LABORATORIUM SYSTEMÓW MOBILNYCH

STWORZENIE PRZYKŁADOWEJ APLIKACJI NA URZĄDZENIE POCKET PC

I. Temat ćwiczenia
Stworzenie przykładowej aplikacji na urządzenie Pocket PC

II. Wymagania
Wiadomości z poprzednich zajęć

III. Ćwiczenie

1. Wprowadzenie

Celem ćwiczenia jest stworzenie mobilnej aplikacji rejestracji pojazdów w serwisie samochodowym. Aplikacja powinna spełniać następujące wymagania:

- Aplikacja mobilna dla urządzenia Pocket PC,
- Baza danych Microsoft SQL Server Mobile Edition,
- Baza danych pojazdów przyjętych do serwisu,
- Baza danych pracowników – serwisantów,
- Rejestr napraw przeprowadzanych przez pracowników na samochodach,
- Dodanie pojazdu do bazy danych,
- Dodanie pracownika do bazy danych,
- Dodanie naprawy do bazy danych.

2. Utworzenie aplikacji

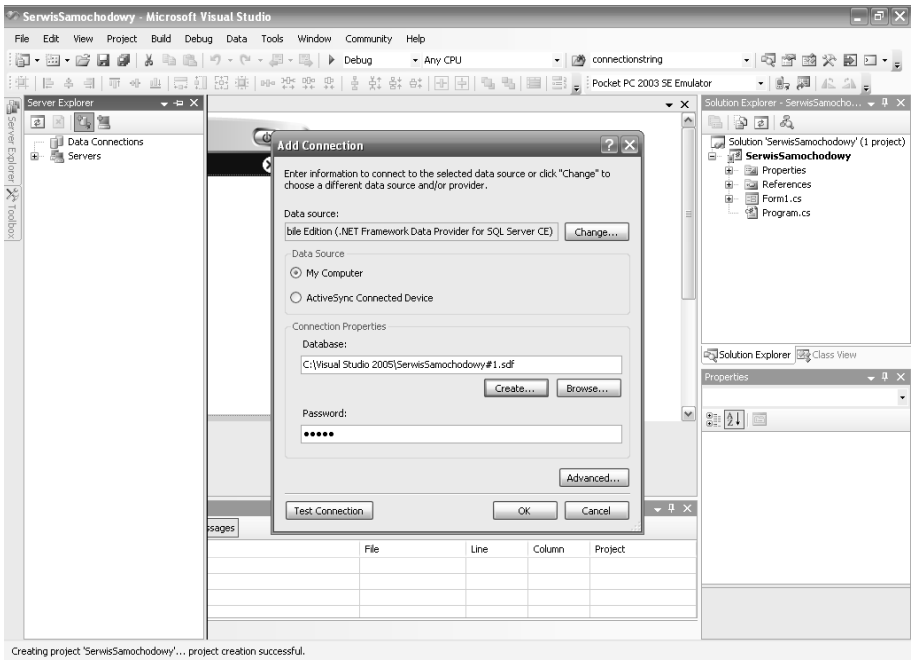
W celu wykonania powyższego zadania wykorzystamy wiedzę zdobytą podczas poprzednich lekcji. Na początku należy utworzyć aplikację na urządzenie Pocket PC z wykorzystaniem bazy danych Microsoft SQL Server Mobile Edition.
Tworzymy następujący schemat bazy danych:

Pojazdy
id
* numer_rej

Naprawy
id
* id_pojazdu
* id_mechanika
* data
* status

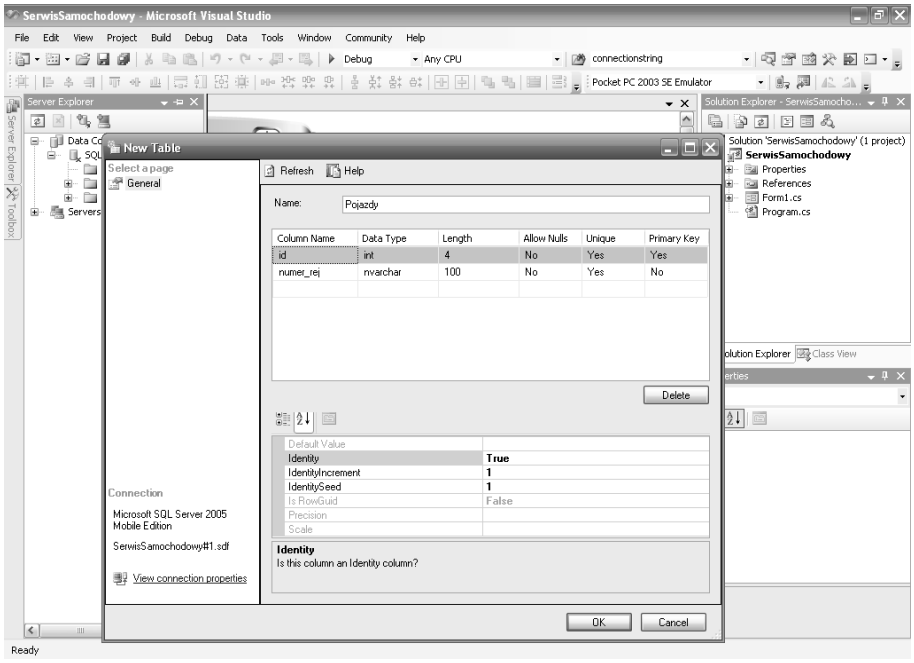
Mechanicy
id
* nazwisko

Z zakładki **Server Explorer** wybieramy **Connect To Database** i w wyświetlonym oknie dialogowym w polu **Data Source** wybieramy bazę danych **Microsoft SQL Server Mobile Edition** (przycisk **Change**). Następnie tworzymy nową bazę danych przyciskiem **Create**. Po określeniu ścieżki do bazy danych oraz hasła dostępu, zatwierdzamy przyciskiem **OK** (Rysunek 1).



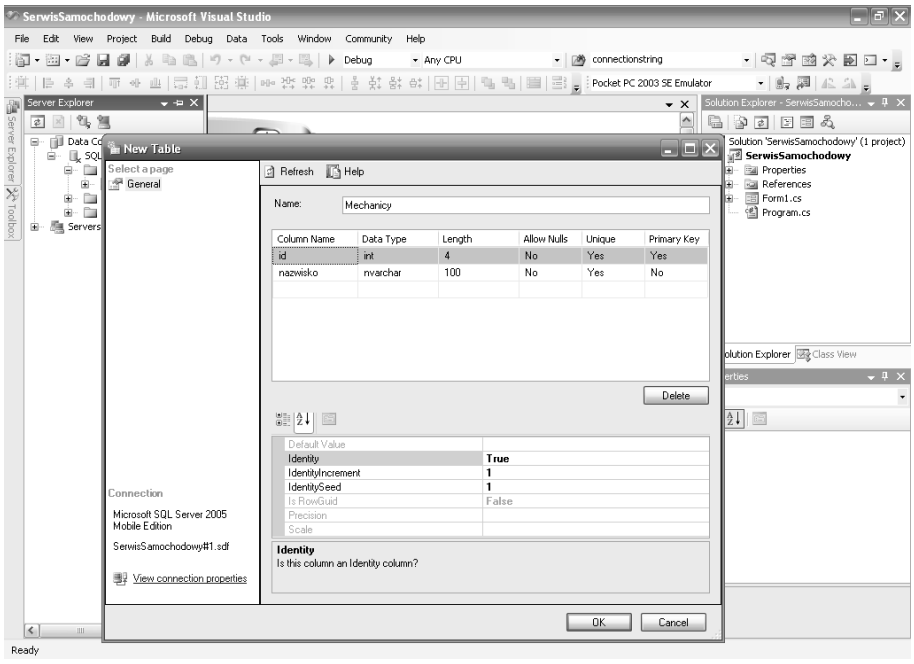
Rysunek 1 – Utworzenie połączenia do bazy danych

Tworzymy schemat bazy danych rozpoczynając od tabeli Pojazdy. Atrybut „*id*” definiujemy jako klucz podstawowy relacji oraz własność **Auto increment** poprzez ustawienie opcji **Identity** na **True** (Rysunek 2).

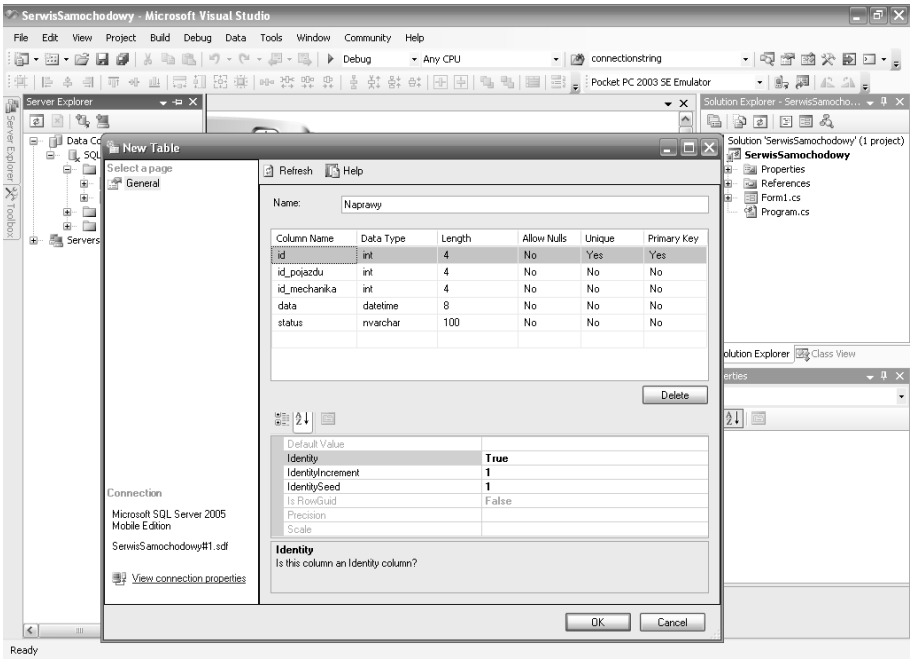


Rysunek 2 – Utworzenie schematu relacji Pojazdy

W podobny sposób definiujemy tabele Mechanicy oraz Naprawy. Dla atrybutu „*id*” definiujemy klucz podstawowy relacji oraz własność **Auto increment** (Rysunki 3 i 4).



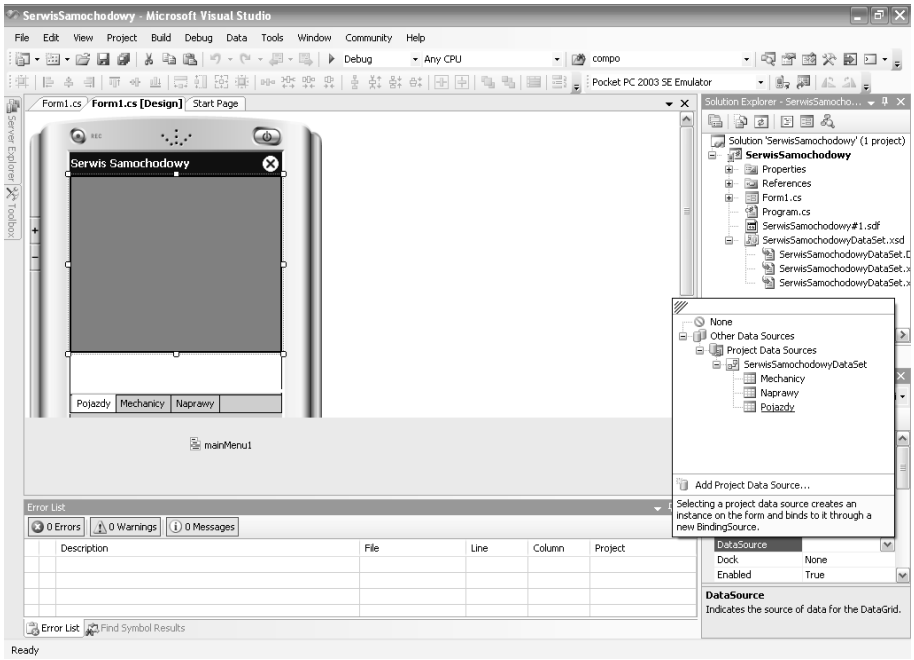
Rysunek 3 – Utworzenie schematu relacji Mechanicy



Rysunek 4 – Utworzenie schematu relacji Naprawy

Możemy przystąpić do utworzenia aplikacji na Pocket PC.

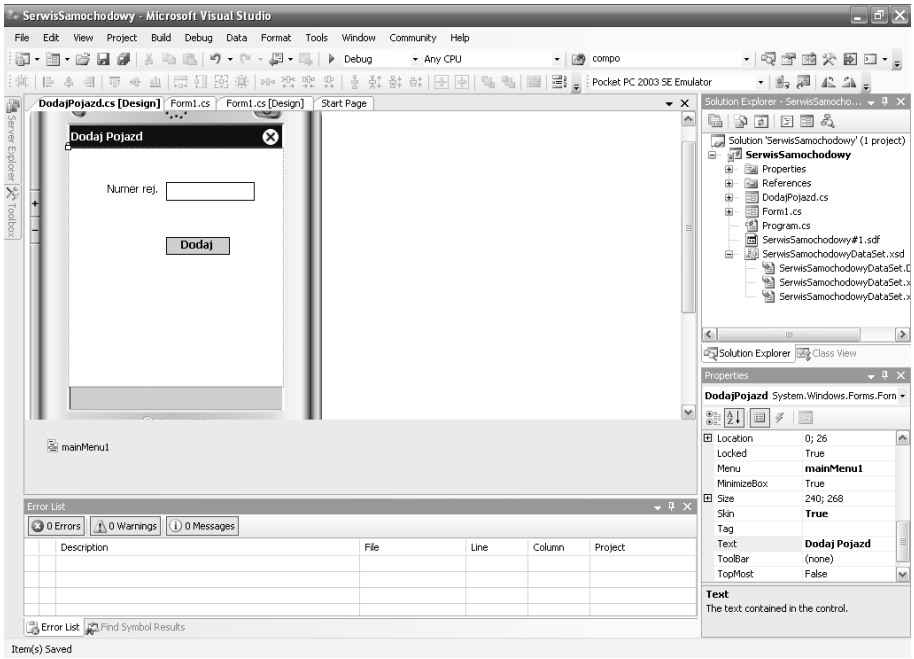
Wykorzystamy komponent **TabControl**, w którym na każdej z zakładek umieścimy osobny komponent **DataGrid** do obsługi poszczególnych relacji bazy danych. Na poszczególnych zakładkach umieszczamy komponenty **DataGrid** i określamy dla nich parametry **Data Source**. Wybieramy **Add Project Data Source**, wybieramy opcję **Database**, następnie wybieramy tabele **(Pojazdy, Mechanicy, Naprawy)**, które umieścimy w obiekcie **DataSet**. Zatwierdzamy wybór. Następnie z dostępnej listy wybieramy tabelę, z którą ma zostać połączony konkretny komponent **DataGrid** (Rysunek 5).



Rysunek 5 – Określenie źródła danych dla komponentu

Możemy teraz na każdej z zakładek tworzonej aplikacji umieścić przyciski do wprowadzania nowych użytkowników. Akcją na naciśnięcie przycisku będzie wyświetlenie nowej formy z polami do dodania nowego rekordu do bazy danych.

Dodawanie danych do bazy danych rozpoczynamy od utworzenia formatki służącej do dodawania nowego pojazdu. Po dodaniu nowej formatki, umieszczamy na nich komponenty: **Label**, **TextBox** oraz **Button**.
Schemat przykładowej formatki do dodawania pojazdu (Rysunek 6).



Rysunek 6 – Przykładowa formatka dodawania nowego pojazdu

Na głównej formie programu, na zakładce *Pojazdy*, dodajemy przycisk który wyświetli formę dodawania pojazdu.

Przykładowy kod otwarcia formatki dodawania pojazdu (Listing 1):

```
[…]
private void button1_Click(object sender, EventArgs e)
{
    DodajPojazd dp = new DodajPojazd(this.serwisDataSet, this.pojazdyTableAdapter);
    dp.ShowDialog();
    this.pojazdyTableAdapter.Fill(this.serwisSamochodowyDataSet.Pojazdy);
}
[…]
```

Listing 1 – Kod otwarcia formatki dodawania pojazdu

Przykładowy kod obsługi formatki dodawania nowego pojazdu do bazy danych (Listing 2):

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using SerwisSamochodowy.SerwisSamochodowyDataSetTableAdapters;

namespace SerwisSamochodowy
{
    public partial class DodajPojazd : Form
    {
        private SerwisSamochodowyDataSet serwisSamochodowyDataSet;
        private PojazdyTableAdapter pojazdyTableAdapter;

        public DodajPojazd(SerwisSamochodowyDataSet dataSet,
            PojazdyTableAdapter tableAdapter)
        {
            this.serwisSamochodowyDataSet = dataSet;
            this.pojazdyTableAdapter = tableAdapter;
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (!String.IsNullOrEmpty(this.textBox1.Text))
            {
                /*
                 * Utworzenie obiektu reprezentującego pojedynczą krotkę w bazie danych
                 */
                SerwisSamochodowyDataSet.PojazdyRow pojazd =
                    this.serwisSamochodowyDataSet.Pojazdy.NewPojazdyRow();

                /*
                 * Wypełnienie obiektu danymi z formatki
                 */
                pojazd.numer_rej = this.textBox1.Text;

                /*
                 * Aktualizacja obiektu DataSet
                 */
                this.serwisSamochodowyDataSet.Pojazdy.Rows.Add(pojazd);

                /*
                 * Aktualizacja zmian z bazie danych z wykorzystaniem obiektu DataAdapter
                 */
                if (this.pojazdyTableAdapter.Update(
                    (SerwisSamochodowyDataSet.PojazdyDataTable)
                    this.serwisSamochodowyDataSet.Pojazdy.GetChanges()) > 0)
                {
                    this.Close();
```

```
                }
            }
            else
            {
                MessageBox.Show("Wypełnij pole: Numer rej.");
            }
        }
    }
}
```

Listing 2 – Kod obsługi formatki dodawania pojazdu

W podobny sposób tworzymy formę do dodawania nowego mechanika. Przykładowy kod realizujący dodawanie nowego mechanika (Listing 3).

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using SerwisSamochodowy.SerwisSamochodowyDataSetTableAdapters;

namespace SerwisSamochodowy
{
    public partial class DodajMechanika : Form
    {
        private SerwisSamochodowyDataSet serwisSamochodowyDataSet;
        private MechanicyTableAdapter mechanicyTableAdapter;

        public DodajMechanika(SerwisSamochodowyDataSet dataSet,
            MechanicyTableAdapter tableAdapter)
        {
            this.serwisSamochodowyDataSet = dataSet;
            this.mechanicyTableAdapter = tableAdapter;
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (!String.IsNullOrEmpty(this.textBox1.Text))
            {
                /*
                 * Utworzenie obiektu reprezentującego pojedynczą krotkę w bazie danych
                 */
                SerwisSamochodowyDataSet.MechanicyRow mechanik =
                    this.serwisSamochodowyDataSet.Mechanicy.NewMechanicyRow();

                /*
                 * Wypełnienie obiektu danymi z formatki
                 */
                mechanik.nazwisko = this.textBox1.Text;

                /*
                 * Aktualizacja obiektu DataSet
                 */
                this.serwisSamochodowyDataSet.Mechanicy.Rows.Add(mechanik);

                /*
                 * Aktualizacja zmian z bazie danych z wykorzystaniem obiektu DataAdapter
                 */
                if (this.mechanicyTableAdapter.Update(
                    (SerwisSamochodowyDataSet.MechanicyDataTable)
                    this.serwisSamochodowyDataSet.Mechanicy.GetChanges()) > 0)
                {
                    this.Close();
                }
            }
            else
            {
                MessageBox.Show("Wypełnij pole: Nazwisko");
            }
```

```
        }
    }
}
```

Listing 3 – Kod obsługi formatki dodawania mechanika

Utworzony schemat bazy danych zawiera ograniczenia integralnościowe unikalności atrybutów relacji. W celu obsłużenia wyjątków zgłaszanych przez aplikację podczas próby dodawania do bazy danych już istniejących krotek, niezbędne jest zmodyfikowanie metod aktualizacji bazy danych znajdujących się w pliku: *SerwisSamochodowyDataSet.Designer.cs*. W tym celu dodajemy referencje do klas zawierających obsługę wyjątków oraz obsługę komunikatów (Listing 4).

```
using System.Windows.Forms;
using System;
```

Listing 4 – Referencje do klasy obsługi wyjątków oraz klasy komponentów graficznych

W metodach aktualizacji relacji Pojazdy oraz Mechanicy definiujemy obsługę zgłaszanych wyjątków poprzez wyświetlenie komunikatów informacyjnych użytkownikowi (Listingi 5-8).

```
[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public virtual int Update(SerwisSamochodowyDataSet.PojazdyDataTable dataTable) {
    return this.Adapter.Update(dataTable);
}
```

Listing 5 – Metoda Update relacji Pojazdy przed zmianą

```
[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public virtual int Update(SerwisSamochodowyDataSet.PojazdyDataTable dataTable) {
    int result = 0;
    try
    {
        result = this.Adapter.Update(dataTable);
    }
    catch (Exception)
    {
        MessageBox.Show("Podany rekord juz istnieje");
    }
    return result;
}
```

Listing 6 – Metoda Update relacji Pojazdy po zmianie

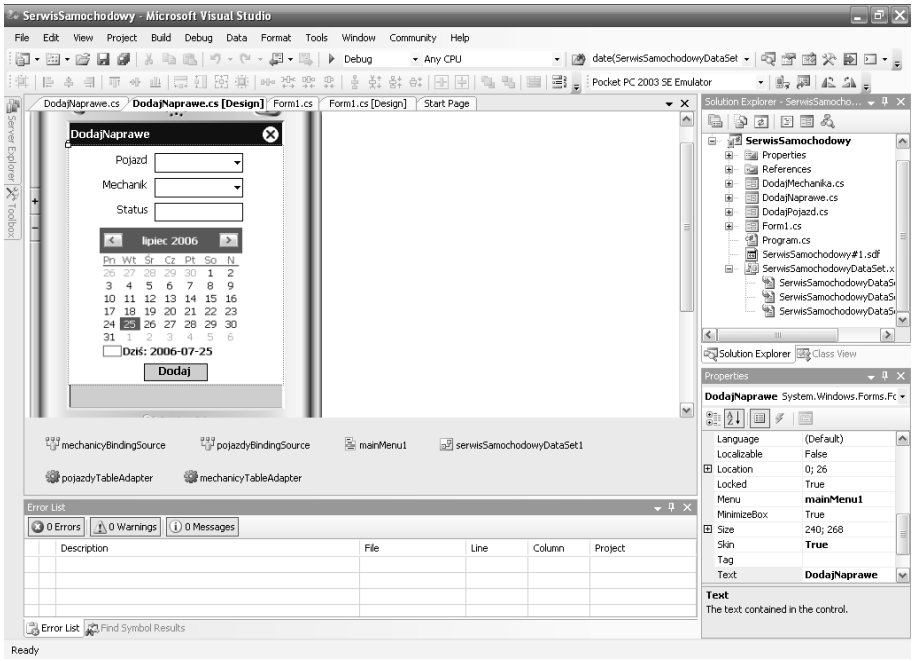
```
[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public virtual int Update(SerwisSamochodowyDataSet.MechanicyDataTable dataTable) {
    return this.Adapter.Update(dataTable);
}
```

Listing 7 – Metoda Update relacji Mechanicy przed zmianą

```
[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public virtual int Update(SerwisSamochodowyDataSet.MechanicyDataTable dataTable) {
    int result = 0;
    try
    {
        result = this.Adapter.Update(dataTable);
    }
    catch (Exception)
    {
        MessageBox.Show("Podany rekord juz istnieje");
    }
    return result;
}
```

Listing 8 – Metoda Update relacji Mechanicy po zmianie

Ostatnim krokiem tworzenia aplikacji jest utworzenie formy dodawania napraw do bazy danych łączących poszczególne pojazdy z konkretnymi mechanikami. Przykładowa forma dodawania napraw wygląda następująco (Rysunek 7).



Rysunek 7 – Przykładowa forma dodawania naprawy do bazy danych

Wykorzystać możemy komponenty **ComboBox**, które umożliwią nam wybór konkretnego pojazdu i mechanika oraz **MonthCallendar** – do określenia daty naprawy. Dodatkowo wykorzystany zostanie komponent **TextBox** do podania statusu naprawy.

Dla komponentów **ComboBox** definiujemy źródło danych poprzez wskazanie odpowiedniej tabeli w atrybucie **DataSource** konkretnego komponentu. Określamy również parametry **DisplayMember** (podajemy atrybut powiązanej relacji, który widoczny będzie na liście po rozwinięciu komponentu **ComboBox**) oraz **ValueMember** (atrybut powiązanej relacji, którego wartości zwracane będą przez komponent **ComboBox** po wybraniu przez użytkownika odpowiedniej wartości z listy).

Możemy przystąpić do zdefiniowania metod obsługi dodawania nowej naprawy do bazy danych. Przykładowy kod realizujący powyższe zadanie (Listing 9).

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using SerwisSamochodowy.SerwisSamochodowyDataSetTableAdapters;

namespace SerwisSamochodowy
{
    public partial class DodajNaprawe : Form
    {
        private SerwisSamochodowyDataSet serwisSamochodowyDataSet;
        private NaprawyTableAdapter naprawyTableAdapter;

        public DodajNaprawe(SerwisSamochodowyDataSet dataSet,
            NaprawyTableAdapter tableAdapter)
        {
            this.serwisSamochodowyDataSet = dataSet;
            this.naprawyTableAdapter = tableAdapter;
            InitializeComponent();
        }

        private void DodajNaprawe_Load(object sender, EventArgs e)
        {
            // Pobranie danych do tabeli Pojazdy obiektu DataSet z bazy danych
            this.pojazdyTableAdapter.Fill(this.serwisSamochodowyDataSet1.Pojazdy);
            // Pobranie danych do tabeli Mechanicy obiektu DataSet z bazy danych
            this.mechanicyTableAdapter.Fill(this.serwisSamochodowyDataSet1.Mechanicy);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (!String.IsNullOrEmpty(this.textBox1.Text))
            {
                /*
                 * Utworzenie obiektu reprezentującego pojedynczą krotkę w bazie danych
                 */
                SerwisSamochodowyDataSet.NaprawyRow naprawa =
                    this.serwisSamochodowyDataSet.Naprawy.NewNaprawyRow();

                /*
                 * Wypełnienie obiektu danymi z formatki
                 */
                naprawa.id_pojazdu = (int)this.comboBox1.SelectedValue;
                naprawa.id_mechanika = (int)this.comboBox2.SelectedValue;
                naprawa.data = this.monthCalendar1.SelectionStart;
                naprawa.status = this.textBox1.Text;

                /*
                 * Aktualizacja obiektu DataSet
                 */
                this.serwisSamochodowyDataSet.Naprawy.Rows.Add(naprawa);

                /*
                 * Aktualizacja zmian z bazie danych z wykorzystaniem obiektu DataAdapter
                 */
                if (this.naprawyTableAdapter.Update(
                    (SerwisSamochodowyDataSet.NaprawyDataTable)
```

```
                    this.serwisSamochodowyDataSet.Naprawy.GetChanges()) > 0)
                {
                    this.Close();
                }
            }
            else
            {
                MessageBox.Show("Wypełnij pole: Status");
            }
        }
    }
}
```

Listing 9 – Przykładowy kod realizujący dodawanie nowej naprawy do bazy danych

Gotową aplikację możemy uruchomić.

3. Zadanie

Dodanie możliwości ewidencjonowania czasu pracy mechaników oraz zajmowanych przez nich stanowisk pracy:

Należy dodać do relacji NAPRAWY pole:
czas_naprawy datetime

oraz stworzyć relację ETATY:
#id_etatu int
nazwa varchar(100)

nastepnie dodać do relacji MECHANICY pole:
id_etatu int

będące kluczem obcym do relacji Etaty.

Po powyższej modyfikacji istniejącej bazy danych należy stworzyć Formatkę Dodawania Nowego Etatu. Należy wzorować się na formatkach już istniejących. Ponadto należy zmodyfikować Formatkę Dodawania Nowego Mechanika o możliwość przypisania mu etatu na jakim będzie pracował. Najlepiej aby był to etat wybierany z ComboBox.