using System;

using Viscont.Core.Service.ImageDataTransmission.Data;

namespace Viscont.Core.Service.ImageDataTransmission.Data;

public interface IImageRepository

{

/// <summary> Отдает количество файлов в репозитории </summary>

int GetCount();

/// <summary> Добавление изображения </summary>

void Add(Guid imageId, ImageEntry imageMetadataRepositoty);

/// <summary> Удаление </summary>

bool Remove(Guid imageId);

/// <summary> Отдает Данные изображения репозитория по Guid </summary>

ImageEntry GetByGuid(Guid guid);

bool TryGetByGuid(Guid imageId, out ImageEntry entry);

}

using System;

using Viscont.Core.Service.ImageDataTransmission.Data;

namespace Viscont.Core.Service.ImageDataTransmission.Data;

public interface IImageRepository

{

/// <summary> Отдает количество файлов в репозитории </summary>

int GetCount();

/// <summary> Добавление изображения </summary>

void Add(Guid imageId, ImageEntry imageMetadataRepositoty);

/// <summary> Удаление </summary>

bool Remove(Guid imageId);

/// <summary> Отдает Данные изображения репозитория по Guid </summary>

ImageEntry GetByGuid(Guid guid);

bool TryGetByGuid(Guid imageId, out ImageEntry entry);

}

using Mallenom.Imaging;

using System;

using System.Collections.Generic;

using System.IO.MemoryMappedFiles;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Viscont.Core.Service.ImageDataTransmission.Data

{

public class MemoryMappedViewAccessorImageDataBuffer : IImageDataBuffer

{

public IntPtr Pointer { get; }

public int Size { get; }

private readonly MemoryMappedViewAccessor \_accessor;

public unsafe MemoryMappedViewAccessorImageDataBuffer(

MemoryMappedViewAccessor accessor,

int bufferSize)

{

byte\* ptr = null;

\_accessor = accessor;

\_accessor.SafeMemoryMappedViewHandle.AcquirePointer(ref ptr);

Pointer = new IntPtr(ptr);

Size = bufferSize;

}

public void Dispose()

{

\_accessor.SafeMemoryMappedViewHandle.ReleasePointer();

\_accessor.Dispose();

}

}

}