

**федеральное государственное автономное образовательное
учреждение высшего образования**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
(ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ)
(Факультет информационных технологий)**

*(Институт Принтмедиа и информационных технологий)
Кафедра Информатики и информационных технологий*

направление подготовки

09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № 3

Дисциплина: BackEnd-разработка

Тема: Создание веб-API приложения на основе ASP.NET Core

Выполнил(а): студент(ка) группы 221-3711

Ежов Тимофей Алексеевич
(Фамилия И.О.)

Дата, подпись 28.02.2024

Проверил: _____
(Фамилия И.О., степень, звание) **(Оценка)**

Дата, подпись _____
(Дата) (Подпись)

Замечания: _____

Москва 2024

Этапы:

- 1) Создание пустого проекта из шаблона веб-аpi ASP.Net core в Visual Studio

Создание проекта

Последние шаблоны проектов

Веб-API ASP.NET Core (Майкрософт) C#

Настроить новый проект

Веб-API ASP.NET Core (Майкрософт) C# Linux macOS Windows API Облако Служба Веб Web API

Имя проекта

LAB3(11)

Расположение

F:\Репозитории\pc-notebook\BackEnd

Имя решения ⓘ

LAB3(11)

☐ Поместить решение и проект в одном каталоге

Проект будет создан в "F:\Репозитории\pc-notebook\BackEnd\LAB3(11)\LAB3(11)\"

⚠ Этот каталог не пуст.

Назад Далее

- 2) Создаём Класс для хранения информации

```
Ссылка: 10
public class Human
{
    Ссылка: 9
    public string Id { get; set; } = "";
    Ссылка: 6
    public string Name { get; set; } = "";
    Ссылка: 6
    public string Mast { get; set; } = "";

    Ссылка: 4
    public RabochiyClass Class { get; set; } = RabochiyClass.aristo;
    Ссылка: 6
    public int Age { get; set; }
}

Ссылка: 6
public enum RabochiyClass
{
    worker, burjua, aristo, science
}
```

И массив из этих классов, в котором будет храниться информация

```
List<Human> Humani = new List<Human>
{
    new() { Id = Guid.NewGuid().ToString(), Name = "Oleg1", Mast = "Prog", Class = RabochiyClass.science, Age = 21 },
    new() { Id = Guid.NewGuid().ToString(), Name = "Oleg2", Mast = "Manager", Class = RabochiyClass.aristo, Age = 38 },
    new() { Id = Guid.NewGuid().ToString(), Name = "Oleg3", Mast = "Ozhaka", Class = RabochiyClass.worker, Age = 22 },
    new() { Id = Guid.NewGuid().ToString(), Name = "Vital", Mast = "SoBR", Class = RabochiyClass.burjua, Age = 22 }
};
```

3) Реализуем работу с помощью метода Map()

```
app.MapGet("/api/users", () => users);

app.MapGet("/api/users/{id}", (string id) =>
{
    Human? user = users.FirstOrDefault(u => u.Id == id);
    if (user == null) return Results.NotFound(new { message = "Пользователь не найден" });

    return Results.Json(user);
});

app.MapDelete("/api/users/{id}", (string id) =>
{
    Human? user = users.FirstOrDefault(u => u.Id == id);

    if (user == null) return Results.NotFound(new { message = "Пользователь не найден" });

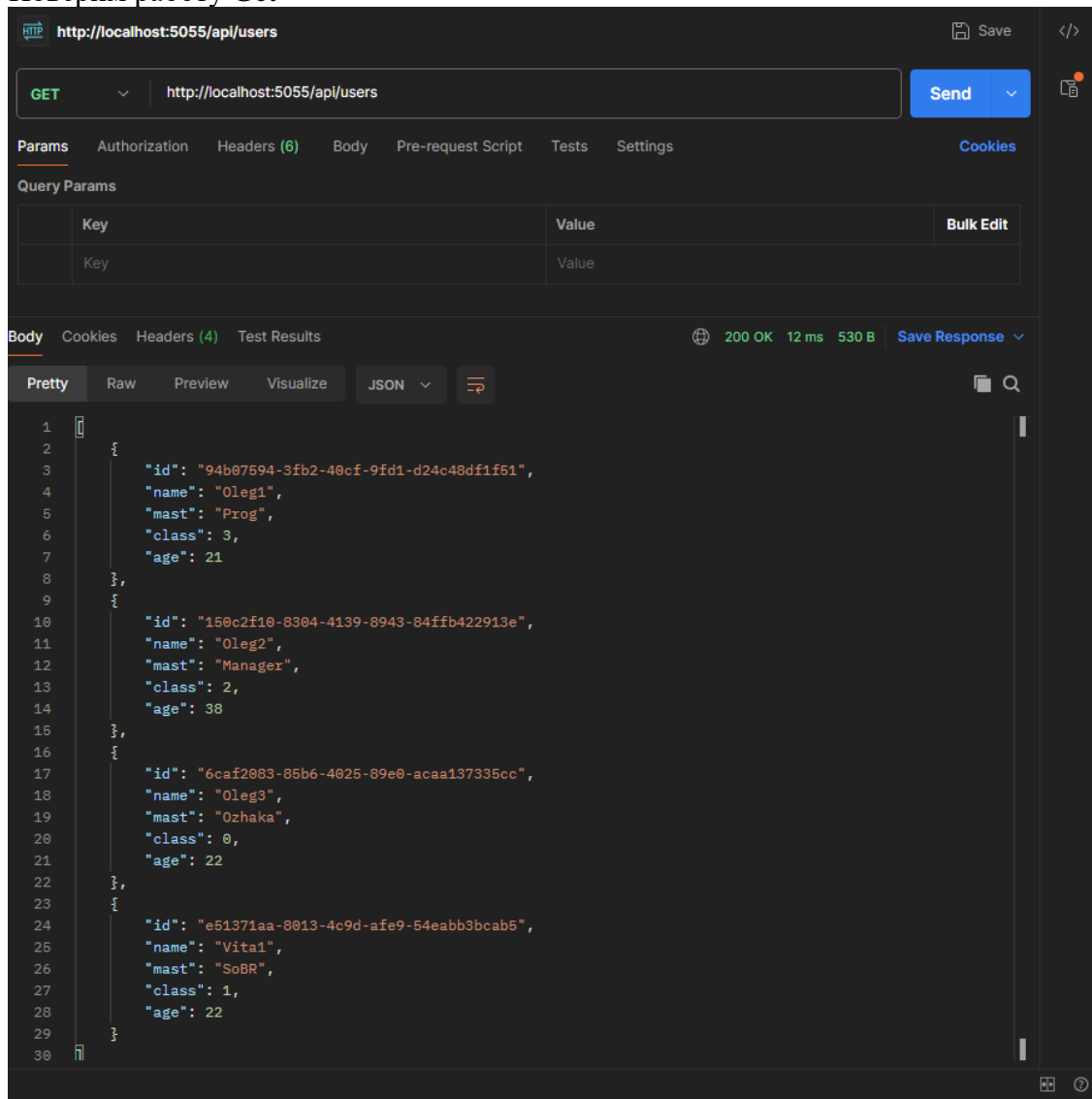
    users.Remove(user);
    return Results.Json(user);
});

app.MapPost("/api/users", (Human user) =>
{
    user.Id = Guid.NewGuid().ToString();
    users.Add(user);
    return user;
});

app.MapPut("/api/users", (Human userData) =>
{
    var user = users.FirstOrDefault(u => u.Id == userData.Id);
    if (user == null) return Results.NotFound(new { message = "Пользователь не найден" });

    user.Age = userData.Age;
    user.Name = userData.Name;
    user.Mast = userData.Mast;
    return Results.Json(user);
});
```

4) Поверим работу Get



The screenshot shows the Chrome DevTools network tab with a GET request to `http://localhost:5055/api/users`. The response status is 200 OK, and the body is a JSON array of four user objects. The response is displayed in the 'Pretty' view.

URL: `http://localhost:5055/api/users`

Method: **GET**

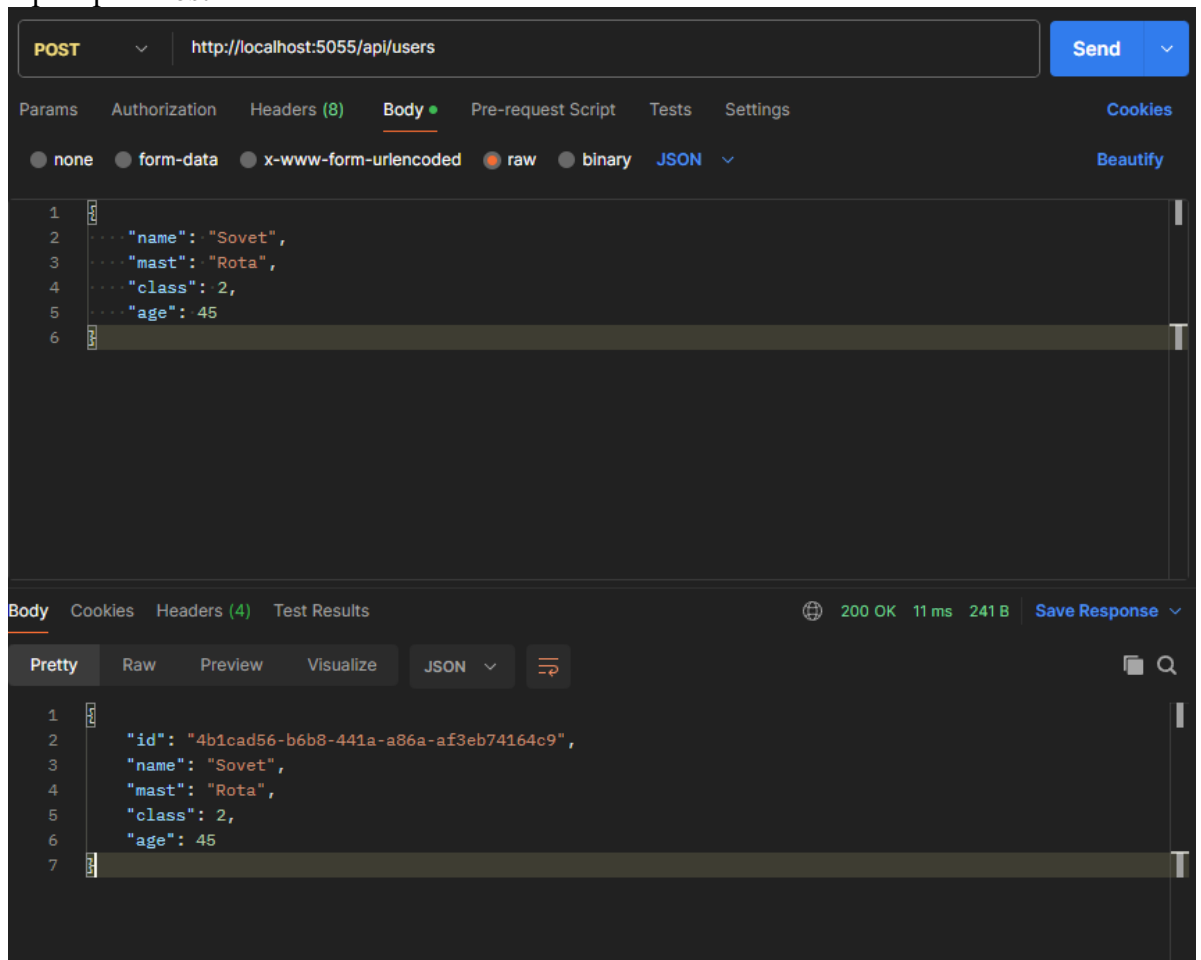
Query Params:

Key	Value
Key	Value

Body:

```
{
  "id": "94b07594-3fb2-40cf-9fd1-d24c48df1f51",
  "name": "Oleg1",
  "mast": "Prog",
  "class": 3,
  "age": 21
},
{
  "id": "150c2f10-8304-4139-8943-84ffb422913e",
  "name": "Oleg2",
  "mast": "Manager",
  "class": 2,
  "age": 38
},
{
  "id": "6caf2083-85b6-4025-89e0-acaa137335cc",
  "name": "Oleg3",
  "mast": "Ozhaka",
  "class": 0,
  "age": 22
},
{
  "id": "e51371aa-8013-4c9d-afe9-54eabb3bcab5",
  "name": "Vita1",
  "mast": "SoBR",
  "class": 1,
  "age": 22
}
```

5) Проверим Post



Id создаётся автоматически и всё работает

6) Проверим Put

The screenshot shows a REST client interface with a PUT request to `http://localhost:5055/api/users`. The request body is a JSON object:

```
1 {
2   "id": "4b1cad56-b6b8-441a-a86a-af3eb74164c9",
3   "name": "Kira",
4   "mast": "Ubera",
5   "class": 2,
6   "age": 45
7 }
```

The response status is 200 OK, 13 ms, 241 B. The response body is displayed in JSON format:

```
1 {
2   "id": "4b1cad56-b6b8-441a-a86a-af3eb74164c9",
3   "name": "Kira",
4   "mast": "Ubera",
5   "class": 2,
6   "age": 45
7 }
```

7) Проверим DELETE

The screenshot shows a REST client interface with a DELETE request to `http://localhost:5055/api/users/d43b4415-eb00-4b38-aea3-53731cef4acb`. The response status is 200 OK, 13 ms, 241 B. The response body is displayed in JSON format:

```
1 {
2   "id": "d43b4415-eb00-4b38-aea3-53731cef4acb",
3   "name": "Vita1",
4   "mast": "SoBR",
5   "class": 1,
6   "age": 22
7 }
```

По id нашли и удалили

Для проверки делаем финальный Get и смотрим что осталось после работы

The screenshot shows a REST client interface with a GET request to `http://localhost:5055/api/users`. The response is a JSON array of four user objects, displayed in the 'Pretty' view. The status is 200 OK, with a response time of 7 ms and a body size of 530 B.

```
[{"id": "1cefc9f1-2c7e-4e49-8d73-474009056174", "name": "Oleg1", "mast": "Prog", "class": 3, "age": 21}, {"id": "c178fdcd-66b4-401d-b558-606aff0bd680", "name": "Oleg2", "mast": "Manager", "class": 2, "age": 38}, {"id": "eaa8592e-a602-402d-8f5c-95fdb990edd2", "name": "Oleg3", "mast": "Ozhaka", "class": 0, "age": 22}, {"id": "4b1cad56-b6b8-441a-a86a-af3eb74164c9", "name": "Kira", "mast": "Ubera", "class": 2, "age": 45}]
```

Листинг

```
public class Human
{
    public string Id { get; set; } = "";
    public string Name { get; set; } = "";
    public string Mast { get; set; } = "";

    public RabochiyClass Class { get; set; } = RabochiyClass.aristo;
    public int Age { get; set; }
}

public enum RabochiyClass
{
    worker, burjua, aristo, science
}
```

```
}  
}
```

```
using LAB3_11_;  
  
List<Human> Humani = new List<Human>  
{  
    new() { Id = Guid.NewGuid().ToString(), Name = "Oleg1", Mast = "Prog", Class =  
RabochiyClass.science, Age = 21 },  
    new() { Id = Guid.NewGuid().ToString(), Name = "Oleg2", Mast = "Manager", Class  
= RabochiyClass.aristo, Age = 38 },  
    new() { Id = Guid.NewGuid().ToString(), Name = "Oleg3", Mast = "Ozhaka", Class  
= RabochiyClass.worker, Age = 22 },  
    new() { Id = Guid.NewGuid().ToString(), Name = "Vital", Mast = "SoBR", Class =  
RabochiyClass.burjua, Age = 22 }  
};  
  
var builder = WebApplication.CreateBuilder();  
var app = builder.Build();  
  
app.UseDefaultFiles();  
app.UseStaticFiles();  
  
app.MapGet("/api/users", () => Humani);  
  
app.MapGet("/api/users/{id}", (string id) =>  
{  
    Human? temp = Humani.FirstOrDefault(u => u.Id == id);  
    if (temp == null) return Results.NotFound(new { message = "Пользователь не  
найден" });  
  
    return Results.Json(temp);  
});  
  
app.MapDelete("/api/users/{id}", (string id) =>  
{  
    Human? temp = Humani.FirstOrDefault(u => u.Id == id);  
  
    if (temp == null) return Results.NotFound(new { message = "Пользователь не  
найден" });  
  
    Humani.Remove(temp);  
    return Results.Json(temp);  
});  
  
app.MapPost("/api/users", (Human user) =>  
{  
    user.Id = Guid.NewGuid().ToString();  
    Humani.Add(user);
```



```
        return user;
    });

app.MapPut("/api/users", (Human userData) =>
{
    var user = Humani.FirstOrDefault(u => u.Id == userData.Id);
    if (user == null) return Results.NotFound(new { message = "Пользователь не найден" });

    user.Age = userData.Age;
    user.Name = userData.Name;
    user.Mast = userData.Mast;
    return Results.Json(user);
});

app.Run();
```