Maciej Karpiński, Tomasz Jakubiak

# Dokumentacja Techniczna

## Programu komputerowego – Komis Samochodowy „Janusz”

W skład niniejszej dokumentacji wchodzą:

**Cel oraz założenia projektu**

**Diagram UML – przypadki użycia**

**Model danych**

**Opis przypadków użycia**

**model interfejsu użytkownika**

**Diagram UML Sekwencji**

**Fragmenty kodów źródłowych i zapytań SQL**

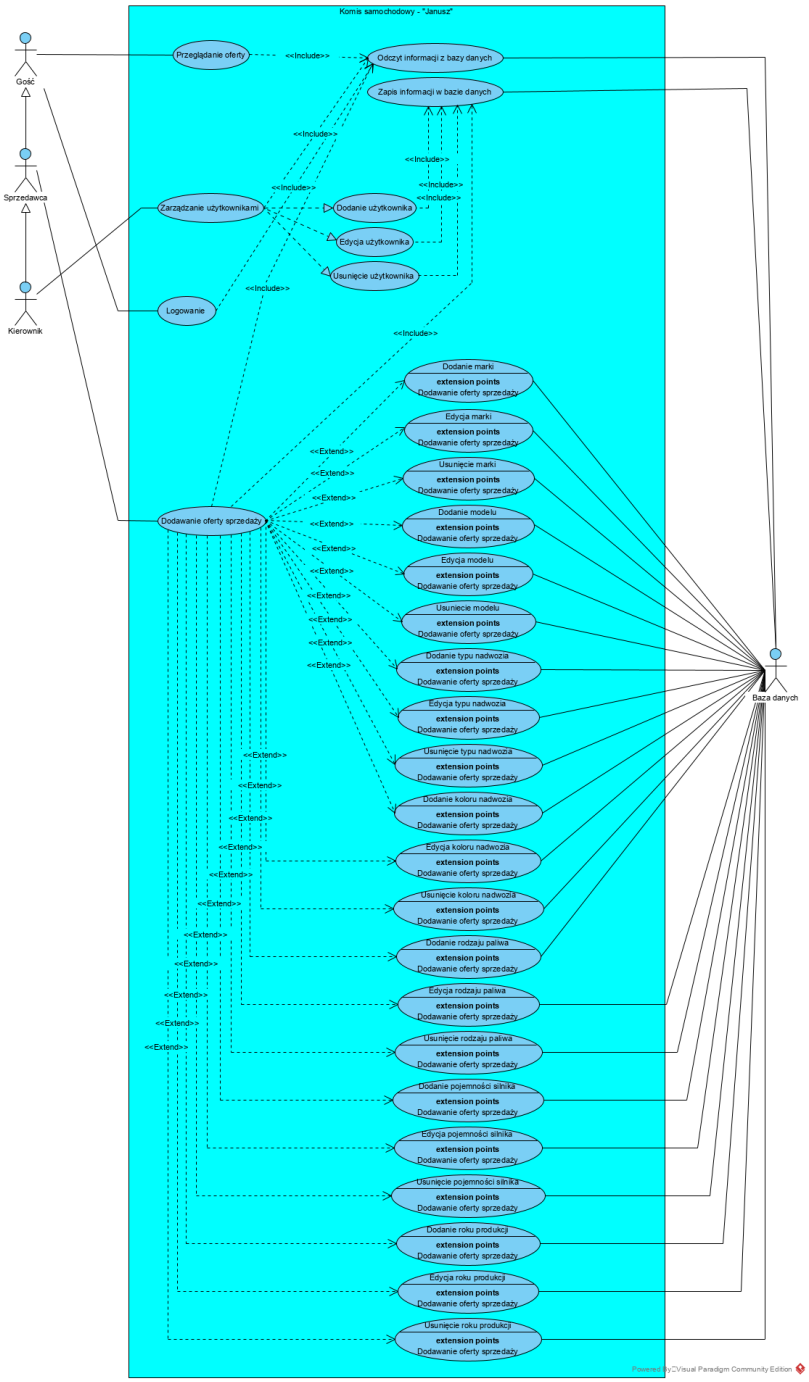
**Podsumowanie**

## Cel oraz założenia projektu

Celem stworzenia projektu jest zautomatyzowanie dodawania, przechowywania oraz wyświetlania ogłoszeń w komisie samochodowym „Janusz”. Najważniejsze założenia:

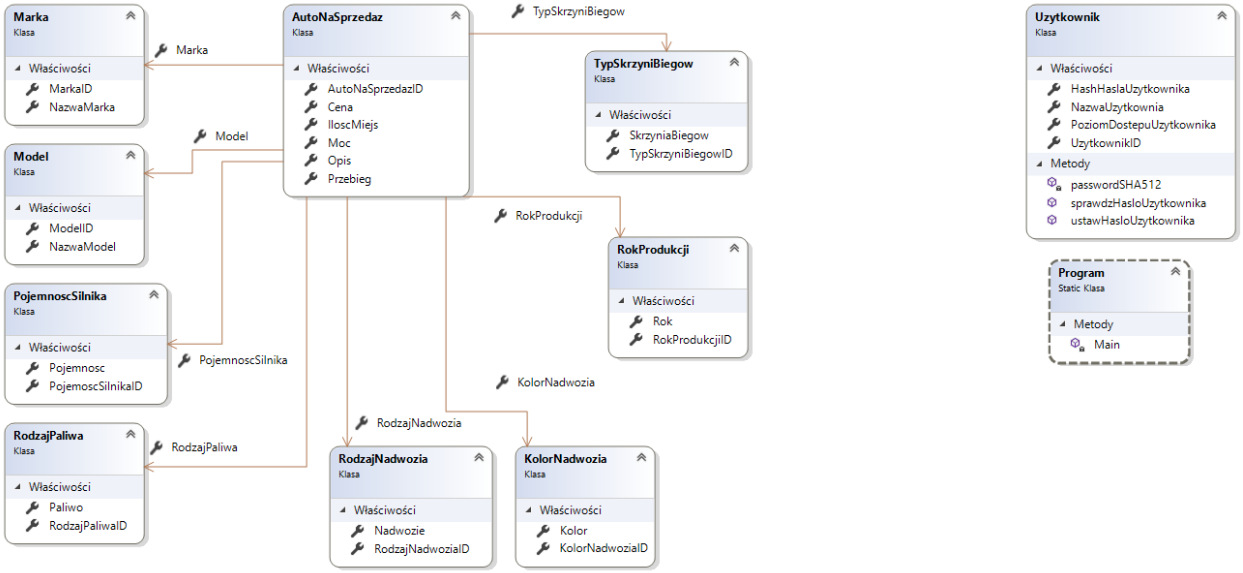
- Aplikacja będzie bardzo intuicyjna i przyjazna dla użytkownika,
- Dane będą przechowywane w lokalnej bazie danych SQL,
- Dodawanie ogłoszeń będzie uproszczone poprzez wybieranie z gotowych danych wprowadzonych do bazy, ale użytkownik będzie miał możliwość dodawania własnych danych,
- W aplikacji będzie można tworzyć nowych, edytować i usuwać istniejących użytkowników,
- Aplikacja będzie obsługiwać 3 typy kont z różnym poziomem dostępów,
- Aplikacja będzie szyfrować hasło użytkownika,
- Hasła użytkowników będą przechowywane w bazie danych w postaci zaszyfrowanej,
- Aplikacja przed rozpoczęciem pracy będzie weryfikować hasło i poziom dostępu do poszczególnych opcji,
- Zdjęcia dodawane do oferty będą zapisywane w ogólnodostępnej lokalizacji.

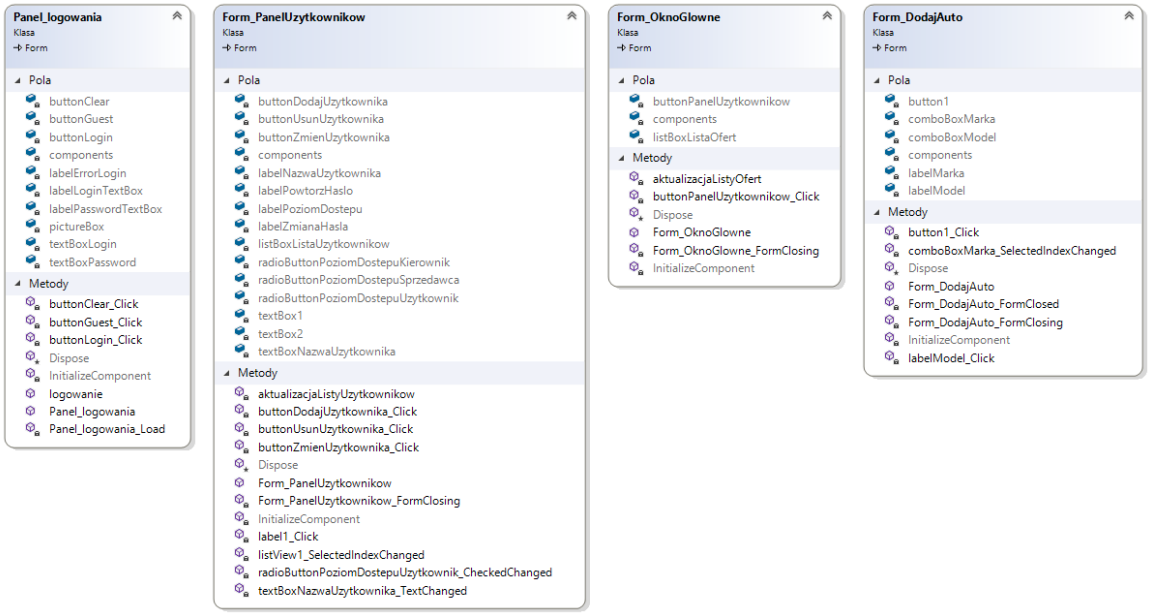
# Diagram UML – przypadki użycia

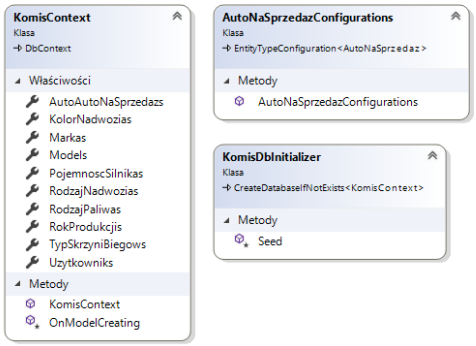


## Model danych

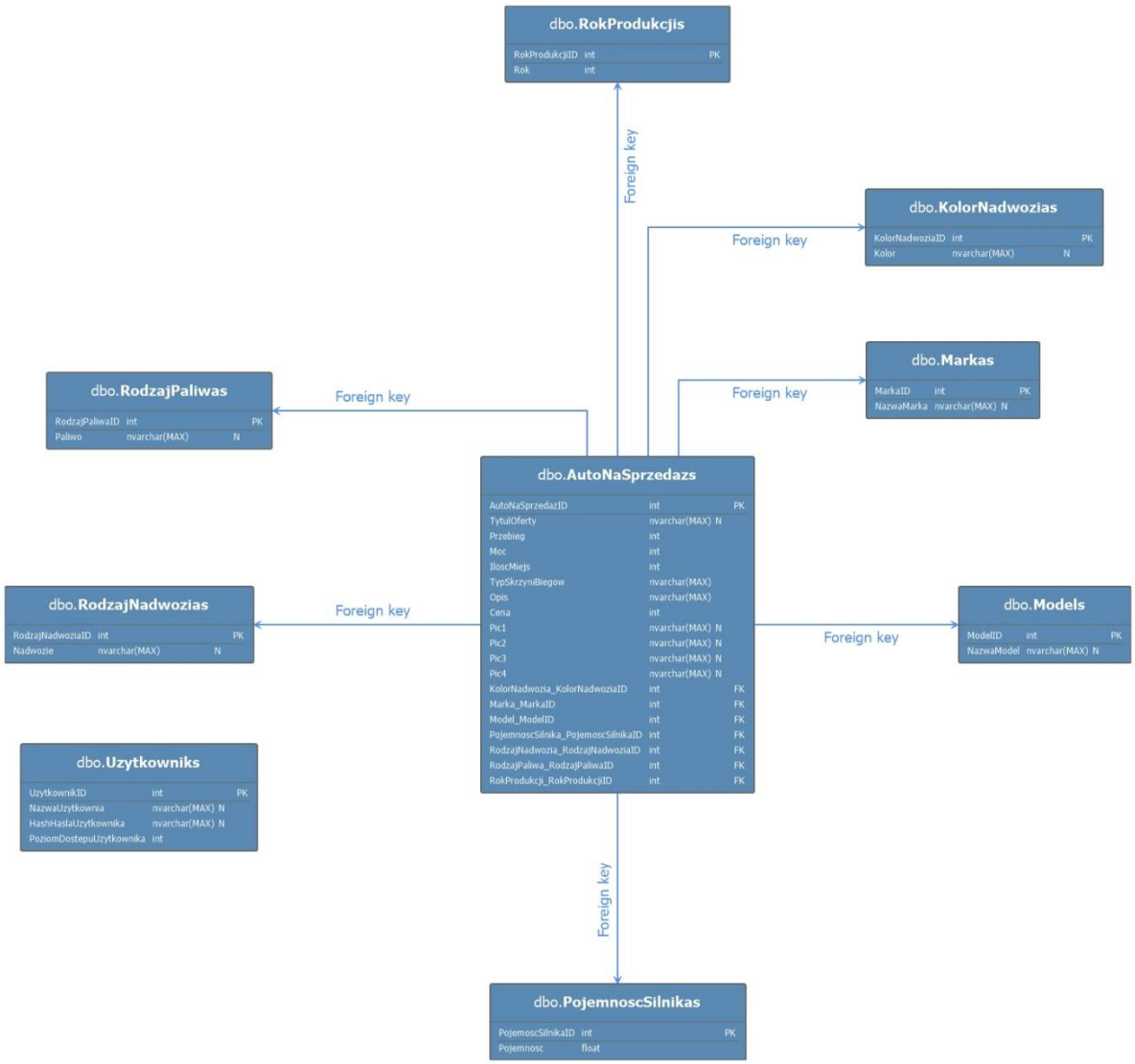
1. Diagram klas:



2. Diagram bazy danych:



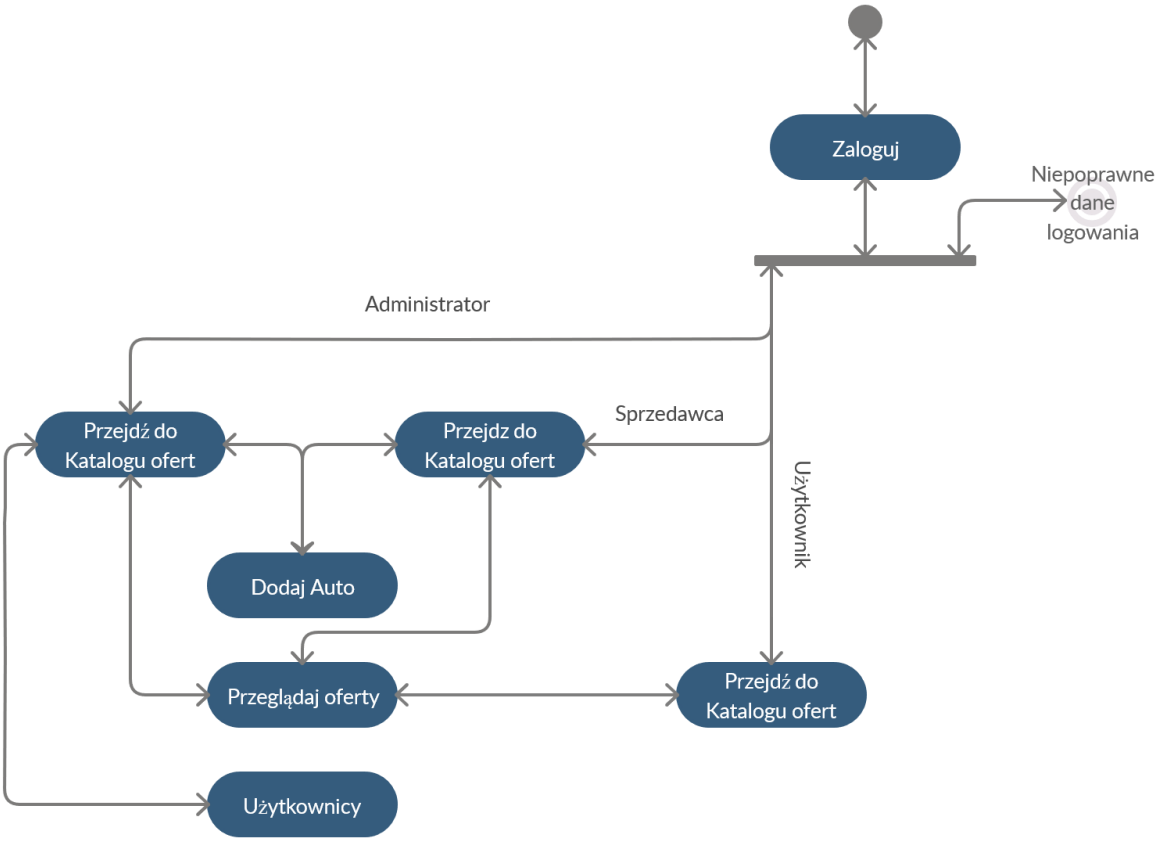
## Opis przypadków użycia

Proces logowania się użytkowników z różnymi uprawnieniami i możliwością korzystania z wybranych formularzy.

Program na początku posiada dwa konta : konto Administratora z pełnymi uprawnieniami oraz konto Gościa z uprawnieniami użytkownika.

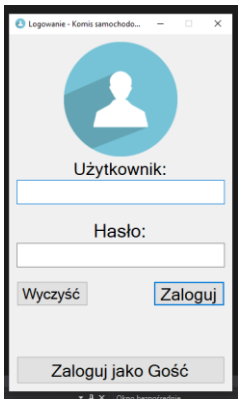
Program posiada 3 rodzaje uprawnień:

- Administrator – konto z najwyższymi uprawnieniami ma dostęp do wszystkich formularzy : Przeglądanie katalogu ofert , obsługa ofert oraz obsługa kont w systemie.
- Sprzedawca – konto posiadające dostęp do dwóch formularzy : Przeglądanie katalogu ofert oraz obsługa ofert.
- Użytkownik – konto posiadające tylko możliwość przeglądania ofert znajdujących się w systemie.

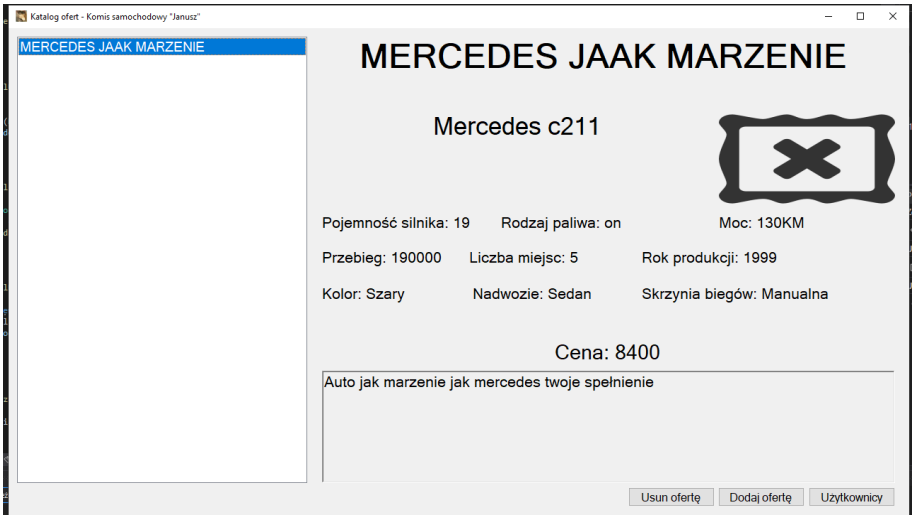


## Model interfejsu użytkownika

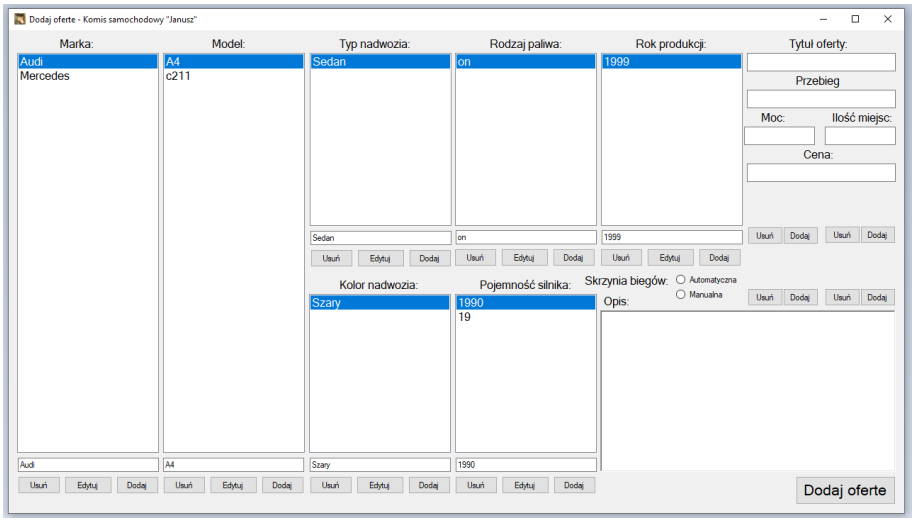
Panel logowania



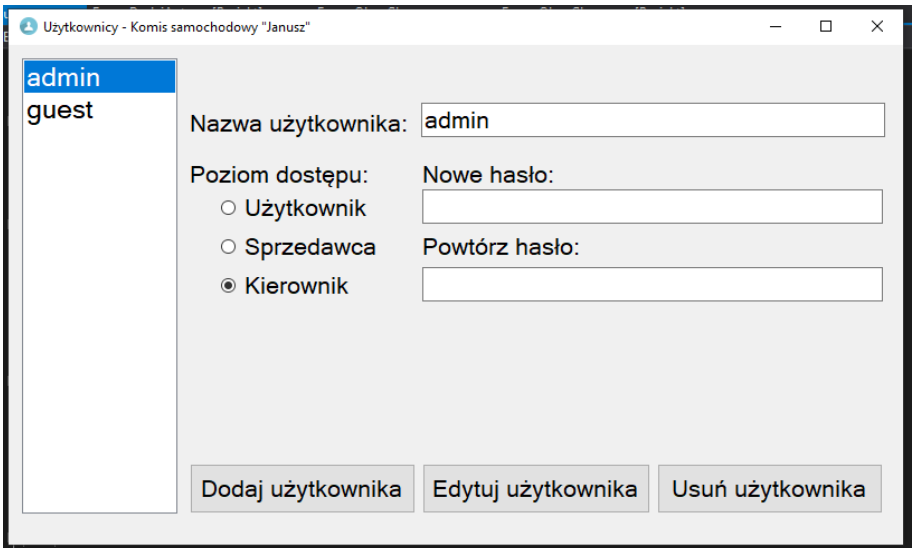

Panel ogłoszeń



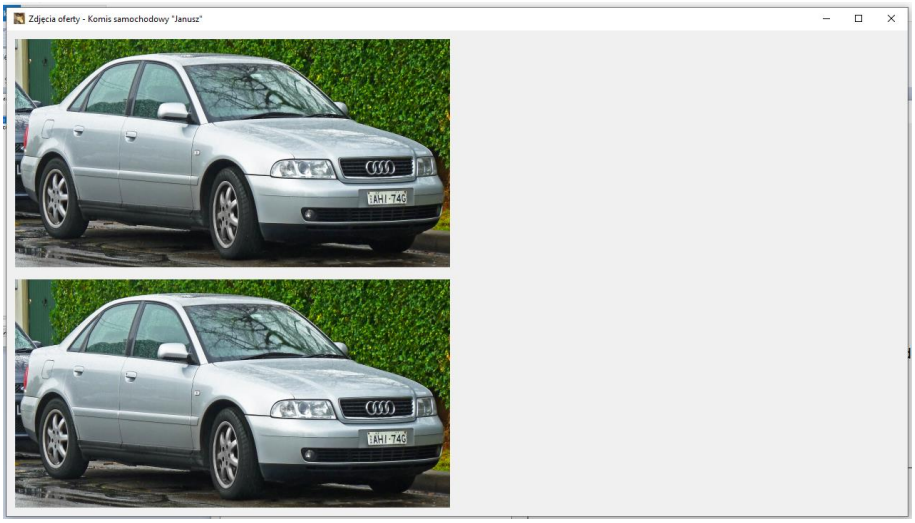

Panel dodawania nowego ogłoszenia



Panel zarządzania użytkownikami

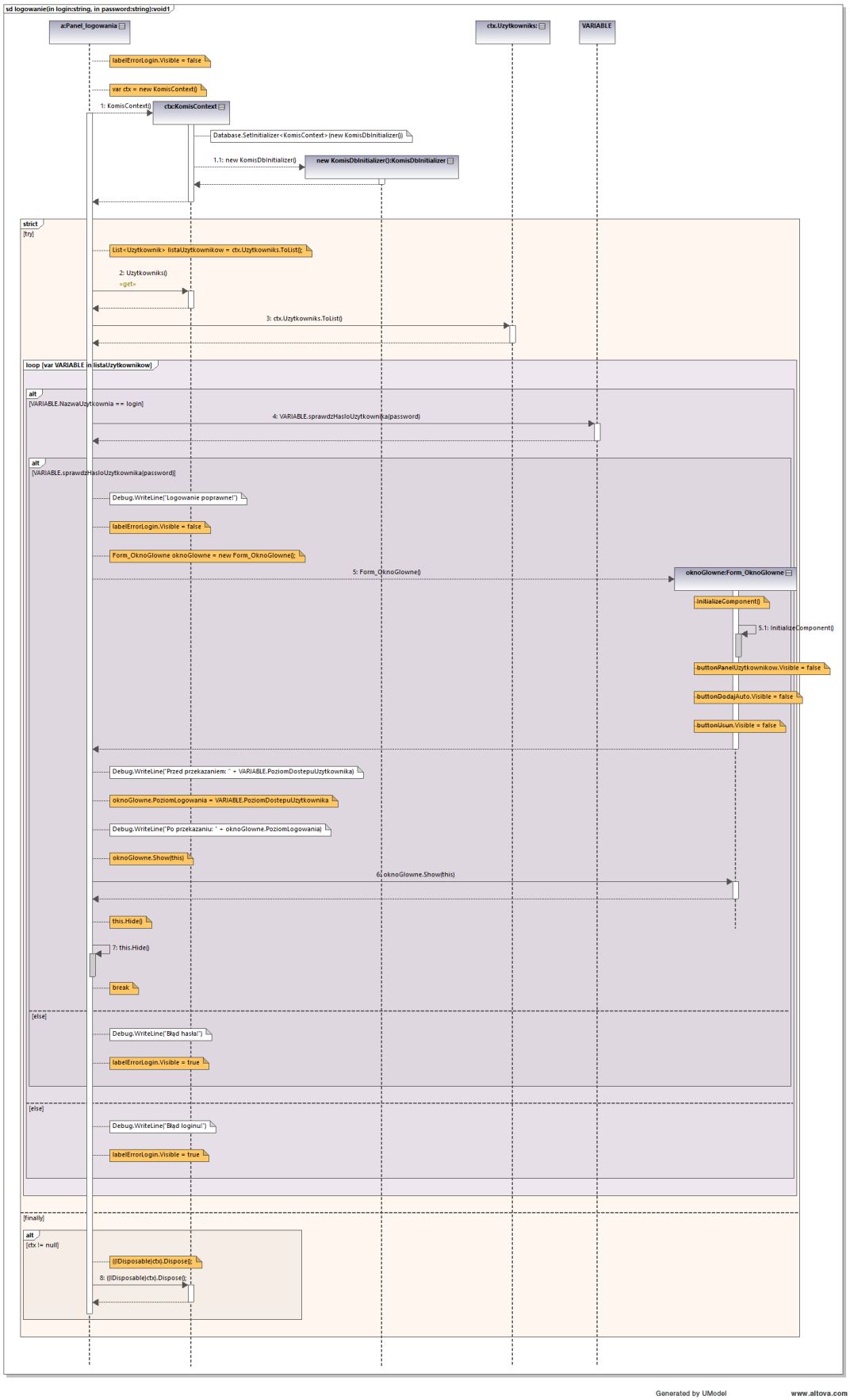



Panel przeglądania zdjęcie w ogłoszeniu

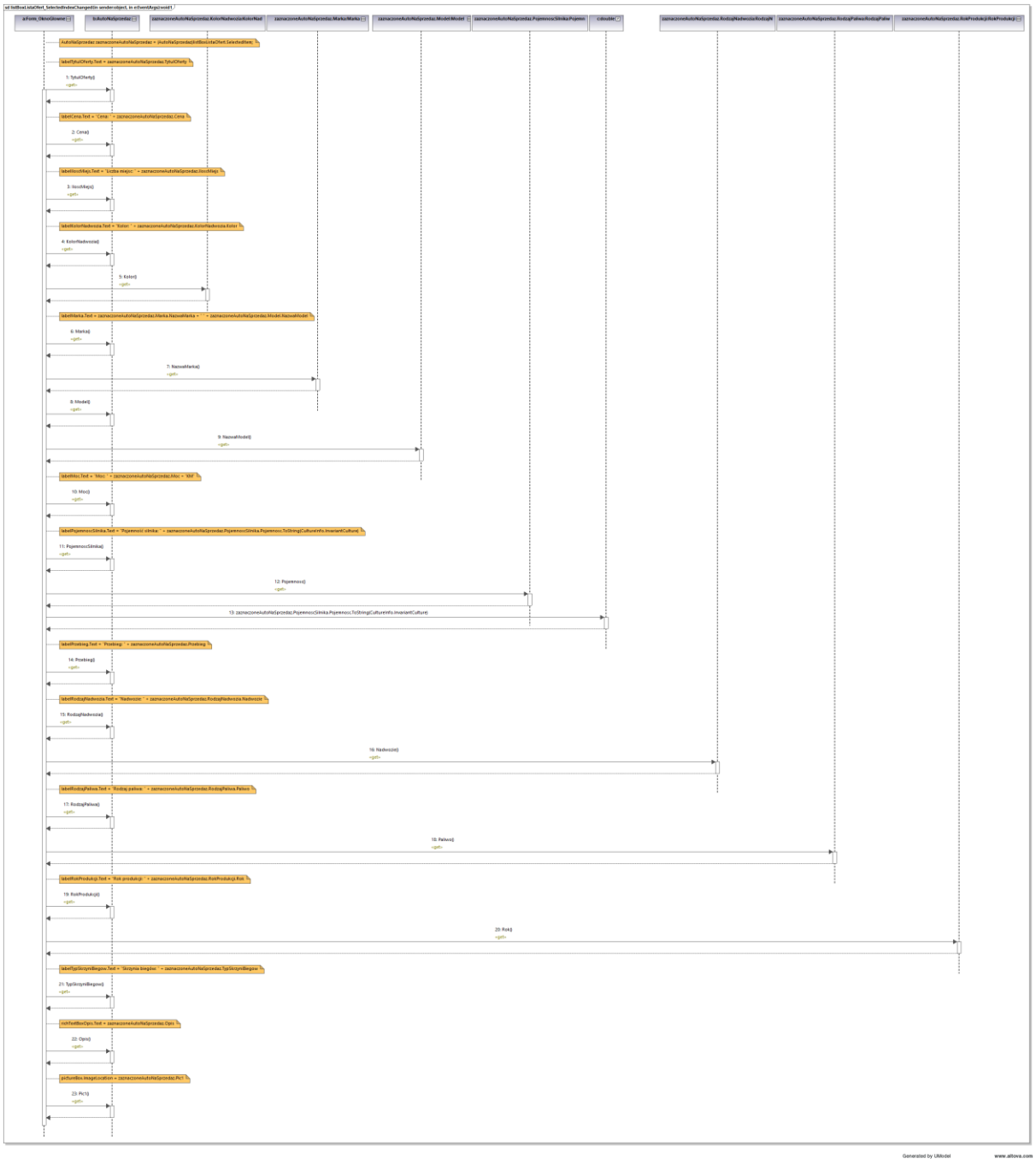


# Diagram UML Sekwencji - przykłady

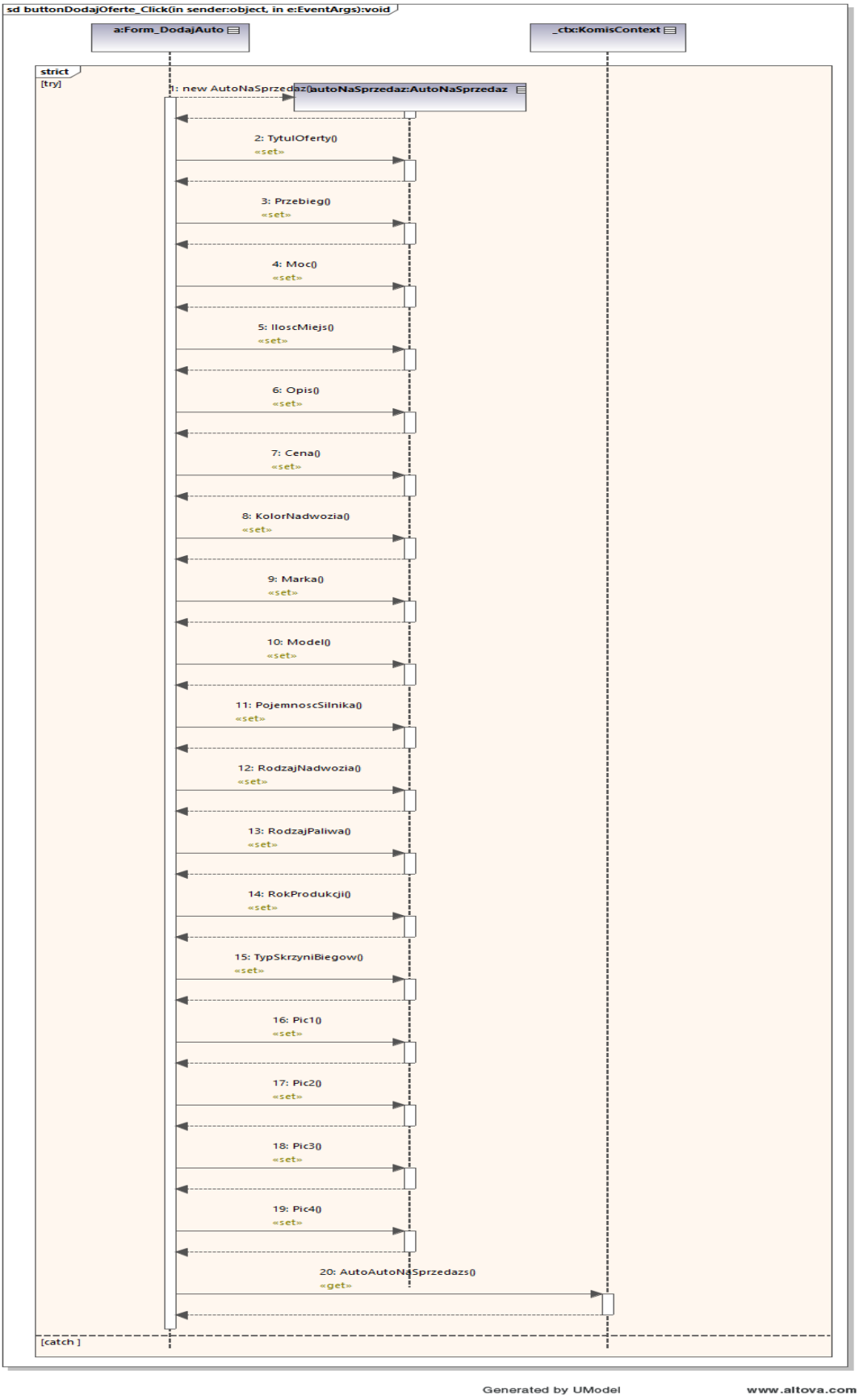
1. Logowanie do aplikacji:



2. Wyświetlanie aktualnej oferty:



3. Dodanie oferty do bazy danych:




Wszystkie diagramy znajdują się w repozytorium w katalogu „Dokumentacja”.

## Fragmenty kodów źródłowych i zapytań SQL

1. Do bazy danych wysyłane jest zapytanie sql, które pobiera listę wszystkich użytkowników a następnie sprawdza czy w bazie danych występują konta o nazwie: „admin” i „guest”.

```
-- Pobranie listy użytkowników:
SELECT

       [UzytkownikID],

       [NazwaUzytkownia],

       [HashHaslaUzytkownika],

       [PoziomDostepuUzytkownika]

FROM [dbo].[Uzytkowniks]
```

```
          //Sprawdzanie czy w bazie danych istnieją domyślni użytkownicy admin i guest
               List<Uzytkownik> listaUzytkownikow = ctx.Uzytkowniks.ToList();
               foreach (var VARIABLE in listaUzytkownikow)
               {
                   if (VARIABLE.NazwaUzytkownia == "admin")
                   {
                       adminExist = true;
                       break;
                   }
               }
               foreach (var VARIABLE in listaUzytkownikow)
               {
                   if (VARIABLE.NazwaUzytkownia == "guest")
                   {
                       guestExist = true;
                       break;
                   }
               }
               //W przypadku braku domyślnych użytkowników admin i guest dodaje ich do
              bazy danych
               if (adminExist == false)
               {
                   var admin = new Uzytkownik();
                   admin.NazwaUzytkownia = "admin";
                   admin.ustawHasloUzytkownika("admin");
                   admin.PoziomDostepuUzytkownika = 2;
                   ctx.Uzytkowniks.Add(admin);
                   ctx.SaveChanges();
               }
               if (guestExist == false)
               {
                   var guest = new Uzytkownik();
                   guest.NazwaUzytkownia = "guest";
                   guest.ustawHasloUzytkownika("guest");
                   guest.PoziomDostepuUzytkownika = 0;
                   ctx.Uzytkowniks.Add(guest);
                   ctx.SaveChanges();
               }
           }
```

2. Przy logowaniu następuje odczytanie poziomu dostępu i ustawienie widoczności odpowiednich przycisków oknie głównym zgodnie z pobranym poziomem dostępu.

```
        private void Form_OknoGlowne_Load(object sender, EventArgs e)
        {
            Debug.WriteLine("Okno główne, przyjęty parametr: " + PoziomLogowania);
            //przyjecie wartości z panelu logowania
            switch (PoziomLogowania)
            {
                case 0:
                    Debug.WriteLine("switch 0");
                    break;
                case 1:
                    Debug.WriteLine("switch 1");
                    buttonDodajAuto.Visible = true;
                    buttonUsun.Visible = true;
                    break;
                case 2:
                    Debug.WriteLine("switch 2");
                    buttonPanelUzytkownikow.Visible = true;
                    buttonDodajAuto.Visible = true;
                    buttonUsun.Visible = true;
                    break;
            }
```

3. Model ogłoszenia jest oparty na klasie AutoNaSprzedaz, na której są budowane zapytania do bazy danych obsługujące pobieranie listy ofert i dodawanie oferty sprzedaży.

```
public class AutoNaSprzedaz
    {
        public int AutoNaSprzedazID { get; set; }
        public string TytulOferty { get; set; }
        public virtual Marka Marka { get; set; }
        public virtual Model Model { get; set; }
        public virtual RokProdukcji RokProdukcji { get; set; }
        public virtual PojemnoscSilnika PojemnoscSilnika { get; set; }
        public virtual RodzajPaliwa RodzajPaliwa { get; set; }
        public int Przebieg { get; set; }
        public int Moc { get; set; }
        public virtual RodzajNadwozia RodzajNadwozia { get; set; }
        public int IloscMiejs { get; set; }
        public virtual KolorNadwozia KolorNadwozia { get; set; }
        public string TypSkrzyniBiegow { get; set; }
        public string Opis { get; set; }
        public int Cena { get; set; }
        public string Pic1 { get; set; }
        public string Pic2 { get; set; }
        public string Pic3 { get; set; }
        public string Pic4 { get; set; }
    }
```

```
-- Pobranie listy ofert:
SELECT

[AutoNaSprzedazID],

    [TytulOferty],

    [Przebieg],

    [Moc],

    [IloscMiejs],

    [TypSkrzyniBiegow],

    [Opis],

    [Cena],

    [Pic1],

    [Pic2],

    [Pic3],

    [Pic4],

    [KolorNadwozia_KolorNadwoziaID],

    [Marka_MarkaID],

    [Model_ModelID],

    [PojemnoscSilnika_PojemoscSilnikaID],

    [RodzajNadwozia_RodzajNadwoziaID],

    [RodzajPaliwa_RodzajPaliwaID],

    [RokProdukcji_RokProdukcjiID]

FROM [dbo].[AutoNaSprzedazs]
```

```
-- Dodanie oferty sprzedaży:

INSERT INTO [dbo].[AutoNaSprzedazs]([TytulOferty], [Przebieg], [Moc], [IloscMiejs],
[TypSkrzyniBiegow], [Opis], [Cena], [Pic1], [Pic2], [Pic3], [Pic4],
[KolorNadwozia_KolorNadwoziaID], [Marka_MarkaID], [Model_ModelID],
[PojemnoscSilnika_PojemoscSilnikaID], [RodzajNadwozia_RodzajNadwoziaID],
[RodzajPaliwa_RodzajPaliwaID], [RokProdukcji_RokProdukcjiID])

VALUES ('TytulOferty', 1234567890, 65, 3, 'Manual', 'Opis oferty', 54321,
'C:\Users\Public\Documents\Komis Samochodowy - Janusz\img\637257407240750501.jpg',
'C:\Users\Public\Documents\Komis Samochodowy - Janusz\img\637257407240750501.jpg',
'C:\Users\Public\Documents\Komis Samochodowy - Janusz\img\637257407240750501.jpg',
'C:\Users\Public\Documents\Komis Samochodowy - Janusz\img\637257407240750501.jpg', 1, 1,
1, 2, 1, 1, 1)
```

4. Klasa “Marka” wykorzystywana jest do przechowywania nazwy Marki w bazie danych. Dodawanie, edycja i usunięcie informacji dokonuje się poprzez odpowiednie zapytania SQL

```
public class Marka
    {
        [Key]
        public int MarkaID { get; set; }
        public string NazwaMarka { get; set; }
    }
```

```
-- Dodanie marki auta:
INSERT INTO [dbo].[Markas]

VALUES ('Mazda')
```

```
-- Aktualizacja marki auta:
UPDATE [dbo].[Markas]

SET [NazwaMarka] = 'Subaru'

WHERE ([MarkaID] = 3)
```

```
-- Usuniecie marki auta:

DELETE [dbo].[Markas]

WHERE ([MarkaID] = 3)
```

Analogicznie wyglądają klasy „Model”, „RokProdukcji”, „PojemnoscSilnika”, „RodzajPaliwa”, „RodzajNadwozia” i „KolorNadwozia” oraz ich zapytania SQL.

5. Za model użytkownika komisu odpowiedzialna jest klasa „Uzytkownik”.

```
public class Uzytkownik
    {
        [Key]
        public int UzytkownikID { get; set; }
        public string NazwaUzytkownia { get; set; }
        public string HashHaslaUzytkownika { get; private set; }
        public int PoziomDostepuUzytkownika { get; set; }
```

Klasa zawiera także metody odpowiedzialne za szyfrowanie hasła użytkownika przy pomocy algorytmu SHA-512.

```
//szyfrowanie hasła przy pomocy SHA512
private string passwordSHA512(string haslo)
{
    var bytes = System.Text.Encoding.UTF8.GetBytes(haslo);
    using (var hash = System.Security.Cryptography.SHA512.Create())
    {
        var hashedInputBytes = hash.ComputeHash(bytes);

        var hashedInputStringBuilder = new System.Text.StringBuilder(128);
        foreach (var b in hashedInputBytes)
        {
            hashedInputStringBuilder.Append(b.ToString("X2"));
        }

        return hashedInputStringBuilder.ToString();
    }
}
```

Do dodawania, aktualizowania i usuwania danych użytkowników wykorzystane zostały zapytania SQL.

```
-- Dodanie użytkownika:
INSERT INTO [dbo].[Uzytkowniks]
VALUES ('Sprzedawca',
'889C55911084DA2C4177BD991D67328757EC50F48E34166A227C208A5434094B216BBFC27A753408823B842
54B2BF3F07A05B805018F371C056C96CDC8654297', 1)
```

```
-- Aktualizacja użytkownika (na przykładzie zmiany hasła):
UPDATE [dbo].[Uzytkowniks]
SET [HashHaslaUzytkownika] =
'4DD2FD3D44E3B80DDB7E59152BB5219F8AB8A8C6F949B6A6A11650E89D7AD7853A48062CC45BF7D895813CD
3AA6F34BC63D188652D0A87C9D05FEA825DD68D7B'
WHERE ([UzytkownikID] = 3)
```

```
-- Usunięcie użytkownika:
DELETE [dbo].[Uzytkowniks]
WHERE ([UzytkownikID] = 3)
```

6. Wciśnięcie przycisku „Dodaj” znajdującym się przy polu zdjęcia, w formularzu tworzenia oferty wywołuje metodę WyborZdjecia(), która zwraca obiekt typu string. Funkcja kopiuje wybrane zdjęcie do ogólnodostępnej lokalizacji „C:\Users\Public\Documents\Komis Samochodowy - Janusz\img” pod zmienioną unikalną nazwą(zapisana w obiekcie typu string), którą zwraca do obiektu „PictureBox”.

```
private void buttonDodajZdjecie1_Click(object sender, EventArgs e)
       {
           pictureBoxZdjecie1.ImageLocation = WyborZdjecia();
       }
```

```
private string WyborZdjecia()
        {

            string targetImagePath = Environment.GetFolderPath(Environment.SpecialFolder
.CommonDocuments) + @"\Komis Samochodowy - Janusz\img\";
            string uniqueFileName = defaultImage;

            //wybiera zdjęcie
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.InitialDirectory = Environment.GetFolderPath(Environment.Spec
ialFolder.Desktop);
            openFileDialog.Filter = "Zdjęcia(*.jpg, *jpeg, *.bmp, *.png) | *.jpg; *.jpeg
; *.bmp; *.png";

            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                string sourceImagePath = openFileDialog.FileName;
                //Kopiuje zdjęcie do katalogu common documents\Komis Samochodowy -
 Janusz\img\ i nadaje mu uniklną nazwe
                uniqueFileName = targetImagePath + string.Format(@"{0}" + Path.GetExtens
ion(openFileDialog.FileName), DateTime.Now.Ticks);
                File.Copy(sourceImagePath, uniqueFileName);
            }
            //zwraca sciezka dostepu do pliku
            return uniqueFileName;
        }
```

## Podsumowanie

Program został wykonany w Visual Studio 2019, przy użyciu języka programowania C# oraz bibliotek EntityFramework i BouncyCastle, jako serwer bazy danych został wykorzystany Microsoft SQL Server 2019 w wersji Express. Wszystkie założenia zostały spełnione zgodnie z wytycznymi.

Kod źródłowy aplikacji oraz wszystkie diagramy znajdują się w repozytorium dostępnym pod adresem: https://github.com/Karpfly2822/Projekt-PSBD