

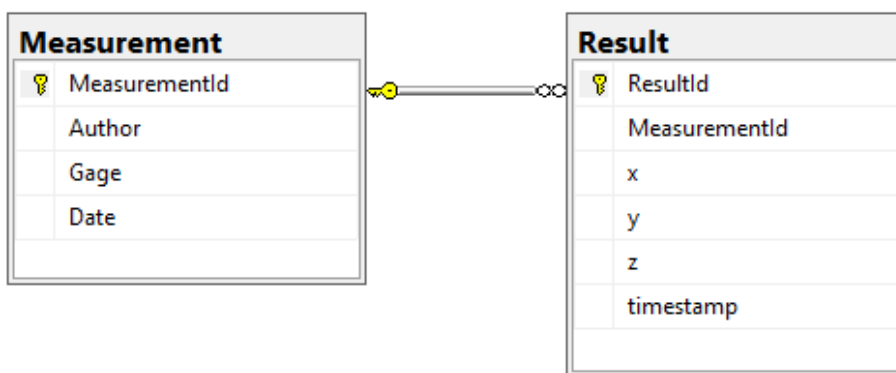
## 6. Usługi internetowe RESTful WebAPI

### Cele ćwiczenia

- Zapoznanie z usługami RESTful WebAPI.
- Zapoznanie z opisami usług sieciowych RESTful i ze sposobami korzystania z tych usług za pomocą dostępnych programów.
- Opanowanie umiejętności korzystania z usług WebAPI w aplikacjach internetowych i desktopowych.

### Usługa RESTful Web API

Na stronie <http://argo.umg.edu.pl/www/RestWebApi/Help> znajduje się opis usługi Web API. Usługa jest sieciowym interfejsem do bazy danych przechowujących wyniki pomiarów i realizuje operacje CRUD na tabelach bazy danych przedstawionej na diagramie z rysunku 6.1.



Rys. 6.1. Diagram bazy danych

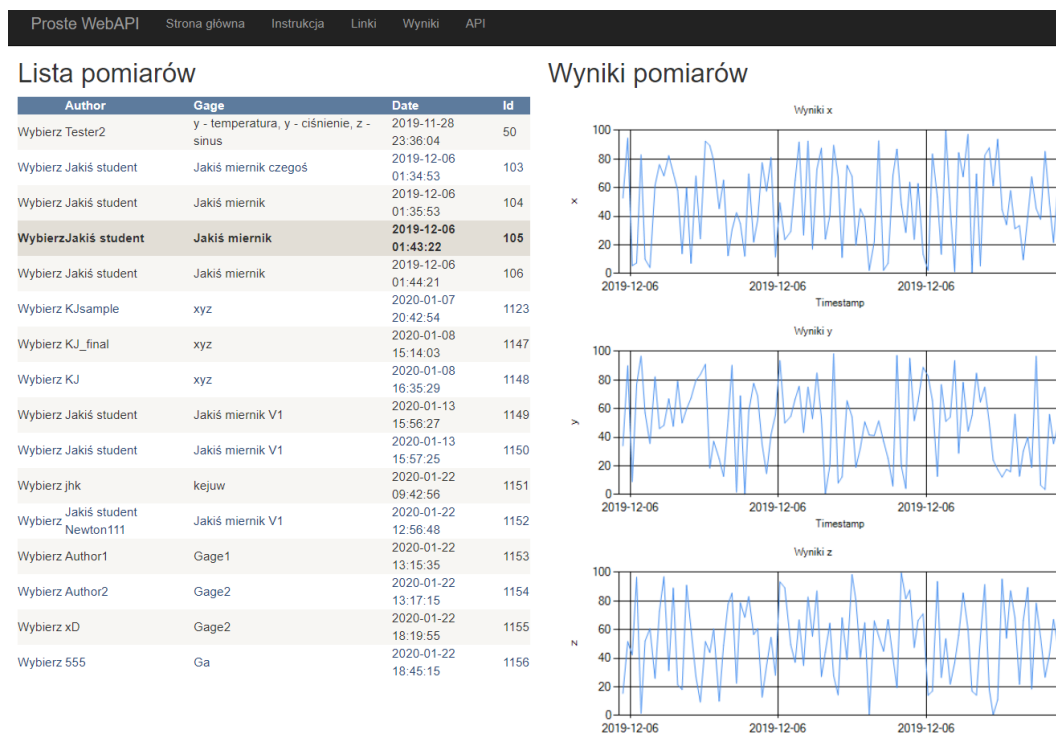
Opis usługi został wygenerowany automatycznie przez środowisko Visual Studio. Elementy Description opisu są pobierane z komentarzy umieszczonych przez programistę w kodzie usługi.

W założeniu, baza danych jest przeznaczona do rejestracji serii pomiarów. Tabela **Measurement** zawiera opis przeprowadzonych pomiarów, kto je wykonał (Author), jakim przyrządem (Gage) i kiedy (Date). Po wprowadzeniu danych pomiaru (Autor, Gage i opcjonalnie Date) system nadaje pomiarowi identyfikator MeasurementId. Tabela **Result** zawiera 3 serie wyników (x, y, z) wykonane w ramach pomiaru o identyfikatorze MeasurementId.

Operacje CRUD na tabeli **Measurement** wykonuje API **Measurements**, a na tabeli **Result**

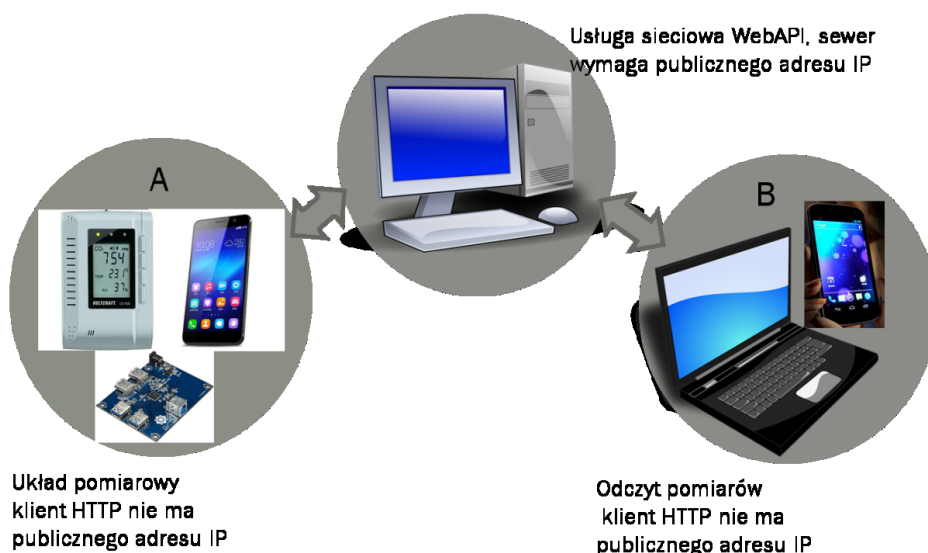


**API Results.** Wprowadzone dane można zobaczyć na stronie Wyniki. Dane pomiarów z tabeli **Measurement** są prezentowane na stronie <http://argo.umg.edu.pl/www/RestWyniki/Pages/WebForm1> w tabeli, a serie pomiarów z tabeli **Result** na wykresach, rysunek 6.2.



Rys. 6.2. Strona prezentująca dane z bazy danych usługi

Na rysunku 6.3 przedstawiono przykład wykorzystania usługi Web API konfiguracji w jakiej jest ona wykorzystywana w laboratorium.



Rys. 6.3. Przykładowa konfiguracja wykorzystania usługi Web API

## Ćwiczenie 6.1. Testowanie usługi w stylu architektury REST

1. Zapoznać się z opisem usługi.
2. Wykonać testy usługi, do testowania usługi można użyć
  - przeglądarki internetowej, wysyła żądanie tylko metodą GET,
  - programu Postman, wysyła żądanie wszystkimi metodami,
  - programu Node-RED, wysyła żądanie wszystkimi metodami,
  - programu [SendHttpRequest](#) lub
  - własnego narzędzia.
3. Pobrać istniejące dane metodą GET, pobrane dane mogą posłużyć jako wzorzec do wykonania dalszych punktów.
  - pobrać dane pomiarów (API Measurements),
  - pobrać dane wyników pomiarów (API Results),
  - porównać dane z danymi przedstawionych na stronie Wyniki,
  - format odpowiedzi określa nagłówek Accept, np. = application/xml albo = application/json.
4. Wprowadzić dane nowego pomiaru – API Measurements metoda POST
  - Wprowadzając dane można pominąć MeasurementId i Date, identyfikator jest nadawany automatycznie a data ma wartość domyślną – bieżąca data i czas,
  - w danych umieścić swoje dane rozpoznawcze (Author) i opis przyrządu lub pomiarów (Gage),
  - zapisać identyfikator MeasurementId nadany przez serwer,

Projekt „SezAM wiedzy, kompetencji i umiejętności” jest współfinansowany przez Unię Europejską ze środków Europejskiego Funduszu Społecznego w ramach Programu Operacyjnego Wiedza Edukacja Rozwój

- zobaczyć swoje dane w tabeli na stronie Wyniki.

W środowisku Node-RED do wykonania zadania można wykorzystać bloczki przedstawione w tabeli 6.1.

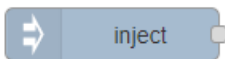
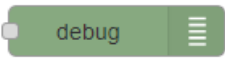

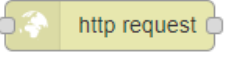
|   |   |
|---|---|
|  | wysyła wiadomość po kliknięciu lub okresowo, w zależności od konfiguracji,                                    |
|  | wyświetla wiadomości w celach kontrolnych,  |
|  | zawiera kod JavaScript, przy jego pomocy można zakodować wysyłaną wiadomość obiekt <code>msg.payload</code> , |
|  | wysyła żądanie, <code>msg.payload</code> na wejściu zawiera przesyłaną treść żądania, na wyjściu odpowiedź.   |


Tabela 6.1. Bloczki Node-RED wykorzystywane w ćwiczeniu

Bloczki można konfigurować za pomocą okien dialogowych, które pojawiają się po dwukrotnym kliknięciu na bloczek. Przykładowy graf realizujący zadanie przedstawiono na rysunku 6.4.



Rys. 6.4. Przykładowy graf klienta usługi dodający nowy pomiar i wyniki pomiarów

Żeby graf działał bloczki należy właściwie skonfigurować. Identyfikator pomiaru można odczytać za pomocą debugera.

5. Wprowadzić przykładowe serie wyników pomiarów – API Results metoda POST wykorzystać identyfikator `MeasurementId` uzyskany w punkcie 4, zobaczyć wprowadzone serie danych na wykresach na stronie Wyniki, można wprowadzać dane losowe, np. w środowisku Node-RED można posłużyć się bloczkiem  z następującym kodem

```
msg.payload = {
  "MeasurementId": 1,
  "x": Math.random(),
  "y": Math.random(),
  "z": Math.random() }
return msg;
```

6. Przy pomocy API dokonać zmian we wprowadzonych danych.
7. Opisać zastosowaną metodę wprowadzania danych.
8. Napisać aplikację konsolową lub Windows Forms wprowadzającą przykładowe serie danych za pomocą API, można wzorować się na programie [SendHttpRequest](#). Zastosowane w programie metody wymagają platformy .NET 4.5 lub wyższej.

Projekt „SezAM wiedzy, kompetencji i umiejętności” jest współfinansowany przez Unię Europejską ze środków Europejskiego Funduszu Społecznego w ramach Programu Operacyjnego Wiedza Edukacja Rozwój

Przykładowy kod dodający nowy pomiar (bez wyników) umieszczono na listingu 6.1. [program.pdf](#)

Listing 6.1. Przykładowy kod dodający nowy pomiar do tabeli **Measurement** i odczytujący identyfikator pomiaru

```
public class Measurements
{
    public int MeasurementId { get; set; }
    public string Author { get; set; }
    public string Gage { get; set; }
}
static async Task Main(string[] args)
{
    Measurements measurements = new Measurements
    {
        Author = "Jakiś student",
        Gage = "Jakiś miernik V1"
    };

    string jsonText = JsonSerializer.Serialize(measurements);
    HttpContent content = new StringContent(jsonText, Encoding.UTF8, "application/json");

    HttpRequestMessage requestMessage = new HttpRequestMessage
    {
        RequestUri = new Uri("http://argo.am.gdynia.pl/www/RestWebApi/Api/Measurements"),
        Method = HttpMethod.Post,
        Content = content
    };
    requestMessage.Headers.Add("Accept", "application/json");

    HttpClient clientHttp = new HttpClient();
    var response = await clientHttp.SendAsync(requestMessage);
    var kontekst = await response.Content.ReadAsStringAsync();

    var measurementsOutput = JsonSerializer.Deserialize<Measurements>(kontekst.ToString());
    Console.WriteLine("Identyfikator pomiaru = " +
measurementsOutput.MeasurementId.ToString());
}
```

W przykładowym programie serializacji obiektu `measurements` do łańcucha tekstu w formacie JSON dokonano wykorzystując pakiet `System.Text.Json`, który wymaga platformy .NET 4.6.1 lub wyższej albo .NET Core. Zamiast `System.Text.Json` można stosować [Newtonsoft.Json](#) jak w przykładzie [Program2.pdf](#).

Program należy uruchomić, następnie rozszerzyć o dodawanie serii przykładowych wyników pomiarów, mogą to być dane losowe lub wykresy funkcji. Wysoko ocenione będzie przerobienie programu tak, aby komunikował się z usługą przesyłając dane w formacie XML. Nagłówki powinny mieć wtedy wartość `= application/xml`.

Informacje jak zainstalować i przygotować Visual Studio Code znajdują się na stronie <https://docs.microsoft.com/pl-pl/dotnet/core/tutorials/with-visual-studio-code>.