Задание 1

Петухов Глеб
11-106

1. Создал класс краулера с полями для названия папки для сохранения файлов, названия файла для ссылок страниц, минимального количества слов и страниц.

```
IL code
class WebCrawler
{
    IL code
    private readonly List<string> _startUrls;
    IL code
    private readonly int _minPages;
    IL code
    private readonly int _minWords;
    IL code
    private readonly HashSet<string> _visitedUrls = new ();
    IL code
    private int _pagesDownloaded = 0;
    private const string IndexFile = "index.txt";
    private const string OutputDir = "downloaded_pages";

    IL code
    public WebCrawler(List<string> startUrls, int minPages = 100, int minWords = 1000)
    {
        _startUrls = startUrls;
        _minPages = minPages;
        _minWords = minWords;

        Directory.CreateDirectory(OutputDir);

        File.WriteAllText(IndexFile, string.Empty);
    }
```

2. Добавил к классу метод проверки валидности URL и проверку что это не ссылка на файл.

```
IL code
private bool IsValidUrl(string url)
{
    // Проверяем, что URL валиден и не является файлом (например, .pdf)
    if (!Uri.TryCreate(url, UriKind.Absolute, out var uri))
        return false;

    if (uri.Scheme != Uri.UriSchemeHttp && uri.Scheme != Uri.UriSchemeHttps)
        return false;

    var invalidExtensions = new[] { ".pdf", ".jpg", ".png", ".gif", ".zip", ".doc", ".docx" };
    return !invalidExtensions.Any(ext => uri.AbsolutePath.EndsWith(ext, StringComparison.OrdinalIgnoreCase));
}
```

3. Добавил метод для получения ссылок из текущей страницы

```
IL code  More...
private async Task<HashSet<string>> GetLinksAsync(string url, HttpClient client)
{
    try
    {
        var response = await client.GetAsync(url);
        response.EnsureSuccessStatusCode();

        var html = await response.Content.ReadAsStringAsync();
        var doc = new HtmlDocument();
        doc.LoadHtml(html);

        var links = new HashSet<string>();

        foreach (var link in doc.DocumentNode.SelectNodes("//a[@href]") ?? Enumerable.Empty<HtmlNode>())
        {
            var href = link.GetAttributeValue("href", string.Empty);
            var absoluteUrl = new Uri(new Uri(url), href).AbsoluteUri;

            if (IsValidUrl(absoluteUrl))
                links.Add(absoluteUrl);
        }

        return links;
    }
    catch (Exception e)
    {
        Console.WriteLine($"Ошибка при получении ссылок с {url}: {e.Message}");
        return new HashSet<string>();
    }
}
```

4. Метод для парсинга текста из html страницы

```
IL code
private string ExtractText(string html)
{
    var doc = new HtmlDocument();
    doc.LoadHtml(html);

    var nodesToRemove = doc.DocumentNode.SelectNodes("//script|//style|//nav|//footer|//head|//iframe");

    foreach (var node in nodesToRemove)
        node.Remove();

    var text = doc.DocumentNode.InnerText;

    // Удаляем лишние пробелы и переносы строк
    text = Regex.Replace(text, @"\s+", " ").Trim();

    return text;
}
```

5. Метод для сохранения страницы в файл. Здесь я проверяю, что количество слов соответствует заданному параметру

```
IL code
private bool SavePage(string url, string text)
{
    var words = text.Split(new[] { ' ', '\t', '\n' }, StringSplitOptions.RemoveEmptyEntries);

    // foreach(var word in words)
    Console.WriteLine(words.Length);

    var wordCount = words.Length;
    if (wordCount < _minWords)
    {
        Console.WriteLine($"Страница {url} содержит менее {_minWords} слов, пропускаем");
        return false;
    }

    _pagesDownloaded++;
    var filename = Path.Combine(OutputDir, $"page_{_pagesDownloaded}.txt");

    File.WriteAllText(filename, text, Encoding.UTF8);

    File.AppendAllText(IndexFile, $"{_pagesDownloaded}\t{url}{Environment.NewLine}", Encoding.UTF8);

    Console.WriteLine($"Сохранена страница {_pagesDownloaded}: {url}");
    return true;
}
```

6. Метод для скачивания страницы с помощью http клиента

```
IL code
private async Task<bool> DownloadPageAsync(string url, HttpClient client)
{
    if (_visitedUrls.Contains(url) || _pagesDownloaded >= _minPages)
        return false;

    _visitedUrls.Add(url);

    try
    {
        var response = await client.GetAsync(url);
        response.EnsureSuccessStatusCode();

        var html = await response.Content.ReadAsStringAsync();
        var text = ExtractText(html);

        if (string.IsNullOrWhiteSpace(text))
            return false;

        return SavePage(url, text);
    }
    catch (Exception e)
    {
        Console.WriteLine($"Ошибка при скачивании {url}: {e.Message}");
        return false;
    }
}
```

7. Основной метод, который пользуется всеми предыдущими методами. Здесь я создаю два хешсета - юрлы текущего уровня, и юрлы которые спарсятся после шага цикла. Цикл идет по хешсету с текущим уровнем, достает ссылки из полученного результата и добавляет их в хешсет следующего уровня. Потом хешсет следующего уровня переходит в хешсет текущего уровня.

```
IL code
public async Task CrawlAsync()
{
    var handler = new HttpClientHandler
    {
        AutomaticDecompression = DecompressionMethods.GZip | DecompressionMethods.Deflate
    };

    using var client = new HttpClient(handler);
    client.DefaultRequestHeaders.UserAgent.ParseAdd("Mozilla/5.0");
    client.Timeout = TimeSpan.FromSeconds(10);

    var currentLevelUrls = new HashSet<string>(_startUrls);
    var nextLevelUrls = new HashSet<string>();

    while (_pagesDownloaded < _minPages && currentLevelUrls.Count > 0)
    {
        foreach (var url in currentLevelUrls)
        {
            if (_pagesDownloaded >= _minPages)
                break;

            if (await DownloadPageAsync(url, client))
            {
                var links = await GetLinksAsync(url, client);
                foreach (var link in links)
                    nextLevelUrls.Add(link);
            }

            // Небольшая задержка чтобы не перегружать сервер
            await Task.Delay(1000);
        }
```