

**Projekt z przedmiotu  
Programowanie**

**TEMAT:  
Panel Zarządzający montażami**

Prowadzący: Wykonał:  
dr Marek Jaszuk Dawid Filipak W57137  
 4IIZ

# 1. Opis założeń projektu:

Celem projektu jest zaprojektowanie programu, który może być wykorzystywany w małej firmie do rejestrowania zadań pracownikom. Głównym założeniem projektu jest umożliwienie zapisywania montaży danym pracownikom, oraz rejestracji i logowania się pracownika po to aby mógł zobaczyć swoje zadania do wykonania.

# 2.Specyfikacja wymagań:

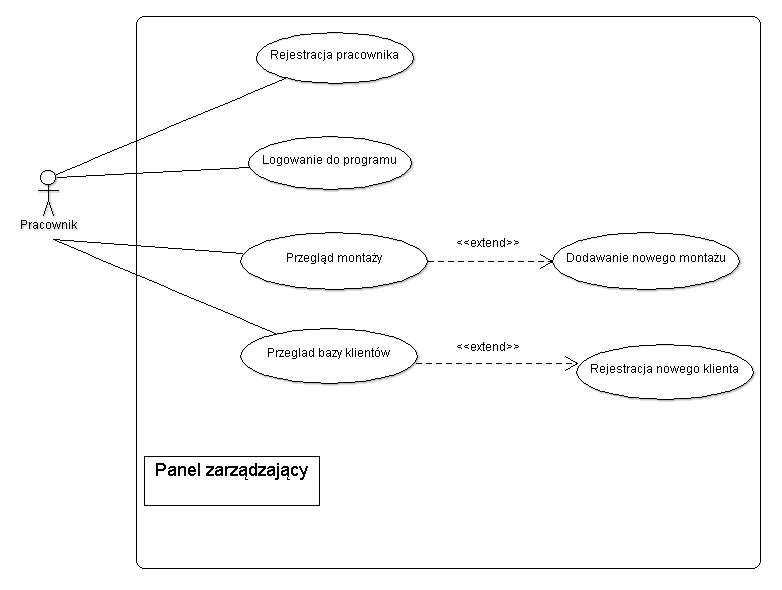
1. Wymagania funkcjonalne

* Przechowywania danych pracowników:
  + Imię i nazwisko
  + Adres
  + Nr telefonu
  + Email
  + Login
  + Hasło
* Logowanie pracownika do programu
* Możliwość rejestrowania nowych montaży oraz ich przeglądania
* Rejestracja nowych klientów do bazy danych:
* Imię i nazwisko
* Adres
* Nr dowodu
* Pesel
* Nr telefonu

1. Wymagania niefunkcjonalne

* program nie powinien przekraczać 500MB
* program ma być przeznaczony tylko na system Windows

# 3. Diagram przypadków użycia

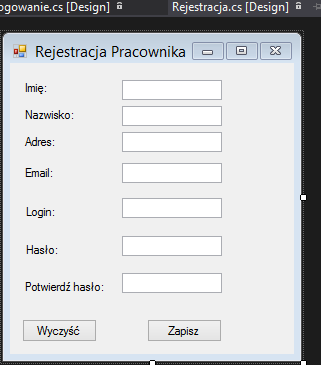


# 3. Opis techniczny projektu

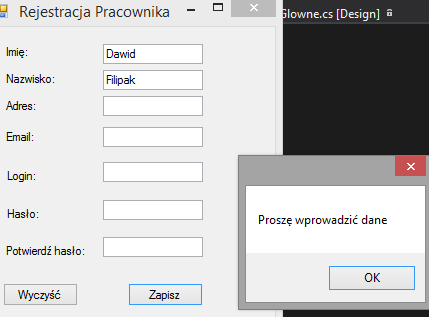
Program będzie miał za zadanie dodawanie pracowników do lokalnej bazy „Pracownicy", dzięki której będziemy mogli się zalogować z kolumny Login i Hasło. Po zalogowaniu otworzy się okno główne programu, w którym będzie możliwość dodawania do bazy danych montaże, w której będą dane Imię, Nazwisko, Adres oraz jaki pracownik zostanie przypisany do nowego montażu. Listę montaży będzie można otworzyć w zakładce „Lista montaży", w której wyświetli sie tabela z danymi.  
 Zakładka „Nowy klient", zawiera dane Imię, Nazwisko, Adres, Nr dowodu, Pesel oraz nr telefonu klienta. Wszystkie wpisane dane dodawana są do tabeli „klienci", którą można otworzyć w zakładce „Baza Klientów".  
 Każda zmienna powinna być wpisana i mieć odpowiednią formę. Wprowadziłem zabezpieczenie tak aby zostały wpisane dane, oraz Pesel, Nr dowodu, nr telefonu posiadał odpowiednia ilość znaków. Po błędnym wpisaniu zostanie wyświetlony komunikat z prośbą o poprawieni danych.

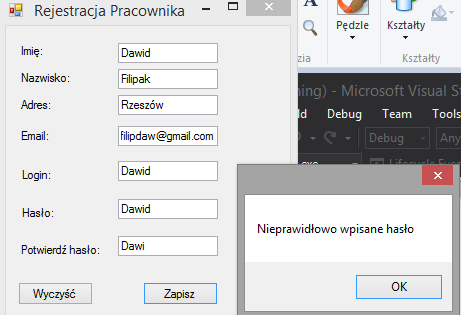
# **4. Prezentacja warstw użytkowej projektu**

a. Dodawanie Pracowników:



W oknie Logowania znajduję się opcja rejestracji po wciśnięci przycisku pojawia się okno „Rejestracji Pracownika", w którym należy wpisać odpowiednie dane oraz Login i Hasło potrzebne do późniejszego logowania. Każdy z TextBoxów zabezpieczony jest tak aby nie było pustych pól. Hasło do logowania należy poprawnie powtórzyć w przeciwnym razie zostanie wyświetlony komunikat o błędnym haśle. Jest także przycisk czyszczący wpisane dane.





Poniżam został przedstawiony kod, w którym dodatkowo zamieściłem komentarze informujące za co odpowiedzialne funkcje programu:

///<summary> Klasa dodaje nowego pracownika </summary>

public partial class Rejestracja : Form

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=D:\STUDIA !!!!!\IV semestr\Programowanie\Projekt windows form Dawid Filipak (57137)\Panel zarządzający\Panel zarządzający\Bazydanych\Bazydanych.mdf; Integrated Security = True";

public Rejestracja()

{

InitializeComponent();

}

/// <summary>

/// Metoda służy do dodawania nowego pracownika do tabeli pracownicy

/// </summary>

/// <returns> Dodaje nowe rekordy do tabeli Pracownicy</returns>

/// <param name="sender"> Odwołanie do przycisku Zapisz </param>

/// <param name="e"> Przekazuje określone zdarzenie, zapisuje dane</param>

private void butt\_zaloguj\_Click(object sender, EventArgs e)

{

if (txtImie.Text == "" || txtNazwisko.Text == "" || txtAdres.Text == "" || txtEmail.Text == "" || txtLogin.Text == "" || txtPassword.Text == "")

{

MessageBox.Show("Proszę wprowadzić dane");

}

else

{

using (SqlConnection sqlCon = new SqlConnection(connectionString))

{

if (txtPassword.Text == txtPotPassword.Text)

{

sqlCon.Open();

SqlCommand sqlCmd = new SqlCommand("PracownicyAdd", sqlCon);

sqlCmd.CommandType = CommandType.StoredProcedure;

sqlCmd.Parameters.AddWithValue("@Imie", txtImie.Text.Trim());

sqlCmd.Parameters.AddWithValue("@Nazwisko", txtNazwisko.Text.Trim());

sqlCmd.Parameters.AddWithValue("@Adres", txtAdres.Text.Trim());

sqlCmd.Parameters.AddWithValue("@Email", txtEmail.Text.Trim());

sqlCmd.Parameters.AddWithValue("@Login", txtLogin.Text.Trim());

sqlCmd.Parameters.AddWithValue("@Password", txtPassword.Text.Trim());

sqlCmd.ExecuteNonQuery();

MessageBox.Show("Pracownik dodawny pomyślnie");

txtImie.Clear();

txtNazwisko.Clear();

txtAdres.Clear();

txtEmail.Clear();

txtLogin.Clear();

txtPassword.Clear();

txtPotPassword.Clear();

}

else

{

MessageBox.Show("Nieprawidłowo wpisane hasło");

txtPassword.Clear();

txtPotPassword.Clear();

}

}

}

}

/// <summary>

/// Metoda do czyszczenia wprowadzonych danych

/// </summary>

/// <param name="sender"> Odwołanie do przycisku Wyczyść </param>

/// <param name="e"> Przekazuje zdarzenie, czyści wprowadzone dane w TextBoxach</param>

private void butt\_wyczysc\_Click(object sender, EventArgs e)

{

txtImie.Clear();

txtNazwisko.Clear();

txtAdres.Clear();

txtEmail.Clear();

txtLogin.Clear();

txtPassword.Clear();

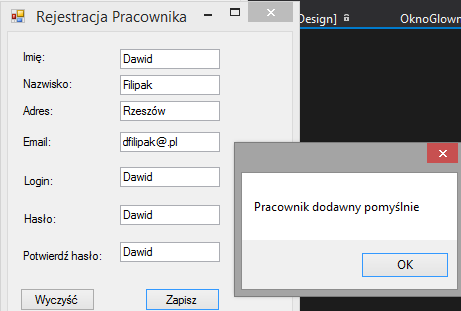
txtPotPassword.Clear();

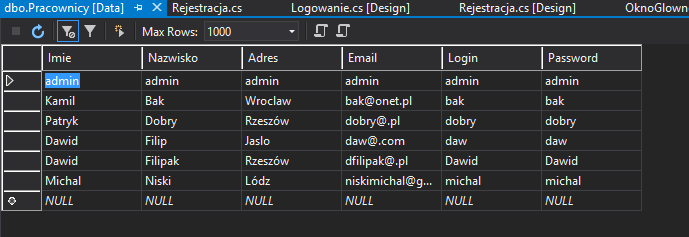
}

}

}

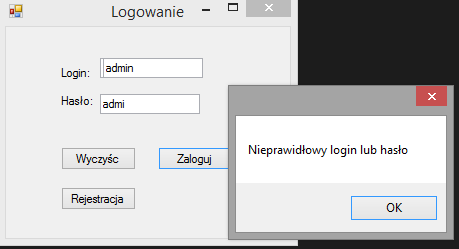
Po wpisaniu prawidłowo danych wyświetlany jest komunikat o pomyślnej operacji i pracownik zostaje dodany do Bazy danych. Poniżej przedstawiam screeny działania programu i tabele „Pracownicy".





### b. Logowanie do programu

Logowanie odbywa się po uruchomieniu programu, należy wpisać Login i hasło pracownika z bazy danych pracownicy. Po wpisaniu błędnych danych wyświetlony zostanie komunikat o błędnych danych.



///<summary> Klasa służąca do logowania</summary>

public partial class Logowanie : Form

{

const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=D:\STUDIA !!!!!\IV semestr\Programowanie\Projekt windows form Dawid Filipak (57137)\Panel zarządzający\Panel zarządzający\Bazydanych\Bazydanych.mdf; Integrated Security = True";

public Logowanie()

{

InitializeComponent();

}

private void butt\_rejestracja\_Click(object sender, EventArgs e)

{

new Rejestracja().Show();

}

///<summary>

///Metoda służąca do sprawdzania danych pracownika

///</summary>

/// <param name="sender"> Odwołanie do przycisku Zaloguj </param>

/// <param name="e"> Przekazuje określone zdarzenie, następuje zalogowanie do głównego okna</param>

private void butt\_zaloguj\_Click(object sender, EventArgs e)

{

DataTable result = null;

using (SqlConnection sqlCon = new SqlConnection(connectionString))

{

sqlCon.Open();

using (SqlCommand cmd = sqlCon.CreateCommand())

{

cmd.CommandType = CommandType.Text;

cmd.CommandText="Select Imie, Nazwisko from Pracownicy where Login = '" + login.Text + "' and password = '" + password.Text +"'";

using (SqlDataAdapter da = new SqlDataAdapter(cmd))

{

result = new DataTable();

da.Fill(result);

if (result.Rows.Count == 1)

{

this.Hide();

new OknoGlowne().Show();

}

else

{

MessageBox.Show("Nieprawidłowy login lub hasło");

}

}

}

}

}

/// <summary>

/// Metoda służąca do czyszczenia wpisanych danych

/// </summary>

/// <param name="sender"> Odwołanie do przycisku Wyczyść </param>

/// <param name="e"> Przekazuje określone zdarzenie, czyści wporwadzone dane</param>

private void butt\_wyczysc\_Click(object sender, EventArgs e)

{

login.Clear();

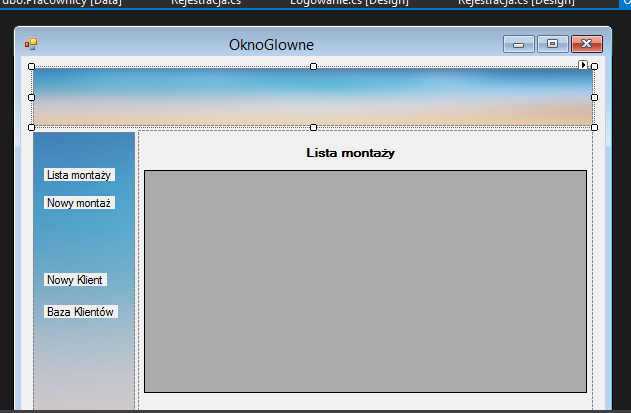
password.Clear();

}

}

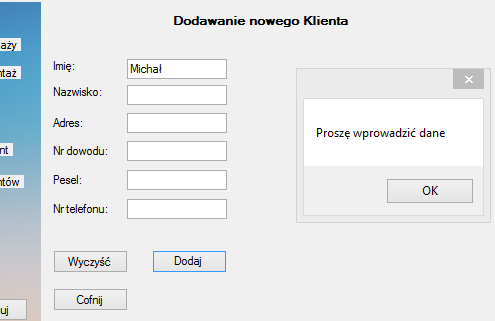
}

Po wpisaniu odpowiedniego loginu i hasła Logujemy się do Głównego okna programu.

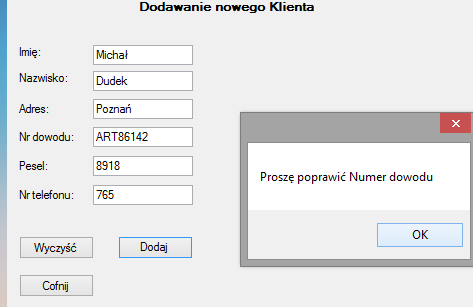


### c. Dodawanie Nowych klientów:

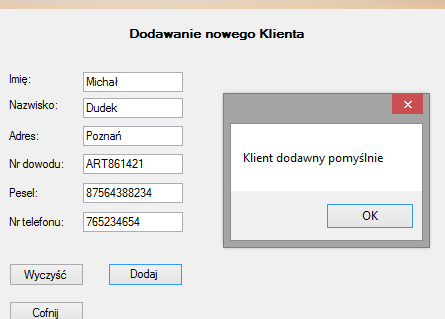
W głównym oknie znajdują się zakładki do odpowiednich funkcji programu. W zakładce Nowy Klient, także należy wprowadzić wszystkie dane, jeśli nie wprowadzimy danych zostanie wyświetlony komunikat.



Po wprowadzeniu błędnie Nr dowodu, peselu, Nr telefonu zostanie wyświetlony odpowiedni komunikat:



Po poprawie wszystkich danych klient zostaje poprawnie dodany do bazy „Klienci". Okno zawiera także przyciski Wyczyść oraz Cofnij, który odpowiada za wycofanie z panelu dodawania klienta.



/// <summary>

/// Sprawdza czy zostały odpowiednio sprawdzone dane jeśli tak klient zostaje dodany

/// </summary>

/// <returns> Dodaje nowe rekordy do tabeli Klienci</returns>

/// <param name="sender"> Odwołanie do przycisku Dodaj </param>

/// <param name="e"> przekazuje określone zdarzenie, zapisuje wprowadzone dane do tabeli</param>

private void buttDodajKlienta\_Click(object sender, EventArgs e)

{

if (tbklImie.Text == "" || tbklNazwisko.Text == "" || tbklAdres.Text == "" || tbklDowod.Text == "" || tbklPesel.Text == "" || tbklTelefon.Text == "")

{

MessageBox.Show("Proszę wprowadzić dane");

}

else

{

using (SqlConnection sqlCon = new SqlConnection(connectionString))

{

if (tbklDowod.Text.Length == 9)

{

if (tbklPesel.Text.Length == 11)

{

if (tbklTelefon.Text.Length == 9)

{

sqlCon.Open();

SqlCommand sqlCmd = new SqlCommand("KlienciAdd", sqlCon);

sqlCmd.CommandType = CommandType.StoredProcedure;

sqlCmd.Parameters.AddWithValue("@Imie", tbklImie.Text.Trim());

sqlCmd.Parameters.AddWithValue("@Nazwisko", tbklNazwisko.Text.Trim());

sqlCmd.Parameters.AddWithValue("@Adres", tbklAdres.Text.Trim());

sqlCmd.Parameters.AddWithValue("@NrDowodu", tbklDowod.Text.Trim());

sqlCmd.Parameters.AddWithValue("@Pesel", tbklPesel.Text.Trim());

sqlCmd.Parameters.AddWithValue("@NrTelefonu", tbklTelefon.Text.Trim());

sqlCmd.ExecuteNonQuery();

MessageBox.Show("Klient dodawny pomyślnie");

tbklImie.Clear();

tbklNazwisko.Clear();

tbklAdres.Clear();

tbklDowod.Clear();

tbklPesel.Clear();

tbklTelefon.Clear();

}

else { MessageBox.Show("Proszę poprawić Numer telefonu"); }

}

else { MessageBox.Show("Proszę poprawić pesel"); }

}

else { MessageBox.Show("Proszę poprawić Numer dowodu"); }

sqlCon.Close();

}

}

}

Poniżej przedstawiam kwerendę odpowiedzialną za dodawanie klientów do bazy danych:

CREATE PROCEDURE [dbo].[KlienciAdd]

@Imie varchar(50),

@Nazwisko varchar(50),

@Adres varchar(50),

@NrDowodu varchar(50),

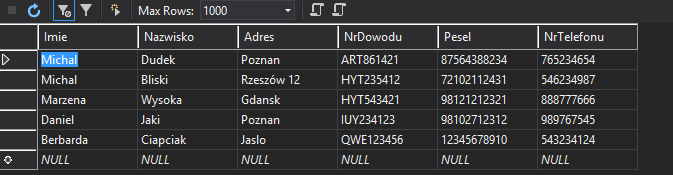
@Pesel varchar(50),

@NrTelefonu varchar(50)

AS

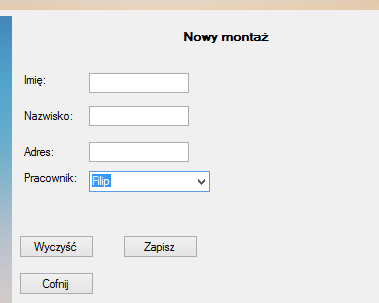
Insert INTO Klienci(Imie,Nazwisko,Adres,NrDowodu,Pesel,NrTelefonu)

Values (@Imie,@Nazwisko,@Adres,@NrDowodu,@Pesel,@NrTelefonu)



### d. Dodawanie Montaży

Dodawanie montaży odbywa się analogicznie do dodawania klienta, w tym że jest dodatkowa opcja wybierania pracownika do wykonania montażu:



/// <summary>

/// Metoda czyści wprowadzone dane dodawania nowego klienta

/// </summary>

/// <param name="sender"> Odwołanie do przycisku Wyczyść </param>

/// <param name="e"> Przekazuje określone zdarzenie, czyści wprowadzone dane</param>

private void buttWyczyscNmontaz\_Click(object sender, EventArgs e)

{

tbNMImie.Clear();

tbNMNazwisko.Clear();

tbNMadres.Clear();

}

private void buttCofNmontaz\_Click(object sender, EventArgs e)

{

panelNMontaz.Hide();

}

/// <summary>

/// Wyświetla tabele z danymi

/// </summary>

private void OknoGlowne\_Load(object sender, EventArgs e)

{

this.pracownicyTableAdapter.Fill(this.bazydanychDataSet.Pracownicy);

}

/// <summary>

/// Metoda służąca do dodawania nowego montażu do bazy danych

/// </summary>

/// <returns> Dodaje nowe rekordy do tabeli montaże</returns>

/// <param name="sender"> Odwołanie do przycisku Zapisz </param>

/// <param name="e"> Przekazuje określone zdarzenie, zapisuje dane do tabeli Montaże</param>

private void ButtZapiszNmontaz\_Click(object sender, EventArgs e)

{

if (tbNMImie.Text == "" || tbNMNazwisko.Text == "" || tbNMadres.Text == "" )

{

MessageBox.Show("Proszę wprowadzić dane");

}

else

{

using (SqlConnection sqlCon = new SqlConnection(connectionString))

{

sqlCon.Open();

SqlCommand sqlCmd = new SqlCommand("MontazeAdd", sqlCon);

sqlCmd.CommandType = CommandType.StoredProcedure;

sqlCmd.Parameters.AddWithValue("@Imie",tbNMImie.Text.Trim());

sqlCmd.Parameters.AddWithValue("@Nazwisko", tbNMNazwisko.Text.Trim());

sqlCmd.Parameters.AddWithValue("@Adres", tbNMadres.Text.Trim());

sqlCmd.Parameters.AddWithValue("@Pracownik", cbPracownik.Text.Trim());

sqlCmd.ExecuteNonQuery();

MessageBox.Show("Montaż dodawny pomyślnie");

tbNMImie.Clear();

tbNMNazwisko.Clear();

tbNMadres.Clear();

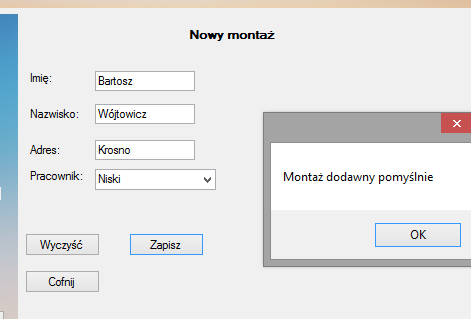
sqlCon.Close();

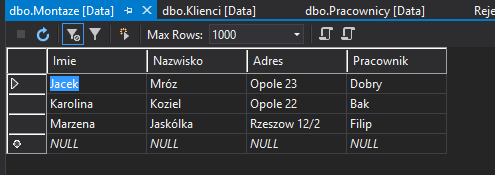
}

}

}

}





Procedura odpowiedzialna za dodawanie montaży:

CREATE PROCEDURE [dbo].[MontazeAdd]

@Imie varchar(50),

@Nazwisko varchar(50),

@Adres varchar(50),

@Pracownik varchar(50)

AS

Insert INTO Montaze(Imie,Nazwisko,Adres,Pracownik)

Values (@Imie,@Nazwisko,@Adres,@Pracownik)

### e. Wyświetlenie Baz danych:

Bazy danych zostają wyświetlone przy pomocy kontrolki DataGridView do której zostają wczytane wszystkie rekordy z baz danych Klienci oraz nowe montaże

/// <summary>

/// Wyświetla tabele z danymi

/// </summary>

private void OknoGlowne\_Load(object sender, EventArgs e)

{

this.pracownicyTableAdapter.Fill(this.bazydanychDataSet.Pracownicy);

} }

/// <summary>

/// Metoda służąca do wyświetlenia montaży

/// </summary>

/// <param name="sender"> Odwołanie do przycisku Lista montaży </param>

/// <param name="e"> Przekazuje określone zdarzenie, wyświetla tabele z danymi</param>

using (SqlConnection sqlCon = new SqlConnection(connectionString))

{

DataTable returnTable = new DataTable();

sqlCon.Open();

SqlCommand cmd = sqlCon.CreateCommand();

cmd.CommandText = "Select \* from Montaze";

SqlDataAdapter adapter = new SqlDataAdapter(cmd);

adapter.Fill(returnTable);

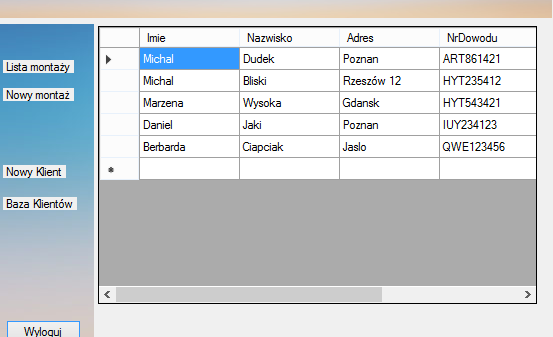
sqlCon.Close();

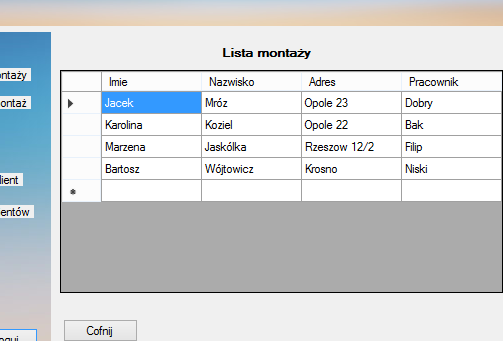
tabelaMontazy.AutoGenerateColumns = true;

tabelaMontazy.DataSource = returnTable;

}

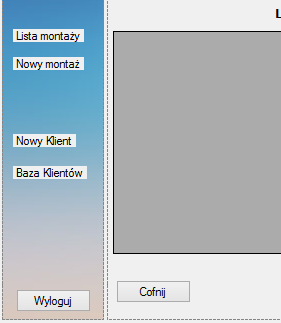
}





### f. Wylogowanie:

W programie umieściłem także przycisk służący za wylogowanie z programu i powrót do okna logowania.



# 5. Testy jednostkowe

Testy jednostkowe dopisałem do klasy Rejestracja. Przedstawiają one podstawowe działania matematyczne takie jak suma, różnica, iloczyn, iloraz:

public int Add(int liczbaA, int liczbaB)

{

return liczbaA + liczbaB;

}

public int Sub(int liczbaA, int liczbaB)

{

return liczbaA - liczbaB;

}

public int Mul(int liczbaA, int liczbaB)

{

return liczbaA \* liczbaB;

}

public int Div(int liczbaA, int liczbaB)

{

return liczbaA / liczbaB;

}

Poniżej przedstawiłem klasę testującą, która wygląda następująco i jak można zauważyć testy zostały przeprowadzone prawidłowo.

[TestClass()]

public class RejestracjaTests

{

[TestMethod()]

public void AddTest()

{

Rejestracja test = new Rejestracja();

int liczbaA = 2;

int liczbaB = 2;

int wynik = 4;

int actual;

actual = test.Add(liczbaA, liczbaB);

Assert.AreEqual(wynik, actual);

}

[TestMethod()]

public void SubTest()

{

Rejestracja test = new Rejestracja();

int liczbaA =4;

int liczbaB = 2;

int wynik = 2;

int actual;

actual = test.Sub(liczbaA, liczbaB);

Assert.AreEqual(wynik, actual);

}

[TestMethod()]

public void MulTest()

{

Rejestracja test = new Rejestracja();

int liczbaA = 2;

int liczbaB = 2;

int wynik = 4;

int actual;

actual = test.Mul(liczbaA, liczbaB);

Assert.AreEqual(wynik, actual);

}

[TestMethod()]

public void DivTest()

{

Rejestracja test = new Rejestracja();

int liczbaA = 4;

int liczbaB = 2;

int wynik = 2;

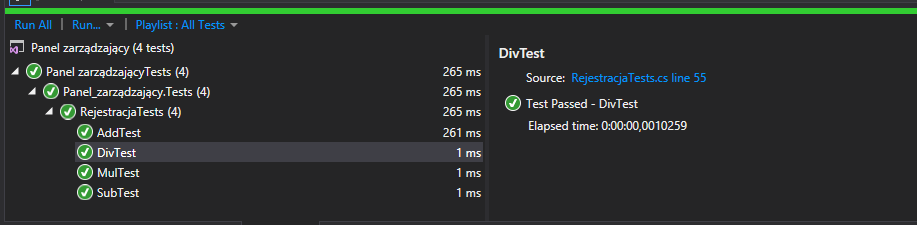
int actual;

actual = test.Div(liczbaA, liczbaB);

Assert.AreEqual(wynik, actual);

}

}



# 6. Kod źródłowy

Kod źródłowy projektu został zamieszczony w poniższym repozytorium: <https://github.com/DawidFilipak/Panel-zarzadzajacy>

Zawartość

[1. Opis założeń projektu: 2](#_Toc528603681)

[2.Specyfikacja wymagań: 2](#_Toc528603682)

[3. Diagram przypadków użycia 3](#_Toc528603683)

[3. Opis techniczny projektu 3](#_Toc528603684)

[4. Prezentacja warstw użytkowej projektu 4](#_Toc528603685)

[a. Dodawanie Pracowników: 4](#_Toc528603686)

[b. Logowanie do programu 7](#_Toc528603687)

[c. Dodawanie Nowych klientów: 8](#_Toc528603688)

[d. Dodawanie Montaży 11](#_Toc528603689)

[e. Wyświetlenie Baz danych: 13](#_Toc528603690)

[f. Wylogowanie: 14](#_Toc528603691)

[5. Testy jednostkowe 14](#_Toc528603692)

[6. Kod źródłowy 14](#_Toc528603693)