Bazy Danych 1 - Dokumentacja Projektu

Helena Jońca

Wypożyczalnia samochodów

1 Projekt koncepcji, założenia

1.1 Zdefiniowanie tematu projektu

Motywem przewodnim projektu jest stworzenie systemu zarządzania wypożyczalnią samochodową z wykorzystaniem baz danych. Celem projektu jest opracowanie funkcjonalnego narzędzia, które umożliwia skuteczne zarządzanie danymi związanymi z wynajmem samochodów, pracownikami oraz klientami. System realizuje następujące zadania:

- dodawanie i przeglądanie rekordów dotyczących wynajmu samochodów, klientów i pracowników,
- generowanie raportów w oparciu o zgromadzone dane,
- zarządzanie strukturą bazy danych, w tym dodawanie i usuwanie rekordów.

1.2 Analiza wymagań użytkownika

Projektowana baza danych systemu zarządzania wypożyczalnią samochodową ma na celu zapewnienie wsparcia w zarządzaniu danymi dotyczącymi wynajmów, klientów, pracowników oraz pojazdów. System umożliwia przechowywanie szczegółowych informacji o markach i modelach samochodów, ich dostępności oraz historii wynajmów.

Baza danych wspiera także obsługę płatności oraz zamówień, umożliwiając rejestrowanie statusu realizacji transakcji i zamówień na konkretne modele pojazdów. Dzięki wbudowanym mechanizmom walidacji, system dba o spójność i poprawność danych, takich jak unikalność numerów rejestracyjnych, poprawność formatów danych kontaktowych czy weryfikacja logicznej kolejności dat w przypadku wynajmów i zamówień.

System wspiera podstawowe operacje, takie jak dodawanie i usuwanie danych, a także dostarcza raporty i zestawienia, które pozwalają na analizę kluczowych danych, takich jak dostępność samochodów, wyniki finansowe czy efektywność działań pracowników.

Funkcjonalności, jakie ma spełniać baza danych:

- **Rejestracja pojazdów:** Przechowywanie danych o samochodach, takich jak marka, model, numer rejestracyjny, klasa, rok produkcji, przebieg oraz dostępność.
- **Rejestracja klientów:** Przechowywanie danych osobowych klientów, takich jak imię, nazwisko, numer telefonu, adres e-mail oraz historia wypożyczeń.
- **Obsługa wypożyczeń:** Możliwość dodawania nowych wypożyczeń, z uwzględnieniem daty rozpoczęcia i zakończenia, przypisania samochodu i klienta, oraz obliczania kosztów.
- **Zarządzanie cennikiem:** Powiązanie klas pojazdów z odpowiednimi stawkami za dzień wypożyczenia.
- **Monitorowanie dostępności:** Sprawdzanie dostępnych samochodów w określonym przedziale czasowym.
- **Obsługa płatności:** Automatyczne generowanie i przechowywanie płatności związanych z wypożyczeniami.

- **Raporty i analizy:** Generowanie raportów o liczbie dostępnych samochodów, popularności marek i modeli, oraz zestawień finansowych.
- **Zarządzanie użytkownikami systemu:** Możliwość dodawania i zarządzania pracownikami wypożyczalni.

1.3 Zaprojektowanie funkcji: określenie podstawowych funkcji realizowanych w bazie danych.

W ramach projektu zaprojektowano zestaw funkcji, które realizują kluczowe operacje związane z przetwarzaniem danych. Ich głównym celem jest zapewnienie spójności, poprawności oraz łatwego dostępu do danych. Do podstawowych funkcjonalności systemu należą:

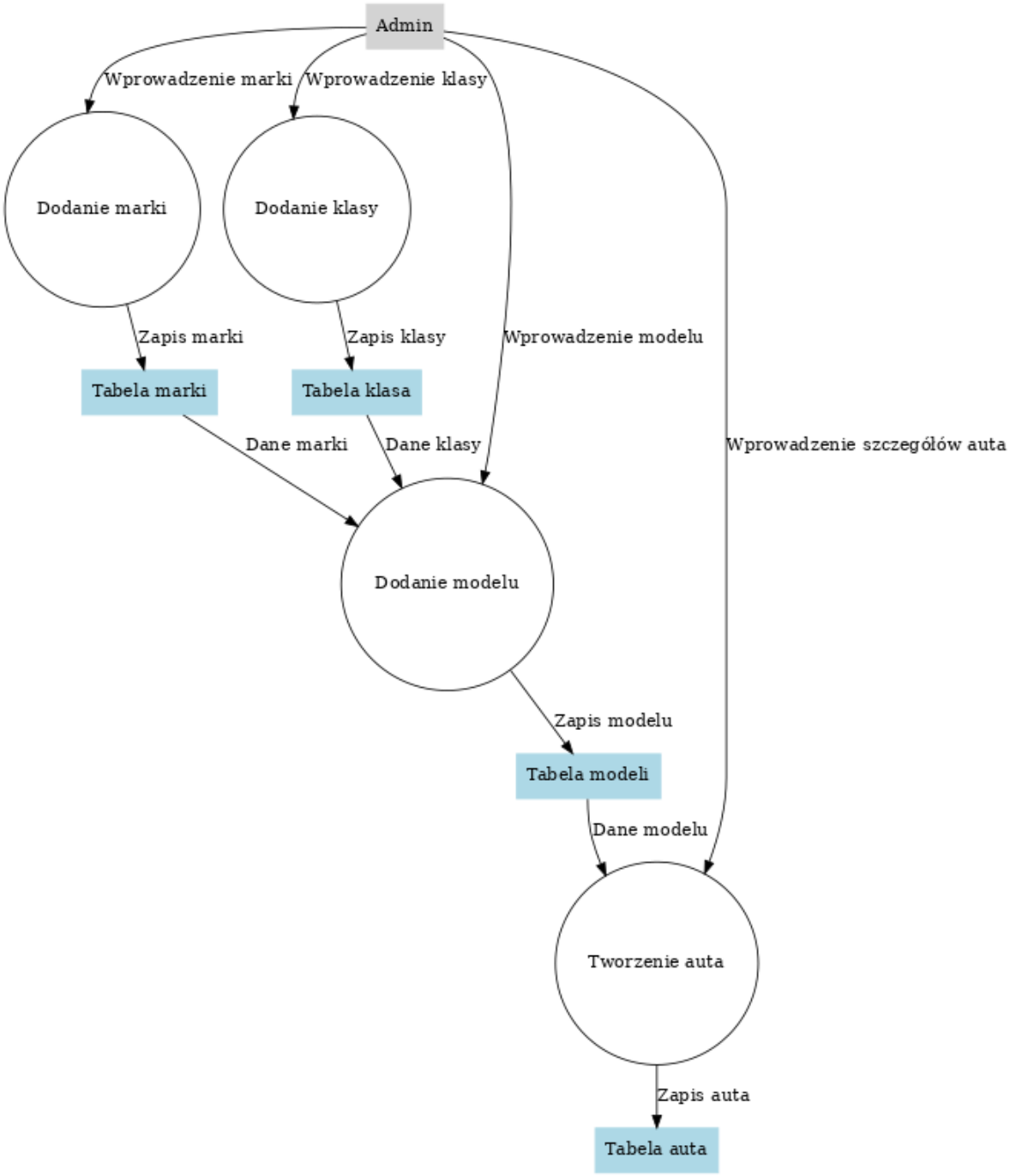
- **Zarządzanie danymi**: W bazie danych zaimplementowano funkcje i widoki umożliwiające łatwe wyszukiwanie modeli oraz marek samochodów na podstawie ich powiązań w systemie.
- **Automatyzacja procesów**: zaprojektowano mechanizmy wspierające automatyczne wybieranie dostępnych pojazdów oraz pracowników do realizacji wynajmu.
- **Walidacja danych**: system kontroluje poprawność wprowadzanych danych, takich jak numery rejestracyjne pojazdów czy numery telefonów, aby zapewnić integralność bazy.
- **Generowanie raportów**: widoki i funkcje wspierają tworzenie przejrzystych zestawień oraz analiz danych, takich jak koszty wynajmów czy szczegóły dotyczące transakcji.

2 Projekt diagramów

2.1 Budowa i analiza diagramu przepływu danych (DFD)

2.1.1 Klasa, Model, Marka, Auto

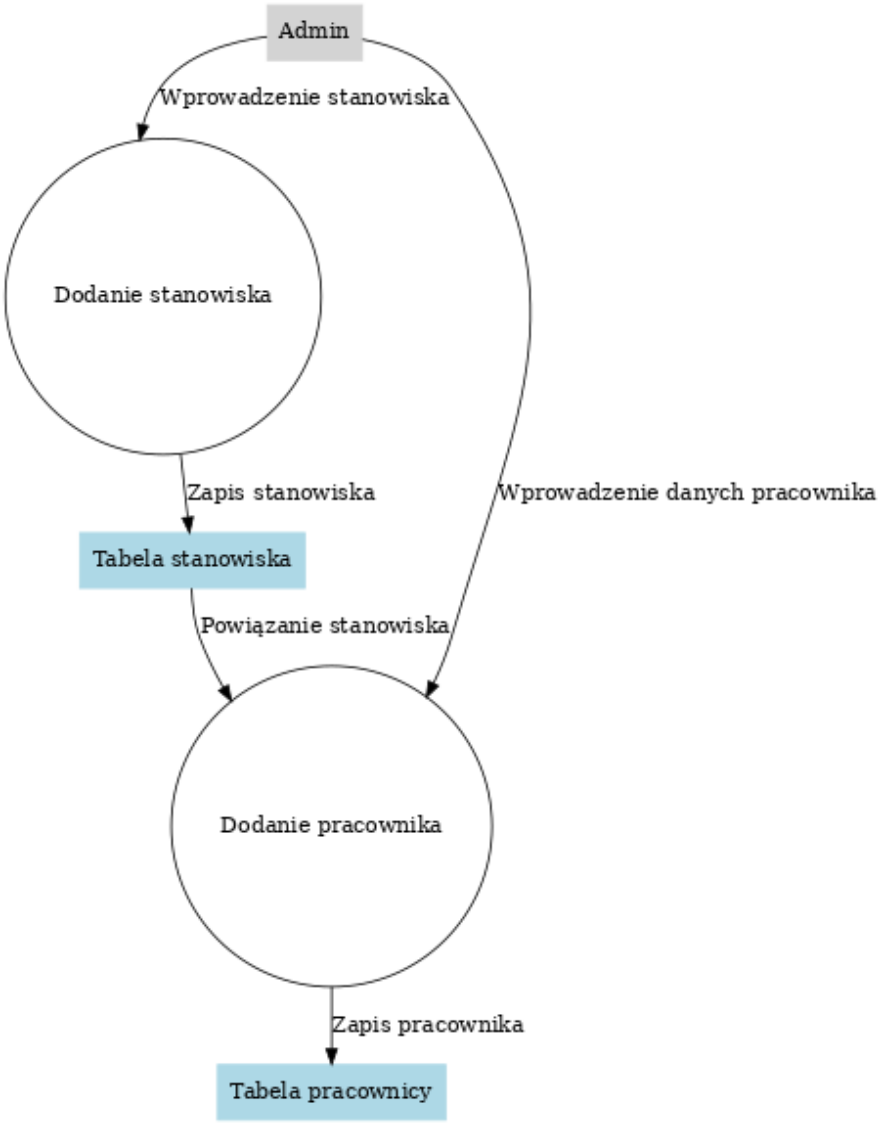
Administrator (użytkownik) wprowadza dane o marce i klasie pojazdu, które są następnie używane do zdefiniowania konkretnego modelu. Informacje o modelu są zapisywane w tabeli, po czym możliwe jest utworzenie rekordu dla konkretnego auta. Proces ilustruje relacje między poszczególnymi tabelami oraz sposób, w jaki dane przepływają w systemie.



Rysunek 1: Diagram przepływu danych przedstawiający proces tworzenia nowego auta oraz dodawania klasy, modelu, marki.

2.1.2 Pracownik, stanowisko

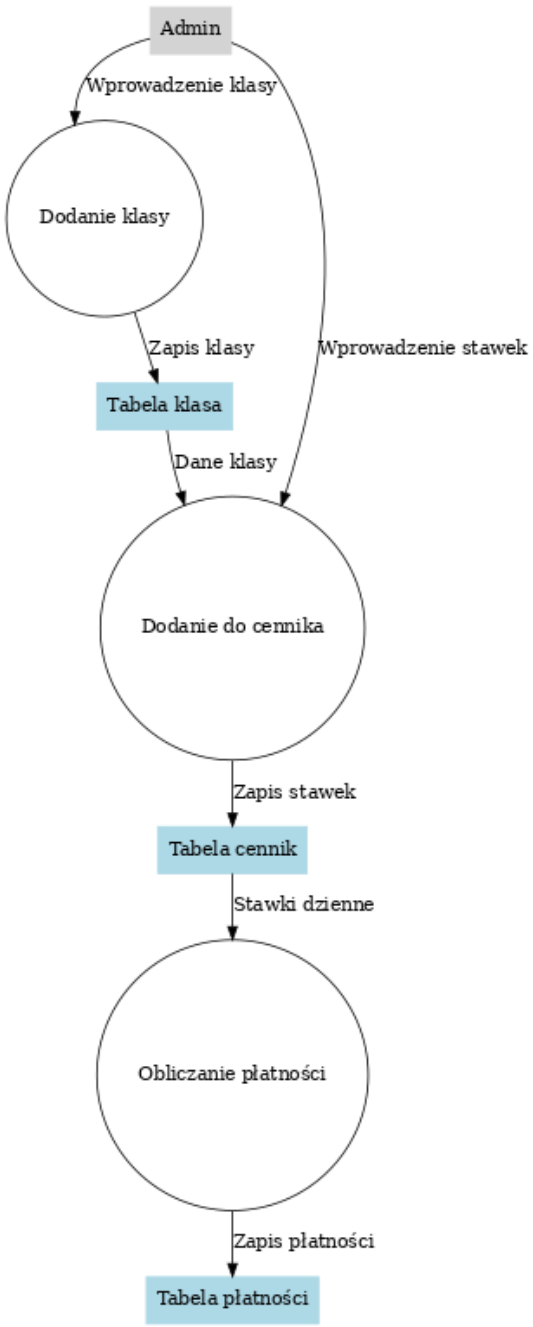
Użytkownik wprowadza informacje o stanowisku, w tym szczegóły takie jak nazwa stanowiska, wysokość wynagrodzenia oraz uprawnienia do realizacji wynajmów. Dane te są zapisywane w tabeli stanowiska. Następnie użytkownik tworzy nowy rekord pracownika, przypisując mu odpowiednie stanowisko oraz pozostałe dane niezbędne do identyfikacji pracownika.



Rysunek 2: Diagram przepływu danych przedstawiający proces dodawania stanowiska i przypisania pracownika.

2.1.3 Klasa, Cennik, Płatność

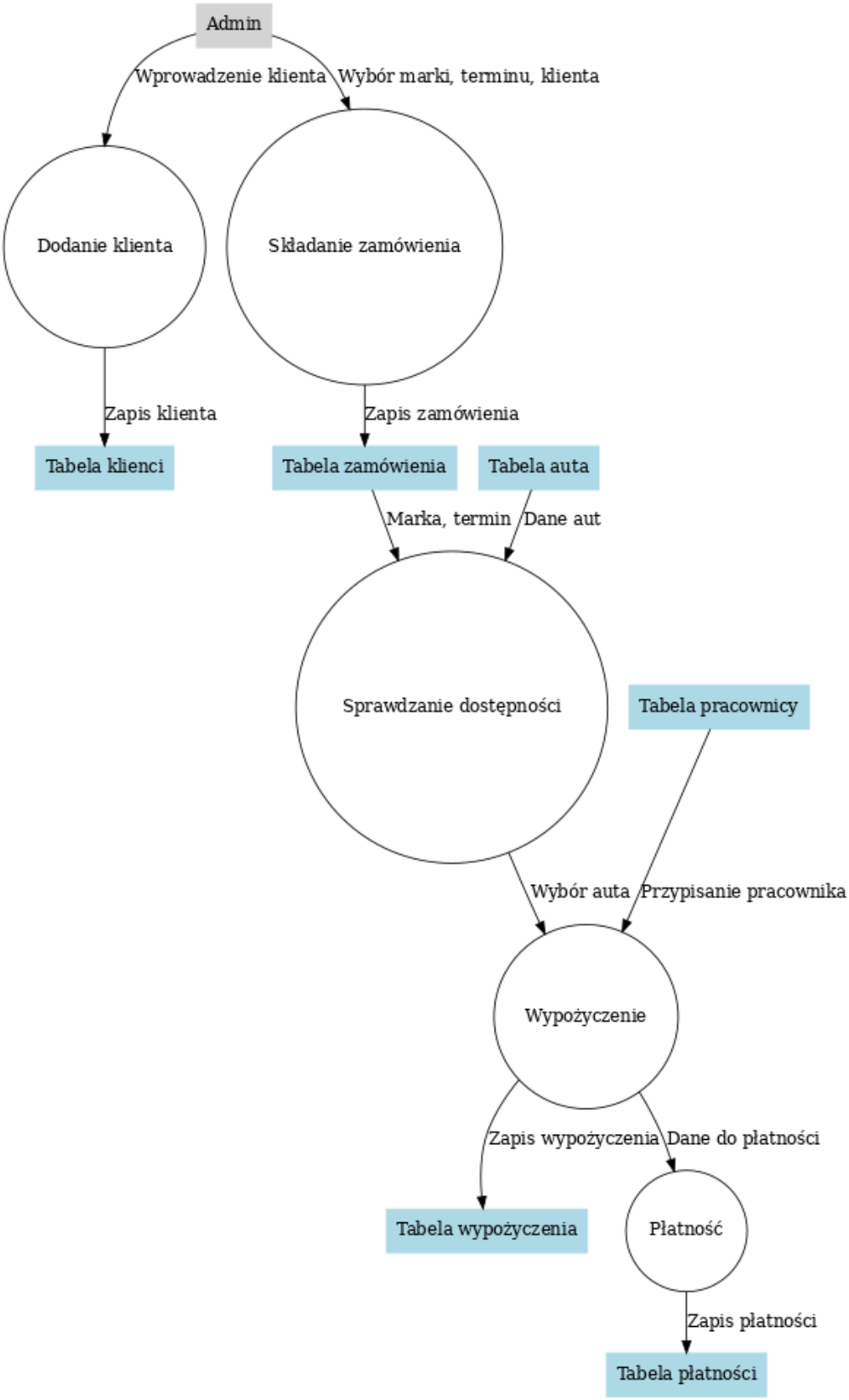
Proces rozpoczyna się od wprowadzenia przez administratora szczegółów dotyczących klasy pojazdu (nazwa, opis), które są zapisywane w tabeli klasa. Następnie na podstawie klasy pojazdu użytkownik określa stawkę dzienną, która jest zapisywana w tabeli cennik. Podczas realizacji wynajmu system korzysta z tych danych, aby obliczyć koszt wynajmu na podstawie liczby dni oraz stawki dziennej, a wynik obliczeń jest zapisywany w tabeli płatności.



Rysunek 3: Diagram przepływu danych przedstawiający proces dodawania klasy pojazdu, wprowadzania danych do cennika oraz obliczania płatności.

2.1.4 Klient, zamówienie, wypożyczenie, płatność

Diagram przedstawia proces obsługi zamówienia w systemie wypożyczalni. Administrator rejestruje klienta, składa zamówienie (wybierając pojazd i termin), a system sprawdza dostępność auta. Jeśli pojazd jest dostępny, zamówienie przekształca się w wypożyczenie, gdzie przypisany jest konkretny pojazd i pracownik. Na końcu obliczana jest płatność, która uwzględnia liczbę dni i klasę pojazdu. Diagram ilustruje przepływ danych między procesami a bazą danych.

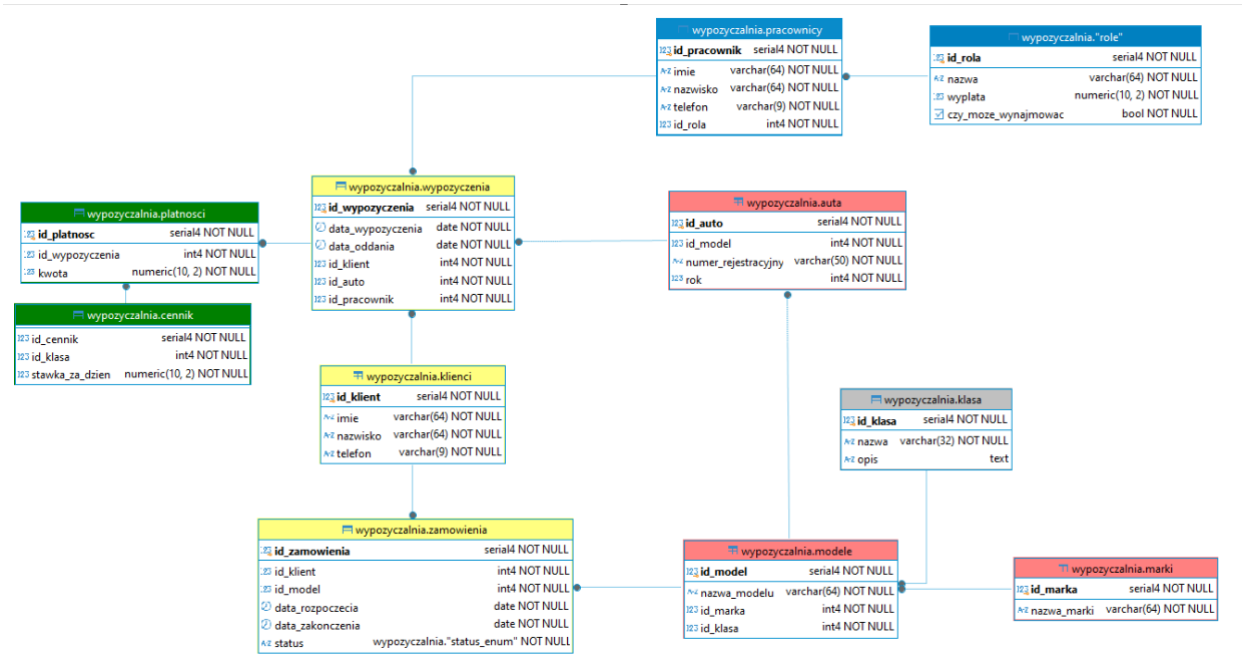


Rysunek 4: Diagram przepływu danych przedstawiający proces składania zamówienia, sprawdzania dostępności pojazdu oraz realizacji wypożyczenia w systemie wypożyczalni samochodowej.

2.2 Zdefiniowanie encji

- Auta - reprezentują pojazdy dostępne w wypożyczalni. Przechowują informacje o modelu, numerze rejestracyjnym oraz roku produkcji.
- Modele - opisują modele samochodów przypisanych do wybranych marek. Zawierają informacje o nazwie modelu, przynależności do konkretnej marki oraz klasie auta.
- Marki - definiują marki samochodów. Przechowują informacje o nazwach marek.
- Klasa - klasyfikuje pojazdy według ich kategorii (np. A, B, C). Zawiera nazwę klasy oraz jej opis.
- Klienci - przechowują podstawowe dane klientów, takie jak imię, nazwisko i unikalny numer telefonu, umożliwiając jednoznaczną identyfikację klienta.
- Pracownicy - przechowuje informacje o pracownikach wypożyczalni. Zawiera podstawowe dane pracowników takie jak imię, nazwisko, telefon oraz przypisane stanowisko.
- Role - definiują role przypisane pracownikom. Zawierają nazwę stanowiska, wysokość wynagrodzenia oraz informację, czy dana rola posiada uprawnienia do realizacji wynajmu pojazdów.
- Zamówienia - reprezentują zamówienia składane przez klientów. Przechowują dane o kliencie, modelu auta, terminach rozpoczęcia i zakończenia wynajmu oraz statusie zamówienia. Dodatkowo rejestrują historię transakcji klienta.
- Wypożyczenia - są powiązane z tabelą zamówień. Przechowują szczegółowe informacje o realizowanych wypożyczeniach, w tym dane klienta, terminy wynajmu, konkretny pojazd oraz pracownika obsługującego wypożyczenie.
- Płatności - rejestrują przepływy finansowe związane z wynajmem pojazdów. Zawierają informacje o kwocie płatności oraz powiązanych wypożyczeniach.
- Cennik - definiuje dzienne stawki wynajmu pojazdów w zależności od ich klasy.

2.3 Zaprojektowanie relacji między encjami (Diagram ERD)



Rysunek 5: Diagram ERD

3 Projekt logiczny

3.1 Projektowanie tabel, kluczy, indeksów:

3.1.1 Marki aut

```
CREATE TABLE wypozyczalnia.marki (
    id_marka        SERIAL          PRIMARY KEY,
    nazwa_marki     VARCHAR(64)     NOT NULL UNIQUE
);
```

Rysunek 6: Tabela marki

Tabela `marki` przechowuje informacje o markach samochodów dostępnych w wypożyczalni.

- `id_marka` – klucz główny (*primary key*), typ `serial`, automatyczne numerowanie.
- `nazwa_marki` – nazwa marki, typ `varchar(64)`, wartość `not null`, unikalna (*unique*).

3.1.2 Modele aut

```
CREATE TABLE wypozyczalnia.modele(
id_model                SERIAL          PRIMARY KEY,
nazwa_modelu            VARCHAR(62)     NOT NULL,
id_marka                INT             NOT NULL,
id_klasa                INT             NOT NULL,

CONSTRAINT fk_model_id_klasa FOREIGN KEY (id_klasa) REFERENCES wypozyczalnia.klasa(id_klasa),
CONSTRAINT fk_model_marka FOREIGN KEY (id_marka) REFERENCES wypozyczalnia.marki (id_marka);
);
```

Rysunek 7: Tabela modele

Tabela `modele` przechowuje informacje o modelach samochodów dostępnych w wypożyczalni.

- `id_model` – klucz główny (*primary key*), typ `SERIAL`, umożliwiający automatyczne numerowanie wartości.
- `nazwa_modelu` – nazwa modelu samochodu, typ `VARCHAR(64)`, wartość `NOT NULL`.
- `id_marka` – identyfikator marki samochodu, typ `INT`, wartość `NOT NULL`, klucz obcy (*foreign key*), odwołujący się do pola `id_marka` w tabeli `marki`.
- `id_klasa` – identyfikator klasy samochodu, typ `INT`, wartość `NOT NULL`, klucz obcy (*foreign key*), odwołujący się do pola `id_klasa` w tabeli `klasy`.

3.1.3 Klasy aut

```
CREATE TABLE wypozyczalnia.klasa (
    id_klasa    SERIAL          PRIMARY KEY,
    nazwa       VARCHAR(32)     NOT NULL UNIQUE,
    opis        TEXT
);
```

Rysunek 8: Tabela klasa

Tabela `klasa` przechowuje informacje o klasach samochodów dostępnych w wypożyczalni.

- `id_klasa` – klucz główny (*primary key*), typ `SERIAL`, umożliwiający automatyczne numerowanie wartości.

- `nazwa` – nazwa klasy samochodu, typ `VARCHAR(32)`, wartość `NOT NULL`, unikalna (*UNIQUE*).
- `opis` – pole tekstowe opisujące klasę, typ `TEXT`, wartość opcjonalna.

3.1.4 Auta

```sql
CREATE TABLE wypozyczalnia.auta (
    id_auto                 SERIAL          PRIMARY KEY,
    id_model                INT             NOT NULL,
    numer_rejestracyjny     VARCHAR(50)     NOT NULL UNIQUE,
    rok                     INT             NOT NULL,

    CONSTRAINT fk_auto_model FOREIGN KEY (id_model) REFERENCES wypozyczalnia.modele (id_model)
);
```

Rysunek 9: Tabela auta

Tabela `auta` przechowuje informacje o samochodach dostępnych w wypożyczalni.

- `id_auto` – klucz główny (*primary key*), typ `SERIAL`, umożliwiający automatyczne numerowanie wartości.
- `id_model` – identyfikator modelu samochodu, typ `INT`, wartość `NOT NULL`, klucz obcy (*foreign key*), odwołujący się do pola `id_model` w tabeli `modele`.
- `numer_rejestracyjny` – numer rejestracyjny samochodu, typ `VARCHAR(50)`, wartość `NOT NULL`, unikalna (*UNIQUE*).
- `rok` – rok produkcji samochodu, typ `INT`, wartość `NOT NULL`.

3.1.5 Klienci

```sql
CREATE TABLE wypozyczalnia.klienci (
    id_klient       SERIAL          PRIMARY KEY,
    imie            VARCHAR(64)     NOT NULL,
    nazwisko        VARCHAR(64)     NOT NULL,
    telefon         VARCHAR(9)      NOT NULL UNIQUE
);
```

Rysunek 10: Tabela klienci

Tabela `klienci` przechowuje informacje o klientach wypożyczalni.

- `id_klient` – klucz główny (*primary key*), typ `SERIAL`, automatyczne numerowanie.
- `imie` – imię klienta, typ `VARCHAR(64)`, wartość `NOT NULL`.
- `nazwisko` – nazwisko klienta, typ `VARCHAR(64)`, wartość `NOT NULL`.
- `telefon` – numer telefonu klienta, typ `VARCHAR(9)`, wartość `NOT NULL`, unikalna (*UNIQUE*).

3.1.6 Role

```sql
CREATE TABLE wypozyczalnia.role (
    id_rola                 SERIAL              PRIMARY KEY,
    nazwa                   VARCHAR(64)         NOT NULL UNIQUE,
    wyplata                 NUMERIC(10,2)       NOT NULL CHECK (wyplata > 0),
    czy_moze_wynajmowac     BOOLEAN             NOT NULL DEFAULT FALSE
);
```

Rysunek 11: Tabela role

Tabela `role` przechowuje informacje o rolach przypisanych pracownikom wypożyczalni.

- `id_rola` – klucz główny (*primary key*), typ `SERIAL`, automatyczne numerowanie.
- `nazwa` – nazwa roli, typ `VARCHAR(64)`, wartość `NOT NULL`, unikalna (*UNIQUE*).
- `wyplata` – wynagrodzenie dla roli, typ `NUMERIC(10,2)`, wartość `NOT NULL`, warunek sprawdzający (`CHECK`) wymaga, aby `wyplata > 0`.
- `czy_moze_wynajmowac` – pole logiczne wskazujące, czy rola ma możliwość wynajmowania samochodów, typ `BOOLEAN`, wartość `NOT NULL`, domyślnie ustawiona na `FALSE`.

3.1.7 Pracownicy

```sql
CREATE TABLE wypozyczalnia.pracownicy (
    id_pracownik        SERIAL          PRIMARY KEY,
    imie                VARCHAR(64)     NOT NULL,
    nazwisko            VARCHAR(64)     NOT NULL,
    telefon             VARCHAR(9)      NOT NULL UNIQUE,
    id_rola             INT             NOT NULL,

    CONSTRAINT fk_pracownicy_rola FOREIGN KEY (id_rola) REFERENCES wypozyczalnia.role (id_rola)
);
```

Rysunek 12: Tabela pracownicy

Tabela `pracownicy` przechowuje informacje o pracownikach wypożyczalni.

- `id_pracownik` – klucz główny (*primary key*), typ `SERIAL`, automatyczne numerowanie.
- `imie` – imię pracownika, typ `VARCHAR(64)`, wartość `NOT NULL`.
- `nazwisko` – nazwisko pracownika, typ `VARCHAR(64)`, wartość `NOT NULL`.
- `telefon` – numer telefonu pracownika, typ `VARCHAR(9)`, wartość `NOT NULL`, unikalna (*UNIQUE*).
- `id_rola` – identyfikator roli, typ `INT`, wartość `NOT NULL`, klucz obcy (*foreign key*) odwołujący się do tabeli `role` (`id_rola`).

3.1.8 Wypożyczenia

```sql
CREATE TABLE wypozyczalnia.wypozyczenia (
    id_wypozyczenia     SERIAL  PRIMARY KEY,
    data_wypozyczenia   DATE    NOT NULL,
    data_oddania        DATE    NOT NULL CHECK (data_oddania > data_wypozyczenia),
    id_klient           INT     NOT NULL,
    id_auto             INT     NOT NULL,
    id_pracownik        INT     NOT NULL,

    CONSTRAINT fk_wypozyczenia_id_klient FOREIGN KEY (id_klient) REFERENCES wypozyczalnia.klienci (id_klient),
    CONSTRAINT fk_wypozyczenia_id_auto FOREIGN KEY (id_auto) REFERENCES wypozyczalnia.auta (id_auto),
    CONSTRAINT fk_wypozyczenia_id_pracownik FOREIGN KEY (id_pracownik) REFERENCES wypozyczalnia.pracownicy (id_pracownik)
);
```

Rysunek 13: Tabela wypożyczenia

- `id_wypozyczenia` – klucz główny (*primary key*), typ `SERIAL`, umożliwiający automatyczne numerowanie wartości.
- `data_wypozyczenia` – data rozpoczęcia wypożyczenia, typ `DATE`, wartość `NOT NULL`.
- `data_oddania` – data zakończenia wypożyczenia, typ `DATE`, wartość `NOT NULL`. Warunek `CHECK` wymaga, aby `data_oddania > data_wypozyczenia`.
- `id_klient` – identyfikator klienta, typ `INT`, wartość `NOT NULL`, klucz obcy (*foreign key*) odwołujący się do tabeli `klienci`.

- `id_auto` – identyfikator samochodu, typ `INT`, wartość `NOT NULL`, klucz obcy odwołujący się do tabeli `auta`.
- `id_pracownik` – identyfikator pracownika obsługującego wypożyczenie, typ `INT`, wartość `NOT NULL`, klucz obcy odwołujący się do tabeli `pracownicy`.

3.1.9 Typ status

```sql
CREATE TYPE wypozyczalnia.status_enum AS ENUM ('udane', 'nieudane', 'oczekujące');
```

Rysunek 14: Typ enum status

Typ `status_enum` definiuje zestaw możliwych statusów zamówienia.

- Wartości dostępne w typie: `'udane'`, `'nieudane'`, `'oczekujące'`.

3.1.10 Zamówienia

```sql
CREATE TABLE wypozyczalnia.zamowienia (
    id_zamowienia           SERIAL          PRIMARY KEY,
    id_klient               INT             NOT NULL,
    id_model                INT             NOT NULL,
    data_rozpoczecia        DATE            NOT NULL,
    data_zakonczenia        DATE            NOT NULL CHECK (data_zakonczenia > data_rozpoczecia),
    status wypozyczalnia.status_enum        NOT NULL,

    CONSTRAINT fk_zamowienia_id_klient FOREIGN KEY (id_klient) REFERENCES wypozyczalnia.klienci (id_klient),
    CONSTRAINT fk_zamowienia_id_model FOREIGN KEY (id_model) REFERENCES wypozyczalnia.modele (id_model)
);
```

Rysunek 15: Tabela zamówienia

Tabela `zamowienia` przechowuje informacje o zamówieniach klientów na samochody.

- `id_zamowienia` – klucz główny (*primary key*), typ `SERIAL`, umożliwiający automatyczne numerowanie wartości.
- `id_klient` – identyfikator klienta, typ `INT`, wartość `NOT NULL`, klucz obcy (*foreign key*) odwołujący się do tabeli `klienci`.
- `id_model` – identyfikator modelu samochodu, typ `INT`, wartość `NOT NULL`, klucz obcy odwołujący się do tabeli `modele`.
- `data_rozpoczecia` – data rozpoczęcia zamówienia, typ `DATE`, wartość `NOT NULL`.
- `data_zakonczenia` – data zakończenia zamówienia, typ `DATE`, wartość `NOT NULL`. Warunek `CHECK` wymaga, aby `data_zakonczenia > data_rozpoczecia`.
- `status` – status zamówienia, typ `status_enum`, wartość `NOT NULL`.

3.1.11 Płatności

```sql
CREATE TABLE wypozyczalnia.platnosci (
    id_platnosc         INT             PRIMARY KEY,
    id_wypozyczenia     INT             NOT NULL,
    kwota               NUMERIC(10,2)   NOT NULL,

    CONSTRAINT fk_platnosci_id_wypozyczenia FOREIGN KEY (id_wypozyczenia) REFERENCES wypozyczalnia.wypozyczenia (id_wypozyczenia)
);
```

Rysunek 16: Tabela płatności

Tabela `platnosci` przechowuje informacje o płatnościach za wypożyczenia samochodów.

- `id_platnosc` – klucz główny (*primary key*), typ `INT`.
- `id_wypozyczenia` – identyfikator wypożyczenia, typ `INT`, wartość `NOT NULL`, klucz obcy (*foreign key*) odwołujący się do tabeli `wypozyczenia`.
- `kwota` – kwota płatności, typ `NUMERIC(10,2)`, wartość `NOT NULL`.

3.1.12 Cennik

```
CREATE TABLE wypozyczalnia.cennik (
    id_cennik           SERIAL          PRIMARY KEY,
    id_klasa            INT             NOT NULL UNIQUE,
    stawka_za_dzien     NUMERIC(10,2)   NOT NULL CHECK (stawka_za_dzien > 0),

    CONSTRAINT fk_cennik_id_klasa FOREIGN KEY (id_klasa) REFERENCES wypozyczalnia.klasa (id_klasa)
);
```

Rysunek 17: Cennik

- `id_cennik` – klucz główny (*primary key*), typ `SERIAL`.
- `id_klasa` – identyfikator klasy, typ `INT`, wartość `NOT NULL`, klucz obcy (*foreign key*) odwołujący się do tabeli `klasa`.
- `stawka_za_dzien` – okresla stawke dniowa, typ `NUMERIC(10,2)`, wartość `NOT NULL`. Warunek `CHECK` wymaga, aby `stawka_za_dzien > 0`.

3.2 Analiza zależności funkcyjnych i normalizacja tabel

Baza danych została zaprojektowana zgodnie z zasadami trzeciej postaci normalnej (3NF), co zapewnia eliminację redundancji oraz minimalizację anomalii przy operacjach wstawiania, usuwania i aktualizacji danych. Każda tabela w systemie została znormalizowana, tak aby spełniać poniższe wymagania:

- **Zależności funkcyjne** – każda kolumna w tabeli jest funkcjonalnie zależna od klucza głównego, co oznacza, że wartości w kolumnach są jednoznacznie określone przez klucz główny tabeli.
- **Brak częściowych zależności funkcyjnych** – w tabelach z kluczami złożonymi żaden atrybut nie zależy tylko od części klucza, ale od całego klucza głównego.
- **Brak przechodnich zależności funkcyjnych** – atrybuty w tabeli zależą bezpośrednio od klucza głównego, a nie od innych atrybutów niezależnych od klucza głównego.

Unikalność rekordów w każdej tabeli jest gwarantowana przez autogenerowane klucze główne, które są typu `serial`. Klucze te są niezależne od jakiejkolwiek funkcji obliczeniowej oraz niedostępne do edycji przez użytkownika, co dodatkowo zabezpiecza spójność danych.

Dzięki zastosowanej normalizacji każda tabela została zoptymalizowana do postaci 3NF, a w przypadku konieczności dekompozycji schemat relacyjny został odpowiednio rozdzielony w celu zapewnienia zgodności z założeniami normalizacji.

3.3 Operacje na danych

3.3.1 Widoki

- **modele_marki_klasy():** łączy dane z tabel `modele`, `marki` i `klasa`, prezentując informacje o modelach samochodów, ich markach oraz przypisanych klasach. Umożliwia łatwy dostęp do danych o powiązanych modelach, markach i klasach w jednej strukturze.
- **szczegoly_aut():** prezentuje szczegółowe informacje o samochodach dostępnych w wypożyczalni, łącząc dane dotyczące pojazdów, ich modeli, marek oraz przypisanych klas. Widok umożliwia szybki dostęp do kompleksowych danych o samochodach w jednym zestawieniu.

- **szczegoly_wypozyczenia():** widok przedstawia szczegółowe informacje o wypożyczeniach, łącząc dane dotyczące klientów, samochodów oraz pracowników obsługujących wypożyczenia. Zawiera m.in. dane o kliencie, samochodzie (marka, model, numer rejestracyjny), daty wypożyczenia i oddania oraz pracowniku odpowiedzialnym za transakcję. Widok umożliwia łatwy dostęp do wszystkich danych związanych z wypożyczeniem w jednym zestawieniu.
- **koszty_wypozyczenia():** widok oblicza całkowity koszt wypożyczenia dla każdego klienta, łącząc dane z widoku `szczegoly_wypozyczenia` oraz `cennik`. Uwzględnia czas trwania wypożyczenia (różnica między datą oddania a datą wypożyczenia) oraz stawkę za dzień odpowiadającą klasie samochodu. Widok pozwala w prosty sposób uzyskać informacje o kosztach wypożyczeń w oparciu o szczegóły transakcji.
- **raport_finansowy:** Widok generuje podsumowanie finansowe wypożyczalni, obliczając całkowity oraz średni przychód.

3.3.2 Funkcje

- **wyszukaj_modele(marka_id int):** Funkcja zwraca tabelę z modelami samochodów powiązanymi z daną marką, na podstawie identyfikatora marki.
- **wyszukaj_marki(model_id int):** Funkcja zwraca tabelę z marką samochodu powiązaną z danym modelem, na podstawie identyfikatora modelu.
- **wybierz_pracownika():** Funkcja zwraca identyfikator pracownika, który posiada uprawnienia do obsługi wypożyczeń i nie jest przypisany do stanowiska o identyfikatorze `id_rola = 1` (menedżer). Pracownik jest wybierany na podstawie najmniejszej liczby obsłużonych wypożyczeń, co pozwala na równomierne rozdzielenie obciążenia. W przypadku braku dostępnych pracowników funkcja zwraca wyjątek.
- **sprawdz_numer_rejestracyjny():** Funkcja weryfikuje poprawność numeru rejestracyjnego samochodu zgodnie z ustalonym wzorcem.
- **sprawdz_numer_telefonu():** Funkcja sprawdza, czy numer telefonu wstawiany lub aktualizowany w tabeli spełnia wymagany wzorzec – dokładnie 9 cyfr.
- **zloz_zamowienie():** Funkcja obsługuje proces składania zamówienia w tabeli `zamowienia`. Po złożeniu zamówienia przez klienta funkcja sprawdza, czy dostępny jest samochód o wymaganym modelu, który nie jest już zarezerwowany w danym okresie. Jeśli samochód jest dostępny:
  - Przypisuje identyfikator (`id_auto`) dostępnego samochodu oraz identyfikator pracownika (`id_pracownik`) wybranego przez funkcję `wybierz_pracownika()`.
  - Wstawia nowe wypożyczenie do tabeli `wypozyczenia`.
  - Ustawia status zamówienia na `'udane'`.

  W przypadku braku dostępnych samochodów status zamówienia zostaje ustawiony na `'nieudane'`.
- **dostepne_auta_w_danym_terminie(data_rozpoczecia date, data_zakonczenia date):** Funkcja zwraca tabelę z samochodami dostępnymi w określonym przedziale czasowym.
- **policz_auta_dostepne_w_danym_terminie(data_rozpoczecia date, data_zakonczenia date):** Funkcja zwraca liczbę samochodów dostępnych w określonym przedziale czasowym. Wynik jest zwracany jako liczba całkowita.
- **policz_marki_auta_dostepne_w_danym_terminie(data_rozpoczecia date, data_zakonczenia date):** Funkcja zwraca tabelę z markami i modelami samochodów oraz liczbą dostępnych egzemplarzy każdego modelu w określonym przedziale czasowym.
- **dodaj_platnosc()**: Funkcja automatycznie oblicza kwotę należną za wypożyczenie samochodu na podstawie stawki dniowej i czasu trwania wypożyczenia. Obliczona kwota zostaje automatycznie wstawiona do tabeli `platnosci` razem z identyfikatorem wypożyczenia.

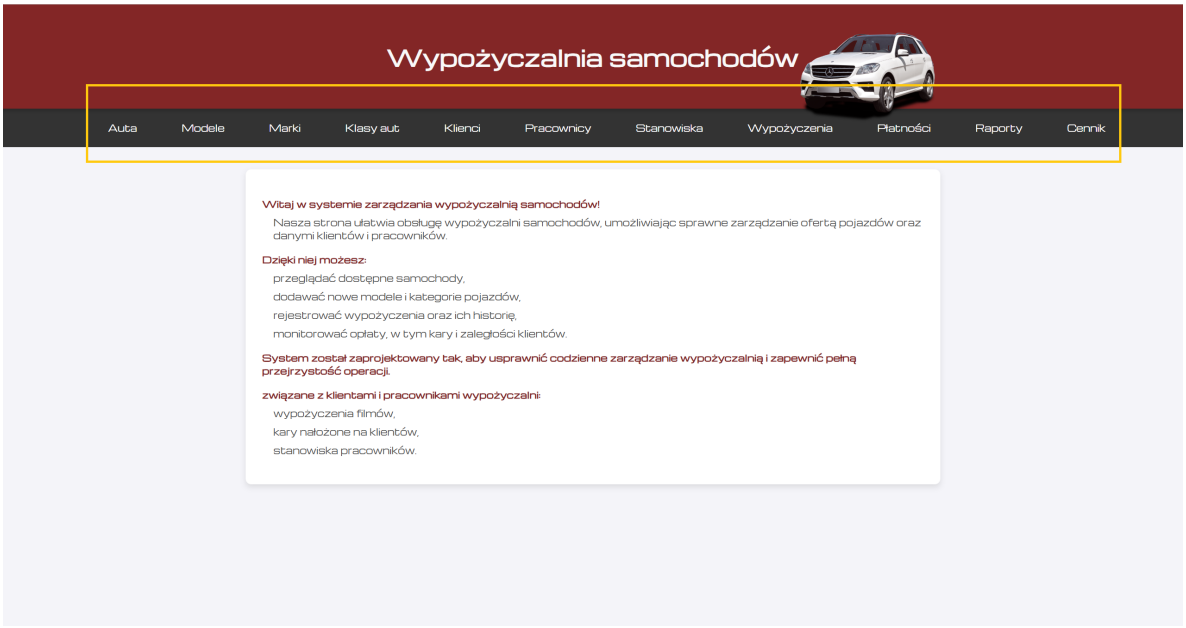
- **najpopularniejsze_modele(min_wypozyczenia int):** Funkcja zwraca listę modeli samochodów, które były wypożyczane więcej niż określoną liczbę razy (`min_wypozyczenia`).
- **przychody_na_klasy_aut():** Funkcja generuje raport przychodów dla poszczególnych klas aut w tym liczbę wypożyczeń oraz całkowity przychód wygenerowany przez auta danej klasy.

3.3.3 Wyzwawalacze

- **trigger_sprawdz_numer_rejestracyjny:** Trigger wywołuje funkcję `sprawdz_numer_rejestracyjny()` przed każdą operacją `INSERT` lub `UPDATE` na tabeli `auta`. Jego celem jest zapewnienie, że numer rejestracyjny samochodu spełnia wymagania formalne. W przypadku błędnego formatu operacja jest przerywana.
- **trigger_sprawdz_numer_telefonu_klient:** Trigger wywołuje funkcję `sprawdz_numer_telefonu()` przed każdą operacją `INSERT` lub `UPDATE` na tabeli `klienci`. Zapewnia, że numer telefonu klienta składa się dokładnie z 9 cyfr. W przypadku błędnego formatu operacja jest przerywana.
- **trigger_sprawdz_numer_telefonu_pracownik:** Trigger wywołuje funkcję `sprawdz_numer_telefonu()` przed każdą operacją `INSERT` lub `UPDATE` na tabeli `pracownicy`. Zapewnia, że numer telefonu pracownika spełnia wymagany format – dokładnie 9 cyfr. W przypadku niezgodności operacja jest przerywana.
- **zloz_zamowienie():** Funkcja obsługuje proces składania zamówienia w tabeli `zamowienia`. Po złożeniu zamówienia przez klienta funkcja sprawdza, czy dostępny jest samochód o wymaganym modelu, który nie jest już zarezerwowany w danym okresie. Jeśli samochód jest dostępny:
- **trigger_zloz_zamowienie:** Trigger wywołuje funkcję `zloz_zamowienie()` przed każdą operacją `INSERT` na tabeli `zamowienia`. Automatycznie przetwarza zamówienie, sprawdzając dostępność samochodów oraz rezerwując je, jeśli spełnione są wymagane warunki.
- **trigger_dodaj_platnosc:** Trigger wywołuje funkcję `dodaj_platnosc()` po każdej operacji `INSERT` na tabeli `wypozyczenia`.
- **trigger_usun_powiazane_platnosci**: Wyzwalacz uruchamiany **po usunięciu rekordu** z tabeli `wypozyczenia`. Jego działanie polega na automatycznym usunięciu wszystkich powiązanych płatności z tabeli `platnosci`, które odnoszą się do usuniętego wypożyczenia. Mechanizm ten zapewnia utrzymanie **integralności danych** w bazie.

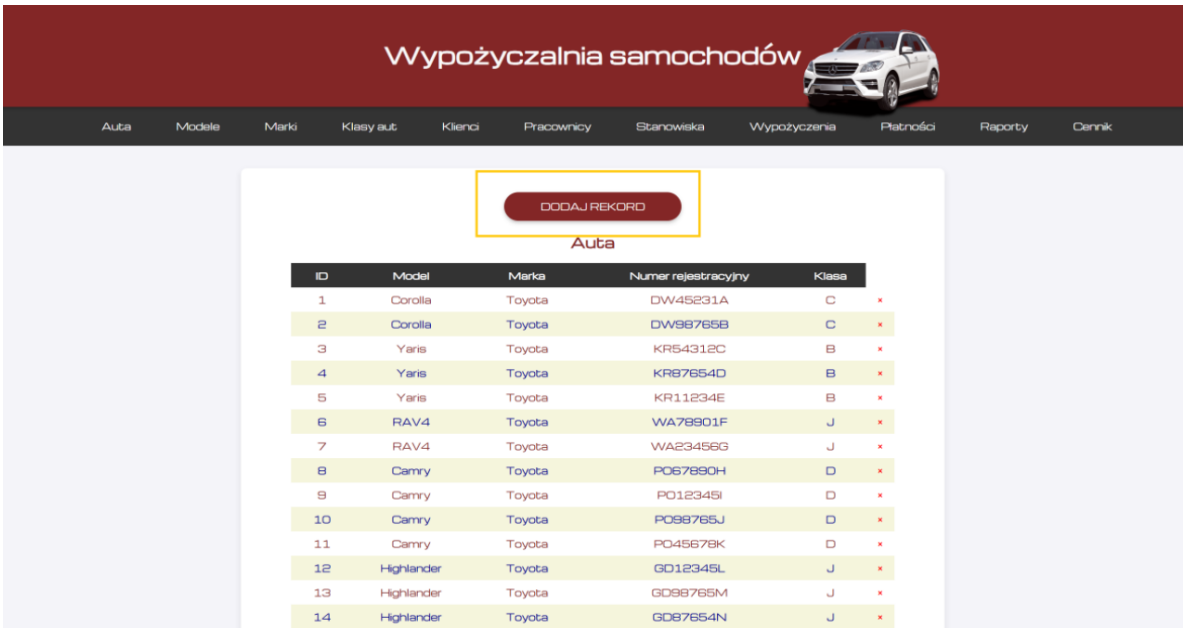
4 Projekt funkcjonalny

Użytkownik ma dostęp do wprowadzania danych do poszczególnych tabel poprzez zakładki wyświetlone na liście u góry interfejsu graficznego:



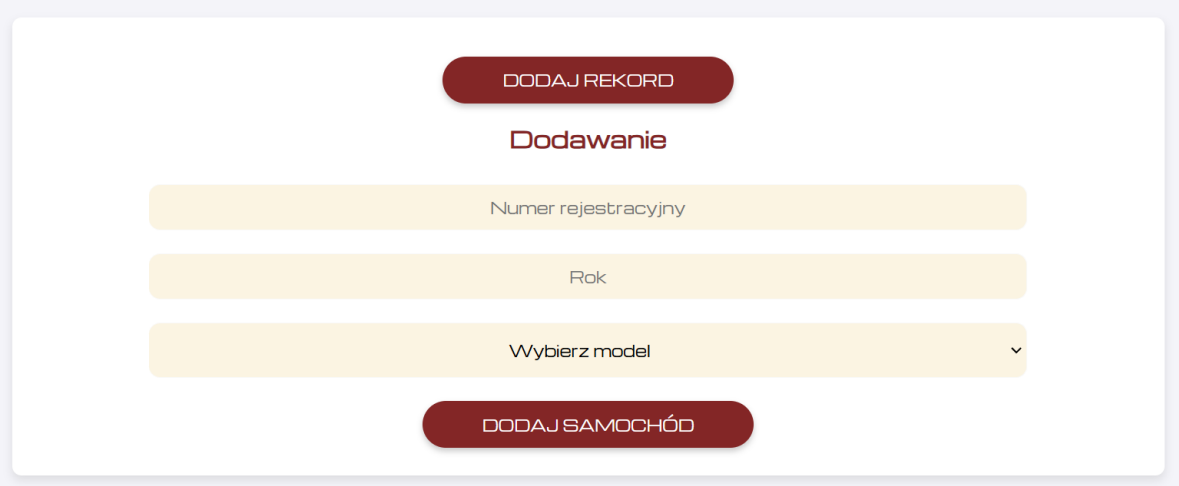
Rysunek 18: Interfejs graiczny użytkownika

Domyślnie, po wejściu do wybranej zakładki, automatycznie wyświetlane są wszystkie istniejące rekordy. Użytkownik ma dostęp do przycisku **Dodaj**, który umożliwia dodanie nowego rekordu. Ponadto obok każdego rekordu znajdują się ikony umożliwiające jego usunięcie – kliknięcie w ikonę pozwala użytkownikowi na usunięcie wybranego rekordu.



Rysunek 19: Dodawanie i wyświetlanie danych

Po kliknięciu przycisku Dodawanie, zostajemy przekierowani do strony, która wyświetla formularz pozwalający na dodanie nowego rekordu.



Rysunek 20: Formularz dodawania rekordu dla tabeli auta

5 Dokumentacja

5.1 Wprowadzanie danych

Dane wprowadzane są do bazy z poziomu interfejsu użytkownika poprzez formularze w odpowiednich zakładkach. Baza dostępna do testowania i oceny została przeze mnie wypełniona przykładowymi danymi.

5.2 Dokumentacja użytkownika

Interfejs użytkownika został zaprojektowany z myślą o maksymalnej intuicyjności i prostocie obsługi. W górnej części okna stale dostępne są zakładki, które umożliwiają przeglądanie i dodawanie danych. Niektóre zakładki, takie jak **Wypożyczenia** lub **Raporty** posiadają rozwijane menu, które daje dostęp do dodatkowych opcji, takich jak przeglądanie zamówień (udanych i nieudanych) oraz wypożyczeń lub wyświetlanie różnych raportów.

Dokładny opis klas i metod znajduje się w skompresowanym folderze html, w pliku index.html.

5.3 Opracowanie dokumentacji technicznej

Aplikacja internetowa została napisana w języku Python korzystając z jego biblioteki Flask oraz dependencji tej biblioteki wypisanych w załączonym pliku "requirements.txt". Dodatkowo skorzystałam z systemu zarządzania pakietami i środowiskiem Conda. Baza danych zawarta w pliku "baza.sql" jest przechowywana w systemie tembo. Użytkownik aby uruchomić lokalnie projekt musiałby mieć dostęp do bibliotek z wyżej wymienionego pliku z poziomu katalogu zawierającego wszystkie pliki projektu, a następnie wywołać w terminalu komendę "flask run", która uruchamia lokalny serwer na którym działa aplikacja. Aby ułatwić przeglądanie i testowanie działania projektu, jest on również dostępny dzięki platformie Render pod linkiem: kliknij. Całość funkcjonalności projektu zawarta jest w pliku "app.py". Szablony do witryn html w katalogu "/templates" zostały wygenerowane przez bibliotekę Jinja, natomiast obsługą bazy danych zajmuje się biblioteka Psycopg2. Dostęp do aplikacji poprzez Render zapewnia Gunicorn.

5.4 Wykaz literatury

- https://realpython.com/flask-by-example-part-1-project-setup/
- https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3
- https://www.hyperone.com/services/storage/database/tutorials/python-psycopg2.html