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 albumu161885
III EF-ZI

POLITECHNIKA RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

Katedra
Informatyki i Automatyki
Politechnika Rzeszowska

Spis treści

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

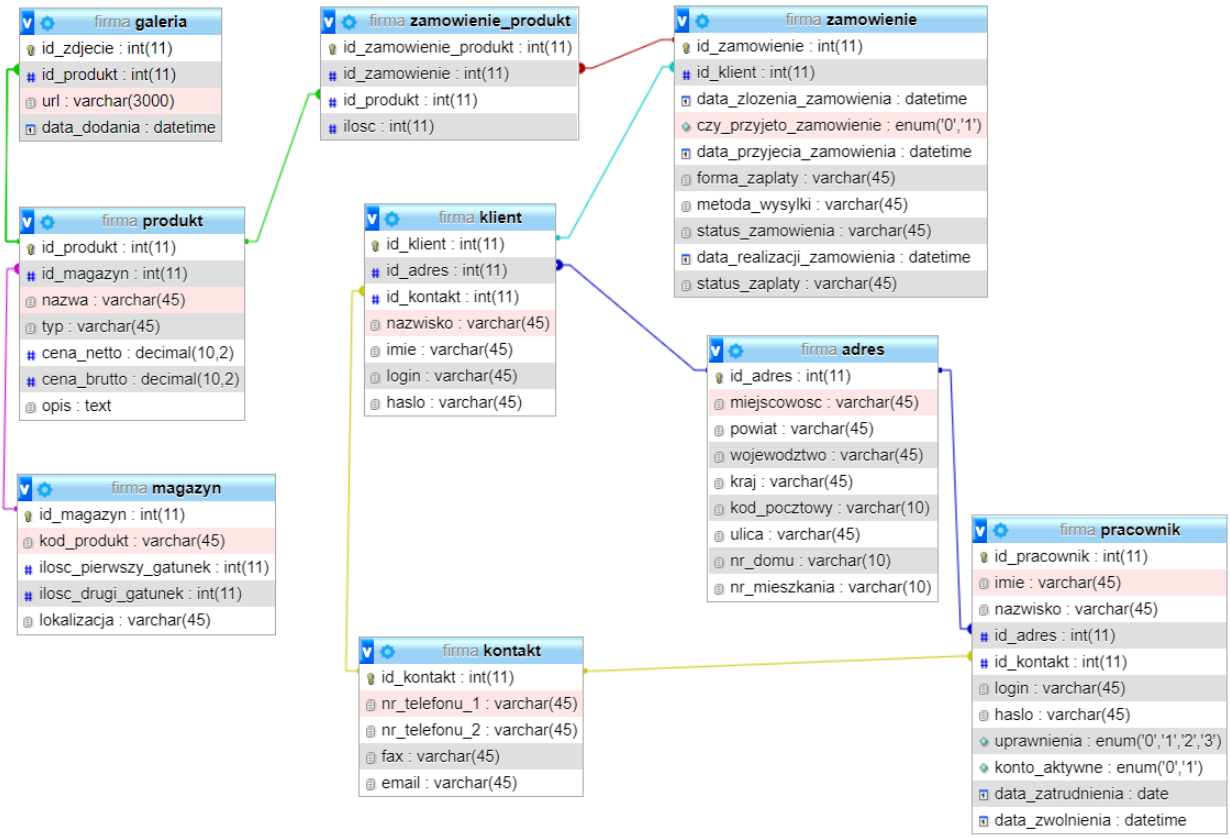
Repozytorium znajduje się na stronie internetowej github.

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_produkt 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_produkt 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.

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`

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `Calkowita_ilosc_odmiany_w_magazynie`(`nazwa` VARCHAR(45)) RETURNS int(11)
    NO SQL
BEGIN
DECLARE wynik INT DEFAULT 0;
DECLARE pierwszy_gatunek INT DEFAULT 0;
DECLARE drugi_gatunek INT DEFAULT 0;
DECLARE id INT DEFAULT 0;
SET id =(SELECT id_magazyn FROM produkt WHERE produkt.nazwa = nazwa);
SET pierwszy_gatunek = (SELECT ilosc_pierwszy_gatunek FROM magazyn WHERE magazyn.id_magazyn = id);
SET drugi_gatunek = (SELECT ilosc_drugi_gatunek FROM magazyn WHERE magazyn.id_magazyn = id);
SET wynik=pierwszy_gatunek+drugi_gatunek;
RETURN wynik;
END$$
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`

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `Ilosc_Pierwszego_Gatunku`(`nazwa` VARCHAR(45)) RETURNS int(11)
    NO SQL
BEGIN
DECLARE wynik INT DEFAULT 0;
DECLARE wsk INT DEFAULT 0;
DECLARE ilosc INT DEFAULT 0;

 SET wsk = (select id_magazyn from produkt where produkt.nazwa = nazwa);
 SET ilosc = (select ilosc_pierwszy_gatunek from magazyn where magazyn.id_magazyn = wsk);
SET wynik=ilosc;

RETURN wynik;

END$$
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`

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

Rysunek 4 Procedura Rejestracja_nowego_uzytkownika.

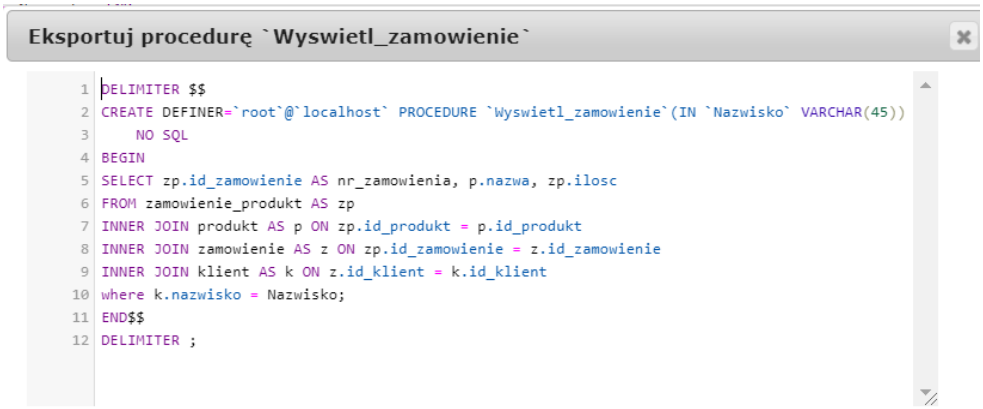
Nazwa procedury: Usun_zdjecie
Parametry: Atrybut id_zdjecie z tabeli galeria.
Wynik: Procedura usunie wybrany rekord z tabeli galeria.
Zasada działania:
Po podaniu wartości id_zdjecie, procedura usunie wybrany rekord.

Eksportuj procedurę `Usun_zdjecie`

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `Usun_zdjecie`(IN `id_zdjecie` INT)
    NO SQL
BEGIN
DELETE FROM galeria WHERE (id_zdjecie=id_zdjecie);
END$$
DELIMITER ;
```

Rysunek 5 Procedura Usun_zdjecie

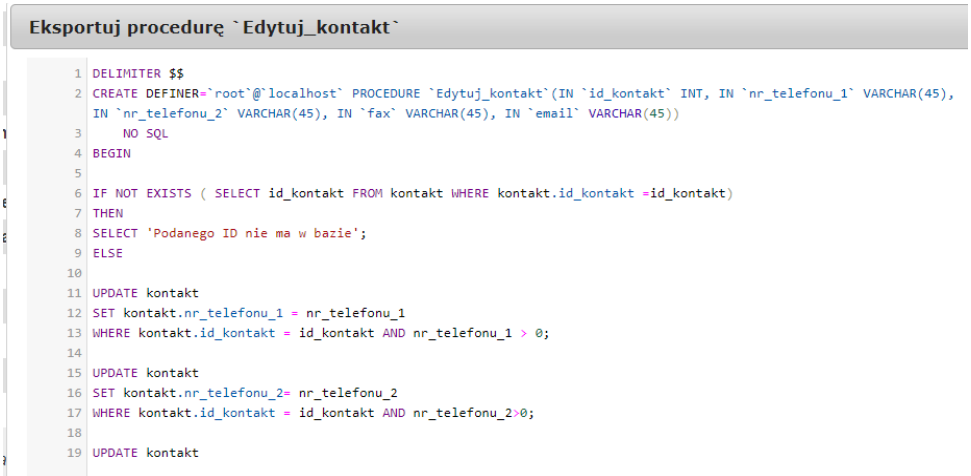
Nazwa procedury: Wyswietl_zamowienie
Parametry: Atrybut nazwisko z tabeli klient.
Wynik: Procedura wyświetli zamówiony towar przez danego klienta .
Zasada działania: Za pomocą złączenia INNER JOIN zapytanie zwraca wartości id_zamowienie z tablicy zamówienie_produkty, nazwa z tablicy produkt i ilosc z tablicy zamówienie_produkty.


Eksportuj procedurę `Wyswietl_zamowienie`

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `Wyswietl_zamowienie`(IN `Nazwisko` VARCHAR(45))
    NO SQL
BEGIN
SELECT zp.id_zamowienie AS nr_zamowienia, p.nazwa, zp.ilosc
FROM zamowienie_produkt AS zp
INNER JOIN produkt AS p ON zp.id_produkt = p.id_produkt
INNER JOIN zamowienie AS z ON zp.id_zamowienie = z.id_zamowienie
INNER JOIN klient AS k ON z.id_klient = k.id_klient
where k.nazwisko = Nazwisko;
END$$
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`

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `Edytuj_kontakt`(IN `id_kontakt` INT, IN `nr_telefonu_1` VARCHAR(45),
IN `nr_telefonu_2` VARCHAR(45), IN `fax` VARCHAR(45), IN `email` VARCHAR(45))
    NO SQL
BEGIN

IF NOT EXISTS ( SELECT id_kontakt FROM kontakt WHERE kontakt.id_kontakt =id_kontakt)
THEN
SELECT 'Podanego ID nie ma w bazie';
ELSE

UPDATE kontakt
SET kontakt.nr_telefonu_1 = nr_telefonu_1
WHERE kontakt.id_kontakt = id_kontakt AND nr_telefonu_1 > 0;

UPDATE kontakt
SET kontakt.nr_telefonu_2= nr_telefonu_2
WHERE kontakt.id_kontakt = id_kontakt AND nr_telefonu_2>0;

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