1. Strona Tytułowa

Temat projektu: System "Rent-A-Car OS" – Kompleksowe zarządzanie wypożyczalnią samochodów

Przedmiot: Bazy Danych

Uczelnia: Uniwersytet Rzeszowsi

Kierunek: Informatyka

Rok/Semestr: II rok, Semestr 3.

Wykonawcy:

1. Artur Jurkowski (Nr albumu: 134916)
2. Mikołaj Kopacz (Nr albumu: 134926)

Data: 14.01.2026

---

2. Specyfikacja tematu projektu

Projektowaną rzeczywistością jest średniej wielkości firma zajmująca się krótkoterminowym i średnioterminowym wynajmem samochodów osobowych ("Rent-A-Car").

System ma na celu informatyzację kluczowych procesów biznesowych przedsiębiorstwa, do których należą:

- **Zarządzanie flotą:** Ewidencja pojazdów, klasyfikacja cenowa, monitorowanie stanu technicznego oraz przebiegu.
- **Obsługa klientów:** Rejestracja nowych klientów, historia wypożyczeń, weryfikacja uprawnień (prawo jazdy) oraz system lojalnościowy (statusy VIP).
- **Proces rezerwacji:** Sprawdzanie dostępności aut w zadanym terminie, rezerwacja, wydanie i odbiór pojazdu, naliczanie opłat.
- **Zarządzanie personelem:** Konta pracowników, podział na role (Menadżer/Sprzedawca), monitorowanie efektywności sprzedaży.
- **Serwis i utrzymanie:** Prognozowanie przeglądów na podstawie przebiegu, rejestracja usterek i kosztów napraw.
- **Raportowanie finansowe:** Analiza przychodów w ujęciu miesięcznym i rocznym.

---

3. Aspekt projektowy bazy danych

3.1. Zdefiniowane struktury tabel

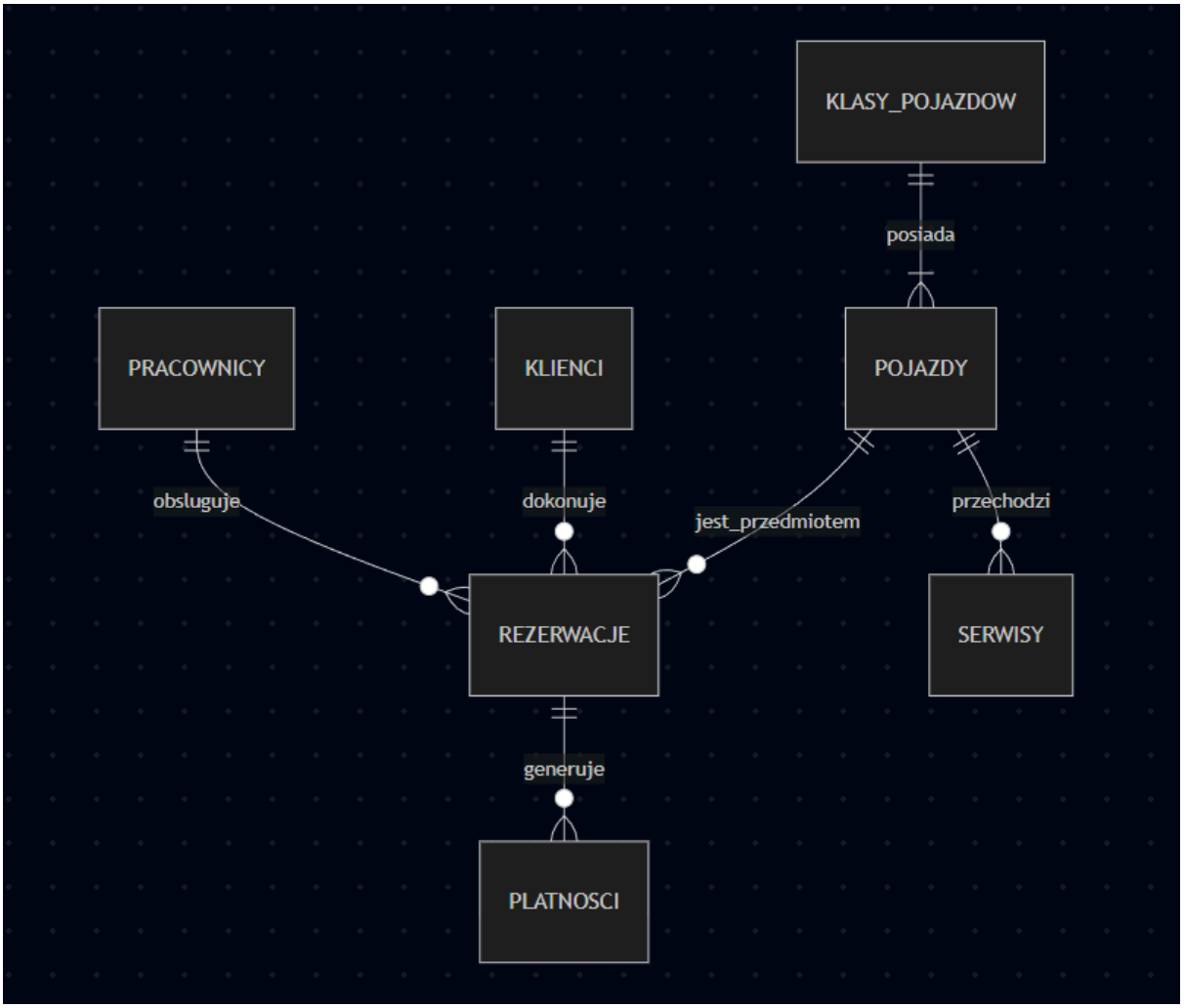
Baza danych składa się z następujących kluczowych encji:

1. **Klasy_Pojazdow**: Słownik klas (np. Economy, Premium) wraz z cennikiem za dobę.
2. **Pojazdy**: Główna tabela floty. Przechowuje markę, model, nr rejestracyjny, aktualny przebieg, stan techniczny oraz status dostępności. Powiązana z klasą cenową.
3. **Klienci**: Dane osobowe, kontaktowe oraz numer PESEL i prawa jazdy (unikalne).
4. **Pracownicy**: Dane personelu, login, hasło oraz stanowisko (definiujące uprawnienia w systemie).
5. **Rezerwacje**: Kluczowa tabela łącząca Klienta, Pojazd i Pracownika. Zawiera daty wynajmu, status (Potwierdzona/Zakończona/Anulowana) oraz wyliczoną cenę.
6. **Serwisy**: Historia napraw i przeglądów. Zawiera datę, koszt, opis usterki oraz przebieg w chwili serwisu.
7. **Platnosci**: Ewidencja wpłat za rezerwacje (gotówka/karta/przelew).

3.2. Powiązania pomiędzy tabelami

- **Pojazdy -> Klasy_Pojazdow (N:1):** Wiele pojazdów może należeć do tej samej klasy cenowej. Pozwala to na łatwą zmianę cennika dla grupy aut.
- **Rezerwacje -> Klienci (N:1):** Jeden klient może mieć wiele rezerwacji w historii, ale jedna rezerwacja dotyczy jednego klienta.
- **Rezerwacje -> Pojazdy (N:1):** Dany pojazd może być rezerwowany wielokrotnie (w różnych terminach).
- **Rezerwacje -> Pracownicy (N:1):** Rezerwacja jest obsługiwana przez konkretnego pracownika (rozliczalność).
- **Serwisy -> Pojazdy (N:1):** Samochód może mieć bogatą historię serwisową.
- **Platnosci -> Rezerwacje (N:1):** Do jednej rezerwacji może być przypisana płatność (lub w przyszłości wiele rat).

3.3. Diagram związków encji (ERD)



3.4. Kod realizujący bazę danych (DDL)

Struktura bazy danych jest tworzona za pomocą skryptu `setup.sql`. Poniżej fragment tworzący kluczowe tabele:

```
SQL
CREATE TABLE Klienci (
    ID_Klienta SERIAL PRIMARY KEY,
    Imie VARCHAR(50) NOT NULL,
    Nazwisko VARCHAR(50) NOT NULL,
    PESEL VARCHAR(11) UNIQUE NOT NULL,
    Numer_Prawa_Jazdy VARCHAR(20) UNIQUE NOT NULL,
    -- ... reszta kolumn
);
```

```
CREATE TABLE Rezerwacje (
    ID_Rezerwacji SERIAL PRIMARY KEY,
    ID_Klienta INT REFERENCES Klienci(ID_Klienta),
    ID_Pojazdu INT REFERENCES Pojazdy(ID_Pojazdu),
    -- ... klucze obce i daty
    CHECK (Data_Zwrotu >= Data_Odbioru)
);
-- Pełny kod znajduje się w pliku setup.sql
```

---

4. Aspekt projektowy funkcjonalności (Backend Bazy Danych)

Większość logiki biznesowej została przeniesiona do warstwy bazy danych (PostgreSQL) w postaci Procedur Składowanych i Funkcji, co zapewnia spójność danych i bezpieczeństwo.

4.1. Mechanizmy CRUD

Dla każdej głównej encji (Klienci, Pracownicy, Pojazdy) zaprojektowano zestaw procedur składowanych:

- **Create (Dodawanie):** `sp_dodaj_...` – Procedury te zawierają walidację danych (np. czy PESEL już istnieje, czy login jest wolny) przed wstawieniem rekordu.
- **Update (Edycja):** `sp_aktualizuj_...` – Pozwalają na zmianę danych z wykorzystaniem `COALESCE`, co umożliwia aktualizację tylko wybranych pól.
- **Delete (Usuwanie):** `sp_usun_...` – Zawierają logikę zabezpieczającą więzy integralności (np. blokada usunięcia klienta, który ma historię rezerwacji – RODO/wymogi księgowe).

4.2. Pytania algorytmiczne i projektowe

Zdefiniowano następujące problemy projektowe, rozwiązywane przez dedykowane funkcje SQL:

1. **Dostępność pojazdów:** Jak znaleźć wolne auto w zadanym terminie, wykluczając te, które są już zarezerwowane lub są w serwisie?
2. **Prognoza serwisowa:** Które auta wymagają przeglądu na podstawie przebiegu (np. co 15 000 km) lub zgłoszonej usterki?
3. **Raport finansowy:** Jakie są przychody firmy w rozbiciu na miesiące i metody płatności?
4. **Ranking VIP:** Którzy klienci wydają najwięcej pieniędzy i jaki status lojalnościowy im przysługuje?
5. **Efektywność pracowników:** Który pracownik wygenerował największy obrót dla firmy?

---

5. Przedstawienie powstałej bazy danych

5.1. Przykład implementacji CRUD (Tabela: Klienci)

Dodawanie Klienta (`sp_dodaj_klienta`): Procedura sprawdza unikalność numeru PESEL oraz Prawa Jazdy. Jeśli dane się powtarzają, rzucany jest wyjątek (`RAISE EXCEPTION`), który jest przechwytywany przez aplikację.

```
SQL
CREATE OR REPLACE PROCEDURE sp_dodaj_klienta(...) AS $$
BEGIN
   IF EXISTS (SELECT 1 FROM Klienci WHERE PESEL = p_pesel) THEN
      RAISE EXCEPTION 'Klient o tym PESEL już istnieje!';
   END IF;
   INSERT INTO Klienci (...) VALUES (...);
END;
$$;
```

Usuwanie Klienta (`sp_usun_klienta`): Zanim klient zostanie usunięty, sprawdzamy, czy nie posiada aktywnych lub archiwalnych rezerwacji. Jeśli posiada – operacja jest blokowana.

5.2. Realizacja pytań algorytmicznych

Problem 1: Wyszukiwanie dostępnych aut (`ZnajdzDostepnePojazdy`) *Kroki (Pseudokod):*

1. Pobierz listę wszystkich aut, które nie są w naprawie.
2. Dla każdego auta sprawdź tabelę `Rezerwacje`.
3. Jeśli istnieje rezerwacja, której zakres dat (Od-Do) nakłada się na żądany termin – odrzuć auto.
4. Zwróć listę pozostałych aut wraz z ceną (z tabeli Klasy).

Implementacja (PL/pgSQL): Wykorzystano operator nakładania się zakresów (`&&`) oraz typ `daterange`.

```
SQL
SELECT ... FROM Pojazdy p
WHERE NOT EXISTS (
   SELECT 1 FROM Rezerwacje r
   WHERE r.ID_Pojazdu = p.ID_Pojazdu
   AND daterange(r.Data_Odbioru, r.Data_Zwrotu, '[]') && daterange(p_data_od, p_data_do,
'[]')
);
```

Problem 2: Prognoza Serwisowa (`PrognozaSerwisowa`) *Kroki:*

1. Dla każdego pojazdu pobierz jego aktualny przebieg.
2. Znajdź przebieg przy ostatnim serwisie w tabeli `Serwisy`.
3. Oblicz różnicę: `Limit (15000) - (Aktualny - Ostatni_Serwis)`.
4. Jeśli wynik < 0 lub w polu Stan Techniczny widnieje usterka -> Zwróć ALERT.

---

6. Koncepcja dostępu zdalnego

System został zaprojektowany w architekturze **Klient-Serwer**:

- **Warstwa Danych (Serwer):** Baza danych PostgreSQL, przechowująca dane i logikę biznesową (procedury).
- **Warstwa Aplikacji (Klient/Serwer App):** Aplikacja napisana w języku **Python** z wykorzystaniem frameworka **Streamlit**.
- **Komunikacja:** Biblioteka `psycopg2` realizująca połączenia SQL.

Planowane funkcjonalności aplikacji:

1. **Panel Logowania:** Autoryzacja użytkowników (Pracownik/Menadżer).
2. **Pulpit (Dashboard):** Wizualizacja kluczowych statystyk (liczba aut, alerty serwisowe, wykres obłożenia).
3. **Moduł Floty:** Przeglądanie aut, szybkie zgłaszanie usterek, dodawanie/usuwanie aut (tylko Menadżer).
4. **Moduł Rezerwacji:** Wyszukiwanie aut, dodawanie klientów, generowanie potwierdzeń PDF.
5. **Moduł Raportów:** Podgląd finansów i efektywności pracowników.

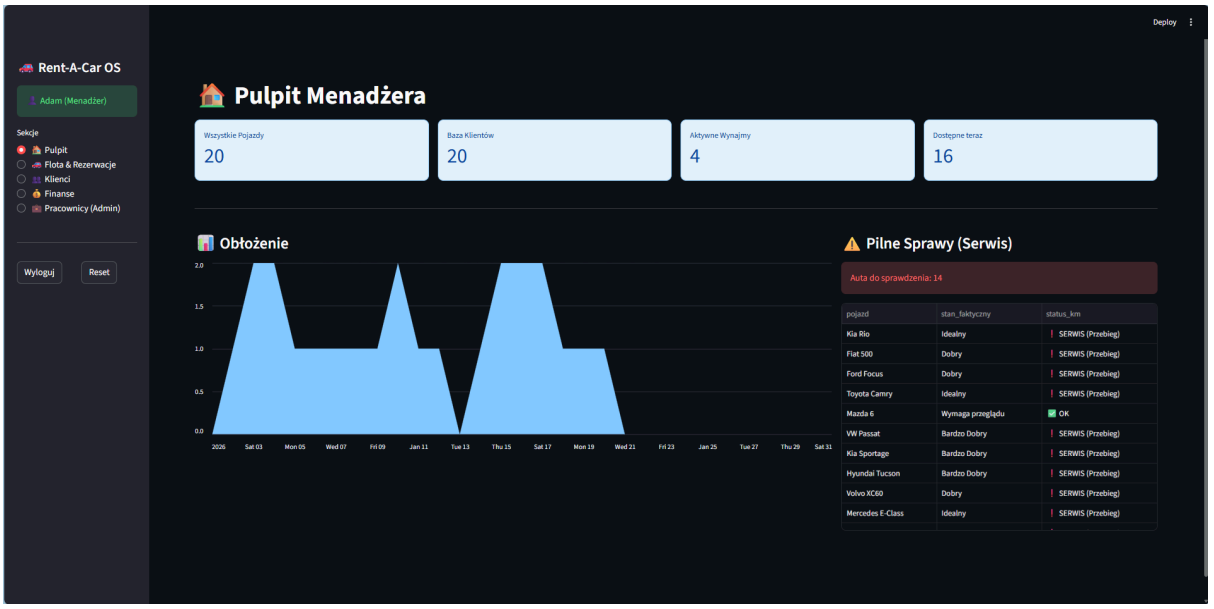
---

7. Opis realizacji dostępu zdalnego (Aplikacja)

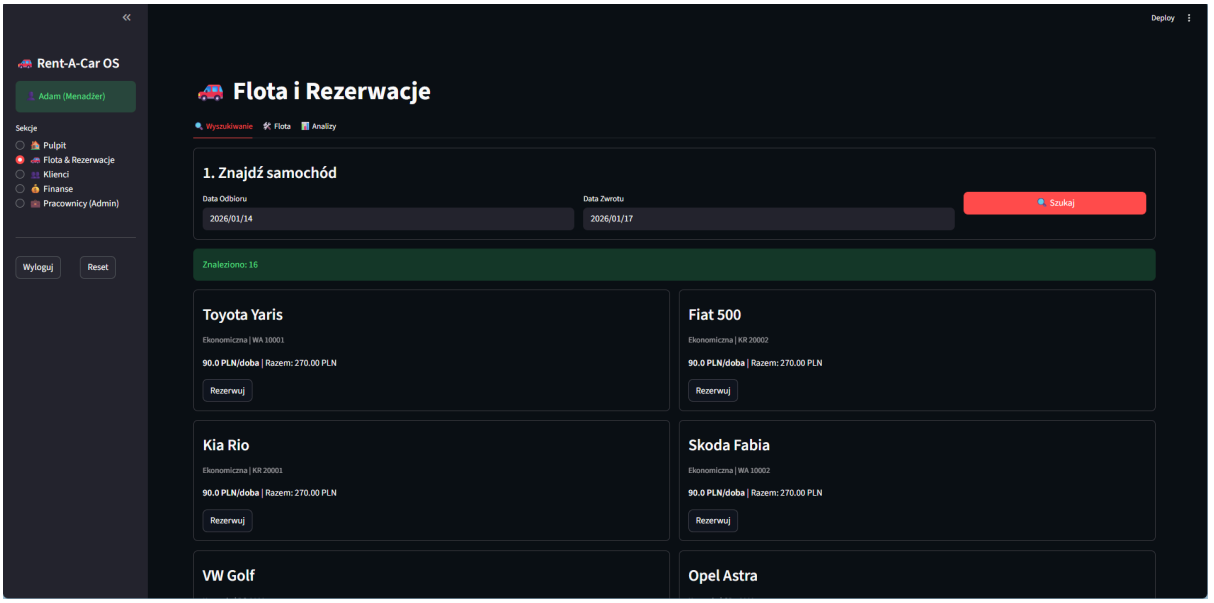
Aplikacja kliencka (`app.py`) zapewnia graficzny interfejs użytkownika (GUI) w przeglądarce internetowej.

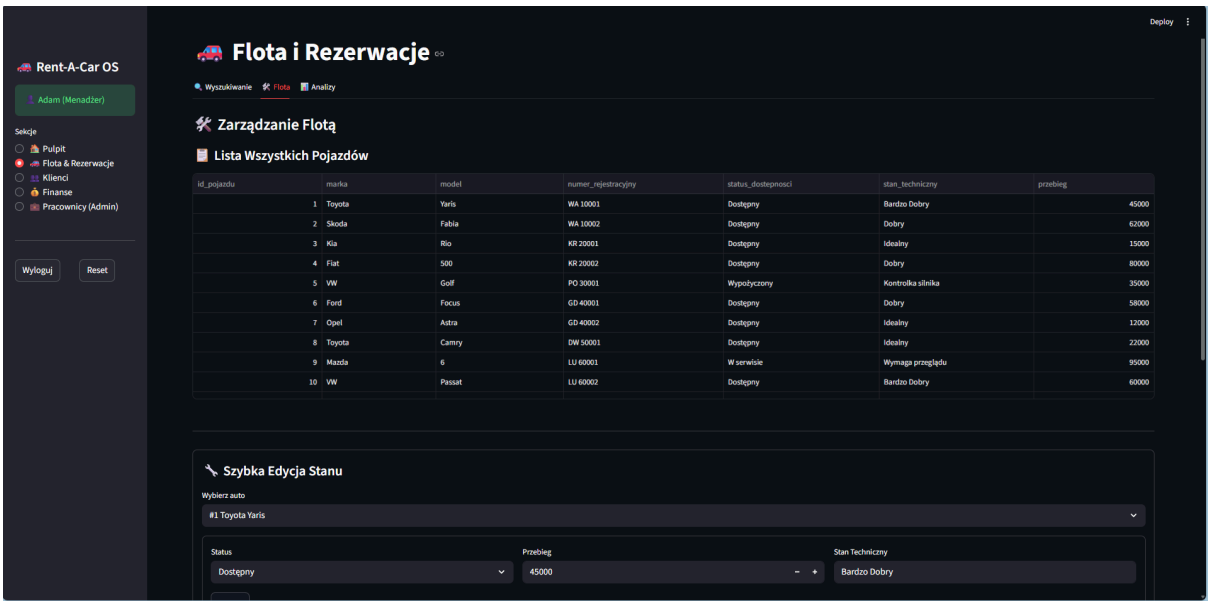
Główne ekrany aplikacji:

1. **Pulpit:** Wyświetla kafelki (Metrics) z liczbą dostępnych aut oraz czerwoną tabelę "Pilne Sprawy", która automatycznie pobiera dane z funkcji `PrognozaSerwisowa`.

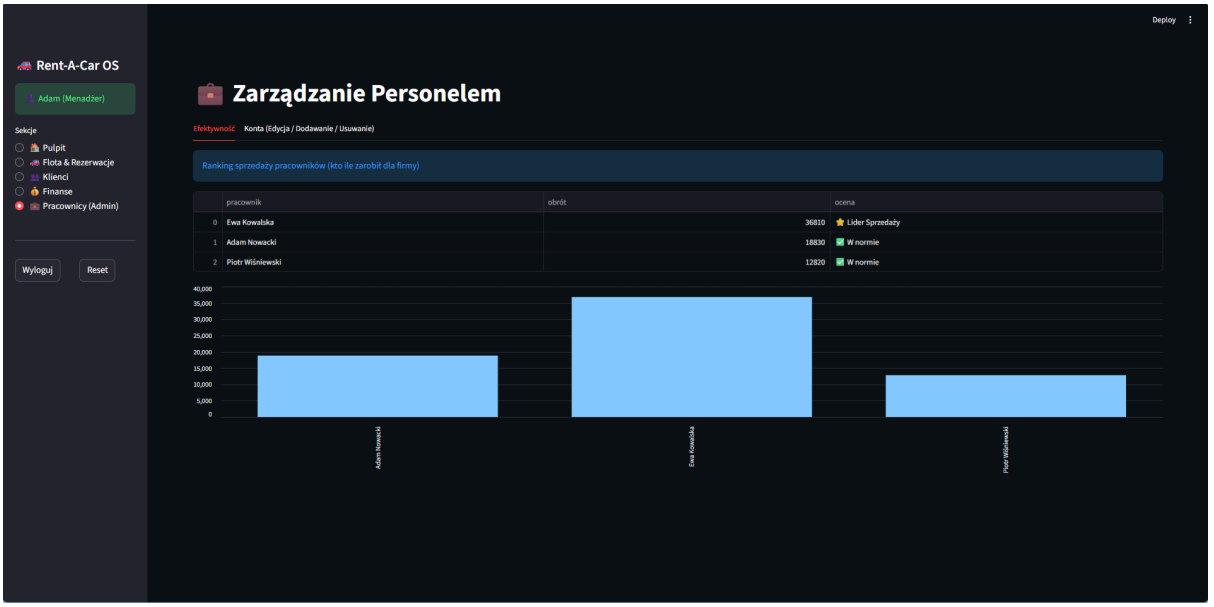


2. **Flota i Rezerwacje:** Umożliwia filtrowanie aut po datach. Po kliknięciu "Rezerwuj" wyświetla formularz danych klienta. Tutaj też zgłasza się usterki.





3. **Zarządzanie Personelem (Widok Admina):** Dostępny tylko dla zalogowanego Menadżera. Pozwala na dodawanie, edycję i usuwanie kont pracowników.



Instrukcja uruchomienia

Aby uruchomić system na własnym komputerze:

Wymagania:

- Zainstalowany Python (wersja 3.8+)
- Zainstalowany serwer PostgreSQL

Kroki:

1. **Baza danych:**
    a. Utwórz nową bazę danych w PostgreSQL (np. `wypozyczalnia`).
    b. Uruchom skrypt `setup.sql`, aby utworzyć strukturę tabel i funkcje.
    c. Uruchom skrypt `dane.sql`, aby wypełnić bazę przykładowymi danymi.
2. **Konfiguracja:**
    a. W pliku `src/db.py` dostosuj dane logowania do bazy (host, user, password, dbname).
3. **Instalacja bibliotek:** Otwórz terminal w folderze projektu i wpisz:

    pip install streamlit psycopg2-binary pandas fpdf

4. **Uruchomienie:** W terminalu wpisz:

    streamlit run src/app.py

5. Aplikacja uruchomi się automatycznie w domyślnej przeglądarce.
6. **Logowanie:**
    a. **Menadżer:** Login: `admin`, Hasło: `admin`
    b. **Pracownik:** Login: `ewa`, Hasło: `ewa`