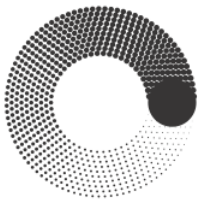
федеральное государственное автономное образовательное учреждение высшего образования



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 № 4

Дисциплина: BackEnd-разработка

Тема: Работа с кросс-доменными запросами веб-приложение на основе ASP.NET Core

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

Ежов Тимофей Алексеевич
(Фамилия И.О.)

Дата, подпись 11.03.2024

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

Дата, подпись _____ _____
(Дата) (Подпись)

Замечания: _____

_____

_____

Москва 2024

CORS (Cross-Origin Resource Sharing, «разделение ресурсов между разными источниками») — это механизм безопасности, который позволяет веб-страницам делать запросы к серверу, находящемуся на другом домене, отличном от домена страницы. В качестве основы для реализации кросс доменных запросов была взята программа из лабораторной работы № 10. Для начала работы с CORS нужно создать политику в файле Program.cs

```
builder.Services.AddCors(options =>
{
    // Политика для разрешения запросов с домена второго проекта
    options.AddPolicy("AllowProvDomain",
    policy => policy.WithOrigins("https://localhost:7072") //Определения источников с доступом к ресурсам приложения
    .WithMethods("GET") // Определения доступных HTTP методов
    .AllowAnyHeader()); // Управление заголовками
});
```

Указываем имя политики, далее настраиваем ее, сначала можно настроить какие источники будут иметь доступ к ресурсам и методам приложения, указываем локальный сервер дополнительного проекта, который будет использован для тестирования (будет разобран позже). Далее идет определения доступных методов HTTP, разрешаем посылать только GET запросы. Заголовки запросов разрешаем любые. Чтобы использовать эту политику прописываем app.UseCors(); В скобках можно указать название политики, какую конкретно использовать, в тех случаях если политик много, и нужно использовать в данный момент не все.

```
var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
    // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseCors();
app.MapControllers();

app.MapRazorPages();

app.Run();
```

После этого нужно указать в API контроллере к какими конкретно методами будет разрешено пользоваться домену, указанному в политике, для этого используется специальный атрибут, с названием политики.

```
// Метод для возвращения HTML-страницы
[HttpGet("html")] // Определяем маршрут для GET запросов
[EnableCors("AllowProvDomain")] // применяем нашу политику(разрешаем данный запрос)
Ссылок: 0
public IActionResult GetHtmlPage()
{
    // Создаем HTML страницу
    var htmlContent = @"
    <html>
        <body>
            <h1>Заголовок маленькой страницы</h1>
            <p>Возврат HTML контента</p>
            <ul>
                <li>Смысловая нагрузка</li>
                <li>Смысловая нагрузка x2</li>
            </ul>
        </body>
    </html>";
    return Content(htmlContent, "text/html"); // Возвращаем HTML-контент с MIME - типом text / html
}



[HttpGet("json")]
[EnableCors("AllowProvDomain")]// применяем нашу политику(разрешаем данный запрос)
Ссылок: 0
public IActionResult GetJsonData()
{
    // Создаем данные о пользователе для вывода
    var userData = new
    {
        Id = 7,
        Name = "Тима",
        Email = "timofeyezhov@gmail.com",
        Roles = new[] { "Бог-Император", "Омниссия", "Еретик" },
        Preferences = new
        {
            Language = "Высокий готик",
            Theme = "Имперская тема"
        }
    };
    return Ok(userData); // Возвращаем данные JSON со статусом 200
}
```

Разрешаем использовать только запросы для получения HTML данных и JSON, запросы на получение текстового файла и изображения не разрешаем, так как будем использовать запросы с веб-страницы, на которой не будет смысла получать файлы. Далее перейдем ко второму проекту, который был создан для тестирования корректной работы CORS политики. Для этого создаем новое пустое веб приложение. В нем создаем папку wwwroot в которой создадим файл CorsTest.html, так же включаем поддержку статических файлов, чтобы создать пользовательский интерфейс в файле HTML

```
app.UseStaticFiles();
```

Для проверки будем использовать 4 кнопки, по 1й на каждый запрос, в HTML файле определим эти кнопки.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CORS Test</title>
</head>
<body>
    <h2>CORS Test Page</h2>
    <button id="testHtml">Test HTML Response</button>
    <button id="testJson">Test JSON Response</button>
    <button id="testFile">Test File Response</button>
    <button id="testImage">Test Image Response</button>
    <script>
```

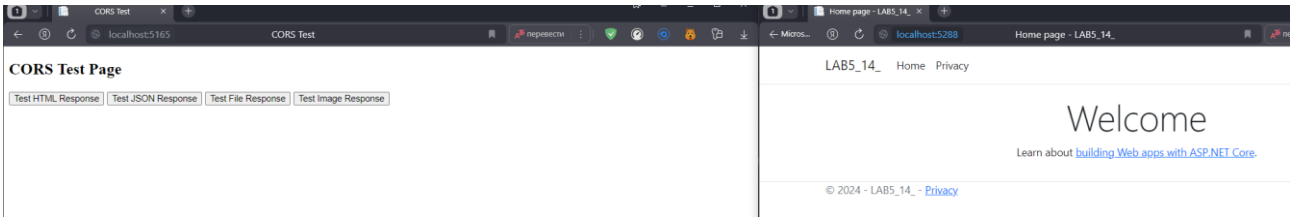
Далее добавим им обработчики событий.

```
    // Обработчики событий для кнопок вызывающие метод выполнения запроса по указанному адресу
    document.getElementById('testHtml').addEventListener('click', function () {
fetchAndLog('http://localhost:5288/responses/html');
    });
    document.getElementById('testJson').addEventListener('click', function () {
fetchAndLog('http://localhost:5288/responses/json');
    });
    document.getElementById('testFile').addEventListener('click', function () {
fetchAndLog('http://localhost:5288/responses/file');
    });
    document.getElementById('testImage').addEventListener('click', function () {
fetchAndLog('http://localhost:5288/responses/image');
    });
</script>
```

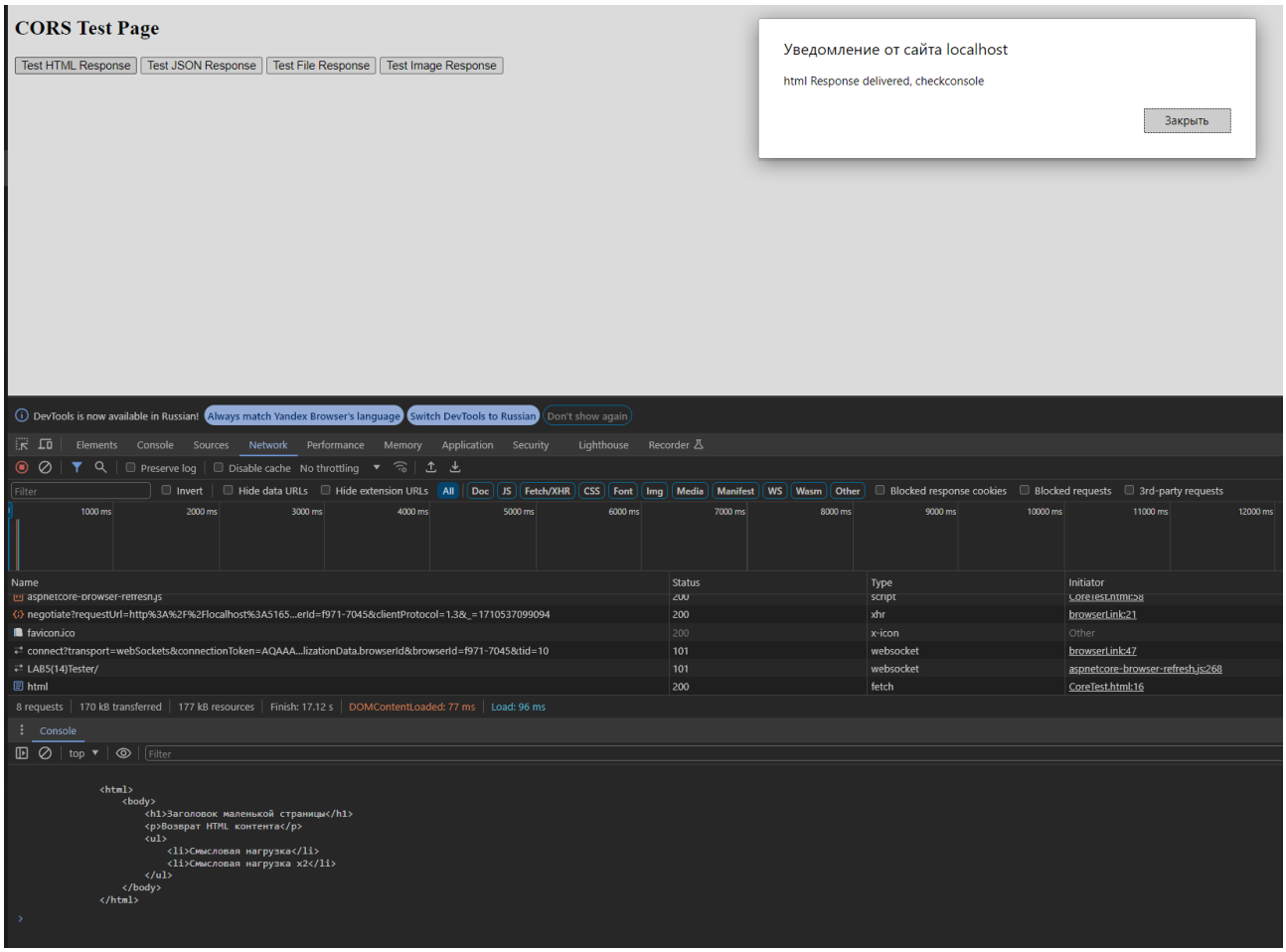
После чего реализуем функцию получения ответа от приложения и вывода его в консоль или обработку исключения если возникла ошибка или было отказано в доступе.

```
<script>
    // Функция для выполнения запроса по указанному URL и вывода результата в консоль
    function fetchAndLog(url) {
        fetch(url).then(response => {
                if (response.ok) {
                    // Ответ успешен -> обработка в зависимости от типа
                    if (url.endsWith('json')) {
                        return response.json(); // JSON ответ
                    }
                    else if (url.endsWith('file') || url.endsWith('image')) {
                        return response.blob(); // Файловый ответ
                    }
                    else {
                        return response.text(); // текстовый ответ
                    }
                }
                throw new Error('Network error'); // случай ошибки с соединением если не удается получить ответ
            })
            .then(data => {
                console.log(data); // вывод в консоль
                alert(url.split('/').pop() + ' Response delivered, checkconsole'); // уведемление об успешном запросе
            })
            .catch(error => {
                console.error('CORS error:', error); // ошибка возникающая при запрете доступа
                alert('CORS error, check console'); // ошибка в консоль и на экран
            });
    }
```

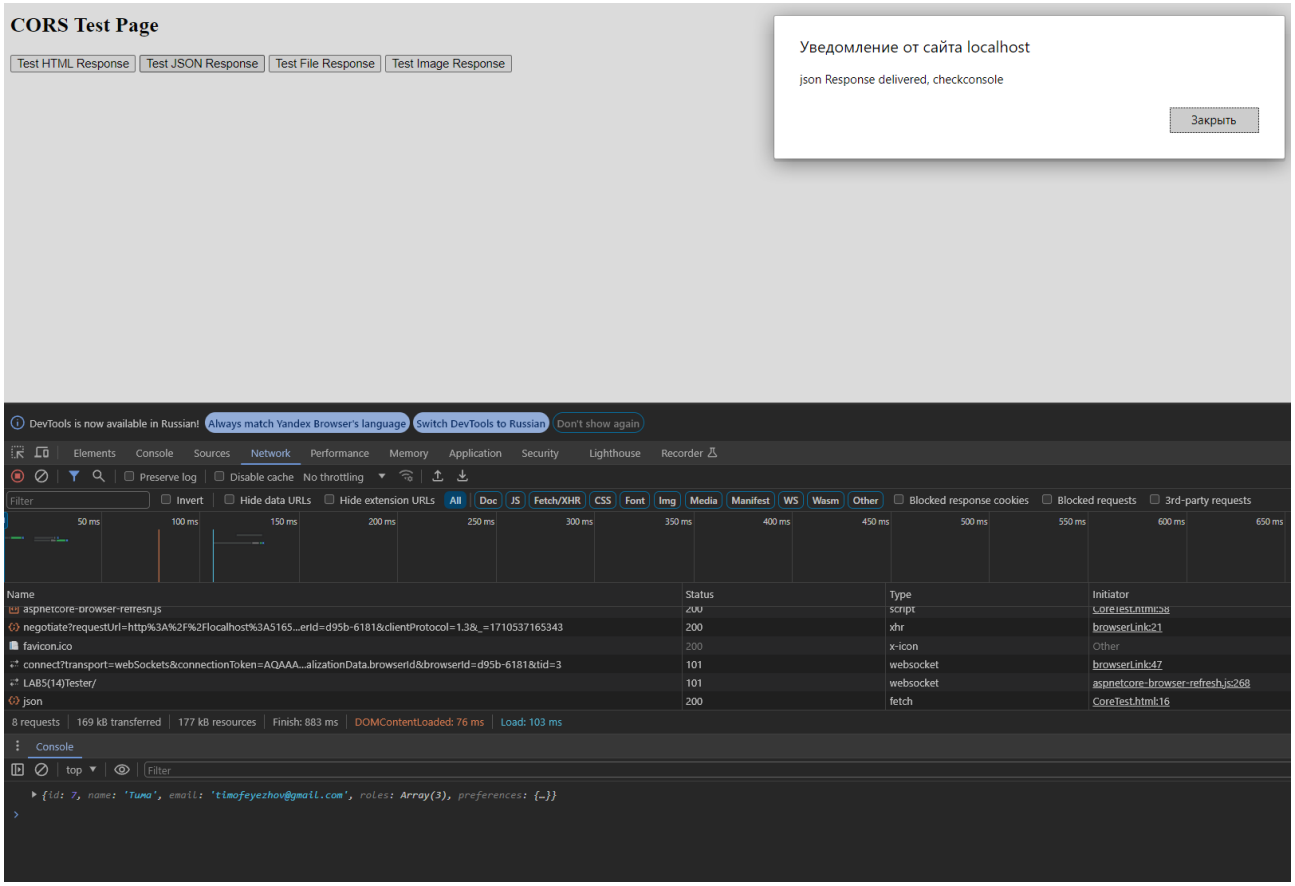
После реализации и настройки CORS политики, а так же реализации второго проекта для проверки функциональности, пришло время тестирования. Запускаем оба проекта.



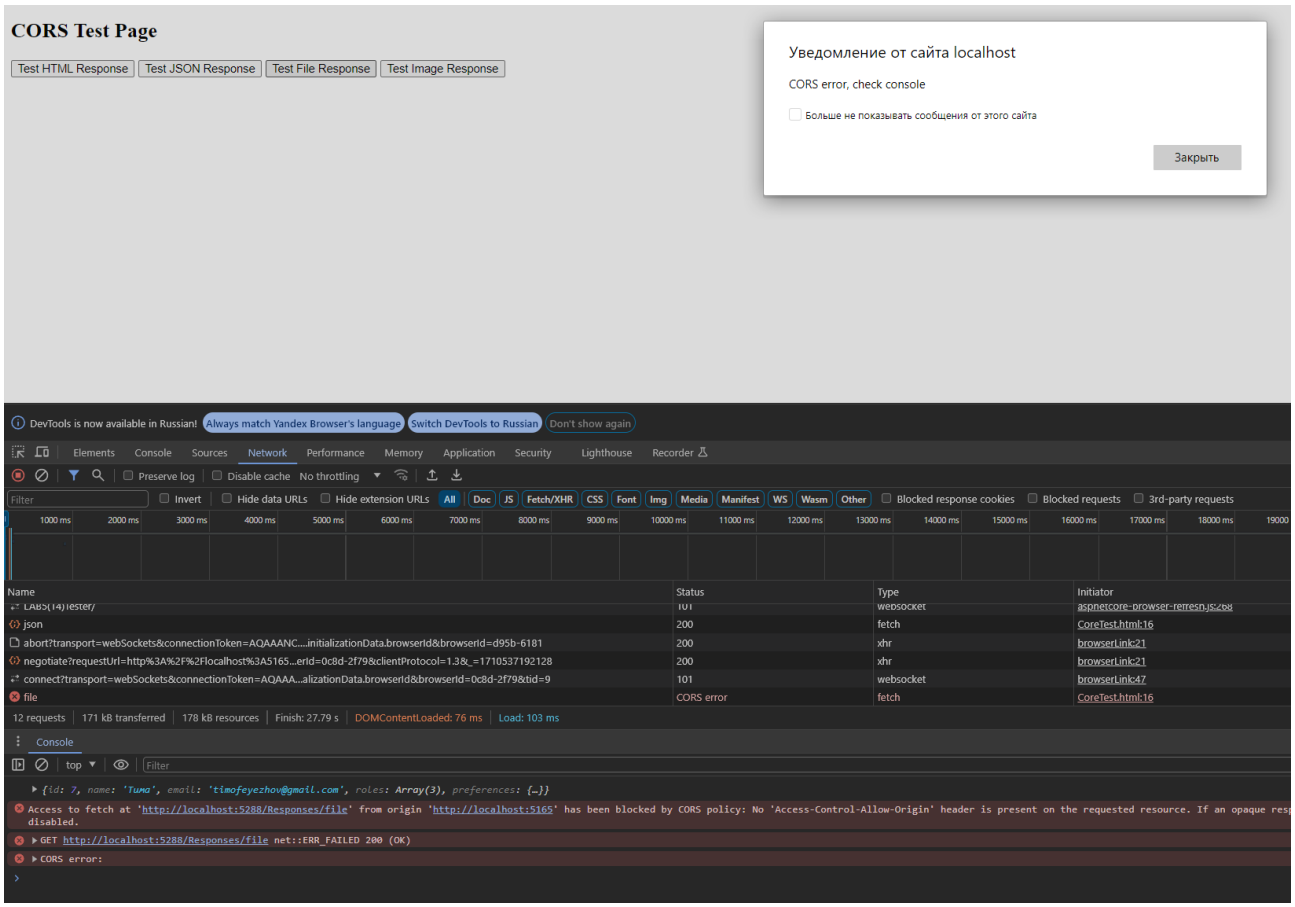
Страница проекта тестирования успешно отображает интерфейс, попробуем отправить запрос, открыв клавишей f12 консоль



JSON так же работает.



В это же время попытки отправки запроса на получение файла с текстом или изображения приводят к ошибке, так как политика CORS не разрешает использование этих запросов.



Практические выводы из проделанной работы: кросс доменные запросы, являются полезным и безопасным (так как разработчики имеют полный контроль над политиками предоставления доступа) инструментом для предоставления данных внешним ресурсам, CORS позволяет гибко настраивать кому предоставлять доступ и к каким данным, что так же является удобным инструментом для интеграции с другими веб сервисами и API приложениями.

Код основного проекта:

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddRazorPages();
builder.Services.AddControllers();

builder.Services.AddCors(options =>
{
    // ********** ??? ************ ********** ? ??????? ???????? ????????
    options.AddPolicy("AllowProvDomain",
    policy => policy.WithOrigins("http://localhost:5165") //??????????? ??????????? ? ********* ? ********* ???????????
    .WithMethods("GET") // ??????????? ?????????? HTTP ????????
    .AllowAnyHeader()); // ??????????? ???????????
});


var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
    // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}
app.MapControllerRoute(name: "default", pattern: "{controller=Home}/{action=Index}");
app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseCors();
```

```
app.MapControllers();

app.MapRazorPages();

app.Run();
```

```
using Microsoft.AspNetCore.Cors;
using Microsoft.AspNetCore.Mvc;

namespace LAB5_14_.Responses
{
    [ApiController]
    [Route("[controller]")]
    public class ResponsesController : Controller
    {
        // Метод для возвращения HTML-страницы
        [HttpGet("html")] // Определяем маршрут для GET запросов
        [EnableCors("AllowProvDomain")] // применяем нашу политику(разрешаем данный
запрос)
        public IActionResult GetHtmlPage()
        {
            // Создаем HTML страницу
            var htmlContent = @"
            <html>
                <body>
                    <h1>Заголовок маленькой страницы</h1>
                    <p>Возврат HTML контента</p>
                    <ul>
                        <li>Смысловая нагрузка</li>
                        <li>Смысловая нагрузка x2</li>
                    </ul>
                </body>
            </html>";
            return Content(htmlContent, "text/html"); // Возвращаем HTML-контент с
MIME - типом text / html
        }



        [HttpGet("json")]
        [EnableCors("AllowProvDomain")]// применяем нашу политику(разрешаем данный
запрос)
        public IActionResult GetJsonData()
        {
            // Создаем данные о пользователе для вывода
            var userData = new
            {
                Id = 7,
                Name = "Тима",
```

```
                Email = "timofeyezhov@gmail.com",
                Roles = new[] { "Бог-Император", "Омниссия", "Еретик" },
                Preferences = new
                {
                    Language = "Высокий готик",
                    Theme = "Имперская тема"
                }
            };
            return Ok(userData); // Возвращаем данные JSON со статусом 200
        }



        [HttpGet("file")]
        public IActionResult GetFile()
        {
            var filePath = @"Response.txt"; //Путь к файлу
            var bytes = System.IO.File.ReadAllBytes(filePath); // Считываем файл
вмассив байтов
            return File(bytes, "text/plain", Path.GetFileName(filePath));
//Возвращаемфайл с MIME - типом text / plain
        }



        // Метод для возвращения изображения
        [HttpGet("image")]
        public IActionResult GetImage()
        {
            var filePath = @"HEHE.jpeg"; // Путь к изображению
            var bytes = System.IO.File.ReadAllBytes(filePath); // Считываем
изображение в массив байтов
            return File(bytes, "image/jpeg"); // Возвращаем изображение с MIME-
типом  image / jpg
        }

    }
}
```

Код Тестировщика:

```
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddEndpointsApiExplorer();
var app = builder.Build();
if (!app.Environment.IsDevelopment())
{
    app.UseHttpsRedirection();
}
app.UseStaticFiles(); // ►►►►►►►►►► ►►►►►►►►►► ►►►►►►►►►►►►
►►►►►►
```

```
app.MapGet("/", () => "Hello World!");
app.Run();
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CORS Test</title>
</head>
<body>
    <h2>CORS Test Page</h2>
    <button id="testHtml">Test HTML Response</button>
    <button id="testJson">Test JSON Response</button>
    <button id="testFile">Test File Response</button>
    <button id="testImage">Test Image Response</button>
    <script>
         // Функция для выполнения запроса по указанному URL и вывода результата в
консоль
        function fetchAndLog(url) {
            fetch(url)
                .then(response => {
                    if (response.ok) {
                        // Ответ успешен -> обработка в зависимости от типа
                        if (url.endsWith('json')) {
                            return response.json(); // JSON ответ
                        }
                        else if (url.endsWith('file') || url.endsWith('image')) {
                            return response.blob(); // Файловый ответ
                        }
                        else {
                            return response.text(); // текстовый ответ
                        }
                    }
                    throw new Error('Network error'); // случай ошибки с
соединением если не удается получить ответ
                })
                .then(data => {
                    console.log(data); // вывод в консоль
                    alert(url.split('/').pop() + ' Response delivered,
checkconsole'); // уведемление об успешном запросе
                })
                .catch(error => {
                    console.error('CORS error:', error); // ошибка возникающая при
запрете доступа
                    alert('CORS error, check console'); // ошибка в консоль и на
экран
                });
        }
```

```
        // Обработчики событий для кнопок вызывающие метод выполнения запроса по
указанному адресу
        document.getElementById('testHtml').addEventListener('click', function () {
    fetchAndLog('http://localhost:5288/Responses/html');
        });
        document.getElementById('testJson').addEventListener('click', function () {
    fetchAndLog('http://localhost:5288/Responses/json');
        });
        document.getElementById('testFile').addEventListener('click', function () {
    fetchAndLog('http://localhost:5288/Responses/file');
        });
        document.getElementById('testImage').addEventListener('click', function ()
{
    fetchAndLog('http://localhost:5288/Responses/image');
        });
    </script>
</body>
</html>
```