# Tytuł projektu: System Serwisu Samochodowego

Projekt przedstawia prostą aplikację webową stworzoną w językach **PHP** oraz **HTML**, której backend opiera się na **bazie danych PostgreSQL** zarządzanej za pomocą narzędzia **pgAdmin**. System ten ma na celu wsparcie prowadzenia działalności gospodarczej w zakresie **serwisu samochodowego** – od przyjmowania klientów i ich pojazdów, przez wykonywanie usług serwisowych, aż po obsługę płatności i generowanie raportów finansowych.

Zastosowana baza danych zawiera tabele logicznie powiązane ze sobą oraz zestaw **triggerów**, które automatyzują operacje aktualizacji raportów i statusów płatności. Dzięki temu system zapewnia integralność danych i minimalizuje potrzebę ręcznego nadzorowania spójności finansowej.

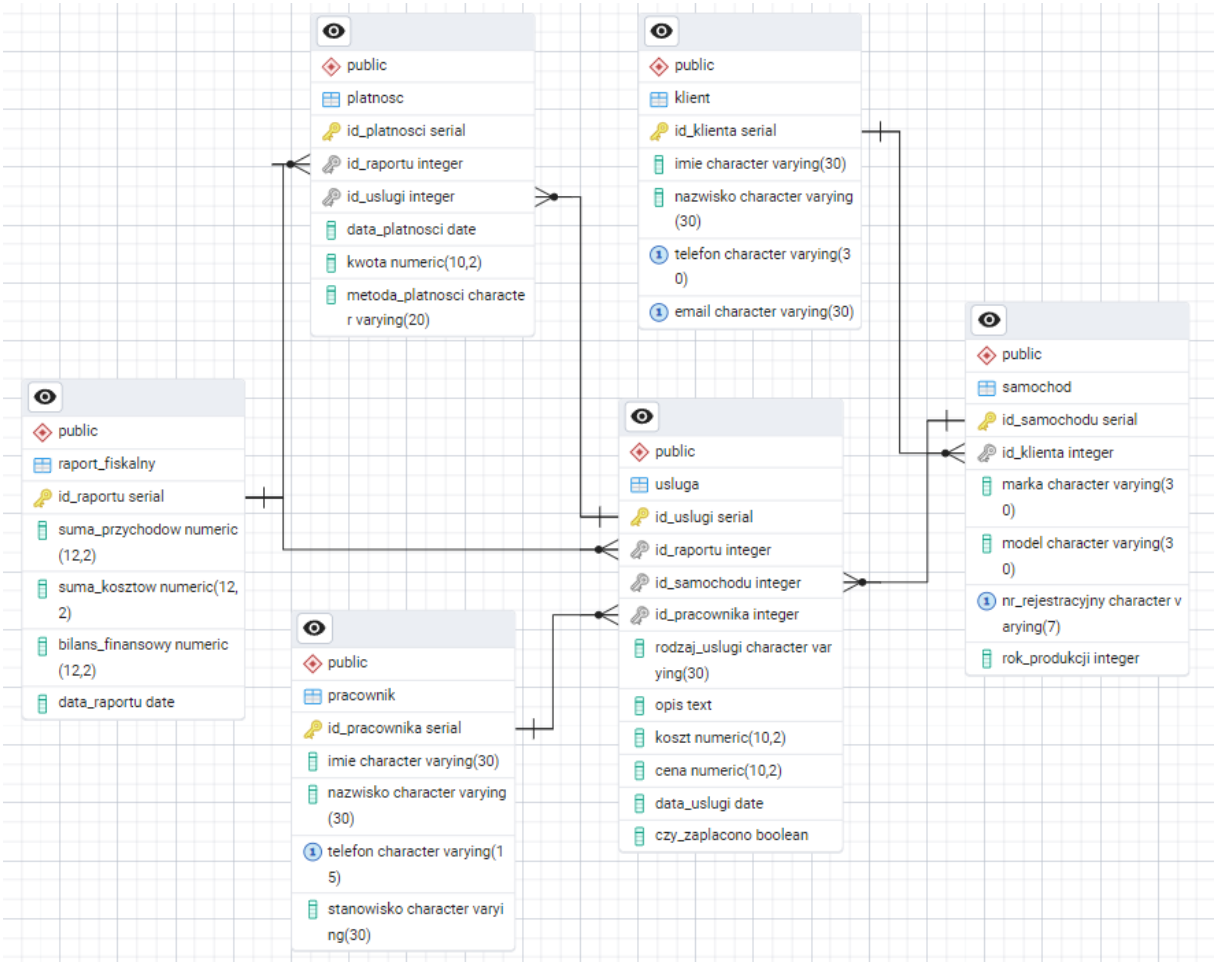
**Proces działania systemu krok po kroku**

1. **Dodanie klienta i samochodu**
    - Użytkownik wprowadza dane osobowe klienta oraz informacje o jego pojeździe (marka, model, numer rejestracyjny, rok produkcji).
    - Te dane trafiają do tabel klient i samochod, połączonych przez id_klienta.
2. **Dodanie pracownika (np. mechanika)**
    - Do systemu dodawani są pracownicy z informacją o ich stanowisku (np. mechanik, kierownik, recepcjonistka).
3. **Zlecenie usługi**
    - Pracownik przypisuje usługę do konkretnego samochodu i raportu finansowego.
    - Usługa ma określony **rodzaj**, **opis**, **koszt** i **cenę** oraz datę wykonania.
    - Wartość czy_zaplacono domyślnie ustawiona jest na FALSE.
4. **Zarejestrowanie płatności**
    - Płatność przypisywana jest do usługi. Możliwa jest zapłata **gotówką, przelewem** lub **kartą**.
    - Jeżeli kwota płatności równa się cenie usługi, trigger automatycznie ustawia czy_zaplacono = TRUE.
5. **Automatyczne aktualizacje danych finansowych**

- Dzięki triggerom, każda zmiana w płatnościach lub usługach powoduje **automatyczne zaktualizowanie raportu fiskalnego**, który zawiera:
  - sumę kosztów
  - sumę przychodów
  - bilans finansowy (obliczany automatycznie)

## Schemat encji:

Schemat relacji:



Kod SQL

```
-- tabela klient
CREATE TABLE klient (
  id_klienta SERIAL PRIMARY KEY,
  imie VARCHAR(30) NOT NULL,
  nazwisko VARCHAR(30) NOT NULL,
  telefon VARCHAR(30) NOT NULL UNIQUE,
  email VARCHAR(30) NOT NULL UNIQUE
);
-- tabela samochod
CREATE TABLE samochod (
  id_samochodu SERIAL PRIMARY KEY,
  id_klienta INT NOT NULL REFERENCES klient(id_klienta) ON DELETE CASCADE,
  marka VARCHAR(30) NOT NULL,
  model VARCHAR(30) NOT NULL,
  nr_rejestracyjny VARCHAR(7) UNIQUE NOT NULL,
  rok_produkcji INT NOT NULL CHECK(rok_produkcji BETWEEN 1900 AND 2100)
);
-- tabela pracownik
CREATE TABLE pracownik (
  id_pracownika SERIAL PRIMARY KEY,
  imie VARCHAR(30) NOT NULL,
  nazwisko VARCHAR(30) NOT NULL,
  telefon VARCHAR(15) NOT NULL UNIQUE,
  stanowisko VARCHAR(30)
);
-- tabela raport_fiskalny
CREATE TABLE raport_fiskalny (
```

```
  id_raportu SERIAL PRIMARY KEY,
  suma_przychodow DECIMAL(12,2) DEFAULT 0.00,
  suma_kosztow DECIMAL(12,2) DEFAULT 0.00,
  bilans_finansowy DECIMAL(12,2) GENERATED ALWAYS AS(suma_przychodow  - suma_kosztow) STORED,
  data_raportu DATE NOT NULL
);
-- tabela usluga
CREATE TABLE usluga (
  id_uslugi SERIAL PRIMARY KEY,
  id_raportu INT REFERENCES raport_fiskalny(id_raportu) ON DELETE SET NULL,
  id_samochodu INT NOT NULL REFERENCES samochod(id_samochodu) ON DELETE CASCADE,
  id_pracownika INT REFERENCES pracownik(id_pracownika) ON DELETE SET NULL,
  rodzaj_uslugi VARCHAR(30) NOT NULL,
  opis TEXT NOT NULL,
  koszt DECIMAL(10,2) NOT NULL CHECK (koszt >=0),
  cena DECIMAL(10,2) NOT NULL CHECK(cena >=0),
  data_uslugi DATE NOT NULL
czy_zaplacono BOOLEAN DEFAULT FALSE
);
-- tabela platnosc
CREATE TABLE platnosc (
  id_platnosci SERIAL PRIMARY KEY,
  id_raportu INT REFERENCES raport_fiskalny(id_raportu) ON DELETE SET NULL,
  id_uslugi INT NOT NULL REFERENCES usluga(id_uslugi) ON DELETE CASCADE,
  data_platnosci DATE NOT NULL,
  kwota DECIMAL(10,2) NOT NULL CHECK (kwota >= 0),
```

```
   metoda_platnosci VARCHAR(20) NOT NULL CHECK (metoda_platnosci IN ('gotowka',
'karta')),
);
--TRIGGERY
--trigger aktualizyje przychody w raporcie przez liczenie sumy przychodów z platnosci
CREATE OR REPLACE FUNCTION aktualizuj_przychody()
RETURNS TRIGGER AS $$
BEGIN
 UPDATE raport_fiskalny
 SET suma_przychodow = (
  SELECT COALESCE(SUM(kwota), 0)
  FROM platnosc
  WHERE id_raportu = NEW.id_raportu
 )
 WHERE id_raportu = NEW.id_raportu;

 RETURN NULL;
END;
$$ LANGUAGE plpgsql;

-- INSERT
CREATE TRIGGER trg_przychody_insert
AFTER INSERT ON platnosc
FOR EACH ROW
EXECUTE FUNCTION aktualizuj_przychody();

-- UPDATE
CREATE TRIGGER trg_przychody_update
```

```
AFTER UPDATE ON platnosc
FOR EACH ROW
EXECUTE FUNCTION aktualizuj_przychody();

-- DELETE
CREATE TRIGGER trg_przychody_delete
AFTER DELETE ON platnosc
FOR EACH ROW
EXECUTE FUNCTION aktualizuj_przychody();

--trigger atkualizuj koszty w raporcie przez sume kosztów w usługach
CREATE OR REPLACE FUNCTION aktualizuj_koszty()
RETURNS TRIGGER AS $$
BEGIN
 UPDATE raport_fiskalny
 SET suma_kosztow = (
  SELECT COALESCE(SUM(koszt), 0)
  FROM usluga
  WHERE id_raportu = NEW.id_raportu
 )
 WHERE id_raportu = NEW.id_raportu;

 RETURN NULL;
END;
$$ LANGUAGE plpgsql;

-- INSERT
CREATE TRIGGER trg_koszty_insert
```

```
AFTER INSERT ON usluga
FOR EACH ROW
EXECUTE FUNCTION aktualizuj_koszty();

-- UPDATE
CREATE TRIGGER trg_koszty_update
AFTER UPDATE ON usluga
FOR EACH ROW
EXECUTE FUNCTION aktualizuj_koszty();

-- DELETE
CREATE TRIGGER trg_koszty_delete
AFTER DELETE ON usluga
FOR EACH ROW
EXECUTE FUNCTION aktualizuj_koszty();

--tryger aktualizujacy status platnosci na true kiedy zostanie zaplaconoa kwota rowna cenie usługi

CREATE OR REPLACE FUNCTION aktualizuj_status_platnosci()
RETURNS TRIGGER AS $$
DECLARE
  cena_uslugi DECIMAL(10,2);
BEGIN
  SELECT cena INTO cena_uslugi
  FROM usluga
  WHERE id_uslugi = NEW.id_uslugi;
```

```
  IF NEW.kwota = cena_uslugi THEN
    NEW.czy_zaplacono := TRUE;
  ELSE
    NEW.czy_zaplacono := FALSE;
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_aktualizuj_status_platnosci
BEFORE INSERT OR UPDATE ON platnosc
FOR EACH ROW
EXECUTE FUNCTION aktualizuj_status_platnosci();
```

--triger wyswietlacjacy blad jezeli szef będzie chciał usunąć pracownika który ma przydzielona jakas usluge

```
CREATE OR REPLACE FUNCTION blokuj_usuniecie_pracownika()
RETURNS TRIGGER AS $$
BEGIN
  IF EXISTS (
    SELECT 1 FROM usluga WHERE id_pracownika = OLD.id_pracownika
  ) THEN
    RAISE EXCEPTION 'Nie można usunąć pracownika, ponieważ ma przypisane usługi.';
  END IF;
  RETURN OLD;
```

```
END;
$$ LANGUAGE plpgsql;



CREATE TRIGGER trg_blokuj_usuniecie_pracownika
BEFORE DELETE ON pracownik
FOR EACH ROW
EXECUTE FUNCTION blokuj_usuniecie_pracownika();

--dane

-- Klienci
INSERT INTO klient (imie, nazwisko, telefon, email) VALUES
('Jan', 'Kowalski', '123456789', 'jan.kowalski@example.com'),
('Anna', 'Nowak', '987654321', 'anna.nowak@example.com'),
('Piotr', 'Wiśniewski', '111222333', 'piotr.wisniewski@example.com'),
('Kasia', 'Lewandowska', '444555666', 'kasia.lewandowska@example.com'),
('Michał', 'Kaczmarek', '777888999', 'michal.kaczmarek@example.com');

-- Samochody
INSERT INTO samochod (id_klienta, marka, model, nr_rejestracyjny, rok_produkcji) VALUES
(1, 'Toyota', 'Corolla', 'XYZ1234', 2015),
(2, 'Ford', 'Focus', 'ABC5678', 2018),
(3, 'Volkswagen', 'Golf', 'DEF2345', 2017),
(4, 'BMW', '320i', 'GHI3456', 2020),
(5, 'Audi', 'A4', 'JKL4567', 2019);

-- Pracownicy
```

```
INSERT INTO pracownik (imie, nazwisko, telefon, stanowisko) VALUES

('Marek', 'Zieliński', '555111222', 'Mechanik'),

('Ewa', 'Malinowska', '555333444', 'Kierownik'),

('Tomasz', 'Kowalczyk', '555555555', 'Mechanik'),

('Agnieszka', 'Wójcik', '555666777', 'Recepcjonistka'),

('Paweł', 'Nowicki', '555888999', 'Mechanik');


-- Raporty fiskalne

INSERT INTO raport_fiskalny (data_raportu) VALUES

('2025-06-01'),

('2025-06-02'),

('2025-06-03'),

('2025-06-04'),

('2025-06-05');


-- Usługi

INSERT INTO usluga (id_raportu, id_samochodu, id_pracownika, rodzaj_uslugi, opis, koszt,
cena, data_uslugi) VALUES

(1, 1, 1, 'Wymiana oleju', 'Wymiana oleju silnikowego i filtra', 100.00, 150.00, '2025-06-01'),

(2, 2, 2, 'Naprawa hamulców', 'Wymiana klocków hamulcowych', 200.00, 300.00, '2025-06-
02'),

(3, 3, 3, 'Przegląd techniczny', 'Comiesięczny przegląd', 50.00, 80.00, '2025-06-03'),

(4, 4, 4, 'Wymiana opon', 'Sezonowa wymiana opon', 150.00, 200.00, '2025-06-04'),

(5, 5, 5, 'Naprawa silnika', 'Naprawa układu zapłonowego', 500.00, 750.00, '2025-06-05');


-- Płatności

INSERT INTO platnosc (id_raportu, id_uslugi, data_platnosci, kwota, metoda_platnosci,
czy_zaplacono) VALUES

(1, 1, '2025-06-01', 150.00, 'karta', TRUE),
```

(2, 2, '2025-06-02', 300.00, 'gotowka', TRUE),

(3, 3, '2025-06-03', 80.00, 'karta', TRUE),

(4, 4, '2025-06-04', 100.00, 'gotowka', FALSE),

(5, 5, '2025-06-05', 750.00, 'karta', TRUE);