



POLITECHNIKA POZNAŃSKA

Wydział Informatyki I Telekomunikacji

Instytut Radiotelekomunikacji

Dokumentacja systemu dostępu do widma.

Spis treści

Spis rysunków	3
Wstęp	4
1. Część. Dokumentacja użytkownika	5
1.1. Uruchamianie aplikacji webowej	6
1.2. Rejestracja i logowanie do systemu (w trakcie realizacji)	6
1.3. Dodawanie stacji w aplikacji internetowej (w trakcie realizacji)	7
1.4. Dodawanie stacji w aplikacji desktopowej	8
2. Część. Dokumentacja techniczna.....	10
2.1. Elementy systemu.....	11
2.2. Ogólne działanie systemu	12
2.3. Szczegóły implementacji	13
2.3.1. Interfejs użytkownika.....	13
2.3.2. Baza danych	18
2.3.3. Obliczenia.....	18

Spis rysunków

Rys. 1.1. Logowanie do systemu.	6
Rys. 1.2. Rejestracja użytkownika.	7
Rys. 1.3. Dodawanie stacji.	8
Rys. 1.4. Logowanie do aplikacji desktopowej.	8
Rys. 1.5. Widok okna aplikacji desktopowej.	9
Rys. 2.1. Schemat działania projektu.	12
Rys. 2.2. Okienko Logowania.	13
Rys. 2.3. Główne okno dodawania stacji.	13
Rys. 2.4. Funkcja Plotmap.....	14
Rys. 2.5. Funkcja PlotStation2.	15
Rys. 2.6. Fragment kodu sugerujący nową stację.	16
Rys. 2.7. Komponenty do wprowadzania danych.....	17
Rys. 2.8. Funkcja odpowiedzialna za otwarcie bazy danych.	18
Rys. 2.9. Funkcja odpowiedzialna za dodawanie stacji do bazy danych.	18
Rys. 2.10. Funkcja BaseStation.	18
Rys. 2.11. Funkcja obliczająca dystans.....	19
Rys. 2.12. Obliczanie straty mocy.	19
Rys. 2.13. Obliczanie sumy i mocy odbiornika.....	20
Rys. 2.14. Obliczanie SINR.	21

Wstęp

Aplikacja Wid-Control ma pomóc w wyznaczeniu odpowiednich parametrów do podstawienie stacji nadawczo-odbiorczej w odpowiedniej lokalizacji.

Niniejsza instrukcja ma na celu wprowadzenie użytkowników w aplikację oraz omówienie wszystkich jej funkcjonalności.

1. Część.

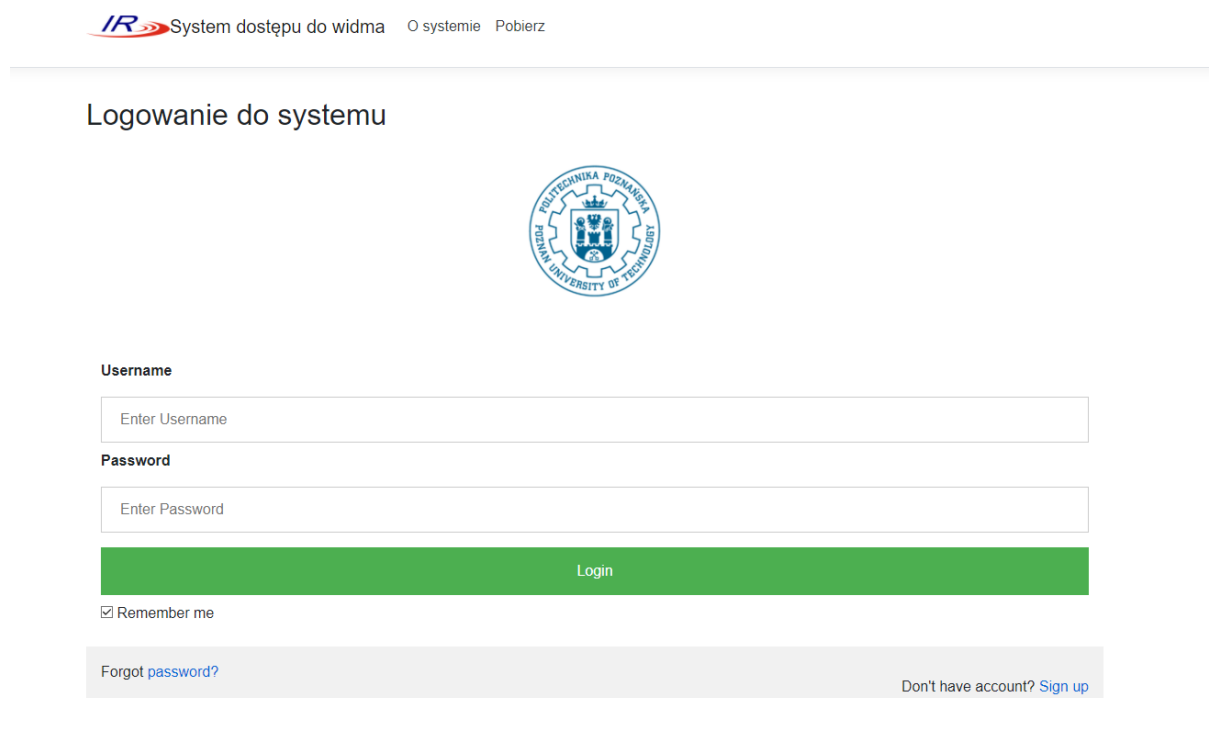
Dokumentacja użytkownika

1.1. Uruchamianie aplikacji webowej

W celu rozpoczęcia pracy z systemem należy przejść pod adres: Do uruchomienia aplikacji webowej potrzebna jest przeglądarka internetowa, najlepiej w najnowszej wersji. Po zarejestrowaniu się w aplikacji webowej lub zalogowaniu (funkcjonalność na razie w fazie wdrażania), można używać systemu poprzez tą aplikację, albo pobrać aplikację desktopową (tylko Windows).

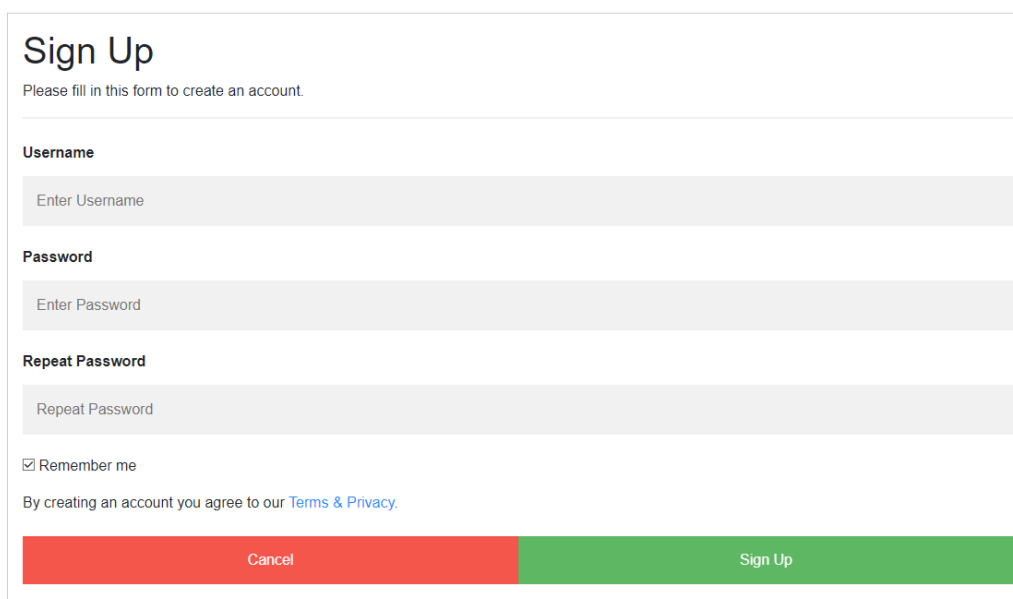
1.2. Rejestracja i logowanie do systemu (w trakcie realizacji)

Dostęp do aplikacji Wid-Control realizowany jest poprzez mechanizm kont użytkownika. Konta są identyfikowane poprzez nazwę użytkownika i hasło. Jeśli nie posiadamy konta, należy kliknąć „Sign up” w prawym dolnym rogu. Strona przeniesie nas do systemu rejestracji (Rys. 1.2). Po udanej rejestracji można przystąpić do logowania (Rys. 1.1). Wpisujemy swoje dane podane podczas rejestracji. Jeśli, zapomnimy hasła, mamy możliwość odzyskania go klikając „Forgot password” po lewej stronie.



The screenshot shows the login interface for the 'IR System dostępu do widma'. At the top, there is a header with the 'IR' logo and links for 'O systemie' and 'Pobierz'. The main heading is 'Logowanie do systemu'. Below this is the logo of the 'Polska Akademia Nauk - PAN' (Polish Academy of Sciences). The login form consists of two input fields: 'Username' with a placeholder 'Enter Username' and 'Password' with a placeholder 'Enter Password'. A green 'Login' button is positioned below the password field. There is a checkbox labeled 'Remember me' and a link 'Forgot password?' below the password field. At the bottom right, there is a link 'Don't have account? Sign up'.

Rys. 1.1. Logowanie do systemu.



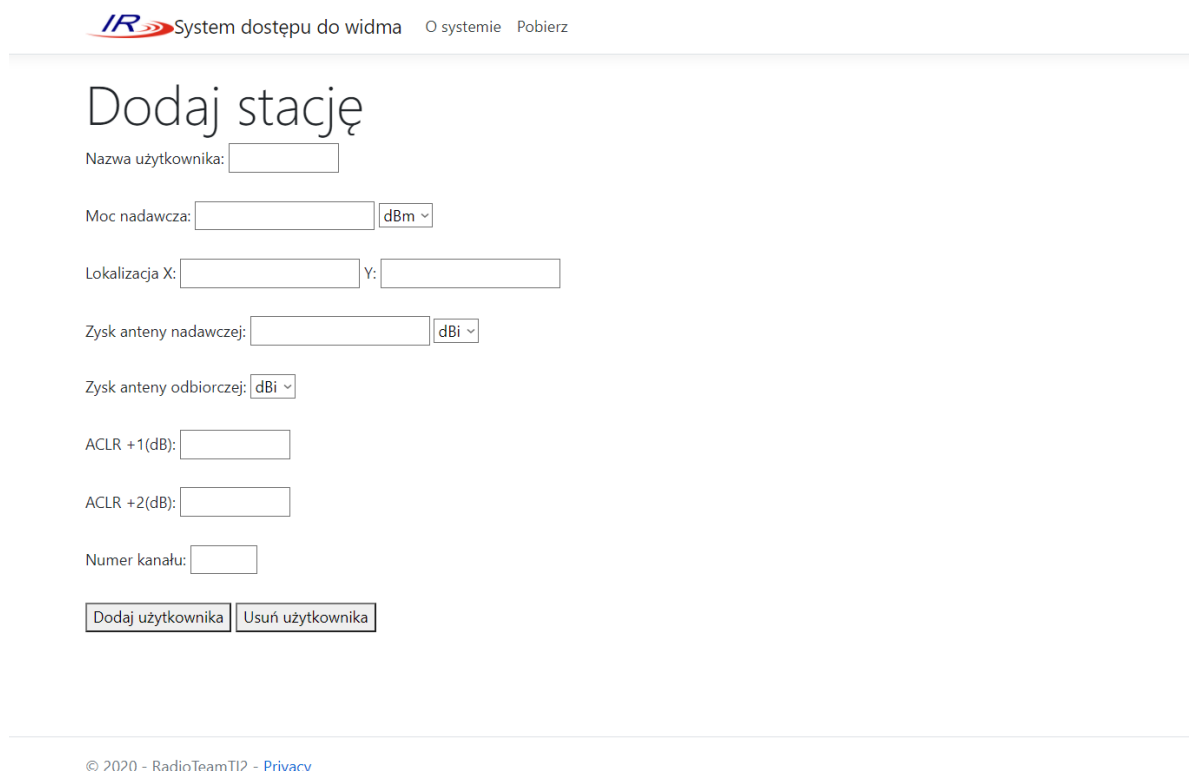
The image shows a 'Sign Up' form with the following elements:

- Sign Up** (Section Header)
- Please fill in this form to create an account.
- Username** (Label)
- Enter Username (Input field)
- Password** (Label)
- Enter Password (Input field)
- Repeat Password** (Label)
- Repeat Password (Input field)
- ☒ Remember me (Checkbox)
- By creating an account you agree to our [Terms & Privacy](#).
- Cancel** (Red button)
- Sign Up** (Green button)

Rys. 1.2. Rejestracja użytkownika.

1.3. Dodawanie stacji w aplikacji internetowej (w trakcie realizacji)

Po zalogowaniu się wyświetla nam się obraz przedstawiony na rysunku 1.3. W celu dodania użytkownika do systemu należy uzupełnić wszystkie pola i zatwierdzić przyciskiem "Dodaj użytkownika". Jeżeli nasza lokalizacja nam nie odpowiada, albo po prostu chcemy zrezygnować z systemu należy wprowadzić dane lokalizacyjne i zatwierdzić przyciskiem "Usuń użytkownika".



IR System dostępu do widma O systemie Pobierz

Dodaj stację

Nazwa użytkownika:

Moc nadawcza: dBm ▾

Lokalizacja X: Y:

Zysk anteny nadawczej: dBi ▾

Zysk anteny odbiorczej: dBi ▾

ACLR +1(dB):

ACLR +2(dB):

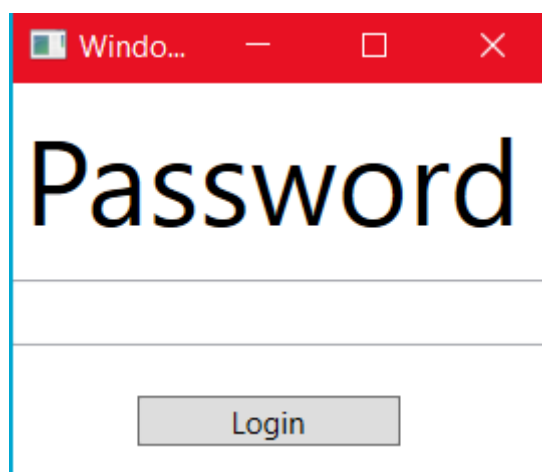
Numer kanału:

© 2020 - RadioTeamT12 - [Privacy](#)

Rys. 1.3. Dodawanie stacji.

1.4. Dodawanie stacji w aplikacji desktopowej

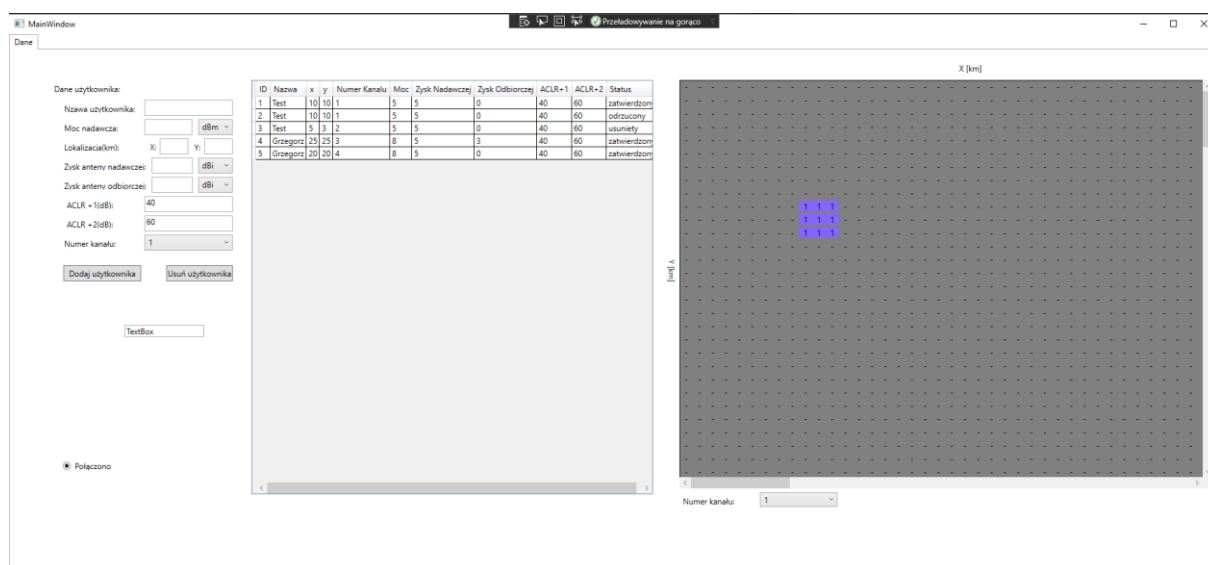
Korzystanie z aplikacji jest możliwe dopiero po wprowadzeniu danych logowania. Na rysunku 1.4 przedstawiono okno, w którym należy wprowadzić hasło użytkownika (w 1 wersji aplikacji jest jedno hasło dla wszystkich. W kolejnych wersjach po wprowadzeniu działania aplikacji webowej, hasło będzie inne dla każdego użytkownika).



Rys. 1.4. Logowanie do aplikacji desktopowej.

Po prawidłowym wprowadzeniu danych logowania ukazuje się nam okno widoczne na rysunku 1.5. W okienki podajemy dane stacji, jaką użytkownik chce umieścić na mapie. W aplikacji jest możliwość wybrania wygodnych dla nas jednostek liczbowych, które są później odpowiednio przeliczane. Wszystkie wprowadzone dane są przesyłane oraz zapisywane do odpowiedniej tabeli w bazie danych.

Na środku widnieje tabela, z której możemy odczytać użytkowników znajdujących się aktualnie w systemie oraz ich wybrane parametry. Obok tabeli znajduje się mapa, na której umiejscowieni są użytkownicy zgodnie z prowadzoną lokalizacją. Gdy najedziemy kursorem na danego użytkownika możemy zobaczyć parametry istniejących już stacji. Aktualizacja mapy odbywa się ręcznie poprzez chwilową zmianę numeru kanału. Lista użytkowników i statusów aktualizuje się automatycznie. Kliknięcie przycisku „usuń użytkownika” powoduje usunięcie użytkownika z bazy danych.



Rys. 1.5. Widok okna aplikacji desktopowej.

2. Część.

Dokumentacja techniczna

2.1. Elementy systemu

Program, Wid-Control został opracowany przy użyciu środowiska programistycznego Visual Studio Community 2019. Kod źródłowy został wykonany przy wykorzystaniu języka C#. Cały kod źródłowy został udostępniony na githubie pod adresem: <https://github.com/ASZ1997/Radiowe>.

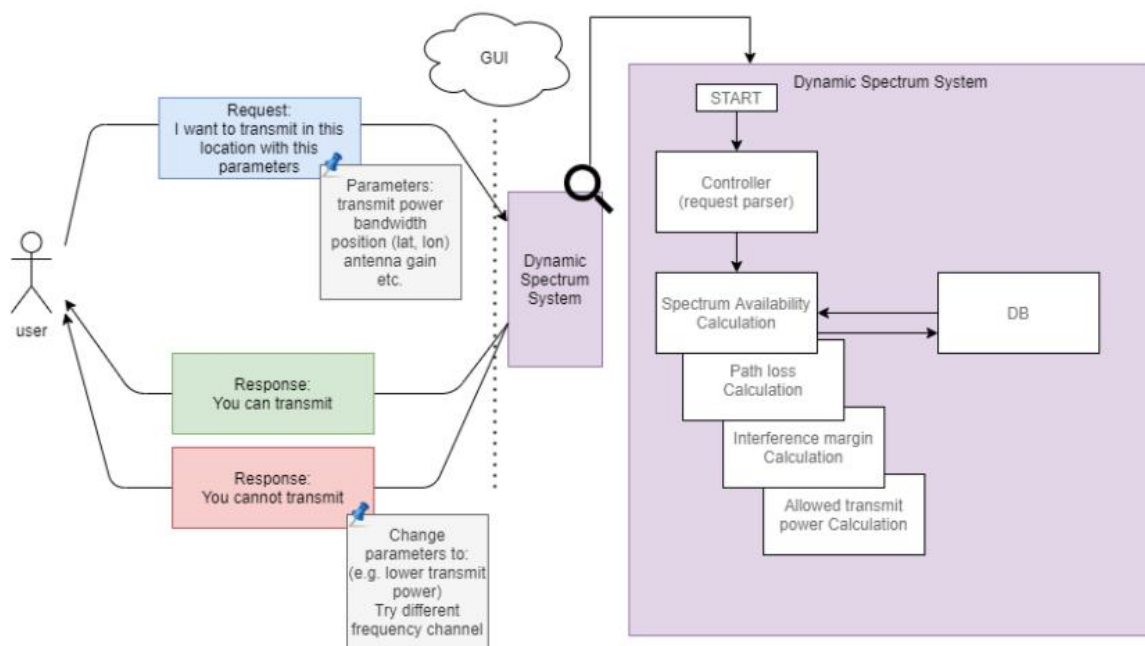
System ten składa się z dwóch elementów:

- Aplikacja webowa (w trakcie implementacji) - została zaprojektowana przy wykorzystaniu technologii ASP.NET. Jest to technologia, w której wygląd aplikacji projektuje się wykorzystując HTML i CSS, natomiast logika aplikacji może być projektowana w językach Visual Basic .NET, C# lub w dowolnym innym języku wspierającym technologię Microsoft .NET Framework. W tym przypadku wykorzystani C#. Aplikacja działa w każdej przeglądarce.
- Aplikacja desktopowa - do zaprojektowania interfejsu wykorzystano technologię WPF. WPF używa Extensible Application Markup Language (XAML). Aplikacja może działać na każdym systemie Windows.

Dla oby dwóch elementów działa jedna baza danych, która została zaimplementowana w chmurze firmy Microsoft – Azure i wykorzystuje język MSSQL. Aplikacji nie trzeba instalować, działa w wersji portable.

2.2. Ogólne działanie systemu

Na rysunku 2.1 przedstawiono ogólny schemat działania systemu. Użytkownik na początku w GUI podaje odpowiednie parametry dotyczące stawianej stacji. Po podaniu danych, zostają one wysłane do systemu kontrolera, który przekazuje te dane do bazy danych, która zaś jest połączona z obliczenia. Tam wykonuje się szereg obliczeń sprawdzających czy dana stacja może zostać zatwierdzona. Po wykonaniu się obliczeń, wysyłana jest informacja zwrotna do użytkownika o akceptacji/odrzuconiu parametrów. W momencie akceptacji stacja zostaje naniesiona na mapę.

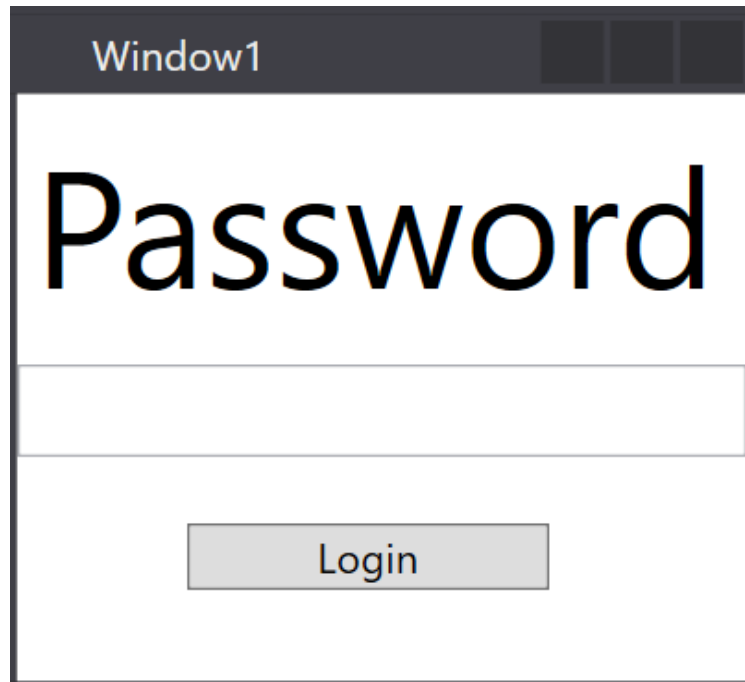


Rys. 2.1. Schemat działania projektu.

2.3. Szczegóły implementacji

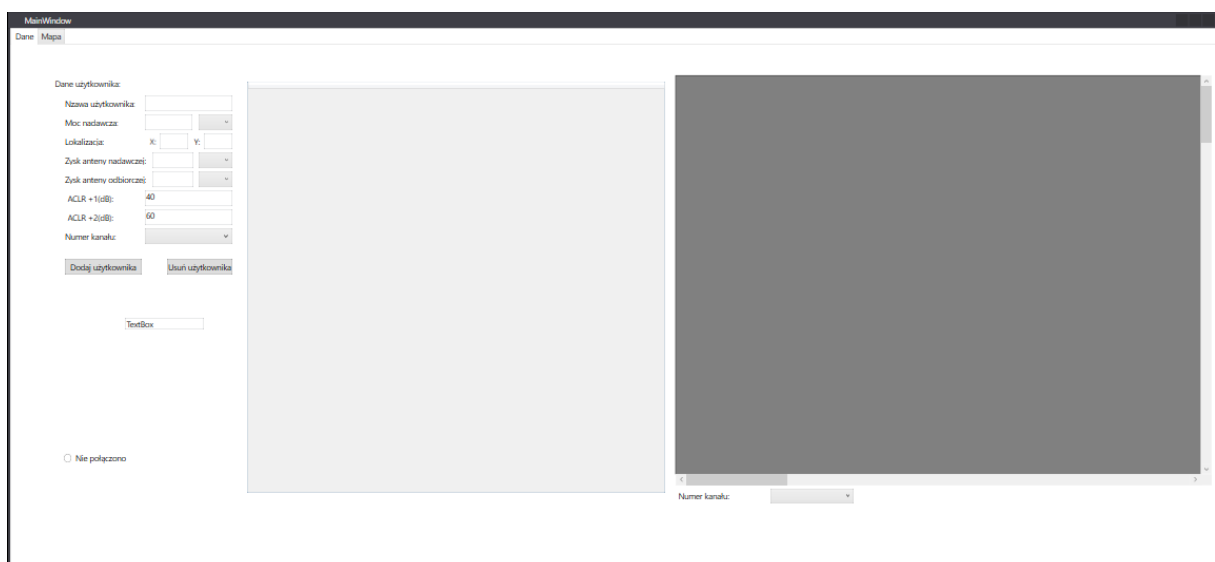
2.3.1. Interfejs użytkownika

Włączając aplikację desktopową, pojawia się najpierw okienko z rysunku 2.2. Ta funkcjonalność służy walidacji danych w aplikacji.



The image shows a simple login window titled "Window1". It has a white background and a dark gray title bar. The word "Password" is displayed in a large, bold, black font. Below it is a rectangular text input field. At the bottom center is a gray button with the word "Login" in black text.

Rys. 2.2. Okienko Logowania.



The image shows a complex software window titled "Main Window". It has two tabs: "Dane" (selected) and "Mapa". The "Dane" tab contains a form for user data with fields for "Nazwa użytkownika", "Moc nadawcy", "Lokalizacja" (with X and Y coordinates), "Zisk anteny nadawczej", "Zysk anteny odbiorczej", "ACLR +10dB", "ACLR +20dB", and "Numer kanału". There are also buttons for "Dodaj użytkownika" and "Usuń użytkownika", a "TextBox" field, and a radio button labeled "Nie połączono". The "Mapa" tab is currently empty. At the bottom right, there is a "Numer kanału" dropdown menu.

Rys. 2.3. Główne okno dodawania stacji.

Na rysunku 2.3 widać główne okno dodawania stacji w aplikacji desktopowej, które ukazuje się użytkownikowi po zalogowaniu się. Po lewej stronie znajdują się nazwy i odpowiednie okienka do wprowadzenia danych. Pod nimi znajdują się przyciski do dodawania/usuwania stacji. W środkowej części umieszczony jest DataGridView, do którego są pobierane dane o użytkownikach z bazy danych. Po prawej stronie znajduje się mapa, na której są rysowane stacje, które zostały zatwierdzone. W projekcie przyjęto domyślnie jednostki na mapie – kilometry. Według początkowych założeń projektu, nie zakładano wykorzystania skalowania mapy, stąd bardzo trudne byłoby zmienianie połowy projektu, dlatego przyjęto jedną, stałą jednostkę.

Funkcje odpowiedzialna za rysowanie i aktualizowanie mapy.

```
static Random color = new Random();
1 odwołanie
public static void plotmap(Grid Grid2, int x, int y)
{
    Grid2.RowDefinitions.Clear();
    Grid2.ColumnDefinitions.Clear();
    Grid2.Height = 200 * 20;
    Grid2.Width = 200 * 20;
    for (int i = 0; i < x; i++)
    {
        ColumnDefinition c = new ColumnDefinition();
        RowDefinition r = new RowDefinition();
        //c.Width = new GridLength(1, GridUnitType.Star);
        Grid2.ColumnDefinitions.Add(c);
        Grid2.RowDefinitions.Add(r);
    }

    for (int i = 0; i < x; i++)
    {
        //int x = 0;
        for (int j = 0; j < y; j++)
        {
            TextBlock tt = new TextBlock();
            tt.Name = string.Format("TextBlock_{0}_{1}", i, j);
            //Borde
            tt.Text = "-";
            tt.TextAlignment = TextAlignment.Center;
            //tt.HorizontalAlignment = HorizontalAlignment.Center;
            tt.VerticalAlignment = VerticalAlignment.Center;
            //tt.TextAlignment = (TextAlignment)HorizontalAlignment.Center;
            //tt.TextAlignment = System.Windows.TextAlignment.Center;
            Grid2.Children.Add(tt);
            Grid.SetRow(tt, i);
            Grid.SetColumn(tt, j);
        }
    }
}
```

Rys. 2.4. Funkcja Plotmap.

```

public static void plotStation2(Grid Grid2, DataTable name, DataTable SINR, DataTable SNR)
{
    List<string> table = new List<string>();
    List<Color> colortab = new List<Color>();
    Color actualcolor = Color.FromArgb((byte)color.Next(0, 256), (byte)color.Next(0, 256), (byte)color.Next(0, 256), 255);
    string actualname = "";
    string actualSINR = "";
    string actualSNR = "";
    for (int i = 0; i < name.Rows.Count; i++)
    {
        for (int j = 0; j < name.Columns.Count - 1; j++)
        {
            if (name.Rows[i][j].ToString() != "")
            {
                var o:UIElement = Grid2.Children[i * 200 + j];
                for (int x = 0; x < table.Count; x++)
                {
                    if (name.Rows[i][j].ToString() == table[x])
                    {
                        actualcolor = colortab[x];
                        actualname = name.Rows[i][j].ToString();
                        actualSINR = SINR.Rows[i][j].ToString();
                        actualSNR = SNR.Rows[i][j].ToString();
                        break;
                    }
                    else if (x == table.Count - 1)
                    {
                        table.Add(name.Rows[i][j].ToString());
                        actualcolor = Color.FromArgb((byte)color.Next(0, 256), (byte)color.Next(0, 256), (byte)color.Next(0, 256), 255);
                        colortab.Add(actualcolor);
                        actualname = name.Rows[i][j].ToString();
                        actualSINR = SINR.Rows[i][j].ToString();
                        actualSNR = SNR.Rows[i][j].ToString();
                        break;
                    }
                }
            }
            if (table.Count == 0)
            {
                table.Add(name.Rows[i][j].ToString());
                actualcolor = Color.FromArgb((byte)color.Next(0, 256), (byte)color.Next(0, 256), (byte)color.Next(0, 256), 255);
                colortab.Add(actualcolor);
                actualname = name.Rows[i][j].ToString();
                actualSINR = SINR.Rows[i][j].ToString();
                actualSNR = SNR.Rows[i][j].ToString();
            }

            if (o is TextBlock)
            {
                TextBlock tt = o as TextBlock;
                tt.Text = actualname;
                tt.Background = new SolidColorBrush(actualcolor);
                //tt.ToolTip = string.Format("X={0} Y={1}", x, y);
                tt.ToolTip = string.Format("X={0} Y={1} SINR={2} SNR={3}", i, j, actualSINR, actualSNR);
                //tt.ToolTip = string.Format("ID:{0} X={1} Y={2}", (int32)data.Rows[i][0], (int32)data.Rows[i][1], (int32)data.Rows[i][2]);
            }
        }
    }
}

```

Rys. 2.5. Funkcja PlotStation2.

Inteligencja w dodawaniu stacji

Jeżeli stacja nie zostanie zaakceptowana przez moduł obliczeń, zostanie wysłane kolejne zgłoszenie, najpierw ze zmniejszoną mocą nadawczą o 1 dBm, jeżeli to nie jest wystarczające, nastąpi próba dodania stacji na częstotliwości przesuniętej o 2 kanały. Jeżeli takie przesunięcie nie będzie wystarczające, zostanie podjęta trzecia, ostatnia próba z takimi samymi parametrami i przesunięciem w częstotliwości o 4 kanały w stosunku do pierwotnego wpisu. Jeżeli żaden z tych warunków nie jest wystarczający, stacja nie może zostać dodana w tym miejscu.

Wykorzystano biblioteki System.Threading w celu opóźnienia działania kodu i System.Data w celu użycia kontera DataRow i DataTable, do sprawdzenia statusu obliczeń; oraz przygotowaną wcześniej klasę DataBase i metodę AddUser aby zakolejkować kolejne próby dodania użytkownika, jeżeli pierwsza zakończyła się niepowodzeniem.

```
Thread.Sleep(2000);
DTUsers = DataBase.BaseTable("dbo.Users2");
DataRow lastRow = DTUsers.Rows[DTUsers.Rows.Count - 1];
if(lastRow["Status"].ToString() == "odrzucony"){
    var MocNadawcza_10 = (_mocNadawcza-1).ToString().Replace(',', '.');
    DataBase.AddUser(NazwaUzytkownika, LokalizacjaX, LokalizacjaY, MocNadawcza_10, ZyskAntenyN, ZyskAntenyO, NrKanal, aclr1, aclr2, status);
}
Thread.Sleep(2000);
DTUsers = DataBase.BaseTable("dbo.Users2");
lastRow = DTUsers.Rows[DTUsers.Rows.Count - 1];
if(lastRow["Status"].ToString() == "odrzucony"){
    DataBase.AddUser(NazwaUzytkownika, LokalizacjaX, LokalizacjaY, MocNadawcza, ZyskAntenyN, ZyskAntenyO, ((NrKanal+2)%10)+1, aclr1, aclr2, status);
}
Thread.Sleep(2000);
DTUsers = DataBase.BaseTable("dbo.Users2");
lastRow = DTUsers.Rows[DTUsers.Rows.Count - 1];
if(lastRow["Status"].ToString() == "odrzucony"){
    DataBase.AddUser(NazwaUzytkownika, LokalizacjaX, LokalizacjaY, MocNadawcza, ZyskAntenyN, ZyskAntenyO, ((NrKanal+4)%10)+1, aclr1, aclr2, status);
}
```

Rys. 2.6. Fragment kodu sugerujący nową stację.

Aplikacja webowa (w trakcie realizacji)

Ponizej znajduje się implementacja odpowiednich pól do wprowadzania danych.

```
<h3 class="display-4">Dodaj stację</h3>
<form method="post">
  <label for="nazwa">Nazwa użytkownika:</label> <input type="text" align="left" size="10" asp-for="Stacja.nazwa" /><br><br>

  <label for="moc">Moc nadawcza:</label>
  <input type="number" min="0" asp-for="Stacja.moc" />
  <select id="moc">
    <option value="dBm">dBm</option>
    <option value="W">W</option>
  </select><br><br>

  <label for="location">Lokalizacja:</label>
  <label for="x">X:</label>
  <input type="number" min="0" asp-for="Stacja.x" size="2" />
  <label for="y">Y:</label>
  <input type="number" min="0" asp-for="Stacja.y" size="2" /><br><br>

  <label for="zyskW">Zysk anteny nadawczej:</label>
  <input type="number" min="0" asp-for="Stacja.zyskW" />
  <select id="zyskW">
    <option value="dBi">dBi</option>
    <option value="W">W</option>
  </select><br><br>

  <label for="zysko">Zysk anteny odbiorczej:</label>
  <select id="zysko">
    <option value="dBi">dBi</option>
    <option value="W">W</option>
  </select><br><br>

  <label for="aclr1">ACLR +1(dB):</label>
  <input type="text" id="location" name="location" size="10"><br><br>

  <label for="aclr2">ACLR +2(dB):</label>
  <input type="text" id="location" name="location" size="10"><br><br>

  <label for="nrK">Numer kanału:</label>
  <input type="number" min="1" max="10" asp-for="Stacja.nrkanalu" />
  <br><br>

  <button type="submit" Click="ButtonDodajUzytkownika_Click">Dodaj użytkownika</button>
  <button type="submit">Usuń użytkownika</button>
```

Rys. 2.7. Komponenty do wprowadzania danych.

2.3.2. Baza danych

```
public static bool Open()
{
    try
    {
        String conn = string.Format("Server = {0}; Database = {3}; User ID = {1}; Password = {2}; Connection Timeout=30000; multisubnetfailover = false;",
            "radiowesystemy.database.windows.net", "przemo", "Radiowe2020", "radiowe");

        connection = new SqlConnection(conn);

        connection.Open();
        _p = true;
        return true;
    }
    catch (Exception ex)
    {
        MessageBox.Show(messageBoxText: "Nie można połączyć z serwerem baz danych.\n Skontaktuj się z Administratorem bazy danych." + ex.Message, caption: "Radio");

        _p = false;
        return false;
    }
}
```

Rys. 2.8. Funkcja odpowiedzialna za otwarcie bazy danych.

```
public static void AddUser(string NazwaUzytkownika, int LokalizacjaX, int LokalizajaY, string MocNadawcza,
    string ZyskAnteny0, string ZyskAntenyN, int NumerKanal, double aclr1, double aclr2, string status)
{
    var a:string = string.Format("INSERT dbo.Users2(Nazwa,X,Y,Moc,[Zysk Nadawczej],[Zysk Odbiorczej],[Numer Kanalu], " +
        "[ACLR+1], [ACLR+2], [Status]) VALUES ('{0}', {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, '{9}')",
        NazwaUzytkownika, LokalizacjaX, LokalizajaY, MocNadawcza, ZyskAnteny0, ZyskAntenyN, NumerKanal, aclr1, aclr2, status);
    Command(a);
    //Command(string.Format("INSERT dbo.Users2() VALUES ('{0}', {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9});",
    //NazwaUzytkownika, LokalizacjaX, LokalizajaY, NumerKanal, MocNadawcza, ZyskAntenyN, ZyskAnteny0, aclr1, aclr2, status));
}
```

Rys. 2.9. Funkcja odpowiedzialna za dodawanie stacji do bazy danych.

2.3.3. Obliczenia

Funkcja z pliku BaseStation.cs odpowiedzialna za stworzenie stacji. Funkcja przyjmuje wartości zmiennych z bazy danych.

```
public BaseStation(int x, int y, double antenna_gain, double power, int number_of_channel, double antenna_gain_receiver,
    string name, double ACLR_1, double ACLR_2, double band = 10000000)
{
    location_ = new Tuple<int, int>(x, y);
    antenna_gain_ = antenna_gain;
    power_ = power;
    band_ = band;
    channel_ = number_of_channel;
    antenna_gain_receiver_ = antenna_gain_receiver;
    name_ = name;
    aclr_1_ = ACLR_1;
    aclr_2_ = ACLR_2;
}
```

Rys. 2.10. Funkcja BaseStation.

W pliku Calculation.cs znajdują się wszystkie najważniejsze obliczenia wykonywane w programie.

Funkcja odpowiedzialna za obliczanie dystansu.

```

Odwolania: 1
public double CalculateTheDistance(double x_b, double y_b, double x_u, double y_u)
{
    the_distance_ = Math.Sqrt(Math.Pow(x_b - x_u, 2) + Math.Pow(y_b - y_u, 2));
    return the_distance_;
}

Odwolania: 2
public double CalculateTheDistance2(double x_b, double y_b, double x_u, double y_u)
{
    the_distance_2 = Math.Sqrt(Math.Pow(x_b - x_u, 2) + Math.Pow(y_b - y_u, 2));
    return the_distance_2;
}
    
```

Rys. 2.11. Funkcja obliczająca dystans.

Funkcja odpowiedzialna za obliczanie straty mocy pomiędzy anteną nadawczą i odbiorczą na określonej odległości.

```

public double CalculateFSPL(double bandwidth, double condition)
{
    // FSPL_ = 32.44d + 20 * Math.Log10(the_distance_) + 20 * Math.Log10(band);
    FSPL_ = 92.45d + 20 * Math.Log10(the_distance_) + 20 * Math.Log10(bandwidth); // distance w km, band w GHz
    if (condition == 0)
    {
        FSPL_ = 92.45d + 20 * Math.Log10(bandwidth);
    }
    return FSPL_;
}

Odwolania: 2
public double CalculateFSPL2(double bandwidth, double condition)
{
    // FSPL_ = 32.44d + 20 * Math.Log10(the_distance_) + 20 * Math.Log10(band);
    FSPL_2 = 92.45d + 20 * Math.Log10(the_distance_2) + 20 * Math.Log10(bandwidth);
    if (condition == 0)
    {
        FSPL_2 = 0;
    }
    return FSPL_2;
}
    
```

Rys. 2.12. Obliczanie straty mocy.

Funkcja odpowiedzialna za obliczanie szumu i mocy odbiornika

```
public double CalculateReceiverPower(double transmitter_power, double transmitter_gain, double receiver_gain)
{
    receiver_power = transmitter_power - FSPL_ + transmitter_gain + receiver_gain - 4; // 4 odpowiada NF (noise figure)

    return receiver_power;
}

Odwołania: 2
public double CalculateI(double transmitter_interference_power, double transmitter_interference_gain, double receiver_gain)
{
    I_ = transmitter_interference_power - FSPL_2 + transmitter_interference_gain + receiver_gain - 4;
    return I_;
}

Odwołania: 6
public double CalculateNoise(double band)
{
    N_ = -174 + 10 * Math.Log10(band);

    return N_;
}

Odwołania: 3
public double CalculateSNR_Receiver()
{
    SNR_ = receiver_power - N_;
    // SNR w dB
    return SNR_;
}
```

Rys. 2.13. Obliczanie sumy i mocy odbiornika.

Funkcja odpowiedzialna za obliczanie współczynnika SINR. Funkcja „ToReplaceSINR” wylicza nową wartość SINR po dodaniu kolejnych stacji na mapie.

```
public double CalculateSINR(double channel_diff, double ACLR1, double ACLR2)
{
    if (channel_diff == 1)
    {
        ACLR_ = ACLR1;
        I_linear = Math.Pow(10, (I_ - ACLR_) / 10) / 1000;
    }
    else if (channel_diff == 2)
    {
        ACLR_ = ACLR2;
        I_linear = Math.Pow(10, (I_ - ACLR_) / 10) / 1000;
    }
    else if (channel_diff == 0)
    {
        ACLR_ = 0;
        I_linear = Math.Pow(10, (I_ - ACLR_) / 10) / 1000;
    }
    else
    {
        I_linear = 0;
    }

    N_linear = Math.Pow(10, N_ / 10) / 1000;
    double suma = 10 * Math.Log10(N_linear + I_linear) + 30;
    SINR_ = receiver_power - suma;
    return SINR_;
}

Odwołania: 4
public double ToReplaceSINR(double old_SINR, double new_SINR)
{
    Console.WriteLine("stary sinr: " + old_SINR + " nowy sinr: " + new_SINR);
    double roznica = 10 * Math.Log10(Math.Abs(Math.Pow(10, old_SINR / 10) - Math.Pow(10, new_SINR / 10)));
    //Console.WriteLine("nowy sinr:" + new_SINR);
    //Console.WriteLine("ToReplaceSinr: " + Math.Log10(Math.Pow(10, old_SINR / 10)) + " drugi czcion " + Math.Pow(10, new_SINR / 10));
    Console.WriteLine("ROZNICA: " + roznica);
    //if(Double.IsNaN(roznica))
    //{
    //    return old_SINR;
    //}
    return roznica;
}
```

Rys. 2.14. Obliczanie SINR.

Pierwsza dodawana stacja nie sprawdza żadnych warunków – jest dodawana od razu. Każda kolejna jest sprawdzana z dodanymi stacjami i następuje dwuetapowa weryfikacja. Sprawdzamy czy nowo dodana stacja nie zagłusza już dodanych i czy dodane nie zagłuszą proponowanej do postawienia. Jeśli stacja spełni wszystkie warunki, może zostać dodana. W momencie dodania stacji do systemu, następuje aktualizacja bazy danych.