenia w aplikacji dostępowej

## 7.2 Disaster recovery

Aby zabezpieczyć naszą bazę danych przed utratą jakichkolwiek danych, stworzyliśmy dwa krótkie skrypty w pythonie, które umożliwiają tworzenia backupa naszej bazy oraz przywrócenie jej ze stworzonego backupa.

```
1 import os
2 os.system('cmd /c "mysqldump --user root --password=root elektroniczny_indeks > elektroniczny_indeks.sql"')
```

Rysunek 74: Backup

```
import os
os.system('cmd /c "mysql --user root --password=root elektroniczny_indeks < elektroniczny_indeks.sql"')
```

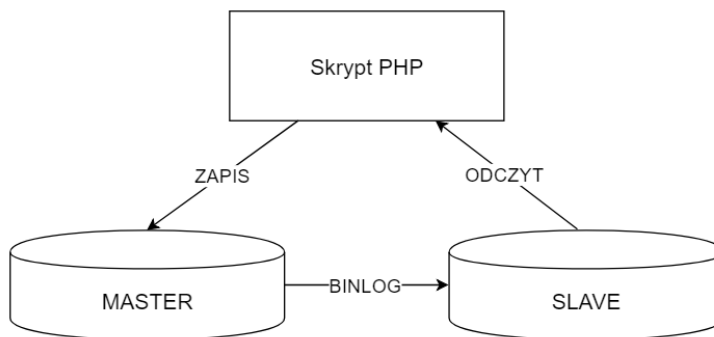
Rysunek 75: Recovery

## 7.3 Skalowanie bazy danych

Domyślnie nasza aplikacja (Indeks Elektroniczny Uczelni Wyższej) jest przewidziana dla jednego wydziału uczelni. Gdy pojawia się problem rozbudowy systemu i potrzeba skalowania bazy danych, warto rozważyć kilka możliwości. Główną decyzją do podjęcia to wybór skalowania pionowego lub poziomego i poniesienia konsekwencji związanych z tą decyzją. W naszym przypadku skalując bazę danych, wykorzystalibyśmy skalowanie poziome. Zakładając sytuację w której nasza aplikacja wraz z bazą danych miałyby obsługiwać nie tylko jeden wydział, a całą uczelnię, najwygodniejszą opcją będzie stworzenie kilku baz danych (jednej dla każdego wydziału) nie narażając się w ten sposób na ogromne koszty wynikające z wymianom sprzętu który musiałby być dostosowany do jednej, ogromnej bazy. Kolejnym argumentem przemawiającym przeciw skalowaniu pionowemu jest powolne działanie spowodowane ogromnym rozmiarem bazy danych oraz potrzebą partycjonowania bazy przy przekroczeniu maksymalnego rozmiaru pliku dla danego systemu plików w których nasza baza jest przechowywana. Dodatkowym atutem wybranego przez nas rozwiązania będzie szybszy dostęp do danych spowodowany przeglądaniem nie jednej, ogromnej bazy danych, a jednej, mniejszej bazy danych, zawierającej informacje tylko odnośnie wybranego wydziału. Pomysł taki wymagałby oczywiście zmian w aplikacji dostępowej (dodanie pola wyboru wydziału do którego należymy), jednak taka zmiana byłaby mniej kosztowna w porównaniu do konserwacji i utrzymywania pojedynczej, dużej bazy danych. Oprócz tego warto wspomnieć o zwiększeniu bezpieczeństwa i zmniejszeniu awaryjności poszczególnych baz danych - w razie awarii bazy danych odpowiadającej za jeden wydział, pozostałe dalej mogą pracować.

## 7.4 Replikacja bazy danych

Domyślnie nasza aplikacja (Indeks Elektroniczny Uczelni Wyższej) jest przewidziana dla jednego wydziału uczelni. Gdyby pojawiła się sytuacja, w której baza danych nie posiadałaby wystarczającej liczby zasobów, aby sprostać nadysłanym zapytaniom to w takim przypadku zastosowalibyśmy replikację bazy danych typu Master-Slave. Replikacja polegałaby na stworzeniu różnorodnych serwerów docelowych dla zapytań SQL. Nasz dotychczasowy serwer staje się Masterem, a pozostałe będą jego poddanymi – Slave. Replikacja nie polega na wykonywaniu ciągłych kopii zapasowych i wczytywaniu ich do kolejnego serwera. Jest to asynchroniczne powielenie wszystkich operacji jakie znajdują się w dzienniku zdarzeń jednego serwera - w tym przypadku Mastera. W takim wypadku jest możliwy podział, Master dokonuje operacji związanych z zapisem danych do bazy danych, natomiast Slave dokonuje operacje odczytu. Proces ten przedstawia poniższy schemat.



Rysunek 76: Replikacja bazy danych

Zapisy niemal we wszystkich systemach są bardziej „zasobożerne” i powodują blokady. Natomiast odczyt z bazy danych powinien być szybki, dlatego takie rozwiązanie jest bardzo opłacalne.

## 7.5 Zabezpieczenie przed SQL injection

Testowaliśmy czy nasza aplikacja jest podatna na ataki SQL injection. Dzięki skrupulatnej walidacji danych, niedopuszczaniu spacji oraz znaków specjalnych typu \*, użytkownik nie ma możliwości przeprowadzenia ataku typu SQL injection. Nie było więc potrzeby implementowania dodatkowych zabezpieczeń tego typu.

## 7.6 Strategia aktualizacji bazy danych

Gdy znajdzie konieczność aktualizacji naszej bazy danych o nowe tabele lub pojedyncze rekordy w już istniejących tabelach, należy podjąć odpowiednie kroki, aby aktualizując bazę nie uszkodzić danych znajdujących się wewnątrz. Pierwszym i najważniejszym krokiem przy takiej operacji będzie wykorzystanie napisanego przez nas skryptu do utworzenia kopii zapasowej bazy danych. Następnie przygotujemy polecenia SQL które dodadzą tabelę, lub pojedyncze rekordy w odpowiednich miejscach. Wypełnianie nowych tabel/rekordów odbędzie się na podstawie relacji poszczególnych tabel lub w przypadku rekordów, pozostałych danych w tabeli. Do aktualizacji bazy danych wykorzystane zostanie narzędzie Flyway, które umożliwia wersjonowanie i umożliwia powrót do poprzednich wersji.

## 8 Podsumowanie i wnioski

- Projekt został zrealizowany, wszystkie założenia zostały spełnione. Zarówno baza danych jak i aplikacja dostępowa oraz dodatkowe mechanizmy umożliwiające łatwiejszą edycję w przyszłości zostały połączone w jedną całość. Dając jako efekt poprawnie działający Indeks Elektroniczny Uczelni Wyższej.
- Projekt bazy danych i aplikacji dostępowej za pomocą modelu konceptualnego, fizycznego i logicznego oraz makiety aplikacji pozwala na łatwiejszą implementację bazy danych oraz aplikacji. Dodatkowo pozwala na wcześniejsze wychwycenie błędów lub całkowite ich uniknięcie.
- Środowisko VisualStudio i język C# sprawdziły się doskonale do realizacji tego typu projektu. Zarówno stosunkowo łatwo implementuje się front-end jak i back-end aplikacji. W prosty sposób można wygenerować plik wykonywalny .exe i rozpowszechniać naszą aplikację.
- Repozytorium: <https://github.com/patrykwieczorek03/IndeksElektronicznyNew>

## Spis rysunków

1	Model konceptualny . . . . .	5
2	Model logiczny . . . . .	6
3	Model fizyczny . . . . .	7
4	Diagram uprawnień . . . . .	15
5	Diagram przypadków użycia . . . . .	16
6	Projekt interfejsu graficznego . . . . .	17
7	Model bazy danych z tabelami, widokami oraz procedurami składowymi . . . . .	37
8	Testy procedury w bazie danych związanej z zabezpieczeniami . . . . .	38
9	Testy procedury w bazie danych związanej z zabezpieczeniami . . . . .	39
10	Testy procedur w bazie danych związanych z dodawaniem rekordów . . . . .	40
11	Testy procedur w bazie danych związanych z dodawaniem rekordów . . . . .	41
12	Testy procedur w bazie danych związanych z wprowadzaniem ocen . . . . .	42
13	Testy procedur w bazie danych związanych z reklamowaniem ocen . . . . .	42
14	Testy procedur w bazie danych związanych z zatwierdzaniem ocen . . . . .	42
15	Testy procedur w bazie danych związanych z odrzucaniem reklamacji . . . . .	42
16	Testy procedur w bazie danych związanych z poprawianiem ocen . . . . .	43
17	Testy procedur w bazie danych związanych z usuwaniem użytkowników . . . . .	43
18	Testy procedur w bazie danych związanych z widokami . . . . .	45
19	Testy widoków w bazie danych . . . . .	47
20	Testy widoków w bazie danych . . . . .	48
21	Tabela zależności czasu tworzenia próbnej bazy danych od ilości tworzonych użytkowników . . . . .	50
22	Wykres zależności czasu tworzenia próbnej bazy danych od ilości tworzonych użytkowników . . . . .	50
23	Tabela zależności czasu wykonania procedury "dodaj_uzytkownika" od ilości istniejących użytkowników . . . . .	51
24	Wykres zależności czasu wykonania procedury "dodaj_uzytkownika" od ilości istniejących użytkowników . . . . .	51
25	Folder z programem . . . . .	52
26	Pliki instalacyjne . . . . .	52
27	Ekran logowania . . . . .	52
28	Ekran komunikatu studenta . . . . .	53
29	Ekran danych osobowych studenta . . . . .	53
30	Ekran edycji danych osobowych studenta . . . . .	54
31	Ekran danych studenta . . . . .	54
32	Ekran edycji indeksu studenta . . . . .	55
33	Ekran przeglądania grup zajęciowych . . . . .	55
34	Ekran planu zajęć studenta . . . . .	56
35	Ekran logowania . . . . .	57
36	Ekran komunikatu prowadzącego . . . . .	57
37	Ekran danych osobowych prowadzącego . . . . .	58
38	Ekran edycji danych osobowych prowadzącego . . . . .	58
39	Ekran prowadzonych kursów prowadzącego . . . . .	59
40	Ekran prowadzonych grup kursów prowadzącego . . . . .	59
41	Ekran studentów prowadzącego . . . . .	60
42	Ekran studentów prowadzącego . . . . .	60
43	Ekran logowania . . . . .	61
44	Ekran komunikatu pracownika dziekanatu . . . . .	61
45	Ekran danych osobowych pracownika dziekanatu . . . . .	62
46	Ekran edycji danych osobowych pracownika dziekanatu . . . . .	62
47	Ekran listy studentów dziekanatu . . . . .	63
48	Ekran studenta dziekanatu . . . . .	63
49	Ekran logowania . . . . .	64
50	Ekran komunikatu administratora . . . . .	64
51	Ekran danych osobowych administratora . . . . .	65
52	Ekran edycji administratora dziekanatu . . . . .	65
53	Ekran listy użytkowników administratora . . . . .	66
54	Ekran użytkownika administratora . . . . .	66
55	Alert błędne dane logowania . . . . .	67
56	Alert zmiana danych przebiegła pomyślnie . . . . .	67
57	Alert niepoprawna ilość znaków . . . . .	68

58	Alert wprowadzenie oceny . . . . .	68
59	Alert zatwierdzenie oceny . . . . .	69
60	Alert dodano studenta . . . . .	69
61	Alert błędne dane . . . . .	70
62	Alert dodano użytkownika . . . . .	70
63	Alert błędne dane . . . . .	71
64	Tworzenie kontenera danych . . . . .	72
65	Wyświetlenie danych utworzonego kontenera . . . . .	72
66	Plik konfiguracyjny Flyway . . . . .	72
67	Flyway url . . . . .	72
68	Flyway user . . . . .	72
69	Flyway password . . . . .	73
70	Flyway location . . . . .	73
71	Pliki SQL . . . . .	73
72	Tworzenie pierwszej wersji aplikacji . . . . .	73
73	Konfiguracja połączenia w aplikacji dostępowej . . . . .	73
74	Backup . . . . .	73
75	Recovery . . . . .	73
76	Replikacja bazy danych . . . . .	74