



**POLITECHNIKA  
RZESZOWSKA**  
im. IGNACEGO ŁUKASIEWICZA



**Katedra  
Informatyki i Automatyki**  
Politechnika Rzeszowska

# **Bazy Danych**

## **Dokumentacja Projektu**

**pt.: „Implementacja bazy danych dla szkółki ogrodniczej”**

**Data wykonania: 17.12.2020**

**Grupa: L02**  
**MIZERA DAWID**  
**Nr albumu 161885**  
**III EF-ZI**

## **Spis treści**

Spis treści .....	2
1. Cel pracy .....	3
2. Przebieg Pracy .....	4
2.1. Zaprojektowanie bazy danych.....	4
2.2. Zaprezentowanie wykorzystanych procedur i funkcji .....	5
3. Wnioski .....	9
4. Spis rysunków i tabel .....	9

## **1. Cel pracy**

Celem projektu było zaprojektowanie i zaimplementowanie bazy danych dla prywatnej firmy, szkółki ogrodniczej której zakresem działalności jest hodowla i handel różami ogrodowymi. Firma posiada zaplecze magazynowe i własny sklep internetowy przez który prowadzi sprzedaż.

### **Funkcja bazy danych**

Baza danych ma przechowywać informacje o klientach, zamówieniach oraz sprzedawanym i zmagazynowanym asortymencie. Baza danych ma za zadanie umożliwić sprawne zarządzanie posiadanymi danymi oraz wspomóc działanie działu sprzedaży. Jej priorytetem jest niezawodność działania, zapewnienie bezpieczeństwa przechowywanym danym oraz prostota obsługi.

### **Technologia i narzędzia**

Bazę danych zbudowano w oparciu o MySQL. Wykorzystano phpMyAdmin dostępny w pakiecie XAMPP.

### **Repozytorium**

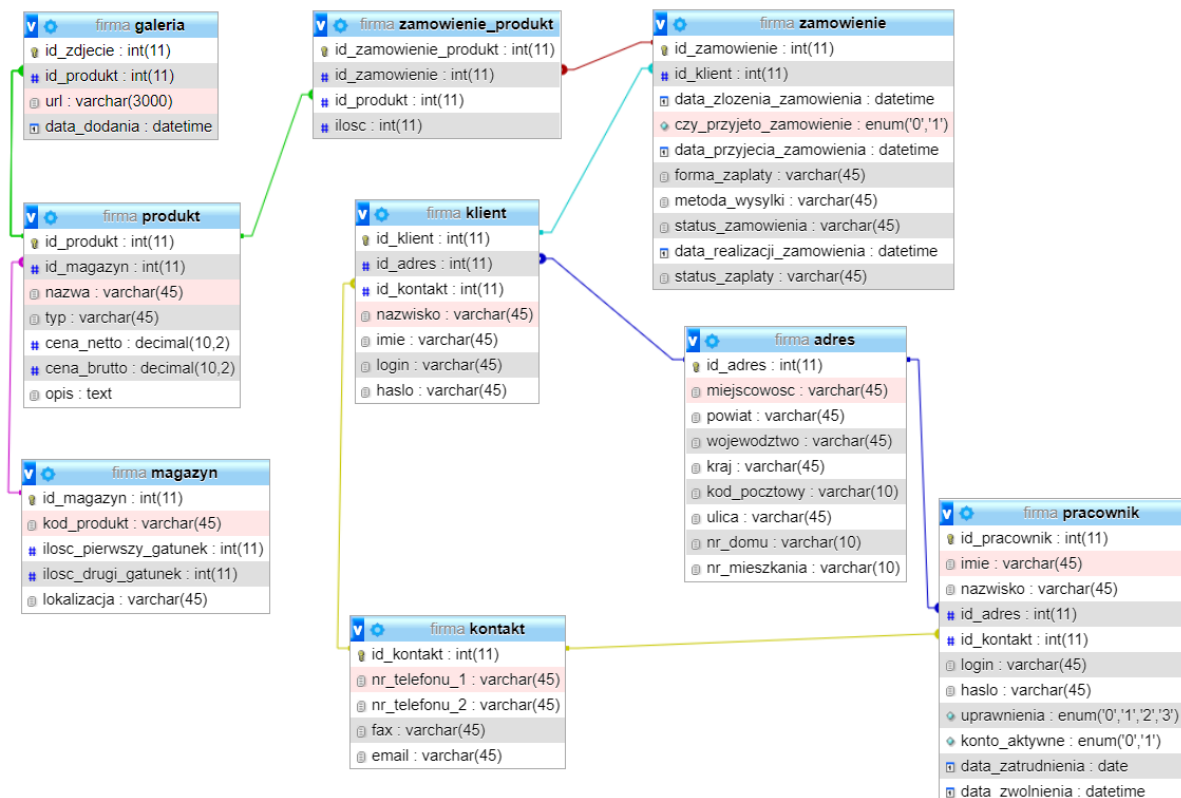
Repozytorium znajduje się na stronie internetowej github.

<https://github.com/Brom7/ProjektBazyDanych>

## 2. Przebieg Pracy

### 2.1 Zaprojektowanie bazy danych

Po ustaleniu początkowych założeń projektu, następnym krokiem było zaprojektowanie bazy danych. W tym celu stworzyłem diagram ERD w którym zawarte zostały zawartości tabel i relacje pomiędzy nimi.



**Rysunek 1 Diagram ERD**

Omówienie zawartości tabel i relacji pomiędzy nimi:

**galeria:** Tabela przechowuje informacje o zdjęciach. Zawiera klucz obcy id\_produkty z tabeli produkt.

**produkt:** Tabela zawiera informacje o konkretnym produkcie w asortymencie sklepu, jego cenę i krótki opis. Zawiera klucz obcy id\_magazyn z tabeli magazyn

**magazyn:** Tabela zawiera informacje o ilości dostępnego asortymentu w magazynie z rozróżnieniem na pierwszy i drugi gatunek oraz jego położeniu.

**zamowienie\_produkty:** Tabela zawiera informacje o ilości produktów które zostały zamówione przez klienta w konkretnym zamówieniu. Zawiera klucze obce: id\_produkty z tabeli produkt oraz id\_zamowienie z tabeli zamowienie.

**zamowienie:** Tabela zawiera informacje o konkretnym zamówieniu, date jego przyjęcia i realizacji, metodę płatności i sposobie realizacji. Zawiera klucz obcy id\_klient z tabeli klient.

**klient:** Tabela zawiera podstawowe informacje o kliencie. Zawiera klucze obce: id\_kontakt z tabeli kontakt oraz id\_adres z tabeli adres.

**adres:** Tabela zawiera dane adresowe.

**kontakt:** Tabela zawiera dane kontaktowe.

**pracownik:** Tabela zawiera dane personalne pracownika, jego uprawnienia w systemie. Zawiera klucze obce: id\_kontakt z tabeli kontakt oraz id\_adres z tabeli adres.

Następnym etapem było stworzenie tabel oraz wypełnienie ich danymi.  
Ostatnim etapem było napisanie procedur i funkcji których zadaniem jest obsługa bazy.

## 2.2 Zaprezentowanie wybranych wykorzystanych procedur i funkcji.

**Nazwa funkcji:** Calkowita\_ilosc\_odmiany\_w\_magazynie

**Parametry:** Atrybut nazwa z tabeli produkt.

**Wynik:** Funkcja która jako wynik zwraca sumę ilości pierwszego i drugiego gatunku wybranego produktu.

**Zasada działania:**

Deklaracja zmiennych wynik, pierwszy\_gatunek, drugi\_gatunek, id

Pobranie z tabeli produkt wartości pola id\_magazyn który jest przypisany do wpisanej nazwy produktu.

Pobranie z tabeli magazyn wartości pól ilosc\_pierwszy\_gatunek oraz ilosc\_drugi\_gatunek które są przypisane do wartości pola id\_magazyn.

Wykonanie działania dodawania i zwrócenie wyniku.

Eksportuj procedurę `Calkowita\_ilosc\_odmiany\_w\_magazynie`

```
1 DELIMITER $$
2 CREATE DEFINER='root'@'localhost' FUNCTION `Calkowita_ilosc_odmiany_w_magazynie`(`nazwa` VARCHAR(45)) RETURNS int(11)
3 NO SQL
4 BEGIN
5 DECLARE wynik INT DEFAULT 0;
6 DECLARE pierwszy_gatunek INT DEFAULT 0;
7 DECLARE drugi_gatunek INT DEFAULT 0;
8 DECLARE id INT DEFAULT 0;
9 SET id =(SELECT id_magazyn FROM produkt WHERE produkt.nazwa = nazwa);
10 SET pierwszy_gatunek = (SELECT ilosc_pierwszy_gatunek FROM magazyn WHERE magazyn.id_magazyn = id);
11 SET drugi_gatunek = (SELECT ilosc_drugi_gatunek FROM magazyn WHERE magazyn.id_magazyn = id);
12 SET wynik=pierwszy_gatunek+drugi_gatunek;
13 RETURN wynik;
14 END$$
15 DELIMITER ;
```

Rysunek 2 Funkcja Calkowita\_ilosc\_odmiany\_w\_magazynie



**Nazwa funkcji:** Ilosc\_Pierwszego\_Gatunku

**Parametry:** Atrybut nazwa z tabeli produkt.

**Wynik:** Funkcja która jako wynik zwraca ilości pierwszego gatunku wybranego produktu.

**Zasada działania:**

Deklaracja zmiennych wynik, ilosc, wsk

Pobranie z tabeli produkt wartości pola id\_magazyn który jest przypisany do wpisanej nazwy produktu.

Pobranie z tabeli magazyn wartość pola ilosc\_pierwszy\_gatunek który jest przypisany do wartości pola id\_magazyn.

Zwrócenie wyniku.

```
Eksportuj procedurę `Ilosc_Pierwszego_Gatunku`  
  
1 DELIMITER $$  
2 CREATE DEFINER=`root`@`localhost` FUNCTION `Ilosc_Pierwszego_Gatunku`(`nazwa` VARCHAR(45)) RETURNS int(11)  
3     NO SQL  
4 BEGIN  
5 DECLARE wynik INT DEFAULT 0;  
6 DECLARE wsk INT DEFAULT 0;  
7 DECLARE ilosc INT DEFAULT 0;  
8  
9 SET wsk = (select id_magazyn from produkt where produkt.nazwa = nazwa);  
10 SET ilosc = (select ilosc_pierwszy_gatunek from magazyn where magazyn.id_magazyn = wsk);  
11 SET wynik=ilosc;  
12  
13 RETURN wynik;  
14  
15 END$$  
16 DELIMITER ;
```

**Rysunek 3 Funkcja Ilosc\_Pierwszego\_Gatunku**

**Nazwa procedury:** Rejestracja\_nowego\_uzytkownika

**Parametry:** Atrybuty z tabel klient, adres i kontakt.

**Wynik:** Procedura tworzy nowe rekordy w tabelach klient, adres i kontakt oraz powiązuje utworzone rekordy w tabelach adres i kontakt z utworzonym rekordem klient.

**Zasada działania:**

Deklaracja zmiennych IDADRES, IDKONTAKT.

Utworzenie nowego rekordu w tabeli adres i wypełnienie go danymi.

Pobranie wartości klucza głównego nowoutworzonej tabeli i przypisanie go do zmiennej IDADRES.

Utworzenie nowego rekordu w tabeli kontakt i wypełnienie go danymi.

Pobranie wartości klucza głównego nowoutworzonej tabeli i przypisanie go do zmiennej IDKONTAKT.

Utworzenie nowego rekordu w tabeli klient i wypełnienie go danymi, w polai id\_adres i id\_kontakt zostaną wprowadzone wartości IDADRES i IDKONTAKT.



Eksportuj procedurę `Rejestracja\_nowego\_uzytkownika`

```
1 DELIMITER $$
2 CREATE DEFINER=`root`@`localhost` PROCEDURE `Rejestracja_nowego_uzytkownika`(IN `miejscowosc` VARCHAR(45), IN `powiat` VARCHAR(45), IN
`wojewodztwo` VARCHAR(45), IN `kraj` VARCHAR(45), IN `kod_pocztowy` VARCHAR(10), IN `ulica` VARCHAR(45), IN `nr_domu` VARCHAR(10), IN
`nr_mieszkania` VARCHAR(10), IN `nr_telefonu_1` VARCHAR(45), IN `nr_telefonu_2` VARCHAR(45), IN `fax` VARCHAR(45), IN `email`
VARCHAR(45), IN `nazwisko` VARCHAR(45), IN `imie` VARCHAR(45), IN `login` VARCHAR(45), IN `haslo` VARCHAR(45))
3 NO SQL
4 BEGIN
5 DECLARE IDADRES INT DEFAULT 1;
6 DECLARE IDKONTAKT INT DEFAULT 1;
7 INSERT INTO firma.adres(miejscowosc,powiat,wojewodztwo,kraj,kod_pocztowy,ulica,nr_domu,nr_mieszkania)
8 VALUES (miejscowosc,powiat,wojewodztwo,kraj,kod_pocztowy,ulica,nr_domu,nr_mieszkania);
9 SET IDADRES = LAST_INSERT_ID();
10 INSERT INTO firma.kontakt(nr_telefonu_1,nr_telefonu_2,fax,email)
11 VALUES (nr_telefonu_1,nr_telefonu_2,fax,email);
12 SET IDKONTAKT = LAST_INSERT_ID();
13 INSERT INTO firma.klient(nazwisko,imie,login,haslo,id_adres,id_kontakt)
14 VALUES (nazwisko,imie,login,haslo,IDADRES,IDKONTAKT);
15 END$$
16 DELIMITER ;
```

Rysunek 4 Procedura Rejestracja\_nowego\_uzytkownika.

**Nazwa procedury:** Usun\_zdjecie

**Parametry:** Atrybut id\_zdjecie z tabeli galeria.

**Wynik:** Procedura usunie wybrany rekord z tabeli galeria.

**Zasada działania:**

Po podaniu wartości id\_zdjecie, procedura usunie wybrany rekord.

Eksportuj procedurę `Usun\_zdjecie`

```
1 DELIMITER $$
2 CREATE DEFINER=`root`@`localhost` PROCEDURE `Usun_zdjecie`(IN
`id_zdjecie` INT)
3 NO SQL
4 BEGIN
5 DELETE FROM galeria WHERE (id_zdjecie=id_zdjecie);
6 END$$
7 DELIMITER ;
```

Rysunek 5 Procedura Usun\_zdjecie



**Nazwa procedury:** Wyswietl\_zamowienie

**Parametry:** Atrybut nazwisko z tabeli klient.

**Wynik:** Procedura wyświetli zamówiony towar przez danego klienta .

**Zasada działania:** Za pomocą złączenia INNER JOIN zapytanie zwraca wartości id\_zamowienie z tablicy zamowienie\_produkty, nazwa z tablicy produkt i ilosc z tablicy zamowienie\_produkty.

```
Eksportuj procedurę `Wyswietl_zamowienie`

1 DELIMITER $$
2 CREATE DEFINER='root'@'localhost' PROCEDURE `Wyswietl_zamowienie`(IN `Nazwisko` VARCHAR(45))
3 NO SQL
4 BEGIN
5 SELECT zp.id_zamowienie AS nr_zamowienia, p.nazwa, zp.ilosc
6 FROM zamowienie_produkty AS zp
7 INNER JOIN produkt AS p ON zp.id_produkty = p.id_produkty
8 INNER JOIN zamowienie AS z ON zp.id_zamowienie = z.id_zamowienie
9 INNER JOIN klient AS k ON z.id_klient = k.id_klient
10 where k.nazwisko = Nazwisko;
11 END$$
12 DELIMITER ;
```

**Rysunek 6 Procedura Wyswietl\_zamowienie**

**Nazwa procedury:** Edytuj\_kontakt

**Parametry:** Atrybuty z kontakt.

**Wynik:** Procedura pozwala dokonać modyfikacji wartości w tabeli adres.

**Zasada działania:** Następuje sprawdzenie czy podany atrybut id\_adres występuje w tabeli adres. Jeśli nie, zostanie wyświetlony komunikat „Podanego ID nie ma w bazie”. Jeśli id\_adres występuje w tabeli adres, zostaną wykonane kolejne zapytania, które sprawdzą czy użytkownik podał odpowiednie wartości do modyfikacji pól w rekordzie. Jeśli zostały podane, nastąpi modyfikacja odpowiedniego pola w tabeli, jeśli nie zostały podane pola nie zostaną zmodyfikowane.

```
Eksportuj procedurę `Edytuj_kontakt`

1 DELIMITER $$
2 CREATE DEFINER='root'@'localhost' PROCEDURE `Edytuj_kontakt`(IN `id_kontakt` INT, IN `nr_telefonu_1` VARCHAR(45),
3 IN `nr_telefonu_2` VARCHAR(45), IN `fax` VARCHAR(45), IN `email` VARCHAR(45))
4 NO SQL
5 BEGIN
6 IF NOT EXISTS ( SELECT id_kontakt FROM kontakt WHERE kontakt.id_kontakt = id_kontakt)
7 THEN
8 SELECT 'Podanego ID nie ma w bazie';
9 ELSE
10
11 UPDATE kontakt
12 SET kontakt.nr_telefonu_1 = nr_telefonu_1
13 WHERE kontakt.id_kontakt = id_kontakt AND nr_telefonu_1 > 0;
14
15 UPDATE kontakt
16 SET kontakt.nr_telefonu_2 = nr_telefonu_2
17 WHERE kontakt.id_kontakt = id_kontakt AND nr_telefonu_2 > 0;
18
19 UPDATE kontakt
```

**Rysunek 7 Procedura Edytuj\_kontakt**



### 3. Wnioski

Zaprojektowano i wykonano bazę danych spełniającą przyjęte założenia początkowe. Skutecznie zaimplementowano część funkcji i procedur pomocnych do jej obsługi i użytkowania. Podczas wykonywania projektu wystąpiły trudności związane z brakiem doświadczenia w użytkowaniu wykorzystywanych narzędzi. Podczas pracy nad projektem nabyłem praktycznych umiejętności w tworzeniu baz danych w programie phpMyAdmin. Utworzoną bazę danych umieściłem w formie skryptu(firma.sql) w repozytorium.

### 4. Spis rysunków i tabel

Rysunek 1 Diagram ERD.....	4
Rysunek 2 Funkcja Calkowita_ilosc_odmiany_w_magazynie.....	5
Rysunek 3 Funkcja Ilosc_Pierwszego_Gatunku .....	6
Rysunek 4 Procedura rejestracja_nowego_uzytkownika.....	7
Rysunek 5 Procedura Usun_zdjecie.....	7
Rysunek 8 Procedura Wyswietl_zamowienie.....	8
Rysunek 7 Procedura Edytuj_kontakt .....	8