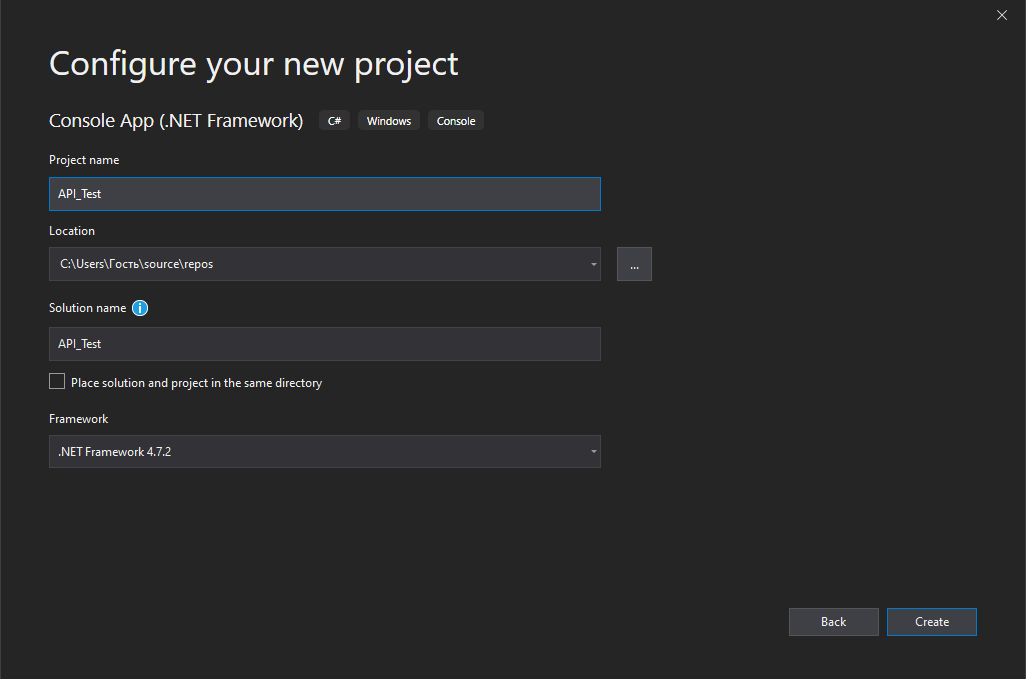
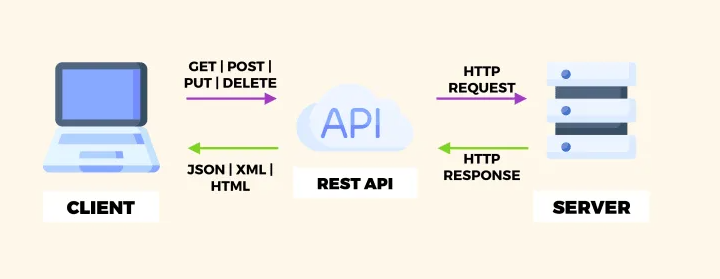
Тестовый API, который будем тестировать - <https://reqres.in/>



1 Создаём консольное приложение

**Как работает API:**



**Нас интересует только два понятия – Запрос (request) и Ответ (response) от сервера (API).**

**У запроса есть методы – GET, POST, DELETE, PUT.**

**Каждый метод позволяет взаимодействовать с нашим API:**

**GET – Получить какие-то данные или их массив;**

**POST – Отправить на сервер какие-либо данные;**

**DELETE – Удалить какие-то данные или их массив;**

**PUT – Обновить конкретные данные;**

**Можно привести аналогию к SQL-запросам:**

**GET – SELECT, POST – INSERT INTO, DELETE – DELETE, PUT – UPDATE;**

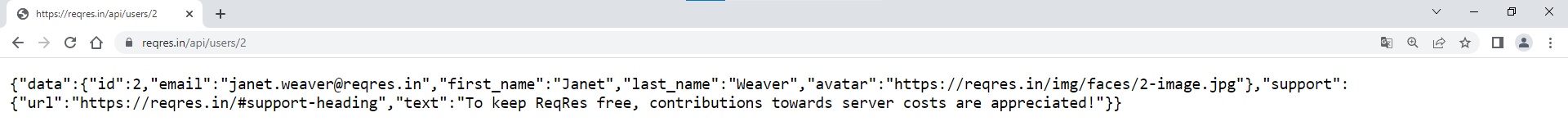
**Причем тут SQL запросы? API связан с БД, отправляет и получает информацию оттуда, поэтому мы будем делать все эти запросы, используя запросы и ответы.**

**Запросы к API в браузере осуществляются в адресной строке:**

****

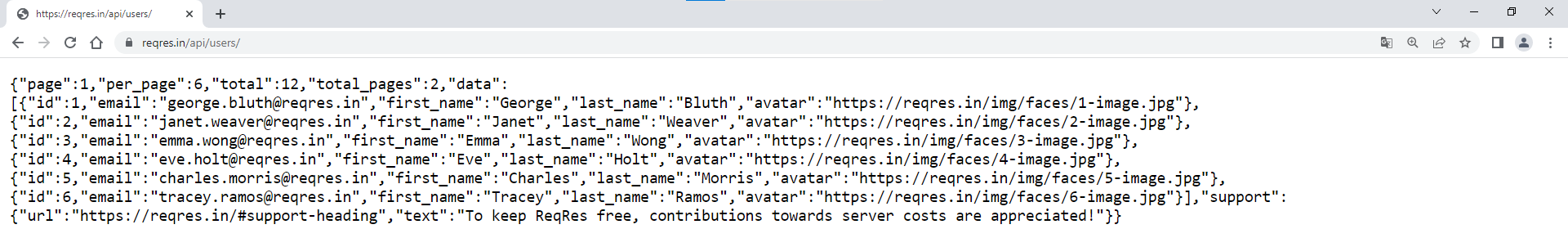
**Из названия в адресной строке понятно, что данный метод API работает с пользователями какой-либо БД. api – говорит о том, что используется непосредственно сам api, users – о том, что мы работаем с пользователями (из какой-то БД), 2 – в данном случае id пользователя.**

**Если перейти по этой ссылке, получим следующий ответ от сервера:**

****

**{"data":{"id":2,"email":"janet.weaver@reqres.in","first\_name":"Janet","last\_name":"Weaver","avatar":"https://reqres.in/img/faces/2-image.jpg"},"support":{"url":"https://reqres.in/#support-heading","text":"To keep ReqRes free, contributions towards server costs are appreciated!"}}**

**Сервер вернул нам ответ, в формате JSON.**

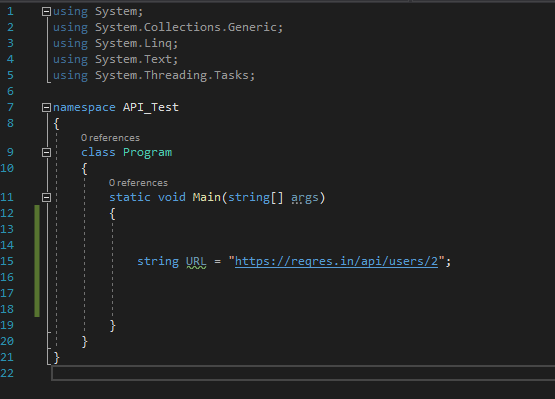
**Если ввести ту же строку, но без id пользователя, получим:** ****

**{"page":1,"per\_page":6,"total":12,"total\_pages":2,"data":[{"id":1,"email":"george.bluth@reqres.in","first\_name":"George","last\_name":"Bluth","avatar":"https://reqres.in/img/faces/1-image.jpg"},{"id":2,"email":"janet.weaver@reqres.in","first\_name":"Janet","last\_name":"Weaver","avatar":"https://reqres.in/img/faces/2-image.jpg"},{"id":3,"email":"emma.wong@reqres.in","first\_name":"Emma","last\_name":"Wong","avatar":"https://reqres.in/img/faces/3-image.jpg"},{"id":4,"email":"eve.holt@reqres.in","first\_name":"Eve","last\_name":"Holt","avatar":"https://reqres.in/img/faces/4-image.jpg"},{"id":5,"email":"charles.morris@reqres.in","first\_name":"Charles","last\_name":"Morris","avatar":"https://reqres.in/img/faces/5-image.jpg"},{"id":6,"email":"tracey.ramos@reqres.in","first\_name":"Tracey","last\_name":"Ramos","avatar":"https://reqres.in/img/faces/6-image.jpg"}],"support":{"url":"https://reqres.in/#support-heading","text":"To keep ReqRes free, contributions towards server costs are appreciated!"}}**

**Тот же формат JSON, но теперь сервер вернул нам всех пользователей.**

**Осталось научиться обращаться к серверу на ЯП C# и преобразовывать JSON для выборки конкретного значения. Приступим.**

**Прежде мы создали пустое консольное приложение, в него нам необходимо добавить ссылку на API:**

****

**Далее, нам нужно создать запрос по данной ссылке, ведущей на API:**

****

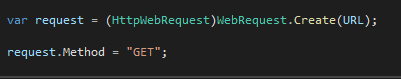
**Но мы преобразуем WebRequest к HttpWebRequest (позднее узнаете зачем):**

****

**Или**

****

**Запрос создан, по стандарту он имеет метод – GET, но мы укажем его, чтобы потом можно было менять метод:**

****

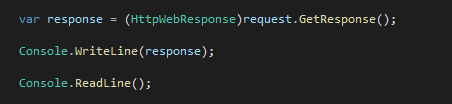
**Запрос создан, приложение можно запускать, но оно нам ничего не покажет, потому что не получен ответ от сервера. Получим его:**

****

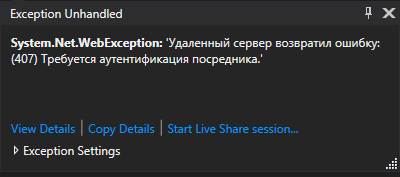
**Также преобразуем к HttpWebResponse:**

****

**Попробуем запустить и вывести ответ от сервера:**

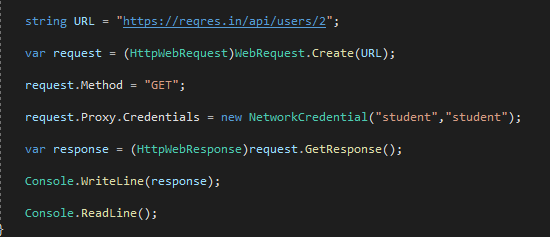
****

**…**

****

**Данная ошибка – из-за прокси нашего колледжа, который требует авторизоваться (ввести student student). Проблема решается одной строчкой:**

**request.Proxy.Credentials = new NetworkCredential("student","student");**

****

**Запустим еще раз:**

****

**Как видим, выводится ответ в формате запроса, нам надо его чем-то прочитать, в этом нам поможем StreamReader:**

****

**А в него поместим результат запроса:**

****

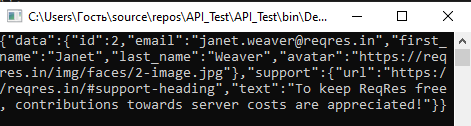
**Теперь, используя reader, поместим весь текст в обычный string элемент:**

****

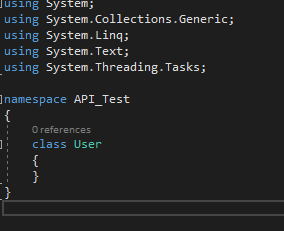
**И выведем Text:**

****

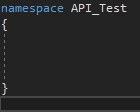
**…**

****

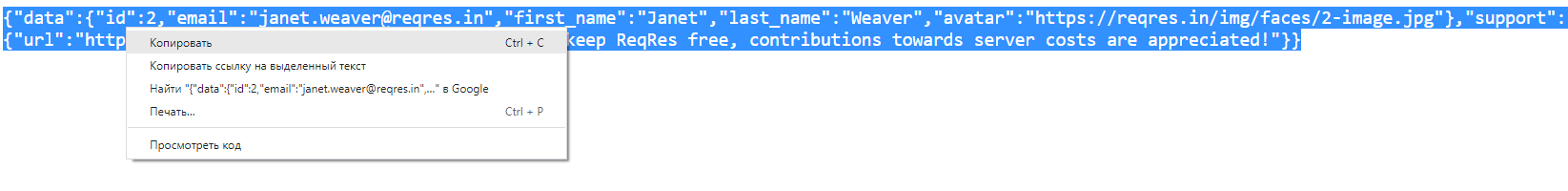
**Как видим, мы добились того же результата, что видели в браузере. Теперь нам необходимо сделать из этих данных что-нибудь реальное. Первым делом создадим класс User:**

****

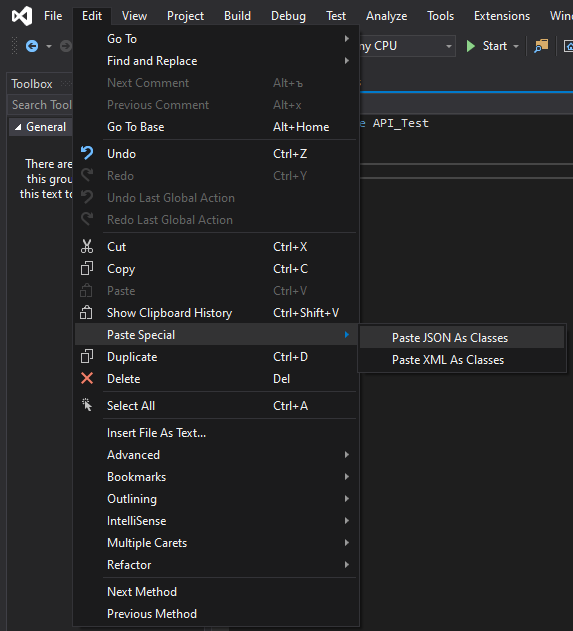
**Далее, чтобы меньше отвлекало, удалим все библиотеки using, а также сам класс User:**

****

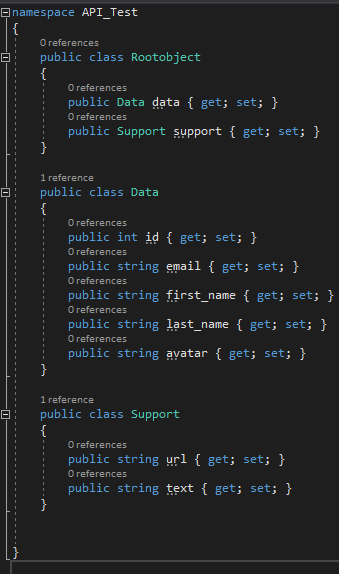
**Далее, идем по ссылке, и копируем ответ сервера, в формате json:**

****

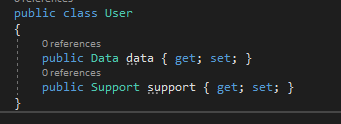
**Затем, используя специальную вставку, Visual Studio сам создаст нам класс:**

****

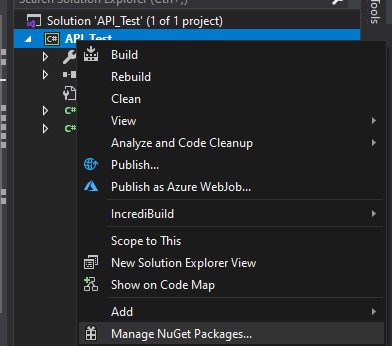
**…**

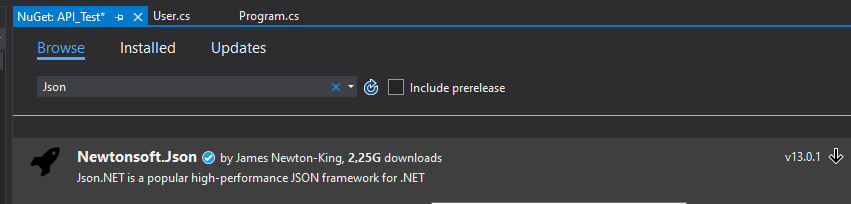
****

**Далее, нам необходимо поменять название класса Rootobject на User, этот класс объединяет в себе класс Data и класс Support:**

****

**Теперь вернемся к коду для того, чтобы привести ответ JSON к нашему классу, понадобится специальный пакет NuGet:**

****

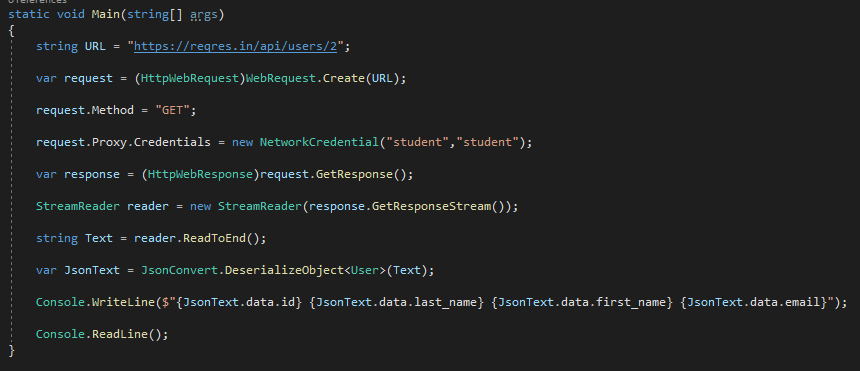
****

****

**Устанавливаем. Сейчас мы будем десериализовывать (подробнее –** [**тут**](https://www.c-sharpcorner.com/article/json-serialization-and-deserialization-in-c-sharp/#:~:text=%D0%A1%D0%B5%D1%80%D0%B8%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F%20%2D%20%D1%8D%D1%82%D0%BE%20%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%20%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F,%D0%BA%D0%B0%D0%BA%20%D0%B4%D0%B5%D1%81%D0%B5%D1%80%D0%B8%D0%B0%D0%BB%D0%B8%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%20JSON%20%D0%B2%20C%23) **) наш JSON ответ:**

****

**А вот теперь мы можем брать из JsonText любые данные:**

****

**…**

**Результат:**

****

**Можем поменять id пользователя в ссылке:**

****

**Результат программы:**

****

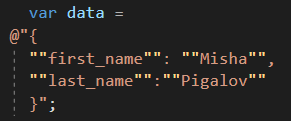
**Тем самым, при помощи метода GET, мы получаем данные, используя API.**

**Далее попробуем метод POST.**

**Меняем метод с GET на POST.**

****

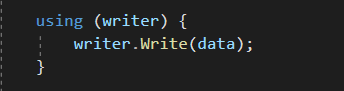
**Далее думаем, какого пользователя мы создадим:**

****

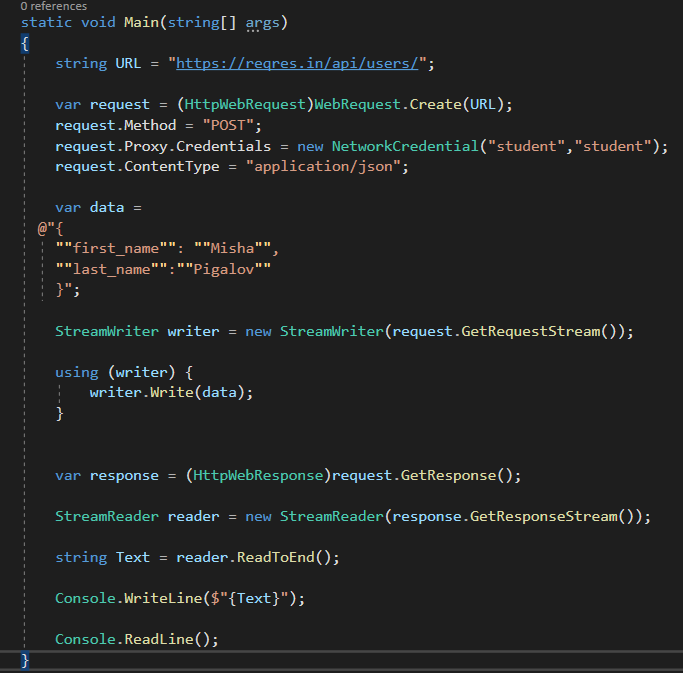
**Для того чтобы записать на сервер нового пользователя нам нужно использовать StreamWriter:**

****

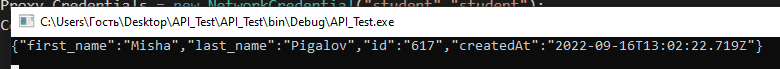
**Осталось только записать:**

****

**А теперь, по старой схеме нам необходимо получить ответ от сервера, чтобы узнать, что же там произошло, но теперь, преобразовывать к классу нам не надо:**

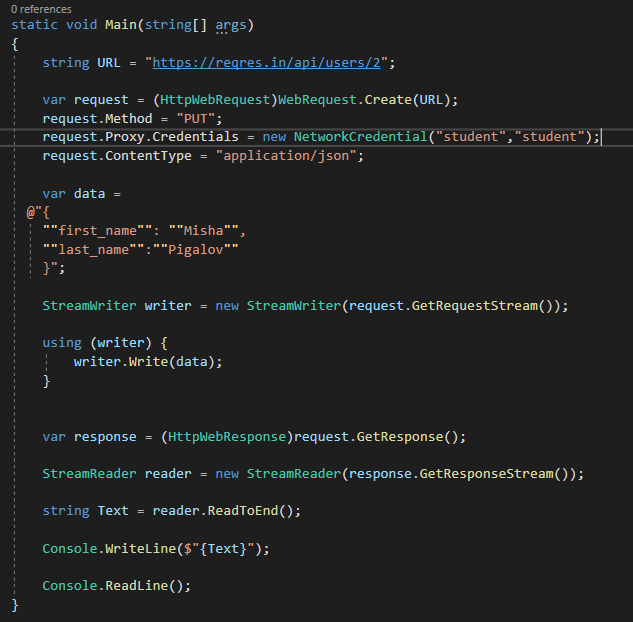
****

**…**

**  
Как видим, все успешно!**

**Метод PUT:**

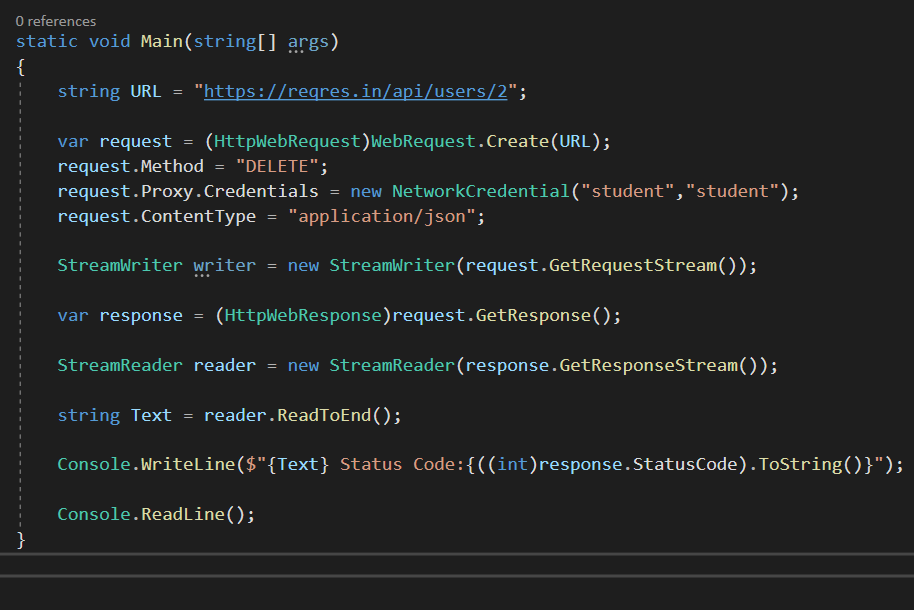
**Тут всё аналогично методу POST, единственное, надо поменять метод, а также в строке указать Id пользователя, которого будем менять:**

****

**…**

****

**Метод DELETE:**

****