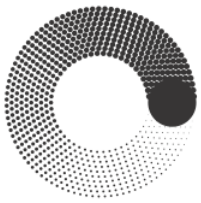
федеральное государственное автономное образовательное учреждение высшего образования



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: Изучение выдачи ответов в веб-приложении 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

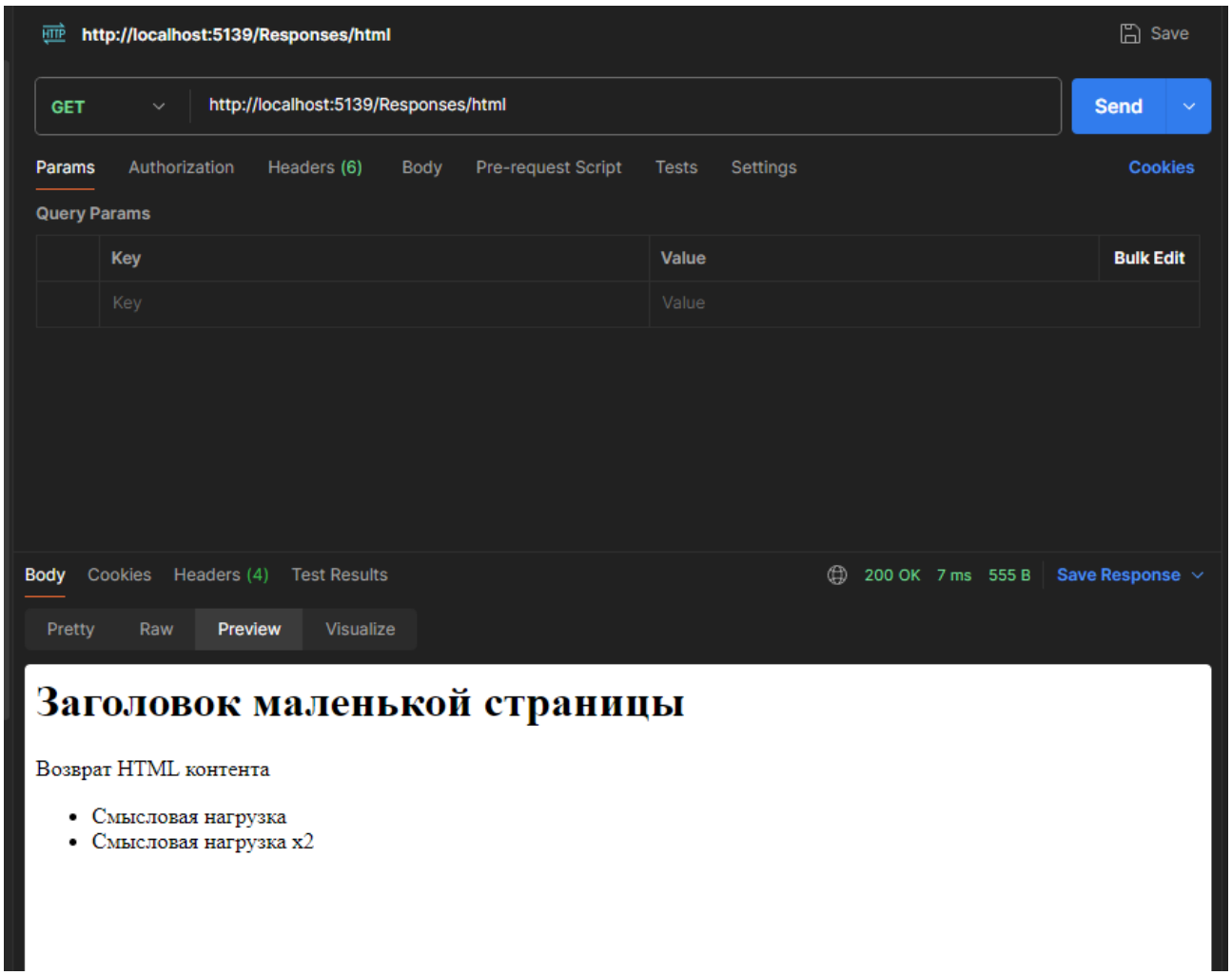
В ходе данной лабораторной работы были изучены способы возвращения ответов ввиду HTML контента для возвращения HTML страницы, JSON данных, а также файлов (в том числе изображений). Рассмотрим подробнее каждый из типов ответов.

HTML страница: Ответы в формате HTML чаще всего используются для отображения страниц пользовательского интерфейса. Метод Content создаёт объект ContentResult, который содержит строку htmlContent в качестве тела ответа. Второй аргумент "text/html" указывает MIME-тип ответа, информируя клиента (например, веб-браузер), что возвращаемый контент является HTML.

```
[ApiController]
[Route("[controller]")]
Ссылок: 0
public class ResponsesController : Controller
{

    // Метод для возвращения HTML-страницы
    [HttpGet("html")] // Определяем маршрут для GET запросов
    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
    }
```

Метод Content создаёт объект ContentResult, который содержит строку htmlContent в качестве тела ответа. Второй аргумент text/html указывает MIMEтип ответа, информируя клиента, что возвращаемый контент является HTML. Теперь перейдем в Postman и отправим GET запрос для проверки.



Как видим, текст HTML страницы выводится корректно. Теперь разберем преимущества, ограничения и лучшие варианты использования. Преимущества:

• Универсальность и широкая поддержка браузерами.

• Возможность создания красивого интерфейса с использованием CSS и JavaScript.

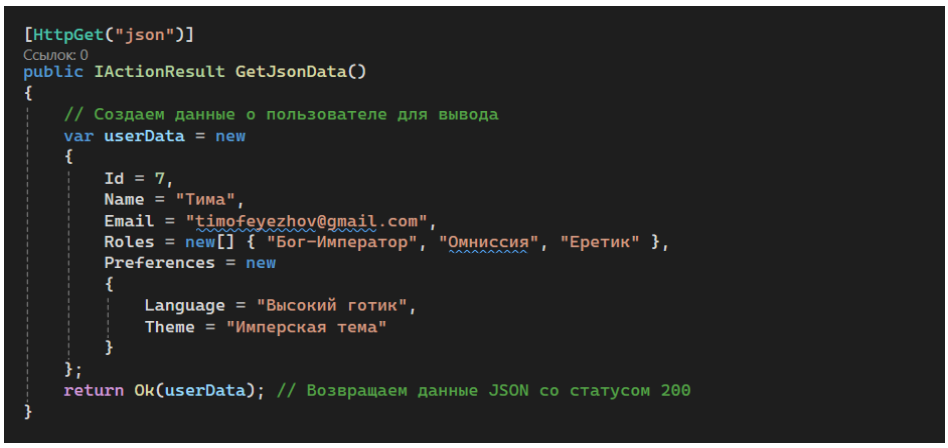
Ограничения:

• Плохо подходит для обмена данными с другими приложениями при использовании API.

Лучшие ситуации для применения:

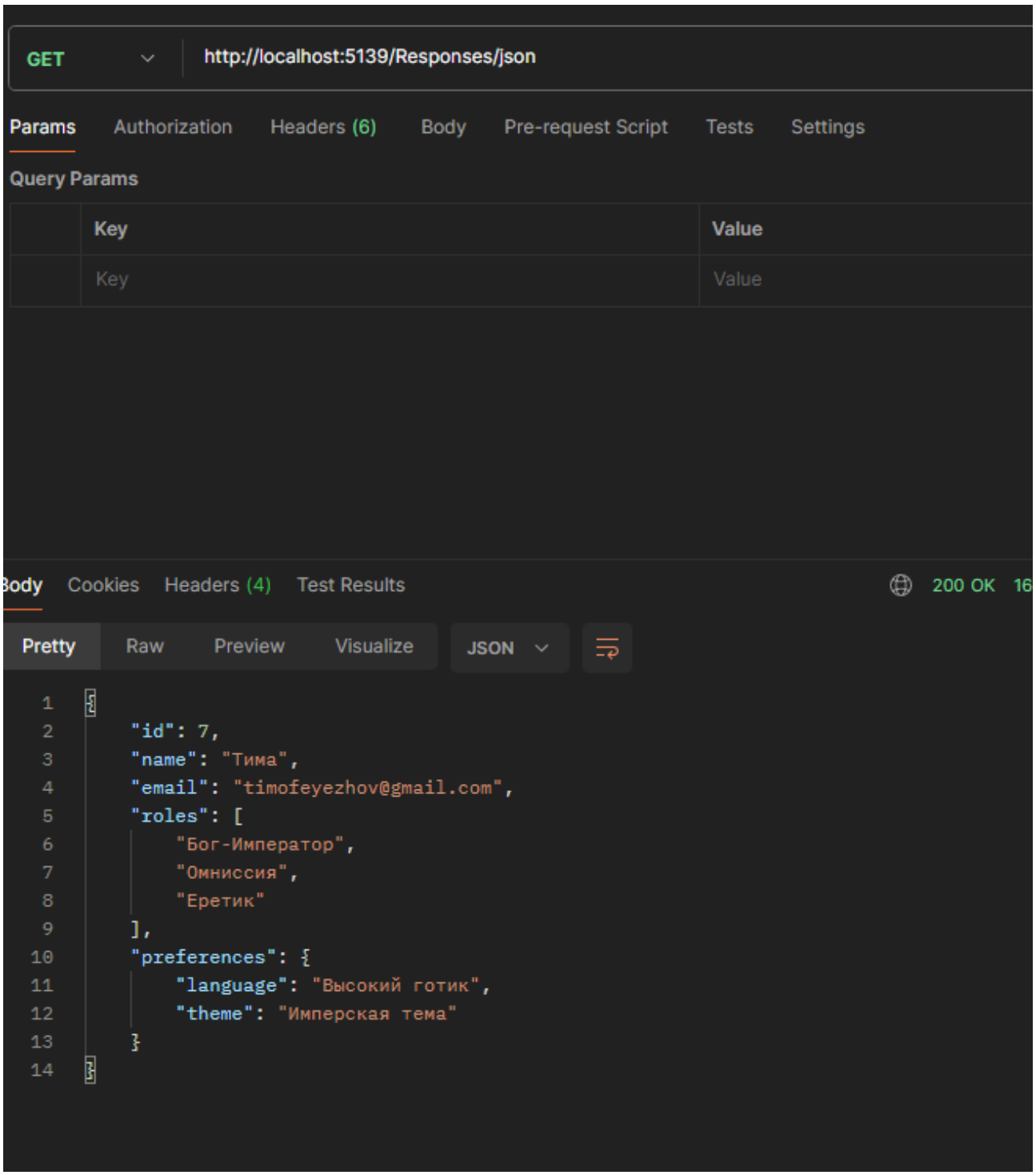
• Использование для отображения пользовательских интерфейсов веб-приложений.

JSON данные: JSON часто используется для обмена данными между сервером и клиентами, а также для более легкого структурирования данных.



```
[HttpGet("json")]
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
}
```

Используем JSON формат для вывода данных о пользователе, создадим объект userData и выводим его со статусом 200 (ok). Так же проверим вывод через Postman.



Преимущества:

• Лёгкость чтения и записи как для людей, так и для программ.

• Поддерживается большинством современных технологий и языков программирования.

• Эффективен для передачи структурированных данных.

Ограничения:

• Не подходит для передачи бинарных данных, таких как изображения.

Лучшие ситуации для применения:

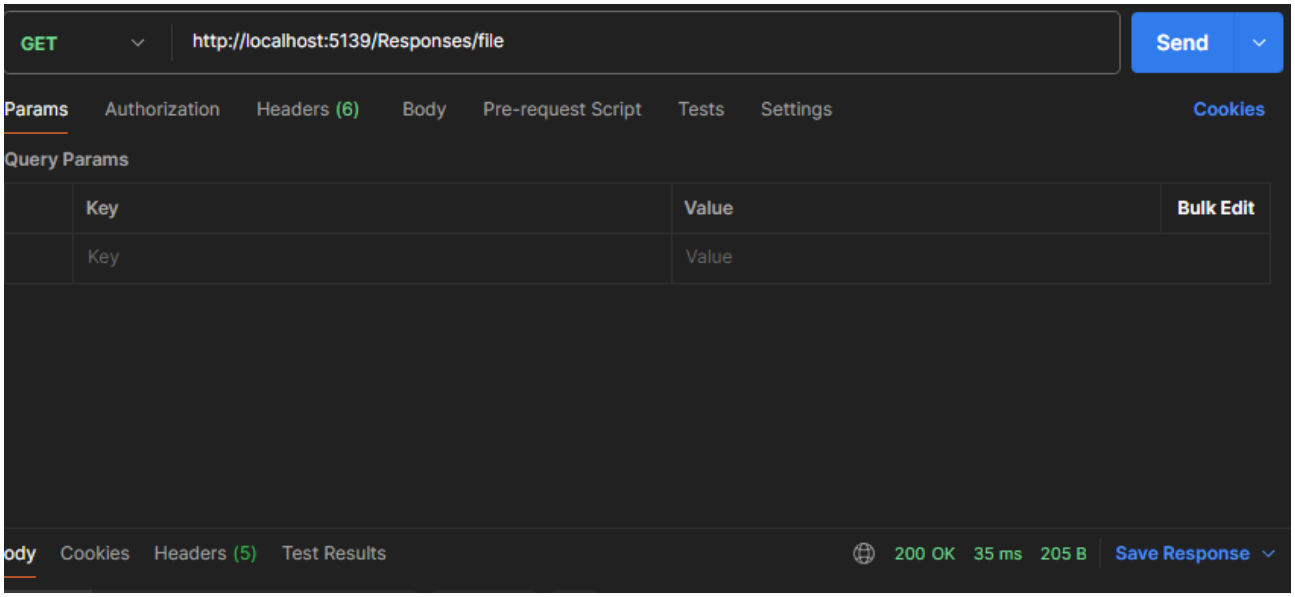
• Передача структурированных данных.

• Основной формат обмена данных в API.

Вывод файлов: рассмотрим возвращение двух типов файлов, текстового и изображения.

```
[HttpGet("file")]
Ссылок: 0
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
}
```

Для вывода текстового файла создадим метод GetFile(), метод вывода File отправляет массив байтов в который мы преобразовали текстовый файл, затем указываем тип файла и выводим содержащийся текст. Так же передается и путь к файлу для возможности загрузки.



Код 200 - всё окей

Преимущества:

• Текстовые документы могут быть прочитаны на любом устройстве и не требуют специальных программ.

• Небольшой размер файла позволяет быстро передавать его.

• Текстовые файлы могут быть легко обработаны программой для дальнейшего использования.

Ограничения:

• Проблемы с форматированием для некоторых типов документов, требующих специальное оформление, например, таблицы или изображения внутри текстового документа.

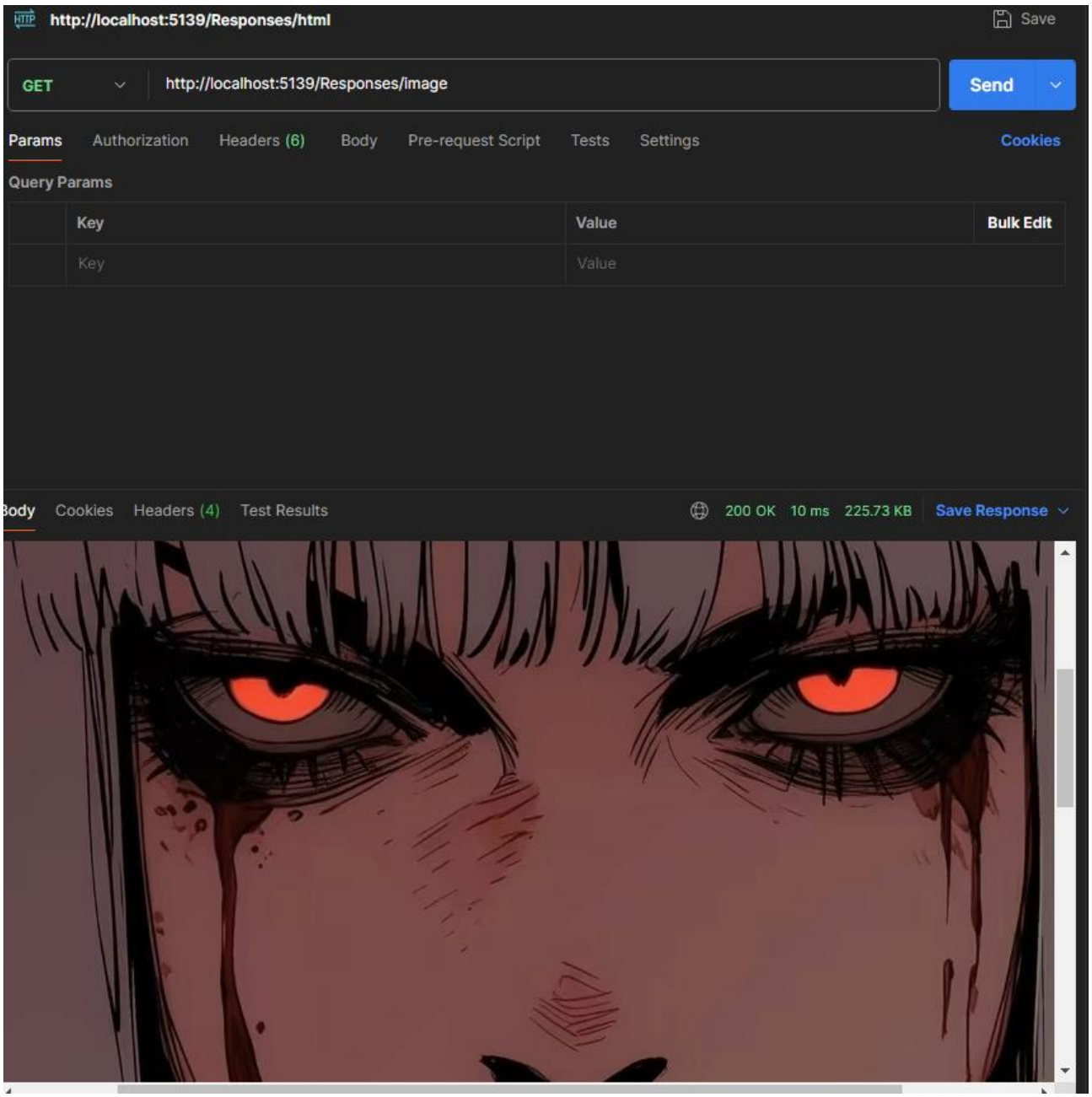
Лучшие ситуации для применения:

- Предоставление логов и отчетов.
- Передача конфигурационных файлов.

Далее рассмотрим вывод изображения в качестве ответа, код работает схожим образом.

```
// Метод для возвращения изображения
[HttpGet("image")]
Ссылок: 0
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
}
```

Здесь метод File, возвращает изображение в формате jpeg, такой тип ответа полезен для контролируемого доступа к медиа-файлам. Отправим запрос в Postman.



Изображение отображается корректно – всё работает

Преимущества:

• Возможность предоставление доступа к скачиванию изображений или для динамических изображений, когда требуется контролировать доступ к ним.

Ограничения:

• Увеличивает нагрузку на сервер при большом объеме файла.

• Изображение должно подходить под форматы всех устройств, что усложняет логику вывода и увеличивает объём хранимых данных.

Лучшие ситуации для применения:

• Вывод изображения как часть контента веб страницы.

• Контроль доступа к изображениям.

Выводы: Сравнивая данные типы ответов можно прийти к следующим выводам, лучшим форматом пользовательских интерфейсов является HTML, для обмена данными JSON, а для функциональности скачивания или контроля доступа к медиа-контенту является возврат файлов.

Код контроллера:

```
using Microsoft.AspNetCore.Mvc;
using static System.Net.Mime.MediaTypeNames;

namespace LAB4_10_.Responses
{
    [ApiController]
    [Route("[controller]")]
    public class ResponsesController : Controller
    {

        // Метод для возвращения HTML-страницы
        [HttpGet("html")] // Определяем маршрут для GET запросов
        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")]
        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
        }
    }
}
```