# ORW Ćwiczenie 6 – Tworzenie Web API

Autor: Kornel Ostrowski

Grupa: 125NCI\_B

Link do projektu: <https://github.com/ostrowskik/WebApi_orw_cw6>

Język programowania: C#

**Zadania do zrealizowania:**

1. Za pomocą wybranego przez siebie języka programowania (wraz z frameworkiem) zademonstruj tworzenie Web API.
2. Bądź gotowy do prezentacji możliwości i modyfikacji napisanego programu.

**Przykładowe pytania:**

* Przedstaw 5 klas kodów statusowych dla protokołu HTML. Podaj nazwę każdej z klas oraz czego dotyczą wymienione w niej kody oraz wymień co najmniej 4 kody, które uważasz za najbardziej istotne?

1xx – kody informacyjne

2xx – kody powodzenia

3xx – kody przekierowania

4xx – kody błędów klienta

5xx – kody błędów serwera

Najważniejsze:

1. 100 – „Continue”- prośba o dalsze wysłanie zapytania.
2. 200 – „OK” – treść dokumentu.
3. 201 – „Created” – dokument zapisany na serwerze.
4. 400 – złe zapytanie.
5. 403 – brak dostępu.
6. 404 – brak.
7. 500 – wewnętrzny błąd serwera.

* Wymień podstawowe metody protokołu HTML. Podaj ich znaczenie oraz wskaż ich odpowiedniki w języku SQL

GET – pobranie zasobu – SELECT

HEAD – pobiera informacje o zasobie

PUT – przyjęcie uaktualnianych danych od klienta do serwera – UPDATE

POST – przyjęcie nowych danych od klienta do serwera – INSERT

DELETE – żądanie usunięcia – DELETE

OPTIONS – informacja o opcjach

TRACE – diagnostyka, analiza kanału

CONNECT – pośredniczenie w tunelowaniu

PATCH – aktualizacja części danych - UPDATE

* Narysuj przykładową mapę strony zawierającą co najmniej dwie z ww. metod.
* Co to jest Web API? Omów na podstawie przykładowej architektury.

Jest to interfejs komunikacyjny korzystający z protokołu HTTP i formatu JSON lub XML

* Na czym polega serializacja w sieci WWW? Omów na przykładzie.

Serializacja – proces zamiany obiektów w ciągi znaków je reprezentujące.

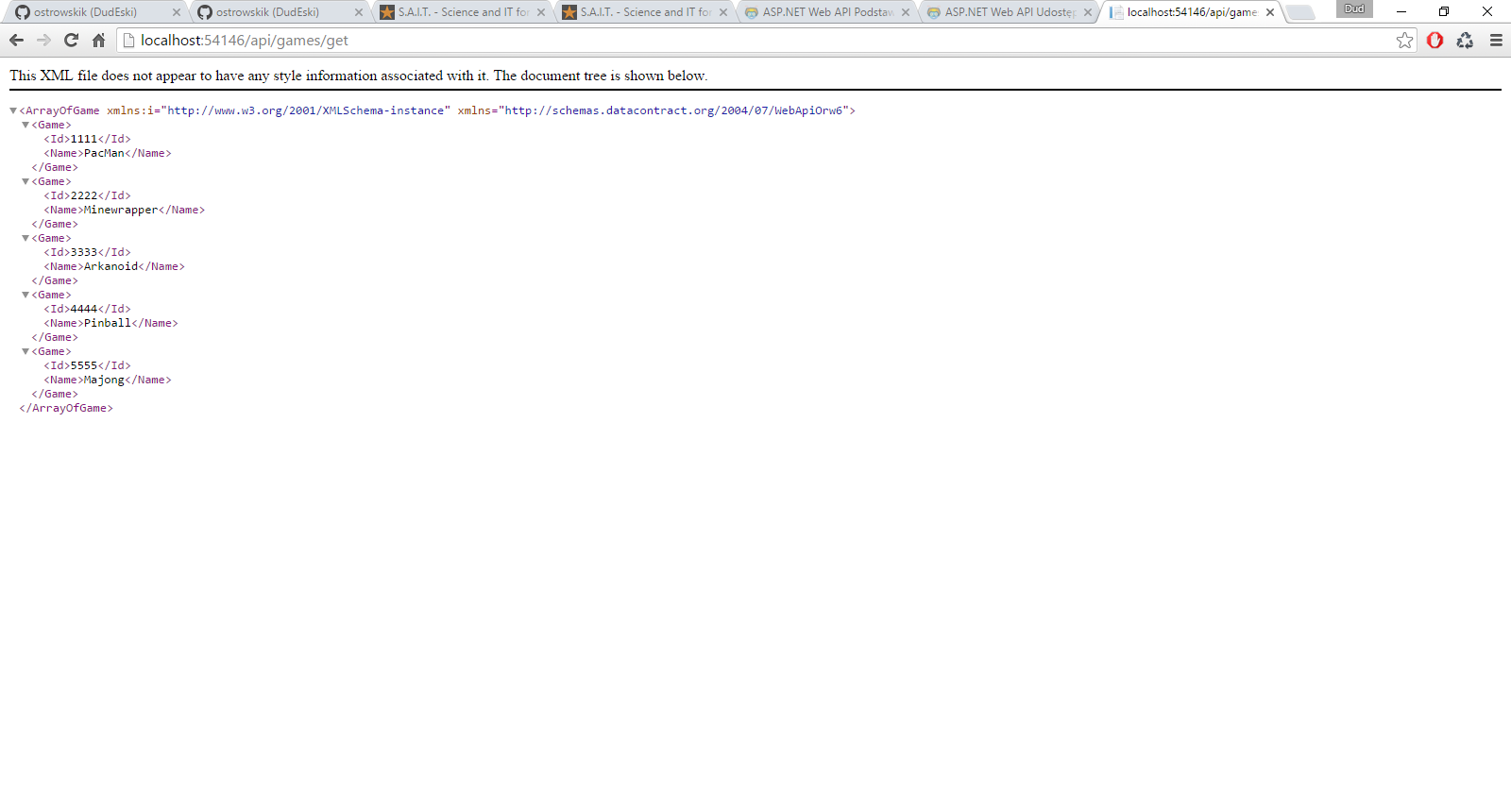
W sieci WWW serializacja przyjmuje najczęściej postać standardu SOAP, XML lub JSON – standardy szeroko używane przez różne Web API.

# Realizacja zadań

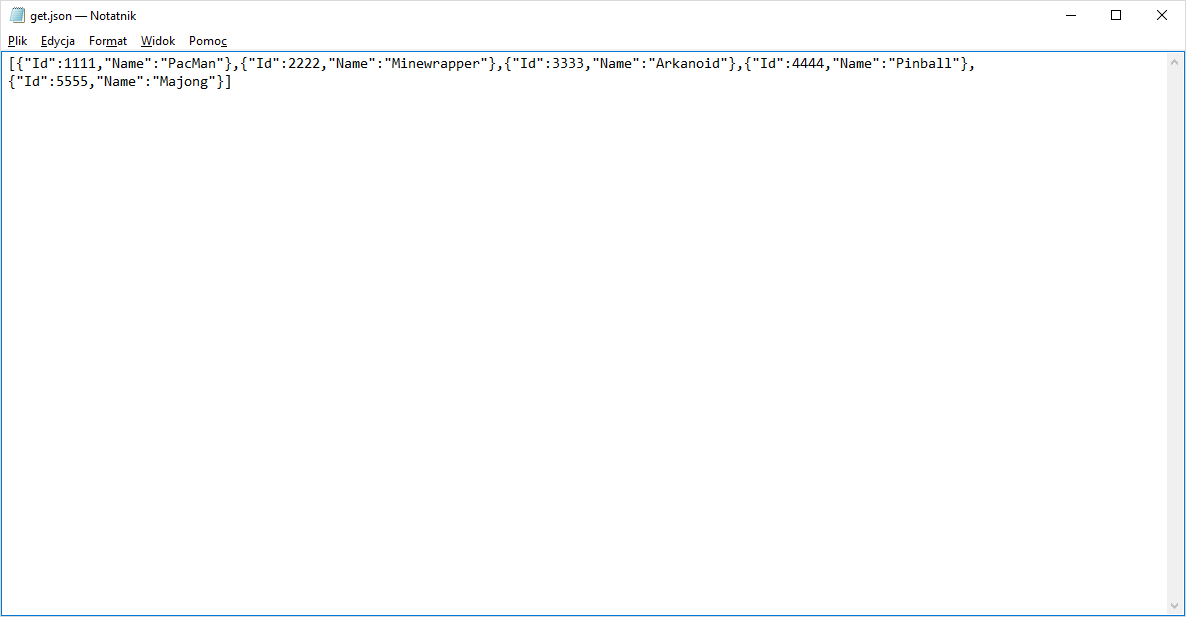
Stworzona została prosta aplikacja, która może być manipulowana przy użyciu protokołu HTTP. WebApi opracowane zostało do komunikacji między serwerem i danymi na serwerze, a – z założenia – aplikacjami mobilnymi (jednak ta funkcjonalność może być rozszerzona). Komunikacja odbywa się przez wysyłanie ciągów JSON.

API pozwala na manipulację danymi zawartymi w aplikacji. Zawarte zostały funkcje takie jak GET, POST, PUT i DELETE.

Przykładowe wywołanie pobrania danych z aplikacji w formacie XML:



Lub te same informacje pobrane w formacie JSON:



Aplikacja posiada klasę obiektu Games, która posiada 2 cechy: Id, Name.

public class Game

{

public int Id { get; set; }

public string Name { get; set; }

}

Dla funkcjonowania API potrzebny jest kontroler, który w naszej aplikacji nazwany jest GamesController, to w nim zawarta jest lista obiektów oraz możliwe funkcje:

public class GamesController : ApiController

{

private static IList<Game> list = new List<Game>()

{

new Game(){Id = 1111,Name="PacMan"},

new Game(){Id = 2222,Name="Minewrapper"},

new Game(){Id = 3333,Name="Arkanoid"},

new Game(){Id = 4444,Name="Pinball"},

new Game(){Id = 5555,Name="Majong"}

};

// GET api/Games

[HttpGet]

public IEnumerable<Game> Get()

{

return list;

}

// GET api/Games/12345

[HttpGet]

public Game Get(int id)

{

return list.First(e => e.Id == id);

}

// POST api/Games

[HttpPost]

public void Post(Game Game)

{

int maxId = list.Max(e => e.Id);

Game.Id = maxId + 1111;

list.Add(Game);

}

// PUT api/Games/12345

[HttpPut]

public void Put(int id, Game Game)

{

int index = list.ToList().FindIndex(e => e.Id == id);

list[index] = Game;

}

// DELETE api/Games/12345

[HttpDelete]

public void Delete(int id)

{

Game Game = Get(id);

list.Remove(Game);

}

}

Aby aplikacja funkcjonowała z serwerem potrzebujemy również zdefiniować tablicę trasowania danych, którą umieszczamy w ustawieniach globalnych aplikacji (Global.asax) :

RouteTable.Routes.MapHttpRoute(

name: "WebApiOrw6",

routeTemplate: "api/{controller}/{action}/{id}",

defaults: new { id = System.Web.Http.RouteParameter.Optional}

);