

# Aplikacja Administracji Polski

Przedmiot: Bazy Danych 2 Prowadzący: dr inż. Roman Ptak

Nr grupy	1
Skład grupy	Jan Zieniewicz, Adrian Kotula, Jędrzej Koba
Termin	Pt 11:15 rok akademicki 2023/34
Temat projektu	„Podział administracyjny Polski”

## Aplikacja "Podział administracyjny RP"

**Cel:** Stworzenie aplikacji ułatwiającej odczytywanie danych administracyjnych

### Opis:

Stworzony przez nas system przedstawiający podział administracyjny Rzeczypospolitej Polskiej składa się z aplikacji dostępowej oraz bazy danych.

Baza danych składa się z trzech encji odpowiadającym stopniom jednostek administracyjnych w Polsce- województwa (1 stopień), powiatu (2stopień) oraz gmin (3 stopień).

**Encja województw przechowuje dane dot. odpowiednich województw takich jak:**

- nazwa
- niepowtarzalny numer id- teryt
- powierzchnia
- ludność
- miasto wojewódzkie
- współrzędne miasta wojewódzkiego
- inne dane

**Encja powiatów przechowuje dane dot. odpowiednich powiatów takich jak:**

- nazwa
- niepowtarzalny numer idv
- powierzchnia
- ludność
- typ powiatu- miasto na prawie powiatu etc.
- stolica powiatu
- współrzędne stolicy powiatu
- województwo w którym się znajduje
- inne dane

**Encja gmin przechowuje dane dot. odpowiednich gmin takich jak:**

- nazwa
- niepowtarzalny numer id- teryt
- powierzchnia
- ludność
- typ gminy- miejska/wiejska/wiejsko-miejska
- stolica gminy
- współrzędne stolicy gminy
- powiat, w którym się znajduje
- inne dane

**Z systemu będą korzystać: odbiorcy danych (pracownicy urzędów), administratorzy danych oraz administratorzy IT.**

**Odbiorca danych:**

- wyświetlanie danych o województwach
- wyświetlanie danych o powiatach/ gminach na podstawie danych dotyczących województw
- wyświetlanie danych o powiatach
- wyświetlanie danych o województwach/ gminach na podstawie danych dot. powiatów
- wyświetlanie danych o gminach
- wyświetlanie danych o województwach/ powiatach na podstawie danych dot. gmin
- wyświetlanie danych spełniających warunki zadane przez użytkownika
- składanie wniosku o edycję istniejących danych lub dodanie nowych

**Administrator danych:**

- administrator danych posiada dostęp do takich samych funkcjonalności co odbiorca danych
- dodawanie nowych województw
- usuwanie województw
- modyfikowanie danych podstawowych województwa
- modyfikowanie danych statystycznych województwa
- modyfikowanie danych administracyjnych województwa
- modyfikowanie danych geolokalizacyjnych województwa
- wyszukiwanie województw
- dodawanie nowych powiatów
- usuwanie powiatu
- modyfikowanie danych podstawowych powiatu
- modyfikowanie danych statystycznych powiatu
- modyfikowanie danych administracyjnych powiatu
- modyfikowanie danych geolokalizacyjnych powiatu
- wyszukiwanie powiatów

- dodawanie nowych gmin
- usuwanie gmin
- modyfikowanie danych podstawowych gminy
- modyfikowanie danych statystycznych gminy
- modyfikowanie danych administracyjnych gminy
- modyfikowanie danych geolokalizacyjnych gminy
- wyszukiwanie gmin
- zapisywanie danych
- rozpatrywanie wniosków o dodanie/zmianę/usunięcie danych złożonych przez użytkownika

**Administrator IT:**

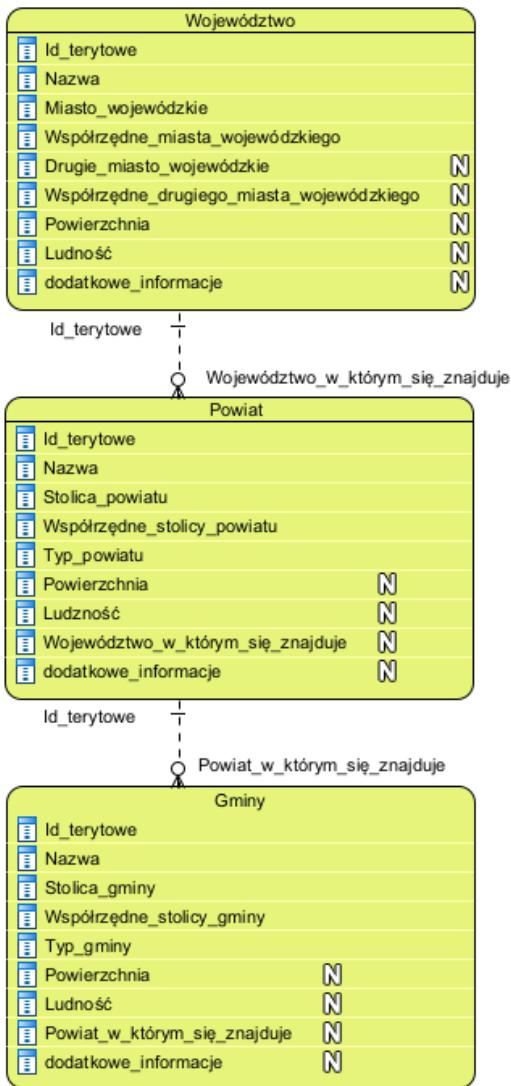
- administrator IT ma dostęp do takich samych funkcjonalności co administrator danych oraz odbiorca danych
- konfiguracja zasad tworzenia automatycznej kopii zapasowej
- tworzenie kont dla nowych użytkowników
- zarządzanie bazą danych

**Cechy niefunkcjonalne:**

- Aplikacja dostępowża do bazy danych będzie w formie aplikacji webowej.
- Aplikacja dostępowża będzie napisana w Python3 z wykorzystaniem framework'u Django.
- Aplikacja będzie posiadać intuicyjny interfejs użytkownika.
- Aplikacja będzie działać na przeglądarkach dostępnych na systemach: Windows, Linux, macOS.
- Baza danych będzie wykorzystywać technologię MySQL.
- Stabilność zapewniają testy manualne dla aplikacji Django.
- System gwarantuje integralność danych, poprzez weryfikację użytkowników.
- System obsługuje 300 zapytań na sekundę
- System jest ciągle dostępny z wyjątkiem czasu, w którym prowadzone są prace konserwacyjne.
- Prace konserwacyjne odbywają się w śródę w godzinach 11-16.
- Zapewniamy całodobowe wsparcie techniczne.
- Monitorujemy użycie zasobów.
- największa encja (gmin) ma około 2500 rekordów.

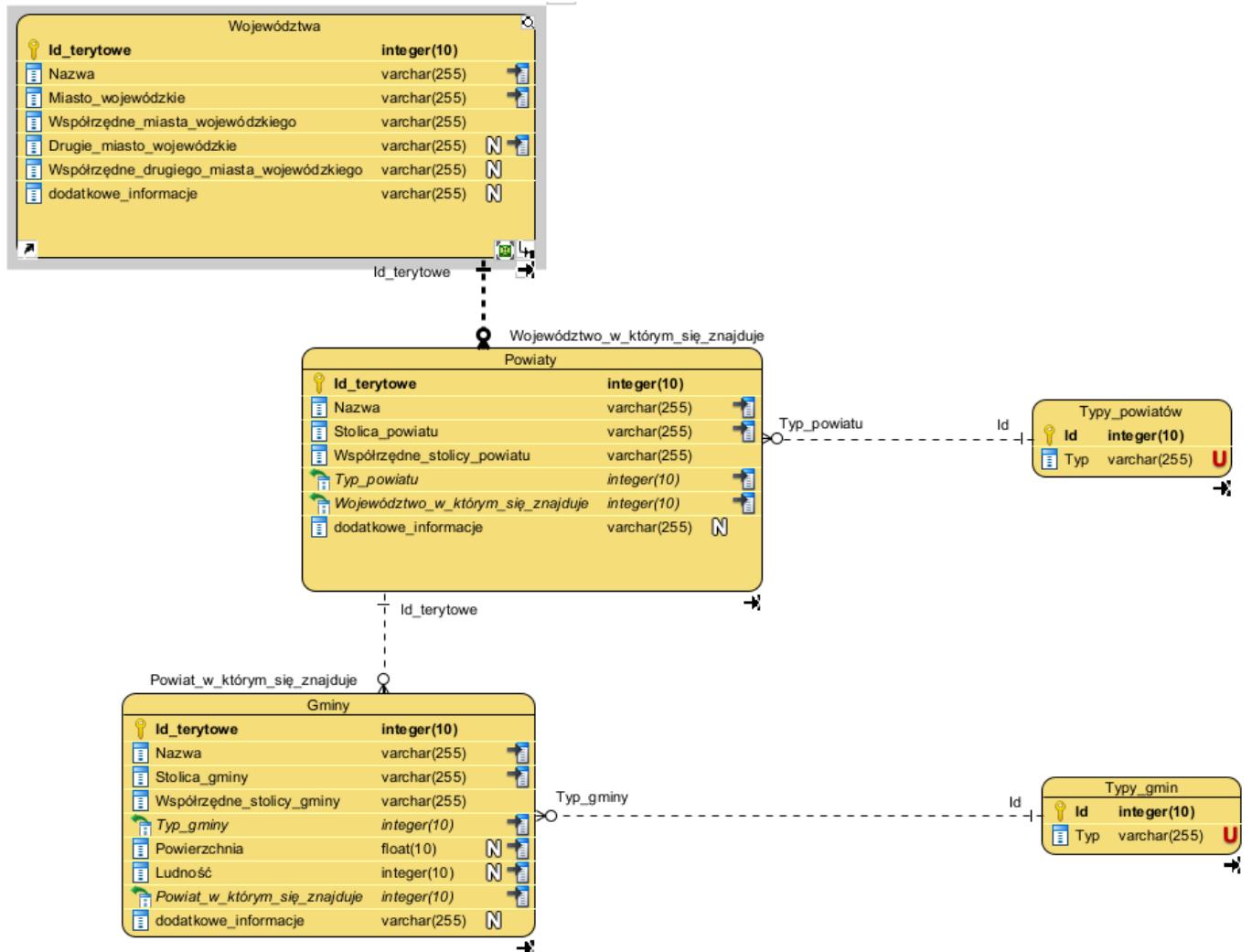
# Projekt Bazy Danych

## Model konceptualny



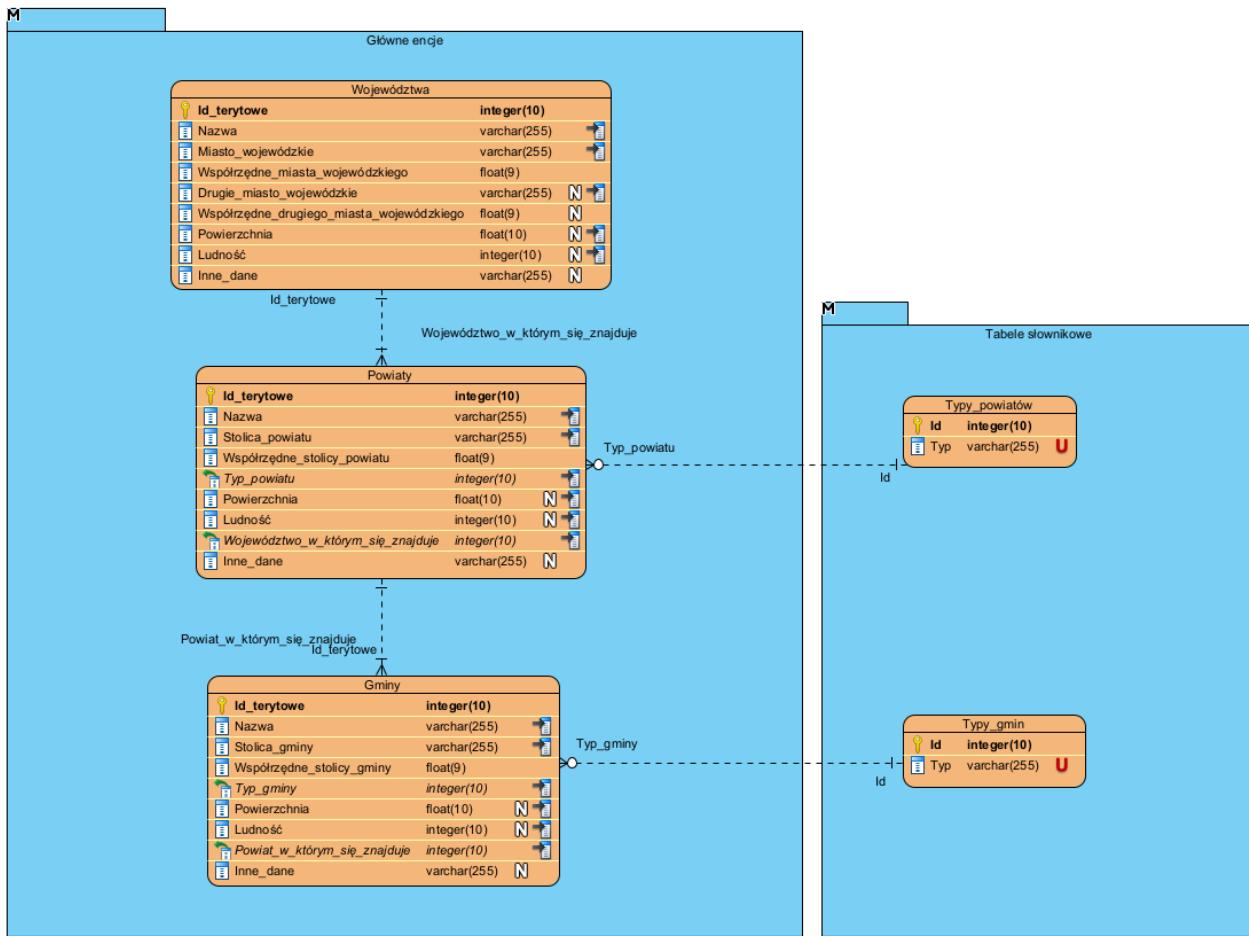
Nasz model odwzorowuje aktualny podział administracyjny Rzeczypospolitej Polski. Każda z tabel odwzorowuje odpowiadającą jej (tabeli) jednostkę administracyjną.

## Model logiczny



Zdecydowaliśmy się na dodanie tabel słownikowych w celu ograniczenia powtórzeń i ułatwienia zarządzania bazą danych.

## Model fizyczny



## Inne elementy bazy danych

Baza danych przechowuje tylko aktualny podział administracyjny Polski. Nie przechowuje ona historii edycji.

Wszystkie parametry tabel do jakich się odnosimy poniżej, takie jak "nazwa [nazwa jednostki administracyjnej]", "ludność [nazwa jednostki administracyjnej]" etc. Mają w swojej nazwie nazwę tabeli. To znaczy, że kiedy odwołujemy się do nazwy gminy, parametr "nazwa gminy" bierzemy z tabeli gminy. Nazwa jednostki administracyjnej jest nazwą tabeli.

Poniżej może pojawić się zapis: "nazwa, stolica, ludność [nazwa jednostki administracyjnej]", w takim wypadku wszystkie parametry odnoszą się do tabeli wskazanej na końcu. Taki zapis poprawia czytelność i zwiększa przejrzystość dokumentacji.

### Indeksy pojedyncze:

- Nazwa województwa (tab. wojewodztwa)
- Nazwa powiatu (tab. powiaty)
- Nazwa gminy (tab. gminy)

- Ludność województwa (tab. wojewodztwa)
- Ludność powiatu (tab. powiaty)
- Ludność gminy (tab. gminy)
- Powierzchnia województwa (tab. wojewodztwa)
- Powierzchnia powiatu(tab. powiaty)
- Powierzchnia gminy(tab. gminy)
- Miasto wojewódzkie (tab. wojewodztwa)
- Drugie miasto wojewódzkie (tab. wojewodztwa)
- Stolica powiatu (tab. powiaty)
- Stolica gminy(tab. gminy)
- Id terytoryjne województwa (tab. wojewodztwa)
- Id terytoryjne powiatu (tab. powiaty)
- Id terytoryjne gminy (tab. gminy)

**Indeksy podwójne:**

- Nazwa województwa, stolica województwa(tab. wojewodztwa)
- Nazwa powiatu, stolica powiatu (tab. powiaty)
- Nazwa gminy, stolica gminy (tab. gminy)
- Nazwa powiatu, typ powiatu (tab. powiaty)
- Nazwa gminy, typ gminy (tab. gminy)

**Widoki:**

- Widok pokazujący nazwę województwa, stolicę województwa, współrzędne stolicy województwa, możliwej drugiej stolicy województwa i powierzchnię województwa
- Widok pokazujący nazwę powiatu, stolicę powiatu, współrzędne stolicy powiatu i powierzchnię powiatu
- Widok pokazujący nazwę gminy, stolicę gminy, współrzędne stolicy gminy i powierzchnię gminy
- Widok pokazujący nazwę gminy, stolicę gminy, typ gminy, to w jakim powiecie gmina się znajduje (nazwa powiatu) i w jakim województwie gmina się znajduje (nazwa województwa)
- Widok pokazujący nazwę powiatu, stolicę powiatu, typ powiatu, to w jakim województwie (nazwa województwa) powiat się znajduje i jakie gminy (nazwy gmin) znajdują się w tym powiecie
- Widok pokazujący nazwę województwa, stolicę województwa i możliwą drugą stolicę województwa, to jakie powiaty są w tym województwie (nazwy powiatów) i jakie gminy są w tym województwie (nazwy gmin)
- Widok pokazujący nazwę województwa, stolicę województwa, powierzchnię województwa, ludność województwa i inne dane województwa
- Widok pokazujący nazwę powiatu, stolicę powiatu, powierzchnię powiatu, ludność powiatu i inne dane powiatu
- Widok pokazujący nazwę gminy, stolicę gminy, powierzchnię gminy, ludność gminy i inne dane gminy

**Poziomy uprawnień:**

- Użytkownik podstawowy, może tylko odczytywać dane z bazy (dostęp do tablicy województw, powiatów i gmin typu read only).
- Administrator danych, może edytować informacje, które zawiera baza danych (dostęp do całej bazy, edycja danych, dodawanie rekordów, usuwanie ich)
- Administrator aplikacji, może zarządzać bazą danych jak i tworzyć wymagane konta użytkowników, i wykonywać kopie zapasowe bazy danych

**Procedury składowane:**

- Procedura realizująca dodawanie nowych województw do bazy danych
- Procedura realizująca dodawanie nowych powiatów do bazy danych
- Procedura realizująca dodawanie nowych gmin do bazy danych
- Procedura pozwalająca na edycję i aktualizację danych województw
- Procedura pozwalająca na edycję i aktualizację danych powiatów
- Procedura pozwalająca na edycję i aktualizację danych gmin
- Procedura pozwalająca na usuwanie województw (cascade delete)
- Procedura pozwalająca na usuwanie powiatów (cascade delete)
- Procedura pozwalająca na usuwanie gmin
- Procedura pozwalająca na wyszukiwanie województwa (nazwa województwa i powierzchnia województwa) według powierzchni z sortowaniem malejąco
- Procedura pozwalająca na wyszukiwanie powiatu (nazwa powiatu, powierzchnia powiatu) według powierzchni z sortowaniem malejąco
- Procedura pozwalająca na wyszukiwanie gmin (nazwa gminy, powierzchnia gminy) według powierzchni z sortowaniem malejąco
- Procedura pozwalająca na wyszukiwanie województwa (nazwa województwa, powierzchnia województwa) według powierzchni z sortowaniem rosnąco
- Procedura pozwalająca na wyszukiwanie powiatu (nazwa powiatu, powierzchnia powiatu) według powierzchni z sortowaniem rosnąco
- Procedura pozwalająca na wyszukiwanie gmin (nazwa gminy, powierzchnia gminy) według powierzchni z sortowaniem rosnąco
- Procedura pozwalająca na wyszukiwanie województwa (nazwa województwa, liczba ludności) według ludności z sortowaniem malejąco
- Procedura pozwalająca na wyszukiwanie powiatu (nazwa powiatu, liczba ludności) według ludności z sortowaniem malejąco
- Procedura pozwalająca na wyszukiwanie gmin (nazwa gminy, liczba ludności) według ludności z sortowaniem malejąco Procedura pozwalająca na wyszukiwanie województwa (nazwa województwa, liczba ludności) według ludności z sortowaniem rosnąco
- Procedura pozwalająca na wyszukiwanie powiatu (nazwa powiatu, liczba ludności) według ludności z sortowaniem rosnąco
- Procedura pozwalająca na wyszukiwanie gmin (nazwa gminy, liczba ludności) według ludności z sortowaniem rosnąco
- Procedura pozwalająca na wyszukiwanie województwa (nazwa województwa) według nazwy z sortowaniem alfabetycznym A-Z

- Procedura pozwalająca na wyszukiwanie powiatu (nazwa powiatu) według nazwy z sortowaniem alfabetycznym A-Z
- Procedura pozwalająca na wyszukiwanie gmin (nazwa gminy) według nazwy z sortowaniem alfabetycznym A-Z
- Procedura pozwalająca na wyszukiwanie województwa (nazwa województwa, stolica województwa) według stolic województw z sortowaniem alfabetycznym A-Z
- Procedura pozwalająca na wyszukiwanie powiatu (nazwa powiatu, stolica powiatu) według stolicy powiatu z sortowaniem alfabetycznym A-Z
- Procedura pozwalająca na wyszukiwanie gmin (nazwa gminy, stolica gminy) według stolicy gminy z sortowaniem alfabetycznym A-Z

**Triggery:**

- Trigger after insert obliczający aktualną liczbę ludności w powiecie po dodaniu gminy do powiatu
- Trigger after delete obliczający aktualną liczbę ludności w powiecie po usunięciu gminy z powiatu
- Trigger after update obliczający aktualną liczbę ludności w powiecie po edycji danych gminy ludności w powiecie
- Trigger after insert obliczający aktualną liczbę ludności w województwie po dodaniu powiatu do województwa
- Trigger after delete obliczający aktualną liczbę ludności w województwie po usunięciu powiatu z województwa
- Trigger after update obliczający aktualną liczbę ludności w województwie po edycji danych ludności powiatu w województwie
- Trigger after insert obliczający aktualną powierzchnię województwa po dodaniu powiatu do województwa
- Trigger after delete obliczający aktualną powierzchnię województwa po usunięciu powiatu z województwa
- Trigger after update obliczający aktualną powierzchnię województwa po edycji danych powierzchni powiatu w województwie
- Trigger after insert obliczający aktualną powierzchnię powiatu po dodaniu gminy do powiatu
- Trigger after delete obliczający aktualną powierzchnię powiatu po usunięciu gminy z powiatu
- Trigger after update obliczający aktualną powierzchnię powiatu po edycji danych powierzchni gminy w powiecie

**Cascade:**

- Cascade delete używany przy usuwaniu województwa (usuwa także przypisane do niego powiaty i gminy)
- Cascade delete używany przy usuwaniu powiatu (usuwa także przypisane do niego gminy)

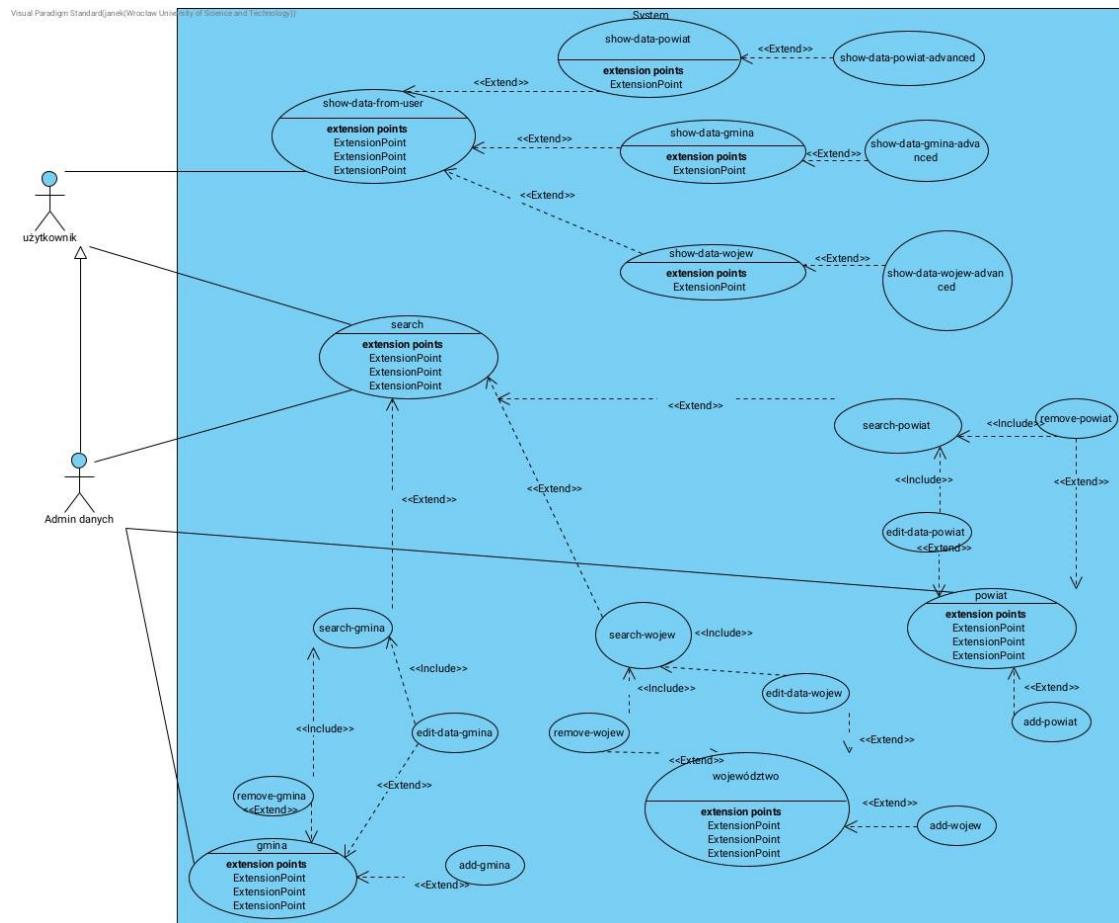
## Dodawanie danych:

Przy dodawaniu nowego rekordu za pomocą operacji check następuje sprawdzenie czy liczba jest nieujemna (liczba ludności) lub dodatnia (powierzchnia), np.:

- Przy dodaniu województwa:
  - check powierzchnia > 0
- Przy dodaniu powiatu:
  - check powierzchnia > 0
- Przy dodaniu gminy:
  - check powierzchnia > 0
  - check ludność >= 0

## Projekt Aplikacji użytkownika

### Architektura aplikacji i diagramy projektowe



Powyższy schemat prezentuje przypadki użycia w naszej aplikacji oraz funkcje dostępne dla użytkowników aplikacji.

## **Metoda podłączenia do bazy:**

Aplikacja do podłączenia do bazy danych będzie wykorzystywać wbudowaną funkcję frameworku "Django". Korzystanie z tej funkcji będzie gwarantować bezpieczeństwo oraz niezawodność połączenia.

## **Projekt zabezpieczeń na poziomie abstrakcji:**

Zabezpieczenie na poziomie abstrakcji jest realizowane poprzez:

- logowanie użytkowników z prawami edycji
- minimalna długość hasła
- przygotowane funkcje dodawania, usuwania oraz edycji danych uniemożliwiające podanie danych w złym formacie

## **Implementacja bazy danych**

Zgodnie z dokumentacją stworzoną na poprzednich etapach i zatwierdzoną przez prowadzącego, zgodnie z wymaganiami stworzyliśmy skrypty implementujące naszą bazę danych w MYSQL 8.

## **Skrypty:**

### **Schemat i tabele:**

```
drop database if exists administracjaRP;
create database administracjaRP;
use administracjaRP;

CREATE TABLE Gminy (Id_terytowe int(10) NOT NULL AUTO_INCREMENT,
                     Nazwa varchar(50) NOT NULL,
                     Stolica_gminy varchar(50) NOT NULL,
                     Wspolrzedne_stolicy_x float NOT NULL,
                     Wspolrzedne_stolicy_y float NOT NULL,
                     Powierzchnia float,
                     Ludnosc int(10),
                     Inne_dane varchar(10000),
                     Powiat_w_ktorym_sie_znajduje int(10) NOT NULL,
                     Typ_gminy int(10) NOT NULL, PRIMARY KEY (Id_terytowe),
                     INDEX (Nazwa), INDEX (Stolica_gminy),
                     INDEX (Powierzchnia),
                     INDEX (Ludnosc), INDEX (Powiat_w_ktorym_sie_znajduje),
                     INDEX (Typ_gminy),
                     CHECK (ludnosc >= 0 AND powierzchnia > 0));

CREATE TABLE Typ_gminy (Id int(10) NOT NULL AUTO_INCREMENT, Typ varchar(50) NOT NULL UNIQUE, PRIMARY KEY (Id));
```

```

CREATE TABLE Powiaty (Id_terytowe int(10) NOT NULL AUTO_INCREMENT,
                      Nazwa varchar(50) NOT NULL,
                      Stolica_powiatu varchar(50) NOT NULL,
                      Wspolrzedne_stolicy_x float NOT NULL,
                      Wspolrzedne_stolicy_y float NOT NULL,
                      Powierzchnia float, Ludnosc int(10),
                      Inne_dane varchar(10000),
                      Wojewodztwo_w_ktorym_sie_znajduje int(10) NOT NULL,
                      Typ_powiatu int(10) NOT NULL, PRIMARY KEY (Id_terytowe),
                      INDEX (Nazwa),
                      INDEX (Stolica_powiatu),
                      INDEX (Powierzchnia),
                      INDEX (Ludnosc),
                      INDEX (Wojewodztwo_w_ktorym_sie_znajduje),
                      INDEX (Typ_powiatu));

CREATE TABLE Typy_powiatow (Id int(10) NOT NULL AUTO_INCREMENT, Typ varchar(50) NOT NULL UNIQUE, PRIMARY KEY (Id));

```

```

CREATE TABLE Wojewodztwa (Id_terytowe int(10) NOT NULL AUTO_INCREMENT,
                          Nazwa varchar(50) NOT NULL,
                          Miasto_wojewodzkie varchar(50) NOT NULL,
                          Wspolrzedne_miasta_wojewodzkiego_x float NOT NULL,
                          Wspolrzedne_miasta_wojewodzkiego_y float NOT NULL,
                          Drugie_miasto_wojewodzkie varchar(50),
                          Wspolrzedne_drugiego_miasta_wojewodzkiego_x float,
                          Wspolrzedne_drugiego_miasta_wojewodzkiego_y float,
                          Powierzchnia float, Ludnosc int(10),
                          Inne_dane varchar(10000),
                          PRIMARY KEY (Id_terytowe),
                          INDEX (Nazwa),
                          INDEX (Miasto_wojewodzkie),
                          INDEX (Drugie_miasto_wojewodzkie),
                          INDEX (Powierzchnia),
                          INDEX (Ludnosc));

ALTER TABLE Gminy ADD CONSTRAINT FKGminy238740 FOREIGN KEY (Powiat_w_ktorym_sie_znajduje) REFERENCES Powiaty (Id_terytowe);
ALTER TABLE Gminy ADD CONSTRAINT FKGminy440456 FOREIGN KEY (Typ_gminy) REFERENCES Typy_powiatow (Id);
ALTER TABLE Powiaty ADD CONSTRAINT FKPowiaty461300 FOREIGN KEY (Wojewodztwo_w_ktorym_sie_znajduje) REFERENCES Wojewodztwa (Id_terytowe);
ALTER TABLE Powiaty ADD CONSTRAINT FKPowiaty612684 FOREIGN KEY (Typ_powiatu) REFERENCES Typy_powiatow (Id);

```

## Definicja użytkowników:

```

create user 'adminDanych'@'localhost' identified by '0000';
grant create, alter, drop, insert, update, delete, select, references on administracjaRP.* to 'adminDanych'@'localhost' WITH GRANT OPTION;
create user 'gosc'@'localhost';
grant select on administracjaRP.* to 'gosc'@'localhost' with grant option;

```

## Procedury składowe:

```
CREATE PROCEDURE DodajWojewodztwo
(
    IN p_Nazwa varchar(255),
    IN p_MiastoWojewodzkie varchar(255),
    IN p_WspolrzedneMiastaWojewodzkiegoX varchar(255),
    IN p_WspolrzedneMiastaWojewodzkiegoY varchar(255),
    IN p_DrugieMiastoWojewodzkie varchar(255),
    IN p_WspolrzedneDrugiegoMiastaWojewodzkiegoX varchar(255),
    IN p_WspolrzedneDrugiegoMiastaWojewodzkiegoY varchar(255),
    IN p_DodatkoweInformacje varchar(255)
)
BEGIN
    INSERT INTO administracjaRP.Wojewodztwa (Nazwa, Miasto_wojewodzkie, Wspolrzedne_miasta_wojewodzkiego_x, Wspolrzedne_miasta_wojewodzkiego_y, Drugie_miasto_wojewodzkie, Wspolrzedne_drugiego_miasta_wojewodzkiego_x, Wspolrzedne_drugiego_miasta_wojewodzkiego_y, p_DodatkoweInformacje)
    VALUES (p_Nazwa, p_MiastoWojewodzkie, p_WspolrzedneMiastaWojewodzkiegoX, p_WspolrzedneMiastaWojewodzkiegoY, p_DrugieMiastoWojewodzkie, p_WspolrzedneDrugiegoMiastaWojewodzkiegoX, p_WspolrzedneDrugiegoMiastaWojewodzkiegoY, p_DodatkoweInformacje)
END;

CREATE PROCEDURE EdytujDaneWojewodztwa
(
    IN p_IdTerytorye integer,
    IN p_Nazwa varchar(255),
    IN p_MiastoWojewodzkie varchar(255),
    IN p_WspolrzedneMiastaWojewodzkiegoX varchar(255),
    IN p_WspolrzedneMiastaWojewodzkiegoY varchar(255),
    IN p_DrugieMiastoWojewodzkie varchar(255),
    IN p_WspolrzedneDrugiegoMiastaWojewodzkiegoX varchar(255),
    IN p_WspolrzedneDrugiegoMiastaWojewodzkiegoY varchar(255),
    IN p_DodatkoweInformacje varchar(255)
)
BEGIN
    UPDATE administracjaRP.Wojewodztwa
    SET
        Nazwa = p_Nazwa,
        Miasto_wojewodzkie = p_MiastoWojewodzkie,
        Wspolrzedne_miasta_wojewodzkiego_x = p_WspolrzedneMiastaWojewodzkiegoX,
        Wspolrzedne_miasta_wojewodzkiego_y = p_WspolrzedneMiastaWojewodzkiegoY,
        Drugie_miasto_wojewodzkie = p_DrugieMiastoWojewodzkie,
        Wspolrzedne_drugiego_miasta_wojewodzkiego_x = p_WspolrzedneDrugiegoMiastaWojewodzkiegoX,
        Wspolrzedne_drugiego_miasta_wojewodzkiego_y = p_WspolrzedneDrugiegoMiastaWojewodzkiegoY,
        inne_dane = p_DodatkoweInformacje
    WHERE id_terytorye = p_IdTerytorye;
END;
```

```
CREATE PROCEDURE wyszukajWojewodztwo
(
    IN p_Kryterium varchar(255)
)
BEGIN
    SELECT *
    FROM administracjaRP.Wojewodztwa
    WHERE Nazwa LIKE CONCAT('%', p_Kryterium, '%')
        OR Miasto_wojewodzkie LIKE CONCAT('%', p_Kryterium, '%')
        OR inne_dane LIKE CONCAT('%', p_Kryterium, '%')
        OR Drugie_miasto_wojewodzkie LIKE CONCAT('%', p_Kryterium, '%');
END;

CREATE PROCEDURE DodajPowiat
(
    IN p_Nazwa varchar(255),
    IN p_StolicaPowiatu varchar(255),
    IN p_WspolrzedneStolicyPowiatuX varchar(255),
    IN p_WspolrzedneStolicyPowiatuY varchar(255),
    IN p_TypPowiatu integer(10),
    IN p_Wojewodztwo integer(10),
    IN p_DodatkoweInformacje varchar(255)
)
BEGIN
    INSERT INTO administracjaRP.Powiaty (Nazwa, Stolica_powiatu, Wspolrzedne_stolicy_x, Wspolrzedne_stolicy_y, Typ_powiatu, Wojewodztwo_w_ktorym_sie_znajduje, inne_dane)
    VALUES (p_Nazwa, p_StolicaPowiatu, p_WspolrzedneStolicyPowiatuX, p_WspolrzedneStolicyPowiatuY, p_TypPowiatu, p_Wojewodztwo, p_DodatkoweInformacje);
END;
```

```

CREATE PROCEDURE EdytujDanePowiatu
(
    IN p_IdTerytowe integer(10),
    IN p_Nazwa varchar(255),
    IN p_StolicaPowiatu varchar(255),
    IN p_WspolrzedneStolicyPowiatuX varchar(255),
    IN p_WspolrzedneStolicyPowiatuY varchar(255),
    IN p_TypPowiatu integer(10),
    IN p_Wojewodztwo integer(10),
    IN p_DodatkoweInformacje varchar(255)
)
BEGIN
    UPDATE administracjaRP.Powiaty
    SET
        Nazwa = p_Nazwa,
        Stolica_powiatu = p_StolicaPowiatu,
        Wspolrzedne_stolicy_x = p_WspolrzedneStolicyPowiatuX,
        Wspolrzedne_stolicy_y = p_WspolrzedneStolicyPowiatuY,
        Typ_powiatu = p_TypPowiatu,
        Wojewodztwo_w_ktorym_sie_znajduje = p_Wojewodztwo,
        inne_dane = p_DodatkoweInformacje
    WHERE Id_terytowe = p_IdTerytowe;
END;

CREATE PROCEDURE WyszukajPowiat
(
    IN p_Kryterium varchar(255)
)
BEGIN
    SELECT *
    FROM administracjaRP.Powiaty
    WHERE Nazwa LIKE CONCAT('%', p_Kryterium, '%')
        OR Stolica_powiatu LIKE CONCAT('%', p_Kryterium, '%')
        OR inne_dane LIKE CONCAT('%', p_Kryterium, '%');
END;

CREATE PROCEDURE DodajGminę
(
    IN p_Nazwa varchar(255),
    IN p_StolicaGminy varchar(255),
    IN p_WspolrzedneStolicyGminyX varchar(255),
    IN p_WspolrzedneStolicyGminyY varchar(255),
    IN p_TypGminy integer(10),
    IN p_Powierzchnia float(10),
    IN p_Ludnosc integer(10),
    IN p_Powiat integer(10),
    IN p_DodatkoweInformacje varchar(255)
)
BEGIN
    INSERT INTO administracjaRP.Gminy (Nazwa, Stolica_gminy, Wspolrzedne_stolicy_x,Wspolrzedne_stolicy_y,Typ_gminy,Powierzchnia,Ludnosc,Powiat_w_ktorym_sie_znajduje,inne_dane)
    VALUES (p_Nazwa, p_StolicaGminy, p_WspolrzedneStolicyGminyX,p_WspolrzedneStolicyGminyY, p_TypGminy, p_Powierzchnia, p_Ludnosc, p_Powiat, p_DodatkoweInformacje);
END;

```

```

CREATE PROCEDURE EdytujDaneGminy
(
    IN p_IdTerytowe integer(10),
    IN p_Nazwa varchar(255),
    IN p_StolicaGminy varchar(255),
    IN p_WspolrzedneStolicyGminyX varchar(255),
    IN p_WspolrzedneStolicyGminyY varchar(255),
    IN p_TypGminy integer(10),
    IN p_Powierzchnia float(10),
    IN p_Ludnosc integer(10),
    IN p_Powiat integer(10),
    IN p_DodatkoweInformacje varchar(255)
)
BEGIN
    UPDATE administracjaRP.Gminy
    SET
        Nazwa = p_Nazwa,
        Stolica_gminy = p_StolicaGminy,
        Wspolrzedne_stolicy_x = p_WspolrzedneStolicyGminyX,
        Wspolrzedne_stolicy_y = p_WspolrzedneStolicyGminyY,
        Typ_gminy = p_TypGminy,
        Powierzchnia = p_Powierzchnia,
        Ludnosc = p_Ludnosc,
        Powiat_w_ktorym_sie_znajduje = p_Powiat,
        inne_dane = p_DodatkoweInformacje
    WHERE Id_terytowe = p_IdTerytowe;
END;

CREATE PROCEDURE WyszukajGmine
(
    IN p_Kryterium varchar(255)
)
BEGIN
    SELECT *
    FROM administracjaRP.Gminy
    WHERE Nazwa LIKE CONCAT('%', p_Kryterium, '%')
        OR Stolica_gminy LIKE CONCAT('%', p_Kryterium, '%')
        OR inne_dane LIKE CONCAT('%', p_Kryterium, '%');
END;

```

## Triggery:

```
CREATE TRIGGER after_insert_powiat_population
AFTER INSERT ON administracjaRP.Powiaty
FOR EACH ROW
BEGIN
    UPDATE administracjaRP.Wojewodztwa W
    SET W.Ludnosc = (SELECT SUM(P.Ludnosc) FROM administracjaRP.Powiaty P WHERE P.Wojewodztwo_w_ktorym_sie_znajduje = W.Id_terytowe)
    WHERE W.Id_terytowe = NEW.Wojewodztwo_w_ktorym_sie_znajduje;
END;

CREATE TRIGGER after_update_powiat_population
AFTER UPDATE ON administracjaRP.Powiaty
FOR EACH ROW
BEGIN
    UPDATE administracjaRP.Wojewodztwa W
    SET W.Ludnosc = (SELECT SUM(P.Ludnosc) FROM administracjaRP.Powiaty P WHERE P.Wojewodztwo_w_ktorym_sie_znajduje = W.Id_terytowe)
    WHERE W.Id_terytowe = NEW.Wojewodztwo_w_ktorym_sie_znajduje;
END;

CREATE TRIGGER after_delete_powiat_population
AFTER DELETE ON administracjaRP.Powiaty
FOR EACH ROW
BEGIN
    UPDATE administracjaRP.Wojewodztwa W
    SET W.Ludnosc = (SELECT SUM(P.Ludnosc) FROM administracjaRP.Powiaty P WHERE P.Wojewodztwo_w_ktorym_sie_znajduje = W.Id_terytowe)
    WHERE W.Id_terytowe = OLD.Wojewodztwo_w_ktorym_sie_znajduje;
END;

CREATE TRIGGER after_delete_powiat
AFTER DELETE ON administracjaRP.Powiaty
FOR EACH ROW
BEGIN
    UPDATE administracjaRP.Wojewodztwa W
    SET W.Powierzchnia = (SELECT SUM(P.Powierzchnia) FROM administracjaRP.Powiaty P WHERE P.Wojewodztwo_w_ktorym_sie_znajduje = W.Id_terytowe)
    WHERE W.Id_terytowe = OLD.Wojewodztwo_w_ktorym_sie_znajduje;
END;
```

```

CREATE TRIGGER after_delete_gmina
AFTER DELETE ON administracjaRP.Gminy
FOR EACH ROW
BEGIN
    UPDATE administracjaRP.Powiaty P
    SET P.Powierzchnia = (SELECT SUM(G.Powierzchnia) FROM administracjaRP.Gminy G WHERE G.Powiat_w_ktorym_sie_najduje = P.Id_terytowe)
    WHERE P.Id_terytowe = OLD.Powiat_w_ktorym_sie_najduje;
END;

CREATE TRIGGER after_insert_powiat
AFTER INSERT ON administracjaRP.Powiaty
FOR EACH ROW
BEGIN
    UPDATE administracjaRP.Wojewodztwa W
    SET W.Powierzchnia = (SELECT SUM(P.Powierzchnia) FROM administracjaRP.Powiaty P WHERE P.Wojewodztwo_w_ktorym_sie_najduje = W.Id_terytowe)
    WHERE W.Id_terytowe = NEW.Wojewodztwo_w_ktorym_sie_najduje;
END;

CREATE TRIGGER after_insert_gmina
AFTER INSERT ON administracjaRP.Gminy
FOR EACH ROW
BEGIN
    UPDATE administracjaRP.Powiaty P
    SET P.Powierzchnia = (SELECT SUM(G.Powierzchnia) FROM administracjaRP.Gminy G WHERE G.Powiat_w_ktorym_sie_najduje = P.Id_terytowe)
    WHERE P.Id_terytowe = NEW.Powiat_w_ktorym_sie_najduje;
END;

CREATE TRIGGER after_update_powiat
AFTER UPDATE ON administracjaRP.Powiaty
FOR EACH ROW
BEGIN
    UPDATE administracjaRP.Wojewodztwa W
    SET W.Powierzchnia = (SELECT SUM(P.Powierzchnia) FROM administracjaRP.Powiaty P WHERE P.Wojewodztwo_w_ktorym_sie_najduje = W.Id_terytowe)
    WHERE W.Id_terytowe = NEW.Wojewodztwo_w_ktorym_sie_najduje;
END;

CREATE TRIGGER after_update_gmina
AFTER UPDATE ON administracjaRP.Gminy
FOR EACH ROW
BEGIN
    UPDATE administracjaRP.Powiaty P
    SET P.Powierzchnia = (SELECT SUM(G.Powierzchnia) FROM administracjaRP.Gminy G WHERE G.Powiat_w_ktorym_sie_najduje = P.Id_terytowe)
    WHERE P.Id_terytowe = NEW.Powiat_w_ktorym_sie_najduje;
END;

```

## Widoki:

```
CREATE VIEW WidokInformacjiGminInne AS
SELECT
    Nazwa,
    Stolica_gminy,
    Powierzchnia,
    Ludnosc,
    Inne_dane
FROM
    administracjaRP.Gminy;

-- Widok pokazujący nazwę województwa, stolicę województwa, powiaty jakie się w nim znajdują

CREATE VIEW WidokWojewodztwAwszystkieTypy AS
SELECT
    W.Nazwa AS WojewodztwoNazwa,
    W.Miasto_wojewodzkie AS MiastoWojewodzkie,
    TP.Typ AS TypPowiatu,
    TG.Typ AS TypGminy,
    P.Nazwa AS PowiatNazwa,
    G.Nazwa AS GminaNazwa
FROM
    administracjaRP.Wojewodztwa W
JOIN
    administracjaRP.Powiaty P ON W.Id_terytory = P.Wojewodztwo_w_którym_sie_znajduje
JOIN
    administracjaRP.Gminy G ON P.Id_terytory = G.Powiat_w_którym_sie_znajduje
JOIN
    administracjaRP.Typy_powiatow TP ON P.Typ_powiatu = TP.Id
JOIN
    administracjaRP.Typ_gminy TG ON G.Typ_gminy = TG.Id;
```

```
CREATE VIEW WidokInformacjiWojewodztwInne AS
SELECT
    Nazwa,
    Miasto_wojewodzkie,
    Powierzchnia,
    Ludnosc,
    Inne_dane
FROM
    administracjaRP.Wojewodztwa;

-- Widok pokazujący nazwę powiatu, stolicę powiatu, typ powiatu, typ gminy, nazwy gminy

CREATE VIEW WidokInformacjiPowiatowInne AS
SELECT
    Nazwa,
    Stolica_powiatu,
    Powierzchnia,
    Ludnosc,
    Inne_dane
FROM
    administracjaRP.Powiaty;
```

```
CREATE VIEW WidokInformacjiPowiatowNazwyWojewodztGmin AS
SELECT
    P.Nazwa AS PowiatNazwa,
    P.Stolica_powiatu,
    P.Wspolrzedne_stolicy_x,
    P.Wspolrzedne_stolicy_y,
    P.Powierzchnia AS PowiatPowierzchnia,
    P.Ludnosc AS PowiatLudnosc,
    P.Inne_dane AS PowiatInneDane,
    W.Nazwa AS WojewodztwoNazwa,
    G.Nazwa AS GminaNazwa,
    TP.Typ AS TypPowiatu
FROM
    administracjaRP.Powiaty P
JOIN
    administracjaRP.Wojewodztwa W ON P.Wojewodztwo_w_którym_sie_znajduje = W.Id_terytory
LEFT JOIN
    administracjaRP.Gminy G ON P.Id_terytory = G.Powiat_w_którym_sie_znajduje
JOIN
    administracjaRP.Typy_powiatow TP ON P.Typ_powiatu = TP.Id;
```

```
CREATE VIEW WidokInformacjiGminNazwaPowiatuWojewodztwa AS
SELECT
    administracjaRP.Gminy.Nazwa AS GminaNazwa,
    administracjaRP.Gminy.Stolica_gminy,
    administracjaRP.Typ_gminy.Typ,
    administracjaRP.Powiaty.Nazwa AS PowiatNazwa,
    administracjaRP.Wojewodztwa.Nazwa AS WojewodztwoNazwa
FROM
    administracjaRP.Gminy
JOIN
    administracjaRP.Powiaty ON Gminy.Powiat_w_ktorym_sie_znajduje = Powiaty.Id_terytowe
JOIN
    administracjaRP.Wojewodztwa ON Powiaty.Wojewodztwo_w_ktorym_sie_znajduje = Wojewodztwa.Id_terytowe
JOIN
    administracjaRP.Typ_gminy ON Gminy.Id_terytowe = Typ_gminy.Id;
```

```
CREATE VIEW WidokInformacjiWojewodztwNazwyPowiatowGmin AS
SELECT
    W.Nazwa AS WojewodztwoNazwa,
    W.Miasto_wojewodzkie AS MiastoWojewodzkie,
    W.Drugie_miasto_wojewódzkie AS DrugieMiastoWojewodzkie,
    P.Nazwa AS PowiatNazwa,
    G.Nazwa AS GminaNazwa
FROM
    administracjaRP.Wojewodztwa W
LEFT JOIN
    administracjaRP.Powiaty P ON W.Id_terytowe = P.Wojewodztwo_w_ktorym_sie_znajduje
LEFT JOIN
    administracjaRP.Gminy G ON P.Id_terytowe = G.Powiat_w_ktorym_sie_znajduje;
```

```
CREATE VIEW WidokWojewodztw AS
SELECT
    Nazwa,
    Miasto_wojewodzkie,
    Wspolrzedne_miasta_wojewodzkiego_x,
    Wspolrzedne_miasta_wojewodzkiego_y,
    Drugie_miasto_wojewodzkie,
    Wspolrzedne_drugiego_miasta_wojewodzkiego_x,
    Wspolrzedne_drugiego_miasta_wojewodzkiego_y,
    Powierzchnia
FROM
    administracjaRP.Wojewodztwa;

-- Widok pokazujący nazwę powiatu, stolicę powiatu, współrzędne
CREATE VIEW WidokPowiatow AS
SELECT
    Nazwa,
    Stolica_powiatu,
    Wspolrzedne_stolicy_x,
    Wspolrzedne_stolicy_y,
    Powierzchnia
FROM
    administracjaRP.Powiaty;

-- Widok pokazujący nazwę gminy, stolicę gminy, współrzędne
CREATE VIEW WidokGmin AS
SELECT
    Nazwa,
    Stolica_gminy,
    Wspolrzedne_stolicy_x,
    Wspolrzedne_stolicy_y,
    Powierzchnia
FROM
    administracjaRP.Gminy;
```

## Testy

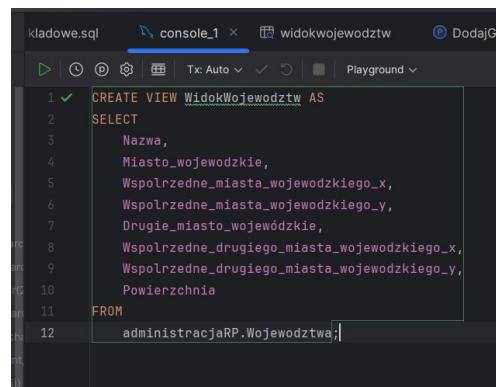
Zgodnie z ustaleniami z prowadzącym oraz ze specyfiką naszej bazy danych- nie przeprowadzaliśmy testów wydajnościowych, tylko skupiliśmy się na poprawnym działaniu implementacji bazy danych.

Uznaliśmy, że w celu pokazania działania bazy danych, mechanizmy w niej działające pokażemy na niewielkich rozmiarowo przykładach. Dla wielu danych zrzuty ekranu były bardziej skomplikowane i trudniej było sprawdzić ich poprawność (zliczanie danych dot. Populacji oraz powierzchni dla powiatów oraz województw na podstawie danych gmin- najmniejszej jednostki administracyjnej w naszej bazie).

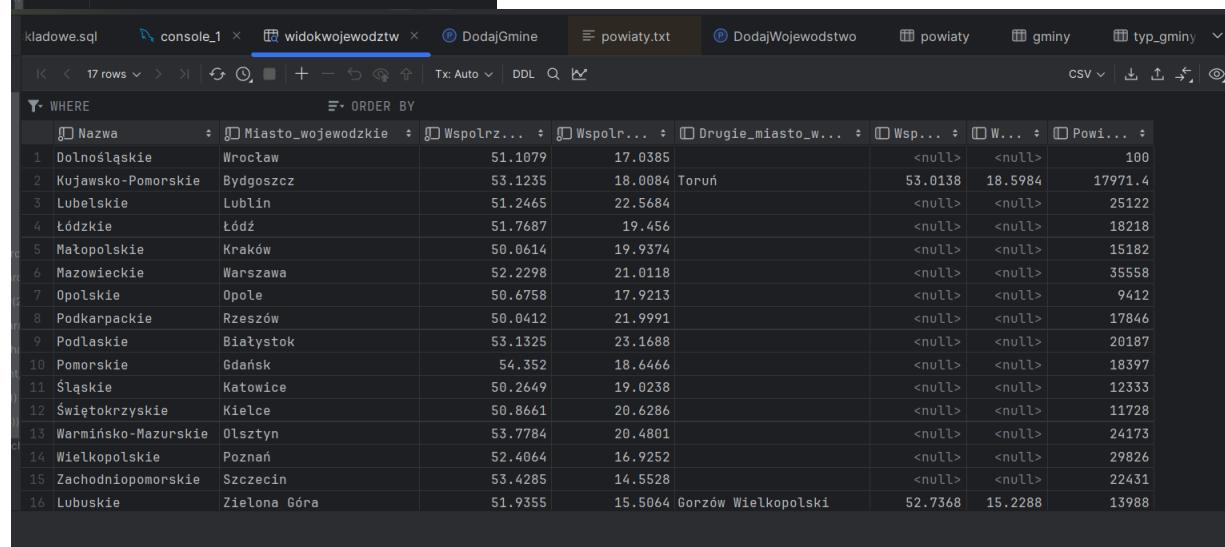
### Widoki:

Testy widoków opierają się na dwóch udokumentowanych zrzutach ekranu (1 zrzut ekranu kodu 1 zrzut ekranu wyniku testu po skompilowaniu kodu). Celem testu jest sprawdzenie poprawności generowania widoków i kompletności żądanych kolumn (ilość danych jest nieistotna).

### Widok Województw



```
CREATE VIEW WidokWojewodztw AS
SELECT
    Nazwa,
    Miasto_wojewodzkie,
    Wspolrzedne_miasta_wojewodzkiego_x,
    Wspolrzedne_miasta_wojewodzkiego_y,
    Drugie_miasto_wojewodzkie,
    Wspolrzedne_drujiego_miasta_wojewodzkiego_x,
    Wspolrzedne_drujiego_miasta_wojewodzkiego_y,
    Powierzchnia
FROM
    administracjaRP.Wojewodztwa;
```



Nazwa	Miasto_wojewodzkie	Wspolrz..._x	Wspolrz..._y	Drugie_miasto_w..._x	Wsp..._y	W..._y	Powi..._y
Dolnośląskie	Wrocław	51.1079	17.0385	<null>	<null>	100	
Kujawsko-Pomorskie	Bydgoszcz	53.1235	18.0084	Toruń	53.0138	18.5984	17971.4
Lubelskie	Lublin	51.2465	22.5684	<null>	<null>	25122	
Łódzkie	Łódź	51.7687	19.456	<null>	<null>	18218	
Małopolskie	Kraków	50.0614	19.9374	<null>	<null>	15182	
Mazowieckie	Warszawa	52.2298	21.0118	<null>	<null>	35558	
Opolskie	Opole	50.6758	17.9213	<null>	<null>	9412	
Podkarpackie	Rzeszów	50.0412	21.9991	<null>	<null>	17846	
Podlaskie	Białystok	53.1325	23.1688	<null>	<null>	20187	
Pomorskie	Gdańsk	54.352	18.6466	<null>	<null>	18397	
Śląskie	Katowice	50.2649	19.0238	<null>	<null>	12333	
Świętokrzyskie	Kielce	50.8661	20.6286	<null>	<null>	11728	
Warmińsko-Mazurskie	Olsztyn	53.7784	20.4801	<null>	<null>	24173	
Wielkopolskie	Poznań	52.4064	16.9252	<null>	<null>	29826	
Zachodniopomorskie	Szczecin	53.4285	14.5528	<null>	<null>	22431	
Lubuskie	Zielona Góra	51.9355	15.5064	Gorzów Wielkopolski	52.7368	15.2288	13988

## WidokPowiatow

The screenshot shows a database console interface with a dark theme. In the top bar, there are several tabs: 'console\_1' (highlighted), 'widokpowiatow', 'widokwojewodztw', 'DodajGminie', 'powiaty.txt', 'DodajWojewodstwo', 'powiaty', 'gminy', and 'ty'. Below the tabs, the main area displays a SQL script for creating a view:

```
1 ✓ CREATE VIEW WidokPowiatow AS
2   SELECT
3     Nazwa,
4     Stolica_powiatu,
5     Wspolrzedne_stolicy_x,
6     Wspolrzedne_stolicy_y,
7     Powierzchnia
8   FROM
9     administracjaRP.Powiaty;
```

Below the script, there is a table preview with the following data:

	WHERE	ORDER BY
1	Nazwa : Nie	Stolica_powiatu : tak
		Wspolrzedne_stolicy_x : 1
		Wspolrzedne_stolicy_y : 2
		Powierzchnia : 100

## WidokGmin

The screenshot shows a database console interface with a dark theme. In the top bar, there are several tabs: 'console\_1' (highlighted), 'widokgmin', 'widokpowiatow', 'widokwojewodztw', 'DodajGminie', 'DodajWojewodstwo', 'powiaty', 'gminy', and 'ty'. Below the tabs, the main area displays a SQL script for creating a view:

```
1 ✓ CREATE VIEW WidokGmin AS
2   SELECT
3     Nazwa,
4     Stolica_gminy,
5     Wspolrzedne_stolicy_x,
6     Wspolrzedne_stolicy_y,
7     Powierzchnia
8   FROM
9     administracjaRP.Gminy;
```

Below the script, there is a table preview with the following data:

	WHERE	ORDER BY
1	Nazwa : Warszawa	Stolica_gminy : Mokotów
		Wspolrzedne_stolicy_x : 52.2298
		Wspolrzedne_stolicy_y : 21.0118
		Powierzchnia : 70
2	Nazwa : Kraków	Stolica_gminy : Podgórze
		Wspolrzedne_stolicy_x : 50.0614
		Wspolrzedne_stolicy_y : 19.9374
		Powierzchnia : 30

## WidokInformacjiGminNazwaPowiatuWojewodztwa

```

1 ✓ CREATE VIEW WidokInformacjiGminNazwaPowiatuWojewodztwa AS
2   SELECT
3     administracjaRP.Gminy.Nazwa AS GminaNazwa,
4     administracjaRP.Gminy.Stolica_gminy,
5     administracjaRP.Typ_gminy.Typ,
6     administracjaRP.Powiaty.Nazwa AS PowiatNazwa,
7     administracjaRP.Wojewodztwa.Nazwa AS WojewodztwoNazwa
8   FROM
9     administracjaRP.Gminy
10   JOIN
11     administracjaRP.Powiaty ON Gminy.Powiat_w_ktorym_sie_znajduje = Powiaty.Id_terytory
12   JOIN
13     administracjaRP.Wojewodztwa ON Powiaty.Wojewodztwo_w_ktorym_sie_znajduje = Wojewodztwa.Id_terytory
14   JOIN
15     administracjaRP.Typ_gminy ON Gminy.Id_terytory = Typ_gminy.Id;

```

WHERE ORDER BY

GminaNazwa	Stolica_gminy	Typ	PowiatNazwa	WojewodztwoNazwa
Warszawa	Mokotów	gmina miejska	Nie	Dolnośląskie
Kraków	Podgórze	gmina wiejska	Nie	Dolnośląskie

## WidokInformacjiPowiatowNazwyWojewodztGmin

```

1 ✓ CREATE VIEW WidokInformacjiPowiatowNazwyWojewodztGmin AS
2   SELECT
3     P.Nazwa AS PowiatNazwa,
4     P.Stolica_powiatu,
5     P.Wspolrzedne_stolicy_x,
6     P.Wspolrzedne_stolicy_y,
7     P.Powierzchnia AS PowiatPowierzchnia,
8     P.Ludnosc AS PowiatLudnosc,
9     P.Inne_dane AS PowiatInneDane,
10    W.Nazwa AS WojewodztwoNazwa,
11    G.Nazwa AS GminaNazwa,
12    TP.Typ AS TypPowiatu
13   FROM
14     administracjaRP.Powiaty P
15   JOIN
16     administracjaRP.Wojewodztwa W ON P.Wojewodztwo_w_ktorym_sie_znajduje = W.Id_terytory
17   LEFT JOIN
18     administracjaRP.Gminy G ON P.Id_terytory = G.Powiat_w_ktorym_sie_znajduje
19   JOIN
20     administracjaRP.Typy_powiatow TP ON P.Typ_powiatu = TP.Id;

```

WHERE ORDER BY

PowiatNazwa	Stolica_powiatu	Ws...	Ws...	Powi...	Po...	P...	Wojewo...	GminaNazwa	TypPowiatu
Nie	tak	1	2	100	1000	<null>	Dolnośląskie	Kraków	Miasto
Nie	tak	1	2	100	1000	<null>	Dolnośląskie	Warszawa	Miasto

## WidokInformacjiWojewodztwNazwyPowiatowGmin

```
CREATE VIEW WidokInformacjiWojewodztwNazwyPowiatowGmin AS
SELECT
    W.Nazwa AS WojewodztwoNazwa,
    W.Miasto_wojewodzkie AS MiastoWojewodzkie,
    W.Drugie_miasto_wojewodzkie AS DrugieMiastoWojewodzkie,
    P.Nazwa AS PowiatNazwa,
    G.Nazwa AS GminaNazwa
FROM
    administracjaRP.Wojewodztwa W
LEFT JOIN
    administracjaRP.Powiaty P ON W.Id_terytory = P.Wojewodztwo_w_ktorym_sie_znajduje
LEFT JOIN
    administracjaRP.Gminy G ON P.Id_terytory = G.Powiat_w_ktorym_sie_znajduje;
```

	WHERE	ORDER BY	CSV
1	Dolnośląskie	Wrocław	Nie Kraków
2	Dolnośląskie	Wrocław	Nie Warszawa
3	Kujawsko-Pomorskie	Bydgoszcz	Toruń <null> <null>
4	Lubelskie	Lublin	<null> <null>
5	Łódzkie	Łódź	<null> <null>
6	Małopolskie	Kraków	<null> <null>
7	Mazowieckie	Warszawa	<null> <null>
8	Opolskie	Opole	<null> <null>
9	Podkarpackie	Rzeszów	<null> <null>
10	Podlaskie	Białystok	<null> <null>
11	Pomorskie	Gdańsk	<null> <null>
12	Śląskie	Katowice	<null> <null>
13	Świętokrzyskie	Kielce	<null> <null>
14	Warmińsko-Mazurskie	Olsztyń	<null> <null>
15	Wielkopolskie	Poznań	<null> <null>
16	Zachodniopomorskie	Szczecin	<null> <null>
17	Lubuskie	Zielona Góra	Gorzów Wielkopolski <null> <null>
18	Ma	Mie	<null> <null>

# **Widok Informacji Województw Innych**

```
CREATE VIEW WidokInformacjiWojewodztwInne AS
SELECT
    Nazwa,
    Miasto_wojewodzkie,
    Powierzchnia,
    Ludnosc,
    Inne_dane
FROM
    administracjaRP.Wojewodztwa;
```

	WHERE	ORDER BY			
	Nazwa	Miasto_wojewodzkie	Powierzchnia	Ludnosc	Inne_dane
1	Dolnośląskie	Wrocław	100	1000	
2	Kujawsko-Pomorskie	Bydgoszcz	17971.4	2085323	
3	Lubelskie	Lublin	25122	2142212	
4	Łódzkie	Łódź	18218	2493603	
5	Małopolskie	Kraków	15182	3412400	
6	Mazowieckie	Warszawa	35558	5360692	
7	Opolskie	Opole	9412	986132	
8	Podkarpackie	Rzeszów	17846	2127275	
9	Podlaskie	Białystok	20187	1182873	
10	Pomorskie	Gdańsk	18397	2334556	
11	Śląskie	Katowice	12333	4532111	
12	Świętokrzyskie	Kielce	11728	1241347	
13	Warmińsko-Mazurskie	Olsztyn	24173	1435602	
14	Wielkopolskie	Poznań	29826	3479776	
15	Zachodniopomorskie	Szczecin	22431	1692720	
16	Lubuskie	Zielona Góra	13988	979976	
17	Ma	Mie	<null>	<null>	istnieje

## WidokInformacjiPowiatowInne

The screenshot shows a database interface with a query editor and a results grid. The query editor contains the following SQL code:

```
CREATE VIEW WidokInformacjiPowiatowInne AS
SELECT
    Nazwa,
    Stolica_powiatu,
    Powierzchnia,
    Ludosc,
    Inne_dane
FROM
    administracjaRP.Powiaty;
```

The results grid shows the following data:

Nazwa	Stolica_powiatu	Powierzchnia	Ludosc	Inne_dane
Nie	tak	100	1000	<null>

## WidokInformacjiGminInne

The screenshot shows a database interface with a query editor and a results grid. The query editor contains the following SQL code:

```
CREATE VIEW WidokInformacjiGminInne AS
SELECT
    Nazwa,
    Stolica_gminy,
    Powierzchnia,
    Ludosc,
    Inne_dane
FROM
    administracjaRP.Gminy;
```

The results grid shows the following data:

Nazwa	Stolica_gminy	Powierzchnia	Ludosc	Inne_dane
Warszawa	Mokotów	70	700	Stolica Polski
Kraków	Podgórze	30	300	Stolica Małopolski

## WidokWojewodztwaWszystkieTypy

The screenshot shows a database interface with a code editor and a results table. The code editor contains the following SQL script:

```
1 ✓ CREATE VIEW WidokWojewodztwaWszystkieTypy AS
2
3     W.Nazwa AS WojewodztwoNazwa,
4     W.Miasto_wojewodzkie AS MiastoWojewodzkie,
5     TP.Typ AS TypPowiatu,
6     TG.Typ AS TypGminy,
7     P.Nazwa AS PowiatNazwa,
8     G.Nazwa AS GminaNazwa
9
10    FROM administracjaRP.Wojewodztwa W
11    JOIN administracjaRP.Powiaty P ON W.Id_terytoryowe = P.Wojewodztwo_w_ktorym_sie_znajduje
12    JOIN administracjaRP.Gminy G ON P.Id_terytoryowe = G.Powiat_w_ktorym_sie_znajduje
13    JOIN administracjaRP.Typy_powiatow TP ON P.Typ_powiatu = TP.Id
14    JOIN administracjaRP.Typ_gminy TG ON G.Typ_gminy = TG.Id;
```

The results table below the code editor displays data for two entries:

WojewodztwoNazwa	MiastoWojewodzkie	TypPowiatu	TypGminy	PowiatNazwa	GminaNazwa
Dolnośląskie	Wrocław	Miasto	Miasto	Nie	Warszawa
Dolnośląskie	Wrocław	Miasto	Miasto	Nie	Kraków

### Procedury:

Testy procedur zostały przeprowadzone pod kątem działania procedury po wprowadzeniu właściwych danych oraz możliwych testów po wprowadzeniu danych zawierających błędy.

## DodajWojewodstwo

```
1 ✓ call DodajWojewodstwo( p_Nazwa: 'Ma', p_MiastoWojewodzkie: 'Mie',  
2   p_WspolrzedneMiastaWojewodzkiegoX: 10, p_WspolrzedneMiastaWojewodzkiegoY: 18.232, p_DrugieMiastoWojewodzkie: null,  
3   p_WspolrzedneDrugiegoMiastaWojewodzkiegoX: null, p_WspolrzedneDrugiegoMiastaWojewodzkiegoY: null, p_DodatkoweInformacje: 'istnieje');
```

id_owe	Nazwa	Miasto_wojewodzkie	... lat	W... lon	Dr... zakres	Wsp... x	Wsp... y	Po... wiaty	... zakres	Inne_dane
45	Łódzkie	Kraków	51.7687	19.456	<null>	<null>	<null>	18218	2493663	
46	Małopolskie	Warszawa	52.2298	21.0118	<null>	<null>	<null>	15182	3412400	
47	Mazowieckie	Opole	50.6758	17.9213	<null>	<null>	<null>	35558	5360692	
48	Opolskie	Rzeszów	50.0412	21.9991	<null>	<null>	<null>	9412	986132	
49	Podkarpackie	Białystok	53.1325	23.1688	<null>	<null>	<null>	17846	2127275	
50	Podlaskie	Gdańsk	54.352	18.6466	<null>	<null>	<null>	20187	1182873	
51	Pomorskie	Katowice	50.2649	19.0238	<null>	<null>	<null>	18397	2334556	
52	Śląskie	Kielce	50.8661	20.6286	<null>	<null>	<null>	12353	4532111	
53	Świętokrzyskie	Olsztyn	53.7784	20.4801	<null>	<null>	<null>	11728	1241347	
54	Warmińsko-Mazurskie	Poznań	52.4064	16.9252	<null>	<null>	<null>	24173	1435602	
55	Wielkopolskie	Szczecin	53.4285	14.5528	<null>	<null>	<null>	29826	3479776	
56	Zachodniopomorskie	Zielona Góra	51.9355	15.5064	Gorzów Wi...	52.7368	15.2288	13988	979976	
57	Lubuskie	Mie	10	18.232	<null>	<null>	<null>	<null>	<null>	istnieje

## Błąd null value

The screenshot shows a database interface with a code editor and a console window.

**Code Editor:**

```
1 ① call DodajWojewodstwo( p.Nazwa: 'Ma', p.MiastoWojewodzkie: null,
2                               p.WspolrzedneMiastaWojewodzkiegoX: 10, p.WspolrzedneMiastaWojewodzkiegoY: 18.232, p.DrugieMiastoWojewodzkie: null,
3                               p.WspolrzedneDrugiegoMiastaWojewodzkiegoX: null, p.WspolrzedneDrugiegoMiastaWojewodzkiegoY: null, p.DodatkoweInformacje: 'istnieje');
```

**Console Output:**

```
[23000][1048] Column 'Miasto_wojewodzkie' cannot be null

[2023-12-21 23:11:27] [42S22][1054] Unknown column 'k' in 'field list'
[2023-12-21 23:11:27] [42S22][1054] Unknown column 'k' in 'field list'
administracjarp> call DodajWojewodstwo('Ma',null,
                                         10,18.232,null,
                                         null,null,'istnieje')
[2023-12-21 23:12:35] [23000][1048] Column 'Miasto_wojewodzkie' cannot be null
[2023-12-21 23:12:35] [23000][1048] Column 'Miasto_wojewodzkie' cannot be null
```

**Status Bar:** 3:23 CRLF UTF-8 4 spaces

## Błąd varchar value for float field

The screenshot shows a database interface with a code editor and a console window.

**Code Editor:**

```
1 ① call DodajWojewodstwo( p.Nazwa: 'Ma', p.MiastoWojewodzkie: 'Mie',
2                               p.WspolrzedneMiastaWojewodzkiegoX: k, p.WspolrzedneMiastaWojewodzkiegoY: 18.232, p.DrugieMiastoWojewodzkie: null,
3                               p.WspolrzedneDrugiegoMiastaWojewodzkiegoX: null, p.WspolrzedneDrugiegoMiastaWojewodzkiegoY: null, p.DodatkoweInformacje: 'istnieje');
```

**Console Output:**

```
[42S22][1054] Unknown column 'k' in 'field list'

[2023-12-21 23:10:39] [42S22][1054] Unknown column 'k' in 'field list'
[2023-12-21 23:10:39] [42S22][1054] Unknown column 'k' in 'field list'
administracjarp> call DodajWojewodstwo('Ma','Mie',
                                         k,18.232,null,
                                         null,null,'istnieje')
[2023-12-21 23:11:27] [42S22][1054] Unknown column 'k' in 'field list'
[2023-12-21 23:11:27] [42S22][1054] Unknown column 'k' in 'field list'
```

**Status Bar:** 1:32 CRLF UTF-8 4 spaces

## DodajPowiat

The screenshot shows a database interface with a query editor and a results grid. The query editor contains the following SQL code:

```
call DodajPowiat( p_Nazwa: 'Niemanowo', p_StolicaPowiatu: 'Możowo', p_WspolrzedneStolicyPowiatuX: 12,
                    p_WspolrzedneStolicyPowiatuY: 31, p_TypPowiatu: 3, p_Wojewodztwo: 46, p_DodatkoweInformacje: 'niemoże')
```

The results grid shows the output of the stored procedure:

Id_terytoryowe	Nazwa	Stolica_powiatu	Wspolrzedne_stolicy_x	Wspolrzedne_stolicy_y	Powierzchnia	Ludosc	Inne_dane
41	Nie	tak	1	2	150	1500	<null>
43	Niemanowo	Możowo	12	31	<null>	<null>	niemoże

## DodajGminę:

The screenshot shows a database interface with a query editor and a results grid. The query editor contains the following SQL code:

```
call DodajGmine( p_Nazwa: 'Warszawa', p_StolicaGminy: 'Mokotów', p_WspolrzedneStolicyGminyX: 52.2298,
                   p_WspolrzedneStolicyGminyY: 21.0118, p_TypGminy: 1, p_Powierzchnia: 517.24,
                   p_Ludosc: 1765005, p_Powiat: 41, p_DodatkoweInformacje: 'Stolica Polski')
```

The results grid shows the output of the stored procedure:

Id_terytoryowe	Nazwa	... StolicaGminy	... WspolrzedneStolicyGminyX	... WspolrzedneStolicyGminyY	Ludosc	Inne_dane	... TypGminy		
3	Warszawa	Mokotów	52.2298	21.0118	517.24	1765005	Stolica Polski	41	1

At the bottom of the interface, there is a terminal window showing the connection status and a SELECT query:

```
[2023-12-21 23:30:25] Connected
administracjarp> use administracjarp
[2023-12-21 23:30:25] completed in 0 ms
administracjarp> SELECT t.*  
  FROM administracjarp.gminy t  
  LIMIT 501
[2023-12-21 23:30:25] 1 row retrieved starting from 1 in 32 ms (execution: 3 ms, fetching: 29 ms)
```

## Błąd varchar value for float field

The screenshot shows a database console window with a dark theme. At the top, there is a code editor with the following SQL command:

```
1 call DodajGminę( p_Nazwa: 'Kaków', p_StolicaGminy: 'górze', p_WspółzneStolicyGminyX: 'kak', p_WspółzneStolicyGminyY: 19.9374, p_TypGminy: 1, p_Powierzchnia: 327.00, p_Ludosc: null, p_Powiat: 41, p_DodatkoweInformacje: 'Stolica Małopolski' )
```

Below the code editor, a message box displays the error: [01000][1265] Data truncated for column 'Współzne\_stolicy\_x' at row 1.

## EdytujDaneWojewództwa

The screenshot shows a database console window with a dark theme. At the top, there is a code editor with the following SQL command:

```
1 ✓ call EdytujDaneWojewództwa( p_IdTerytowe: 42, p_Nazwa: 'dolnośląskie', p_MiastoWojewódzkie: 'Wrocław', p_WspółzneMiastaWojewódzkiegoX: 19.23, p_WspółzneMiastaWojewódzkiegoY: 52.43, p_DrugieMiastoWojewódzkie: null, p_WspółzneDrugiegoMiastaWojewódzkiegoX: null, p_WspółzneDrugiegoMiastaWojewódzkiegoY: null, p_DodatkoweInformacje: '4:48' )
```

Below the code editor, a table viewer displays the updated data for the voivodeships:

	Id_terytowe	Nazwa	MiastoWojewódzkie	WspółzneMiastaWojewódzkiegoX	WspółzneMiastaWojewódzkiegoY	DrugieMiastoWojewódzkie	WspółzneDrugiegoMiastaWojewódzkiegoX	WspółzneDrugiegoMiastaWojewódzkiegoY	DodatkoweInformacje	
1	42	dolnośląskie	Wrocław	19.23	52.43	<null>	<null>	<null>	150	1500 4:48
2	43	Kujawsko-Pomorskie	Bydgoszcz	53.1235	18.0084	Toruń	53.0138	18.5984	17971.4	2085323
3	44	Lubelskie	Lublin	51.2465	22.5684				25122	2142212

## EdytujDanePowiatu

The screenshot shows a database console window with a dark theme. At the top, there is a code editor with the following SQL command:

```
1 ✓ call EdytujDanePowiatu( p_IdTerytowe: 41, p_Nazwa: 'jużNieNie', p_StolicaPowiatu: 'możeDalejTak', p_WspółzneStolicyPowiatuX: 12, p_WspółzneStolicyPowiatuY: 32, p_TypPowiatu: 3, p_Województwo: 42, p_DodatkoweInformacje: null )
```

Below the code editor, a table viewer displays the updated data for the districts:

	Id_terytowe	Nazwa	Stolica_powiatu	WspółzneStolicyPowiatuX	WspółzneStolicyPowiatuY	Po..._L..._Inne_dane	Typ_powiatu
1	43	Niemano	Możowo	12	31	<null> niemoże	46 3
2	41	jużNieNie	możeDalejTak	12	32	1500 <null>	42 3

## EdytujDaneGminy

The screenshot shows a database interface with a query window containing the following code:

```
call EdytujDaneGminy( p_IdTerytowe: 8, p_Nazwa: 'możeNowe', p_StolicaGminy: 'stolnica',
p_WspolrzedneStolicyGminyX: 31, p_WspolrzedneStolicyGminyY: 51, p_TypGminy: 5, p_Powierzchnia: 50,
p_Ludosc: 123, p_Powiat: 41, p_DodatkoweInformacje: null)
```

Below the code, a table is displayed with the following data:

	Id_terytowe	Nazwa	Stolica_gminy	Wspolrzedne_stolicy_x	Wspolrzedne_stolicy_y	Powierzchnia	Ludosc	Inne_dane	Po...	Typ_gminy
1	3	Warszawa	Mokotów	52.2298	21.0118	70	700	Stolica Polski	41	1
2	4	Kraków	Podgórze	50.0614	19.9374	30	300	Stolica Małopolski	41	1
3	8	możeNowe	stolnica	31	51	50	123 <null>		41	5

## Triggery:

Testy triggerów polegają na ich poprawnym uruchamianiu się oraz poprawnych obliczeniach. Każdy test składa się z trzech screenów (kolejno screen akcji uruchamiającej trigger, screen niezaktualizowanych danych, screen po aktualizacji danych).

## after\_insert\_gmina + after\_insert\_gmina\_population

The screenshot shows three database screens illustrating the execution of triggers:

- Screen 1 (Top):** Shows the insertion of a new gmina record ('nowy' with 'nadgórze') into the 'gminy' table.
- Screen 2 (Middle):** Shows the execution of the 'after\_insert\_gmina' trigger, which updates the 'powiaty' table. It shows a row being inserted with 'Id\_terytowe' 41 and 'Nazwa' 'nie'.
- Screen 3 (Bottom):** Shows the execution of the 'after\_insert\_gmina\_population' trigger, which updates the 'powiaty' table again. It shows a row being updated with 'Id\_terytowe' 41 and 'Nazwa' 'tak'.

## after\_update\_gmina + after\_update\_gmina\_population

The screenshot shows a database interface with three tables displayed vertically. The top table has columns: Id\_terytoryowe, Nazwa, ... (repeated), Ludnosc, Inne\_dane, Powiat\_..., Ty... (repeated). It contains two rows: one for Warszawa (Mokotów) with Ludnosc 70 and another for Kraków (Podgórz.) with Ludnosc 30. The middle table has columns: Stolica\_powiatu, Wspolrzedne\_stolicy\_x, Wspolrzedne\_stolicy\_y, Powierzchnia, Ludnosc, Inne\_dane, Wojewodztwo\_w\_którym\_s... (repeated). It contains one row for 'tak' with Ludnosc 40 and Inne\_dane 700. The bottom table has columns: Stolica\_powiatu, Wspolrzedne\_stolicy\_x, Wspolrzedne\_stolicy\_y, Powierzchnia, Ludnosc, Inne\_dane, Wojewodztwo\_w\_którym\_s... (repeated). It also contains one row for 'tak' with Ludnosc 100 and Inne\_dane 1000.

Id_terytoryowe	Nazwa	...	...	Ludnosc	Inne_dane	Powiat_...	Ty...		
3	Warszawa	Mokotów	52.2298	21.0118	70	700	Stolica Polski	41	1
4	Kraków	Podgórz.	50.0614	19.9374	30	300	Stolica Małopolski	41	1

Stolica_powiatu	Wspolrzedne_stolicy_x	Wspolrzedne_stolicy_y	Powierzchnia	Ludnosc	Inne_dane	Wojewodztwo_w_którym_s...
tak	1	2	40	700	<null>	

Stolica_powiatu	Wspolrzedne_stolicy_x	Wspolrzedne_stolicy_y	Powierzchnia	Ludnosc	Inne_dane	Wojewodztwo_w_którym_s...
tak	1	2	100	1000	<null>	

## after\_delete\_gmina + after\_delete\_gmina\_population

The screenshot shows three separate queries or tables in a database interface:

- Top Table:** Shows data from a table with columns: Id\_terytowe, Nazwa, ... (partially visible), Ludnosc, Inne\_dane, Powiat\_..., Typ... (partially visible). Rows include: 3 Warszawa Mokotów 52.2298 21.0118 10 400 Stolica Polski 41 1; 4 Kraków Podgó... 50.0614 19.9374 30 300 Stolica Małopolsk... 41 1; 6 Łódź Bałuty 51.7687 19.456 35 100 Stolica Łódzki... 41 5.
- Middle Table:** WHERE ORDER BY Ludnosc DESC. Shows a single row: Stolica\_powiatu: 1, Wspolrzedne\_stolicy\_x: 1, Wspolrzedne\_stolicy\_y: 2, Powierzchnia: 75, Ludnosc: 1000, Inne\_dane: null, Wojewodztwo\_w\_ktorym\_sie\_nalazi: null.
- Bottom Table:** WHERE ORDER BY Ludnosc DESC. Shows a single row: Stolica\_powiatu: 1, Wspolrzedne\_stolicy\_x: 1, Wspolrzedne\_stolicy\_y: 2, Powierzchnia: 40, Ludnosc: 700, Inne\_dane: null, Wojewodztwo\_w\_ktorym\_sie\_nalazi: null.

## after\_insert\_powiat + after\_insert\_powiat\_population

The screenshot shows three separate queries or tables in a database interface:

- Top Table:** WHERE ORDER BY Wspolrzedne\_stolicy\_y. Shows two rows: 41 Nie tak 1 2 100 1000 null 42 1; 42 Może Niewiemowo 12 13 400 3000 null 42 1.
- Middle Table:** WHERE ORDER BY. Shows three rows of Województwo data: 42 Dolnośląskie Wrocław 51.1079 17.0385 null null 100 1000; 43 Kujawsko-Pomorskie Bydgoszcz 53.1235 18.0084 Toruń 53.0138 18.5984 17971.4 2085323; 44 Lubelskie Lublin 51.2465 22.5684 null null 25122 2142212.
- Bottom Table:** WHERE ORDER BY. Shows three rows of Województwo data: 42 Dolnośląskie Wrocław 51.1079 17.0385 null null 500 4000; 43 Kujawsko-Pomorskie Bydgoszcz 53.1235 18.0084 Toruń 53.0138 18.5984 17971.4 2085323; 44 Lubelskie Lublin 51.2465 22.5684 null null 25122 2142212.

## after\_update\_powiat + after\_update\_powiat\_population

Table: powiaty

Id_terytory	Nazwa	Sto...	...	Powierzchnia	Ludnosc	Inne_dane	Wo...	Typ_powiatu
41	Nie	tak	1	2	100	1000 <null>	42	1
42	Może	Niewiemowo	12	13	100	1000 <null>	42	1

Table: gminy

Id_terytory	Nazwa	Miasto_wojewodzkie	...	W...	Dr...	Wsp...	Wsp...	Po...	...	Inne_dane
41	Nie	Wrocław	51.1079	17.0385		<null>	<null>	500	4000	
42	Może	Bydgoszcz	53.1235	18.0084	Toruń	53.0138	18.5984	17971.4	2085323	
43	Kujawsko-Pomorskie	Bydgoszcz	53.1235	18.0084	Toruń	53.0138	18.5984	17971.4	2085323	
44	Lubelskie	Lublin	51.2465	22.5684		<null>	<null>	25122	2142212	

Table: wojewodztwa

Id_terytory	Nazwa	Miasto_wojewodzkie	...	W...	Dr...	Wsp...	Wsp...	Po...	...	Inne_dane
41	Nie	Wrocław	51.1079	17.0385		<null>	<null>	200	2000	
42	Dolnośląskie	Wrocław	51.1079	17.0385		<null>	<null>	200	2000	
43	Kujawsko-Pomorskie	Bydgoszcz	53.1235	18.0084	Toruń	53.0138	18.5984	17971.4	2085323	
44	Lubelskie	Lublin	51.2465	22.5684		<null>	<null>	25122	2142212	
45	Łódzkie	Łódź	51.7687	19.456		<null>	<null>	18218	2493603	

## after\_delete\_powiat + after\_delete\_powiat\_population

Table: powiaty

Id_terytory	Nazwa	Sto...	...	Powierzchnia	Ludnosc	Inne_dane	Wo...	Typ_powiatu
41	Nie	tak	1	2	100	1000 <null>	42	1
42	Może	Niewiemowo	12	13	100	1000 <null>	42	1

Table: gminy

Id_terytory	Nazwa	Miasto_wojewodzkie	...	W...	Dr...	Wsp...	Wsp...	Po...	...	Inne_dane
41	Nie	Wrocław	51.1079	17.0385		<null>	<null>	200	2000	
42	Dolnośląskie	Wrocław	51.1079	17.0385		<null>	<null>	100	1000	
43	Kujawsko-Pomorskie	Bydgoszcz	53.1235	18.0084	Toruń	53.0138	18.5984	17971.4	2085323	
44	Lubelskie	Lublin	51.2465	22.5684		<null>	<null>	25122	2142212	
45	Łódzkie	Łódź	51.7687	19.456		<null>	<null>	18218	2493603	
46	Małopolskie	Kraków	50.0614	19.9374		<null>	<null>	15182	3412400	

## TESTY WSTAWIANIA WARTOSCI UJEMNYCH:

```
INSERT INTO Gminy (Id_terytowe, Nazwa, Stolica_gminy,  
Wspolrzedne_stolicy_x,  
Wspolrzedne_stolicy_y, Powiat_w_ktorym_sie_znajduje, Typ_gminy,  
Ludnosc)  
VALUES (2, 'Nazwa_powiatu', 'Miasto_powiatowe', 12.34, 56.78, 2, 1,  
-1);
```

```
administrator@jar> INSERT INTO Gminy (Id_terytory, Nazwa, Stolica_gminy, Wspolrzedne_stolicy_x, Wspolrzedne_stolicy_y, Powiat_w_ktorym_sie_znajduje, Typ_gminy, Ludnosc)
VALUES (2, 'Nazwa_powiatu', 'Miasto_powiatowe', 12.34, 56.78, 2, 1)
[2023-12-22 03:00:49] [HY000][3819] Check constraint 'gminy_chk_1' is violated.
[2023-12-22 03:00:49] [HY000][3819] Check constraint 'gminy_chk_1' is violated.
```

## Generowanie danych do tabel:

Dane umieszczone w tabelach beda pobierane z terytu za pomocą poniższego programu:

```

        float randomX = random.nextFloat() + 49 +
randomint;
        float randomintY = random.nextInt(9);
        float randomY = random.nextFloat() + randomintY ;
        int randomStringLength = random.nextInt(1001);
//wojewodztwo
        if(values[1] == ""){
            currentWojewodztwoId =
Integer.parseInt(values[0]);
            String insertWoj = "INSERT INTO Wojewodztwa
(Id_terytowe,Nazwa, Miasto_wojewodzkie,
Wspolrzedne_miasta_wojewodzkiego_x,
Wspolrzedne_miasta_wojewodzkiego_y, " +
                    " Inne_dane) VALUES
(?, ?, ?, ?, ?, ?)";
            PreparedStatement prepareWoj =
connection.prepareStatement(insertWoj);
            String randomString =
generateRandomString(randomStringLength);
            prepareWoj.setString(1, values[0]);
            prepareWoj.setString(2, values[4]);
            prepareWoj.setString(3, miasto);
            prepareWoj.setString(4,
Float.toString(randomX));
            prepareWoj.setString(5,
Float.toString(randomY));
            prepareWoj.setString(6, randomString);

            prepareWoj.addBatch();
            prepareWoj.executeBatch();
        } else if (values[2] == "") {

            String insertPowiat = "INSERT INTO Powiaty
(Id_terytowe,Nazwa, Stolica_powiatu, Wspolrzedne_stolicy_x,
Wspolrzedne_stolicy_y, " +
                    " Inne_dane,
Wojewodztwo_w_ktorym_sie_znajduje, Typ_powiatu) VALUES
(?, ?, ?, ?, ?, ?, ?, ?, ?)";
            PreparedStatement preparePowiat =
connection.prepareStatement(insertPowiat);
            String randomString =
generateRandomString(randomStringLength);
            String idPowiatu = values[0] + values[1];
            currentPowiatId =
Integer.parseInt(idPowiatu);
            String typ_powiatu;
            if(values[5].equals("powiat")) {

```

```

        typ_powiatu = "1";
    } else {
        typ_powiatu = "2";
    }
    preparePowiat.setString(1, idPowiatu);
    preparePowiat.setString(2, values[4]);
    preparePowiat.setString(3, miasto);
    preparePowiat.setString(4,
Float.toString(randomX));
        preparePowiat.setString(5,
Float.toString(randomY));
        preparePowiat.setString(6, randomString);
        preparePowiat.setString(7,
Integer.toString(currentWojewodztwoId));
        preparePowiat.setString(8, typ_powiatu);
        preparePowiat.addBatch();
        preparePowiat.executeBatch();
    } else{
System.out.println("AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA");
        String insertGmina = "INSERT INTO Gminy
(Id_terytowe,Nazwa, Stolica_gminy, Wspolrzedne_stolicy_x,
Wspolrzedne_stolicy_y, " +
                    "Powierzchnia, Ludnosc, Inne_dane,
Powiat_w_ktorym_sie_znajduje, Typ_gminy) VALUES
(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        PreparedStatement prepareGmina =
connection.prepareStatement(insertGmina);
        String randomString =
generateRandomString(randomStringLength);
        int typ_gminy = Integer.parseInt(values[3]);
        if(typ_gminy > 5){
            typ_gminy = typ_gminy - 2;
        }

        String idGminy = values[0] + values[1] +
values[2];

        if(lastIdGminy == Integer.parseInt(idGminy)){
            continue;
        }
        lastIdGminy = Integer.parseInt(idGminy);
        prepareGmina.setString(1, idGminy);
        prepareGmina.setString(2, values[4]);
        prepareGmina.setString(3, miasto);
        prepareGmina.setString(4,

```

```
Float.toString(randomX));
                prepareGmina.setString(5,
Float.toString(randomY));
                prepareGmina.setString(6,
Float.toString(randomPowierzchnia));
                prepareGmina.setString(7,
Integer.toString(randomPopulacja));
                prepareGmina.setString(8, randomString);
                prepareGmina.setString(9,
Integer.toString(currentPowiatId));
                prepareGmina.setString(10,
Integer.toString(typ_gminy));

                prepareGmina.addBatch();
                prepareGmina.executeBatch();
            }
        }

    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    } catch (IOException e) {
        throw new RuntimeException(e);
    } ;
} catch (SQLException e) {
    throw new RuntimeException(e);
}
}

public static String generateRandomString(int length) {
    int leftLimit = 97; // 'a'
    int rightLimit = 122; // 'z'
    Random random = new Random();

    String generatedString = random.ints(leftLimit,
rightLimit + 1)
        .limit(length)
        .collect(StringBuilder::new,
StringBuilder::appendCodePoint, StringBuilder::append)
        .toString();

    return generatedString;
}
```

## Test użytkowników:

```
[janek@localhost ~]$ mysql -u adminDanych -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.32 Source distribution

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use administracjaRP
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> create table test (
    -> id int (10) not null auto_increment,
    -> nazwa varchar (10),
    -> primary key (id));
Query OK, 0 rows affected, 1 warning (0.63 sec)

mysql> alter table test
    -> rename column test to zbrutalizowanyAdmin
    -> ;
ERROR 1054 (42S22): Unknown column 'test' in 'test'
mysql> alter table test rename column nazwa to mamTogdzieś;
Query OK, 0 rows affected (0.25 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> drop table test;
Query OK, 0 rows affected (0.20 sec)

mysql>
```

Uprawnienia administratora danych działają poprawnie dla schematu administracjaRP

```
[janek@localhost ~]$ mysql -u gosc
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.32 Source distribution

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show database
-> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL s
ase' at line 1
mysql> show database;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL s
ase' at line 1
mysql> show databases;
+-----+
| Database      |
+-----+
| administracjaRP |
| information_schema |
| performance_schema |
+-----+
3 rows in set (0.00 sec)

mysql> use administracjaRP;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> create view widokTest as
-> select Nazwa,
->       Inne_dane
->     from administracjaRP.Gminy;
ERROR 1142 (42000): CREATE VIEW command denied to user 'gosc'@'localhost' for table 'widokTest'
mysql> drop view administracjaRP.WidokInformacjiGminInne;
ERROR 1142 (42000): DROP command denied to user 'gosc'@'localhost' for table 'WidokInformacjiGminInne'
mysql> select * from Gminy;
Empty set (0.29 sec)

mysql> _
```

Uprawnienia gościa nie pozwalają mu na edycje danych w bazie danych.

### **Instalacja i konfigurowanie systemu: Administracja RP**

- Zalecane jest stworzenie wirtualnego środowiska
- Sklonuj repozytorium na komputer/server hostujący aplikacje
- w terminalu użyj komendy `pip install django mysqlclient django-crispy-forms django-bootstrap4`
- Aby stworzyć użytkownika z uprawnieniami administratora wejdź do folderu z aplikacją i w terminalu wpisz `python .\manage.py createsuperuser`
- Uzupełnij wymagane dane
- Aby uruchomić aplikację: `python .\manage.py runserver`
- Aplikacja połączy się z naszymi serverami automatycznie.
- Aplikacja powinna działać na zdefiniowanym w kontrakcie ip/domenie

## Instrukcja użytkowania aplikacji- administrator bazy danych:

Do nadzoru aplikacji sugerujemy wykorzystanie panelu administratora, który dostępny jest dla użytkownika pod adresem: `ip/admin` lub nazwadomoeny internetowej `/admin`. Dostęp wymaga zalogowania za pomocą danych, podanych podczas tworzenia użytkownika z uprawnieniami administratora.

The image shows two screenshots of a Windows desktop environment displaying the Django administration interface. The top screenshot shows the 'Log in | Django site admin' window, which includes fields for 'Username' and 'Password' and a 'Log in' button. The bottom screenshot shows the 'Site administration | Django site' dashboard, featuring a sidebar with links like 'APLIKACJA' (Gminy, Powiaty, Województwa), 'AUTHENTICATION AND AUTHORIZATION' (Groups, Users), and 'Recent actions' and 'My actions' sections. The status bar at the bottom of both windows indicates the URL as `127.0.0.1:8000/admin/`.

Po zalogowaniu administrator ma dostęp do panelu za pomocą, którego może edytować danymi "zaciągniętymi" z bazy danych przez aplikację, dodawać nowe oraz edytować i usuwać istniejące.

The screenshots illustrate the Django admin interface for managing gminy (municipalities) within an application.

**Screenshot 1: Select gminy to change**

This screen shows a list of gminy to change. The sidebar includes sections for APLIKACJA (Gminys, Powiatys, Wojewodztwa) and AUTHENTICATION AND AUTHORIZATION (Groups, Users). The main area lists gminy with checkboxes:

- GMINY
- Świnoujście
- Szczecin
- Koszalin
- Węgorzyno
- Resko
- Radowo Małe
- Lobez
- Dobra
- Walcz
- Tuczno
- Miroslawiec
- Człopa
- Walcz
- Świdwin
- Slawoborze
- Rabino
- Polczyn-Zdrój

**Screenshot 2: Change gminy - Świnoujście**

This screen shows the detailed edit form for the Świnoujście gmina. The form fields are:

- Nazwa: Świnoujście
- Stolica gminy: miasto4263
- Współrzędne stolicy x: 52,5538
- Współrzędne stolicy y: 8,24753
- Powierzchnia: 12,0628
- Ludność: 4917
- Inne dane: tpmonospilkzbxncmwpxdlqogpbhexuzwec
- Powiat w którym się znajduje: Świnoujście
- Typ gminy: gmina miejska

Action buttons at the bottom include: SAVE, Save and add another, Save and continue editing, and Delete.

Administrator może również za pomocą aplikacji dodawać, usuwać oraz edytować użytkowników (kolejnych administratorów, administratorów danych bądź zdefiniowanych przez siebie) oraz przypisywać im grupy z odpowiednimi uprawnieniami.

- Dodawanie użytkowników:

- Edytowanie i usuwanie użytkowników:

The screenshot shows the Django admin interface for managing user permissions. On the left, there's a sidebar with various application icons. The main content area has two tabs: 'User permissions' and 'Chosen user permissions'. Under 'Available user permissions', there is a list of numerous specific permissions such as 'admin | log entry | Can add log entry', 'admin | log entry | Can change log entry', etc. Under 'Chosen user permissions', there is a smaller list of selected permissions. Below these tabs, there are sections for 'Important dates' showing 'Last login' (Date: 2024-01-17, Time: 12:46:08) and 'Date joined' (Date: 2024-01-15, Time: 12:57:44). At the bottom, there are buttons for 'SAVE', 'Save and add another', 'Save and continue editing', and a red 'Delete' button.

- Tworzenie grup:

The screenshot shows the Django admin interface for managing groups. The sidebar on the left is identical to the previous screenshot. The main content area is titled 'Select group to change' and features a search bar at the top. Below the search bar, there's a section for 'Action:' with a dropdown menu and a 'Go' button. Underneath, there are checkboxes for 'GROUP' and 'admin\_danych'. A summary at the bottom indicates '1 group'. There's also an 'ADD GROUP +' button in the top right corner.

- Edycja i usuwanie grup

Django administration

Home > Authentication and Authorization > Groups > admin danych

Change group

**admin danych**

Name: admin danych

Permissions:

Available permissions

Chosen permissions

Choose all

Remove all

SAVE Save and add another Save and continue editing Delete

## Instrukcja użytkowania aplikacji- administrator danych

Konta administratorów danych będą tworzone przez administratora bazy danych, z użyciem panelu administracyjnego. Hasło użytkownika musi mieć więcej niż 8 znaków w tym znaki i litery.

Aby zalogować się jako administrator danych, użytkownik na stronie głównej musi nacisnąć przycisk zaloguj:

Strona Główna

Przeglądaj dane jako gość

Zaloguj na konto administratora

Po naciśnięciu przycisku będzie musiał podać swoje dane uwierzytelniające:

A screenshot of a web-based login form. It consists of two input fields: 'Username\*' containing 'adminDanych' and 'Password\*' containing '.....'. Below the fields is a 'Login' button.

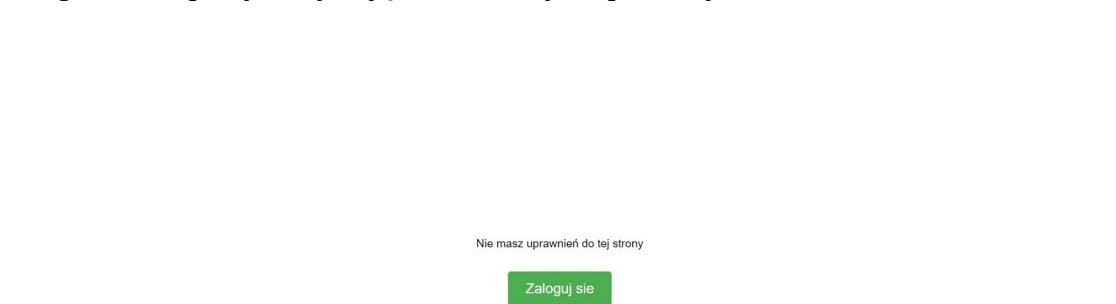
W

przypadku podania złych danych zostanie wyświetlony komunikat:

- Please enter a correct username and password. Note that both fields may be case-sensitive.

A screenshot of a web-based login form. The 'Username\*' field contains 'adminDanych' and the 'Password\*' field is empty. Below the fields is a 'Login' button. A red validation message is displayed above the password field: 'Please enter a correct username and password. Note that both fields may be case-sensitive.'

Jeśli ktoś chciałby uzyskać dostęp do panelu administratora danych, bez wcześniejszego zalogowania np. wykorzystując URL, otrzyma poniższy komunikat:



Po zalogowaniu administrator danych będzie mieć dostęp do poniższego panelu:

Wybierz katergorię do edycji:

Wyszukaj dane:

Administrator danych może dodawać Gminy, Powiaty oraz Województwa za pomocą odpowiedniego przycisku.

W celu edytowania lub usunięcia danych administrator musi wybrać jaki obiekt usunąć. **W tym przykładzie posłużymy się powiatami, jednak proces jest taki sam dla pozostałych jednostek terytorialnych!**

Po wybraniu powiatów pokazuje nam się lista wszystkich powiatów:

## **Lista Powiatów**

- [bolesławiecki](#)
- [dzierżoniowski](#)
- [głogowski](#)
- [górowski](#)
- [jaworski](#)
- [karkonoski](#)
- [kamiennogórski](#)
- [kłodzki](#)
- [legnicki](#)
- [lubański](#)
- [lubiński](#)
- [lwówecki](#)
- [milicki](#)
- [oleśnicki](#)
- [oławski](#)
- [polkowicki](#)
- [strzeliński](#)
- [średzki](#)
- [świdnicki](#)
- [trzebnicki](#)
- [wałbrzyski](#)
- [wołowski](#)
- [wrocławski](#)
- [ząbkowicki](#)
- [zgorzelecki](#)
- [złotoryjski](#)
- [Jelenia Góra](#)
- [Legnica](#)
- [Wrocław](#)
- [Wałbrzych](#)
- [aleksandrowski](#)
- [brodnicki](#)
- [bydgoski](#)
- [chełmiński](#)
- [golubsko-dobrzyński](#)
- [grudziądzki](#)
- [inowrocławski](#)
- [lipnowski](#)
- [mogileński](#)
- [nakielski](#)
- [radziejowski](#)
- [typiński](#)
- [sępoleński](#)
- [świecki](#)
- [toruński](#)
- [tucholski](#)
- [wąbrzeski](#)
- [włocławski](#)
- [żniński](#)

Następnie wybieramy interesujący nas powiat i wyświetla się nam panel edycji z opcją usunięcia

**Edycja Powiatu**

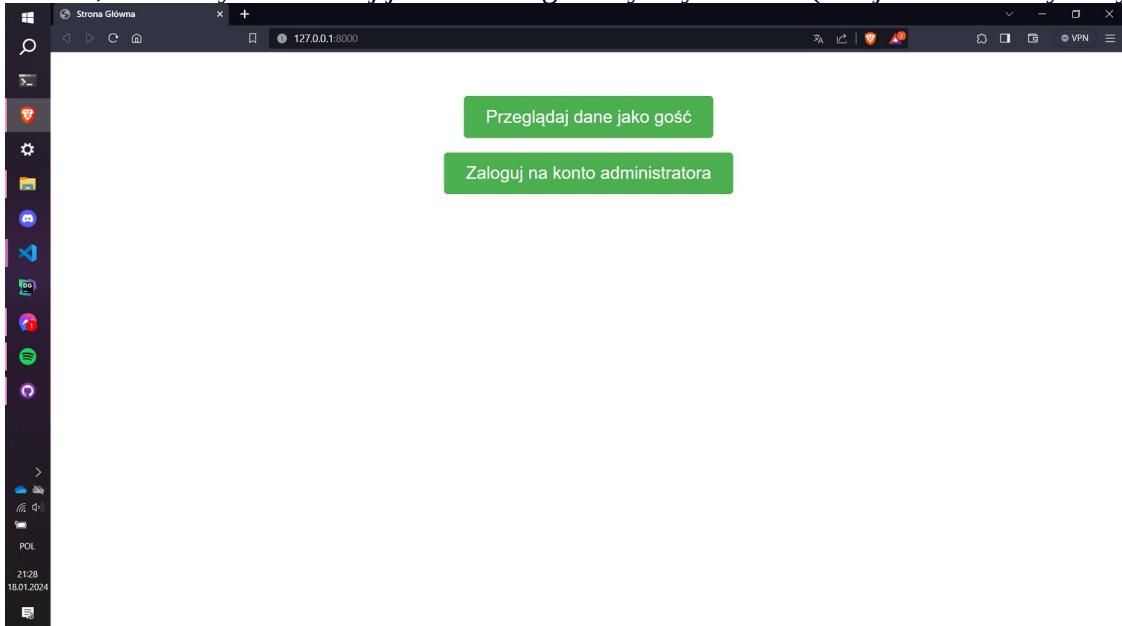
Nazwa:	mogileński
Stolica powiatu:	miasto420
Współrzędne stolicy x:	52.6973
Współrzędne stolicy y:	7.26827
Powierzchnia:	53.2054
Ludność:	16184
Inne dane:	xzqzraawxxmxdhlyhptpzta;
Województwo w którym się znajduje:	KUJAWSKO-POMORSKIE
Typ powiatu:	powiat

**Usuń** **Zapisz**

obiektu: \_\_\_\_\_

### Instrukcja użytkowania aplikacji- niezalogowany użytkownik

Po wpisaniu nr ip/nazwy domeny internetowej użytkownik zostanie skierowany na stronę główną. Na stronie głównej użytkownik będzie miał opcje zalogowania się, jeśli jest pracownikiem, lub korzystania dalej jako niezalogowany użytkownik (dalej również nazywany



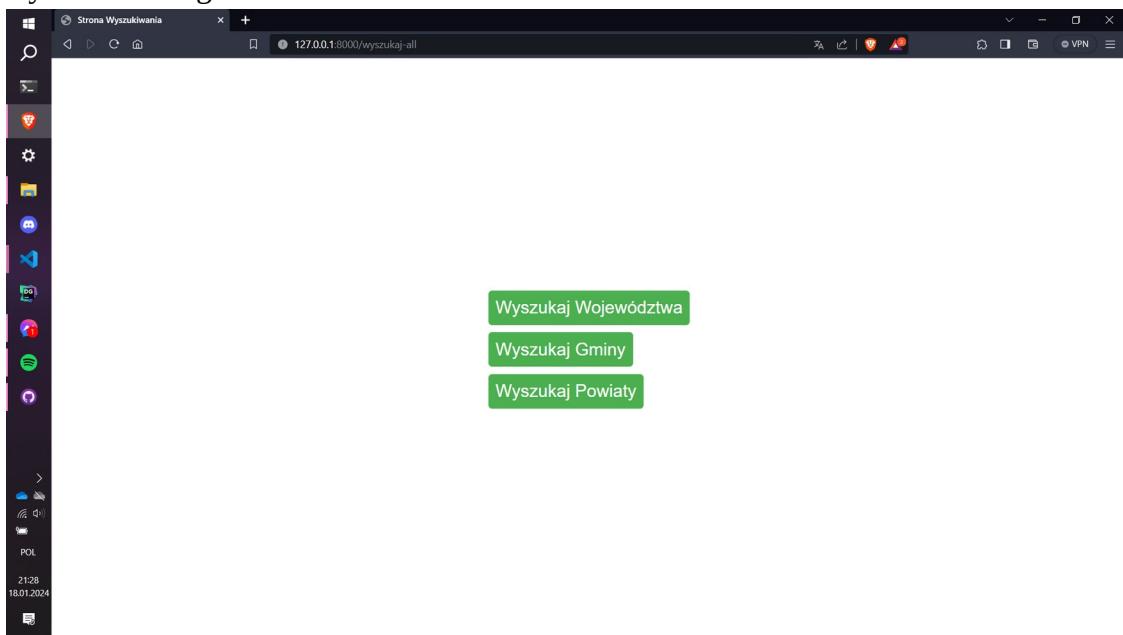
NZU).

Jako NZU, użytkownik będzie miał do wyboru:

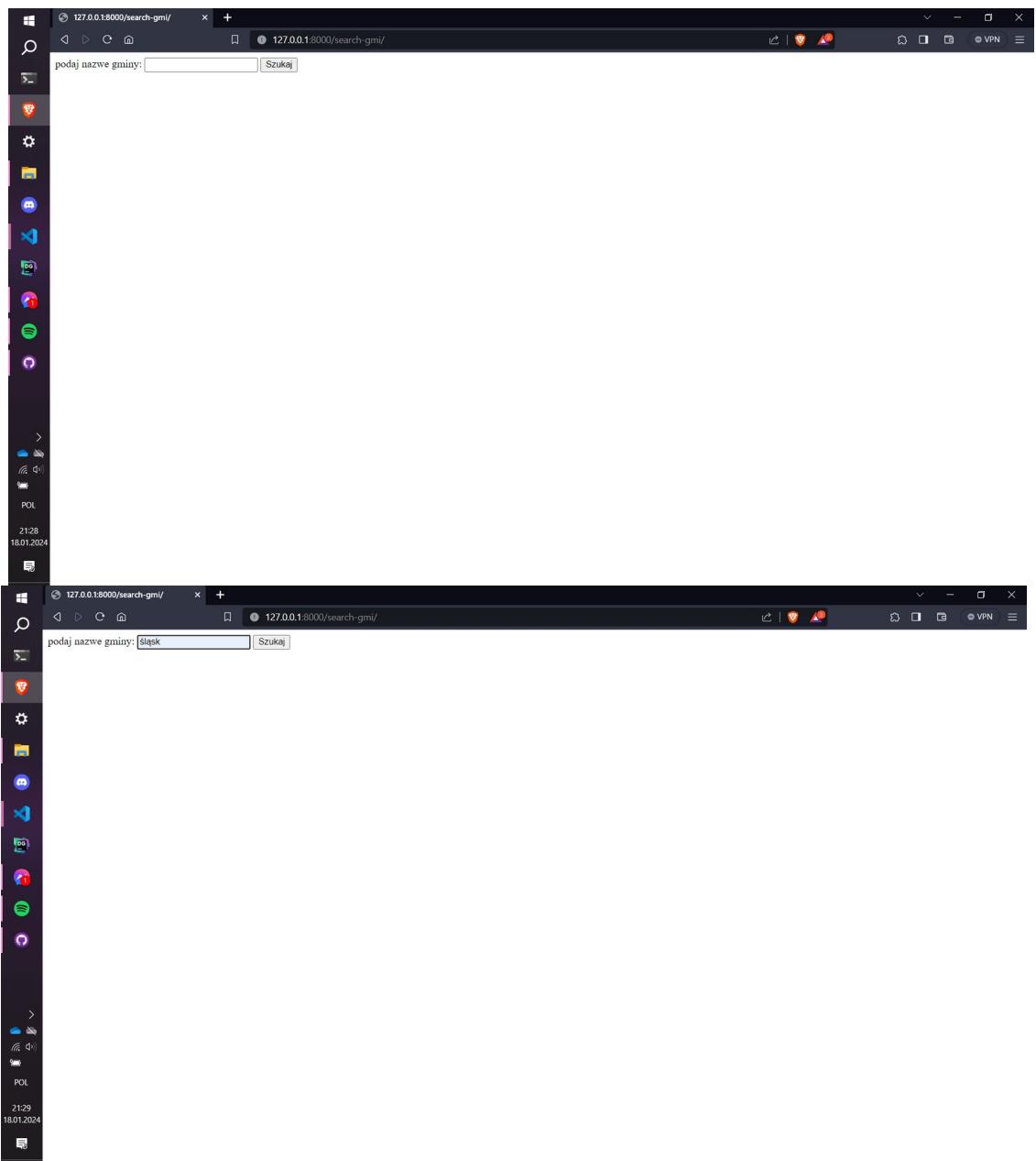
### Funkcjonalności dla niezalogowanego użytkownika

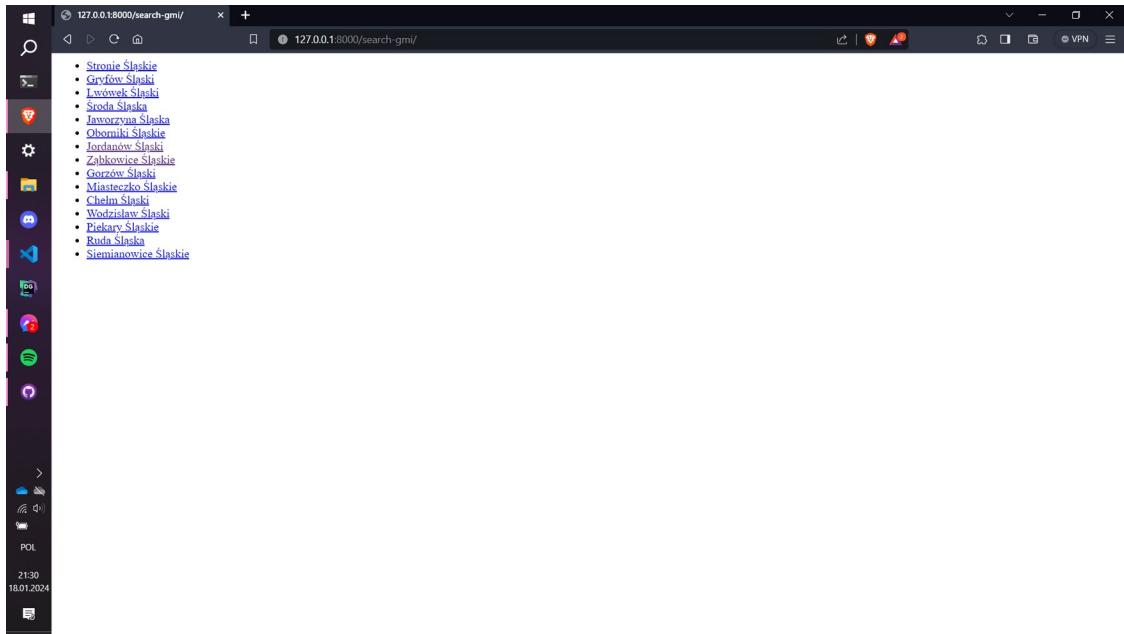
- Wyszukiwanie województw

- Wyszukiwanie powiatów
- Wyszukiwanie gmin



**Dla przykładu zaprezentowano opcje wyszukania Gminy. Dla wyszukiwania powiatów i województw kroki będą jednakowe**

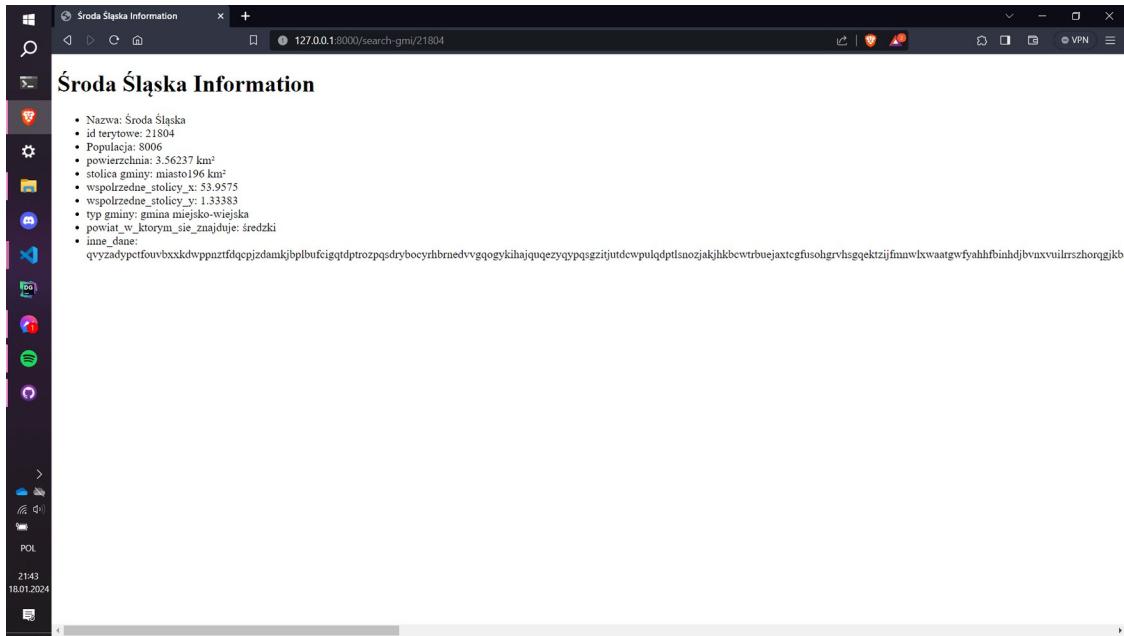




Po wyszukaniu interesujących użytkownika danych po nazwie, użytkownik może sprawdzić szczegóły interesującego go wyniku np.: dla gminy będzie to:

- nazwa
- id z terytu
- populacja
- powierzchnia
- powiat w którym się znajduje
- stolica gminy
- typ gminy
- współrzędne geograficzne x oraz y stolicy gminy

- inne dane



Dla powiatu użytkownik będzie miał podgląd do informacji takich jak:

- nazwa
- id z terytu
- stolica powiatu
- współrzędne geograficzne x oraz y dla miasta powiatowego
- powierzchnia powiatu
- ludność powiatu
- typ powiatu
- województwo, w którym znajduje się powiat
- inne dane

Dla województwa użytkownik będzie miał podgląd do informacji takich jak::

- nazwa
- id z terytu
- miasto wojewódzkie
- współrzędne geograficzne x oraz y dla miasta wojewódzkiego
- powierzchnia województwa
- ludność województwa
- drugie miasto wojewódzkie
- współrzędne geograficzne x oraz y dla drugiego miasta wojewódzkiego
- inne dane

NZU nie będzie mieć dostępu do funkcji modyfikujących zawartość bazy danych przez względy bezpieczeństwa. Zmiany do bazy danych mogą wprowadzać tylko uprawnieni pracownicy.

## Testowanie opracowanych funkcji systemu

wszystkie funkcjonalności użytkownika zostały uważnie przetestowane, przez światowej klasy testerów manualnych specjalistów od cyberbezpieczeństwa. Po każdej aktualizacji i dodaniu nowej funkcjonalności aplikacja była testowana na losowo wybranej grupie badawczej.

## Omówienie rozwiązań programistycznych

Aplikacja korzysta ze standardowej struktury projektu Django.

Wybraliśmy architekturę MVT (stosowaną przez Django) i różniącą się w swoich założeniach od architektury MVC. (widoki skupione na logice, a nie prezentacji danych etc.)

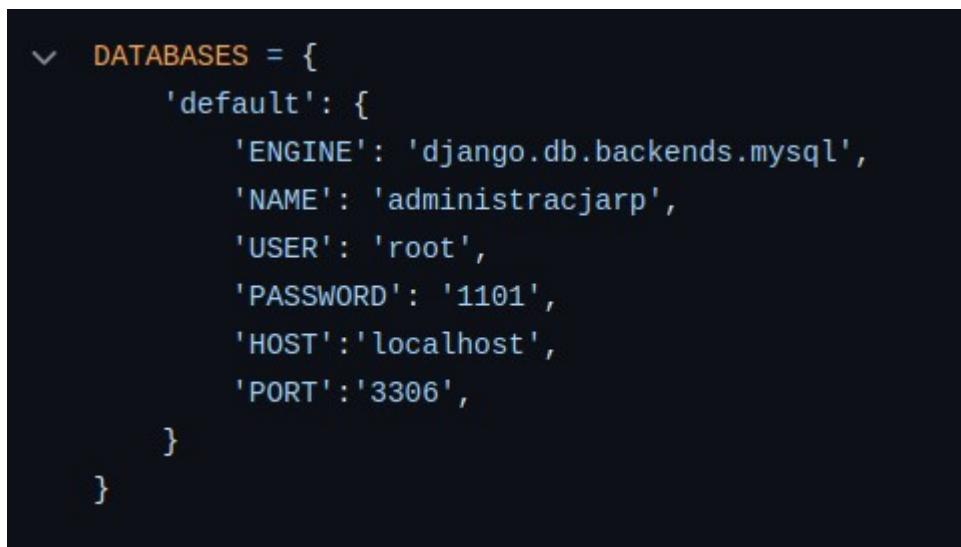
W części aplikacji dostępnej dla NZU, nie zdecydowaliśmy się na dynamiczne przekierowywanie użytkownika.

Aplikacja nie korzysta z zewnętrznych zależności, nie licząc wcześniej wspomnianego mysqlclient. Zapewnia to większe bezpieczeństwo, poprzez minimalizację ryzyka wystąpienia błędów lub szkodliwego oprogramowania.

## Implementacja interfejsu dostępu do bazy danych

Do połączenia się z bazą danych wykorzystaliśmy wbudowany w Django 'connector' do baz MySQL, jednak by móc z niego skorzystać musieliśmy pobrać pakiet mysqlclient, który dostarcza niskopoziomowy interfejs do komunikacji z bazą danych MySQL.

Połączenie z bazą danych odbyło się w bardzo prosty sposób poprzez ustawienie odpowiednich wartości w słowniku DATABASES, przechowywanym w pliku z ustawieniami aplikacji Django.



```
▼  DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'administracjarp',
        'USER': 'root',
        'PASSWORD': '1101',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

Modele  
Django

wykorzystywane do reprezentacji obiektów z bazy danych wygenerowaliśmy za pomocą komendy wbudowanej w Django `python manage.py inspectdb`.

```
    class Wojewodztwa(models.Model):
        id_terytowe = models.AutoField(db_column='Id_terytowe', primary_key=True) # Field name made lowercase.
        nazwa = models.CharField(db_column='Nazwa', max_length=50) # Field name made lowercase.
        miasto_wojewodzkie = models.CharField(db_column='Miasto_wojewodzkie', max_length=50) # Field name made lowercase.
        wspolrzedne_miasta_wojewodzkiego_x = models.FloatField(db_column='Wspolrzedne_miasta_wojewodzkiego_x') # Field name made lowercase.
        wspolrzedne_miasta_wojewodzkiego_y = models.FloatField(db_column='Wspolrzedne_miasta_wojewodzkiego_y') # Field name made lowercase.
        drugie_miasto_wojewodzkie = models.CharField(db_column='Dругие_миasta_wojewodzkie', max_length=50, blank=True, null=True) # Field name made lowercase.
        wspolrzedne_druziego_miasta_wojewodzkiego_x = models.FloatField(db_column='Wspolrzedne_druziego_miasta_wojewodzkiego_x', blank=True, null=True) # Field name made lowercase.
        wspolrzedne_druziego_miasta_wojewodzkiego_y = models.FloatField(db_column='Wspolrzedne_druziego_miasta_wojewodzkiego_y', blank=True, null=True) # Field name made lowercase.
        powierzchnia = models.FloatField(db_column='Powierzchnia', blank=True, null=True) # Field name made lowercase.
        ludnosc = models.IntegerField(db_column='Ludnosc', blank=True, null=True) # Field name made lowercase.
        inne_dane = models.CharField(db_column='Inne_dane', max_length=10000, blank=True, null=True) # Field name made lowercase.

    def __str__(self):
        return self.nazwa

    class Meta:
        db_table = 'wojewodzta'
```

```
    class Powiaty(models.Model):
        id_terytowe = models.AutoField(db_column='Id_terytowe', primary_key=True) # Field name made lowercase.
        nazwa = models.CharField(db_column='Nazwa', max_length=50) # Field name made lowercase.
        stolica_powiatu = models.CharField(db_column='Stolica_powiatu', max_length=50) # Field name made lowercase.
        wspolrzedne_stolicy_x = models.FloatField(db_column='Wspolrzedne_stolicy_x') # Field name made lowercase.
        wspolrzedne_stolicy_y = models.FloatField(db_column='Wspolrzedne_stolicy_y') # Field name made lowercase.
        powierzchnia = models.FloatField(db_column='Powierzchnia', blank=True, null=True) # Field name made lowercase.
        ludnosc = models.IntegerField(db_column='Ludnosc', blank=True, null=True) # Field name made lowercase.
        inne_dane = models.CharField(db_column='Inne_dane', max_length=10000, blank=True, null=True) # Field name made lowercase.
        wojewodztwo_w_ktorym_sie_znajduje = models.ForeignKey('Wojewodztwa', db_column='Wojewodztwo_w_ktorym_sie_znajduje', on_delete=models.CASCADE) # Field name made lowercase.
        typ_powiatu = models.ForeignKey('Typy_Powiatow', db_column='Typ_powiatu', on_delete=models.CASCADE) # Field name made lowercase.

    def __str__(self):
        return self.nazwa

    class Meta:
        db_table = 'powiaty'
```

```
class Gminy(models.Model):
    id_terytowe = models.AutoField(db_column='Id_terytowe', primary_key=True) # Field name made lowercase.
    nazwa = models.CharField(db_column='Nazwa', max_length=50) # Field name made lowercase.
    stolica_gminy = models.CharField(db_column='Stolica_gminy', max_length=50) # Field name made lowercase.
    wspolrzedne_stolicy_x = models.FloatField(db_column='Wspolrzedne_stolicy_x') # Field name made lowercase.
    wspolrzedne_stolicy_y = models.FloatField(db_column='Wspolrzedne_stolicy_y') # Field name made lowercase.
    powierzchnia = models.FloatField(db_column='Powierzchnia', blank=True, null=True) # Field name made lowercase.
    ludnosc = models.IntegerField(db_column='Ludnosc', blank=True, null=True) # Field name made lowercase.
    inne_dane = models.CharField(db_column='Inne_dane', max_length=10000, blank=True, null=True) # Field name made lowercase.
    powiat_w_ktorym_sie_znajduje = models.ForeignKey('Powiaty', db_column='Powiat_w_ktorym_sie_znajduje', on_delete=models.CASCADE) # Field name made lowercase.
    typ_gminy = models.ForeignKey('TypGminy', db_column='Typ_gminy', on_delete=models.CASCADE) # Field name made lowercase.

    def __str__(self):
        return self.nazwa

    class Meta:
        db_table = 'gminy'
```

```
class TypGminy(models.Model):
    id = models.AutoField(db_column='Id', primary_key=True) # Field name made lowercase.
    typ = models.CharField(db_column='Typ', unique=True, max_length=50) # Field name made lowercase.

    def __str__(self):
        return self.typ

    class Meta:
        db_table = 'typ_gminy'

class Typy_Powiatow(models.Model):
    id = models.AutoField(db_column='Id', primary_key=True) # Field name made lowercase.
    typ = models.CharField(db_column='Typ', unique=True, max_length=50) # Field name made lowercase.

    def __str__(self):
        return self.typ

    class Meta:
        db_table = 'typy_powiatow'
```

Za pomocą naszej aplikacji można wykonywać wszystkie operacje CRUD (Create, Read, Update, Delete).

### Mechanizmy bezpieczeństwa

W naszej aplikacji wykorzystaliśmy domyślne zabezpieczenia Django takie jak:

- sprawdzenie czy hasło ma min. 8 znaków
- czy hasło nie jest w bazie najpopularniejszych haseł
- czy hasło nie składa się z samych cyfr
- czy hasło nie pokrywa się za bardzo z atrybutami użytkownika

```
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

Dodatkowo korzystamy również z wbudowanych w Django funkcjonalności takich jak:

- **Cross Site Scripting (XSS) Protection:** Django szablonuje silnik, który domyślnie zabezpiecza przed większością ataków XSS, poprzez automatyczne zabezpieczanie wyjścia szablonów.

- **Cross Site Request Forgery (CSRF) Protection:** Django ma wbudowane zabezpieczenia przeciwko atakom CSRF. Wymaga to użycia odpowiedniego znacznika w każdym formularzu, który prowadzi do zmiany danych.
- **SQL Injection Protection:** Django chroni przed atakami SQL Injection poprzez użycie ORM (Object-Relational Mapping). Każde zapytanie do bazy danych jest automatycznie zabezpieczane.
- **Clickjacking Protection:** Django ma wbudowane zabezpieczenia przeciwko atakom Clickjacking, poprzez użycie middleware X-Frame-Options.
- **Host Header Validation:** Django chroni przed atakami, które mogą prowadzić do niebezpiecznego zachowania poprzez manipulację nagłówkiem hosta.
- **Secure Password Handling:** Django używa bezpiecznych algorytmów do przechowywania haseł. Hasła są przechowywane jako "hash", co oznacza, że oryginalne hasło nie jest zapisywane i nie może być odzyskane.
- **HTTPS Support:** Django obsługuje użycie HTTPS, co jest ważne dla bezpiecznej komunikacji w Internecie oraz ułatwia migrację strony z http na https.

## Wnioski:

1. Projekt wyraźnie pokazuje, że Django jest potężnym narzędziem do szybkiego tworzenia aplikacji webowych opartych na bazie danych. Umożliwia ono zdefiniowanie modeli danych, zarządzanie nimi oraz łatwe tworzenie interfejsu użytkownika.
2. Zastosowanie Bootstrapa i HTML przyczyniło się do stworzenia responsywnego interfejsu użytkownika. Dzięki temu aplikacja jest łatwa w obsłudze zarówno na komputerze.
3. Poprawna implementacja tabel województw, gmin i powiatów w bazie danych MySQL 8 umożliwia efektywne przechowywanie i zarządzanie informacjami administracyjnymi. Stworzenie odpowiednich relacji pomiędzy tymi tabelami pozwala na efektywne zapytania i uzyskiwanie potrzebnych danych.
4. Django, jako framework, zapewnia wbudowane mechanizmy bezpieczeństwa, takie jak ochrona przed atakami SQL Injection czy Cross-Site Scripting (XSS). Warto jednak stale monitorować aktualizacje bezpieczeństwa oraz dostosować aplikację do najlepszych praktyk w zakresie bezpieczeństwa.
5. Projekt pozostawia otwarte drzwi dla rozwoju poprzez dodawanie nowych funkcji. Na przykład, można rozważyć dodanie funkcji wyszukiwania, filtrowania czy generowania raportów, co z pewnością zwiększyłoby użyteczność aplikacji.
6. Implementacja testów pozwoliła na przeanalizowanie poprawności napisanych skryptów jak i działania samej aplikacji dostępowej.

## **Kontakt**

**W razie jakichkolwiek problemów z działaniem bazy danych, proszę skontaktować się z działem wsparcia:**

**Nr tel: 123 123 123 koszt według stawek operatora.**

**Email: [administracjaRP@nie.ponosimy.odpowiedzialnosci.za.nic.com](mailto:administracjaRP@nie.ponosimy.odpowiedzialnosci.za.nic.com)**

**Link do repozytorium:**

**<https://github.com/213N10/BazyDanych2>**

## **Źródła:**

- <https://www.wikipedia.org/>
- [https://www.youtube.com/watch?v=z4lfVsb\\_7MA&ab\\_channel=TechWithTim](https://www.youtube.com/watch?v=z4lfVsb_7MA&ab_channel=TechWithTim)
- <https://chat.openai.com/>
- <https://www.djangoproject.com/>
- <https://youtu.be/UmljXZIypDc?si=Zmc-3eotImx1SLTM>