Temat: Wypożyczalnia Samochodów Osobowych

Autorzy: Paweł Dziwisz, Tomasz Dębosz, Piotr Kotarba, Ihor-Severyn Dolhopolov

1. Opis systemu

Celem projektu jest stworzenie systemu bazy danych, który będzie obsługiwał wypożyczalnie aut osobowych. W dzisiejszych czasach wiele osób, stawia na rozwiązania tymczasowe, dlatego stawiają na wypożyczenie auta na jakiś okres czasu i niemartwienie się o naprawy. Jest to dobra alternatywa, zakupu auta na własność. System będzie umożliwiał wypożyczenie samochodu na ustalony czas i automatycznie liczył koszt wypożyczenia na określony okres czasu. System będzie sprawdzał dostępność samochodów oraz sprawdzał okres końca najmu dla osób, które już wypożyczyły samochód. System będzie kontrolował, czy auta są gotowe do wypożyczenia (sprawy techniczne, wizualne).

Samochody podzielone zostały na klasy względem przygotowania do jazdy miejskiej lub jazdy terenowej. Każda klasa będzie nazwana literami związana tak samo jak nazywa ją producent. W przypadku wyboru przez klienta auta będzie on musiał pierw wybrać klasę pojazdu, czyli kategorie oraz wartość auta. W przypadku, gdy klient wybierze sobie odpowiednią klasę, a nie będzie dostępnego auta, które chce to będzie ono zastąpione innym autem z tej samej klasy.

Klient będzie miał pełna kontrole nad wynajmem auta, czyli będzie w każdej chwili mógł kontrolować ilość pozostałego czasu oraz kwestię związaną z finansami. Wszystkie dane które będą podawane przez klienta będą w pełni zabezpieczone i poufne dla osób, które nie są upoważnione do ich wglądu.

W przypadku awarii kontakt z wypożyczalnia będzie bardzo prosty, a w razie wypadku wypożyczalnia będzie zajmować się sprawami takimi jak kontakt z ubezpieczycielem i wezwanie lawety. Natomiast osoba wypożyczająca będzie zmuszona tylko do opisu wypadku/kolizji.

2. Historyjki użytkownika

Jako pracownik chcę mieć wgląd, które samochody zostały wypożyczone, aby móc swobodnie zarządzać autami do wypożyczenia.

Jako właściciel chcę, aby każdy samochód z osobna posiadał unikalny identyfikator.

Jako właściciel chcę, aby każdy klient miał swoje konto z informacjami do identyfikacji.

Jako właściciel chcę, aby system posiadał pole, które informuje o statusie rezerwacji.

Jako właściciel chcę mieć dostęp do danych finansowych z podziałem na okres czasowy.

Jako właściciel chcę, aby system zapewniał ustalanie różnych cen za wynajem auta, w zależności od jego klasy czy marki.

Jako właściciel chcę mieć wgląd w dostępne samochody z podziałem na kategorie.

Jako właściciel wypożyczalni chcę, aby nie było możliwości wypożyczenia samochodu, który nie jest dostępny w salonie.

Jako właściciel chciałbym mieć dostęp do całej oferty oraz wgląd w stan samochodów.

Jako właściciel chcę, aby system przechowywał takie dane kontaktowe klientów jak numery telefonu, w celu skontaktowania się z nimi.

Jako właściciel chcę, aby w bazie danych istniało pole informujące o faktycznym dniu zwrotu wypożyczonego samochodu.

Jako właściciel chcę, aby w przypadku późniejszego oddania samochodu naliczana została kara od każdego spóźnienia.

Jako właściciel chcę, aby wszystkie pola w tabelach zawierających istotne informacje musiały być wypełnione danymi, aby uniknąć niekompletnych danych. (Brak możliwości wstawienia wartości NULL w kluczowych miejscach)

Jako klient wypożyczalni chcę móc przeglądać ofertę samochodów, aby swobodnie wybrać auto do wypożyczenia.

Jako klient chcę, aby system pokazywał liczbę osób, jaka może podróżować danym samochodem.

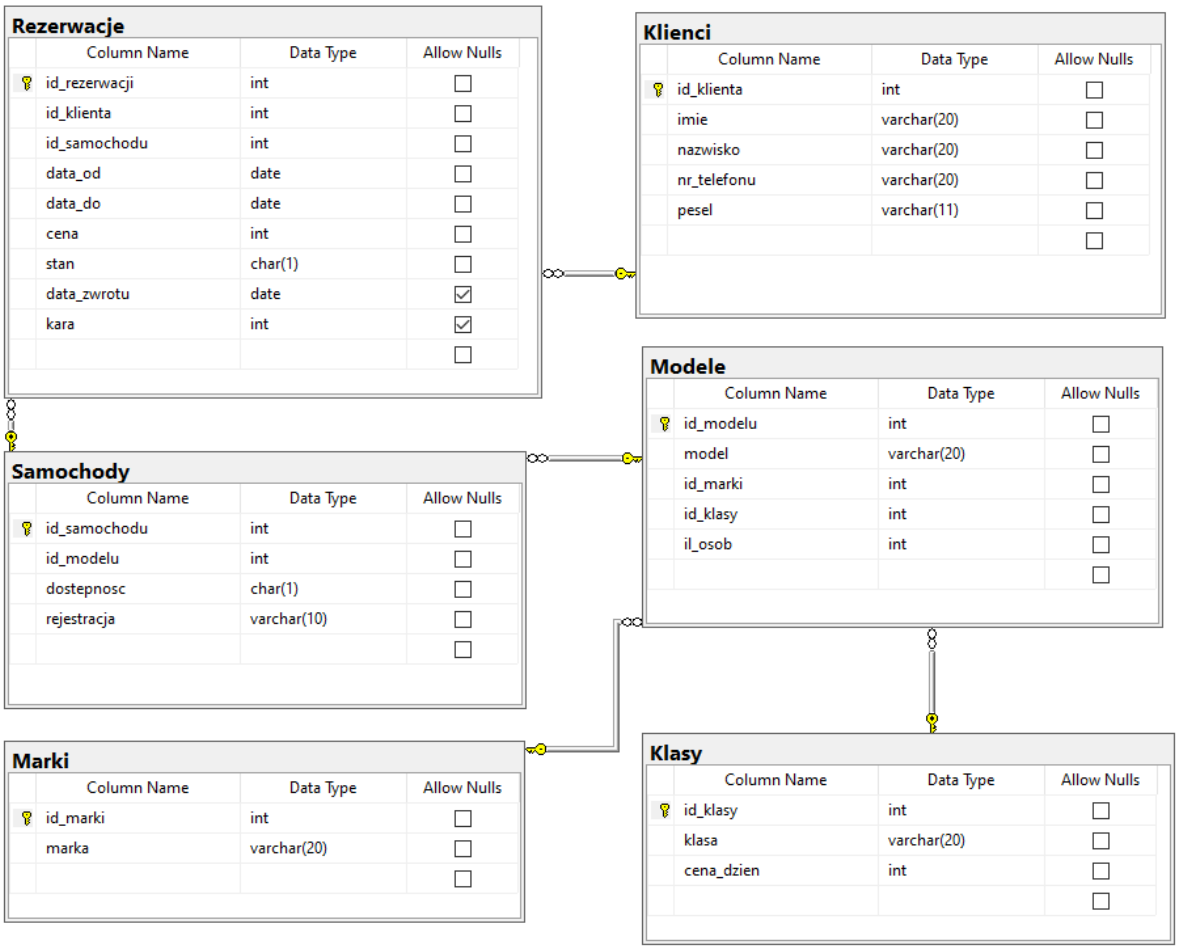
Jako klient chcę móc wyszukiwać samochody według parametrów, które mnie interesują.

Jako klient chciałbym mieć wgląd do informacji o dostępnych samochodów.

Jako klient chcę mieć wgląd w cennik, który posiada ustalone ceny w zależności od modelu i typu samochodu.

3. Projekt bazy danych

Schemat bazy danych



Opis poszczególnych tabel

Nazwa tabeli: Klienci Opis: Tabela zawierająca informacje o klientach korzystających z wypożyczalni oraz ich dane kontaktowe		
Nazwa atrybutu	Typ	Opis/Uwagi
id_klienta	int	Klucz główny. Pole typu autonumerycznego
imie	Varchar(20)	Dane kontaktowe: imię klienta
nazwisko	Varchar(20)	Dane kontaktowe: nazwisko klienta
nr_telefonu	Varchar(20)	Dane kontaktowe: numer telefonu klienta
pesel	Varchar(11)	Dane kontaktowe: pesel

Nazwa tabeli: Rezerwacje Opis: Tabela zawierająca informacje o wypożyczeniach samochodów oraz ID osób, które je wypożyczyły		
Nazwa atrybutu	Typ	Opis/Uwagi
id_rezerwacji	int	PK, Pole typu autonumerycznego
id_klienta	int	Klucz obcy pola id_klienta z tabeli Klienci
id_samochodu	int	Klucz obcy pola id_samochodu z tabeli Samochody
data_od	date	Data rozpoczęcia wynajmu danego samochodu
data_do	date	Termin planowanego oddania samochodu
cena	int	Kwota za okres calego wypozyczenia
stan	Char(1)	Pole informujące, Z-czy wynajęcie samochodu dobiegło końca (stan końcowy, informacja) / W-jest w trakcie wypożyczenia / R-rezerwacja terminu
data zwrotu	date	Faktyczna data zwrotu samochodu
kara	int	Pole informujące o naliczonej ewentualnie karze za nieterminowy zwrot samochodu

Nazwa tabeli: Samochody Opis: Tabela zawierająca informacje o konkretnych samochodach (fizycznych egzemplarzach)		
Nazwa atrybutu	Typ	Opis/Uwagi
Id_samochodu	int	PK, Pole typu autonumerycznego
Id_modelu	int	Klucz obcy pola id_modelu z tabeli Modele
Dostępność	Char(1)	Pole informujące, czy dany samochód jest dostępny w tym momencie (T – tak, N – nie)
Rejestracja	Varchar(10)	Numer tablicy rejestracyjnej pojazdu

Nazwa tabeli: Klasy Opis: Tabela zawierająca informacje o klasach samochodów oraz cenę za 1 dzień wypożyczenia		
Nazwa atrybutu	Typ	Opis/Uwagi
Id_klasy	int	PK, pole typu autonumerycznego
klasa	Varchar(20)	Nazwa klasy samochodu
cena_dzien	int	Cena wynajmu za każdy dzień

Nazwa tabeli: Modele Opis: Tabela zawierająca informacje o modelach samochodów		
Nazwa atrybutu	Typ	Opis/Uwagi
Id_modelu	int	PK, Pole typu autonumerycznego
model	Varchar(20)	Nazwa modelu samochodu
id_marki	int	Klucz obcy pola id_marki z tabeli Marki
Id_klasy	int	Klucz obcy pola id_klasy z tabeli Klasy
Il_osob	int	Pole informujace o pojemnosci samochodu

Nazwa tabeli: Marki Opis: Tabela zawierająca informacje o markach samochodów		
Nazwa atrybutu	Typ	Opis/Uwagi
Id_marki	int	PK, pole typu autonumerycznego
marka	Varchar(20)	Nazwa marki samochodu

4. Implementacja

Kod poleceń DDL

```
CREATE TABLE [dbo].[Klasy](

[id_klasy] [int] IDENTITY(1,1) NOT NULL,

[klasa] [varchar](20) NOT NULL,

[cena_dzien] [int] NOT NULL,

CONSTRAINT [PK_Klasy] PRIMARY KEY CLUSTERED

(

[id_klasy] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]
```

```
CREATE TABLE [dbo].[Klienci](

[id_klienta] [int] IDENTITY(1,1) NOT NULL,

[imie] [varchar](20) NOT NULL,

[nazwisko] [varchar](20) NOT NULL,

[nr_telefonu] [varchar](20) NOT NULL,

[pesel] [varchar](11) NOT NULL,

CONSTRAINT [PK_Klienci] PRIMARY KEY CLUSTERED

(

[id_klienta] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]


ALTER TABLE [dbo].[Klienci]  WITH CHECK ADD  CONSTRAINT [CK_Klienci] CHECK  (([pesel] like '[0-
9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'))

ALTER TABLE [dbo].[Klienci] CHECK CONSTRAINT [CK_Klienci]



CREATE TABLE [dbo].[Marki](

[id_marki] [int] IDENTITY(1,1) NOT NULL,

[marka] [varchar](20) NOT NULL,

CONSTRAINT [PK_Marki] PRIMARY KEY CLUSTERED

(

[id_marki] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]
```

```
CREATE TABLE [dbo].[Modele](

[id_modelu] [int] IDENTITY(1,1) NOT NULL,

[model] [varchar](20) NOT NULL,

[id_marki] [int] NOT NULL,

[id_klasy] [int] NOT NULL,

[il_osob] [int] NOT NULL,

CONSTRAINT [PK_Modele] PRIMARY KEY CLUSTERED

(

[id_modelu] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

ALTER TABLE [dbo].[Modele]  WITH CHECK ADD  CONSTRAINT [FK_Modele_Klasy] FOREIGN
KEY([id_klasy])

REFERENCES [dbo].[Klasy] ([id_klasy])


ALTER TABLE [dbo].[Modele] CHECK CONSTRAINT [FK_Modele_Klasy]

ALTER TABLE [dbo].[Modele]  WITH CHECK ADD  CONSTRAINT [FK_Modele_Marki] FOREIGN
KEY([id_marki])

REFERENCES [dbo].[Marki] ([id_marki])

ALTER TABLE [dbo].[Modele] CHECK CONSTRAINT [FK_Modele_Marki]



CREATE TABLE [dbo].[Rezerwacje](

[id_rezerwacji] [int] IDENTITY(1,1) NOT NULL,

[id_klienta] [int] NOT NULL,

[id_samochodu] [int] NOT NULL,

[data_od] [date] NOT NULL,

[data_do] [date] NOT NULL,
```

```
[cena] [int] NOT NULL,

[stan] [char](1) NOT NULL,

[data_zwrotu] [date] NULL,

[kara] [int] NULL,

CONSTRAINT [PK_Rezerwacje] PRIMARY KEY CLUSTERED

(

[id_rezerwacji] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

ALTER TABLE [dbo].[Rezerwacje]  WITH CHECK ADD  CONSTRAINT [FK_Rezerwacje_Klienci] FOREIGN
KEY([id_klienta])

REFERENCES [dbo].[Klienci] ([id_klienta])

ALTER TABLE [dbo].[Rezerwacje] CHECK CONSTRAINT [FK_Rezerwacje_Klienci]

ALTER TABLE [dbo].[Rezerwacje]  WITH CHECK ADD  CONSTRAINT [FK_Rezerwacje_Samochody]
FOREIGN KEY([id_samochodu])

REFERENCES [dbo].[Samochody] ([id_samochodu])

ALTER TABLE [dbo].[Rezerwacje] CHECK CONSTRAINT [FK_Rezerwacje_Samochody]

ALTER TABLE [dbo].[Rezerwacje]  WITH CHECK ADD  CONSTRAINT [CK_Rezerwacje] CHECK
(([stan]='Z' OR [stan]='W' OR [stan]='R'))

ALTER TABLE [dbo].[Rezerwacje] CHECK CONSTRAINT [CK_Rezerwacje]



CREATE TABLE [dbo].[Samochody](

[id_samochodu] [int] IDENTITY(1,1) NOT NULL,

[id_modelu] [int] NOT NULL,

[dostepnosc] [char](1) NOT NULL,

[rejestracja] [varchar](10) NOT NULL,

CONSTRAINT [PK_Samochody] PRIMARY KEY CLUSTERED
```

```
(

[id_samochodu] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

ALTER TABLE [dbo].[Samochody]  WITH CHECK ADD  CONSTRAINT [FK_Samochody_Modele]
FOREIGN KEY([id_modelu])

REFERENCES [dbo].[Modele] ([id_modelu])

ALTER TABLE [dbo].[Samochody] CHECK CONSTRAINT [FK_Samochody_Modele]

ALTER TABLE [dbo].[Samochody]  WITH CHECK ADD  CONSTRAINT [CK_Samochody] CHECK
(([dostepnosc]='N' OR [dostepnosc]='T'))

ALTER TABLE [dbo].[Samochody] CHECK CONSTRAINT [CK_Samochody]
```

Wprowadzanie danych do tabel:

```
INSERT INTO Marki (marka)

VALUES('Skoda')

INSERT INTO Marki (marka)

VALUES('Opel')

INSERT INTO Marki (marka)

VALUES('BMW')

INSERT INTO Marki (marka)

VALUES('Mercedes')

INSERT INTO Marki (marka)

VALUES('Jeep')


INSERT INTO Klasy (klasa, cena_dzien)

VALUES('Miejskie', 120)

INSERT INTO Klasy (klasa, cena_dzien)
```

```
VALUES('Rodzinne', 200)

INSERT INTO Klasy (klasa, cena_dzien)

VALUES('Terenowe', 450)

INSERT INTO Klasy (klasa, cena_dzien)

VALUES('Sportowe', 600)

INSERT INTO Klasy (klasa, cena_dzien)

VALUES('Luksusowe', 1200)


INSERT INTO Modele (model, id_marki, id_klasy, il_osob)

VALUES('Wrangler', 5, 8, 5)

INSERT INTO Modele (model, id_marki, id_klasy, il_osob)

VALUES('Corsa', 2, 5, 5)

INSERT INTO Modele (model, id_marki, id_klasy, il_osob)

VALUES('Serii 3', 3, 7, 5)

INSERT INTO Modele (model, id_marki, id_klasy, il_osob)

VALUES('Klasa S', 4, 10, 4)

INSERT INTO Modele (model, id_marki, id_klasy, il_osob)

VALUES('Serii 7', 3, 10, 4)

INSERT INTO Modele (model, id_marki, id_klasy, il_osob)

VALUES('Klasa G', 4, 8, 5)

INSERT INTO Modele (model, id_marki, id_klasy, il_osob)

VALUES('Fabia', 1, 5, 5)

INSERT INTO Modele (model, id_marki, id_klasy, il_osob)

VALUES('Octavia', 1, 7, 5)

INSERT INTO Modele (model, id_marki, id_klasy, il_osob)

VALUES('AMG-GT', 4, 9, 2)

INSERT INTO Modele (model, id_marki, id_klasy, il_osob)
```

VALUES('M3', 3, 9, 4)

INSERT INTO Samochody (id_modelu, dostepnosc, rejestracja)

VALUES(3, 'T', 'KLI8744B')

INSERT INTO Samochody (id_modelu, dostepnosc, rejestracja)

VALUES(4, 'T', 'PO87945')

INSERT INTO Samochody (id_modelu, dostepnosc, rejestracja)

VALUES(5, 'T', 'SOS74123')

INSERT INTO Samochody (id_modelu, dostepnosc, rejestracja)

VALUES(6, 'T', 'KR74159')

INSERT INTO Samochody (id_modelu, dostepnosc, rejestracja)

VALUES(7, 'T', 'WAR9634A')

INSERT INTO Samochody (id_modelu, dostepnosc, rejestracja)

VALUES(8, 'T', 'RZE32154')

INSERT INTO Samochody (id_modelu, dostepnosc, rejestracja)

VALUES(9, 'T', 'PO9632G')

INSERT INTO Samochody (id_modelu, dostepnosc, rejestracja)

VALUES(10, 'T', 'KR5569L')

INSERT INTO Samochody (id_modelu, dostepnosc, rejestracja)

VALUES(1, 'T', 'GD25874')

INSERT INTO Samochody (id_modelu, dostepnosc, rejestracja)

VALUES(2, 'T', 'KLI3652B')

INSERT INTO Samochody (id_modelu, dostepnosc, rejestracja)

VALUES(3, 'T', 'OP78912')

INSERT INTO Samochody (id_modelu, dostepnosc, rejestracja)

VALUES(4, 'T', 'WAR85236')

```
INSERT INTO klienci(imie, nazwisko, nr_telefonu, pesel)
VALUES('Severyn', 'Dolhopolov', '730726981', '03052562367')
INSERT INTO klienci(imie, nazwisko, nr_telefonu, pesel)
VALUES('Janusz', 'Kowalski', '678234567', '99092365492')
INSERT INTO klienci(imie, nazwisko, nr_telefonu, pesel)
VALUES('Krystyna', 'Jantar', '696780222', '79842390871')
INSERT INTO klienci(imie, nazwisko, nr_telefonu, pesel)
VALUES('Pawel', 'Dziwisz', '628543971', '99120812050')
INSERT INTO klienci(imie, nazwisko, nr_telefonu, pesel)
VALUES('Piotr', 'Kotarba', '856852654', '12030945601')
INSERT INTO klienci(imie, nazwisko, nr_telefonu, pesel)
VALUES('Tomek', 'Debosz', '765913746', '01112889700')
INSERT INTO klienci(imie, nazwisko, nr_telefonu, pesel)
VALUES('Geralt', 'Rivii', '467222444', '91032643921')
INSERT INTO klienci(imie, nazwisko, nr_telefonu, pesel)
VALUES('Williams', 'Martinez', '888222111', '84102966244')
INSERT INTO klienci(imie, nazwisko, nr_telefonu, pesel)
VALUES('Johnson', 'Miller', '555111222', '61092556127')
INSERT INTO klienci(imie, nazwisko, nr_telefonu, pesel)
VALUES('Davis', 'Rodriguez', '777888000', '74051725371')
```

Widoki

Widok zwracający wszystkie samochody dostępne do wypożyczenia.

```
CREATE VIEW [dbo].[v_dostepne_samochody]
AS
SELECT s.id_samochodu, ma.marka, m.model, s.rejestracja, k.klasa, s.dostepnosc
FROM Samochody AS s
    JOIN Modele AS m
        on s.id_modelu = m.id_modelu
    JOIN Marki AS ma
        on m.id_marki = ma.id_marki
    JOIN Klasy AS k
        on m.id_klasy = k.id_klasy
WHERE dostepnosc = 'T'
```

Widok zwracający wszystkich klientów, u których kiedyś nastąpiło spóźnienie.

```
CREATE VIEW [dbo].[v_opoznione_osoby]
AS
SELECT DISTINCT imie, nazwisko, nr_telefonu, pesel FROM Rezerwacje as r
   JOIN Klienci AS k
      ON r.id_klienta = k.id_klienta
WHERE r.data_zwrotu> r.data_do
```

Widok zwracający dotychczasowe całkowite przychody wypożyczalni.

```
CREATE VIEW [dbo].[v_przychody]

AS

SELECT sum(cena) as cena, sum(kara) as kara

FROM Rezerwacje

WHERE stan = 'Z'
```

Widok zwracający zarezerwowane samochody.

```
CREATE VIEW [dbo].[v_zarezerwowane_samochody]

AS

SELECT      Rezerwacje.id_rezerwacji, Samochody.id_samochodu, Samochody.dostepnosc

FROM        Rezerwacje INNER JOIN

            Samochody ON Rezerwacje.id_samochodu = Samochody.id_samochodu

WHERE       Samochody.dostepnosc = 'R'
```

Procedury składowane

Procedura dodająca nowe (w tym przyszłe) rezerwacje do tabeli Rezerwacje. Procedura dba o to, aby nie było możliwości wypożyczenia danego samochodu w czasie, gdy ten sam samochód jest już zarezerwowany.

Procedura również wylicza wstępną cenę za okres wynajęcia.

```
CREATE procedure [dbo].[p_dodaj_rezerwacje]
   @id_klienta int,
@id_samochodu int,
   @data_od date,
@data_do date

as
begin try
if @id_klienta is null
or @id_samochodu is null
   or @data_od is null
or @data_do is null
   raiserror('argumenty nie mogą przyjmować wartości null', 16, 1)

if @data_od > @data_do
raiserror('Data wypozyczenia nie moze byc pozniejsza niz data zwrotu', 16, 1)

if not exists
   (
      select * from Klienci
      where Klienci.id_klienta = @id_klienta
```

```
   )
   raiserror('klient o podanym id nie istnieje', 16 ,1)

if not exists
   (
      select * from Samochody
      where Samochody.id_samochodu = @id_samochodu
   )
   raiserror('samochod o podanym id nie istnieje', 16 ,1)

if exists
(select * from Rezerwacje
where (stan = 'W' or stan = 'R') and
@id_samochodu = id_samochodu
and ((@data_od>=data_od and @data_do <=data_do) or (@data_od<=data_od and @data_do
>=data_do)
or (@data_od<=data_do and @data_do >=data_do) or (@data_od<=data_od and @data_do
>=data_od))
)
   raiserror('samochod jest wypozyczony w tym terminie', 16, 1)

insert Rezerwacje(id_klienta, id_samochodu, data_od, data_do, cena, stan)
   values(@id_klienta, @id_samochodu, @data_od, @data_do,
(
select cena_dzien * DATEDIFF(DAY, @data_od, @data_do) from Klasy as k
join Modele as m on k.id_klasy = m.id_klasy
join Samochody as s on s.id_modelu = m.id_modelu
where @id_samochodu = s.id_samochodu
```

```
),
'R'
)

end try
begin catch

   print error_message()

   declare @err_msg varchar(100)
   set @err_msg = error_message()

   raiserror(@err_msg, 16, 1)

end catch
```

Procedura, której wykonanie rozpoczyna okres wynajęcia danego samochodu klientowi. Aktualizuje pole ‘stan’ (z tabeli Rezerwacje) oraz pole informujące o dostępności samochodu.

```
CREATE procedure [dbo].[p_odbierz_samochod]
   @id_samochodu int,
@id_rezerwacji int

as
begin try
if
@id_samochodu is null or
@id_rezerwacji is null
   raiserror('argumenty nie mogą przyjmować wartości null', 16, 1)

if not exists
   (
      select * from Samochody
      where Samochody.id_samochodu = @id_samochodu
   )
   raiserror('samochod o podanym id nie istnieje', 16 ,1)

if not exists
   (
      select * from Rezerwacje
      where Rezerwacje.id_rezerwacji = @id_rezerwacji
   )
   raiserror('rezerwacja o podanym id nie istnieje', 16 ,1)
```

```
UPDATE Samochody
SET dostepnosc = 'N'
WHERE id_samochodu = @id_samochodu

update Rezerwacje
  set stan = 'W'
  where id_samochodu = @id_samochodu and id_rezerwacji = @id_rezerwacji

end try
begin catch

   print error_message()

   declare @err_msg varchar(100)
   set @err_msg = error_message()

   raiserror(@err_msg, 16, 1)

end catch
```

Procedura, której wykonanie sygnalizuje zwrócenie samochodu przez klienta oraz kończy okres wynajęcia danego auta. W wypadku spóźnienia naliczana zostaje dodatkowo kara jako dwukrotność kwoty wynajmu za każdy dzień, aktualizuje pole ‘stan’ (z tabeli Rezerwacje) oraz pole informujące o dostępności samochodu.

```
CREATE procedure [dbo].[p_oddaj_samochod]
   @id_rezerwacji int,
   @data_zwrotu date

as
begin try
if @id_rezerwacji is null
   or @data_zwrotu is null
   raiserror('argumenty nie mogą przyjmować wartości null', 16, 1)

   if not exists
   (
      select id_rezerwacji from Rezerwacje as r
      where r.id_rezerwacji = @id_rezerwacji
   )
   raiserror('rezerwacja o podanym id nie istnieje', 16 ,1)

   if @data_zwrotu > (select data_do from Rezerwacje where id_rezerwacji = @id_rezerwacji)
   update rezerwacje
   set kara = (
      select cena_dzien * 2 * (datediff(d, (select data_do from Rezerwacje where id_rezerwacji =
@id_rezerwacji), @data_zwrotu))
      from Klasy as k
```

```
        join modele as m
            on k.id_klasy = m.id_klasy
        join Samochody as s
            on s.id_modelu = m.id_modelu
        join Rezerwacje as r
            on r.id_samochodu = s.id_samochodu
      where id_rezerwacji = @id_rezerwacji
   )
   where id_rezerwacji = @id_rezerwacji

   update Rezerwacje
   set data_zwrotu = @data_zwrotu
   where id_rezerwacji = @id_rezerwacji

   update Rezerwacje
   set stan = 'Z'
   where id_rezerwacji = @id_rezerwacji

   update Samochody
   SET dostepnosc = 'T'
   WHERE id_samochodu = (select id_samochodu from Rezerwacje where id_rezerwacji =
@id_rezerwacji)

end try
begin catch

   print error_message()
```

```
    declare @err_msg varchar(100)

    set @err_msg = error_message()


    raiserror(@err_msg, 16, 1)


end catch
```