Dokumentacja Techniczna Systemu Wypożyczalni Samochodów "Rent-A-Car OS"

Autor: Artur Jurkowski | Mikołaj Kopacz

Temat: System obsługi i zarządzania działalnością wypożyczalni samochodów z wykorzystaniem bazy danych PostgreSQL oraz aplikacji webowej Python Streamlit.

Link do strony:

https://projektbazydanych-tkiczs4cek4rkutnibup8v.streamlit.app/

Przedmiot: Bazy Danych **Kierunek:** Informatyka **Rok:** 2

---

1. Specyfikacja tematu projektu

1.1 Geneza problemu

Współczesny rynek wynajmu pojazdów, mimo rosnącej cyfryzacji, w sektorze małych i średnich przedsiębiorstw (MŚP) często opiera się na rozwiązaniach doraźnych. Wypożyczalnie korzystają z zeszytów rezerwacji, rozproszonych plików tekstowych lub arkuszy kalkulacyjnych, które nie są ze sobą zintegrowane.

Taki model zarządzania prowadzi do krytycznych problemów operacyjnych:

- **Brak integralności danych:** Informacje o klientach i flocie są niespójne (np. literówki w numerach VIN, dublowanie kartotek klientów).
- **Konflikty rezerwacji (Overbooking):** Brak centralnej walidacji dostępności aut w czasie rzeczywistym skutkuje przypisaniem jednego pojazdu dwóm klientom.
- **Ryzyko prawne:** Wydawanie pojazdów bez ważnych przeglądów technicznych z powodu braku automatycznych powiadomień.
- **Utrudniona analityka:** Brak narzędzi do szybkiej oceny rentowności floty i efektywności pracowników.

1.2 Założenia projektowe

Celem projektu było stworzenie zintegrowanego systemu "Rent-A-Car OS", który eliminuje "papierologię" i automatyzuje kluczowe procesy.

Główne filary systemu to:

1. **Architektura "Thick Database":** Przeniesienie logiki biznesowej (walidacja, obliczenia) do warstwy bazy danych PostgreSQL (procedury składowane, triggery, constraints).
2. **Centralizacja:** Wszystkie dane (CRM, Flota, Finanse) w jednej relacyjnej strukturze.
3. **Dostępność:** Interfejs webowy (Streamlit) dostępny przez przeglądarkę, niewymagający instalacji po stronie klienta.
4. **Bezpieczeństwo:** System ról (Menadżer/Pracownik) oraz mechanizmy zapobiegające usuwaniu danych historycznych.
5. **Skalowalność:** Wykorzystanie pomocniczej bazy NoSQL (MongoDB) do gromadzenia logów i telemetrii.

---

2. Analiza wymagań i narzędzia

2.1 Środowisko technologiczne

Projekt zrealizowano w architekturze klient-serwer, wykorzystując nowoczesne i wydajne narzędzia:

- **Warstwa Danych (Backend):** PostgreSQL 15+ (obsługa transakcji, procedury PL/pgSQL).
- **Warstwa Aplikacji (Middleware/Frontend):** Python 3.8+ z frameworkiem Streamlit.
- **Warstwa Logów (NoSQL):** MongoDB.
- **Środowisko programistyczne:** PyCharm / VS Code.

2.2 Kluczowe biblioteki

W pliku `requirements.txt` zdefiniowano zależności niezbędne do działania systemu:

- `streamlit` – framework do budowy interfejsu graficznego (GUI) w modelu webowym.
- `psycopg2-binary` – sterownik bazy danych PostgreSQL, zapewniający bezpieczne (parametryzowane) zapytania SQL.
- `pandas` – biblioteka do przetwarzania wyników zapytań SQL do postaci ramek danych (DataFrames), co ułatwia ich prezentację w tabelach i na wykresach.
- `fpdf` – biblioteka służąca do dynamicznego generowania plików PDF (potwierdzenia rezerwacji).

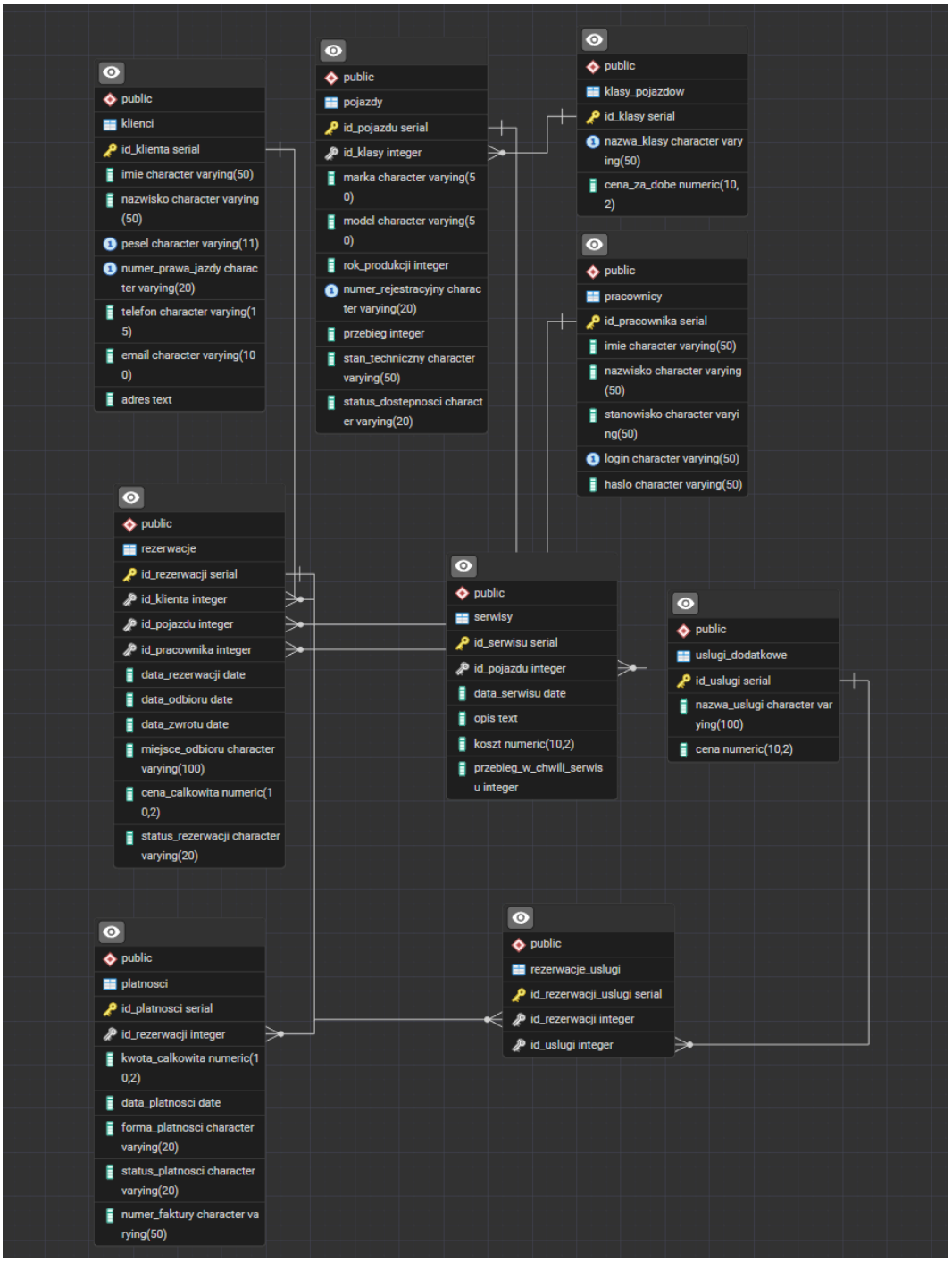
---

3. Projekt Bazy Danych (Struktura Tabel)

Baza danych została zaprojektowana w trzeciej postaci normalnej (3NF). Składa się z **9 tabel**, których szczegółowa specyfikacja znajduje się poniżej.

3.1 Schemat ERD

Diagram obrazuje relacje między encjami. Centralną tabelą jest `Rezerwacje`, łącząca klienta, pojazd i pracownika



Rys. 1. Schemat relacji bazy danych.

3.2 Opis szczegółowy tabel (DDL)

3.2.1 Tabela `Klasy_Pojazdow`

Słownik definiujący segmenty aut. Wydzielenie ceny do tej tabeli pozwala na globalną zmianę stawek dla całej grupy aut.

```SQL
CREATE TABLE Klasy_Pojazdow (
    ID_Klasy SERIAL PRIMARY KEY,
    Nazwa_Klasy VARCHAR(50) UNIQUE NOT NULL,
    Cena_Za_Dobe DECIMAL(10, 2) NOT NULL
);
```

3.2.2 Tabela `Pracownicy`

Zawiera dane personelu. Kolumna `Login` jest unikalna, co jest fundamentem systemu logowania.

```SQL
CREATE TABLE Pracownicy (
    ID_Pracownika SERIAL PRIMARY KEY,
    Imie VARCHAR(50) NOT NULL,
    Nazwisko VARCHAR(50) NOT NULL,
    Stanowisko VARCHAR(50),
    Login VARCHAR(50) UNIQUE NOT NULL,
    Haslo VARCHAR(50) NOT NULL
);
```

3.2.3 Tabela `Klienci`

Baza CRM. Zastosowano kluczowe ograniczenia `UNIQUE` dla numeru PESEL oraz Prawa Jazdy, aby zapobiec dublowaniu klientów.

```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,
    Telefon VARCHAR(15),
    Email VARCHAR(100),
    Adres TEXT
);
```

3.2.4 Tabela `Pojazdy`

Ewidencja floty. Constraint `CHECK` na polu `Status_Dostepnosci` wymusza jeden z trzech stanów: 'Dostępny', 'Wypożyczony', 'W serwisie'.

```SQL
CREATE TABLE Pojazdy (
    ID_Pojazdu SERIAL PRIMARY KEY,
    ID_Klasy INT REFERENCES Klasy_Pojazdow(ID_Klasy),
    Marka VARCHAR(50) NOT NULL,
    Model VARCHAR(50) NOT NULL,
    Rok_Produkcji INT,
    Numer_Rejestracyjny VARCHAR(20) UNIQUE NOT NULL,
    Przebieg INT,
    Stan_Techniczny VARCHAR(50),
    Status_Dostepnosci VARCHAR(20) CHECK (Status_Dostepnosci IN ('Dostępny', 'Wypożyczony', 'W serwisie'))
);
```

3.2.5 Tabela `Rezerwacje`

Kluczowa tabela transakcyjna. Zawiera walidację logiczną dat: data zwrotu musi być późniejsza lub równa dacie odbioru.

```SQL
CREATE TABLE Rezerwacje (
    ID_Rezerwacji SERIAL PRIMARY KEY,
    ID_Klienta INT REFERENCES Klienci(ID_Klienta),
    ID_Pojazdu INT REFERENCES Pojazdy(ID_Pojazdu),
    ID_Pracownika INT REFERENCES Pracownicy(ID_Pracownika),
    Data_Rezerwacji DATE DEFAULT CURRENT_DATE,
    Data_Odbioru DATE NOT NULL,
    Data_Zwrotu DATE NOT NULL,
    Miejsce_Odbioru VARCHAR(100),
    Cena_Calkowita DECIMAL(10, 2),
    Status_Rezerwacji VARCHAR(20) CHECK (Status_Rezerwacji IN ('Potwierdzona', 'Anulowana', 'Zakończona', 'W trakcie')),
    CHECK (Data_Zwrotu >= Data_Odbioru)
);
```

3.2.6 Tabela `Serwisy`

Rejestr napraw i przeglądów. Gromadzi historię kosztów utrzymania floty.

```SQL
CREATE TABLE Serwisy (
    ID_Serwisu SERIAL PRIMARY KEY,
    ID_Pojazdu INT REFERENCES Pojazdy(ID_Pojazdu),
    Data_Serwisu DATE NOT NULL,
    Opis TEXT,
    Koszt DECIMAL(10, 2),
    Przebieg_W_Chwili_Serwisu INT
);
```

3.2.7 Tabela `Uslugi_Dodatkowe`

Słownik usług płatnych extra (np. GPS, fotelik).

```SQL
CREATE TABLE Uslugi_Dodatkowe (
    ID_Uslugi SERIAL PRIMARY KEY,
    Nazwa_Uslugi VARCHAR(100) NOT NULL,
    Cena DECIMAL(10, 2) NOT NULL
);
```

3.2.8 Tabela `Rezerwacje_Uslugi`

Tabela łącząca (wiele-do-wielu), pozwalająca przypisać wiele usług dodatkowych do jednej rezerwacji.

```SQL
CREATE TABLE Rezerwacje_Uslugi (
    ID_Rezerwacji_Uslugi SERIAL PRIMARY KEY,
    ID_Rezerwacji INT REFERENCES Rezerwacje(ID_Rezerwacji),
    ID_Uslugi INT REFERENCES Uslugi_Dodatkowe(ID_Uslugi),
    UNIQUE(ID_Rezerwacji, ID_Uslugi)
);
```

3.2.9 Tabela `Platnosci`

Ewidencja wpływów finansowych z walidacją formy płatności (Gotówka/Karta/Przelew).

```SQL
CREATE TABLE Platnosci (
    ID_Platnosci SERIAL PRIMARY KEY,
```

```
    ID_Rezerwacji INT REFERENCES Rezerwacje(ID_Rezerwacji),
    Kwota_Calkowita DECIMAL(10, 2) NOT NULL,
    Data_Platnosci DATE DEFAULT CURRENT_DATE,
    Forma_Platnosci VARCHAR(20) CHECK (Forma_Platnosci IN ('Gotówka', 'Karta',
'Przelew')),
    Status_Platnosci VARCHAR(20) CHECK (Status_Platnosci IN ('Oczekująca',
'Zrealizowana', 'Anulowana')),
    Numer_Faktury VARCHAR(50)
);
```

---

4. Logika Biznesowa (Backend SQL)

Kluczowe operacje na danych zostały zamknięte w procedurach składowanych (Stored Procedures), co zapewnia bezpieczeństwo i spójność.

4.1 Kluczowe procedury i funkcje

A. Procedura `sp_dodaj_klienta` Bezpieczne dodawanie klienta. Sprawdza duplikaty (PESEL/Prawo Jazdy) przed wykonaniem `INSERT`. W przypadku błędu rzuca wyjątek `RAISE EXCEPTION`.

B. Funkcja `ZnajdzDostepnePojazdy` (Algorytm anty-overbooking) Najważniejsza funkcja systemu. Sprawdza, czy w zadanym przedziale dat auto nie ma kolizji z inną rezerwacją.

```
SQL
CREATE OR REPLACE FUNCTION ZnajdzDostepnePojazdy(...) RETURNS TABLE (...) AS
$$
BEGIN
    RETURN QUERY SELECT ... FROM Pojazdy p
    WHERE p.Status_Dostepnosci != 'W serwisie'
      AND NOT EXISTS (
          SELECT 1 FROM Rezerwacje r
          WHERE r.ID_Pojazdu = p.ID_Pojazdu
          AND r.Status_Rezerwacji IN ('Potwierdzona', 'W trakcie')
          AND daterange(r.Data_Odbioru, r.Data_Zwrotu, '[]') && daterange(p_data_od,
p_data_do, '[]')
      );
END;
$$ LANGUAGE plpgsql;
```

C. Procedura `sp_usun_klienta` Zabezpiecza przed usunięciem klienta posiadającego aktywne (niezwrócone) auto. Jeśli klient ma zakończone rezerwacje, jego dane w tabeli `Rezerwacje` są anonimizowane (`ID_Klienta = NULL`) przed usunięciem rekordu z tabeli głównej.

---

5. Implementacja Aplikacji (Kod Źródłowy Python)

Warstwa aplikacji została podzielona na dwa główne moduły: `db.py` (obsługa bazy) oraz `app.py` (interfejs użytkownika). Poniżej znajduje się analiza najważniejszych fragmentów kodu.

5.1 Moduł komunikacji z bazą (`src/db.py`)

Ten plik pełni rolę "mostu" między aplikacją a bazą danych. Separuje logikę interfejsu od logiki zapytań SQL.

A. Funkcja `get_connection()` Odpowiada za nawiązanie bezpiecznego połączenia z bazą PostgreSQL. Wykorzystuje bibliotekę `psycopg2`.

Python
```
def get_connection():
    config = get_db_config()
    # Nawiązanie połączenia z parametrami (host, user, password, dbname)
    return psycopg2.connect(**config)
```

B. Funkcja `run_command()` (Obsługa transakcji) Uniwersalna funkcja do wykonywania operacji modyfikujących dane (`INSERT`, `UPDATE`, `CALL`). Kluczowym elementem jest obsługa transakcyjności:

- `conn.commit()` – zatwierdza zmiany, jeśli operacja się powiodła.
- `conn.rollback()` – cofa zmiany w przypadku błędu, co chroni bazę przed niespójnością.

C. Funkcja `check_login()` Bezpieczna weryfikacja użytkownika. Używa zapytań parametryzowanych (`%s`), co **chroni aplikację przed atakami SQL Injection**.

Python
```
def check_login(username, password):
    sql = "SELECT ID_Pracownika... FROM Pracownicy WHERE Login=%s AND Haslo=%s"
    # Parametry są przekazywane jako krotka (tuple), sterownik sam je escapuje
    df = run_query(sql, (username, password))
    if not df.empty:
```

```
        return df.iloc[0].to_dict()
    return None
```

5.2 Moduł interfejsu użytkownika (`src/app.py`)

Aplikacja zbudowana w oparciu o framework **Streamlit**, który pozwala na tworzenie reaktywnych interfejsów webowych.

A. Zarządzanie Sesją (`st.session_state`) Aplikacja jest bezstanowa, dlatego wykorzystano obiekt `session_state` do przechowywania informacji o zalogowanym użytkowniku oraz stanie kreatora rezerwacji.

Python
```
if "logged_in" not in st.session_state:
    st.session_state["logged_in"] = False
# ...
if not st.session_state["logged_in"]:
    # Wyświetl ekran logowania
    with st.form("login_form"):
        # ... logika logowania
```

B. Generowanie PDF (`create_pdf_confirmation`) System wykorzystuje bibliotekę **FPDF** do dynamicznego tworzenia dokumentów potwierdzenia rezerwacji. Funkcja tworzy obiekt PDF, dodaje nagłówki, dane klienta i pojazdu, a następnie zwraca ciąg bajtów gotowy do pobrania przez użytkownika.

Python
```
def create_pdf_confirmation(klient_info, auto_info, ...):
    pdf = FPDF()
    pdf.add_page()
    pdf.set_font("Arial", "B", 16)
    pdf.cell(0, 10, clean_text("Potwierdzenie Rezerwacji"), ln=True, align="C")
    # ... dodawanie kolejnych linii tekstu
    return bytes(pdf.output())
```

C. Wizualizacja danych (Pulpit Menadżera) Aplikacja wykorzystuje komponenty `st.metric` do wyświetlania KPI oraz `st.area_chart` do rysowania wykresów obłożenia. Dane do wykresów są pobierane bezpośrednio z funkcji analitycznych SQL (`get_monthly_occupancy`).

[MIEJSCE NA ZDJĘCIE: Dashboard aplikacji] *Rys. 2. Widok panelu menadżerskiego w aplikacji.*

---

6. Integracja z NoSQL (MongoDB)

Uzupełnieniem systemu jest baza MongoDB, zdefiniowana w pliku `setup_mongo.js`.

Kolekcja `logs` – Służy jako Audit Log. Rejestruje każde logowanie, błąd oraz próbę usunięcia danych krytycznych. Struktura JSON pozwala na zapisywanie różnych detali (np. przeglądarka, IP) bez zmiany schematu bazy.

Kolekcja `telemetry` – Gromadzi dane z czujników IoT w samochodach (GPS, prędkość, paliwo). Wybór NoSQL podyktowany jest dużą ilością danych przychodzących (High Velocity).

---

7. Instrukcja Instalacji

- **Środowisko:** Zainstaluj `uv`, Python 3.8+ oraz bazy PostgreSQL i MongoDB.
- **Zależności:** `uv pip install -r requirements.txt`
- **Konfiguracja:**
  - Ustaw hasła do PostgreSQL w pliku `src/db.py`.
  - Upewnij się, że URI do MongoDB w `src/db.py` lub `src/app.py` jest poprawne.
- **Baza PostgreSQL (SQL):**
  - Struktura: `psql -f setup.sql`
  - Dane: `psql -f dane.sql`
- **Baza MongoDB (NoSQL):**
  - Inicjalizacja: `mongosh "mongodb://localhost:27017/CarRentalDB" --file src/setup_mongo.js`
- **Uruchomienie Aplikacji:** `uv run streamlit run src/app.py`

---

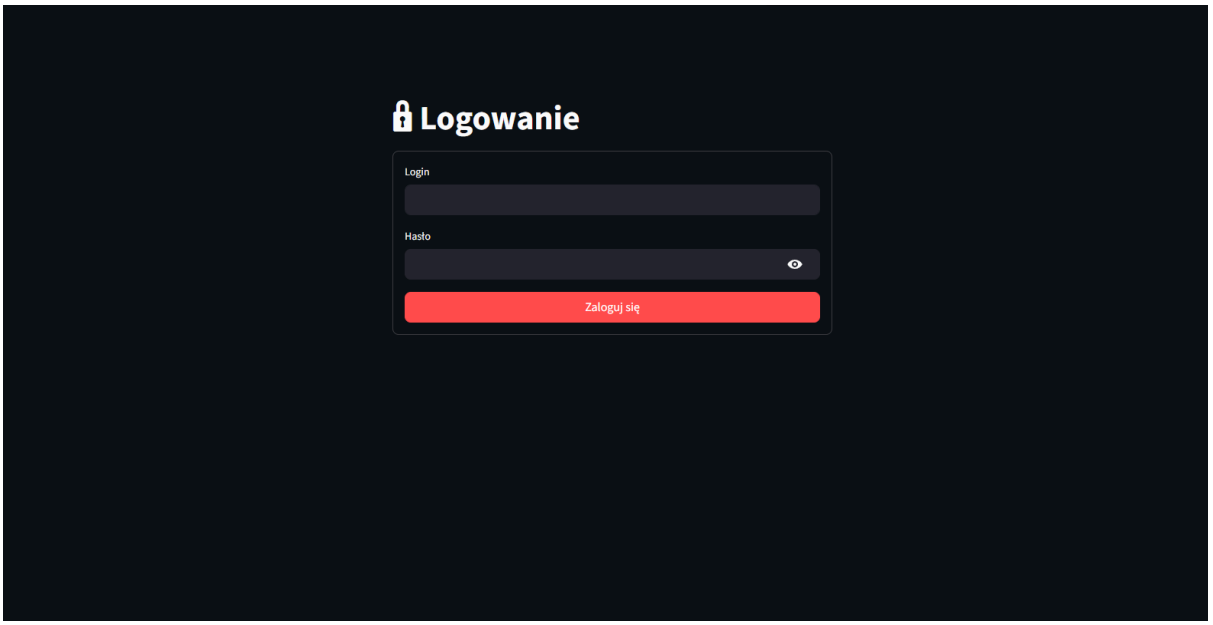
8. Przewodnik po Interfejsie Użytkownika

Aplikacja podzielona jest na panel boczny (nawigację) oraz główne okno robocze.

8.1. Ekran Logowania

Pierwszy ekran widoczny po uruchomieniu aplikacji.

- **Pole "Login":** Miejsce na wpisanie nazwy użytkownika (np. `admin`).
- **Pole "Hasło":** Miejsce na hasło (ukryte znaki).
- **Przycisk "Zaloguj się":** Weryfikuje dane w bazie `Pracownicy`. W przypadku sukcesu przenosi do Pulpitu.
- **Domyślne loginy (login | hasło):**
    - **Menadżer -** admin | admin
    - **Pracownik -** ewa | ewa



8.2. Menu Boczne (Sidebar)

Dostępne przez cały czas pracy z aplikacją.

- **Informacja o użytkowniku:** Wyświetla imię i rolę zalogowanej osoby.
- **Sekcja "Sekcje" (Radio Button):** Pozwala przełączać się między modułami:
    - 🏠 Pulpit
    - 🚗 Flota & Rezerwacje
    - 👥 Klienci
    - 💰 Finanse
    - 💼 Pracownicy (widoczne tylko dla Menadżera)
- **Przycisk "Wyloguj":** Kończy sesję i wraca do ekranu logowania.

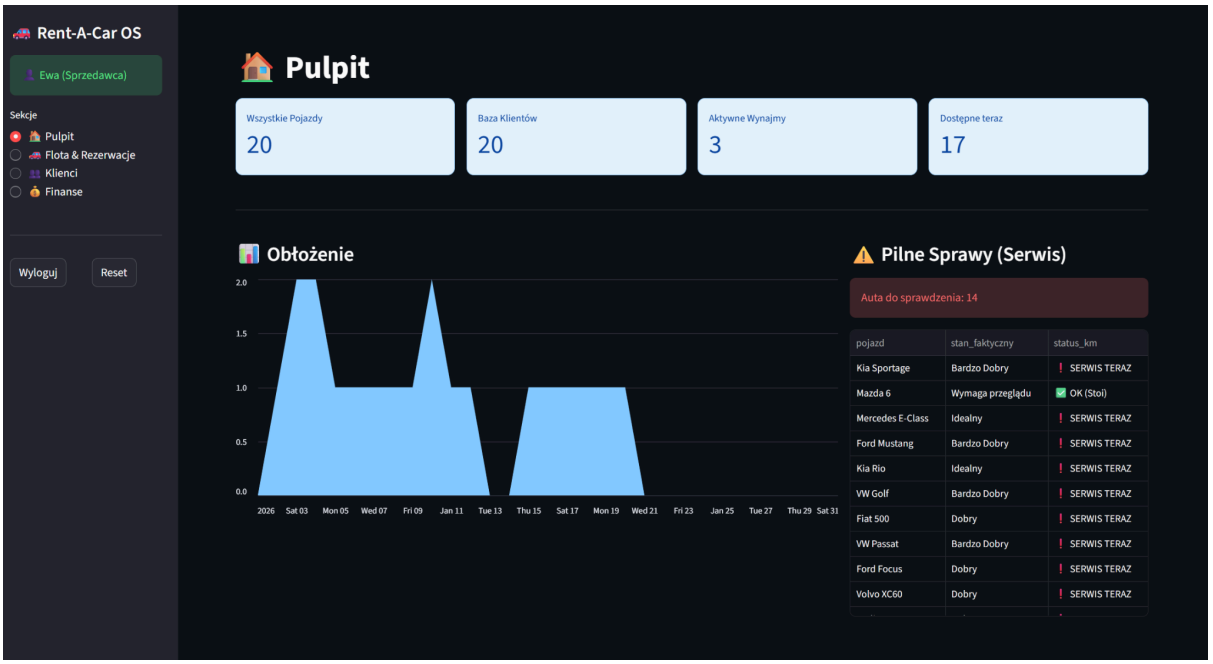
- **Przycisk "Reset":** Czyści pamięć podręczną sesji (np. formularze rezerwacji).

8.3. Moduł: 🏠 Pulpit

Centrum dowodzenia z najważniejszymi statystykami.

1. **Metryki (Górny rząd):**
   - Liczba pojazdów we flocie.
   - Liczba zarejestrowanych klientów.
   - Liczba aktywnych rezerwacji.
   - Liczba dostępnych obecnie aut (obliczana dynamicznie).
2. **Wykres "Obłożenie":** Wykres obszarowy pokazujący liczbę wypożyczonych aut w poszczególnych dniach bieżącego miesiąca.

3. **Sekcja "⚠️ Pilne Sprawy (Serwis)":** Tabela alertów. Wyświetla pojazdy, które przekroczyły limit kilometrów do serwisu lub mają zgłoszony zły stan techniczny.

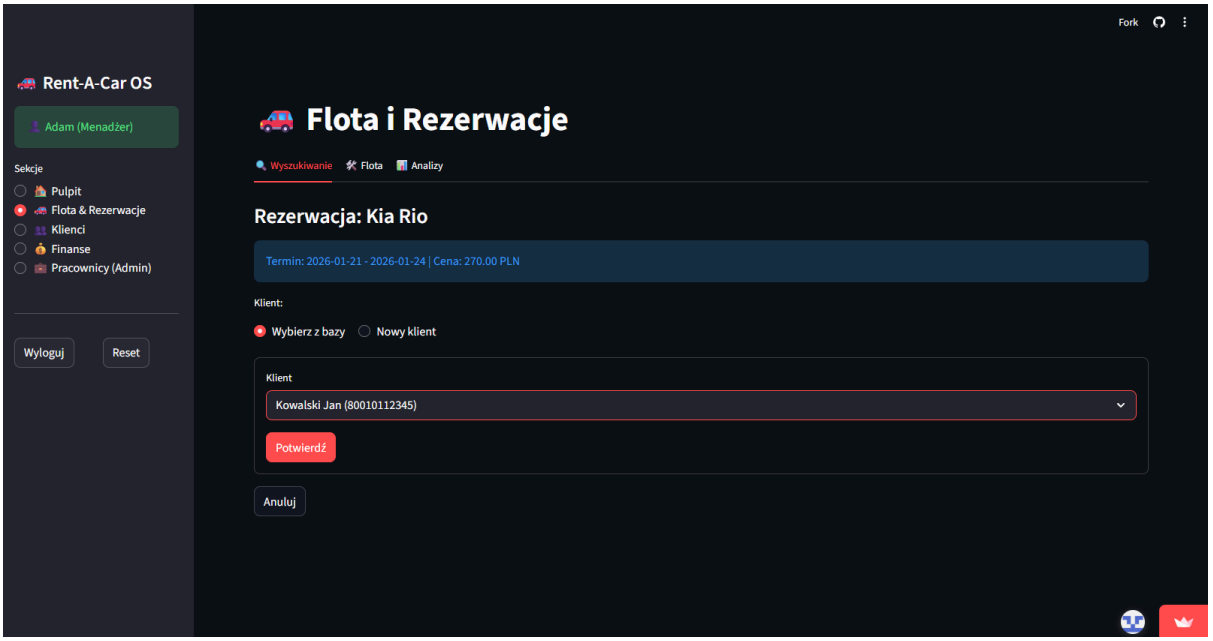


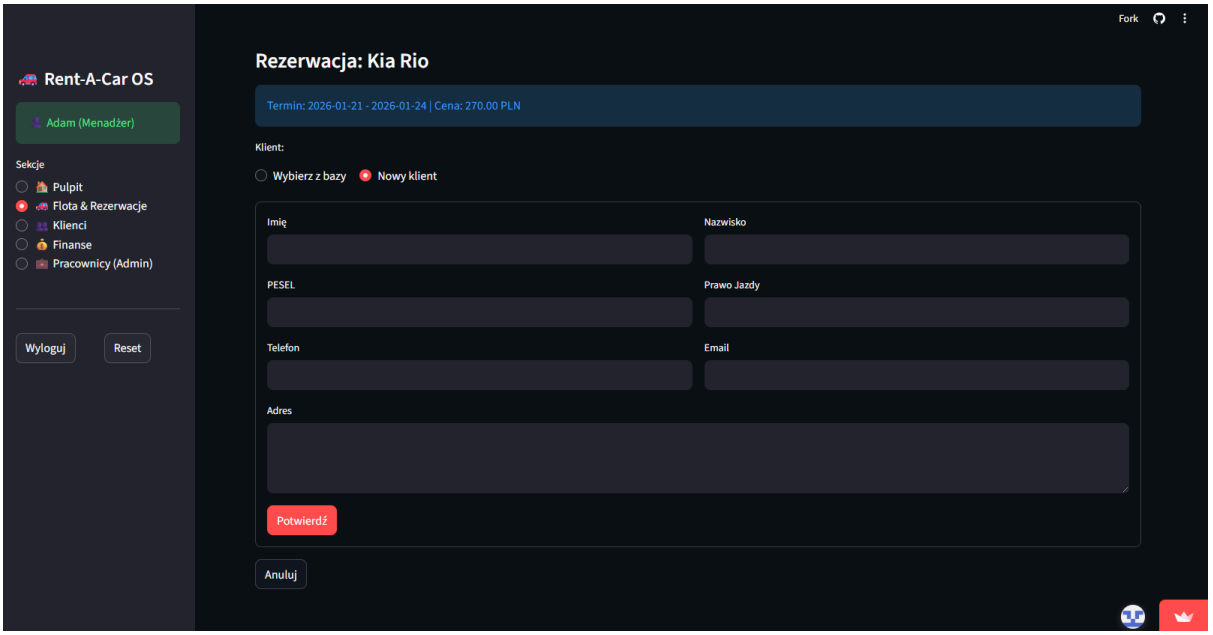
8.4. Moduł: 🚗 Flota & Rezerwacje

Moduł podzielony na trzy zakładki (Tabs).

Zakładka 1: 🔍 Wyszukiwanie (Proces Rezerwacji)

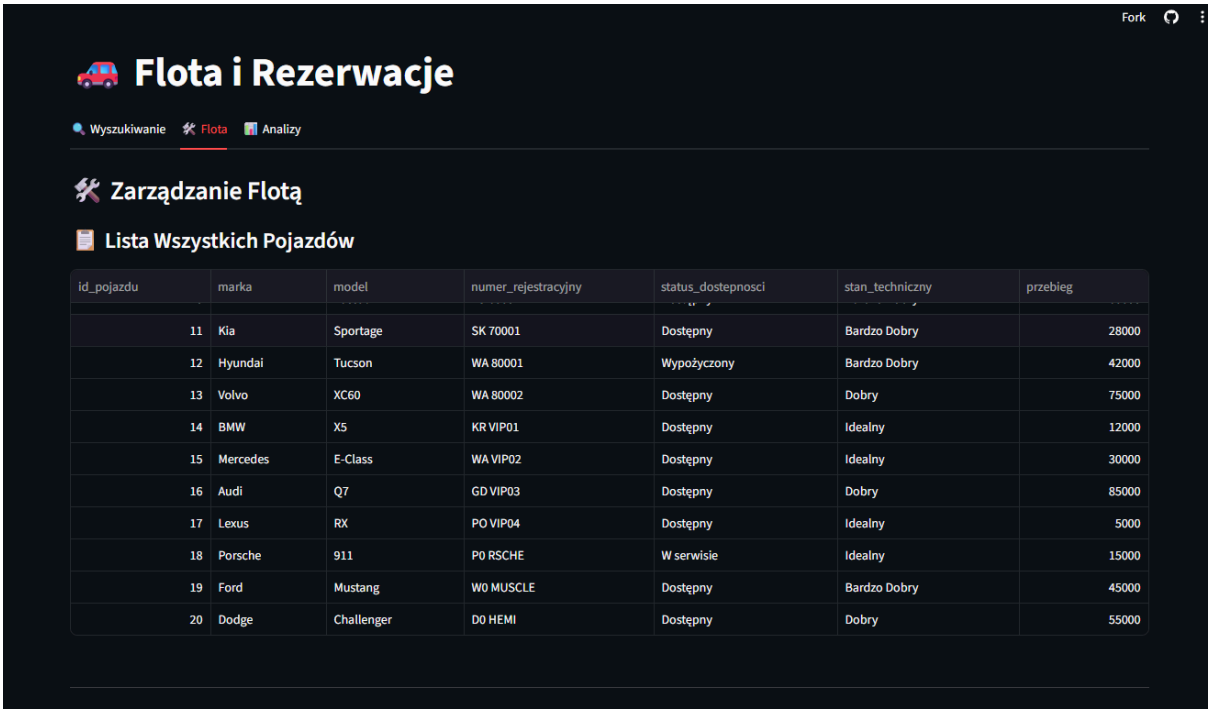
- **Pola "Data Odbioru" / "Data Zwrotu":** Kalendarze do wyboru terminu.
- **Przycisk "🔍 Szukaj":** Uruchamia funkcję SQL `ZnajdzDostepnePojazdy` i wyświetla listę wolnych aut.
- **Lista Wyników:** Karty z pojazdami (Zdjęcie/Nazwa/Cena).
- **Przycisk "Rezerwuj" (przy każdym aucie):** Przechodzi do formularza rezerwacji.
- **Formularz Rezerwacji:**
    - Opcja "Wybierz z bazy": Lista rozwijana obecnych klientów.
    - Opcja "Nowy klient": Pola do wpisania danych (Imię, Nazwisko, PESEL itd.).
    - **Przycisk "Potwierdź":** Finalizuje rezerwację w bazie.
    - **Przycisk "Anuluj":** Wraca do wyszukiwania.
- **Ekran Sukcesu:**
    - **Przycisk "Pobierz PDF":** Generuje i pobiera plik potwierdzenia rezerwacji.

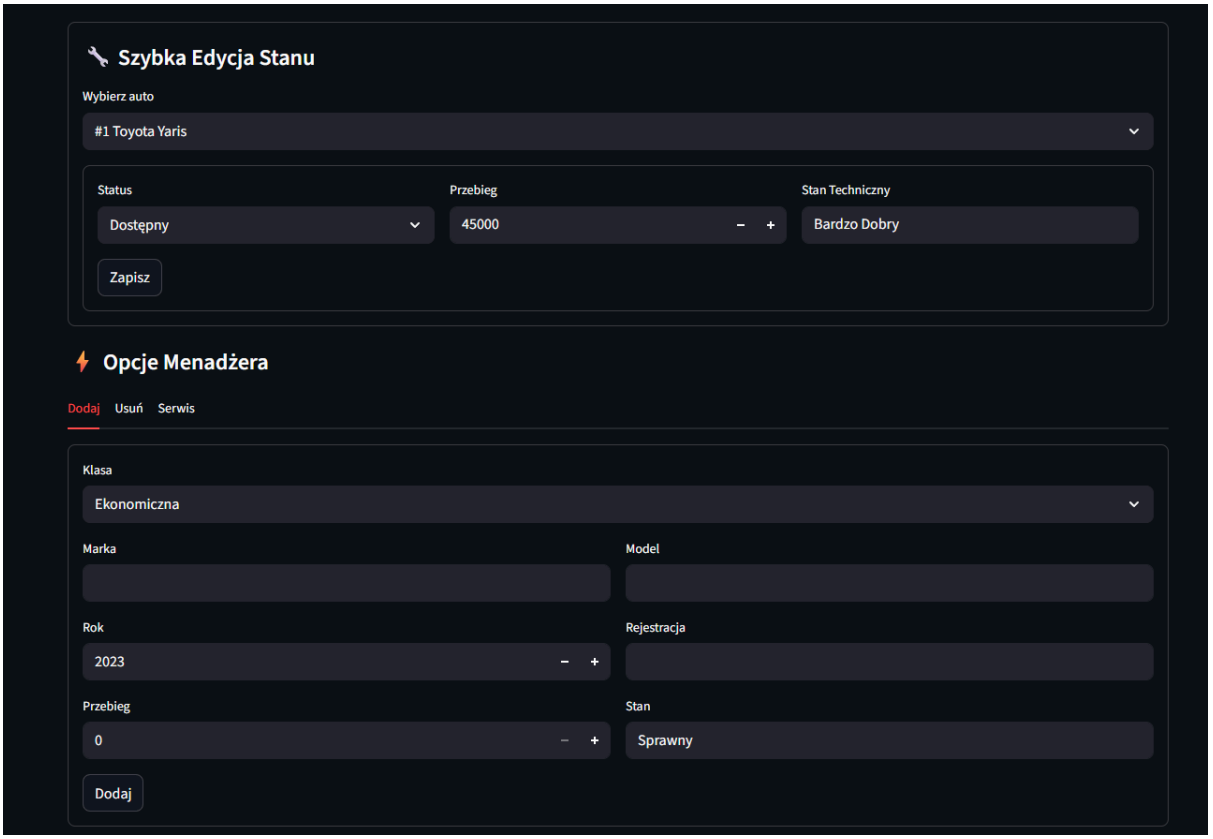




Zakładka 2: 🛠️ Flota

- **Tabela "Lista Wszystkich Pojazdów":** Podgląd całej floty.
- **Sekcja "🔧 Szybka Edycja Stanu":**
  - Lista rozwijana do wyboru auta.
  - Pola edycji: Status (Dostępny/Serwis), Przebieg, Stan Techniczny.
  - **Przycisk "Zapisz":** Aktualizuje status pojazdu (np. przy zwrocie).
- **Sekcja Menadżera (Dostępna dla Admina):**
  - *Tab "Dodaj":* Formularz dodawania nowego auta do bazy. Przycisk **"Dodaj"**.
  - *Tab "Usuń":* Wybór auta do usunięcia. Przycisk **"Usuń trwale"**.
  - *Tab "Serwis":* Formularz rejestracji naprawy (Koszt, Opis). Checkbox **"Naprawa zakończona?"** (automatycznie przywraca dostępność auta). Przycisk **"Rejestruj"**.

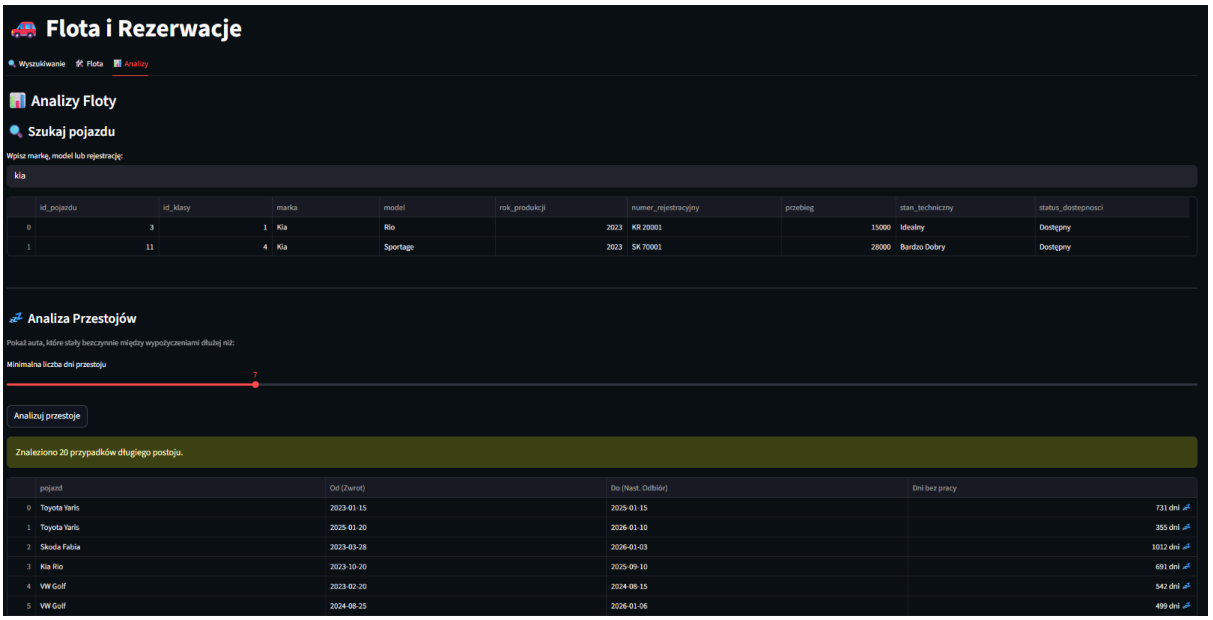




Zakładka 3: 📊 Analizy

- **Wyszukiwarka:** Pole tekstowe do szukania auta po nr rej. lub modelu.
- **Analiza Przestojów:**

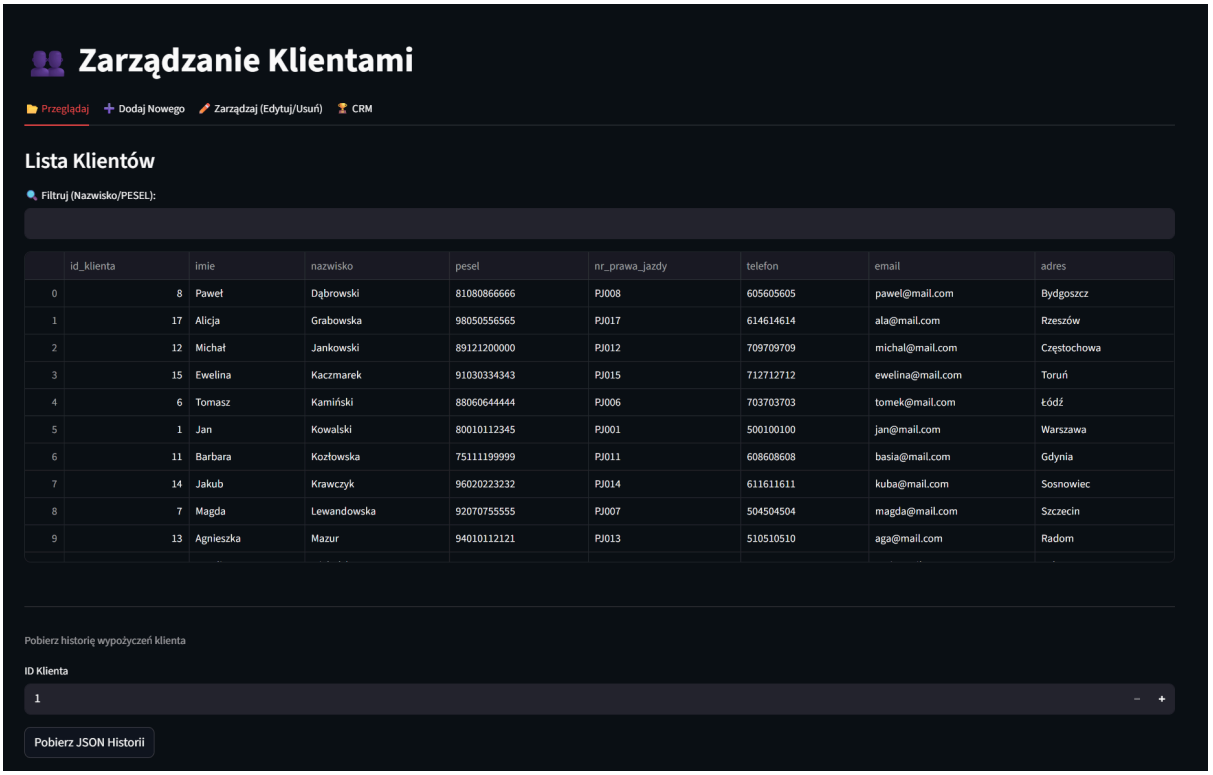
- Suwak "Minimalna liczba dni przestoju".
- **Przycisk "Analizuj przestoje":** Wyświetla tabelę aut, które "kurzą się" na parkingu.



8.5. Moduł: 👥 Klienci

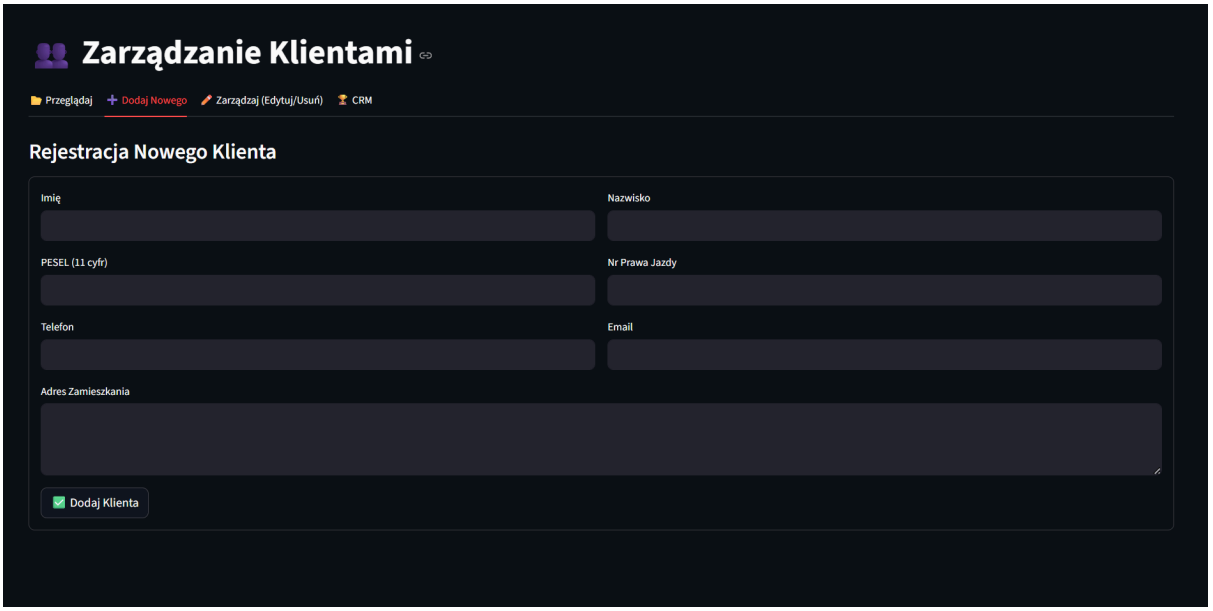
Podzielony na cztery zakładki.

1. 📂 **Przeglądaj:** Tabela klientów z polem filtrowania.
    - **Sekcja Historii:** Po wpisaniu ID klienta i kliknięciu **"Pobierz JSON Historii"**, wyświetla surowy zapis historii wypożyczeń z bazy danych.



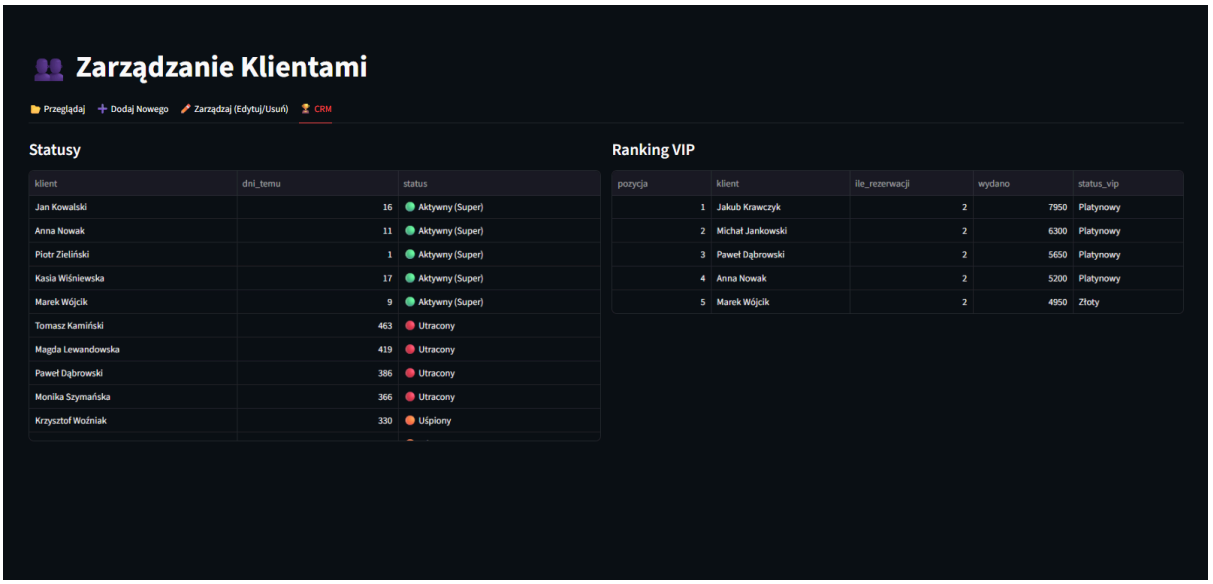
.

2. ➕ **Dodaj Nowego:** Formularz rejestracji klienta (PESEL, Prawo Jazdy, Kontakt). Przycisk **"Dodaj Klienta"**.



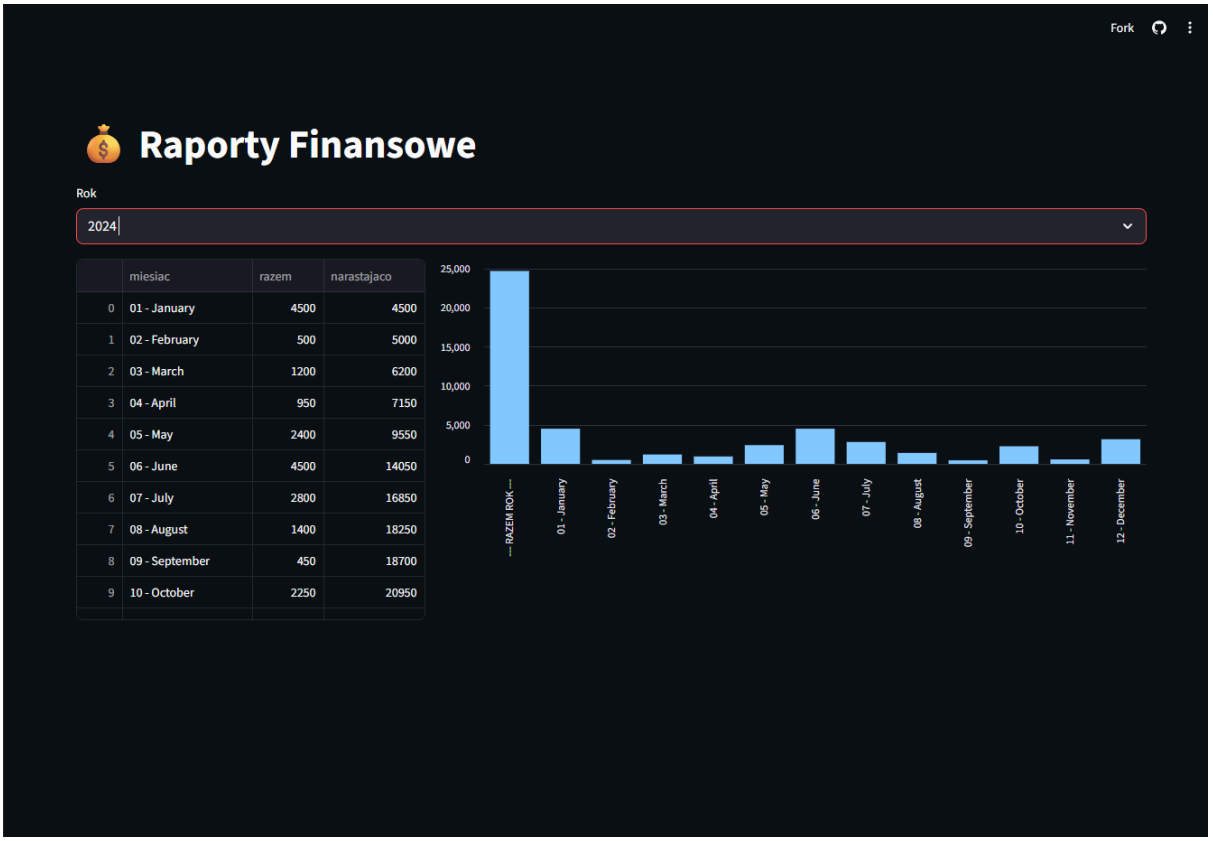
3. ✏️ **Zarządzaj:** Edycja danych osobowych (Przycisk **"Zapisz Zmiany"**) lub usuwanie klienta z bazy (Przycisk **"Usuń trwale"** – działa tylko, jeśli klient nie ma historii, zgodnie z integralnością danych).

4. 🏆 **CRM:**
    - Tabela statusów (kto jest aktywny, kto uśpiony).
    - Ranking VIP (klienci, którzy wydali najwięcej)



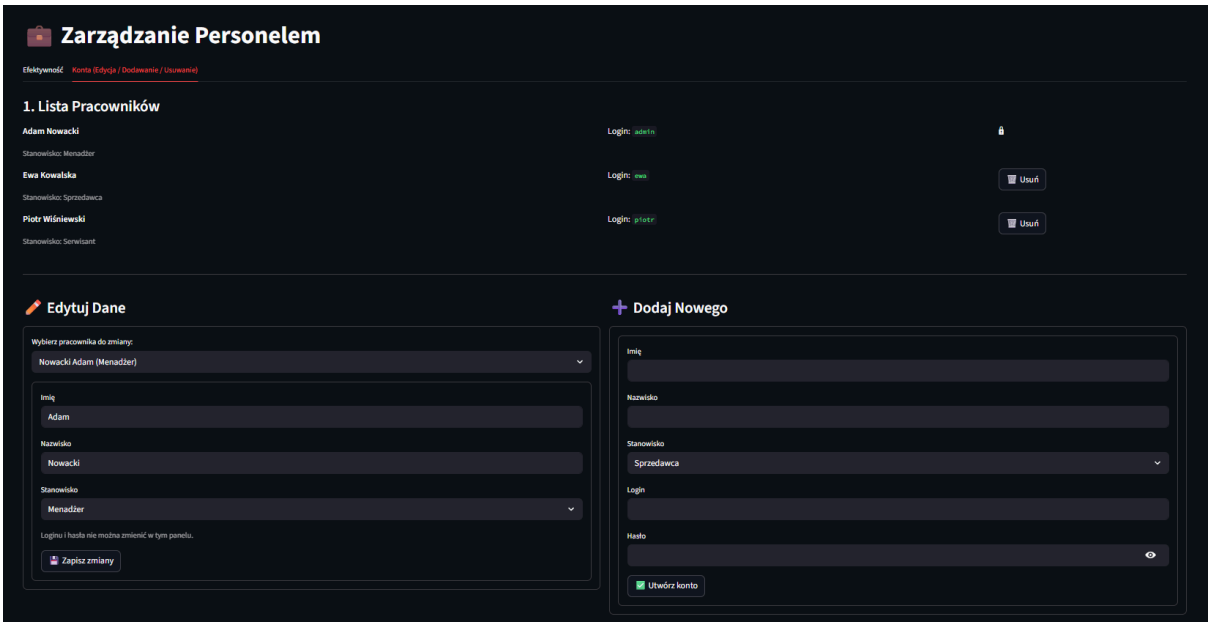
8.6. Moduł: 💰 Finanse

- **Wybór Roku:** Lista rozwijana lat.
- **Tabela i Wykres:** Raport przychodów w podziale na miesiące oraz metoda płatności (Gotówka/Karta/Przelew).



8.7. Moduł: 💼 Pracownicy (Tylko Admin)

1. **Tab "Efektywność":** Ranking pracowników pod względem wygenerowanego obrotu (wykres słupkowy).
2. **Tab "Konta":**
    - Lista pracowników z przyciskami **"Usuń"** (ikona kosza).
    - Edycja danych: Zmiana stanowiska lub nazwiska. Przycisk **"Zapisz zmiany"**.
    - Dodawanie pracownika: Formularz tworzenia nowego loginu i hasła. Przycisk **"Utwórz konto"**.



---

9. Podsumowanie

Projekt "Rent-A-Car OS" to kompletne, gotowe do wdrożenia rozwiązanie klasy ERP dla małych wypożyczalni.

Najważniejsze osiągnięcia:

1. **Pełna cyfryzacja:** System obsługuje cykl życia pojazdu od zakupu, przez wynajmy i serwisy, aż po sprzedaż.
2. **Bezpieczeństwo architektury:** Separacja logiki (SQL) od prezentacji (Python) oraz użycie zapytań parametryzowanych zapewnia odporność na błędy i ataki.

3. **Wsparcie decyzji:** Dashboardy menadżerskie i raporty finansowe dostarczają wiedzy niezbędnej do rozwoju biznesu.
4. **Użyteczność:** Generowanie gotowych do druku plików PDF znacznie przyspiesza pracę obsługi klienta.

---

10. Bibliografia

1. Dokumentacja PostgreSQL 15 – Procedury składowane i typy danych.
2. Dokumentacja Streamlit – Zarządzanie stanem aplikacji i komponenty UI.
3. Dokumentacja PyFPDF – Tworzenie dokumentów PDF w Pythonie.
4. Materiały wykładowe z przedmiotu Bazy Danych (Uniwersytet Rzeszowski).