Dokumentacja techniczna systemu pobierania danych pogodowych

Oracle + C# Worker Service

CerbIT / System pogodowy

19 listopada 2025

1 Wprowadzenie

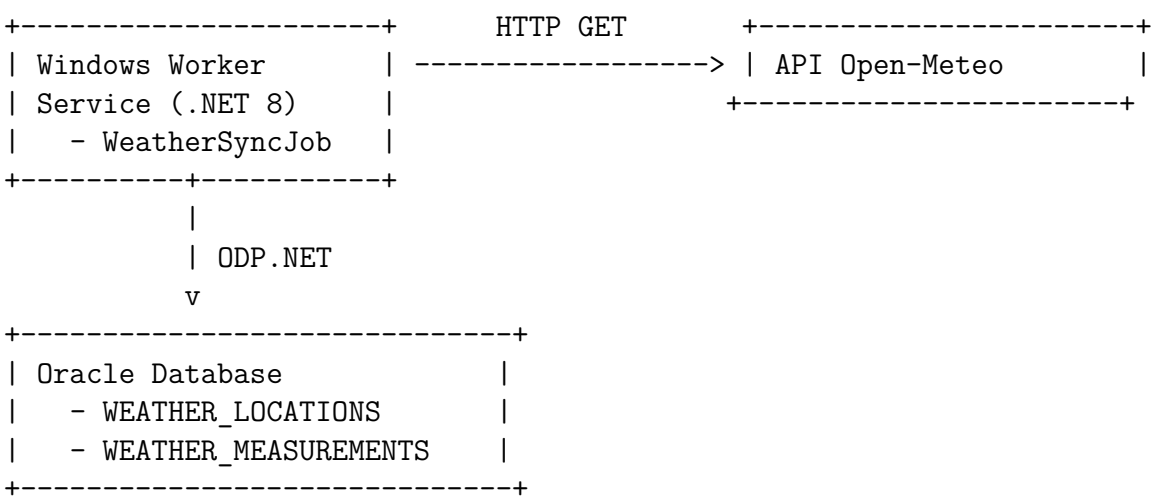
Celem projektu jest automatyczne pobieranie danych pogodowych z darmowego API (Open-Meteo) i zapisywanie ich do bazy danych Oracle. Proces działa jako **Windows Service** napisany w technologii .NET 8 (Worker Service), uruchamiany cyklicznie co 10 minut.

Rozwiazanie składa sie z nastepujacych elementów:

- baza danych Oracle – tabele `WEATHER_LOCATIONS` oraz `WEATHER_MEASUREMENTS`,
- serwis Windows (Worker Service) uruchamiany w tle,
- komponent `WeatherSyncJob` odpowiedzialny za pobieranie i zapis danych,
- konfiguracja w pliku `appsettings.json`.

2 Architektura systemu

2.1 Schemat logiczny



2.2 Opis komponentów

- **Worker (BackgroundService)** Uruchamiany automatycznie przez system Windows. W petli wykonuje zadania co 10 minut.
- **WeatherSyncJob** Główna logika biznesowa:
  - test połaczeń,
  - pobranie danych pogodowych,
  - serializacja/parsowanie JSON,
  - zapis do Oracle.
- **Oracle** Baza przechowujaca lokalizacje oraz pomiary pogody.

3 Struktura bazy danych Oracle

3.1 Tabela LOCATION

```
CREATE TABLE WEATHER_LOCATIONS (
    ID_LOCATION      NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY
        ,
    COUNTRY_CODE     VARCHAR2(2) NOT NULL,
    CITY_NAME        VARCHAR2(100) NOT NULL,
    LATITUDE         NUMBER(9,6) NOT NULL,
    LONGITUDE        NUMBER(9,6) NOT NULL,
    ACTIVE_FLAG      CHAR(1) DEFAULT 'Y' NOT NULL
);
```

3.2 Tabela MEASUREMENTS

```
CREATE TABLE WEATHER_MEASUREMENTS (
    ID_MEASUREMENT   NUMBER GENERATED AS IDENTITY PRIMARY KEY,
    ID_LOCATION      NUMBER NOT NULL,
    MEASURED_AT      DATE NOT NULL,
    TEMP_C           NUMBER(5,2),
    IS_RAIN          CHAR(1),
    HUMIDITY         NUMBER(5,2),
    WIND_SPEED_MS    NUMBER(6,2),
    RAW_JSON         CLOB,
    INSERTED_AT      DATE DEFAULT SYSDATE NOT NULL,
    CONSTRAINT FK_LOC FOREIGN KEY (ID_LOCATION)
       REFERENCES WEATHER_LOCATIONS(ID_LOCATION)
);
```

4 Konfiguracja systemu

4.1 Plik appsettings.json

```
{
  "ConnectionStrings": {
    "Oracle": "User␣Id=USER;Password=PASS;Data␣Source=HOST:1521/SERVICE;
       "
  },
  "WeatherApi": {
    "BaseUrl": "https://api.open-meteo.com/v1/forecast",
    "Params": "hourly=temperature_2m,relativehumidity_2m,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣precipitation,wind_speed_10m&forecast_days=1&timezone=
    auto"
  }
}
```

5 Implementacja serwisu Windows (.NET 8)

5.1 Plik Program.cs

```
var host = Host.CreateDefaultBuilder(args)
    .UseWindowsService()
    .ConfigureServices((context, services) =>
    {
        services.AddSingleton<WeatherSyncJob>();
        services.AddHostedService<Worker>();
    })
    .Build();

await host.RunAsync();
```

5.2 Worker.cs – serwis cykliczny

```
public class Worker : BackgroundService
{
    private readonly WeatherSyncJob _job;

    public Worker(WeatherSyncJob job) => _job = job;

    protected override async Task ExecuteAsync(CancellationToken token)
    {
        while (!token.IsCancellationRequested)
        {
            await _job.RunOnce(token);
            await Task.Delay(TimeSpan.FromMinutes(10), token);
        }
    }
}
```

6 Logika synchronizacji – WeatherSyncJob

6.1 Test połaczeń

```
private async Task TestConnectionsAsync(CancellationToken token)
{
    using var conn = new OracleConnection(_conn);
    await conn.OpenAsync(token);

    var resp = await new HttpClient().GetAsync(_url, token);
}
```

6.2 Parsowanie JSON

```
var doc = JsonDocument.Parse(json);
var hourly = doc.RootElement.GetProperty("hourly");
var last = hourly.GetProperty("time").GetArrayLength() - 1;

var measurement = new WeatherMeasurement {
    TempC = hourly.GetProperty("temperature_2m")[last].GetDouble(),
    ...
};
```

6.3 Zapis pomiaru do Oracle

```
INSERT INTO WEATHER_MEASUREMENTS
(ID_LOCATION, MEASURED_AT, TEMP_C, IS_RAIN,
 HUMIDITY, WIND_SPEED_MS, RAW_JSON)
VALUES (:loc, :time, :temp, :rain, :hum, :wind, :json)
```

7 Instalacja jako usługa Windows

7.1 Publikacja projektu

```
dotnet publish -c Release -r win-x64 --self-contained false -o publish
```

7.2 Rejestracja usługi

```
sc create WeatherService binPath= "C:\Weather\WeatherService.exe"
sc start WeatherService
```

8 Podsumowanie

Przygotowany system umożliwia:

- automatyczne i cykliczne pobieranie danych pogodowych,
- stabilne działanie jako usługa Windows,
- pełna integracje z Oracle (ODP.NET),

- możliwość łatwego rozszerzania o kolejne lokalizacje i parametry.

System jest gotowy do użycia produkcyjnego po dodaniu logowania (np. Serilog) i mechanizmów retry/fallback przy błedach API.