

# Bazy Danych 1 - Dokumentacja Projektu

Bartosz Konopka

# 1 Projekt koncepcji, założenia

## 1.1 Zdefiniowanie tematu projektu

Projekt ma na celu stworzenie kompleksowej bazy danych obsługującej sklep internetowy specjalizujący się w sprzedaży plakatów. Sklep oferuje szeroki wybór plakatów różnych kategorii, od artystycznych po filmowe i muzyczne. Zadaniem projektu jest efektywne zarządzanie informacjami dotyczącymi produktów, zamówień, klientów oraz dostaw.

Cele projektu:

- Efektywne Zarządzanie Produktami: Umożliwienie skutecznego przechowywania informacji o plakatach, takich jak tytuł, kategoria, cena, czy ilość egzemplarzy.
- Obsługa zamówień i dostaw: Przechowywanie informacji o zamówieniach składanych przez klientów, oraz o dostawach realizowanych przez pracowników.
- Informacje o Klientach, Pracownikach i Producentach: Przechowywanie informacji o klientach składających zamówienia, pracownikach realizujących dostawy egzemplarzy, oraz o producentach tworzących plakaty.
- Możliwość oceniania poszczególnych produktów: Zapewnienie klientom możliwości oceny produktu.

Zadania do realizacji:

- Projekt Struktury Bazy Danych: Określenie struktury tabel i relacji między nimi, uwzględniając potrzeby sklepu z plakatami.
- Implementacja Interfejsu Bazy Danych: Opracowanie interfejsu użytkownika umożliwiającego dodawanie i przeglądanie dostępnych tabel, oraz generowanie raportów.
- Dokumentacja Projektu: Stworzenie kompleksowej dokumentacji.

## 1.2 Analiza wymagań użytkownika

Projektowana baza danych spełniać powinna poniższe funkcjonalności:

- Zarządzanie produktami: Tworzenie zarówno typu produktu, jak i nowych produktów, oraz ich właściwości.
- Zarządzanie użytkownikami: Możliwość zarządzania producentami produktów, pracownikami, oraz klientami
- Zarządzanie dostawami produktów: Możliwość realizowania dostaw produktów w celu zwiększenia ilości egzemplarzy produktu.
- Zarządzanie zamówieniami: Możliwość tworzenia zamówień przypisanych do konkretnego klienta
- Zarządzanie ocenami produktów: Opcja wystawiania oceny danemu produktowi, oraz zarządzanie ocenami.

## 1.3 Zaprojektowanie funkcji

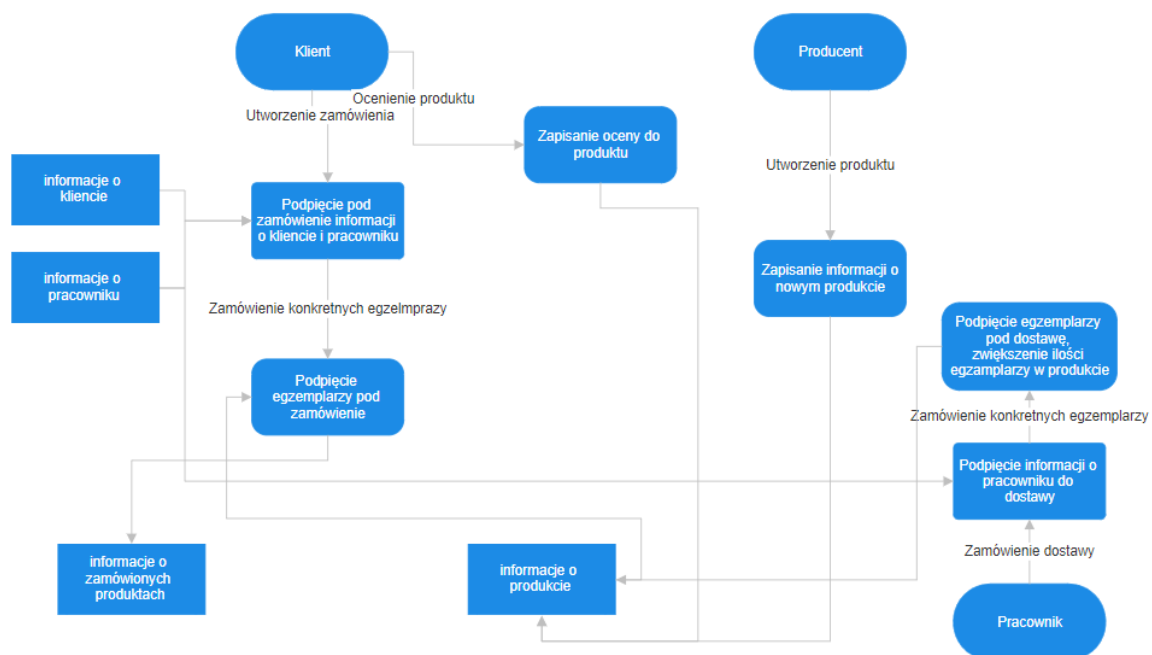
Podstawowe funkcje realizowane w bazie danych:

- Możliwość dokonania zamówienia przez klienta - Każdy klient ma możliwość wykonania zamówienia, w którym znajdować się może nieograniczona ilość produktów, posiadających swoją cenę.
- Możliwość wystawienia oceny produktu przez klienta - Każdy klient ma możliwość wystawienia oceny danemu produktowi.
- Możliwość zamówienia dostawy produktów przez pracownika - pracownicy mają możliwość zamówienia produktów, co zwiększa ilość ich egzemplarzy
- Możliwość stworzenia produktu przez Producenta - każdy producent ma możliwość stworzenia produktu o konkretnym typie.

## 2 Projekt diagramów

### 2.1 Budowa i analiza diagramu przepływu danych

Przepływ danych przedstawiony jest na poniższym diagramie:



Rysunek 1: Schemat DFD

### 2.2 Zdefiniowanie encji (obiektów) oraz ich atrybutów

Poniżej znajduje się definicja encji oraz ich atrybutów:

- Klient - Tabela opisująca encję klienta w bazie danych. Klient ma możliwość składania zamówień i oceny produktów.

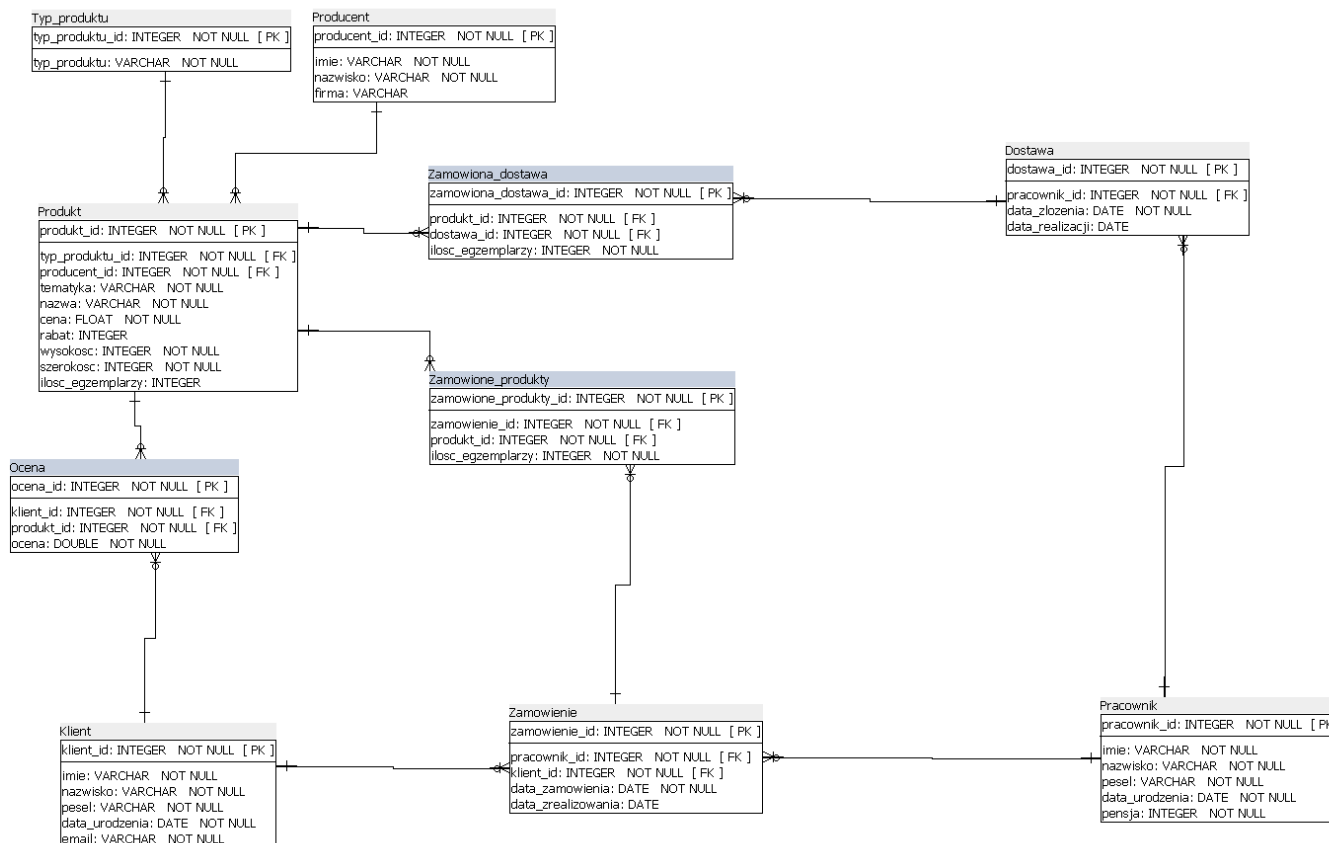
- klient id SERIAL - Unikalne id klienta.
- imie VARCHAR NOT NULL - Imię klienta.
- nazwisko VARCHAR NOT NULL - Nazwisko klienta.
- pesel VARCHAR NOT NULL - Pesel klienta.
- data urodzenia DATE NOT NULL - Data urodzenia klienta.
- email VARCHAR NOT NULL - E-mail klienta.
- Pracownik - Tabela opisująca encję pracownika w bazie danych. Pracownik ma możliwość realizacji zamówień, oraz domawiania produktów przez dostawę.
  - pracownik id SERIAL - Unikalne id pracownika.
  - imie VARCHAR NOT NULL - Imię pracownika.
  - nazwisko VARCHAR NOT NULL - Nazwisko pracownika.
  - pesel VARCHAR NOT NULL - Pesel pracownika.
  - data urodzenia DATE NOT NULL - Data urodzenia pracownika.
  - pensja INTEGER NOT NULL - Miesięczna pensja pracownika.
- Producent - Tabela opisująca encję producenta w bazie danych. Zadaniem Producenta jest tworzenie produktów.
  - producent id SERIAL - Unikalne id producenta.
  - imie VARCHAR NOT NULL - Imię producenta.
  - nazwisko VARCHAR NOT NULL - Nazwisko producenta.
  - firma VARCHAR - Firma w której producent pracuje. Firma może przyjąć wartość NULL, w takim wypadku producent nie podlega żadnej firmie.
- Typ Produktu - Encja typu produktu która definiuje jakiego typu jest produkt. Podstawowe zdefiniowane w baie danych typy to: plakat, obraz i antyrama. Użytkownik ma możliwość stworzenia nowego typu produktu zgodnego z atrybutami produktu.
  - typ produktu id SERIAL - Unikalne id typu produktu.
  - typ produktu VARCHAR DEFAULT 'plakat' NOT NULL - Nazwa typu produktu.
- Produkt - Encja odpowiedzialna za produkt i jego właściwości.
  - produkt id SERIAL - Unikalne id produktu.
  - typ produktu id INTEGER NOT NULL - Klucz obcy definiujący jakiego typu jest produkt.
  - producent id INTEGER NOT NULL - Klucz obcy definujący który producent wykonał produkt.
  - tematyka VARCHAR NOT NULL - Tematyka w jakiej jest produkt.
  - nazwa VARCHAR NOT NULL - Nazwa produktu.
  - cena INTEGER NOT NULL - Cena produktu.

- rabat INTEGER DEFAULT 0 - Rabat jaki przysługuje danemu produktowi.
- wysokość INTEGER NOT NULL - Wysokość produktu.
- szerokość INTEGER NOT NULL - Szerokość produktu.
- ilość egzemplarzy INTEGER DEFAULT 100 - Ilość aktualnie dostępnych produktów w sklepie.
- ocena FLOAT DEFAULT 0 - Aktualna średnia ocena produktu.
- ilość ocen INTEGER DEFAULT 0 - Ilość wystawionych ocen produktowi.
- Ocena - Encja odpowiedzialna za wystawienie oceny produktowi przez klienta.
  - ocena id SERIAL - Unikalne id wystawianej oceny.
  - klient id INTEGER NOT NULL - Klient wystawiający ocenę.
  - produkt id INTEGER NOT NULL - Produkt dla którego wystawiana jest ocena.
  - ocena FLOAT NOT NULL - Ocena która zostanie wystawiona przez klienta produktowi.
- Zamówienie - Encja odpowiedzialna za zamówienie, każdy klient może złożyć zamówienie,
  - zamówienie id SERIAL - Uniklane id zamówienia.
  - pracownik id INTEGER NOT NULL DEFAULT 1 - Pracownik realizujący dane zamówienie.
  - klient id INTEGER NOT NULL - Klient który dokonuje danego zamówienia.
  - data zamówienia DATE NOT NULL - Data w której zamówienie zostało dokonane.
  - data zrealizowania DATE DEFAULT NULL - Data w której zamówienie zostało zrealizowane, wartość NULL oznacza że zamówienie nie zostało jeszcze zrealizowane.
- Zamówione Produkty - Tabela pośrednicząca między produktem a zamówieniem, realizująca połączenie tabel N:M. Dzięki temu klient w jednym zamówieniu może zamówić wiele produktów.
  - zamówione produkty id SERIAL - Unikalne id tabeli pośredniczącej.
  - zamówienie id INTEGER NOT NULL - Zamówienie do którego odnosi się tabela.
  - produkt id INTEGER NOT NULL - Produkt który zamawiany jest w zamówieniu.
  - ilość egzemplarzy INTEGER NOT NULL - ilość egzemplarzy zamawianego produktu.
- Dostawa - Tabela odpowiedzialna za encję dostawa. Pracownicy mogą dokonywać dostaw produktów, zwiększając ilość egzemplarzy produktu.
  - dostawa id SERIAL - Unikalne id dostawy.

- pracownik id INTEGER NOT NULL - Pracownik realizujący dostawę.
- data złożenia DATE NOT NULL - Data złożenia dostawy.
- data realizacji DATE DEFAULT NULL Data zrealizowania dostawy, jeżeli wartość wynosi NULL, dostawa nie została jeszcze zrealizowana.
- Zamówiona Dostawa - Tabela pomocnicza realizująca połączenie N:M między tabelami dostawa, a Produkt. Dzięki niej pracownik w każdej dostawie może zamówić wiele produktów.
  - zamówiona dostawa id SERIAL - unikalne id tabeli pomocniczej.
  - produkt id INTEGER NOT NULL - Domawiany produkt.
  - dostawa id INTEGER NOT NULL - Id dostawy w której produkt zostanie domówiony.
  - ilość egzemplarzy INTEGER NOT NULL - ilość egzemplarzy zamówionego produktu.

## 2.3 Zaprojektowanie relacji pomiędzy encjami

Poniżej znajduje się schemat ERD opisujący relacje między encjami:



Rysunek 2: Schemat ERD

## 3 Projekt logiczny

### 3.1 Projektowanie tabel, kluczy, indeksów

Struktura bazy danych i projekt tabel przedstawiony został dokładnie w podpunkcie 2.2. Wszystkie tabele, encje, oraz odnoszenie się do siebie kluczy obcych zostało również opisane w tym samym podpunkcie. Poniżej przedstawiony jest skrypt SQL inicjujący tabele oraz klucze obce.

```
CREATE SCHEMA IF NOT EXISTS sklep;
```

```
CREATE TABLE sklep.Klient (  
    klient_id SERIAL,  
    imie VARCHAR NOT NULL,  
    nazwisko VARCHAR NOT NULL,  
    pesel VARCHAR NOT NULL,  
    data_urodzenia DATE NOT NULL,  
    email VARCHAR NOT NULL,  
    CONSTRAINT klient_pk PRIMARY KEY (klient_id)  
);
```

```
CREATE TABLE sklep.Pracownik (  
    pracownik_id SERIAL,  
    imie VARCHAR NOT NULL,  
    nazwisko VARCHAR NOT NULL,  
    pesel VARCHAR NOT NULL,  
    data_urodzenia DATE NOT NULL,  
    pensja INTEGER NOT NULL,  
    CONSTRAINT pracownik_pk PRIMARY KEY (pracownik_id)  
);
```

```
CREATE TABLE sklep.Dostawa (  
    dostawa_id SERIAL,  
    pracownik_id INTEGER NOT NULL,  
    data_zlozenia DATE NOT NULL,  
    data_realizacji DATE DEFAULT NULL,  
    CONSTRAINT dostawa_pk PRIMARY KEY (dostawa_id)  
);
```

```
CREATE TABLE sklep.Zamowienie (  
    zamowienie_id SERIAL,  
    pracownik_id INTEGER NOT NULL DEFAULT 1,  
    klient_id INTEGER NOT NULL,  
    data_zamowienia DATE NOT NULL,  
    data_zrealizowania DATE DEFAULT NULL,  
    CONSTRAINT zamowienie_pk PRIMARY KEY (zamowienie_id)  
);
```

```
CREATE TABLE sklep.Producent (  

```

```

    producent_id SERIAL,
    imie VARCHAR NOT NULL,
    nazwisko VARCHAR NOT NULL,
    firma VARCHAR,
    CONSTRAINT producent_pk PRIMARY KEY (producent_id)
);

CREATE TABLE sklep.Typ_produktu (
    typ_produktu_id SERIAL,
    typ_produktu VARCHAR DEFAULT 'plakat' NOT NULL,
    CONSTRAINT typ_produktu_pk PRIMARY KEY (typ_produktu_id)
);

CREATE TABLE sklep.Produkt (
    produkt_id SERIAL,
    typ_produktu_id INTEGER NOT NULL,
    producent_id INTEGER NOT NULL,
    tematyka VARCHAR NOT NULL,
    nazwa VARCHAR NOT NULL,
    cena INTEGER NOT NULL,
    rabat INTEGER DEFAULT 0,
    wysokosc INTEGER NOT NULL,
    szerokosc INTEGER NOT NULL,
    ilosc_egzemplarzy INTEGER DEFAULT 100,
    ocena FLOAT DEFAULT 0,
    ilosc_ocen INTEGER DEFAULT 0,
    CONSTRAINT produkt_pk PRIMARY KEY (produkt_id)
);

CREATE TABLE sklep.Zamowiona_dostawa (
    zamowiona_dostawa_id SERIAL,
    produkt_id INTEGER NOT NULL,
    dostawa_id INTEGER NOT NULL,
    ilosc_egzemplarzy INTEGER NOT NULL,
    CONSTRAINT zamowiona_dostawa_pk PRIMARY KEY (zamowiona_dostawa_id)
);

CREATE TABLE sklep.Ocena (
    ocena_id SERIAL,
    klient_id INTEGER NOT NULL,
    produkt_id INTEGER NOT NULL,
    ocena FLOAT NOT NULL,
    CONSTRAINT ocena_pk PRIMARY KEY (ocena_id)
);

CREATE TABLE sklep.Zamowione_produkty (
    zamowione_produkty_id SERIAL,
    zamowienie_id INTEGER NOT NULL,

```



```
    produkt_id INTEGER NOT NULL,  
    ilosc_egzemplarzy INTEGER NOT NULL,  
    CONSTRAINT Zamowione_produkty_pk PRIMARY KEY (zamowione_produkty_id)  
);
```

```
ALTER TABLE sklep.Zamowienie ADD CONSTRAINT Klient_Zamowienie_fk  
    FOREIGN KEY (klient_id)  
        REFERENCES sklep.Klient (klient_id)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION  
        NOT DEFERRABLE;
```

```
ALTER TABLE sklep.Ocena ADD CONSTRAINT Klient_Ocena_fk  
    FOREIGN KEY (klient_id)  
        REFERENCES sklep.Klient (klient_id)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION  
        NOT DEFERRABLE;
```

```
ALTER TABLE sklep.Zamowienie ADD CONSTRAINT Pracownik_Zamowienie_fk  
    FOREIGN KEY (pracownik_id)  
        REFERENCES sklep.Pracownik (pracownik_id)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION  
        NOT DEFERRABLE;
```

```
ALTER TABLE sklep.Dostawa ADD CONSTRAINT Pracownik_Dostawa_fk  
    FOREIGN KEY (pracownik_id)  
        REFERENCES sklep.Pracownik (pracownik_id)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION  
        NOT DEFERRABLE;
```

```
ALTER TABLE sklep.Zamowiona_dostawa ADD CONSTRAINT Dostawa_Zamowiona_dostawa_fk  
    FOREIGN KEY (dostawa_id)  
        REFERENCES sklep.Dostawa (dostawa_id)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION  
        NOT DEFERRABLE;
```

```
ALTER TABLE sklep.Zamowione_produkty ADD CONSTRAINT Zamowienie_Zamowione_produkty_fk  
    FOREIGN KEY (zamowienie_id)  
        REFERENCES sklep.Zamowienie (zamowienie_id)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION  
        NOT DEFERRABLE;
```

```
ALTER TABLE sklep.Produkt ADD CONSTRAINT Producent_Produkt_fk
```

```

FOREIGN KEY (producent_id)
REFERENCES sklep.Producent (producent_id)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE sklep.Produkt ADD CONSTRAINT Typ_produkту_Produkt_fk
FOREIGN KEY (typ_produkту_id)
REFERENCES sklep.Typ_produkту (typ_produkту_id)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE sklep.Zamowione_produkty ADD CONSTRAINT produkt_Zamowione_produkty_fk
FOREIGN KEY (produkt_id)
REFERENCES sklep.Produkt (produkt_id)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE sklep.Ocena ADD CONSTRAINT produkt_Ocena_fk
FOREIGN KEY (produkt_id)
REFERENCES sklep.Produkt (produkt_id)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE sklep.Zamowiona_dostawa ADD CONSTRAINT produkt_Zamowiona_dostawa_fk
FOREIGN KEY (produkt_id)
REFERENCES sklep.Produkt (produkt_id)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

```

## 3.2 Słowniki danych

Słowniki danych i opis każdego z atrybutów tabeli został zawarty w podpunkcie 2.2.

## 3.3 Zaprojektowanie operacji na danych

Operacje na bazie danych obejmują:

- Złożenie zamówienia przez klienta: Posiadając odpowiednio zdefiniowanego klienta, oraz pracownika możemy złożyć zamówienie. Najpierw definiując nowe zamówienie, podając w nim id klienta który składa zamówienie, oraz id pracownika który dane zamówienie będzie realizował, należy również używając insertu podać w tabeli datę zamówienia. Następnie posiadając unikalne id zamówienia, mamy możliwość dodania Zamówionych produktów poprzez tablicę o tej samej nazwie. Podając w tej tabeli id zamówienia, id produktu, oraz ilość zamawianych egzemplarzy, możemy

zamówić wpisaną ilość egzemplarzy danego produktu. Odpowiedni trygger zmniejsza ilość egzemplarzy produktu po każdym złożeniu Zamówionych produktów:

```
CREATE OR REPLACE FUNCTION sklep.add_zamowienie_function() RETURNS
TRIGGER AS
$$
BEGIN
    UPDATE sklep.Produkt SET ilosc_egzemplarzy =
        ilosc_egzemplarzy - NEW.ilosc_egzemplarzy WHERE produkt_id =
        NEW.produkt_id;
    RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';
```

```
CREATE TRIGGER add_zamowienie_trigger BEFORE INSERT OR UPDATE
ON sklep.Zamowione_produkty
FOR EACH ROW EXECUTE PROCEDURE sklep.add_zamowienie_function();
```

- Wystawienie oceny produktu przez klienta: Klient posiada możliwość wystawienia oceny danemu produktowi. Posiadając istniejącego Klienta oraz Produkt, wykonując przykładowy insert na tabeli ocena, wpisując do niej id klienta który wystawia ocenę, id produktu któremu ocenę wystawiamy oraz ocenę którą chcemy wystawić, za pomocą tryggera zmieni się ilość wystawionych ocen w produkcie, któremu ocenę wystawiamy oraz zmieni się średnia jego ocen:

```
CREATE OR REPLACE FUNCTION sklep.add_ocena_function() RETURNS
TRIGGER AS
$$
BEGIN
    UPDATE sklep.Produkt SET ilosc_ocen = ilosc_ocen + 1 WHERE produkt_id
    = NEW.produkt_id;
    UPDATE sklep.Produkt SET ocena =
        (((SELECT ocena FROM sklep.Produkt WHERE produkt_id = NEW.produkt_id)
        + NEW.ocena) /
        (SELECT ilosc_ocen FROM sklep.Produkt WHERE produkt_id = NEW.produkt_id)
        ) WHERE produkt_id = NEW.produkt_id;
    RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';
```

```
CREATE TRIGGER add_ocena_trigger BEFORE INSERT OR UPDATE ON sklep.Ocena
FOR EACH ROW EXECUTE PROCEDURE sklep.add_ocena_function();
```

- Możliwość zrealizowania dostawy przed pracownika: Posiadając pracownika, mamy możliwość zrealizowania dostawy produktu. Dodając do tabeli Dostawa, odpowiednio id pracownika który dostawę realizuje, a następnie definiując datę złożenia dostawy, mamy możliwość dodawania obiektów do tabeli Zamówiona Dostawa. Dodając

do tej tabeli id dostawy w której będziemy domawiali produktu, a następnie id produktu który domawiamy oraz ilość jego egzemplarzy, mamy możliwość zwiększenia ilości egzemplarzy danego produktu poprzez trygger:

```
CREATE OR REPLACE FUNCTION sklep.add_dostawa_function() RETURNS
TRIGGER AS
$$
BEGIN
    UPDATE sklep.Produkt SET ilosc_egzemplarzy = ilosc_egzemplarzy +
    NEW.ilosc_egzemplarzy WHERE produkt_id = NEW.produkt_id;
    RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';
```

```
CREATE TRIGGER add_dostawa_trigger BEFORE INSERT OR UPDATE ON
sklep.Zamowiona_dostawa
    FOR EACH ROW EXECUTE PROCEDURE sklep.add_dostawa_function();
```

- Stworzenie produktu przez producenta: Posiadając odpowiednio zdefiniowany obiekt Producenta, w którego możemy się wcielić, oraz zdefiniowany typ produktu, mamy możliwość używając id producenta oraz id typu produktu stworzyć produkt. Podając odpowiednie id w trakcie tworzenia obiektu w tabeli Produkt, mamy możliwość utworzenia Produktu, podając interesujące nas współczynniki do tabeli Produkt.

## 4 Projekt funkcjonalny

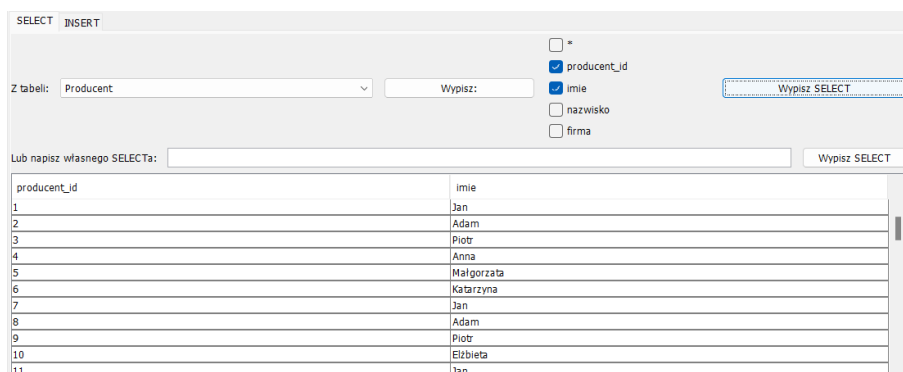
### 4.1 Interfejsy do prezentacji, edycji i obsługi danych

W aplikacji znajdują się dwa formularze, jeden odpowiedzialny za prezentację danych znajdujących się w tabelach oraz drugi odpowiedzialny za wstawianie danych do tabel.

Pierwszy z formularzy jest formularz SELECT odpowiedzialny za prezentację danych znajdujących się w tabelach.

Rysunek 3: Wygląd formularzu SELECT

Formularz gwarantuje nam przegląd wszystkich danych. W pierwszej kolejności należy wybrać tabelę którą chcemy przeglądać. Następnie po wciśnięciu przycisku *Wypisz* : pojawią się możliwe do wpisania kolumny.



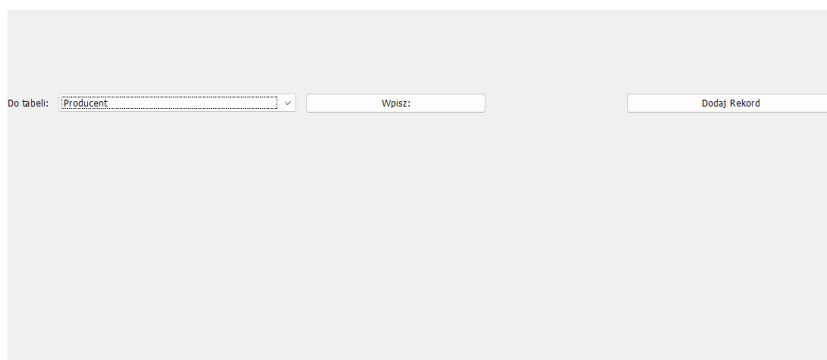
The screenshot shows a web form titled 'SELECT' and 'INSERT'. It has a dropdown menu for 'Z tabeli:' with 'Producent' selected. To the right is a 'Wypisz:' button. Further right are checkboxes for column selection: '\*' (unchecked), 'producent\_id' (checked), 'imie' (checked), 'nazwisko' (unchecked), and 'firma' (unchecked). A 'Wypisz SELECT' button is also present. Below these is a text input field for 'Lub napisz własnego SELECTA:' and another 'Wypisz SELECT' button. At the bottom, a table displays the results of the query.

producent_id	imie
1	Jan
2	Adam
3	Piotr
4	Anna
5	Małgorzata
6	Katarzyna
7	Jan
8	Adam
9	Piotr
10	Elżbieta
11	Jan

Rysunek 4: Wygląd formularzu SELECT z wyborem kolumn

Aplikacja umożliwia nam wybór kolumn które po wciśnięciu przycisku *WypiszSELECT* zostaną wyświetlone w tabeli poniżej. W tym formularzu znajduje się również możliwość wpisania własnego SELECTA, który jeżeli będzie prawidłowy wykona się, a jego wynik przedstawiony zostanie w tabeli.

Drugi formularz odpowiada za INSERT i wprowadzanie danych do tabel.



The screenshot shows a web form for the 'INSERT' operation. It features a dropdown menu labeled 'Do tabeli:' with 'Producent' selected. To the right is a 'Wpisz:' text input field. Further right is a 'Dodaj Rekord' button. The rest of the form area is empty.

Rysunek 5: Wygląd formularzu INSERT

W formularzu mamy możliwość wyboru, do której tabeli chcemy wstawić dane. Następnie po wciśnięciu przycisku *Wpisz* : pojawią się okna pozwalające na wpisanie danych.

SELECT

INSERT

Do tabeli:

Klient

▼

Wpisz:

Dodaj Rekord

imie

VARCHAR NOT NULL

Maciej

nazwisko

VARCHAR NOT NULL

Babacki

pesel

VARCHAR NOT NULL

111111111111

data\_urodzenia

DATE NOT NULL

1990-01-20

email

VARCHAR NOT NULL

babacki@gmail.com

INSERT wykonany pomyślnie

Rysunek 6: Wygląd formularzu INSERT z oknami do wpisania danych

Wszystkie wpisywane dane do tabel są walidowane, w przypadku nieprawidłowości w wpisywanych danych, po wciśnięciu przycisku *Dodaj Rekord*, zauważymy okienko z błędem. Gdy wpisane wartości będą zgadzały się z przewidywanymi wartościami, zauważymy komunikat: Insert wykonano pomyślnie, lub Błąd w wykonaniu polecenia Insert. Formularz ten umożliwia nam wpisywanie danych do każdej z tabel.

## 4.2 Wizualizacja danych

Aplikacja daje możliwość wygenerowania trzech raportów. Dane z raportów reprezentowane są w postaci tabeli znajdującej się w aplikacji. Istnieje również możliwość zapisania raportu do pliku csv.

Raport pierwszy pozwala wygenerować informację o produkcie którego ilość ocen jest większa od zera, a średnia ocena jest większa niż wybrana przez użytkownika wartość. Użytkownik może wpisać wartość średniej oceny powyżej której zostaną wypisane plakaty, a następnie po wciśnięciu przycisku *Wygeneruj raport*, wyświetlony zostanie raport w tabeli pod przyciskiem. Poniżej tabeli znajduje się przycisk pozwalający na wygenerowanie raportu do pliku.

[illegible]

Rysunek 7: Wygląd raportu pierwszego

Raport ten korzysta z poniższego widoku:

```
CREATE VIEW sklep.raportone AS
SELECT produkt_id, tematyka, nazwa, cena, wysokosc, szerokosc, ilosc_egzemplarzy,
ocena, ilosc_ocen FROM sklep.Produkt
WHERE ilosc_ocen > 0
ORDER BY produkt_id;
```

Raport drugi daje możliwość wygenerowania wszystkich złożonych zamówień, oraz koszt każdego zamówienia. Po wciśnięciu przycisku do wygenerowania raportu, zauważymy go w tabeli pod przyciskiem. Pod tabelą znajduje się przycisk pozwalający wygenerować raport do pliku.

zamowienie_id	klient_id	data_zamowienia	data_realizowania	koszt_zamowienie
1	1	2019-01-01	2019-01-02	320
2	2	2019-01-02	2019-01-03	365
3	3	2019-01-03	2019-01-04	310
4	4	2019-01-04	2019-01-05	130
5	5	2019-01-05	2019-01-06	50
6	6	2019-01-06	2019-01-07	300
7	7	2019-01-07	2019-01-08	60
8	8	2019-01-08	2019-01-09	215
9	9	2019-01-09	2019-01-10	190
10	10	2019-01-10	2019-01-11	460
11	1	2019-01-11	2019-01-12	40
12	2	2019-01-12		90
13	3	2019-01-13		40
14	4	2019-01-14		40
15	5	2019-01-15		160

Rysunek 8: Wygląd raportu drugiego

Raport ten korzysta z poniższego widoku:

```
CREATE VIEW sklep.raporttwo AS
SELECT z.zamowienie_id, z.klient_id, z.data_zamowienia, z.data_zrealizowania,
SUM(zp.ilosc_egzemplarzy * (p.cena - p.rabat)) AS koszt_zamowienie
FROM sklep.Zamowienie z
JOIN sklep.Zamowione_produkty zp USING(zamowienie_id)
JOIN sklep.Produkt p USING(produkt_id)
GROUP BY z.zamowienie_id
ORDER BY z.zamowienie_id;
```

Raport trzeci ma za zadanie wygenerować informację o klientach którzy złożyli więcej niż jedno zamówienie i przedstawić średni koszt ich zamówień. Wyświetlone zostaną tylko wartości tabeli które mają średni koszt zamówień większy od liczby zdefiniowanej przez użytkownika. Działanie raportu jest identyczne jak w przypadku raportu pierwszego i drugiego.

[illegible]

Raport ten korzysta z poniższego widoku:

### 4.3 Zdefiniowanie panelu sterowania aplikacji

MainGUIForm

Inicjalizuj bazę danych      Usuń bazę danych

SELECT | INSERT   Raporty

SELECT   INSERT

Z tabeli: Producent    Wypisz:

☒ \*  
☐ producent\_id  
☐ imie  
☐ nazwisko  
☐ firma

Wypisz SELECT

Lub napisz własnego SELECTa:    Wypisz SELECT

producent_id	imie	nazwisko	firma
1	Jan	Kowalski	Kowal
2	Adam	Nowak	Nowak
3	Piotr	Marciniak	Marciniak
4	Anna	Zwierciadło	Lustro
5	Małgorzata	Kleks	Plama
6	Katarzyna	Kowalska	Kowal
7	Jan	Nowak	Nowak
8	Adam	Złocieniec	Sprzedstwo
9	Piotr	Krawczyk	Drzewa
10	Ełzbieta	Wacąg	Plama
11	Jan	Pawłowski	Kremówka
12	23434234234	Pawłowski	Kremówka



Na samym górze okna aplikacji znajdują się przyciski odpowiedzialne za zainicjalizowanie bazy danych, oraz za usunięcie bazy danych. Za pomocą tych przycisków mamy możliwość wygenerowania wszystkich tabeli, widoków, triggerów, oraz wpisanie wartości do tabel, jak i również usunięcie wszystkich tych informacji.

Poniżej znajduje się możliwość wyboru czy chcemy aktualnie korzystać z Formularzy czy Raportów. W zakładce z formularzami nazwanej *SELECT i INSERT*, możemy wybrać czy chcemy korzystać z formularza odpowiedzialnego za SELECT, czy za INSERT. Opis sterowania aplikacji w formularzach zdefiniowany jest w podpunkcie 4.1 dokumentacji.

Mamy również możliwość wybrania zakładki *Raporty*, gdzie znajdziemy zakładki z odpowiednimi raportami. Sposób sterowania raportami znajduje się w podpunkcie 4.2 dokumentacji.

## 5 Dokumentacja

### 5.1 Wprowadzanie danych

Pierwsze dane wykorzystywane w aplikacji i pozwalające jej dobrze funkcjonować, generowane są za pomocą przycisku *Inicjuj bazę danych*. Do wszystkich tabel wprowadzane są przykładowe wartości pozwalające na korzystanie z aplikacji. Przycisk ten zapisuje wszystkie dane na serwerze baz danych, więc nie musi być włączany przy każdym odpaleniu programu. W celu usunięcia bazy danych należy wcisnąć przycisk *Usun bazę danych*. W celu wprowadzenia własnych danych do tabel należy skorzystać z formularza SELECT.

### 5.2 Dokumentacja użytkownika

Na samym początku użytkownik powinien sprawdzić czy baza danych jest zainicjalizowana wciskając przyciski do inicjalizacji, lub usunięcia bazy danych. Otrzymać można dwa komunikaty - o pozytywnym utworzeniu lub usunięciu bazy danych, lub o tym że baza danych jest już utworzona, lub usunięta. Po upewnieniu się że baza danych istnieje i została utworzona, użytkownik ma możliwość skorzystania z raportów lub formularzy. Za pomocą formularzy może przeglądać lub wprowadzać dane do tabel, a za pomocą raportów generować odpowiednie informacje. Sposób używania za równo formularzy, jak i raportów opisany został w punkcie czwartym dokumentacji.

### 5.3 Opracowanie dokumentacji technicznej

Wygenerowana w postaci html dokumentacja techniczna aplikacji znajduje się folderze doc/doc code, w katalogu z programem.