SPRAWOZDANIE PLATFORMY PROGRAMISTYCZNE .NET I JAVA – LAB2

POLITECHNIKA WROCŁAWSKA

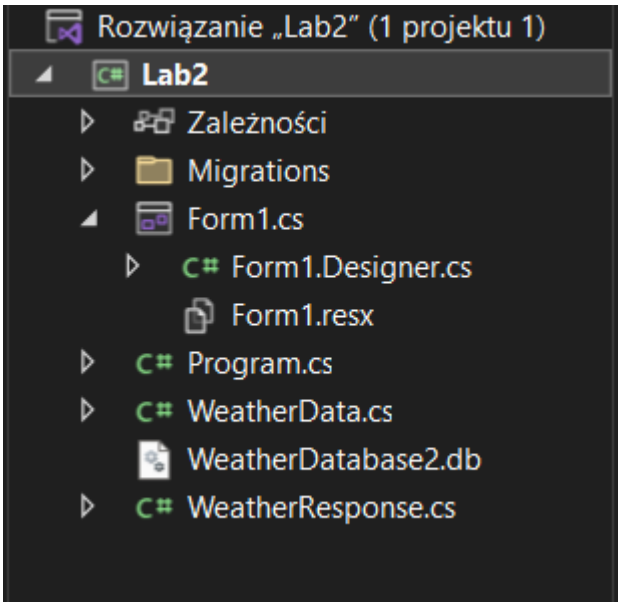
WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI

INFORMATYCZNE SYSTEMY AUTOMATYKI

MICHAŁ WYRZYKOWSKI

INDEKS 2642

1. Kod C#, drzewo projektu oraz aplikacja okienkowa




```
private async void button1_Click(object sender, EventArgs e)
{
    if (float.TryParse(textBox2.Text, out float latitude) && float.TryParse(textBox3.Text, out float longitude))
    {
        var call = $"https://api.openweathermap.org/data/2.5/weather?lat={latitude}&lon={longitude}&appid=184f86296734df65eb5199f43ab1d193";
        string response = await client.GetStringAsync(call);
        var weatherApiResponse = JsonSerializer.Deserialize<WeatherResponse>(response);

        using (var dbContext = new WeatherBase())
        {
            var newMain = new WeatherBaseInfo
            {
                CityName = weatherApiResponse.name,
                Country = weatherApiResponse.sys.country,
                Latitude = weatherApiResponse.coord.lat,
                Longitude = weatherApiResponse.coord.lon,
                Timezone = weatherApiResponse.timezone,
                Temperature = weatherApiResponse.main.temp,
                Pressure = weatherApiResponse.main.pressure,
                Humidity = weatherApiResponse.main.humidity,
                WindSpeed = weatherApiResponse.wind.speed,
                WindDirection = weatherApiResponse.wind.deg,
                WeatherCondition = weatherApiResponse.weather[0].main,
                DateAdded = DateTime.UtcNow
            };

            dbContext.WeatherData.Add(newMain);
            await dbContext.SaveChangesAsync();
        }

        UpdateWeatherInfo(latitude, longitude);
    }
}
```

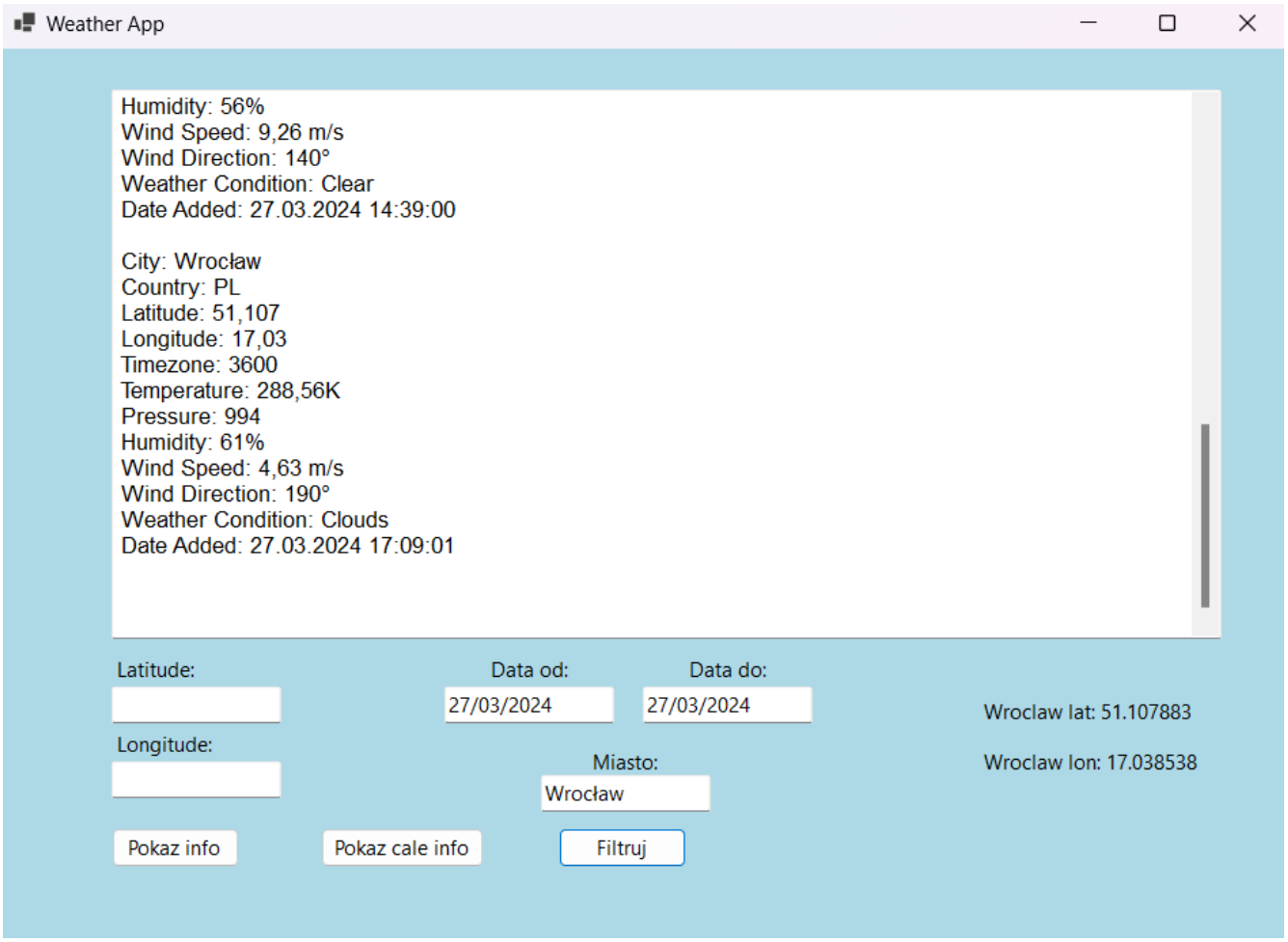
```
private void button2_Click(object sender, EventArgs e)
{
    DisplayAllWeatherData();
}
1 odwołanie
private void DisplayAllWeatherData()
{
    textBox1.Clear();
    using (var dbContext = new WeatherBase())
    {
        var weatherDataList = dbContext.WeatherData.ToList();
        foreach (var weatherData in weatherDataList)
        {
            textBox1.AppendText($"City: {weatherData.CityName}" + Environment.NewLine);
            textBox1.AppendText($"Country: {weatherData.Country}" + Environment.NewLine);
            textBox1.AppendText($"Latitude: {weatherData.Latitude}" + Environment.NewLine);
            textBox1.AppendText($"Longitude: {weatherData.Longitude}" + Environment.NewLine);
            textBox1.AppendText($"Timezone: {weatherData.Timezone}" + Environment.NewLine);
            textBox1.AppendText($"Temperature: {weatherData.Temperature}K" + Environment.NewLine);
            textBox1.AppendText($"Pressure: {weatherData.Pressure}" + Environment.NewLine);
            textBox1.AppendText($"Humidity: {weatherData.Humidity}%" + Environment.NewLine);
            textBox1.AppendText($"Wind Speed: {weatherData.WindSpeed} m/s" + Environment.NewLine);
            textBox1.AppendText($"Wind Direction: {weatherData.WindDirection}°" + Environment.NewLine);
            textBox1.AppendText($"Weather Condition: {weatherData.WeatherCondition}" + Environment.NewLine);
            textBox1.AppendText($"Date Added: {weatherData.DateAdded}" + Environment.NewLine);
            textBox1.AppendText(Environment.NewLine);
        }
    }
}
```

```
private void DisplayFilteredWeatherData(DateTime? startDate, DateTime? endDate, string cityFilter)
{
    textBox1.Clear();
    using (var dbContext = new WeatherBase())
    {
        var filteredData = dbContext.WeatherData.AsQueryable();

        if (startDate.HasValue && endDate.HasValue)
        {
            filteredData = filteredData.Where(item => item.DateAdded.Date >= startDate.Value.Date && item.DateAdded.Date <= endDate.Value.Date);
        }

        if (!string.IsNullOrWhiteSpace(cityFilter))
        {
            filteredData = filteredData.Where(item => item.CityName.ToLower().Contains(cityFilter.ToLower()));
        }

        var weatherDataList = filteredData.ToList();
        foreach (var weatherData in weatherDataList)
        {
            textBox1.AppendText($"City: {weatherData.CityName}" + Environment.NewLine);
            textBox1.AppendText($"Country: {weatherData.Country}" + Environment.NewLine);
            textBox1.AppendText($"Latitude: {weatherData.Latitude}" + Environment.NewLine);
            textBox1.AppendText($"Longitude: {weatherData.Longitude}" + Environment.NewLine);
            textBox1.AppendText($"Timezone: {weatherData.Timezone}" + Environment.NewLine);
            textBox1.AppendText($"Temperature: {weatherData.Temperature}K" + Environment.NewLine);
            textBox1.AppendText($"Pressure: {weatherData.Pressure}" + Environment.NewLine);
            textBox1.AppendText($"Humidity: {weatherData.Humidity}%" + Environment.NewLine);
            textBox1.AppendText($"Wind Speed: {weatherData.WindSpeed} m/s" + Environment.NewLine);
            textBox1.AppendText($"Wind Direction: {weatherData.WindDirection}°" + Environment.NewLine);
            textBox1.AppendText($"Weather Condition: {weatherData.WeatherCondition}" + Environment.NewLine);
            textBox1.AppendText($"Date Added: {weatherData.DateAdded}" + Environment.NewLine);
            textBox1.AppendText(Environment.NewLine);
        }
    }
}
```

```
private void button3_Click(object sender, EventArgs e)
{
    bool isStartDateValid = DateTime.TryParse(textBox4.Text, out DateTime startDate);
    bool isEndDateValid = DateTime.TryParse(textBox5.Text, out DateTime endDate);
    bool isCityFilterActive = !string.IsNullOrWhiteSpace(textBox6.Text.Trim());
    if ((isStartDateValid && isEndDateValid) || isCityFilterActive)
    {
        if (isStartDateValid && isEndDateValid)
        {
            if (isCityFilterActive)
            {
                string cityFilter = textBox6.Text.Trim();
                DisplayFilteredWeatherData(startDate, endDate, cityFilter);
            }
            else
            {
                DisplayFilteredWeatherData(startDate, endDate, null);
            }
        }
        else if (isCityFilterActive)
        {
            string cityFilter = textBox6.Text.Trim();
            DisplayFilteredWeatherData(null, null, cityFilter);
        }
    }
    else
    {
        MessageBox.Show("Please enter at least one filter (start date, end date, or city name).");
    }
}
```

```
public class WeatherResponse
{
    Odwołania: 2
    public Coord coord { get; set; }
    1 odwołanie
    public List<Weather> weather { get; set; }
    Odwołania: 3
    public Main main { get; set; }
    Odwołania: 2
    public Wind wind { get; set; }
    Odwołania: 0
    public Clouds clouds { get; set; }
    1 odwołanie
    public Sys sys { get; set; }
    Odwołania: 0
    public int visibility { get; set; }
    1 odwołanie
    public int timezone { get; set; }
    Odwołania: 0
    public long dt { get; set; }
    Odwołania: 0
    public int id { get; set; }
    1 odwołanie
    public string name { get; set; }
    Odwołania: 0
    public int cod { get; set; }
}

1 odwołanie
public class Coord
{
    1 odwołanie
    public float lon { get; set; }
    1 odwołanie
    public float lat { get; set; }
}
```

```
internal class WeatherBase : DbContext
{
    Odwołania: 5
    public DbSet<WeatherBaseInfo> WeatherData { get; set; }

    Odwołania: 0
    public WeatherBase(DbSet<WeatherBaseInfo> weatherData)
    {
        WeatherData = weatherData;
    }

    Odwołania: 4
    public WeatherBase()
    {
        Database.EnsureCreated();
    }

    Odwołania: 0
    protected override void OnConfiguring(DbContextOptionsBuilder options)
    {
        options.UseSqlite(@"Data Source=WeatherDatabase2.db");
    }
}
Odwołania: 3
public class WeatherBaseInfo
{
    [Key]
    Odwołania: 0
    public int Id { get; set; }
    Odwołania: 5
    public string? CityName { get; set; }
    Odwołania: 4
    public string? Country { get; set; }
    Odwołania: 5
    public required float Latitude { get; set; }
    Odwołania: 5
    public required float Longitude { get; set; }
    Odwołania: 4
    public required int Timezone { get; set; }
    Odwołania: 4
    public required float Temperature { get; set; }
    Odwołania: 4
    public required int Pressure { get; set; }
    Odwołania: 4
    public required int Humidity { get; set; }
    Odwołania: 4
    public required float WindSpeed { get; set; }
    Odwołania: 4
    public required int WindDirection { get; set; }
    Odwołania: 4
    public required string WeatherCondition { get; set; }
    Odwołania: 6
    public required DateTime DateAdded { get; set; }
}
```



2. Opis działania

Form1.cs: Jest to główny plik formularza, który definiuje zachowanie interfejsu użytkownika.

- Przy użyciu HttpClient wysyła zapytanie do API OpenWeatherMap, aby uzyskać aktualne dane pogodowe dla podanych współrzędnych geograficznych.
- Dane odpowiedzi są deserializowane przy użyciu biblioteki System.Text.Json.
- Dane pogodowe są zapisywane do lokalnej bazy danych SQLite za pomocą Entity Framework Core.
- Umożliwia użytkownikowi wyświetlanie wszystkich danych pogodowych z bazy danych oraz filtrowanie danych na podstawie daty i/lub nazwy miasta.
- Interfejs użytkownika został skonfigurowany w metodach InitializeUI, Form1_Resize, button1_Click, button2_Click, button3_Click, DisplayAllWeatherData, UpdateWeatherInfo, DisplayFilteredWeatherData oraz odpowiednich obsługach zdarzeń TextBox i Label.

WeatherBase.cs: Jest to klasa bazowa kontekstu bazy danych, która dziedziczy po DbContext z Entity Framework Core.

- Zawiera definicję DbSet dla encji WeatherBaseInfo, które reprezentują dane pogodowe.

WeatherBaseInfo.cs: Jest to klasa definiująca model danych pogodowych, które będą przechowywane w bazie danych.

WeatherResponse.cs: Zawiera definicje klas odpowiadających za deserializację odpowiedzi z API OpenWeatherMap do obiektów C#.

3. Charakterystyka funkcji używanych w pliku Form1.cs

- InitializeUI(): Ta funkcja inicjalizuje interfejs użytkownika, ustawiając tytuł okna, tło i stylizując TextBox.

- Form1_Resize(object sender, EventArgs e): Ta funkcja jest wywoływana, gdy użytkownik zmienia rozmiar okna. Dostosowuje rozmiar TextBox, aby pasował do nowego rozmiaru okna.

- button1_Click(object sender, EventArgs e): Ta funkcja jest wywoływana po naciśnięciu przycisku "Pobierz pogodę". Pobiera dane pogodowe dla podanych współrzędnych geograficznych, zapisuje je do bazy danych i wywołuje funkcję UpdateWeatherInfo().

- button2_Click(object sender, EventArgs e): Ta funkcja jest wywoływana po naciśnięciu przycisku "Pokaż wszystkie dane". Wyświetla wszystkie dane pogodowe przechowywane w bazie danych.

- DisplayAllWeatherData(): Ta funkcja wyświetla wszystkie dane pogodowe przechowywane w bazie danych w polu tekstowym.

- UpdateWeatherInfo(float latitude, float longitude): Ta funkcja aktualizuje pole tekstowe, wyświetlając dane pogodowe dla podanych współrzędnych geograficznych zapisane w bazie danych.

- button3_Click(object sender, EventArgs e): Ta funkcja jest wywoływana po naciśnięciu przycisku "Filtruj". Filtruje dane pogodowe w zależności od podanych kryteriów (daty i/lub nazwy miasta) i wyświetla je w polu tekstowym.

- DisplayFilteredWeatherData(DateTime? startDate, DateTime? endDate, string cityFilter): Ta funkcja wyświetla dane pogodowe, które zostały przefiltrowane na podstawie podanej daty rozpoczęcia, daty zakończenia i/lub nazwy miasta w polu tekstowym.