Aplikacja Pogodowa - Dokumentacja Techniczna

**1. Wprowadzenie**

**1.1 Cel dokumentacji**

Celem tej dokumentacji jest dostarczenie kompleksowego opisu struktury, funkcji, oraz konfiguracji Aplikacji Pogodowej opartej na platformie ASP.NET, która integruje się z  zewnętrznym serwisem pogodowym.

**1.2 Opis projektu**

Aplikacja Pogodowa umożliwia użytkownikom sprawdzanie aktualnych warunków pogodowych dla różnych lokalizacji. Wykorzystuje zewnętrzny serwis pogodowy do pobierania informacji o temperaturze, wilgotności, prędkości wiatru itp.

**1.3 Zakres aplikacji**

Aplikacja pozwala użytkownikom:

* Wybierać lokalizację.
* Przeglądać aktualne warunki pogodowe.

**2. Architektura Aplikacji**

**2.1 Architektura warstwowa**

Aplikacja opiera się na architekturze warstwowej, zawierającej:

* Warstwę prezentacji (MVC)
* Warstwę dostępu do danych (API Pogodowe)

**2.2 Wybór wzorca projektowego**

Wybrany wzorzec to MVC (Model-View-Controller) dla klarownego rozdzielenia odpowiedzialności pomiędzy modele, widoki a kontrolery.

**2.3 Technologie użyte w architekturze**

* ASP.NET Core MVC
* Entity Framework Core (opcjonalnie, w przypadku przechowywania lokalizacji użytkownika)

**3. Integracja z Zewnętrznym Serwisem Pogodowym**

**3.1 Wybór serwisu pogodowego**

Wykorzystano serwis pogodowy OpenWeatherMap, który dostarcza API do pobierania informacji pogodowych.

**3.2 Konfiguracja klucza API**

W pliku konfiguracyjnym (appsettings.json) przechowywany jest klucz API serwisu pogodowego, który jest wykorzystywany do autoryzacji zapytań API.

**4. Struktura Katalogów**

**4.1 Struktura projektu**

* /Controllers: Kontrolery obsługujące żądania HTTP.
* /Models: Modele danych, takie jak WeatherModel do przechowywania informacji o  pogodzie.
* /Views: Widoki odpowiedzialne za prezentację danych.
* /Configuration: wykorzystywany klucz API

**4.2 Opis katalogów i plików**

* /wwwroot: Zawiera pliki statyczne, takie jak arkusze stylów, skrypty JavaScript.
* /Services: Klasa WeatherService odpowiedzialna za komunikację z serwisem pogodowym.

**5. Warstwa Modelu**

**5.1 Struktura modeli danych**

* City: Klasa reprezentująca dane pogodowe, takie jak temperatura, wilgotność.
* SearchByCity: Klasa reprezentująca wpisywane dane przez użytkownika
  1. **Logika biznesowa**

Logika biznesowa koncentruje się na interakcji z serwisem pogodowym i przetwarzaniu danych.

**5.3 Zarządzanie stanem**

Stan aplikacji jest utrzymywany w modelach danych, takich jak WeatherModel.

**6. Warstwa Kontrolerów**

**6.1 Obsługa zapytań HTTP**

* HomeController: Odpowiada za wyświetlanie strony głównej oraz obsługę formularzy wyboru lokalizacji.
* WeatherForecastController: Zarządza zapytaniami do serwisu pogodowego i przekazuje dane do widoków.

**6.2 Logika kontrolerów**

* Kontrolery obsługują zapytania HTTP, wywołując odpowiednie metody w warstwie biznesowej.

**6.3 Walidacja danych wejściowych**

* Prosta walidacja danych wejściowych, takich jak wybór lokalizacji, odbywa się po stronie klienta (JavaScript) oraz po stronie serwera (Atrybuty walidacji w  modelach).

**7. Warstwa Widoku**

**7.1 Szablony widoków**

* Index.cshtml: Strona główna z formularzem wyboru lokalizacji i prezentacją aktualnej pogody.
* City.cshtml: Widok prezentujący szczegółowe informacje o pogodzie.
* SearchByCity.cshtml : Widok reprezentujący interfejs w którym wpisuje się nazwę miejscowości

**7.2 Obsługa zdarzeń interfejsu użytkownika**

* JavaScript/jQuery obsługuje interakcje użytkownika, takie jak wybieranie lokalizacji.

**7.3 Użyte biblioteki frontendowe**

* jQuery (opcjonalnie)
* Bootstrap (do responsywnego projektowania interfejsu użytkownika)

**8. Bezpieczeństwo**

**8.1 Autentykacja i autoryzacja**

W zastosowaniach typu aplikacja pogodowa, autentykacja i autoryzacja mogą być opcjonalne lub nawet niepotrzebne.

**9. Testowanie**

**9.1 Jednostkowe testy**

* Testy jednostkowe dla kluczowych komponentów, takich jak WeatherService.

**9.2 Testy integracyjne**

* Testy integracyjne sprawdzające poprawność komunikacji z serwisem pogodowym.

**9.3 Testy wydajnościowe**

* Możliwe testy wydajnościowe w zależności od liczby użytkowników.