Sprawozdanie z Projektu

Mikołaj Rusiecki, Patryk Kasztelan

1 Opis Projektu

Projekt jest aplikacją do analizy i wizualizacji danych pogodowych. Aplikacja pozwala użytkownikom na przeglądanie i analizowanie danych pogodowych na podstawie wybranej lokalizacji. Dane te są pobierane z zewnętrznego API pogodowego oraz przechowywane w bazie danych, co umożliwia ich dalszą obróbkę i analizę.

Główne funkcje aplikacji obejmują:

- Pobieranie danych pogodowych z zewnętrznego API.
- Przechowywanie danych w lokalnej bazie danych.
- Wizualizacja danych w formie wykresów.
- Filtracja i analiza danych według wybranych kryteriów.

2 Wymagania Funkcjonalne

1. **Pobieranie Danych Pogodowych:**
   - Aplikacja powinna umożliwiać pobieranie danych pogodowych z zewnętrznego API na podstawie podanej lokalizacji.
   - Dane pogodowe powinny obejmować zarówno prognozy dzienne, godzinne, jak i minutowe.
2. **Przechowywanie Danych:**
   - Aplikacja powinna przechowywać pobrane dane pogodowe w lokalnej bazie danych.
   - Dane powinny być organizowane według lokalizacji, daty i godziny.
3. **Wizualizacja Danych:**
   - Aplikacja powinna samodzielnie tworzyć wykresy na podstawie danych pogodowych.
   - Wykresy powinny umożliwiać wizualizację różnych parametrów pogodowych, takich jak temperatura, opady, prędkość wiatru itp.

4. **Filtracja i Analiza Danych:**
    - Użytkownik powinien mieć możliwość filtrowania danych według lokalizacji.
5. **Interfejs Użytkownika:**
    - Aplikacja powinna posiadać intuicyjny i przyjazny dla użytkownika interfejs, umożliwiający łatwe nawigowanie i korzystanie z funkcji aplikacji.

3 Wymagania Niefunkcjonalne

1. **Wydajność:**
    - Aplikacja powinna działać płynnie i szybko, nawet przy dużych ilościach danych.
    - Czas odpowiedzi na zapytania użytkowników powinien być minimalny.
2. **Skalowalność:**
    - System powinien być skalowalny, aby mógł obsłużyć zwiększoną liczbę użytkowników i większe wolumeny danych w przyszłości.
3. **Bezpieczeństwo:**
    - Dane przechowywane w bazie danych powinny być zabezpieczone przed nieautoryzowanym dostępem.
    - Aplikacja powinna być odporna na typowe ataki, takie jak SQL Injection.
4. **Niezawodność:**
    - Aplikacja powinna być niezawodna i dostępna dla użytkowników przez większość czasu (wysoki uptime).
    - System powinien być odporny na awarie i błędy.
5. **Łatwość Utrzymania:**
    - Kod aplikacji powinien być dobrze udokumentowany i zorganizowany, aby umożliwić łatwe utrzymanie i rozwijanie projektu.
    - Aplikacja powinna być łatwa do aktualizacji i modernizacji.

4 Struktura Projektu

Projekt został stworzony przy użyciu technologii C# oraz frameworka .NET. Aplikacja jest aplikacją Blazor, która umożliwia tworzenie interaktywnych interfejsów użytkownika przy użyciu C# zamiast języka JavaScript.

4.1 Technologie

- **C#**: Język programowania używany do tworzenia logiki aplikacji.
- **.NET**: Framework umożliwiający tworzenie aplikacji wieloplatformowych, w tym aplikacji webowych, desktopowych oraz mobilnych.
- **Blazor**: Framework umożliwiający tworzenie interaktywnych aplikacji webowych przy użyciu C#.
- **Entity Framework Core**: ORM (Object-Relational Mapper) używany do komunikacji z bazą danych.
- **Nominatim**: API do geokodowania i odwrotnego geokodowania, używane do przekształcania współrzędnych geograficznych na adresy i odwrotnie.

4.2 Zawartość Projektu

1. **Components**:
   - **Layout**:
     - 'MainLayout.razor': Główny layout aplikacji definiujący ogólny układ strony.
     - 'NavMenu.razor': Komponent nawigacyjny zawierający menu nawigacyjne aplikacji.
   - **Pages**:
     - 'ApiConnection.razor': Strona zarządzająca połączeniem z API pogodowym.
     - 'CurrentWeather.razor': Strona wyświetlająca bieżącą pogodę.
     - 'Weather.razor': Strona główna dla danych pogodowych.
     - 'WeatherDay.razor': Strona wyświetlająca pogodę dzienną.
     - 'WeatherHour.razor': Strona wyświetlająca pogodę godzinową.
     - 'WeatherMinutely.razor': Strona wyświetlająca pogodę minutową.
     - 'Routes.razor': Plik konfiguracyjny dla routingu w aplikacji Blazor.
2. **Data**:
   - 'DbContext.cs': Plik konfigurujący kontekst bazy danych i zarządzający połączeniami z bazą danych.
3. **Migrations**:
   - Katalog zawierający migracje Entity Framework do zarządzania schematem bazy danych.
4. **Model**:

- **Klasy**:
  - ‘Address.cs‘: Klasa reprezentująca przychodzące dane z API dotyczące lokalizacji.
  - ‘Alerts.cs‘: Klasa reprezentująca alerty pogodowe.
  - ‘CurrentWeather.cs‘: Klasa reprezentująca bieżącą pogodę.
  - ‘Location.cs‘: Klasa reprezentująca lokalizację.
  - ‘NominatimResponseGeocode.cs‘: Klasa do mapowania odpowiedzi geokodowania z Nominatim.
  - ‘NominatimResponseReverseGeocode.cs‘: Klasa do mapowania odpowiedzi odwrotnego geokodowania z Nominatim.
  - ‘Temperature.cs‘: Klasa reprezentująca temperatury.
  - ‘Weather.cs‘: Klasa reprezentująca dane pogodowe.
  - ‘WeatherDaily.cs‘: Klasa reprezentująca dane pogodowe dzienne.
  - ‘WeatherDataResult.cs‘: Klasa reprezentująca wynik zapytania o dane pogodowe.
  - ‘WeatherHourly.cs‘: Klasa reprezentująca dane pogodowe godzinowe.
  - ‘WeatherMinutely.cs‘: Klasa reprezentująca dane pogodowe minutowe.
  - ‘WeatherResponse.cs‘: Klasa do mapowania odpowiedzi z API pogodowego.

5. **Services**:
   - ‘ReadDataBaseService.cs‘: Serwis do odczytu danych z bazy danych.
   - ‘WeatherService.cs‘: Serwis do zarządzania operacjami związanymi z danymi pogodowymi w bazie danych.