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:

Obraz zawierający diagram, origami, wzór

Zawartość wygenerowana przez AI może być niepoprawna.

Schemat relacji:

Obraz zawierający tekst, zrzut ekranu, numer, diagram

Zawartość wygenerowana przez AI może być niepoprawna.

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);