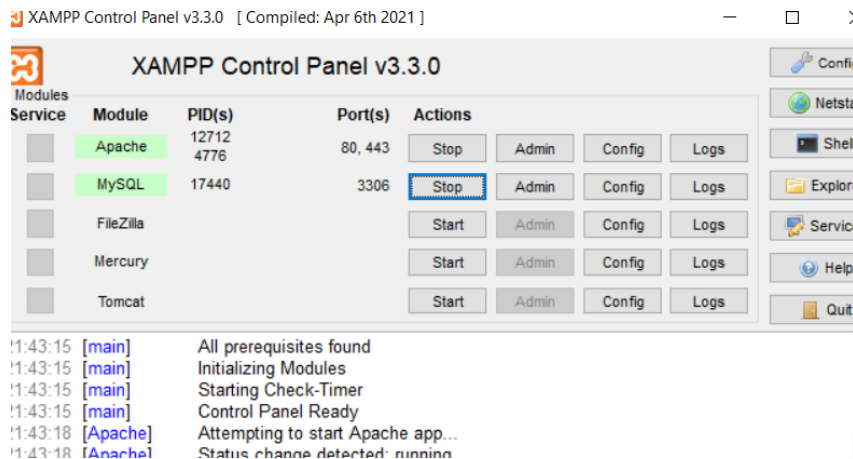


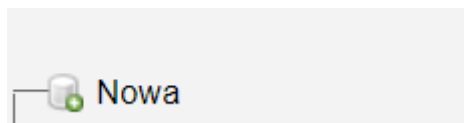
# Dokumentacja

## 1. Baza danych

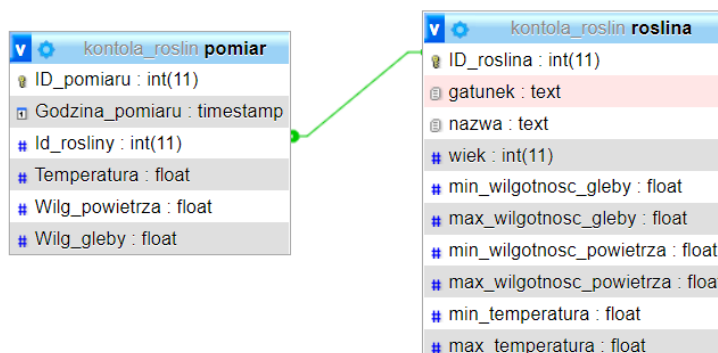
Aby stworzyć bazę danych na serwerze trzeba pobrać serwer lokalny. W moim przypadku będzie to pakiet Xampp, który umożliwi nam stworzenie lokalnego serwera z bazą danych. W tym celu należy po zainstalowaniu włączyć Apache oraz MySQL



Następnie w wyszukiwarce wpisuję localhost/phpmyadmin i tworzę nową bazę danych.



Następnie dodaję tabelki dla rośliny i dla pomiaru. Łączę je relacją. ID rośliny jest kluczem obcym w tabeli pomiary. Oznacza to że możemy mieć wiele roślin i dla każdej z nich będą przypisane pomiary.



Do tabelki dodałem kilka rekordów aby przeprowadzić testy dla nich.

ID_roslina	gatunek	nazwa	wiek	min_wilgotnosc_gleby	max_wilgotnosc_gleby	min_wilgotnosc_powietrza	max_wilgotnosc_powietrza	min_temperatura	max_temperatura
1	azalia	azalia1	1	4.04	6	4.6	5.6	7.4	9
7	tulipan	tulipan1	0	4	8	20	40	1	20
8	bazylia	bazylia1	0	4	8	20	40	5	24
9	roza	roza1	0	4	8	20	40	10	34

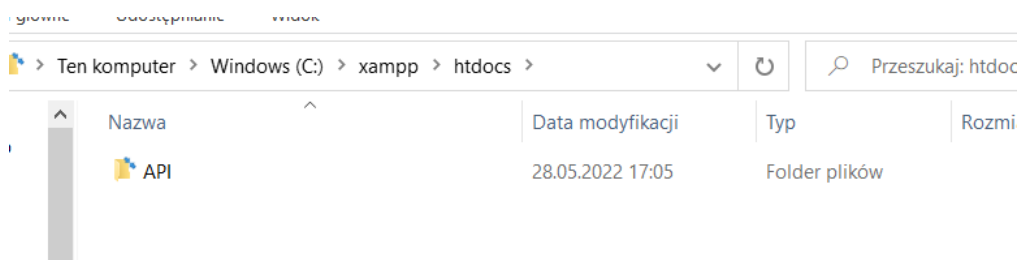
ID_pomiaru	Godzina_pomiaru	Id_rosliny	Temperatura	Wilg_powietrza	Wilg_gleby
1	2022-04-10 14:49:49	1	20	30	30
2	2022-05-31 09:27:39	7	20	8	9
3	2022-05-31 09:27:44	7	20	8	9

## 2.API

Zależy nam na stworzeniu API, które będzie przetwarzać dane w formacie JSON.

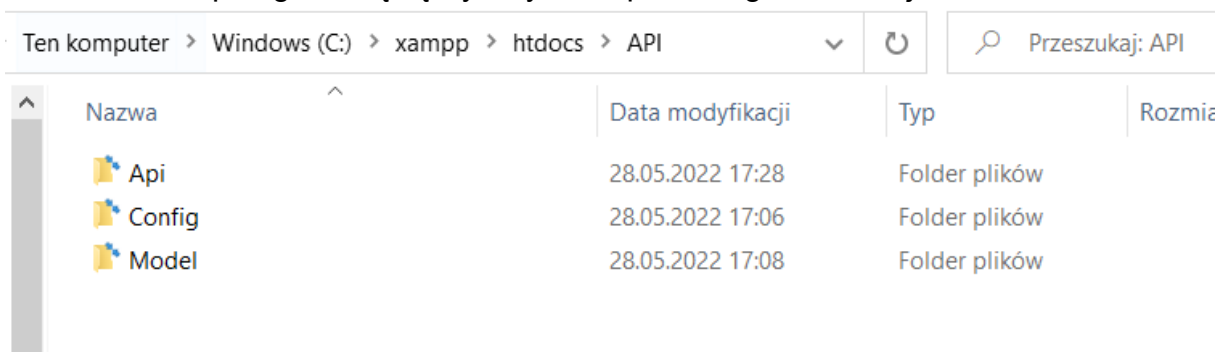
JSON jest sposobem prezentacji danych zaczerpniętym z języka JavaScript ale obsługiwany przez większość języków. Składnia tego zapisu polega na zamknięciu wyrażenia w klamrach {} a każdy atrybut pisany jest w cudzysłowie następnie „:” i wartość atrybutu pisana również w cudzysłowie. Więcej informacji o tym formacie: <https://www.json.org/json-en.html>.

Aby API działało na serwerze należy odnaleźć ścieżkę C:\xampp\htdocs i u stworzyć w niej folder o nazwie API.



Następnie w folderze tworzymy 3 kolejne foldery:

- Model – w którym będą zdefiniowane klasy dla obiektów, które będziemy wywoływać, zapytania SQL oraz szkielety funkcji,
- Config – gdzie będzie przechowywany plik z połączeniem do bazy danych
  - Api – gdzie będą wywoływane poszczególne funkcje



Kod został stworzony w języku PHP, który umożliwia łatwy kontakt z bazą danych. W folderze Config stworzyłem plik Database.php, który zawiera informację o połączeniu z bazą danych. Należy w nim podać nazwę bazy, użytkownika, hasło oraz adres serwera a następnie nawiązać połączenie z bazą. W przypadku hostu lokalnego będą to domyślne ustawienia. Zmienna conn reprezentuje połączenie z bazą i wszelkie zmiany które będą w niej zachodzić będą wywoływane za pomocą conn.

```
<?php
class Database {
    // DB Params
    private $host = 'localhost';
    private $db_name = 'kontola_roslin';
    private $username = 'root';
    private $password = '';
    private $conn;

    // DB Connect
    public function connect() {
        $this->conn = null;

        try {
            $this->conn = new PDO('mysql:host=' . $this->host . ';dbname=' . $this->db_name, $this->username, $this->password);
            $this->conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        } catch(PDOException $e) {
            echo 'Connection Error: ' . $e->getMessage();
        }

        return $this->conn;
    }
}
```

Następnie w folderze Model tworzę klasy zawierające metody i komendy SQL dla Pomiaru i Rośliny.

```
<?php
class Pomiar {
    // DB stuff
    private $conn;
    private $table = 'pomiar';

    // Post Properties
    public $ID_pomiaru;
    public $id_rosliny;
    public $nazwa_rosliny;
    public $temperatura;
    public $wilg_powietrza;
    public $wilg_gleby;
    public $godzina_pomiaru;

    // Constructor with DB
    public function __construct($db) {
        $this->conn = $db;
    }

    // Get Posts
    public function read() {
        // Create query
        $query = 'SELECT r.nazwa as nazwa_rosliny, p.ID_pomiaru, p.Godzina_pomiaru, p.Id_rosliny ,p.Temperatura,p.Wilg_powietrza, p.Wilg
FROM ' . $this->table . ' p
LEFT JOIN
roslina r ON p.Id_rosliny=r.ID_roslina
ORDER BY
p.Godzina_pomiaru DESC';
```

```
// Get Posts
public function read() {
    // Create query
    $query = 'SELECT r.nazwa as nazwa_rosliny, p.ID_pomiaru, p.Godzina_pomiaru, p.Id_rosliny ,p.Temperatura,p.Wilg_powietrza, p.Wilg
FROM ' . $this->table . ' p
LEFT JOIN
roslina r ON p.Id_rosliny=r.ID_roslina
ORDER BY
p.Godzina_pomiaru DESC';

    // Prepare statement
    $stmt = $this->conn->prepare($query);

    // Execute query
    $stmt->execute();

    return $stmt;
}
```

```

<?php
class Roslina {
    // DB Stuff
    private $conn;
    private $table = 'roslina';

    // Properties
    public $ID_roslina;
    public $gatunek;
    public $nazwa;
    public $min_temperatura;
    public $max_temperatura;
    public $min_wilgotnosc_gleby;
    public $max_wilgotnosc_gleby;
    public $min_wilgotnosc_powietrza;
    public $max_wilgotnosc_powietrza;

    // Constructor with DB
    public function __construct($db) {
        $this->conn = $db;
    }
}

```

```

// Get categories
public function read() {
    // Create query
    $query = 'SELECT
        ID_roslina,
        gatunek,
        nazwa,
        min_temperatura,
        max_temperatura,
        min_wilgotnosc_gleby,
        max_wilgotnosc_gleby,
        min_wilgotnosc_powietrza,
        max_wilgotnosc_powietrza
    FROM
        ' . $this->table . '
    ORDER BY
        ID_roslina DESC';



    // Prepare statement
    $stmt = $this->conn->prepare($query);

    // Execute query
    $stmt->execute();

    return $stmt;
}

```

W katalogu Api tworzę osobno foldery dla Pomiaru oraz Rośliny w których znajdować się będą funkcje odpowiedzialne za przetwarzanie danych z bazy.

« Windows (C:) > xampp > htdocs > API > Api >					Przeszukaj: Api
Nazwa		Data modyfikacji	Typ	Rozmiar	
 Pomiar		28.05.2022 22:29	Folder plików		
 Roslina		28.05.2022 20:43	Folder plików		

Do przetwarzania danych za pomocą Api będziemy używać specjalnych komend http:

- GET – pobranie zasobu z bazy danych
- POST – utworzenie zasobu w bazie danych
- PUT – zaktualizowanie zasobu w bazie danych
- DELETE – usunięcie zasobu z bazy

Aby przetestować działanie Api użyjemy programu Postman, który automatycznie pokaże nam działanie programu.

## Roślina odczyt danych

Endpoint: GET 'http://localhost/API/Api/Roslina/read.php'

Do odczytu danych z tabelki rośliny używamy komendy GET oraz znajdującego się w katalogu Roslina pliku read.php. W odpowiedzi powinniśmy uzyskać listę roślin razem z informacjami o niej zapisane w formacie JSON.

```
<?php
// Headers
header('Access-Control-Allow-Origin: *');
header('Content-Type: application/json');

include_once '../Config/Database.php';
include_once '../Model/Roslina.php';

// Instantiate DB & connect
$databse = new Database();
$db = $databse->connect();

// Instantiate roslina object
$roslina = new Roslina($db);

// roslina read query
$result = $roslina->read();

// Get row count
$num = $result->rowCount();

// Check if any categories
if($num > 0) {
    // Cat array
    $cat_arr = array();
    $cat_arr['data'] = array();

    while($row = $result->fetch(PDO::FETCH_ASSOC)) {
        extract($row);
    }
}
```

```

// Check if any categories
if($num > 0) {
    // Cat array
    $cat_arr = array();
    $cat_arr['data'] = array();

    while($row = $result->fetch(PDO::FETCH_ASSOC)) {
        extract($row);

        $cat_item = array(
            'ID_roslina' => $ID_roslina,
            'nazwa' => $nazwa,
            'gatunek' => $gatunek,
            'min_temperatura' => $min_temperatura,
            'max_temperatura' => $max_temperatura,
            'min_wilgotnosc_gleby' => $min_wilgotnosc_gleby,
            'max_wilgotnosc_gleby' => $max_wilgotnosc_gleby,
            'min_wilgotnosc_powietrza' => $min_wilgotnosc_powietrza,
            'max_wilgotnosc_powietrza' => $max_wilgotnosc_powietrza
        );

        // Push to "data"
        array_push($cat_arr['data'], $cat_item);
    }

    // Turn to JSON & output
    echo json_encode($cat_arr);
} else {
    // No Categories
    echo json_encode(
        array('message' => 'No Found')
    );
}
?>

```

GET
http://localhost/API/Api/Roslina/read.php
Send

```

{
  "data": [
    {
      "ID_roslina": 9,
      "nazwa": "roza1",
      "gatunek": "roza",
      "min_temperatura": 10,
      "max_temperatura": 34,
      "min_wilgotnosc_gleby": 4,
      "max_wilgotnosc_gleby": 8,
      "min_wilgotnosc_powietrza": 20,
      "max_wilgotnosc_powietrza": 40
    },
    {
      "ID_roslina": 8,
      "nazwa": "bazyliia1",
      "gatunek": "bazyliia",
      "min_temperatura": 5,
      "max_temperatura": 24,
      "min_wilgotnosc_gleby": 4,
      "max_wilgotnosc_gleby": 8,
      "min_wilgotnosc_powietrza": 20,
      "max_wilgotnosc_powietrza": 40
    }
  ]
}

```

Jak widzimy otrzymujemy dane w formacie JSON zatem Api działa poprawnie.

Możemy również odczytać pojedynczą roślinę wyszukując jej ID. Posłuży do tego plik `read_single.php`

GET
http://localhost/API/Api/Roslina/read\_single.php?ID\_roslina=1
Send

```

{
  "ID_roslina": 1,
  "nazwa": "azalia1",
  "gatunek": "azalia",
  "min_temperatura": 7.4,
  "max_temperatura": 9,
  "min_wilgotnosc_gleby": 4.04,
  "max_wilgotnosc_gleby": 4.04,
  "min_wilgotnosc_powietrza": 4.6,
  "max_wilgotnosc_powietrza": 5.6
}

```

## Roślina tworzenie danych

Endpoint: GET 'http://localhost/API/Api/Roslina/read.php'

Używamy zapytania POST i przesyłamy JSONa do Api za pomocą create.php

POST http://localhost/API/Api/Roslina/create.php

Send

Tworzymy kod JSON, w którym zawarte będą informacje o naszej roślinie którą chcemy dodać

```

1  {
2      "gatunek": "storczyk",
3
4      "nazwa": "moj_storczyk",
5      "min_temperatura": "15",
6      "max_temperatura": "64",
7      "min_wilgotnosc_gleby": "4",
8      "max_wilgotnosc_gleby": "7",
9      "min_wilgotnosc_powietrza": "20",
10     "max_wilgotnosc_powietrza": "40"
11 }

```

Sprawdzamy w bazie danych i widzimy, że pojawił się tam kolejny rekord zawierający wpisane przez nas dane.

ID_roslina	gatunek	nazwa	wiek	min_wilgotnosc_gleby	max_wilgotnosc_gleby	min_wilgotnosc_powietrza	max_wilgotnosc_powietrza	min_temperatura	max_temperatura
1	azalia	azalia1	1	4.04	6	4.6	5.6	7.4	9
7	tulipan	tulipan1	0	4	8	20	40	1	20
8	bazylia	bazylia1	0	4	8	20	40	5	24
9	roza	roza1	0	4	8	20	40	10	34
10	storczyk	moj_storczyk	0	4	7	20	40	15	64

Plik create.php:

```
include_once '../config/Database.php';
include_once '../Model/Roslina.php';
// Instantiate DB & connect
$database = new Database();
$db = $database->connect();

// Instantiate blog post object
$Roslina = new Roslina($db);

// Get raw posted data
$data = json_decode(file_get_contents("php://input"));

$Roslina->gatunek = $data->gatunek;
$Roslina->nazwa = $data->nazwa;
$Roslina->min_temperatura = $data->min_temperatura;
$Roslina->max_temperatura = $data->max_temperatura;
$Roslina->min_wilgotnosc_gleby = $data->min_wilgotnosc_gleby;
$Roslina->max_wilgotnosc_gleby = $data->max_wilgotnosc_gleby;
$Roslina->min_wilgotnosc_powietrza = $data->min_wilgotnosc_powietrza;
$Roslina->max_wilgotnosc_powietrza = $data->max_wilgotnosc_powietrza;
// Create Roslina
if($Roslina->create()) {
    echo json_encode(
        array('message' => 'Created')
    );
} else {
    echo json_encode(
        array('message' => 'Not Created')
    );
}
```

Zapytanie Insert:

```
$query = 'INSERT INTO ' .
    $this->table . '
SET
    gatunek = :gatunek,

    nazwa = :nazwa,
    min_temperatura = :min_temperatura,
    max_temperatura = :max_temperatura,
    min_wilgotnosc_gleby = :min_wilgotnosc_gleby,
    max_wilgotnosc_gleby = :max_wilgotnosc_gleby,
    min_wilgotnosc_powietrza = :min_wilgotnosc_powietrza,
    max_wilgotnosc_powietrza = :max_wilgotnosc_powietrza';
```

**Roślina nadpisywanie danych**



Endpoint: PUT 'http://localhost/API/Api/Roslina/update.php'

Metoda ta działa w bardzo podobny sposób do create.php. Tutaj również wysyłamy JSONa z danymi ale musimy podać ID rośliny, której dane chcemy edytować.

Używamy komendy PUT i pliku update.php

PUT ⌵ http://localhost/API/Api/Roslina/update.php Send ⌵

Będziemy chcieli nadpisać rekord utworzony poprzednio. Ma on ID o numerze 10 więc taki podajemy w JSON.

```
1 {
2     .....
3     "gatunek": "magnolia",
4     .....
5     "nazwa": "moja_magnolia",
6     "min_temperatura": "16",
7     "max_temperatura": "64",
8     "min_wilgotnosc_gleby": "4",
9     "max_wilgotnosc_gleby": "7",
10    "min_wilgotnosc_powietrza": "20",
11    "max_wilgotnosc_powietrza": "40",
12    "ID_roslina": "10"
13 }
```

Sprawdzamy czy w bazie zostały wprowadzone zmiany.

roslina	gatunek	nazwa	wiek	min_wilgotnosc_gleby	max_wilgotnosc_gleby	min_wilgotnosc_powietrza	max_wilgotnosc_powietrza	min_temperatura	max_temperatura
1	azalia	azalia1	1	4.04	6	4.6	5.6	7.4	9
7	tulipan	tulipan1	0	4	8	20	40	1	20
8	bazylia	bazylia1	0	4	8	20	40	5	24
10	magnolia	moja_magnolia	0	4	7	20	40	16	64

Plik update.php:

```
1 header('Content-type: application/json');
2 header('Access-Control-Allow-Methods: PUT');
3 header('Access-Control-Allow-Headers: Access-Control-Allow-Headers, Content-Type, Access-Control-Allow-Methods, Authorization');
4
5 include_once '../Config/Database.php';
6 include_once '../Model/Roslina.php';
7
8 // Instantiate DB & connect
9 $database = new Database();
10 $db = $database->connect();
11
12 // Instantiate blog post object
13 $Roslina = new Roslina($db);
14
15 // Get raw posted data
16 $data = json_decode(file_get_contents("php://input"));
17
18 // Set ID to UPDATE
19 $Roslina->ID_roslina = $data->ID_roslina;
20 $Roslina->gatunek = $data->gatunek;
21 $Roslina->nazwa = $data->nazwa;
22 $Roslina->min_temperatura = $data->min_temperatura;
23 $Roslina->max_temperatura = $data->max_temperatura;
24 $Roslina->min_wilgotnosc_gleby = $data->min_wilgotnosc_gleby;
25 $Roslina->max_wilgotnosc_gleby = $data->max_wilgotnosc_gleby;
26 $Roslina->min_wilgotnosc_powietrza = $data->min_wilgotnosc_powietrza;
27 $Roslina->max_wilgotnosc_powietrza = $data->max_wilgotnosc_powietrza;
28
29 // Update post
30 if ($Roslina->update()) {
31     echo json_encode(
32         array('message' => 'Category Updated')
33     );
34 } else {
35     echo json_encode(
36         array('message' => 'Category not updated')
37     );
38 }
```

Zapytanie SQL użyte do nadpisywania danych:

```
$query = 'UPDATE ' .  
    $this->table . '  
SET  
gatunek = :gatunek,  
  
nazwa = :nazwa,  
min_temperatura = :min_temperatura,  
max_temperatura = :max_temperatura,  
min_wilgotnosc_gleby = :min_wilgotnosc_gleby,  
max_wilgotnosc_gleby = :max_wilgotnosc_gleby,  
min_wilgotnosc_powietrza = :min_wilgotnosc_powietrza,  
max_wilgotnosc_powietrza = :max_wilgotnosc_powietrza  
WHERE  
ID_roslina = :ID_roslina';
```

## Roślina usuwanie danych

Endpoint: DELETE 'http://localhost/API/Api/Roslina/delete.php'

DELETE	▼	http://localhost/API/Api/Roslina/delete.php	Send ▼
--------	---	---	--------

Przesyłamy ID rośliny którą chcemy usunąć.

```
{  
  .....  
  .....  
  .....  
  ..... "ID_roslina": "10"  
}
```

Widzimy że usunięto roślinę, którą wybraliśmy do usunięcia

ID_roslina	gatunek	nazwa	wiek	min_wilgotnosc_gleby	max_wilgotnosc_gleby	min_wilgotnosc_powietrza	max_wilgotnosc_powietrza	m
1	azalia	azalia1	1	4.04	6	4.6	5.6	
7	tulipan	tulipan1	0	4	8	20	40	
8	bazylia	bazylia1	0	4	8	20	40	

Z zaznaczonymi: Edytuj Kopiuuj Usuń Eksport

Zapytanie SQL użyte do usuwania pomiaru

```
// Create query
$query = 'DELETE FROM ' . $this->table . ' WHERE ID_roslina = :ID_roslina';
```

## Pomiary pobieranie

Endpoint: GET 'http://localhost/API/Api/Pomiar/read.php'

W analogiczny sposób do rośliny funkcja read.php zwraca wartości pomiarowe. Interesuje nas tylko ostatni pomiar i tylko ten jest wysyłany. Polecenie SQL, które to wykonuje:

```
// Create query
$query = 'SELECT r.nazwa as nazwa_rosliny, p.ID_pomiaru, p.Godzina_pomiaru, p.Id_rosliny, p.Temperatura, p.Wilg_powietrza, p.Wilg_gleby
FROM ' . $this->table . ' p
LEFT JOIN
roslina r ON p.Id_rosliny=r.ID_roslina
ORDER BY
p.Godzina_pomiaru DESC
LIMIT 0,1';
```

Api pobiera wszystkie pomiary oraz nazwę i ID rośliny dla której zostały one wykonane. Pomiary są posortowane względem czasu wykonania i zostaje przesłane tylko jedno z nich czyli to które zostało wykonane jako ostatnie.

GET

http://localhost/API/Api/Pomiar/read.php

Send

```
{
  "ID_pomiaru": 3,
  "Temperatura": 20,
  "Wilg_gleby": 9,
  "Wilg_powietrza": 8,
  "Id_rosliny": 7,
  "nazwa_rosliny": "tulipan1"
}
```

ID_pomiaru	Godzina_pomiaru	Id_rosliny	Temperatura	Wilg_powietrza	Wilg_gleby
1	2022-04-10 14:49:49	1	20	30	30
2	2022-05-31 09:27:39	7	20	8	9
3	2022-05-31 09:27:44	7	20	8	9

Faktycznie pomiar, który został wykonany jako ostatni został wysłany.

## Pomiary dodawanie

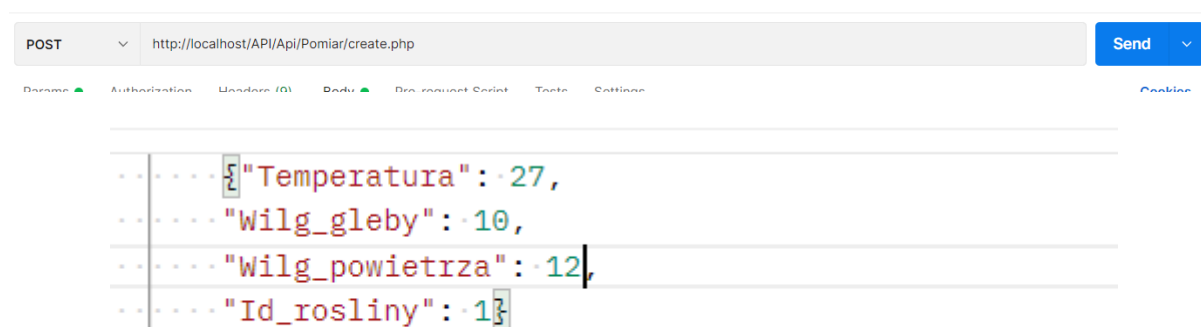
Endpoint: POST 'http://localhost/API/Api/Pomiar/create.php'

Api otrzymuje pomiary oraz ID rosliny w formacie JSON a następnie tworzy nowy pomiar.

Baza danych przed dodaniem pomiaru

ID_pomiaru	Godzina_pomiaru	Id_rosliny	Temperatura	Wilg_powietrza	Wilg_gleby
1	2022-04-10 14:49:49	1	20	30	30
2	2022-05-31 09:27:39	7	20	8	9
3	2022-05-31 09:27:44	7	20	8	9
4	2022-06-18 17:14:07	1	21	9	15

Następnie uruchamiamy naszą funkcję create i wysyłamy pomiary:



W bazie danych został utworzony kolejny rekord więc plik działa poprawnie

ID_pomiaru	Godzina_pomiaru	Id_rosliny	Temperatura	Wilg_powietrza	Wilg_gleby
1	2022-04-10 14:49:49	1	20	30	30
2	2022-05-31 09:27:39	7	20	8	9
3	2022-05-31 09:27:44	7	20	8	9
4	2022-06-18 17:14:07	1	21	9	15
5	2022-06-18 17:19:21	1	27	12	10

Zapytanie SQL użyte do utworzenia tego rekordu

```
// Create query
$query = 'INSERT INTO ' . $this->table . ' SET Temperatura = :Temperatura, Wilg_powietrza = :Wilg_powietrza, Wilg_gleby = :Wilg_gleby, Id_rosliny = :Id_rosliny';
```

