NET Core 6 application that utilizing public weather APIs to collect and store weather data from over ten cities in five different countries

Task 2

Brief Project Documentation

Piotr Stola

Contents

Introduction

This document outlines the procedure for utilizing public weather APIs to collect and store weather data from over ten cities in five different countries. It details how to configure the system to update this data every minute, ensuring timeliness and relevance. The core task involves retrieving essential weather metrics such as country, city, temperature, cloudiness, and wind speed, followed by storing these details in a designated database.

Additionally, the document guides you through creating two distinct graphical representations of the collected data, enabling effective analysis and visualization of weather trends and comparisons.

Used API: https://open-meteo.com/

Changes to requirements:

1. According to API documentation, I had to introduce a change in API call interval to 15 minutes.
2. Instead of calculating the trend between two hours, I decide to use the built-in Trend line functionality in Razdan components.

Technology stack

Projects are based on the following list of technologies and frameworks.

.NET 6.0

EF Core 6.0

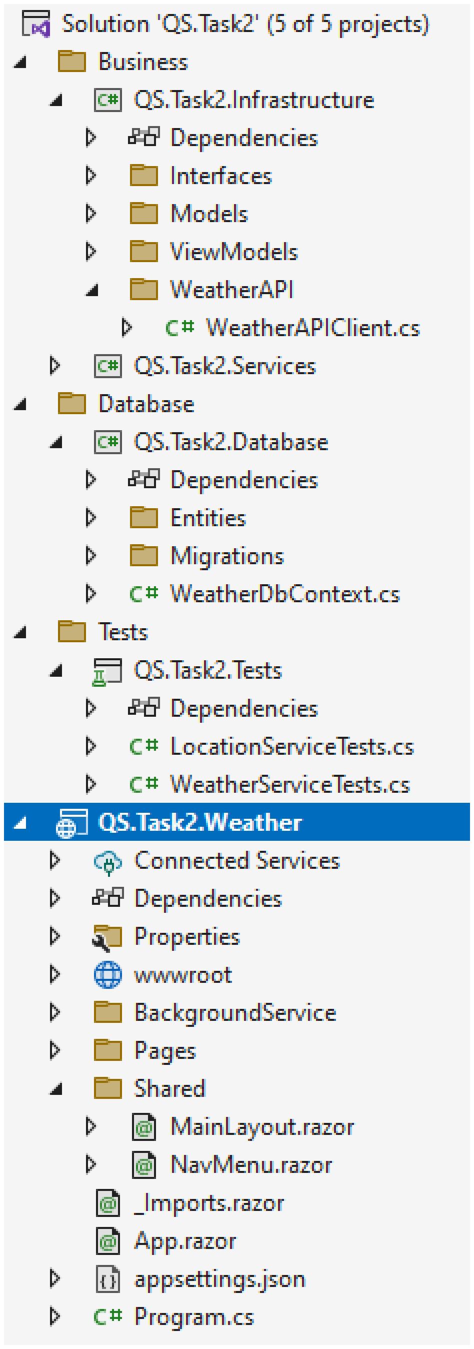
Microsoft Blazor Server 6.0

Radzen Blazor 4.25.9

xUnit Tests – 2.7.0

Moq – 4.20.70

Project structure



```
Solution 'QS.Task2' (5 of 5 projects)
  Business
    QS.Task2.Infrastructure
      Dependencies
      Interfaces
      Models
      ViewModels
      WeatherAPI
        WeatherAPIClient.cs
    QS.Task2.Services
  Database
    QS.Task2.Database
      Dependencies
      Entities
      Migrations
      WeatherDbContext.cs
  Tests
    QS.Task2.Tests
      Dependencies
      LocationServiceTests.cs
      WeatherServiceTests.cs
  QS.Task2.Weather
    Connected Services
    Dependencies
    Properties
    wwwroot
    BackgroundService
    Pages
    Shared
      MainLayout.razor
      NavMenu.razor
    _Imports.razor
    App.razor
    appsettings.json
    Program.cs
```

The solution is architecturally designed to enforce a strict separation between the business layer, database layer, and presentation layer, in line with key software design principles such as Separation of Concerns and the Single Responsibility Principle from the SOLID framework. This clear distinction enhances the system's modularity and maintainability, ensuring that each layer independently manages its specific set of responsibilities.

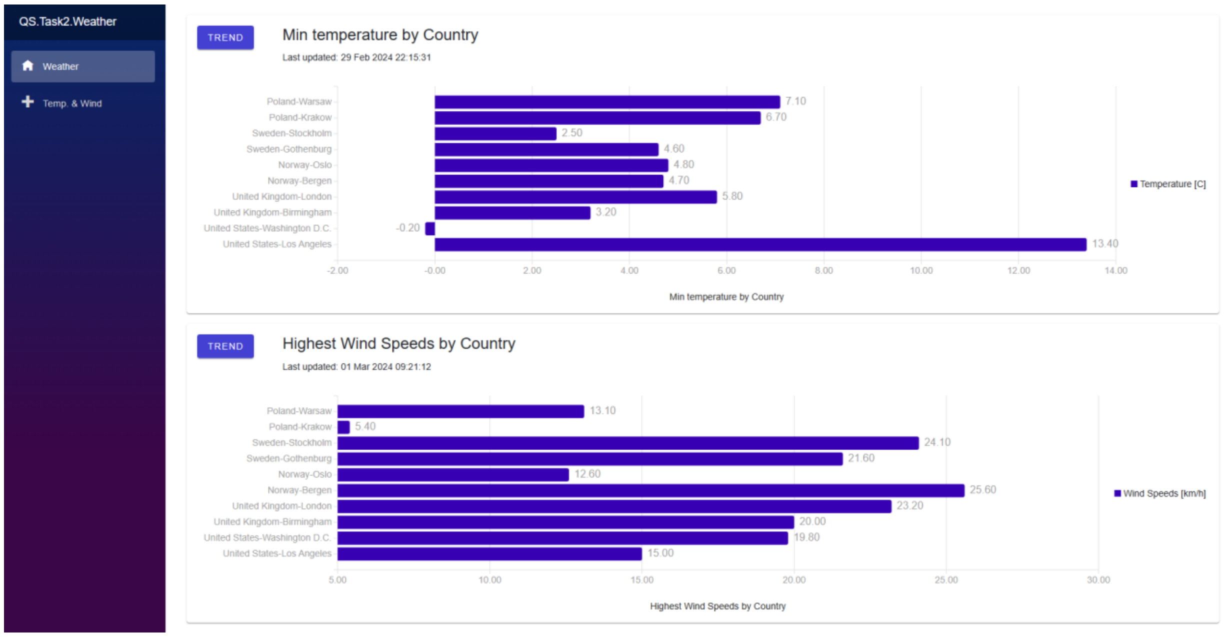
Within the business layer, core classes and interfaces are meticulously outlined to support Dependency Injection, fostering a decoupled and flexible code environment. The database layer is isolated to manage data interactions efficiently, while the presentation layer focuses solely on user interface concerns.

Furthermore, the incorporation of unit testing guarantees robustness and facilitates easier debugging and future updates. By adhering to these structured practices, the solution ensures code reusability, scalability, and straightforward enhancements, setting a solid foundation for a maintainable codebase.

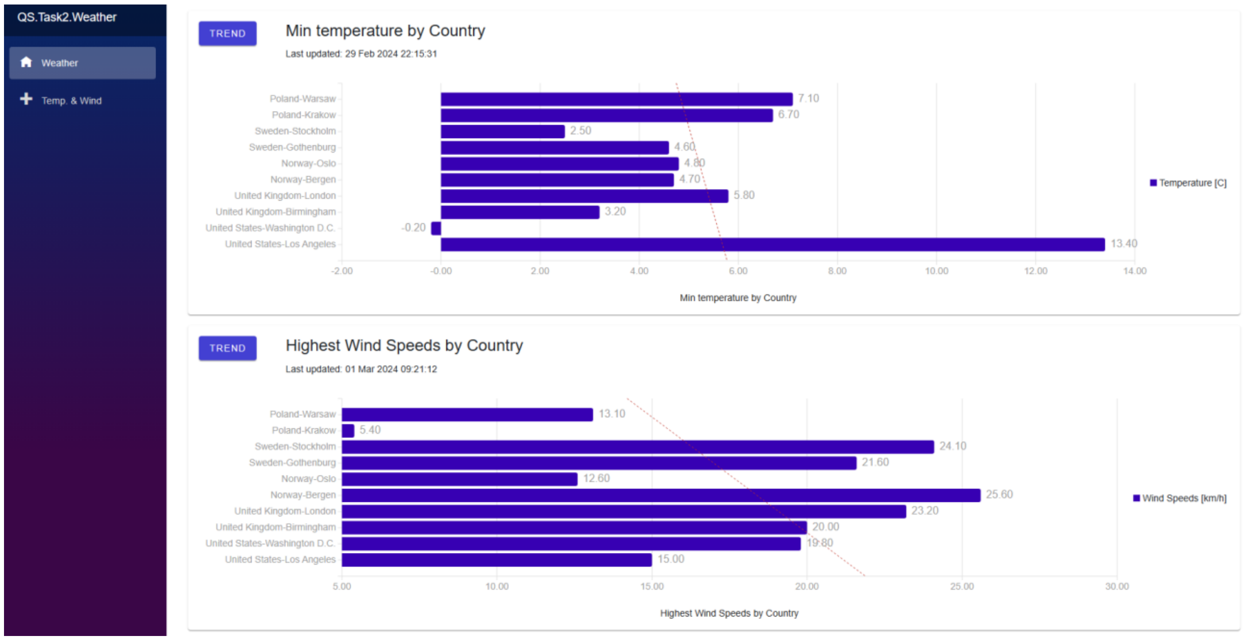
Blazor Server UI Interface and functionalities

Below are two graphical representations:

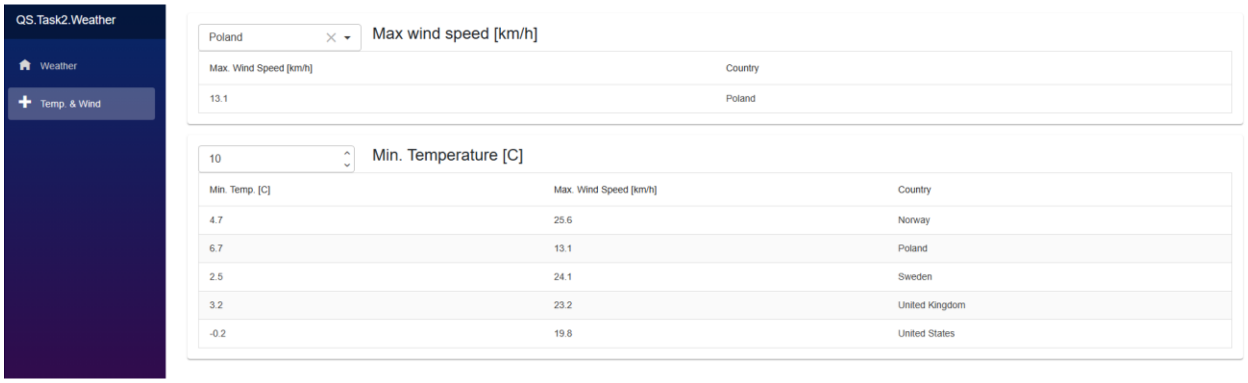
- The first graph displays the minimum temperature, detailing the country, city, temperature, and last update time.
- The second graph illustrates the highest wind speed, showcasing the country, city, wind speed, and last update time.


QS.Task2.Weather
Weather
Temp. & Wind
TREND
Min temperature by Country
Last updated: 29 Feb 2024 22:15:31
Poland-Warsaw 7.10
Poland-Krakow 6.70
Sweden-Stockholm 2.50
Sweden-Gothenburg 4.60
Norway-Oslo 4.80
Norway-Bergen 4.70
United Kingdom-London 5.80
United Kingdom-Birmingham 3.20
United States-Washington D.C. -0.20
United States-Los Angeles 13.40
Temperature [C]
Min temperature by Country
TREND
Highest Wind Speeds by Country
Last updated: 01 Mar 2024 09:21:12
Poland-Warsaw 13.10
Poland-Krakow 5.40
Sweden-Stockholm 24.10
Sweden-Gothenburg 21.60
Norway-Oslo 12.60
Norway-Bergen 25.60
United Kingdom-London 23.20
United Kingdom-Birmingham 20.00
United States-Washington D.C. 19.80
United States-Los Angeles 15.00
Wind Speeds [km/h]
Highest Wind Speeds by Country

Graphs with trend line.


QS.Task2.Weather
Weather
Temp. & Wind
TREND
Min temperature by Country
Last updated: 29 Feb 2024 22:15:31
Temperature [C]
Min temperature by Country
TREND
Highest Wind Speeds by Country
Last updated: 01 Mar 2024 09:21:12
Wind Speeds [km/h]
Highest Wind Speeds by Country

Below is the user interface depiction which demonstrates the functionality of two distinct stored procedures:



Max wind speed [km/h]

Max. Wind Speed [km/h]	Country
13.1	Poland

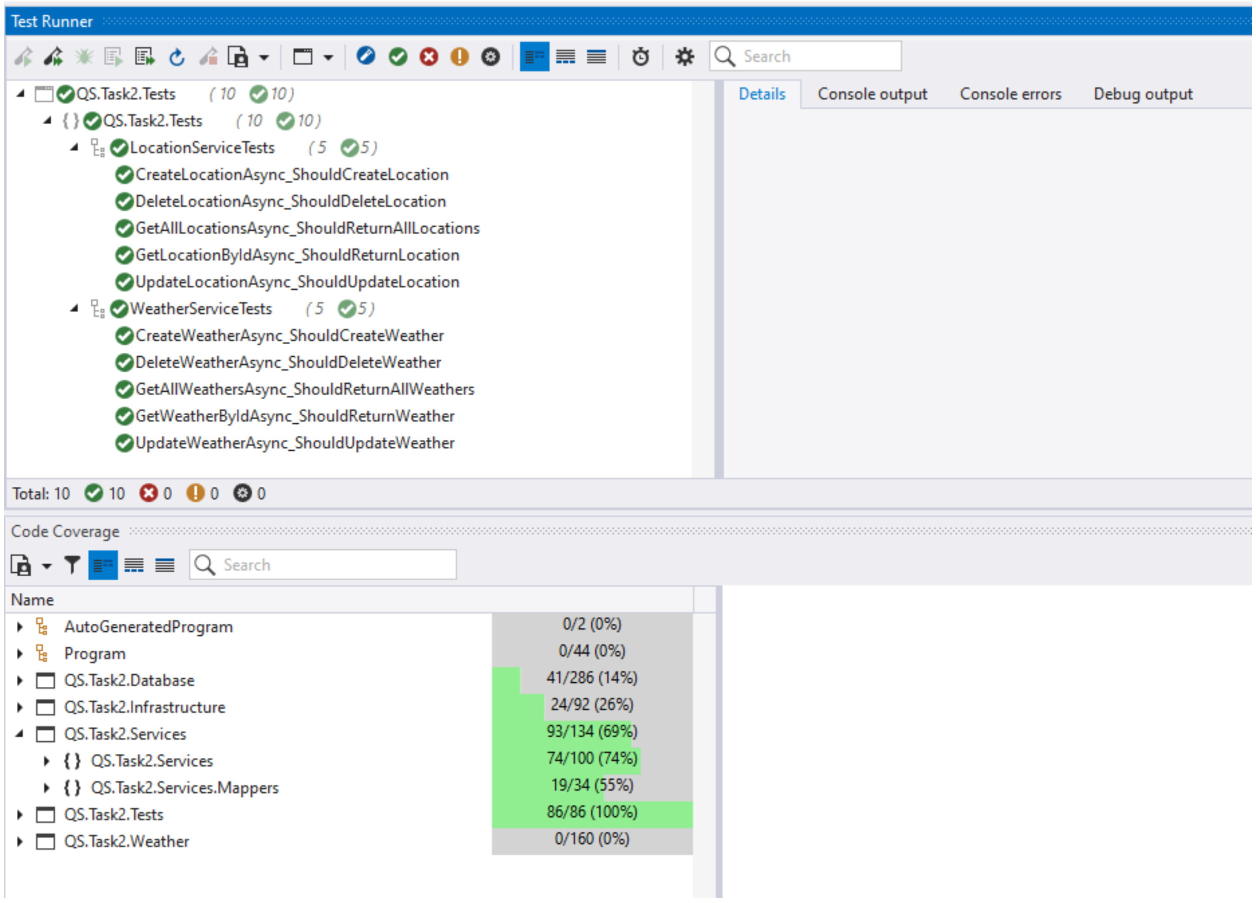
Min. Temperature [C]

Min. Temp. [C]	Max. Wind Speed [km/h]	Country
4.7	25.6	Norway
6.7	13.1	Poland
2.5	24.1	Sweden
3.2	23.2	United Kingdom
-0.2	19.8	United States

The stored procedure definition can be found in the SQL folder of the GitHub repository.

Unit tests and Test Coverage

Please find below unit test for Azure Function code.



```
QS.Task2.Tests        ( 10  10 )
  { } QS.Task2.Tests        ( 10  10 )
    LocationServiceTests        ( 5  5 )
      CreateLocationAsync_ShouldCreateLocation
      DeleteLocationAsync_ShouldDeleteLocation
      GetAllLocationsAsync_ShouldReturnAllLocations
      GetLocationByIdAsync_ShouldReturnLocation
      UpdateLocationAsync_ShouldUpdateLocation
    WeatherServiceTests        ( 5  5 )
      CreateWeatherAsync_ShouldCreateWeather
      DeleteWeatherAsync_ShouldDeleteWeather
      GetAllWeathersAsync_ShouldReturnAllWeathers
      GetWeatherByIdAsync_ShouldReturnWeather
      UpdateWeatherAsync_ShouldUpdateWeather

Total: 10  10  0  0  0
```

Name	
AutoGeneratedProgram	0/2 (0%)
Program	0/44 (0%)
QS.Task2.Database	41/286 (14%)
QS.Task2.Infrastructure	24/92 (26%)
QS.Task2.Services	93/134 (69%)
{ } QS.Task2.Services	74/100 (74%)
{ } QS.Task2.Services.Mappers	19/34 (55%)
QS.Task2.Tests	86/86 (100%)
QS.Task2.Weather	0/160 (0%)

Time report

Time is used to create, design, and develop specific parts of the system.

Task Name	Time [h]
Services	1,5
Unit test	1
API	1
Database and EF Core	1,5
Documentation	1
Total :	**7h**

Future development and improvements

"There's always room for improvement, you know-it's the biggest room in the house." —

Louise Heath Leber

I would like to introduce a few concepts and potential improvements for the project's future evolution.

Project name:

1. QS.Task2.Weather
    a. Implementation of Authentication and Authorization mechanism
    b. we should consider improving AppSettings key management.
    c. Introduce a validation mechanism for user input request parameters.
    d. Introduce Blazor components segregation and separation.
2. QS.Task2.Services
    a. Introduce custom error class
    b. Improve mapping mechanism – with the Entities and ViewModels number increasing it's a good idea to switch to AutoMapper
3. QS.Task2.Tests
    a. Unit test test for API calls and Seeding