**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

Факультет «Радиоэлектронных систем и робототехники»

Кафедра «Систем автоматизации и робототехники»

|  |
| --- |
| Направление подготовки:  27.03.04 Управление в технических системах |
| Направленность (профиль):  Информационные технологии в технических системах |

**ОТЧЁТ ПО ПРАКТИКЕ**

на тему:

|  |
| --- |
| «Разработка веб-приложения для загрузки и хранения видео» |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | Выполнили студенты группы | | ИСТ-351 |
|  |  |  | Кондратенко М.Н.  Мизин А.В | | |
|  |  |  | *Фамилия И. О.* | | |
|  |  |  | Руководитель | Д.Т.Н | |
| *оценка* |  |  |  | *уч. степень, уч. звание* | |
|  |  |  | Акимов С.В. | | |
| *дата, подпись* |  |  | *Фамилия И. О.* | | |

**Введение**

В современном мире цифровых технологий возрастает потребность в удобных инструментах для хранения и обмена мультимедийным контентом. Видеофайлы занимают значительный объем данных, и их загрузка, хранение и управление требуют специализированных решений.

Данный отчет описывает процесс разработки веб-приложения для загрузки и хранения видеофайлов. Приложение предоставляет пользователям возможность загружать видео на сервер, скачивать их и просматривать информацию о загруженных файлах.

**Цели и задачи проекта**

**Цели:**

Разработать простое и удобное веб-приложение для загрузки и хранения видео.

Обеспечить возможность загрузки видеофайлов размером до 30 МБ.

Реализовать базовые функции управления видеоконтентом.

**Задачи:**

1. Спроектировать архитектуру приложения (backend и frontend).
2. Реализовать backend на ASP.NET Core для обработки загрузки и хранения видео.
3. Разработать frontend с использованием HTML и JavaScript для взаимодействия с пользователем.
4. Настроить хранение файлов на сервере.
5. Обеспечить возможность скачивания загруженных видео.
6. Реализовать отображение информации о загруженных файлах.

**Описание проекта**

Проект представляет собой веб-приложение, состоящее из двух основных компонентов:

**Backend** (ASP.NET Core Web API) – отвечает за обработку HTTP-запросов, загрузку файлов, их хранение и выдачу по запросу.

**Frontend** (HTML, JavaScript) – предоставляет пользовательский интерфейс для загрузки и скачивания видео.

**Основные функции:**

Загрузка видео на сервер.

Хранение видеофайлов в указанной директории (wwwroot/uploads).

Скачивание видео по запросу.

Получение информации о загруженных файлах (название, размер, дата загрузки).

**Целевая аудитория:**

Начинающие разработчики, изучающие ASP.NET Core.

Преподаватели и студенты, использующие проект в образовательных целях.

Пользователи, которым требуется простое хранилище для видеофайлов.

**Архитектура приложения**

Приложение построено по клиент-серверной модели

**Backend:**

**ASP.NET Core Web API** – обрабатывает HTTP-запросы.

**VideoController** – содержит методы для загрузки, скачивания и получения информации о видео.

**VideoService** – реализует бизнес-логику работы с файлами.

**Frontend:**

**HTML-страница** (index.html) – содержит форму загрузки и список видео.

**JavaScript** – отправляет запросы к API и обновляет интерфейс.

**Хранение данных:**

Видеофайлы сохраняются локально в папке wwwroot/uploads.

**Используемые технологии**

* **Backend:**
  + ASP.NET Core
  + C#
  + Newtonsoft.Json (для работы с JSON)
* **Frontend:**
  + HTML5
  + JavaScript (Fetch API для взаимодействия с backend)
* **Инструменты разработки:**
  + Visual Studio / Visual Studio Code
  + Swagger (для тестирования API)

**Реализация backend (ASP.NET Core Web API)**

**Основные компоненты:**

**Program.cs** – настройка сервера, регистрация сервисов.

**VideoController.cs** – обработка запросов на загрузку и скачивание.

**VideoService.cs** – работа с файловой системой.

**Ключевые моменты:**

Настройка максимального размера загружаемого файла (30 МБ).

Использование FileExtensionContentTypeProvider для определения MIME-типов.

Обработка ошибок при загрузке и скачивании.

**Реализация frontend (HTML, JavaScript)**

Форма загрузки (<input type="file">).

Отправка файла через FormData и Fetch API.

Динамическое обновление списка видео.

**Тестирование и отладка**

Проверка загрузки файлов разных форматов (MP4, AVI, MOV).

Тестирование скачивания и отображения информации о файлах.

Использование Swagger для проверки API.

**Заключение**

В ходе практики было разработано веб-приложение для загрузки и хранения видео. Проект успешно решает поставленные задачи и может быть расширен (например, добавлением авторизации или облачного хранилища).

Приобретенные навыки:

* Работа с ASP.NET Core Web API.
* Обработка файлов в C#.
* Взаимодействие frontend и backend через Fetch API.

**Список использованных источников**

1. Официальная документация ASP.NET Core.
2. MDN Web Docs (JavaScript, Fetch API).
3. Учебные материалы по C# и веб-разработке.

**Приложения:**

* Исходный код проекта.
* Скриншоты интерфейса.

**Program.cs**

using Microsoft.AspNetCore.Builder;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.AspNetCore.StaticFiles;

using Microsoft.AspNetCore.Http.Features;

using WebApi.Services;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews()

.AddNewtonsoftJson(options =>

{

options.SerializerSettings.MaxDepth = 64;

});

builder.Services.AddScoped<VideoService>(); // Простая регистрация, IConfiguration передается автоматически

builder.Services.AddSingleton<IContentTypeProvider, FileExtensionContentTypeProvider>();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllers();

app.Run();

**VideosController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.StaticFiles;

using WebApi.Services;

using System;

namespace WebApi.Controllers

{

[ApiController]

[Route("api/[controller]")]

public class VideosController : ControllerBase

{

private readonly VideoService \_videoService;

private readonly IContentTypeProvider \_contentTypeProvider;

public VideosController(VideoService videoService, IContentTypeProvider contentTypeProvider)

{

\_videoService = videoService;

\_contentTypeProvider = contentTypeProvider;

}

[HttpPost("upload")]

public async Task<IActionResult> UploadVideo(IFormFile file)

{

if (file == null || file.Length == 0)

{

return BadRequest("No file uploaded or file is empty.");

}

try

{

using (var stream = file.OpenReadStream())

{

string videoId = await \_videoService.UploadVideoAsync(stream, file.FileName);

return Ok(new { VideoId = videoId, Filename = file.FileName });

}

}

catch (Exception ex)

{

return StatusCode(500, $"Internal server error: {ex.Message}");

}

}

[HttpGet("download/{filename}")]

public async Task<IActionResult> DownloadVideo(string filename)

{

try

{

var fileBytes = await \_videoService.DownloadVideoAsync(filename);

if (fileBytes == null || fileBytes.Length == 0)

{

return NotFound($"Video with filename '{filename}' not found.");

}

if (!\_contentTypeProvider.TryGetContentType(filename, out var contentType))

{

contentType = "application/octet-stream";

}

return File(fileBytes, contentType, filename);

}

catch (Exception ex)

{

return StatusCode(500, $"Internal server error: {ex.Message}");

}

}

[HttpGet("info/{filename}")]

public async Task<IActionResult> GetVideoInfo(string filename)

{

try

{

var fileInfo = await \_videoService.GetFileInfoAsync(filename);

if (fileInfo == null)

{

return NotFound($"Video with filename '{filename}' not found.");

}

return Ok(new

{

Filename = fileInfo.Name,

Length = fileInfo.Length,

UploadDate = fileInfo.CreationTime

});

}

catch (Exception ex)

{

return StatusCode(500, $"Internal server error: {ex.Message}");

}

}

}

}

**VideoService.cs**

using Microsoft.Extensions.Configuration;

using System.IO;

using System.Threading.Tasks;

namespace WebApi.Services

{

public class VideoService

{

private readonly string \_videoUploadPath;

private readonly IConfiguration \_configuration;

public VideoService(IConfiguration configuration)

{

\_configuration = configuration;

\_videoUploadPath = configuration["VideoUploadPath"];

}

public async Task<string> UploadVideoAsync(Stream stream, string fileName)

{

string filePath = Path.Combine(\_videoUploadPath, fileName);

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

{

await stream.CopyToAsync(fileStream);

}

return fileName; // Возвращаем имя файла как ID (для простоты)

}

public async Task<byte[]> DownloadVideoAsync(string filename)

{

string filePath = Path.Combine(\_videoUploadPath, filename);

if (!File.Exists(filePath))

{

return null;

}

return await File.ReadAllBytesAsync(filePath);

}

public async Task<System.IO.FileInfo> GetFileInfoAsync(string filename)

{

string filePath = Path.Combine(\_videoUploadPath, filename);

if (!File.Exists(filePath))

{

return null;

}

System.IO.FileInfo fileInfo = new System.IO.FileInfo(filePath);

return fileInfo; // Возвращаем System.IO.FileInfo напрямую

}

}

}

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Video Upload and Download</title>

</head>

<body>

<h1>Video Upload and Download</h1>

<form id="uploadForm" enctype="multipart/form-data">

<label for="videoFile">Select Video File:</label>

<input type="file" id="videoFile" name="file" accept="video/\*">

<button type="submit">Upload</button>

</form>

<h2>Uploaded Videos</h2>

<ul id="videoList">

</ul>

<script>

const uploadForm = document.getElementById('uploadForm');

const videoList = document.getElementById('videoList');

uploadForm.addEventListener('submit', async (event) => {

event.preventDefault();

const fileInput = document.getElementById('videoFile');

const file = fileInput.files[0];

if (!file) {

alert('Please select a video file.');

return;

}

const formData = new FormData();

formData.append('file', file);

try {

const response = await fetch('/api/Videos/upload', {

method: 'POST',

body: formData,

});

if (response.ok) {

const data = await response.json();

console.log('Upload successful:', data);

fetchVideoList();

} else {

console.error('Upload failed:', response.statusText);

alert('Upload failed: ' + response.statusText);

}

} catch (error) {

console.error('Error uploading file:', error);

alert('Error uploading file: ' + error.message);

}

});

async function fetchVideoList() {

videoList.innerHTML = '';

try {

const response = await fetch('/api/Videos/info'); // Получить список

if (!response.ok) {

throw new Error(`HTTP error! Status: ${response.status}`);

}

const videos = await response.json();

videos.forEach(video => {

const listItem = document.createElement('li');

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

<a href="/api/Videos/download/MyVideo.mp4" download="MyVideo.mp4">Скачать</a> // Ссылка на скачивание

downloadLink.textContent = `Download: ${video.filename}`;

downloadLink.download = video.filename;

listItem.appendChild(downloadLink);

videoList.appendChild(listItem);

});

} catch (error) {

console.error('Failed to fetch video list:', error);

videoList.innerHTML = '<li>Failed to load videos.</li>';

}

}

fetchVideoList();

</script>

</body>

</html>

**Appsettins.json**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning",

"Microsoft": "Information"

}

},

"AllowedHosts": "\*",

"VideoUploadPath": "wwwroot\\uploads"

}

**launchSettins.json**

﻿{

"$schema": "https://json.schemastore.org/launchsettings.json",

"iisSettings": {

"windowsAuthentication": false,

"anonymousAuthentication": true,

"iisExpress": {

"applicationUrl": "http://localhost:36905",

"sslPort": 44329

}

},

"profiles": {

"WebApplication351": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"launchUrl": "swagger",

"applicationUrl": "https://localhost:7244;http://localhost:5186",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true,

"launchUrl": "swagger",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

}

}

}