

Programowanie

1. Opis Okna LOGOWANIA



Wstęp

Aplikacja do logowania, którą opisujemy, została stworzona w technologii WPF (Windows Presentation Foundation) i korzysta z języka XAML do definiowania interfejsu użytkownika oraz C# do logiki biznesowej. Jej głównym celem jest umożliwienie użytkownikom logowania się poprzez wprowadzenie adresu e-mail oraz hasła, a następnie weryfikacja tych danych w bazie danych MySQL.

Aplikacja składa się z jednego głównego okna, które jest zaprojektowane za pomocą XAML oraz kodu-behind w C#. Główne elementy okna to:

```

private void ButtonLogowanie_Click(object sender, RoutedEventArgs e)
{
    string email = TextBoxEmail.Text;
    string password = PasswordBoxPassword.Password;

    if (AuthenticateUser(email, password))
    {
        Search search = new();
        search.Show();
        this.Close();
    }
    else
    {
        MessageBox.Show("Błędny email lub password");
    }
}

// reference
private bool AuthenticateUser(string email, string password)
{
    string query = "SELECT COUNT(*) FROM Users WHERE Email = @Email AND Password = @Password";

    try
    {
        connection.Open();
        MySqlCommand command = new(query, connection);
        command.Parameters.AddWithValue("@Email", email);
        command.Parameters.AddWithValue("@Password", password);
        int count = Convert.ToInt32(command.ExecuteScalar());
        return count > 0;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Błąd danych użytkownika " + ex.Message);
        return false;
    }
    finally
    {
        connection.Close();
    }
}

```

- Tło okna
- Etykiety tekstowe i obrazki
- Pola tekstowe do wprowadzania e-maila i hasła
- Przycisk do logowania
- Mechanizmy walidacji e-maila

Definicja Głównego Okna XAML

Oto szczegółowy opis głównego okna, które zostało zdefiniowane w pliku XAML:

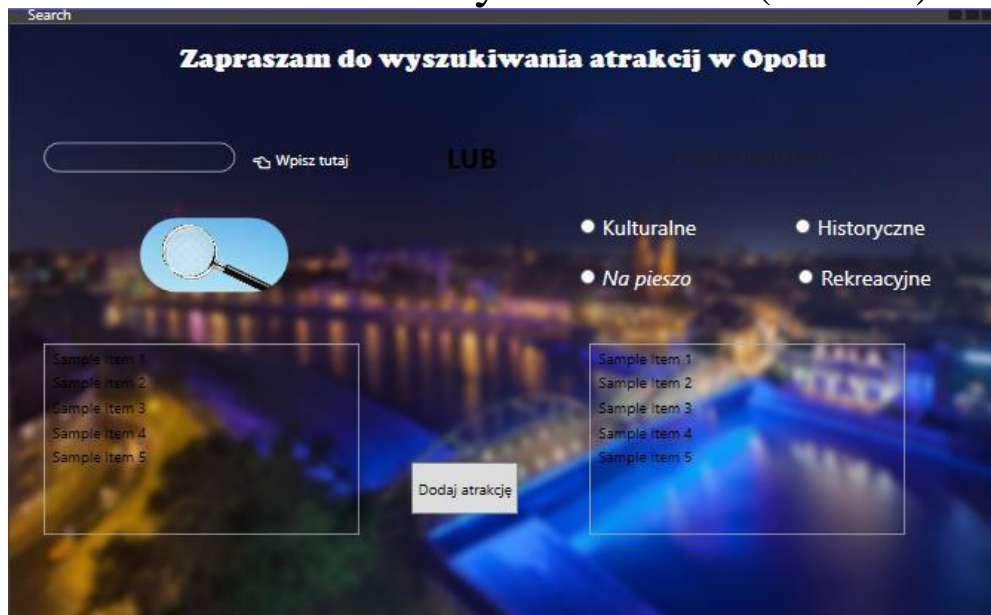
Okno główne aplikacji o tytule „LogowanieFull” ma wymiary 806x550 pikseli. Korzysta z różnych przestrzeni nazw, co umożliwia zastosowanie elementów WPF, takich jak kontrolki, style i zasoby.

W sekcji zasobów zdefiniowano style dla przycisków, pól tekstowych i hasła. Style te nadają jednolity wygląd kontrolkom, z zaokrąglonymi rogami i transparentnym tłem dla pól tekstowych.

W centralnej części okna umieszczono siatkę (Grid), która zawiera elementy UI, takie jak etykiety, obrazy, pola tekstowe i przycisk do logowania. Główne elementy to:

- **Tło:** Ustawiono obraz jako tło, który jest skalowany do rozmiaru okna.
- **Etykiety:** Są używane do wyświetlania powitań i instrukcji.
- **Obrazy:** Reprezentują herb Opola i flagę.
- **Pola tekstowe:** Pozwalają na wprowadzanie adresu e-mail i hasła. Zastosowano tutaj zaokrąglone pola tekstowe i stylizowane hasło.
- **Przycisk logowania:** Ma obraz jako tło i reaguje na kliknięcie, wywołując metodę `ButtonLogowanie_Click`.

2. Okno wyszukiwania (Search)



Okno wyszukiwania (Search.xaml) to kluczowy element aplikacji, który pozwala użytkownikom przeszukiwać bazę danych atrakcji turystycznych oraz przeglądać wyniki wyszukiwania według różnych kategorii. Kod XAML oraz logika okna przedstawiają jego działanie oraz interakcje z użytkownikiem.

2.1. Plik XAML (Search.xaml)

W pliku XAML definiujemy interfejs użytkownika dla okna wyszukiwania.

2.1.1. Layout i tło Całe okno opiera się na siatce (Grid) z tłem ustawionym jako obraz (ImageBrush) przedstawiający grafikę z Opolu

2.1.2. Pole wyszukiwania Pole tekstowe (TextBox) służy do wpisywania wyszukiwanych fraz przez użytkownika

Tło: Ustawione na szary obraz.

Szablon kontrolki: Określa zaokrąglone rogi pola tekstowego.

2.1.3. Wyniki wyszukiwania Lista wyników wyszukiwania (ListBox) prezentuje wyniki wyszukiwania wpisane w pole tekstowe

Tło: Szare tło dla listy wyników.

Zdarzenie: Obsługuje podwójne kliknięcie myszy na wynik, aby otworzyć szczegóły atrakcji.

2.1.4. Przycisk wyszukiwania Przycisk wyszukiwania (Button) inicjuje wyszukiwanie wpisanej frazy

Styl przycisku: Zaokrąglony przycisk z obrazem jako tłem.

Zdarzenie kliknięcia: Uruchamia wyszukiwanie.

2.1.5. Kategorie wyszukiwania Radiobuttony pozwalają na wybór kategorii wyszukiwania:

Zdarzenie Checked: Uruchamia filtrowanie wyników według wybranej kategorii.

2.1.6. Lista wyników według kategorii Lista wyników według kategorii (ListBox):

- **Tło:** Szare tło.
- **Zdarzenie:** Podwójne kliknięcie myszy na wynik otwiera szczegóły atrakcji.

2.1.7. Przycisk dodawania atrakcji Przycisk umożliwia przejście do okna dodawania nowej atrakcji:

- **Zdarzenie kliknięcia:** Przenosi użytkownika do okna dodawania nowych atrakcji.

2.2. Kod C# (Search.xaml.cs)

Plik C# zawiera logikę aplikacji, odpowiedzialną za interakcje z bazą danych oraz obsługę zdarzeń w oknie wyszukiwania.

2.2.1. Inicjalizacja Inicjalizacja okna oraz połączenia z bazą danych:

- **LoadPlacesFromDatabase():** Ładuje miejsca z bazy danych.

2.2.2. Ładowanie miejsc z bazy danych Metoda ładowania miejsc z bazy danych MySQL:

- **Query:** Pobiera nazwy, kategorie, opisy i inne dane miejsc.
- **Zapis do listy:** Przechowuje dane miejsc w liście places.

```
MySQLCommand command = new (query, connection);
connection.Open();

using MySQLDataReader reader = command.ExecuteReader();
{
    while (reader.Read())
    {
        string name = reader.GetString("name");
        string description = reader.GetString("description");
        string address = reader.GetString("address");
        string phone = reader.GetString("phone");
        string website = reader.GetString("website");
        string category = reader.GetString("category");

        places.Add(new string[] { name, description, address, phone, website, category });
    }
}

catch (Exception ex)
{
    MessageBox.Show($"Wystąpił błąd podczas ładowania miejsc z bazy danych: {ex.Message}");
}

finally
{
    connection.Close();
}
```

2.2.3. Wyszukiwanie miejsc Metoda obsługująca wyszukiwanie:

- **Czyszczenie listy:** Czyści listę wyników przed nowym wyszukiwaniem.
- **Filtrowanie wyników:** Dodaje wyniki, które pasują do wyszukiwanego tekstu.

```
private void ButtonSearch_Click(object sender, RoutedEventArgs e)
{
    string searchText = searchTextBox.Text.Trim();

    ListBoxResultsSearch.Items.Clear();

    foreach (string[] placeData in places)
    {
        string name = placeData[0];

        // Перевірка, чи назва місця містить введений текст пошуку
        if (name.ToLower().Contains(searchText.ToLower()))
        {
            ListBoxResultsSearch.Items.Add($" {name}");
        }
    }
}
```

2.2.4. Filtracja według kategorii Metoda ładowania miejsc według wybranej kategorii:

- **GetSelectedCategory():** Pobiera wybraną kategorię z radiobuttonów.

2.2.5. Wyświetlanie wyników według kategorii Metoda wyświetlania miejsc według kategorii:

- **Query:** Pobiera nazwy miejsc według wybranej kategorii.

2.2.6. Podwójne kliknięcie na wynik Metoda obsługująca podwójne kliknięcie na wynik wyszukiwania:

- **Otwieranie nowego okna:** Tworzy nowe okno StronyWindow z danymi wybranego miejsca.

2.2.7. Przycisk dodawania atrakcji Metoda obsługująca kliknięcie przycisku dodawania atrakcji:

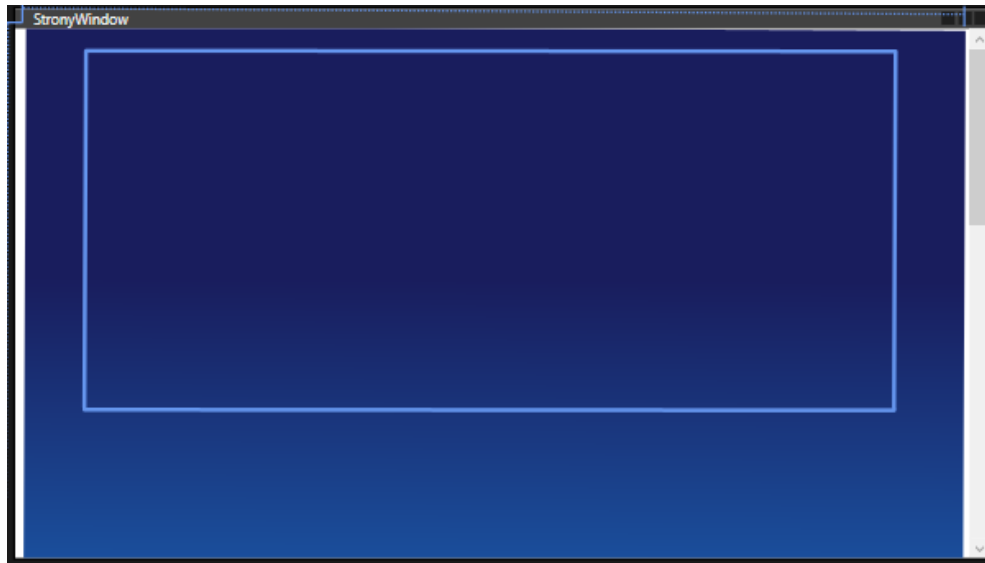
- **Otwieranie nowego okna:** Przenosi użytkownika do okna dodawania nowych atrakcji.

```
private void ButtonDodawanie_Click(object sender, RoutedEventArgs e)
{
    Dodawanie dodawanie = new (connection);
    dodawanie.Show();
}
```

3. Podsumowanie

Aplikacja WPF do wyszukiwania atrakcji turystycznych w Opolu oferuje intuicyjny interfejs umożliwiający użytkownikom wyszukiwanie oraz przeglądanie atrakcji według różnych kategorii. Implementacja oparta na MySQL zapewnia elastyczność w przechowywaniu i zarządzaniu danymi. Aplikacja jest przykładem efektywnego wykorzystania technologii WPF oraz bazy danych MySQL do tworzenia funkcjonalnych i interaktywnych aplikacji desktopowych.

3. Analiza kodu i opis funkcji okna "StronyWindow"



Kod XAML

W kodzie XAML definiujemy interfejs użytkownika oraz układ elementów w oknie StronyWindow. Kluczowe elementy interfejsu obejmują:

2.1. ScrollView

ScrollView pozwala na przewijanie zawartości okna, jeżeli nie mieści się ona na ekranie:

2.2. Grid

Grid stanowi główny kontener dla elementów interfejsu. Posiada gradientowe tło:

2.3. Obrazek

Obrazek, który będzie wyświetlany jako główny obrazek atrakcji:

2.4. Etykiety

Etykiety (Labels) wyświetlają różne informacje na temat atrakcji:

2.5. Mapa

Mapa z komponentem GMapControl wyświetlająca lokalizację atrakcji:

2.6. Pole tekstowe

Pole tekstowe wyświetla opis atrakcji:

3. Kod C# (StronyWindow.xaml.cs)

Kod C# odpowiada za logikę działania okna, w tym inicjalizację elementów interfejsu, wyświetlanie informacji oraz obsługę komponentu mapy.

3.1. Konstruktor

```
2 references
public StronyWindow(string title, string description, string adress1, string numer, string link, string imageName)
{
    InitializeComponent();
    InicjalizujMape();
    // Dodanie markera
    AddMarker(50.667065, 17.918001);
    // Dodatkowi dzii dla встановлення заголовку вікна та зображення
    LabelHead.Content = title;
    TextBoxDescription.Text = description;
    LabelAdres.Content = $"Address: {adress1}";
    LabelNumer.Content = $"Numer: {numer}";
    LabelLink.Content = link;
    ImageMain.Source = new BitmapImage(new Uri(imageName, UriKind.Relative));
}
```

Konstruktor przyjmuje parametry takie jak tytuł, opis, adres, numer kontaktowy, link i nazwę obrazka, a następnie inicjalizuje okno:

- **InicjalizujMape():** Metoda inicjalizująca mapę.
- **AddMarker():** Metoda dodająca marker na mapę.

3.2. Właściwości publiczne

Właściwości publiczne umożliwiają dostęp do etykiety `LabelHead` i obrazka `ImageMain` z zewnątrz:

3.3. Inicjalizacja mapy

Metoda `InicjalizujMape` konfiguruje mapę, ustawiając m.in. dostawcę mapy, poziom zbliżenia oraz pozycję początkową:

- **MapProvider:** Ustawia dostawcę mapy na Google Map.
- **Position:** Ustawia początkową pozycję mapy na współrzędne Opola.
- **ShowCenter:** Ukrywa wskaźnik środka mapy.
- **DragButton:** Ustawia przycisk myszy do przesuwania mapy.

3.4. Dodawanie markera

- **PointLatLng:** Tworzy obiekt lokalizacji na podstawie współrzędnych.
- **GMapMarker:** Tworzy nowy marker na mapie.
- **Offset:** Centruje obraz markera w punkcie lokalizacji.
- **Markers.Add():** Dodaje marker do mapy.

```
// Tworzenie markera
var marker = new GMapMarker(location);
// Ustawienie ikony dla markera (opcjonalnie)
var markerImage = new BitmapImage(new Uri("C:/Users/yarem/source/repos/WpfApp2/pinezka-new.png", UriKind.Absolute)); // Ścieżka do obrazu
var image = new Image { Source = markerImage, Width = 26, Height = 26 }; // Możesz dostosować rozmiar obrazu

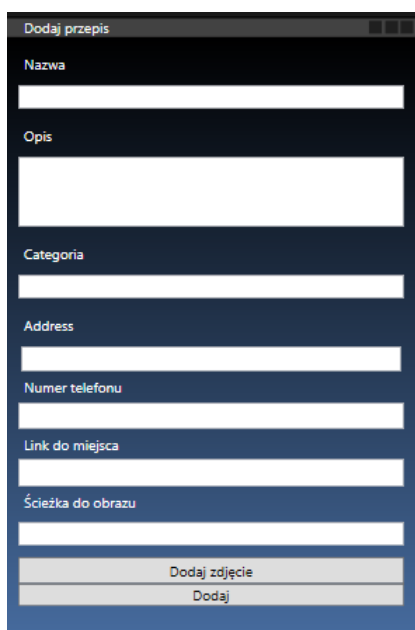
// Dodanie obrazu do markera
marker.Shape = image;
marker.Offset = new System.Windows.Point(-image.Width / 2, -image.Height / 2); // Centrowanie obrazu na współrzędnych

// Dodanie markera do mapy
mapControl.Markers.Add(marker);
```


4. Podsumowanie

`StronyWindow` to okno w aplikacji WPF, które umożliwia wyświetlanie szczegółów wybranej atrakcji turystycznej, w tym tytułu, opisu, danych kontaktowych oraz lokalizacji na mapie. Okno jest zrealizowane w XAML, a jego funkcjonalności są zaimplementowane w kodzie C#. Użycie komponentu GMap.NET pozwala na integrację mapy Google i dodawanie markerów, co zwiększa interaktywność aplikacji.

4. Opis i analiza kodu okna "Dodawanie" do dodawania miejsc w aplikacji WPF



1. Wstęp

Okno `Dodawanie` to interfejs aplikacji WPF, umożliwiający użytkownikowi dodanie nowego miejsca do bazy danych MySQL. Okno pozwala na wprowadzenie informacji o miejscu, takich jak nazwa, opis, kategoria, adres, numer telefonu, link do strony oraz ścieżka do obrazu. Interfejs użytkownika jest zrealizowany za pomocą XAML, a logika aplikacji jest zaimplementowana w C#.

2. Kod XAML

2.1. Ogólne ustawienia okna

Okno `Dodawanie` jest zdefiniowane w pliku XAML i zawiera kilka elementów, które tworzą formularz do wprowadzania danych o nowym miejscu. Zastosowano gradientowe tło, aby nadać oknu estetyczny wygląd:

2.2. ScrollView

`ScrollView` zapewnia przewijanie zawartości okna, co jest przydatne przy dużej liczbie elementów:

2.3. StackPanel

`StackPanel` grupuje wszystkie elementy formularza i układa je jeden pod drugim:

3. Kod C# (Dodawanie.xaml.cs)

3.1. Konstruktor

```
using MySqlCommand command = new (query, connection);
{
    command.Parameters.AddWithValue("@name", name);
    command.Parameters.AddWithValue("@address", address);
    command.Parameters.AddWithValue("@category", category);
    command.Parameters.AddWithValue("@description", description);
    command.Parameters.AddWithValue("@photo", imageBytes);

    // Set phone number parameter only if not empty
    if (!string.IsNullOrEmpty(phoneNumber))
        command.Parameters.AddWithValue("@phone", phoneNumber);
    else
        command.Parameters.AddWithValue("@phone", DBNull.Value); // Set to DBNull if empty

    // Set website parameter only if not empty and starts with "www."
    if (!string.IsNullOrEmpty(website) && website.StartsWith("www."))
        command.Parameters.AddWithValue("@website", website);
    else
        command.Parameters.AddWithValue("@website", DBNull.Value); // Set to DBNull if empty or doesn't start with "www."
}
```

Konstruktor przyjmuje obiekt `MySQLConnection`, który jest używany do komunikacji z bazą danych. Ustawia również domyślną wartość dla `selectedImagePath`:

3.2. Obsługa zapisu danych

Metoda `SaveRecipeButton_Click` obsługuje zapisywanie nowego miejsca do bazy danych. Sprawdza, czy wszystkie wymagane pola są wypełnione, oraz czy numer telefonu ma prawidłowy format:

3.3. Dodawanie zdjęcia

Metoda `DodajButton_Click_1` umożliwia użytkownikowi wybranie pliku obrazu z systemu plików:

3.4. Walidacja numeru telefonu

Metoda `PhoneNumberTextBox_TextChanged` ogranicza długość wprowadzanego numeru telefonu do 10 cyfr:

4. Podsumowanie

Kod zapewnia funkcjonalność dodawania nowego miejsca do bazy danych z walidacją wprowadzanych danych oraz możliwość dodania obrazu do opisu miejsca. Użycie `MySQLConnection` do obsługi bazy danych i możliwość przeglądania plików w systemie operacyjnym sprawia, że aplikacja jest wszechstronna i przyjazna dla użytkownika. Implementacja obejmuje również zarządzanie błędami, co zwiększa stabilność i niezawodność aplikacji.