Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический университет

имени В.Ф.Уткина»

Кафедра ЭВМ

Отчет по лабораторной работе №16

на тему

«Создание графического веб редактора»

по дисциплине  
**«Визуальное программирование»**

Выполнили:

Студенты группы №140

Бригада 5

Сафаров Д. А.

Тимохин Е. С.

Проверили:  
ст. преп. Хизриева Н.И.

ст. преп. Бастрычкин А.С.

**Цель работы**

Получение навыков написания веб-приложения для обработки изображений с использованием Razor Pages.

**Задание**

Доработать графическое веб-приложение в соответствии со следующими требованиями:

1) При загрузке изображения оно будет отображаться на странице.

2) Кнопки для применения фильтров к изображению (увеличение резкости, размытие, усиление края).

3) Изменения изображения также отображаются на странице.

4) Кнопка для открытия диалогового окна сохранения файла на

устройство.

**Практическая часть**

Код программы представлен в приложении А. Результат её работы представлен на рисунке 1.

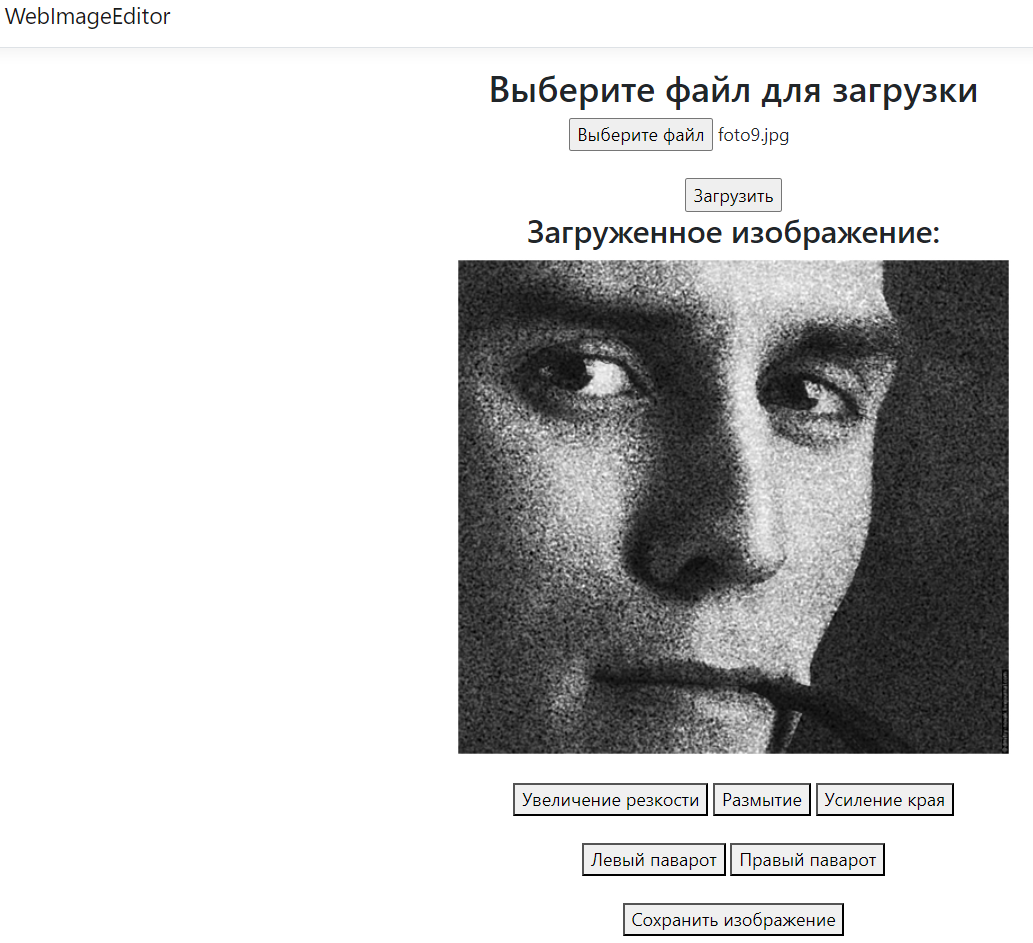


Рисунок 1 – Результат работы программы

Приложение А. Текст программы

**Index.cshtml**

@page

@model IndexModel

@{

ViewData["Title"] = "Графический редактор";

}

<div class="text-center">

<h2>Выберите файл для загрузки</h2>

<form method="post" enctype="multipart/form-data">

<input type="file" name="uploads" /><br /><br />

<input type="submit" value="Загрузить" />

</form>

<div id="image-container" style="display: none;">

<h3>Загруженное изображение:</h3>

<img id="uploaded-image" src="" alt="Uploaded Image" style="max-width: 100%; height: auto;" /><br /><br />

<button id="sharpen-button">Увеличение резкости</button>

<button id="blur-button">Размытие</button>

<button id="edge-button">Усиление края</button>

<br /><br />

<button id="left-button">Левый паварот</button>

<button id="right-button">Правый паварот</button>

<br /><br />

<button id="save-button">Сохранить изображение</button>

</div>

</div>

@section Scripts {

<script>

document.querySelector('form').addEventListener('submit', async function (event) {

event.preventDefault();

const formData = new FormData(event.target);

try {

const response = await fetch('/', {

method: 'POST',

body: formData

});

const imageUrl = await response.text();

if (imageUrl.startsWith("Error:")) {

console.error(imageUrl);

alert(imageUrl);

return;

}

console.log(`Uploaded image URL: ${imageUrl}`);

document.getElementById('uploaded-image').src = imageUrl;

document.getElementById('image-container').style.display = 'block';

} catch (error) {

console.error(`Error during image upload: ${error}`);

}

});

document.getElementById('sharpen-button').addEventListener('click', async function () {

await applyFilter('sharpen');

});

document.getElementById('blur-button').addEventListener('click', async function () {

await applyFilter('blur');

});

document.getElementById('edge-button').addEventListener('click', async function () {

await applyFilter('edge');

});

document.getElementById('left-button').addEventListener('click', async function () {

await applyFilter('left');

}); document.getElementById('right-button').addEventListener('click', async function () {

await applyFilter('right');

});

document.getElementById('save-button').addEventListener('click', function () {

const link = document.createElement('a');

link.href = document.getElementById('uploaded-image').src;

link.download = 'modified\_image.png';

link.click();

});

async function applyFilter(filter) {

const imageUrl = document.getElementById('uploaded-image').src;

console.log(`Applying filter: ${filter} to image URL: ${imageUrl}`);

try {

const relativePath = imageUrl.replace(window.location.origin, '');

const response = await fetch(`/apply-filter?filter=${filter}&imageUrl=${encodeURIComponent(relativePath)}`);

const newImageUrl = await response.text();

if (newImageUrl.startsWith("Error:")) {

console.error(newImageUrl);

alert(newImageUrl);

return;

}

console.log(`Filtered image URL: ${newImageUrl}`);

document.getElementById('uploaded-image').src = newImageUrl;

} catch (error) {

console.error(`Error during filter application: ${error}`);

}

}

</script>

}

**Index.cshtml.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using Microsoft.Extensions.Logging;

using SixLabors.ImageSharp;

using SixLabors.ImageSharp.PixelFormats;

using SixLabors.ImageSharp.Processing;

using System;

using System.IO;

using System.Linq;

namespace WebImageEditor.Pages

{

public class IndexModel : PageModel

{

private readonly ILogger<IndexModel> \_logger;

public IndexModel(ILogger<IndexModel> logger)

{

\_logger = logger;

}

public async Task<IActionResult> OnPostAsync()

{

var request = HttpContext.Request;

var file = request.Form.Files.FirstOrDefault();

if (file == null)

{

\_logger?.LogError("Файла нет");

return Content("Error: Файла нет");

}

var uploadPath = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot/uploads");

Directory.CreateDirectory(uploadPath);

var fullPath = Path.Combine(uploadPath, file.FileName);

using (var fileStream = new FileStream(fullPath, FileMode.Create))

{

await file.CopyToAsync(fileStream);

}

var relativePath = $"/uploads/{file.FileName}";

return Content(relativePath);

}

public IActionResult OnGetApplyFilter(string filter, string imageUrl)

{

try

{

if (\_logger == null) { Console.WriteLine("А где картиночка?"); }

if (string.IsNullOrWhiteSpace(imageUrl))

{

\_logger?.LogError("Файла нет");

return Content("Error: Файла нет");

}

var webRoot = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot");

var fullPath = Path.Combine(webRoot, imageUrl.TrimStart('/'));

if (!System.IO.File.Exists(fullPath))

{

\_logger?.LogError($"Файла нет: {fullPath}");

return Content("Error: Файла нет");

}

using (var image = SixLabors.ImageSharp.Image.Load<Rgba32>(fullPath))

{

\_logger?.LogInformation($"Примененние фильтра: {filter}");

switch (filter.ToLower())

{

case "sharpen":

image.Mutate(x => x.GaussianSharpen());

break;

case "blur":

image.Mutate(x => x.GaussianBlur());

break;

case "edge":

image.Mutate(x => x.DetectEdges());

break;

case "left":

image.Mutate(x => x.Rotate(RotateMode.Rotate270));

break;

case "right":

image.Mutate(x => x.Rotate(RotateMode.Rotate90));

break;

default:

\_logger?.LogError($"Unknown filter: {filter}");

return Content("Error: Unknown filter.");

}

var modifiedImagePath = Path.Combine(webRoot, "uploads", $"modified\_{Path.GetFileName(fullPath)}");

\_logger?.LogInformation($"Сохранение модифицированого изображения в: {modifiedImagePath}");

image.Save(modifiedImagePath);

var relativeModifiedPath = $"/uploads/modified\_{Path.GetFileName(fullPath)}";

return Content(relativeModifiedPath);

}

}

catch (Exception ex)

{

\_logger?.LogError(ex, "Ошибка применения фильтра");

return Content($"Error: {ex.Message}");

}

}

}

}

**Program.cs**

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

namespace WebImageEditor

{

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureLogging(logging =>

{

logging.ClearProviders();

logging.AddConsole();

})

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}