Министерство науки и высшего образования Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и программной инженерии

**Лабораторная работа №4**

**по дисциплине**

**«Технологии программирования»**

**WEB-СЕРВИСЫ (СЛУЖБЫ)**

**Выполнил**:

ст. гр. ПРИ-120

Д. А. Грачев

**Принял**:

Данилов В. В.

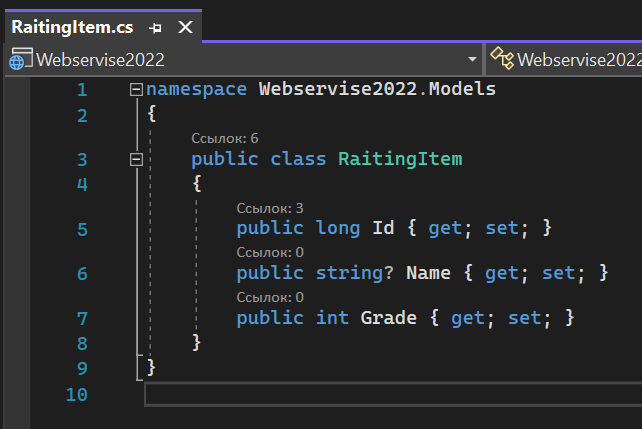
Владимир, 2021

ЦЕЛЬ РАБОТЫ

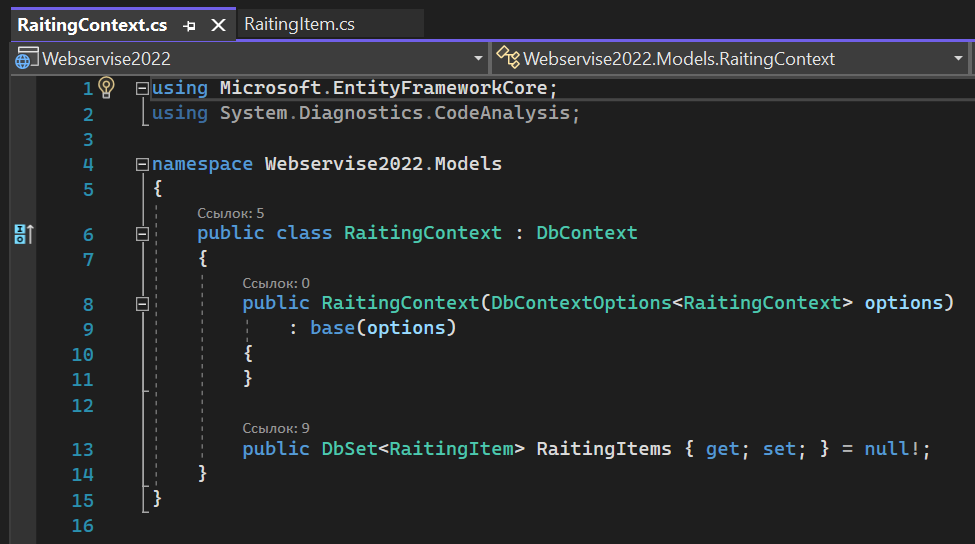
Познакомиться с технологией создания и использования веб-сервисов.

ВЫПОЛНЕНИЕ РАБОТЫ

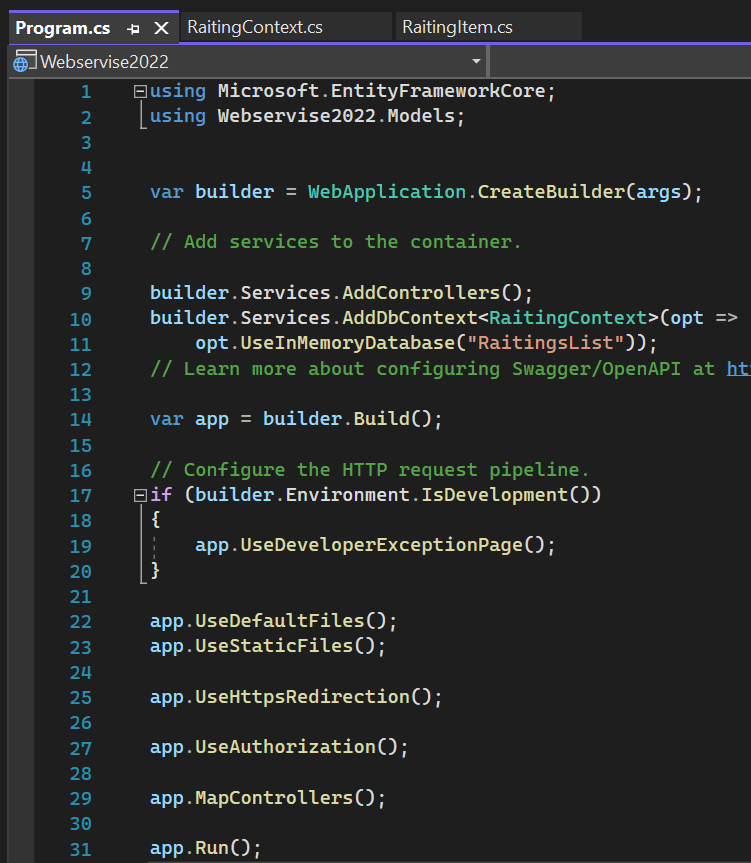
1. Был создан класс модели



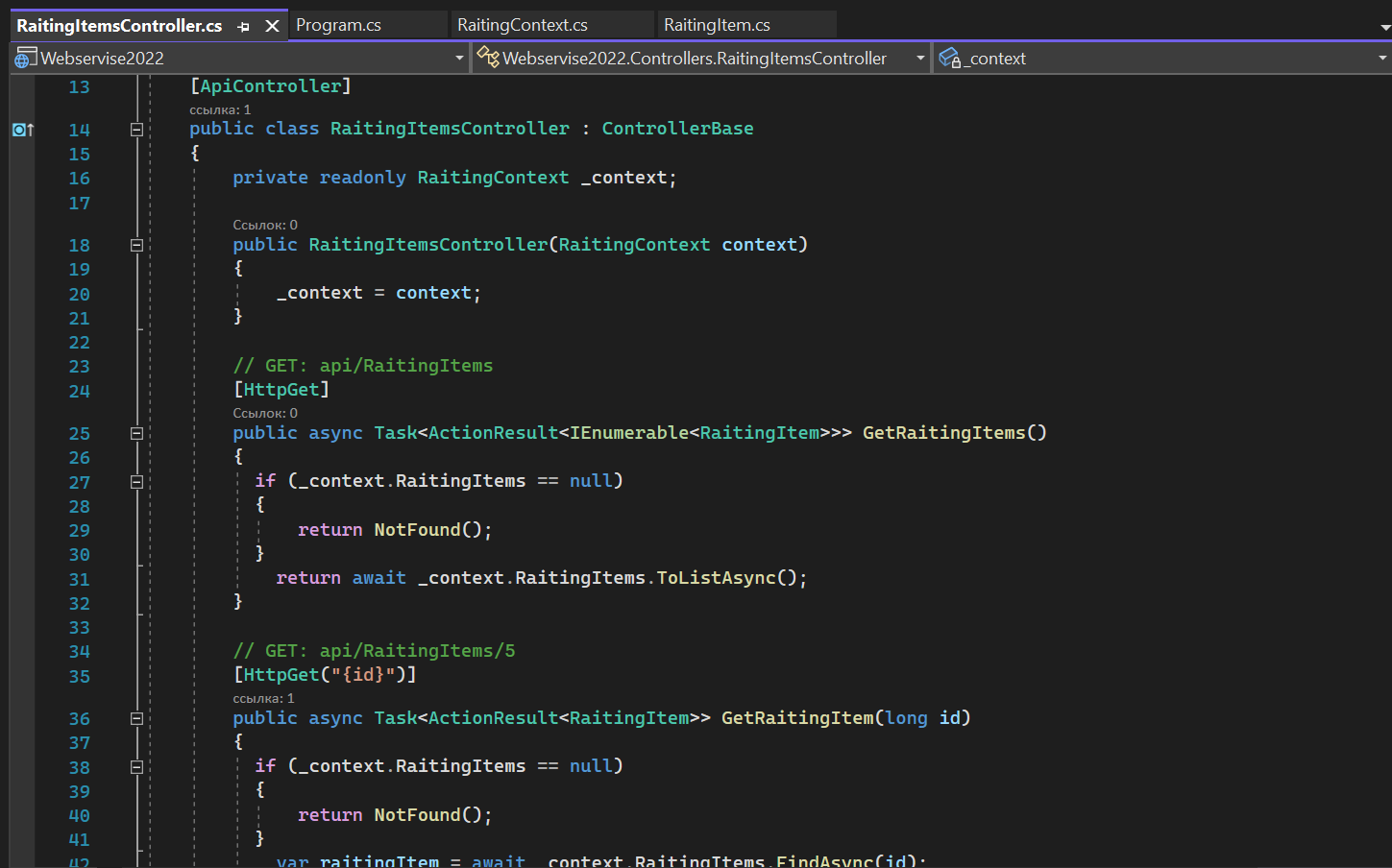
1. Создан контекст базы данных, для сохранения внесенных рейтингов



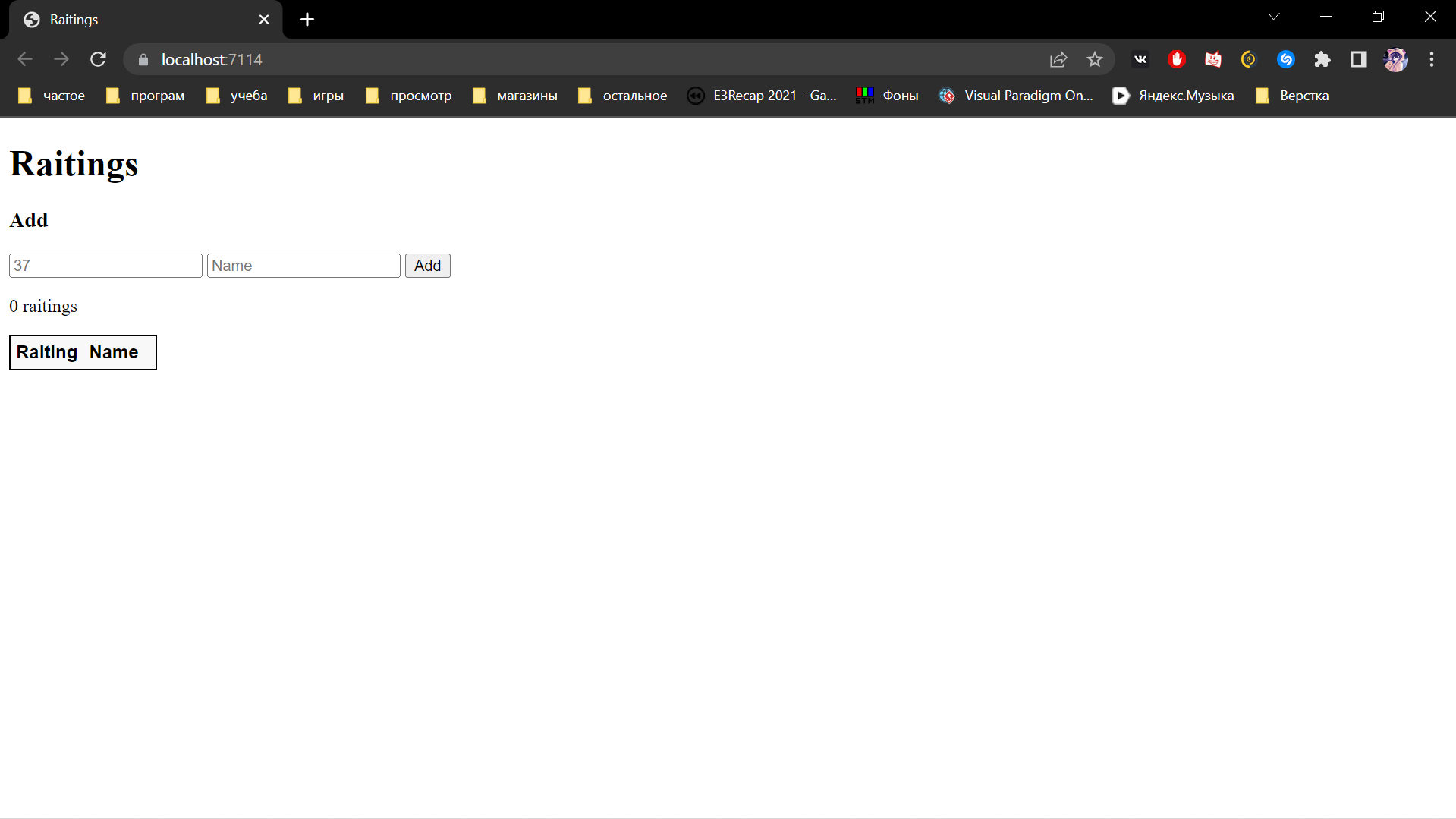
1. В ASP.NET Core службы (такие как контекст базы данных) должны быть зарегистрированы с помощью контейнера внедрения зависимостей. Контейнер предоставляет службу контроллерам.



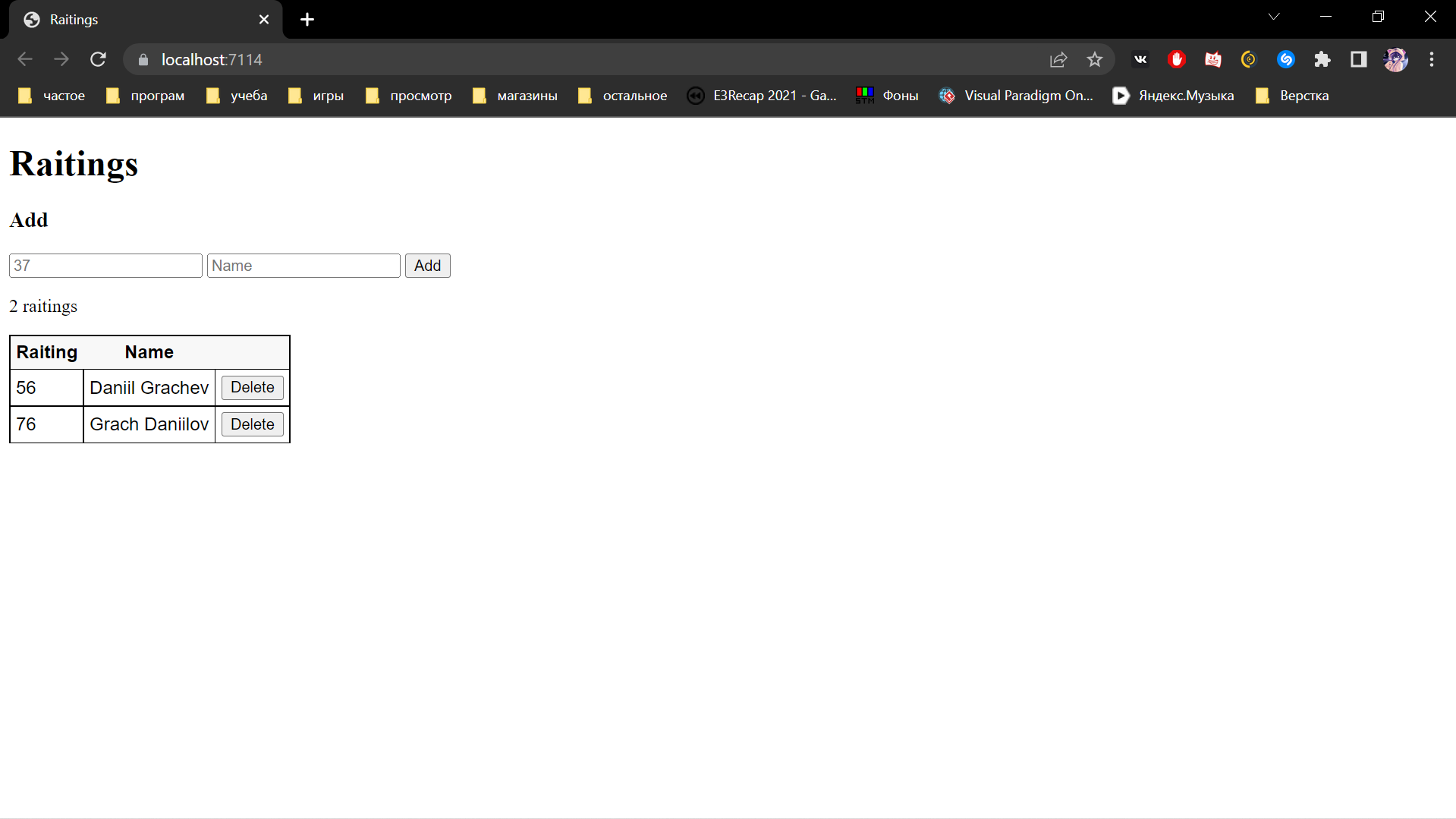
1. Контроллер действий с сервиса



1. Создадим внешний вид сайта для теста сервиса, добавив html и css



1. Тестирование работы программы



Листинг программы:

RaitingItemsController.cs

namespace Webservise2022.Controllers

{

    [Route("api/[controller]")]

    [ApiController]

    public class RaitingItemsController : ControllerBase

    {

        private readonly RaitingContext \_context;

        public RaitingItemsController(RaitingContext context)

        {

            \_context = context;

        }

*// GET: api/RaitingItems*

        [HttpGet]

        public async Task<ActionResult<IEnumerable<RaitingItem>>> GetRaitingItems()

        {

          if (\_context.RaitingItems == null)

          {

              return NotFound();

          }

            return await \_context.RaitingItems.ToListAsync();

        }

*// GET: api/RaitingItems/5*

        [HttpGet("{id}")]

        public async Task<ActionResult<RaitingItem>> GetRaitingItem(long id)

        {

          if (\_context.RaitingItems == null)

          {

              return NotFound();

          }

            var raitingItem = await \_context.RaitingItems.FindAsync(id);

            if (raitingItem == null)

            {

                return NotFound();

            }

            return raitingItem;

        }

*// PUT: api/RaitingItems/5*

*// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754*

        [HttpPut("{id}")]

        public async Task<IActionResult> PutRaitingItem(long id, RaitingItem raitingItem)

        {

            if (id != raitingItem.Id)

            {

                return BadRequest();

            }

            \_context.Entry(raitingItem).State = EntityState.Modified;

            try

            {

                await \_context.SaveChangesAsync();

            }

            catch (DbUpdateConcurrencyException)

            {

                if (!RaitingItemExists(id))

                {

                    return NotFound();

                }

                else

                {

                    throw;

                }

            }

            return NoContent();

        }

*// POST: api/RaitingItems*

*// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754*

        [HttpPost]

        public async Task<ActionResult<RaitingItem>> PostRaitingItem(RaitingItem raitingItem)

        {

            \_context.RaitingItems.Add(raitingItem);

            await \_context.SaveChangesAsync();

            return CreatedAtAction(nameof(GetRaitingItem), new { id = raitingItem.Id }, raitingItem);

        }

*// DELETE: api/RaitingItems/5*

        [HttpDelete("{id}")]

        public async Task<IActionResult> DeleteRaitingItem(long id)

        {

            if (\_context.RaitingItems == null)

            {

                return NotFound();

            }

            var raitingItem = await \_context.RaitingItems.FindAsync(id);

            if (raitingItem == null)

            {

                return NotFound();

            }

            \_context.RaitingItems.Remove(raitingItem);

            await \_context.SaveChangesAsync();

            return NoContent();

        }

        private bool RaitingItemExists(long id)

        {

            return (\_context.RaitingItems?.Any(e => e.Id == id)).GetValueOrDefault();

        }

    }

}

Site.js

const uri = 'api/raitingitems';

let raitings = [];

function getItems() {

    fetch(uri)

        .then(response => response.json())

        .then(data => \_displayItems(data))

        .catch(error => console.error('Unable to get items.', error));

}

function addItem() {

    const addNameTextbox = document.getElementById('add-name');

    const addRaitingTextbox = document.getElementById('add-raiting');

    const item = {

        grade: addRaitingTextbox.value.trim(),

        name: addNameTextbox.value.trim()

    };

    fetch(uri, {

        method: 'POST',

        headers: {

            'Accept': 'application/json',

            'Content-Type': 'application/json'

        },

        body: JSON.stringify(item)

    })

        .then(response => response.json())

        .then(() => {

            getItems();

            addNameTextbox.value = '';

            addRaitingTextbox.value = '';

        })

        .catch(error => console.error('Unable to add item.', error));

}

function deleteItem(id) {

    fetch(`${uri}/${id}`, {

        method: 'DELETE'

    })

        .then(() => getItems())

        .catch(error => console.error('Unable to delete item.', error));

}

function \_displayCount(itemCount) {

    const name = (itemCount === 1) ? 'raiting' : 'raitings';

    document.getElementById('counter').innerText = `${itemCount} ${name}`;

}

function \_displayItems(data) {

    const tBody = document.getElementById('raitings');

    tBody.innerHTML = '';

    \_displayCount(data.length);

    const button = document.createElement('button');

    data.forEach(item => {

        let deleteButton = button.cloneNode(false);

        deleteButton.innerText = 'Delete';

        deleteButton.setAttribute('onclick', `deleteItem(${item.id})`);

        let tr = tBody.insertRow();

        let td1 = tr.insertCell(0);

        let textNodeR = document.createTextNode(item.grade);

        td1.appendChild(textNodeR);

        let td2 = tr.insertCell(1);

        let textNodeN = document.createTextNode(item.name);

        td2.appendChild(textNodeN);

        let td4 = tr.insertCell(2);

        td4.appendChild(deleteButton);

    });

    raitings = data;

}

Index.js

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Raitings</title>

    <link rel="stylesheet" href="css/site.css" />

</head>

<body>

    <h1>Raitings</h1>

    <h3>Add</h3>

    <form action="javascript:void(0);" method="POST" onsubmit="addItem()">

        <input type="text" id="add-raiting" placeholder="37">

        <input type="text" id="add-name" placeholder="Name">

        <input type="submit" value="Add">

    </form>

    <p id="counter"></p>

    <table>

        <tr>

            <th>Raiting</th>

            <th>Name</th>

            <th></th>

        </tr>

        <tbody id="raitings"></tbody>

    </table>

    <script src="js/site.js" asp-append-version="true"></script>

    <script type="text/javascript">

        getItems();

    </script>

</body>

</html>

ВЫВОД

В ходе выполнения лабораторной работы познакомился с технологией создания и использования веб-сервисов.