Zarządzanie flotą samochodową dla Firm

Wykonane przez:

Mateusz Jaruga nr. Indexu: 392803

Mateusz Kalenik nr. Indexu: 392799

Spis treści

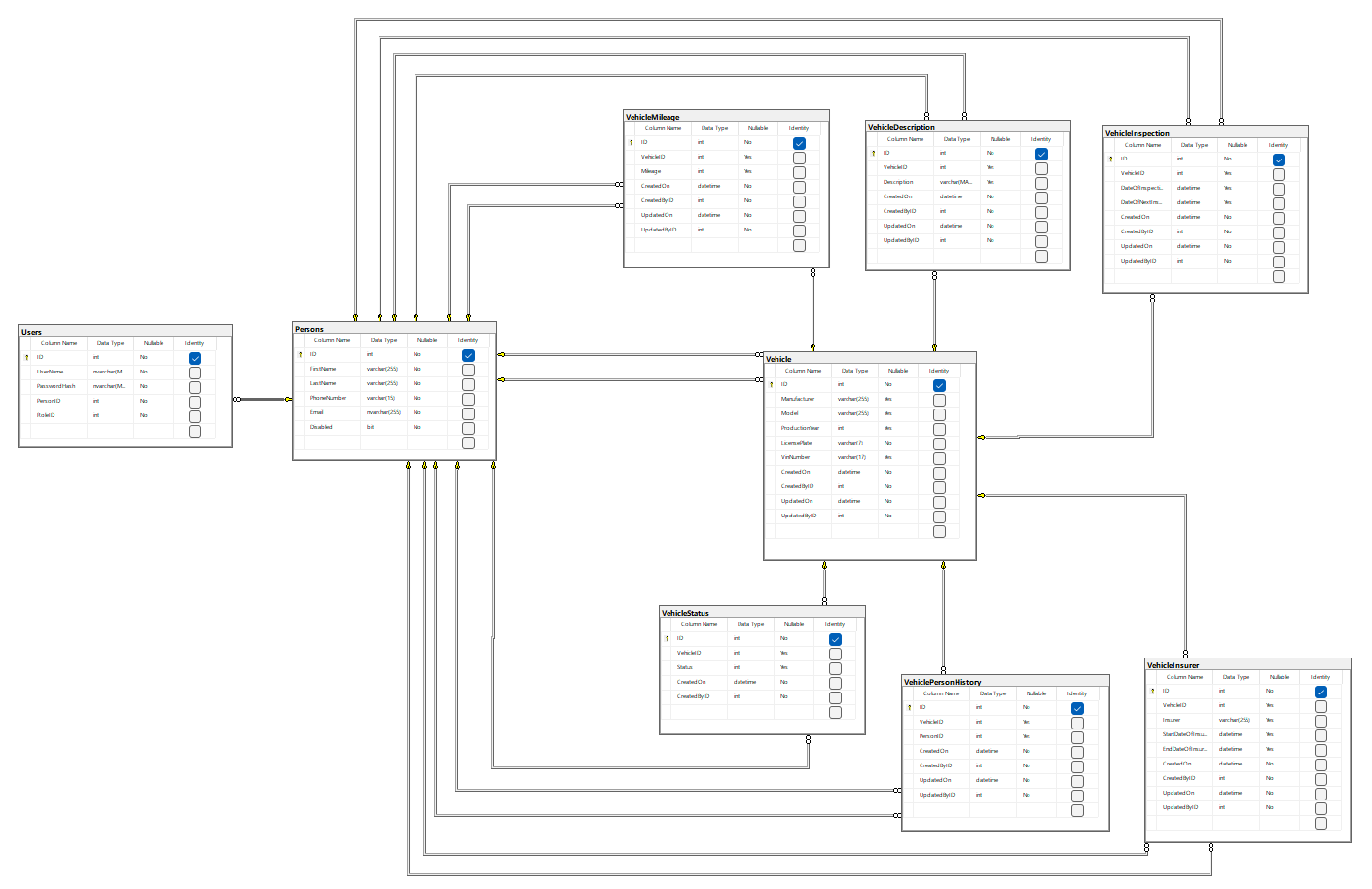
[Diagram baz danych: 1](#_Toc132296482)

[Opis kodu: 1](#_Toc132296483)

[Opis reguł biznesowych: 18](#_Toc132296484)

[Przewodnik po aplikacji: 18](#_Toc132296485)

# Diagram baz danych:



# Opis kodu:

Wstępna informacja: Klasy utworzone w folderze poza klasą Command Response odzwierciedlają tabele z bazy danych, dodatkowo klasa Modelextension jest używana do dziedziczenia zmiennych takich jak UpdatedOn, UpdatedById, CreatedOn, CreatedById

**Lokalizacja**: CarFleetDomain -> Models -> CommandResponse

**Funkcja**: DataResponse i CommandRespone

**Opis działania i zastosowania**: Podane klasy powstały w celu łatwiejszego zwrócenia wyników operacji, Zmienna „Success” w Data response powie nam czy wykonana przez nas operacja się udała czy też nie, Message zawiera wiadomość związaną zazwyczaj z wykonanym zadaniem, natomiast klasa Command response jest klasą generyczną która dziedziczy po Data response. Umożliwia ona dodatkowe zwrócenie danych

**Kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Lokalizacja**: CarFleetDomain -> Context

**Funkcja**: GetTable

**Opis działania i zastosowania**: Funkcja GetTable przyjmuje jako parametry gotową komendę sql „SqlCommand” oraz utworzony wcześniej data set. Zwraca ona Data Response.

Funkcja ta za pomocą Sql Data Adaptera tworzy nową tabelę która przyjmuje nazwę typu T

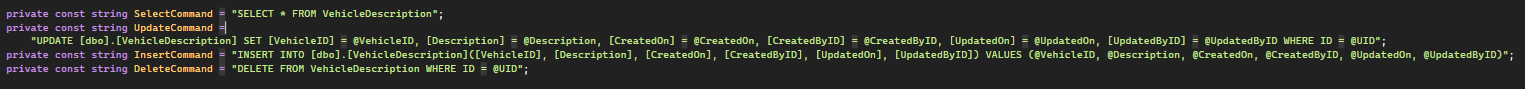
**Kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Poniżej znajdują się funkcje które powtarzają się w prawie każdej klasie znajdującej się w folderze: CarFleetDomain -> Models

Każda z klas ma 4 zmienne typu string które są na stałe przypisane i zawierają one naszykowane kwerendy pod dane działanie np.:



**Lokalizacja**: CarFleetDomain -> Models

**Funkcja**: Get NazwaKlasy Query

**Opis działania i zastosowania**: Funkcja ma na celu pobieranie danych z tabeli z bazy która odpowiada danej klasie. Przyjmuje on utworzony wcześniej data set, oraz opcjonalnie może przyjąć inną kwerendę do wyciągnięcia danych (np. kwerendę która posiada warunek where w sobie). W tej metodzie została zastosowana funkcja GetTable aby zredukować powtarzalność kodu. Zwracanym typem danych jest Data Response informując przy tym czy udało się wyciągnąć poprawnie dane czy też nie, oraz zwraca odpowiednią wiadomość.

**Przykładowy Kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Lokalizacja**: CarFleetDomain -> Models

**Funkcja**: Update NazwaKlasy Command

**Opis działania i zastosowania**: Funkcja służąca do aktualizowania danych w odpowiadającej klasie tabeli w bazie danych. Przyjmuje ona zmodyfikowany data set oraz zwraca data response z wynikiem operacji.

**Przykładowy kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Lokalizacja**: CarFleetDomain -> Models

**Funkcja**: insert NazwaKlasy Command

**Opis działania i zastosowania**: Funkcja służąca do wstawiania nowych danych w odpowiadającej klasie tabeli w bazie danych. Przyjmuje ona zmodyfikowany data set który zawiera nowe dane oraz zwraca Data response z wynikiem działania

**Przykładowy kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Lokalizacja**: CarFleetDomain -> Models

**Funkcja**: Delete NazwaKlasy Command

**Opis działania i zastosowania**: Funkcja mająca na celu usunięcie wiersza w odpowiadającej klasie tabeli w bazie danych. Funkcja ta przyjmuje ID wiersza (np. id opisu) oraz zwraca ona Data response z wynikiem działania

**Przykładowy kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Lokalizacja**: CarFleetDomain -> Functions

**Funkcja**: InsertNewVehicleDescription

**Opis działania i zastosowania**: Funkcja ta służy wstawienia nowego wiersza do data set i zarazem tabeli Vehicle Description, przyjmuje ona utworzony wcześniej data set, id pojazdu dla którego jest tworzony opis, aktualnie zalogowanego użytkownika, oraz tekst opisu.

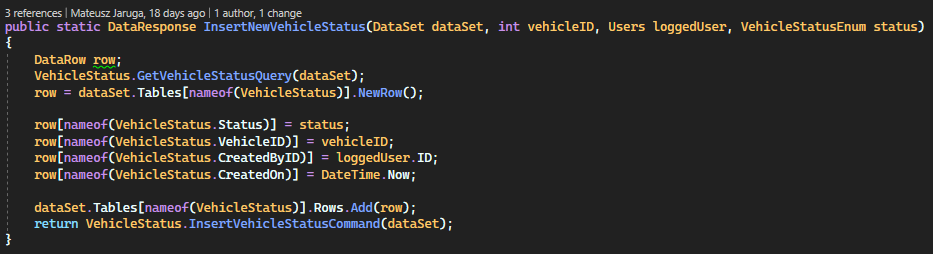
Funkcja ta wpierw pobiera aktualne dane z tabeli Vehicle Description używając gotowej funkcji Get Vehicle Description Query znajdującej się w klasie VehicleDescription, tworzy potem w nowy wiersz i uzupełnia go danymi, po czym dodaje ten wiersz do data seta a następnie wstawia najnowszy wiersz do tabeli Vehicle Description za pomocą funkcji Insert vehicle Description Command znajdującej się w klasie vehicle Description zwracając przy okazji Data Response z wynikiem działania

**Przykładowy kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Powyższy kod działa na podobnej zasadzie dla następujących funkcji:



Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Lokalizacja**: CarFleetDomain -> Functions -> EmployeeSystem

**Funkcja**: GetEmployees

**Opis działania i zastosowania**: Funkcja mająca na celu wyciągnięcie aktualnej tabeli Persons. Zwracanym typem jest Command Response jako data set który będzie zawierał tabele Persons. Funkcja tworzy nowy data set, po czym tworzy kwerendę SQL i za pomocą funkcji GetTable w klasie Context wyciąga dane i je zwraca, lub zwracany jest błąd jeśli nic nie zostało wyciągnięte

**Kod**:

Obraz zawierający tekst, zrzut ekranu, ekran, blisko/zamknięte

Opis wygenerowany automatycznie

**Lokalizacja**: CarFleetDomain -> Functions -> EmployeeSystem

**Funkcja**: GetPersonsWithRole

**Opis działania i zastosowania**: Funkcja ta ma celu wyciągnięcie osoby o danym ID. Jako parametry dostaje on id osoby oraz connection string a zwracane jest Command Response typu Persons. Funkcja ta szykuje wpierw nową kwerendę po czy tworzy nowy SqlDataAdapter w celu wyciągnięcia „nietypowych” danych z bazy, tworzony jest dataSet który jest uzupełniony danymi z bazy, następuje potem sprawdzenie czy został w data set’cie znajdują się dane po czy tworzony jest obiek typu Persons, który zostaje także zwrócony

**Kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Lokalizacja**: CarFleetDomain -> Functions -> EmployeeSystem

**Funkcja**: GetUserRole

**Opis działania i zastosowania**: Funkcja GetUserRole służy do pobierania roli użytkownika na podstawie jego identyfikatora osoby (personID) z tabeli Users w bazie danych. Przyjmuje również jako parametr ciąg połączenia do bazy danych (connectionString), a zwracanym typem jest Command Response wraz z klasą Users. Następnie tworzony jest nowy obiekt DataSet o nazwie dataset, który zostanie wypełniony wynikami zapytania SQL za pomocą metody Fill adaptera. Funkcja sprawdza, czy zwrócone dane zawierają co najmniej jeden wiersz. Jeśli tak, to tworzony jest nowy obiekt Users na podstawie danych z wiersza wyników zapytania SQL. Gdy wszystko przebiegło pomyślnie tabela Users zostaje zaktualizowana.

**Kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Lokalizacja**: CarFleetDomain -> Functions -> EmployeeSystem

**Funkcja**: DisablePerson

**Opis działania i zastosowania**: Funkcja służąca do dezaktywacji/aktywacji konta użytkownika. Jako parametr przyjmuje id, oraz disabled a zwracanym typem jest Command Response wraz z klasą Persons. Funkcja tworzy nowy data set i uzupełnia go danymi z tabeli Persons w bazie za pomocą funkcji GetPersonsQuery w klasie Persons. Następnie wyszukuje wiersz na podstawie przekazanego identyfikatora(id).W zależności od wartości parametru “disabled”(czyli true lub false) ustawia wartość w kolumnie “Disabled” na 0 lub 1. Jeśli wszystko przebiegło pomyślnie tabela Persons zostaje zaktualizowana.

**Kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Lokalizacja**: CarFleetDomain -> Functions -> EmployeeSystem

**Funkcja**: InsertOrUpdateUser

**Opis działania i zastosowania**: Funkcja służy do tworzenia lub aktualizacji konta użytkownika. Jako parametr przyjmuje personID, userName, passwordHash, roleID. a zwracanym typem jest Command Response wraz z klasa Users. Funkcja najpierw sprawdza poprawność wprowadzonych danych po czym w przypadku poprawnej weryfikacji sprawdza czy użytkownik z podanym danymi nie istnieje. Jeśli nie istnieje tworzy nowy data set i uzupełnia go danymi z tabeli Users w bazie za pomocą funkcji GetUsersQuery w klasie Users. Następnie tworzony jest nowy wiersz, który jest uzupełniony danymi i wstawiony ponownie do data seta. Następnie wywołuje funkcję insertUsersCommand z klasy Users która dodaje nowego użytkownika do bazy danych na podstawie obiektu DataSet. Gdy wszystko przebiegło pomyślnie tabela Users zostaje zaktualizowana. Jeśli użytkownik istnieje zostaje wywołana funkcja UpdateUser z przekazanymi parametrami (userName, passwordHash, personID, roleID) która aktualizuje dane użytkownika w tabeli Users na podstawie identyfikatora osoby(personID). Gdy wszystko przebiegło pomyślnie tabela Users zostaje zaktualizowana.

**Kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Lokalizacja**: CarFleetDomain -> Functions -> EmployeeSystem

**Funkcja**: UpdateUser

**Opis działania i zastosowania**: Funkcja służy do aktualizowania danych użytkownika w tabeli Users. Jako parametry przyjmuje userName, passwordHash, personID, roleID, a zwracanym typem jest Command Response wraz z klasa Users. Następnie funkcja tworzy nowy obiekt DataSet i pobiera dane z tabeli Users za pomocą funkcji GetUsersQuery z klasy Users. Następnie wyszukuje wiersz w tabeli Users na podstawie przekazanego identyfikatora personID. Jeśli wiersz zostanie znaleziony, funkcja aktualizuje jego pola na nowe wartości przekazane jako parametry. Następnie wykonuje operację aktualizacji w bazie danych za pomocą funkcji UpdateUsersCommand z klasy Users. Gdy wszystko przebiegło pomyślnie tabela Users zostaje zaktualizowana.

**Kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Lokalizacja**: CarFleetDomain -> Functions -> EmployeeSystem

**Funkcja**: UpdateEmployee

**Opis działania i zastosowania**: Funkcja UpdateEmployee służy do aktualizacji danych pracownika w tabeli Persons w bazie danych.

Przyjmuje jako parametry id, firstName, lastName, phone, email oraz disabled. a zwracanym typem jest Command Response wraz z klasa Persons. Następnie funkcja tworzy nowy obiekt DataSet i pobiera dane z tabeli Persons za pomocą funkcji GetPersonsQuery z klasy Persons. Następnie wyszukuje wiersz w tabeli Persons na podstawie przekazanego identyfikatora id.

Jeśli wiersz zostanie znaleziony, funkcja aktualizuje jego pola FirstName, LastName, PhoneNumber, Email oraz Disabled na nowe wartości przekazane jako parametry.

Następnie wykonuje operację aktualizacji w bazie danych za pomocą funkcji UpdatedPersonsCommand z klasy Persons. Gdy wszystko przebiegło pomyślnie tabela Persons zostaje zaktualizowana.

**Kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Lokalizacja**: CarFleetDomain -> Functions -> EmployeeSystem

**Funkcja**: InsertNewEmployee

**Opis działania i zastosowania**: Funkcja służąca do wstawienia nowego pracownika do tabeli Persons w bazie. Jako parametry przyjmuje imię, nazwisko, numer teleofnu, email, ID roli i czy konto ma być aktywne, a zwracanym typem jest Command Response wraz z klasa Persons. Funkcja wpierw sprawdza poprawność adresu email po czym w przypadku poprawnej weryfikacji tworzy nowy data set i uzupełnia go danymi z tabeli Persons w bazie za pomocą funkcji GetPersonsQuery w klasie Persons. Następnie tworzony jest nowy wiersz, który jest uzupełniony danymi i wstawiony ponownie do data seta. Taki uzupełniony data set jest podawany do funkcji InsertPersonsCommand w klasie Perons aby wstawić nowy wiersz do bazy. Jeśli wstawienie przebiegło pomyślnie zostaje wówczas odnalezione ID najnowszej osoby która została wstawiona i tworzony jest użytkownik na podobnej zasadzie: Pobierz aktualną tabele users -> utwórz hash hasła dla nowego użytkownika -> utwórz nowy wiersz data Row -> wstaw wiersz do data set -> podaj data set do funkcji InsertUsersCommand.

**Kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Lokalizacja**: CarFleetDomain -> Functions -> LoginSystem

**Funkcja**: LoginUser

**Opis działania i zastosowania**: Funkcja służąca do zalogowania użytkownika. Przyjmuje ona login i hasło użytkownika, a zwraca Command response typu Users, czyli w przypadku poprawnego zalogowania się użytkownika klasa users zostanie uzupełniona danymi o aktualnie zalogowanym użytkowniku. Takowy user zostanie także później zapamiętanym na przestrzeni programu. Funkcja na początku tworzy hash hasła potrzebnego do weryfikacji konta, po czym za pomocą funkcji GetUsersQuery w klasie Users pobiera całą tabelę users z bazy i uzupełnia data set po czym za pomocą LINQ zostaje wyszukany pasujący do danych wiersz, a jeśli takowy nie zostanie odnaleziony funkcja się zakończy niepowodzeniem. W przypadku poprawnego odnalezienia wiersza z użytkownikiem pobierana jest tabela persons do data seta za pomocą funkcji GetPersonsQuery w klasie Persons a następnie za pomocą LINQ wyszukiwany jest odpowiedni wiersz w tej tabeli który posiada relację z tabelą Users. Jeśli osoba została znaleziona oraz jej konto jest aktywne wówczas system zwraca pozytywny wynik operacji wraz z uzupełnionym modelem users.

**Kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Lokalizacja**: CarFleet -> Views -> VehicleForms -> CarDetailsForm

**Funkcja**: BtnDeleteDescriptionHandler

**Opis działania i zastosowania**: Funkcja stworzona dla przycisku służącego do usuwania opisu dodanego do pojazdu, przyjmuje ona ID opisu. Funkcja wyświetla message boxa w celu upewnienia się czy użytkownik na pewno chce usunąć opis. W przypadku jeśli kliknie przycisk „Cancel” nic się nie zadzieje, natomiast jeśli został wybrany przycisk „OK” zostanie uruchomiona funkcja DeleteVehicleCommand znajdująca się w klasie VehicleDescription która usunie wiersz w bazie o podanym ID. W celu zaktualizowania tabeli z opisem pojazdów pobierze nowe dane z bazy, do czego służy funkcja GetVehicleDescriptionQuery, przypisze ona od nowa data seta do data grid view. Na koniec funkcja wyświetli komunikat o pomyślnym usunięciu opisu.

**Kod**:

Obraz zawierający tekst

Opis wygenerowany automatycznie

# Opis reguł biznesowych:

Aplikacja zawiera w sobie następujące reguły biznesowe:

Oznaczanie pojazdów których data ważności przeglądu kończy się za 2 tygodnie

Oznaczanie pojazdów których data ważności ubezpieczenia kończy się za 2 tygodnie

Oznaczenie statusu pojazdów w zależności czy:

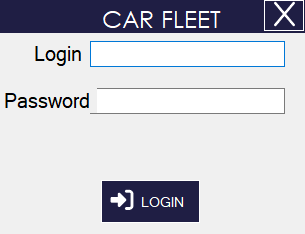
* Jest gotowy do przydzielenia pracownikowi,
* Jest aktualnie w użyciu,
* Jest zarezerwowany dla pracownika,
* Jest w serwisie,
* Zakończył się jego okres wsparcia przez firmę,

Aplikacja automatycznie uzupełnia kiedy może odbyć się następny przegląd (rok od daty wykonania przeglądu)

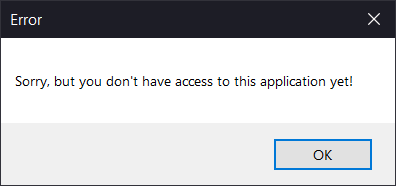
Aplikacja automatycznie uzupełnia datę końca ubezpieczenia (rok od daty rozpoczęcia ubezpieczenia)

# Przewodnik po aplikacji:

Po uruchomieniu aplikacji pojawia się ekran Logowania.



Po próbie zalogowania się na konto urzytkownika wyświetla się podana poniżej wiadomość.



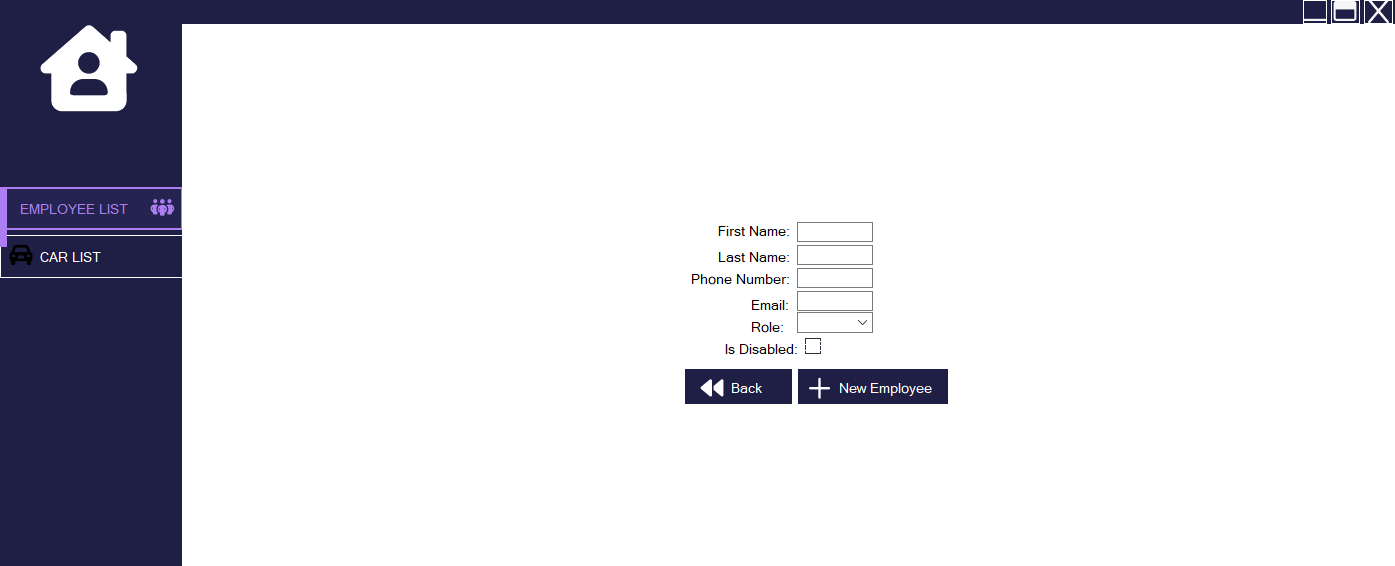
Po zalogowaniu się na konto administratora pojawia się panel administracyjny z dwoma przyciskami Employee List oraz Car List.

Po wybraniu Employee List ukazuje się panel z wyświetloną listą pracowników. Mamy możliwość edytować każdego pracownika, jak również dezaktywowania jego konta. Po prawej stronie znajduje się przycisk New Employee który umożliwia nam dodanie nowego pracownika. Poniżej mamy możliwość przeszukania listy pracowników w celu znalezienia konkretnej osoby.

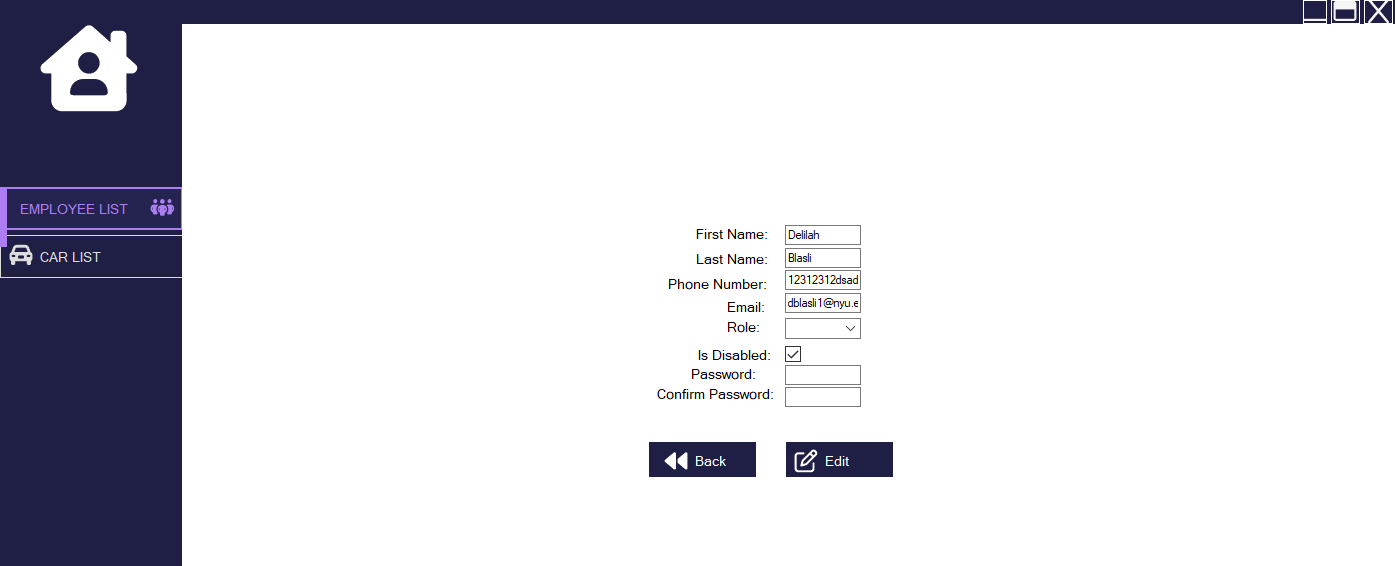
Obraz zawierający stół

Opis wygenerowany automatycznie

Po naciśnięciu New Employee pojawia się okno dodawania. W celu dodania pracownika należy podać jego imie, nazwisko, numer telefonu, email, role jaką pełni w firmie, oraz czy jego konto jest aktywne czy wyłączone.



Po naciśnięciu przycisku Edit mamy możliwość edycji danych pracownika. Możemy zmienić jego imię, nazwisko, numer telefonu, email, role w firmie, czy konto jest aktywne, oraz hasło.

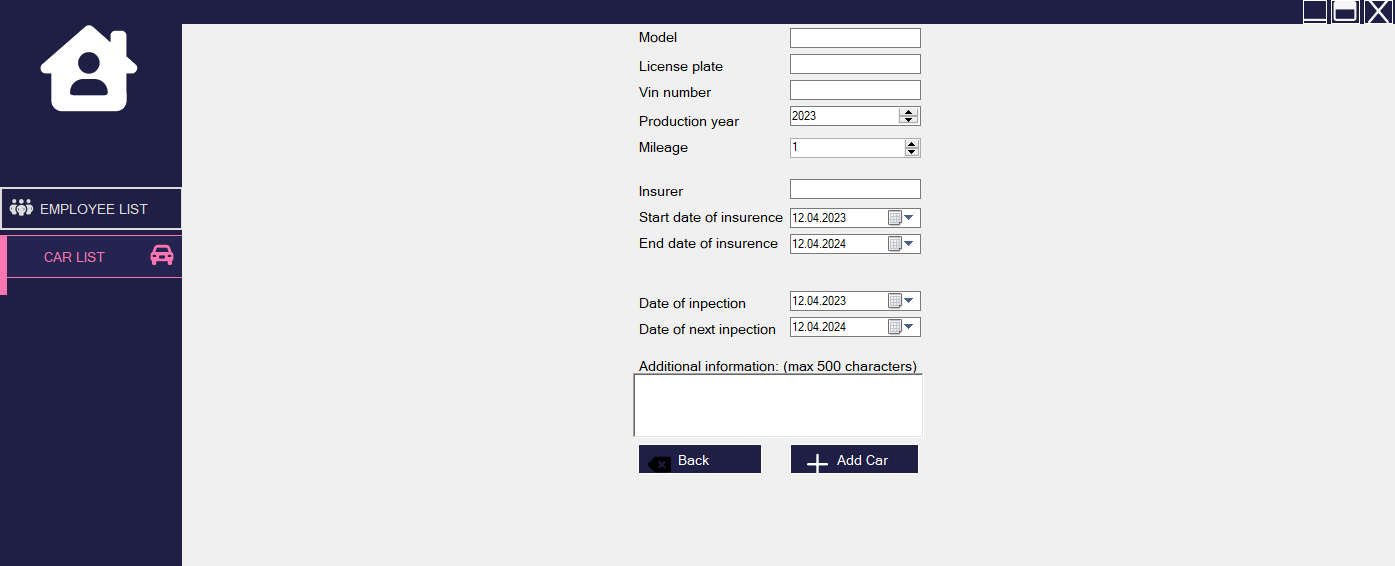


Po naciśnięciu przycisku Car List pojawia nam się lista samochodów. Mamy możliwość dodania nowego pojazdu jak również sprawdzenie detali samochodu.

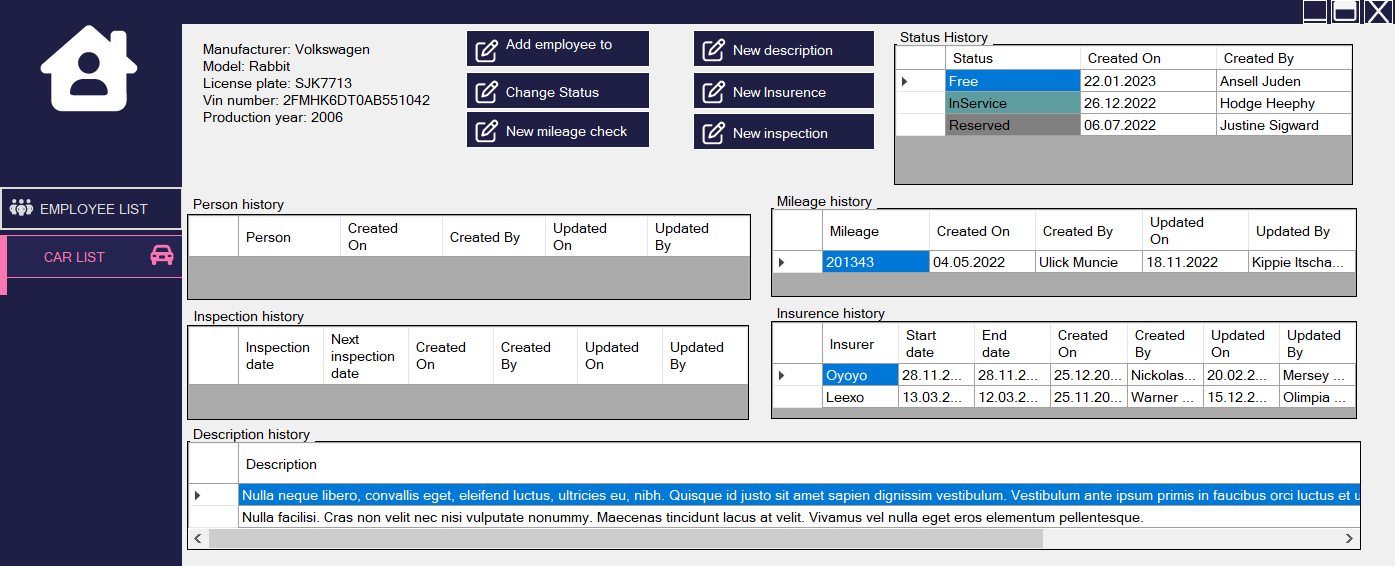
Obraz zawierający stół

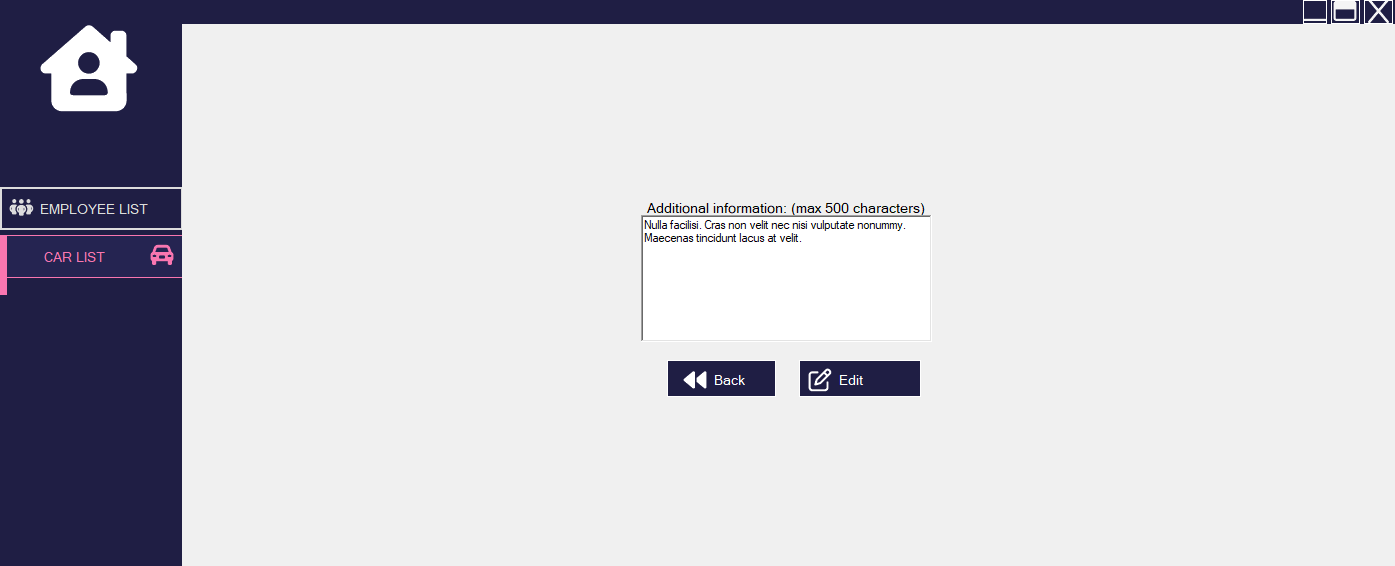
Opis wygenerowany automatycznie

Po naciśnięciu przycisku Add new vehicle przekierowuje nas do panelu dodawania pojazdu. Aby dodać nowy pojazd należy podać model samochodu, tablice rejestracyjne, numer vin, rok produkcji pojazdu, przebieg, nazwę ubezpieczyciela, daty ubezpieczenia, daty przeglądu, oraz informacje dodatkowe

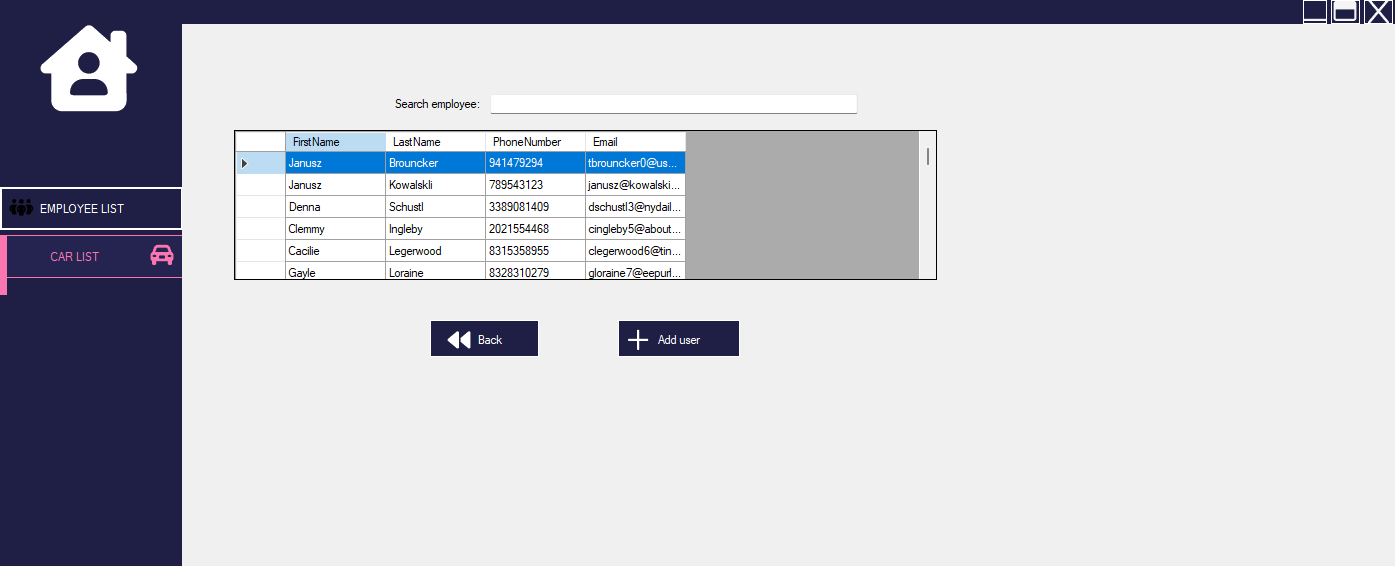


Po wybraniu przycisku Details pojawia nam się panel z detalami pojazdu. W lewym górnym rogu widzimy markę, model, numer rejestracyjny, numer vin i rok produkcji samochodu. Widzimy również person history tabelę z listą osób które użytkowały bądź użytkują pojazd, osoba obecnie korzystająca z pojazdu znajduje się na samej górze tabeli. Następną tabelą jest Inspection History gdzie znajdują się informacje o przeglądach pojazdu. Kolejną tabelą jest Status History gdzie mamy historię użytkowania pojazdu. Poniżej jest tabela Mileage history w której mamy informacje o historii przebiegu pojazdu. Następnie widzimy Insurence history czyli tabela z informacją na temat historii ubezpieczeń pojazdu. Ostatnią tabelą jest Description history gdzie mamy historię opisów pojazdu, mamy w niej możliwość usunięcia bądź edycji opisu. Mamy też możliwość wprowadzenia nowych danych do każdej z podanych tabel poprzez kliknięcie odpowiedniego przycisku.

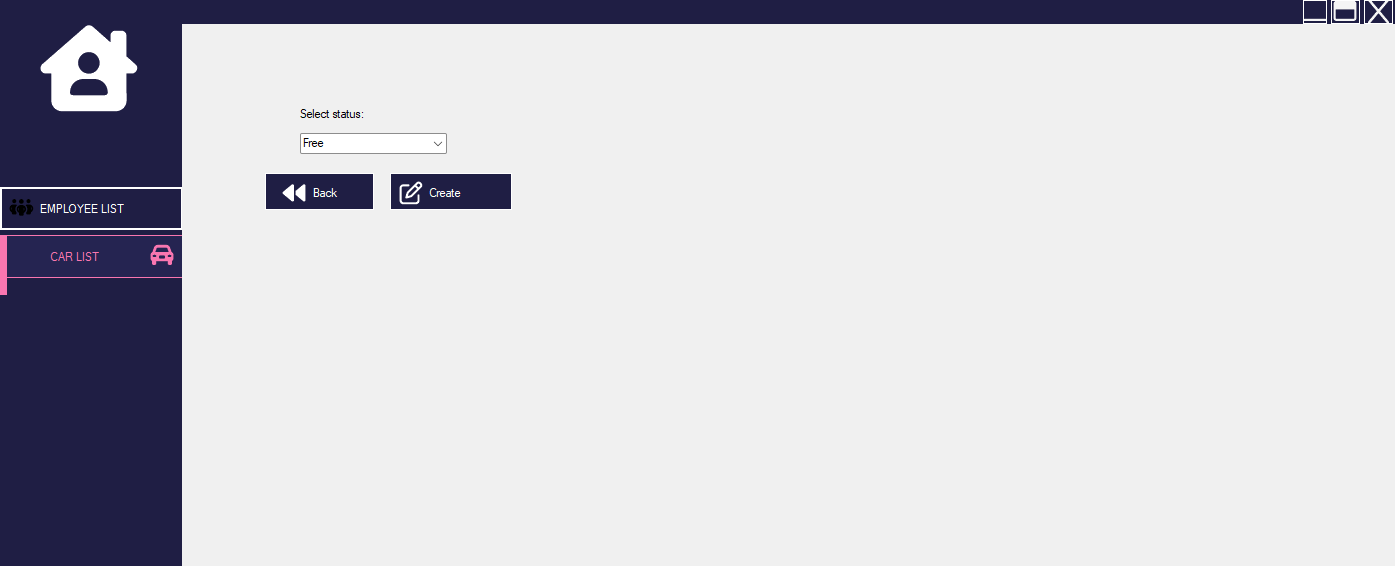
Po naciśnięciu Edit w tabeli Description history przenosi nas do okna w którym mamy możliowść edycji opisu pojazdu.



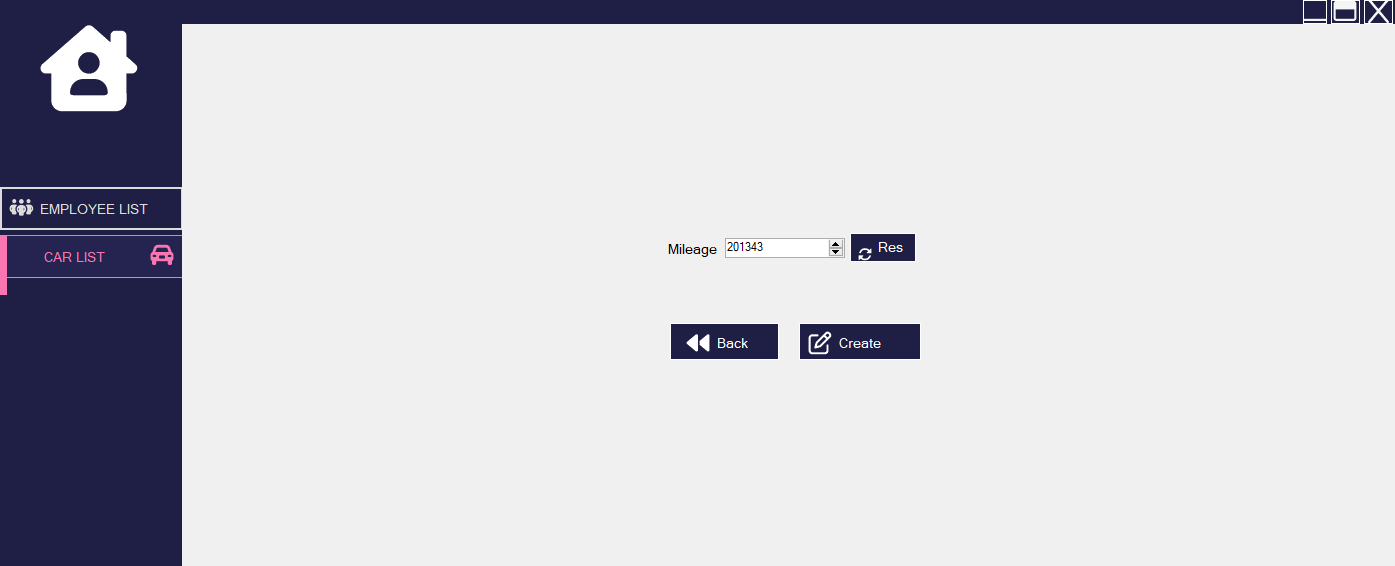
Po naciśnięciu przycisku Add employee to przenosi nas do okna w którym mamy dodajemy użytkownika do pojazdu, w celu ułatwienia znalezienia użytkownika mamy możliwość przeszukania tabeli.



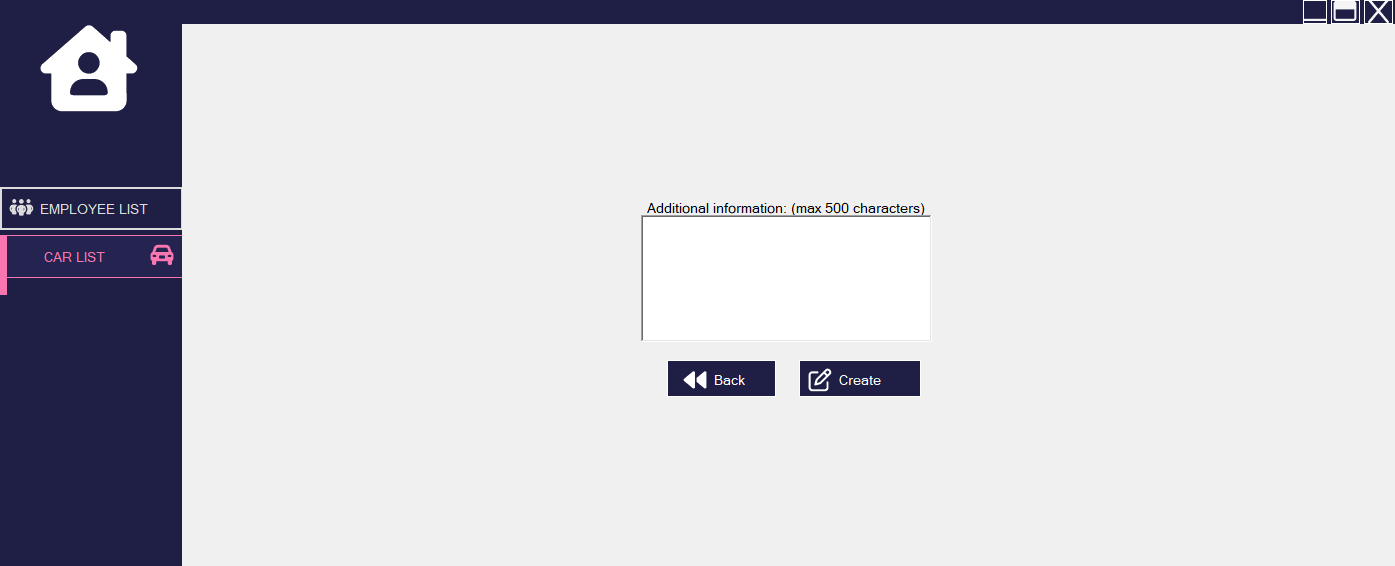
Po naciśnięciu przycisku Change Status przenosi nas do okna w którym mamy możliwość zmiany statusu pojazdu np. z free na InService.



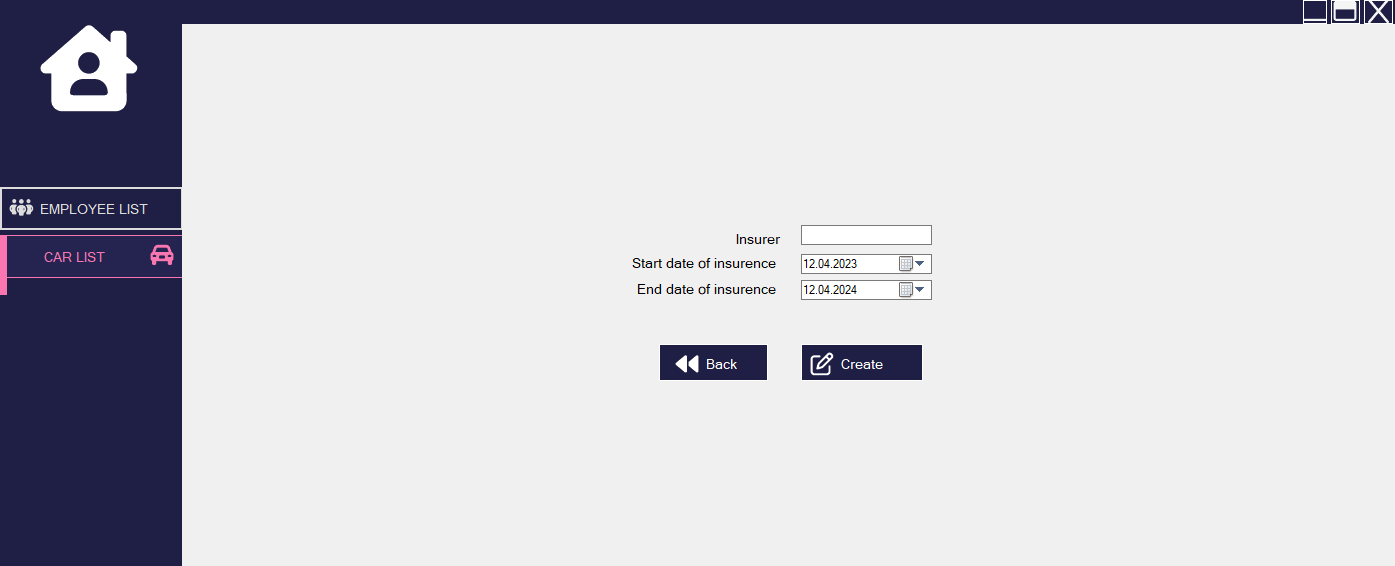
Po naciśnięciu przycisku New mileage change mamy możliwość zmiany przebiegu pojazdu. W razie popełnienia blędu i wpisania złej wartości możemy nacisnąć przycik Resetu który nam przywróci ostatnią zapisaną wartość przebiegu.



Po naciśnięciu przycisku New description mamy możliwość dodania nowego opisu pojazdu.



Po naciśnięciu przycisku New insurence mamy możliwość dodania nowego ubezpieczyciela poprzez podanie nazwy ubezpieczyciela jak również daty ubezpieczenia pojazdu a także daty wygaśnięcia ubezpieczenia.



Po naciśnięciu przycisku New inspection mamy możliwość dodania nowego przeglądu pojazdu poprzez podanie daty dokonania inspekcji jak również daty wygaśnięcia przeglądu.

