using System;

using System.Text;

using System.Text.Json;

using System.Threading.Tasks;

using System.Net.Http;

using Microsoft.AspNetCore.SignalR.Client;

using Mallenom.Framework;

using Mallenom.Imaging;

using Viscont.Core.Framework.ImageDataTransmission;

namespace Viscont.Core.Client.ImageDataTransmission;

public class ImageDataTransmissionClient : IImageDataTransmissionClient, IDisposable

{

#region Data

private readonly HttpClient \_httpClient;

private readonly IImageDataWriter \_writer;

#endregion

#region .ctor

public ImageDataTransmissionClient()

{

\_httpClient = new HttpClient()

{

BaseAddress = new Uri(Url.BaseUrl)

};

\_writer = new ImageDataWriter();

}

public void Dispose()

{

\_httpClient.Dispose();

}

#endregion

#region Implementation

public async Task<Guid> SaveImageAsync(IReadOnlyReference<ImageData> imageReference)

{

var url = Url.ImagesUrl + Url.SaveUrl + Url.Image;

//Create temp Guid

var guid = Guid.NewGuid();

var model = new ImageMetadataModel(

FileName: guid.ToString(),

Width: imageReference.Value.Width,

Height: imageReference.Value.Height,

Format: imageReference.Value?.Format.Name,

FileFormat: string.Empty);

var jsonContent = JsonSerializer.Serialize(model);

var content = new StringContent(

jsonContent,

Encoding.UTF8,

"application/json");

using var writer = \_writer.WriteImageToMemory(guid, imageReference.Value!);

var result = \_httpClient.PostAsync(url, content).Result;

var imageId = await result.Content.ReadAsStringAsync();

imageId = imageId.Substring(1, imageId.Length-2);

if(!Guid.TryParse(imageId, out guid))

{

throw new Exception("Guid.TryParse can't parse");

}

result.Dispose();

return guid;

}

public async Task<HubConnection> SubscribeOnNewImage(Action<Guid> action)

{

var hubConnection = new HubConnectionBuilder()

.WithUrl(Url.BaseUrl)

.Build();

hubConnection.On<Guid>(Url.NewImageMethod, message => action(message));

await hubConnection.StartAsync();

return hubConnection;

}

#endregion

}