using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using Microsoft.AspNetCore.SignalR;

using Mallenom.Framework;

using Mallenom.Imaging;

using System;

using System.Text.Json;

using System.IO;

using System.IO.MemoryMappedFiles;

using Viscont.Core.Framework.ImageDataTransmission;

using Viscont.Core.Service.ImageDataTransmission.Data;

using Viscont.Core.Service.ImageDataTransmission.Hubs;

using Mallenom.Imaging.Jpeg;

namespace Viscont.Core.Service.ImageDataTransmission.Controllers;

[ApiController]

[Route("Images")]

public class ImageDataTransmissionController : ControllerBase

{

#region Data

private readonly IImageRepository \_imageRepository;

private readonly ILogger<ImageDataTransmissionController> \_log;

#endregion

#region .ctor

public ImageDataTransmissionController(

IImageRepository imageRepository,

IHubContext<NotificationHub> hub,

ILogger<ImageDataTransmissionController> logger)

{

\_imageRepository = imageRepository;

\_log = logger;

}

#endregion

#region GET Get image data

/// <summary> Отдает изображение с репозитория </summary>

[HttpGet]

[Route("Get/Image")]

public IActionResult GetImage([FromBody]ImageMetadataModel metadataModel)

{

var guid = Guid.Parse(metadataModel.FileName);

if(!\_imageRepository.TryGetByGuid(guid, out var imageEntry))

{

return NotFound();

}

using var imageData = imageEntry.RepresentAsImageData();

var stream = new MemoryStream();

switch(metadataModel.FileFormat)

{

case "bmp":

BmpImage.Write(imageData, stream);

break;

case "jpg" or "jpeg":

JpegImage.Encode(imageData, stream);

break;

default:

throw new ArgumentOutOfRangeException(nameof(metadataModel.FileFormat));

}

stream.Seek(0, SeekOrigin.Begin);

return Ok(stream);

}

/// <summary> Отдает данные изображения с репозитория </summary>

[HttpGet]

[Route("Get/ImageMetadata")]

public IActionResult GetImageMetadata(Guid guid)

=> \_imageRepository.TryGetByGuid(guid, out var imageEntry) ? Ok(imageEntry.ImageMetadata) : NotFound();

#endregion

#region GET Save image

/// <summary> Считывает и сохраняет в репозиторий файл из памяти </summary>

[HttpPost]

[Route("Save/Image")]

public Guid SaveImage([FromBody]ImageMetadataModel metadataModel)

{

var guid = Guid.NewGuid();

var dataFormat = ImageDataFormat.GetFormatByName(metadataModel.Format)

?? throw new Exception($"Format not found. Source: {this}");

var memoryMappedFile = OperatingSystem.IsWindows()

? MemoryMappedFile.OpenExisting(metadataModel.FileName)

: throw new PlatformNotSupportedException();

try

{

var metaData = new ImageMetadata(

metadataModel.FileName,

dataFormat,

metadataModel.Width,

metadataModel.Height,

string.Empty);

\_imageRepository.Add(

guid,

new ImageEntry(metaData, memoryMappedFile));

\_log.LogInformation("Save new image: " + guid.ToString());

return guid;

}

catch

{

memoryMappedFile.Dispose();

throw;

}

}

#endregion

#region GET Remove image

/// <summary> Удаляет файл из репозиторий </summary>

[HttpDelete]

[Route("Remove/Image")]

public IActionResult RemoveImage(Guid guid)

=> \_imageRepository.Remove(guid) ? Ok() : NotFound();

#endregion

}