ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № _17_

Дисциплина: _Backend разработка

Тема:

_____

Выполнил(а): студент(ка) группы __231-336__

_____Канищев И.М_____
(Фамилия И.О.)

Дата, подпись _____ _____
(Дата) (Подпись)

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

Дата, подпись _____ _____
(Дата) (Подпись)

Замечания:

Москва

2025

1. Типы кеширования в ASP.NET Core

1.1 Внутренний кэш (In-Memory Cache)

- Хранит данные в памяти приложения.
- Подходит для небольших данных, быстро доступен.
- Используется через IMemoryCache.

Пример использования:

```
public class WeatherService
{
    private readonly IMemoryCache _cache;

    public WeatherService(IMemoryCache cache)
    {
        _cache = cache;
    }

    public string GetWeather()
    {
        return _cache.GetOrCreate("weather", entry =>
        {
            entry.AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(30);
            return "Sunny"; // Эмуляция данных
        });
    }
}
```

1.2 Кэш диска (Distributed File Cache)

- Хранит данные на диске.
- Подходит для больших данных и сценариев, когда нужно сохранять кэш между перезапусками приложения.
- В ASP.NET Core реализуется через IDistributedCache с файловым провайдером.

Пример использования:

```
public class FileCacheService
{
    private readonly string _cacheFolder = "CacheFiles";
    private readonly string _cacheFile;

    public FileCacheService()
    {
        if (!Directory.Exists(_cacheFolder))
            Directory.CreateDirectory(_cacheFolder);

        _cacheFile = Path.Combine(_cacheFolder, "weather.txt");
    }

    public async Task<string> GetWeatherAsync()
    {
        if (File.Exists(_cacheFile))
        {
            var info = new FileInfo(_cacheFile);
            if (DateTime.Now - info.LastWriteTime < TimeSpan.FromMinutes(1))
```

```
                return await File.ReadAllTextAsync(_cacheFile);
        }

        var weather = "Cloudy"; // Эмуляция данных
        await File.WriteAllTextAsync(_cacheFile, weather);
        return weather;
    }
}
```

1.3 Response Caching

- Позволяет кешировать HTTP-ответы для уменьшения нагрузки на сервер.
- Настраивается через middleware и атрибут [ResponseCache].

Пример контроллера:

```
[ResponseCache(Duration = 60)]
[HttpGet("/weather/response")]
public IActionResult GetResponseCache()
{
    return Ok("Response cached at " + DateTime.Now.ToLongTimeString());
}
```

1.4 Контроллер для работы с кешем

Пример использования:

```
[ApiController]
[Route("[controller]")]
public class WeatherController : ControllerBase
{
    private readonly WeatherService _memoryCache;
    private readonly FileCacheService _fileCache;

    public WeatherController(WeatherService memoryCache, FileCacheService
fileCache)
    {
        _memoryCache = memoryCache;
        _fileCache = fileCache;
    }

    [HttpGet("memory")]
    public IActionResult GetMemoryCache()
    {
        return Ok(_memoryCache.GetWeather());
    }

    [HttpGet("file")]
    public async Task<IActionResult> GetFileCache()
    {
        return Ok(await _fileCache.GetWeatherAsync());
    }

    [ResponseCache(Duration = 60)]
    [HttpGet("response")]
    public IActionResult GetResponseCache()
    {
        return Ok("Response cached at " + DateTime.Now.ToLongTimeString());
    }
}
```

2. Тестирование кеширования

2.1 In-Memory Cache

- Первый запрос создаёт данные.
- Повторные запросы в течение 30 секунд возвращают кэшированные данные.
- После 30 секунд кэш истекает — данные пересоздаются.

2.2 File-based Cache

- Первый запрос создаёт файл `CacheFiles/weather.txt`.
- Повторные запросы читают данные из файла, если срок жизни не истёк (1 минута).
- После 1 минуты файл перезаписывается новым значением.

2.3 Response Caching

- Первый запрос создаёт HTTP-ответ и кеширует его.
- Повторные запросы в течение 60 секунд возвращают **тот же ответ**.
- После 60 секунд кэш истекает — создаётся новый ответ.

2.4 Методы оценки производительности

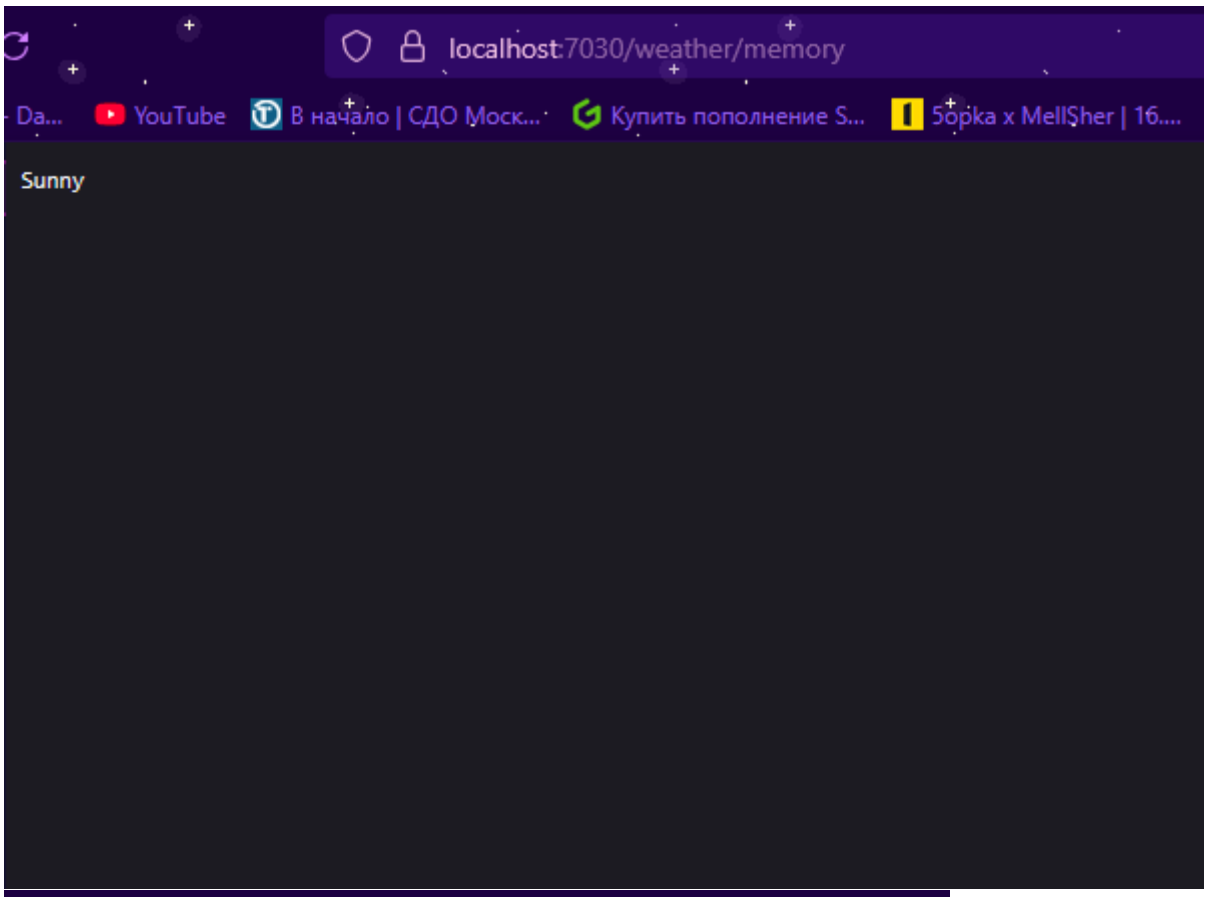
- Первый запрос (кэш пуст) — больше времени.

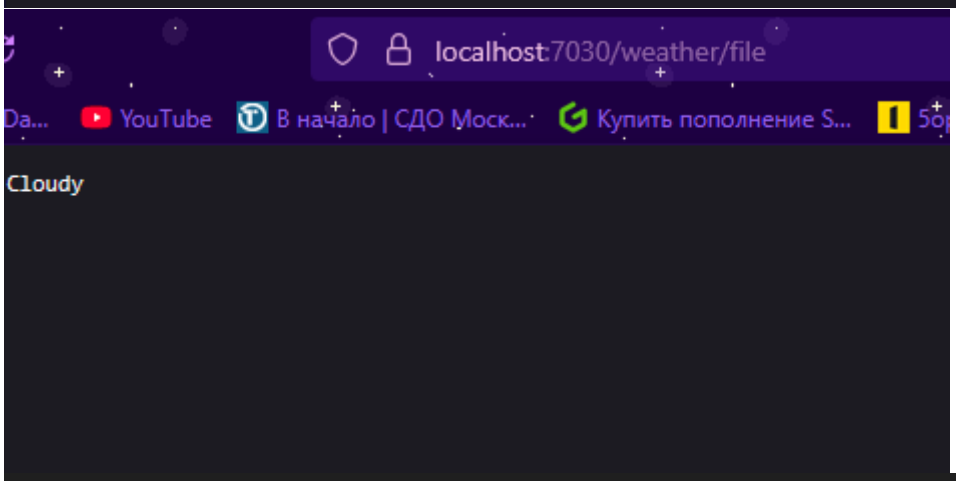
- Последующие запросы (кэш заполнен) — значительно меньше времени.

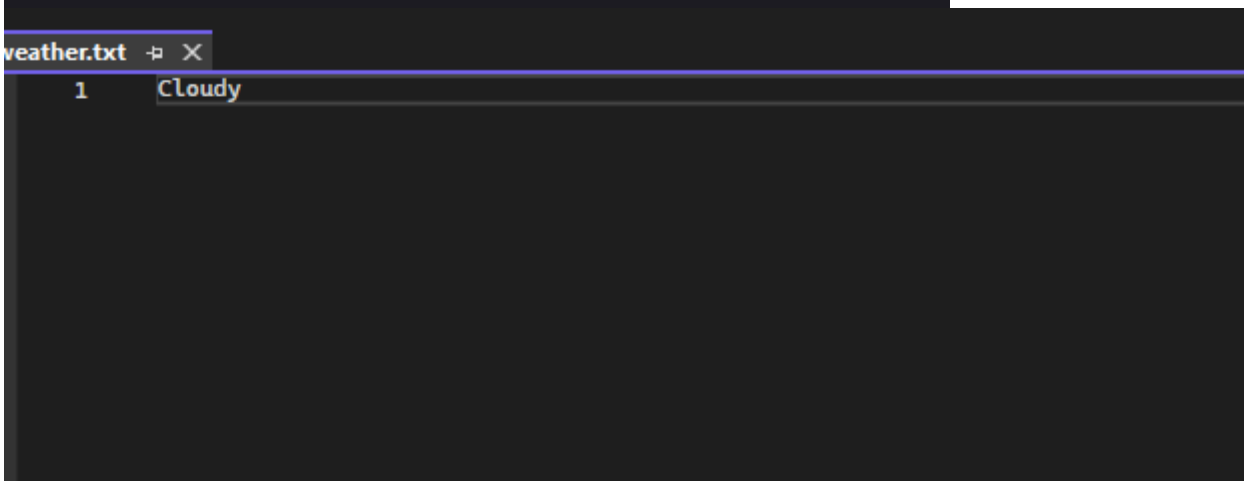
Пример измерения времени ответа:

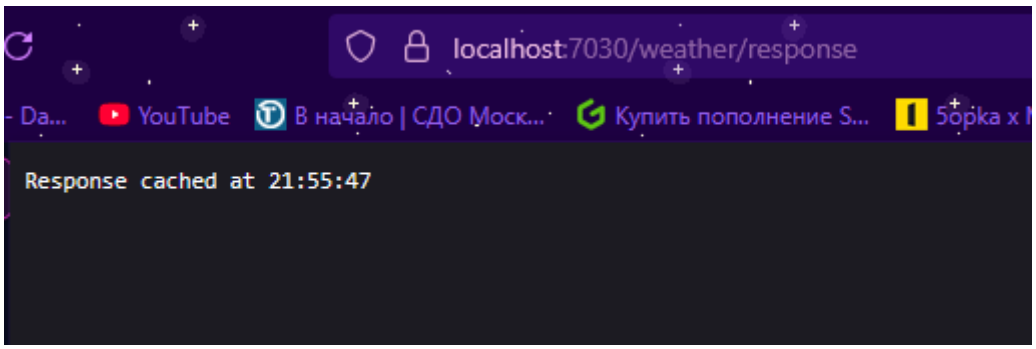
```
var stopwatch = Stopwatch.StartNew();
var data = await cacheService.GetWeatherAsync();
stopwatch.Stop();
Console.WriteLine($"Время получения данных: {stopwatch.ElapsedMilliseconds}
ms");
```

Скрины работы приложения


localhost:7030/weather/memory
Da...
YouTube
В начало | СДО Моск...
Купить пополнение S...
5opka x MellSher | 16....
Sunny


localhost:7030/weather/file
Da...
YouTube
В начало | СДО Моск...
Купить пополнение S...
Cloudy


weather.txt
1
Cloudy


localhost:7030/weather/response
- Da...
YouTube
В начало | СДО Моск...
Купить пополнение S...
5opka x
Response cached at 21:55:47