Ćwiczenia 2

Zadanie 1

Podczas dzisiejszych ćwiczeń rozpoczniemy pracę z aplikacjami serwerowymi typu REST (ang. *Representational State Transfer*). Do wygodnego testowania aplikacji serwerowej przydatne będzie narzędzie pozwalające ręcznie preparować żądania http i wysyłać do naszego serwera. W tym celu zainstalujemy narzędzie Postman. Proszę o zainstalowanie narzędzia z poniższej strony. Jeśli ktoś pracuje na komputerze szkolnym – może również zainstalować narzędzie Postman w formie pluginu do przeglądarki Google Chrome.

Zadanie 2

W niniejszym zadaniu proszę przygotować nowy projekt aplikacji ASP.NET Core Web Application. Proszę wybrać szablon aplikacji typu „API". Przygotowywana aplikacja ma wspierać uczelnię w procesie zarządzania danymi studentów.

1. Proszę przygotować kontroler o nazwie **StudentsController**
2. Kontroler powinien mieć **4 metody publiczne** związane z zarządzaniem danymi na temat studentów.
3. Aplikacja powinna zapisywać wszystkie dane w **lokalnej bazie danych w pliku CSV**. Plik powinien mieć następującą postać.

Imie,nazwisko,numerIndeksu,dataUrodzenia,studia,tryb,email,imię ojca, imię matki

Jan,Kowalski,s1234,3/20/1991,Informatyka,Dzienne,kowalski@wp.pl,Jan,Anna

4. API powinno zwracać dane w formacie JSON.
5. Pierwsza końcówka powinna odpowiadać na żądanie **typu HTTP GET na adres „/students".** W przypadku żądania bez żadnych parametrów końcówka powinna **pobrać listę studentów z lokalnej bazy danych – pliku**. Możemy powiedzieć, że końcówka powinna zwracać rezultat podobny do zapytania „select * from students".
6. Dodatkowo końcówka GET „/students" pozwala na przekazania parametru w querystring'u, który daje nam możliwość pobrania konkretnego studenta.
   Przykład żądania **GET „/students/s1234"**
   W takim wypadku powinniśmy zwrócić pojedynczego studenta o konkretnym numerze indeksu.
7. Druga końcówka odpowiadająca na żądanie **PUT „/students/s1234"** pozwala nam **zaktualizować dane na temat konkretnego studenta**. Końcówka przyjmuje dane przesłane w formacie JSON. Uwaga: końcówka oczekuje przesłania kompletnych danych na temat studenta. Wyłącznie pole „numer indeksu" nie ulega modyfikacji i traktowane jest jako identyfikator studenta. Dodatkowo w przypadku sukcesu końcówka powinna zwracać aktualne dane związane z zapisanym studentem.
8. Trzecia końcówka powinna umożliwić nam **dodanie nowego studenta wykonując żądanie POST** na adres „/students". Końcówka przyjmuje dane nowego studenta w ciele żądania w formacie JSON. Przed dodaniem nowego studenta powinniśmy upewnić się czy:

   a. Wszystkie dane na temat studenta są kompletne. W przeciwnym wypadku zwracamy odpowiedni kod błędu.
   b. Sprawdzamy czy wybrany numer indeksu jest unikalny i czy jego format jest poprawny.

   Studenta dodajemy na koniec pliku CSV stanowiącego naszą bazę danych.
9. Czwarta końcówka powinna reagować na żądania typu DELETE „/students/s1234” i pozwala nam **usunąć konkretnego studenta z bazy danych**.
10. Pamiętaj o zwracaniu poprawnych kodów HTTP i ewentualnej obsłudze błędów.