# LAB4. Stworzenie micro serwisu do zarządzania kontami użytkowników

W celu przygotowania do laboratorium zachęcam do ściągnięcia brancha z gita o nazwie **lab4**, przeczytanie informacji znajdujących się pod linkami na dole instrukcji oraz zaopatrzenie się w narzędzie do testowania restowych requestów. (polecam postman – dodatek do chroma).

Na wstępie proszę wykonać polecenie Update-Database w Package Manager Console, ustawiając jako default project: UsersDataAccess – w przypadku problemów, można usunąć bazę UniversityIot.Users i wykonać polecenie jeszcze raz.

Wszystkie zadania poruszają się w obrębie projektu UniversityIot.UsersService oraz polegają na rozszerzaniu kontrolera UsersController.

Powodzenia! ☺

**Zadania:**

1. Utworzyć metodę umożliwiającą pobranie danych istniejącego użytkownika.
   Adres metody:
   GET /users/{id}
   Metoda powinna:
   - Przyjmować id użytkownika jako parametr
   - Wykorzystać metodę z dataservice: GetUser
   - Zwrócić kod HTTP 404, gdy użytkownik z danym id nie istnieje
   - Zwracać kod HTTP 200 z obiektem użytkownika (model UserViewModel z pominięciem jego hasła)

2. Utworzyć metodę umożliwiającą dodanie nowego użytkownika.
   Adres metody:
   POST /users
   Metoda powinna:
   - Użyć przygotowanego modelu AddUserViewModel jako parametr wejściowy
   - Walidować wejściowy obiekt
     - Password oraz Name nie może być puste
     - CustomerNumber nie może być puste oraz musi mieć 10 znaków
   - Wykorzystać metody z dataservice: GetUser, AddUser
   - Zwrócić kod HTTP 400, jeśli któryś z parametrów jest niewłaściwy wraz z listą błędów.
   - Zwrócić kod HTTP 200 oraz obiekt dodanego użytkownika (z pominięciem jego hasła) – w przypadku sukcesu

3. Utworzyć metodę umożliwiającą usunięcie danego użytkownika
   Adres metody:
   DELETE /users/{id}
   Metoda powinna:
   - Przyjmować id użytkownika jako parametr
   - Wykorzystać metody z dataservice: GetUser, DeleteUser
   - Zwrócić kod HTTP 404 gdy użytkownik z danym id nie istnieje
   - Zwracać kod HTTP 200, gdy operacja zakończy się sukcesem

4. Utworzyć metodę umożliwiającą edycję danych istniejącego użytkownika.
   Adres metody:
   PUT /users/{id}
   Metoda powinna:
   - Przyjmować id użytkownika jako parametr oraz model EditUserViewModel jako BODY
   - Walidować wejściowy obiekt
     - CustomerNumber nie może być puste oraz musi mieć 10 znaków
   - Wykorzystać metodę z dataservice: UpdateUser
   - Zwrócić kod HTTP 400, jeśli któryś z parametrów jest niewłaściwy wraz z listą błędów.
   - Zwrócić kod HTTO 404 gdy użytkownik z danym id nie istnieje
   - Zwracać kod HTTP 200 z zaktualizowanym obiektem użytkownika (z pominięciem jego hasła)

5. * Zastąpienie ręcznego mapowania modeli AutoMapperem

**Użyteczne linki:**

Rest ogólnie:
http://www.yarpo.pl/2012/07/29/rest-ciekawszy-sposob-na-komunikacje-client-server/

REST przy użyciu WebAPI:
https://www.exceptionnotfound.net/using-http-methods-correctly-in-asp-net-web-api/

Routing parameters:
https://www.asp.net/web-api/overview/web-api-routing-and-actions/attribute-routing-in-web-api-2

Walidacja:
https://www.asp.net/web-api/overview/formats-and-model-binding/model-validation-in-aspnet-web-api

AutoMapper:
https://github.com/AutoMapper/AutoMapper/wiki/Getting-started