Gliwice, 18.06.2020

Bazy danych - Projekt

System wspomagający pracę sieci salonów samochodowych

Skład sekcji:
Dorian Barański
Dominika Kulczyk
Anna Nawrot
Dawid Marczewski
Radosław Płachta
Maciej Wysowski

1. Treść zadania

Tematem naszego projektu było stworzenie systemu, który umożliwi pracę w sieci salonów samochodowych. W bazie miały znajdować się informacje o użytkownikach; kierowniku salonu, sprzedającym, serwisancie czy administratorze. Dodatkowo poprzez tworzenie danych słownikowych musiały znaleźć się informacje o samochodzie, modelach, markach, salonach, pracownikach. Należało zawrzeć dane o klientach, umożliwić realizację zamówienia, z możliwością montażu dodatkowych opcji i usług. Przeprowadzić ewidencję sprzedaży samochodów, w jakim salonie został sprzedany, komu i przez kogo. Kierownik salonu ma możliwość zarządzania stanem posiadanych samochodów. Należało zawrzeć również system raportowy, który umożliwia stworzenie wykazu zrealizowanych zamówień, historii zamówień danego klienta, ranking najlepiej sprzedających salonów. Help dla osoby obsługującej system.

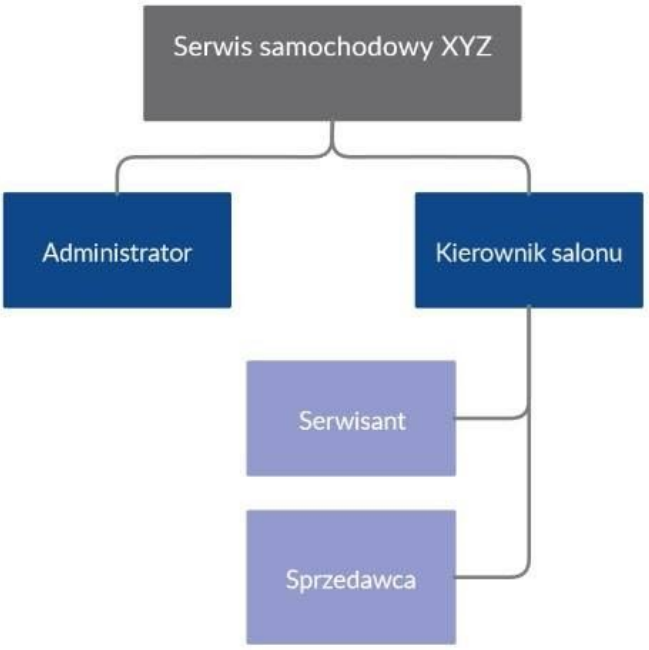
2. Oprogramowanie i sprzęt

Nasz projekt stworzyliśmy pracując w Visual Studio, front został zrobiony w Window Forms, bazy danych SQL, język C#.

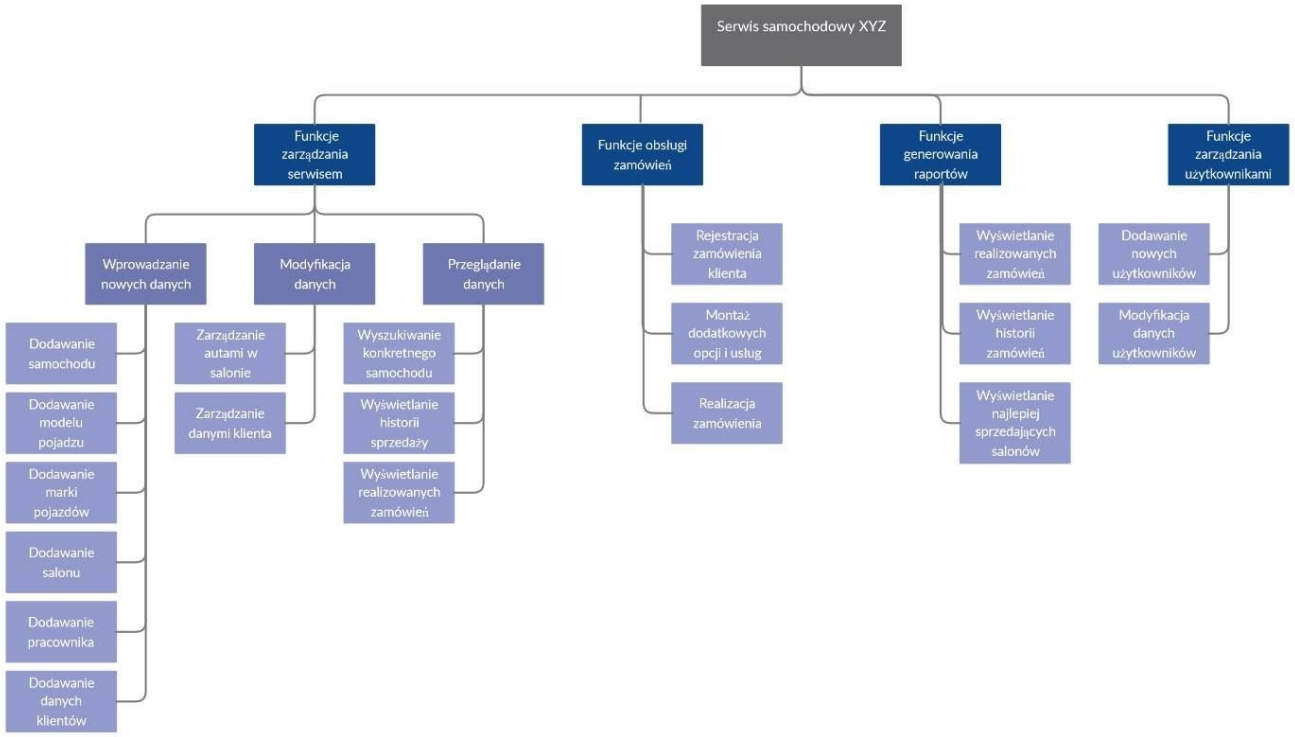
Aby uruchomić system nie trzeba posiadać wygórowanego sprzętu oraz oprogramowania. Program może być uruchomiony w trybie offline.

3. Hierarchie

3.1 Hierarchia ról

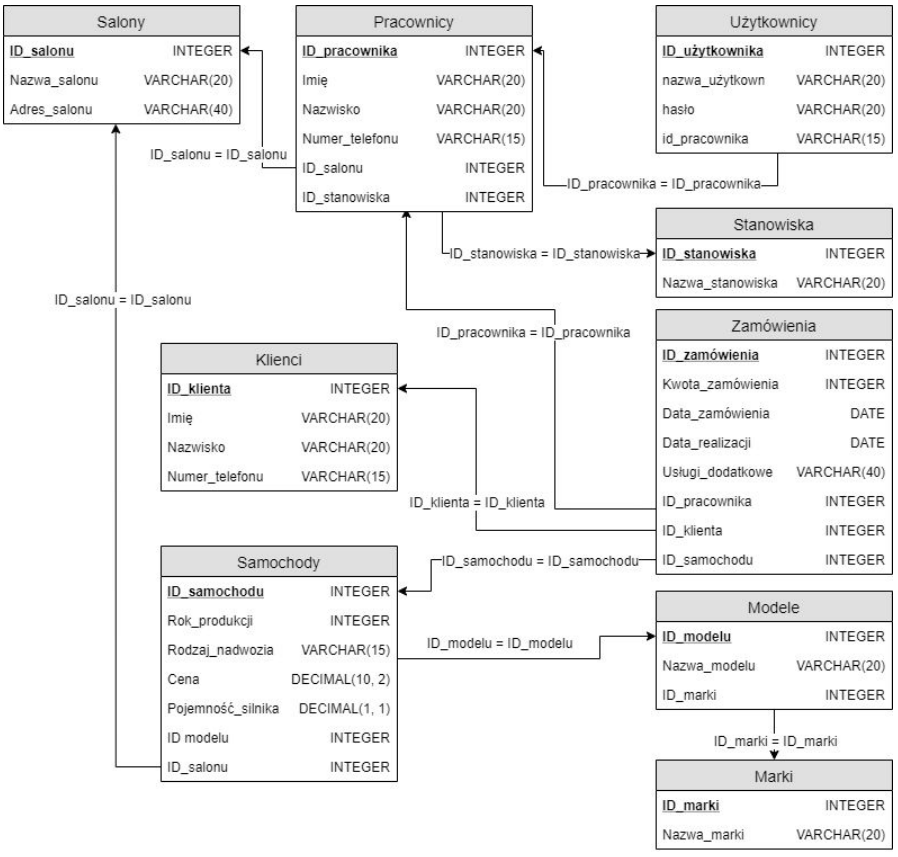


3.2 Hierarchia funkcji

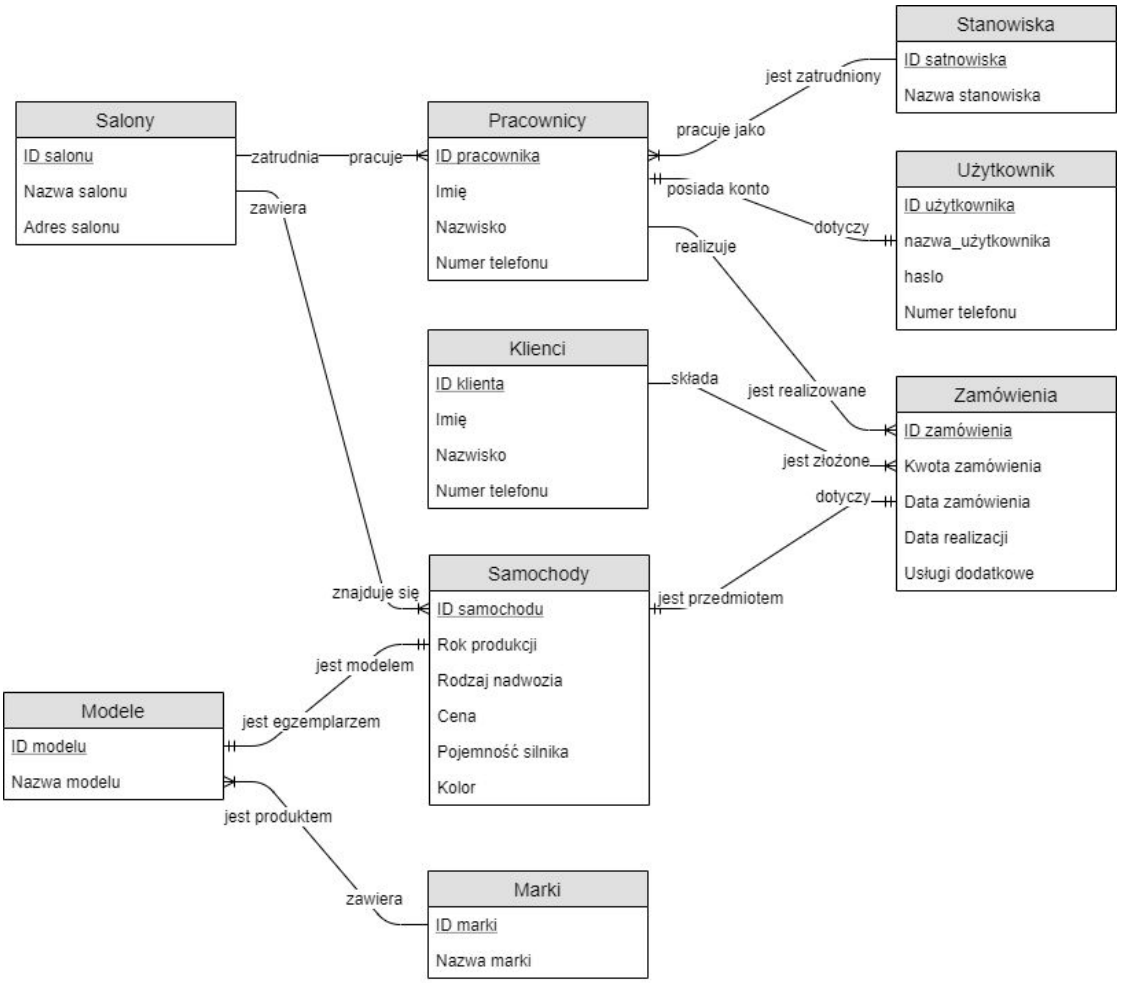


4. Modele

4.1 Model fizyczny



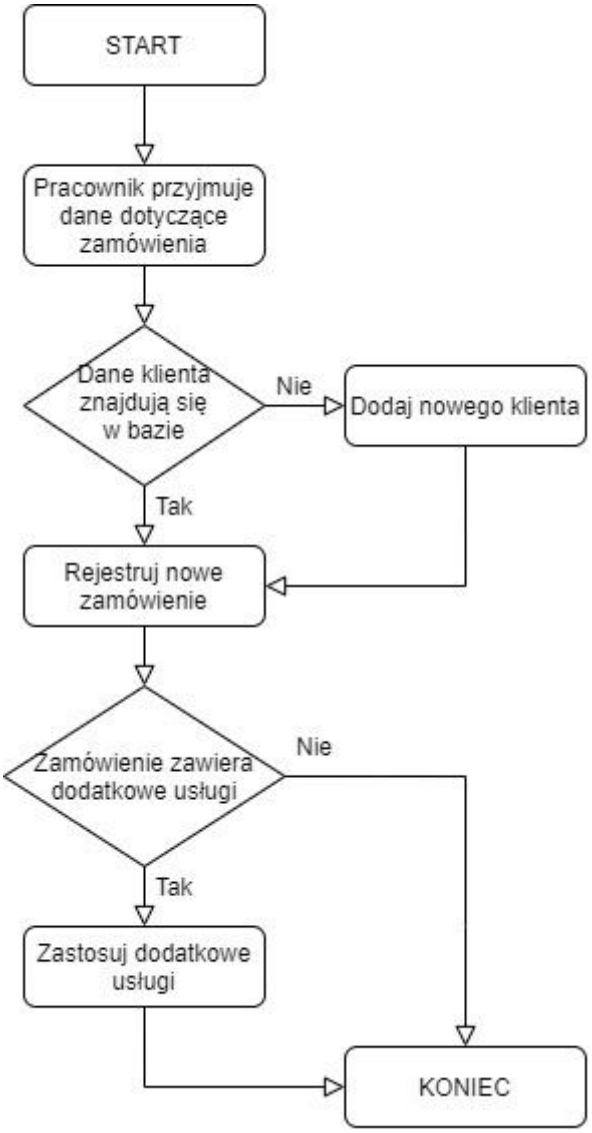
4.2 Model logiczny



5. Scenariusze

5.1 Scenariusz generowania raportu

5.2 Scenariusz przyjęcia zamówienia



6. Specyfikacja wewnętrzna

6.1 Struktura projektu

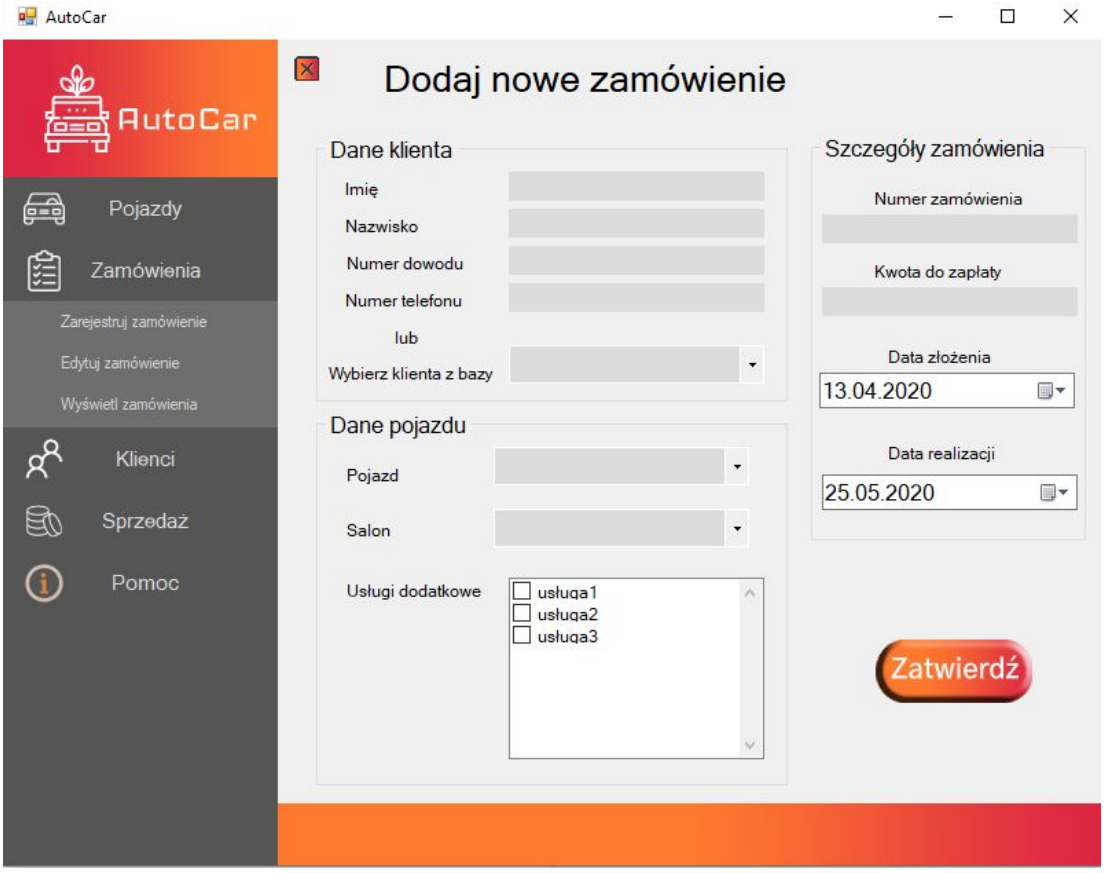
Baza danych zaprojektowana w SQL, podłączona do formatki stworzonej w Windows Forms.

6.2 Funkcjonalności

Poniżej zostały opisane dwie funkcjonalności z programu.

6.2.1 Przyjęcie zamówienia:

Aby przyjąć zamówienie trzeba być zalogowanym na konto z tymi uprawnieniami. Po zalogowaniu po lewej strony z listy wybieramy zakładkę “Zamówienia” i następnie pole “Zarejestruj zamówienie”. (rys 1.1)



rys 1.1

Aby dodać zamówienie należy sprawdzić czy dany klient występuje już w bazie (rys 1.2) , czyli sprawdzamy tabele Clients; odpowiednio kolumna FirstName i LastName. Do tego służy metoda FillComboBoxes, która tworzy zmienną clients przyjmującą dane z tabeli Clients, następnie w pętli do comboBoxCustomer dla każdego wiersza w tabeli wypisuje zawartość kolumn FirstName oraz LastName.

```
var clients = databaseContext.Set<Clients>();
        if(clients.Any())
        {
            foreach (Clients client in clients)
            {
                comboBoxCustomer.Items.Add(client.FirstName + " " + client.LastName);
            }
        }
```

Fragment kodu odpowiadający za wypełnienie comboboxa.

rys 1.2

W przeciwnym razie jeśli klient nie istnieje w bazie dodajemy go do bazy wypełniając wszystkie pola z danymi w tabeli Clients: Imię (FirstName), Nazwisko (LastName), Numer dowodu (DocumentNumber) oraz numer telefonu (PhoneNumber).

	Id	FirstName	LastName	PhoneNumber	DocumentNu...
▶	1	Adam	Lipka	590800744	4
	2	Krzysztof	Lipka	711003438	3
*	NULL	NULL	NULL	NULL	NULL

Dane istniejące w bazie

Wybieramy pojazd z dostępnych już pojazdów (rys 1.3). FillComboBoxes to metoda odpowiedzialna również za uzupełnienie comboBoxa z dostępnymi pojazdami oraz salonem. Działa ona w sposób następujący: tworzy zmienną cars przyjmującą dane z tabeli Cars, następnie w pętli za pomocą zmiennych “model” oraz “brand” uzyskujemy dostęp do danych w tabelach Models i CarBrands. Do comboBoxCar dla każdego wiersza w tabeli wypisuje odpowiednio ID, nazwę marki, model, rok produkcji oraz rodzaj nadwozia danego samochodu. W podobny sposób do comboboxa dodawana jest lista salonów (rys 1.4).

rys 1.3

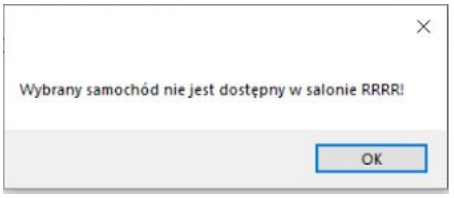
rys 1.4

```
var cars = databaseContext.Set<Cars>().Include(c => c.Model).Include(c => c.CarShowroom);
        foreach (Cars car in cars)
        {
            var model = databaseContext.Models.Include(m => m.Brand).First(m => m == car.Model);
            var brand = databaseContext.CarBrands.First(b => b == model.Brand);
            comboBoxCar.Items.Add(
                car.Id.ToString() + ". " + brand.Name + " " + model.Name + ", " + car.ProductionYear + " " +
car.Body);
        }

        var showrooms = databaseContext.Set<CarShowrooms>();
        foreach (CarShowrooms showroom in showrooms)
        {
            comboBoxShowroom.Items.Add(showroom.Name);
        }
```

Fragment kodu odpowiedzialny za wypełnienie comboboxa.

Dalszym etapem jest wybór dodatkowych usług, które klient sobie zażyczył. Usługi znajdują się w CheckedListBox. Wypełniamy również pole z kwotą do zapłaty. Podajemy datę złożenia oraz datę realizacji . Klikamy przycisk zatwierdź. Odpowiedzialna za to funkcja buttonSave_Click(object sender, EventArgs e). W funkcji sprawdzane jest za pomocą funkcji if czy wybraliśmy klienta z bazy czy był to nowy klient. Jeśli klient był nowy to do tabeli Clients dodajemy nowy wiersz z danymi z textboxów. Następnie sprawdzamy czy wybrany przez klienta pojazd znajduję się w wybranym salonie, jeśli nie dostaniemy komunikat o błędzie (rys 1.5).



rys 1.5

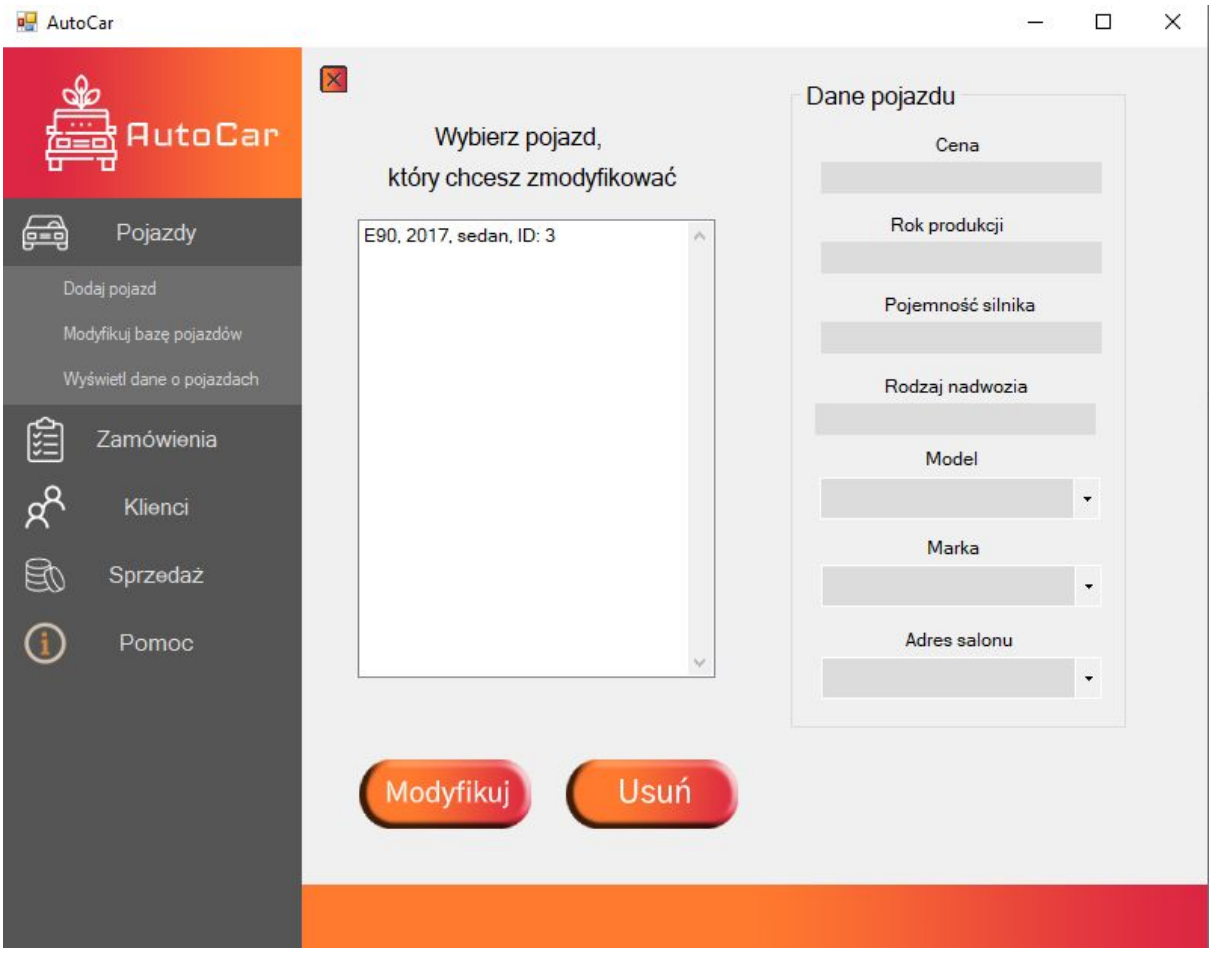
W kolejnym ifie sprawdzamy czy data realizacji zamówienia nie jest wcześniej niż data złożenia zamówienia, jeśli jest dostaniemy komunikat: “Wybrana data realizacji zamówienia jest błędna!” Następnie dodajemy wybrane przez klienta usługi i tabelę Orders uzupełniamy danymi: Cars, Client, Price, OrderDate, RealizationDate, AdditionalServices. Jeśli klient nie wybrał dodatkowych usług pole AdditionalServices pozostaje puste. Zapisujemy zmiany i zamówienie zostaje przyjęte.

	Id	Price	OrderDate	RealizationDate	AdditionalServ...	EmployeeId	ClientId	CarId
▷	1	34000	13.04.2020 16:3...	25.05.2020 14:3...	usługa1,	*NULL*	1	3

Dane zapisane do bazy

6.2.2 Modyfikacja bazy pojazdów:

Aby zmodyfikować bazę pojazdów trzeba być zalogowanym na konto z tymi uprawnieniami. Po zalogowaniu po lewej strony z listy wybieramy zakładkę “Pojazdy” i następnie pole “Modyfikuj bazę pojazdów”. (rys 2.1)



rys 2.1

Widzimy listę pojazdów, i wybieramy pojazd który chcemy zmodyfikować lub usunąć. Za wyświetlanie listy pojazdów odpowiada funkcja public EditCarForm() która wyświetla następujące informacje o samochodzie: Model(Models), Rok produkcji(Cars), Rodzaj nadwozia(Cars), ID samochodu(Cars). Po wyborze pojazdu po prawej stronie wyświetlają nam się szczegółowe informacje o samochodzie: Cena(Cars), Rok produkcji(Cars), Pojemność silnika(Cars), Rodzaj nadwozia(Cars), Model(Models), Marka(Brands), Adres Salonu(CarShowrooms). Za wyświetlanie tych informacji odpowiedzialna jest funkcja *listBoxCars_SelectedIndexChanged(object sender, EventArgs e)* Każde z pól możemy edytować zmieniając wartości i zatwierdzając za pomocą przycisku Modyfikuj. Służy d otego metoda *buttonModify_Click(object sender, EventArgs e),* która nadpisuje dane w tabeli Cars nowo podanymi wartościami, wszystkie pola muszą zostać wypełnione, w przeciwnym wypadku użytkownik ujrzy MassageBox'a "Nie wszystkie wymagane pola zostały uzupełnione!" "Błąd modyfikowania pojazdu." Wciśnięcie przycisku Modyfikuj bez wybranego pojazdu skutkuje wyświetleniem się informacji Nie wybrano pojazdu do modyfikacji!", "Błąd modyfikowania pojazdu."

Usunięcie pojazdu jest niemożliwe, jeżeli został on przypisany do realizowanego obecnie zamówienia, gdy będziemy próbować tak robić zostanie wyrzucony wyjątek. Metoda *buttonDelete_Click(object sender, EventArgs e)* usuwa dany pojazd z tabeli Cars.

	Id	ProductionYear	Body	Price	EngineCapacity	ModelId	CarShowroomId
▷	3	2017	sedan	30000,00	2,00	3	1
◇	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Dane zapisane do bazy

7. Specyfikacja zewnętrzna

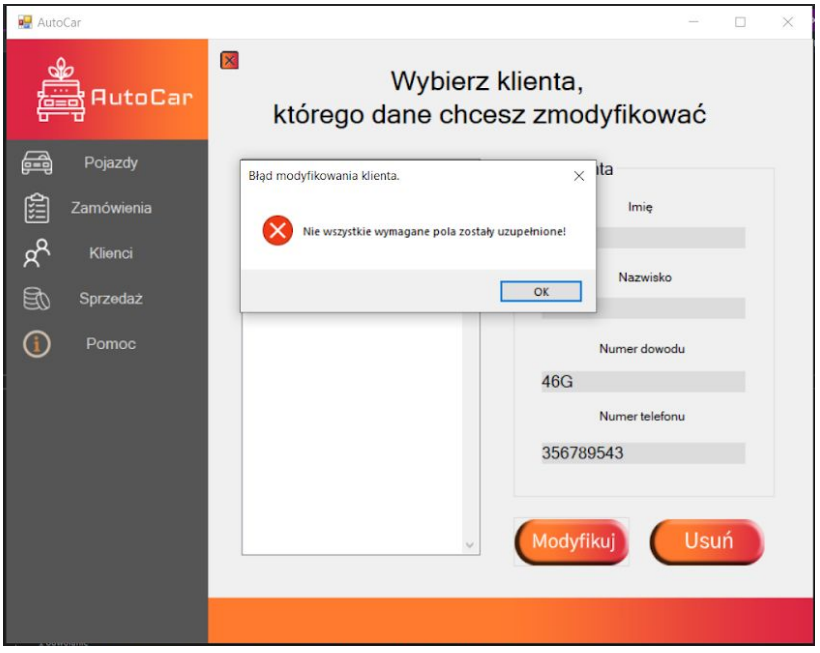
7.1. Sposób uruchomienia programu

Aby uruchomić system należy kliknąć dwukrotnie w ikonkę pliku exe. Następnie wyskoczy okno do logowania dostępne dla każdego pracownika salonu. W zależności od pełnionej funkcji, po zalogowaniu pojawi się odpowiedni panel sterowania.

Reszta programu jest intuicyjna, nie wymaga obszernego opisu. Dodatkowo istnieje zakładka “POMOC” zawierająca opis funkcjonalności, która może ułatwić pracę użytkownikowi.

7.2. Format danych wejściowych

Aby wprowadzać dane do serwera należy posiadać klawiaturę lub skorzystać z klawiatury ekranowej. W programie zostały wprowadzone wszelkie zabezpieczenia dotyczące formatu danych wejściowych, nie ma możliwości wpisania wyrazu w miejsca np ceny. O wszelkich błędach użytkownik zostaje powiadomiony po próbie zapisania wprowadzanych zmian.



Przykładowy komunikat o błędzie.

8. Raport z systemu

W systemie istnieje możliwość wygenerowania prostego raportu dotyczącego sprzedaży w postaci dokumentu PDF. Raport taki zawiera informacje o wszystkich zamówieniach w trakcie realizacji i zrealizowanych oraz podsumowanie zamówień złożonych w każdym z salonów. Na poniższym zdjęciu przedstawiona jest forma przykładowego raportu.

Zamówienia w trakcie realizacji:

6. Skoda Rapid, 2018, sedan - Kentucky, Dave

Suma zamówien w trakcie realizacji: 1

Zamówienia zrealizowane:

2. Audi A6, 2003, kombi - Freeman, Morgan

4. Opel Astra, 2017, sedan - Janowicz, Jan

5. Skoda Octavia, 2018, sedan - Kowalski, Adam

Suma zrealizowanych zamówien: 3

xD, xd Street 1

Zrealizowanych zamówien: 1

Zamówien w sumie: 1

Kwota sumaryczna zamówien: 120000

Random Showroom, Random Street 2

Zrealizowanych zamówien: 1

Zamówien w sumie: 1

Kwota sumaryczna zamówien: 45000

Salon abc, Ul. Salonowa 1

Zrealizowanych zamówien: 1

Zamówien w sumie: 2

Kwota sumaryczna zamówien: 102000

Przykładowy raport sprzedaży

9. Podsumowanie

Stworzony przez nas system obsługi salonu jest bardzo prosty w obsłudze. Wszystkie zakładki są czytelne i intuicyjne. W razie jakichkolwiek problemów użytkowania został dodany Help w postaci zakładki zawierającej opis funkcjonalności.

Pracę zaczęliśmy od zaprojektowania bazy danych co znacznie ułatwiło dalszą część pracy. Na podstawie przygotowanych diagramów stworzyliśmy formatkę dla użytkowników i administratorów. Ostatnim etapem było połączenie tych dwóch odrębnych części programu, co okazało się być najcięższym elementem w implementacji naszego systemu. Jednak pomimo licznych przeszkód udało nam się dopracować program. Finalnie zostały spełnione wszystkie początkowe założenia. Program został przetestowany na różnych komputerach i wymagania sprzętowe nie odgrywają żadnej roli w poprawności jego działania.