Титул «Курсовая работа»

Титул «задание»

Реферат

Пояснительная записка курсового проекта 44 страниц, 8 рисунков, 10 источников, 2 приложения.

ПРОГРАММА, КОНВЕРТИРУЮЩАЯ ИЗОБРАЖЕНИЯ, ФОРМАТЫ ИЗОБРАЖЕНИЯ, КЛАССЫ КОНВЕРТАЦИИ ИЗОБРАЖЕНИЙ, ПРОЕКТИРОВАНИЕ, ДИАГРАММЫ, КЛАССЫ, C#, WPF, MVVM

Целью работы является создание программы для конвертации пакета изображений в формат JPG.

Результатом работы является десктоп-приложение, реализующее все необходимые функции для конвертации пакетов изображения в JPG формат, а именно: выбор файлов; редактирование списка изображений подлежащих конвертации; конвертирование выбранных файлов, и их сохранение в заданный пользователем путь;

Данный программный продукт может в дальнейшем использоваться как пользователями, работающими в сфере медиа, так и рядовые пользователи ПК.

Содержание

Введение

Целью курсового проекта является проектирование и разработка простой программы для конвертации пакетов изображений в формат JPG.

Основные задачи:

- изучить предметную область и провести анализ существующих расширений;
- подробнее изучить формат JPG;
- спроектировать программный продукт;
- реализовать программный продукт;
- составить пояснительную записку проекта.

Не смотря на существование более совершенных программ для конвертации изображения, приведенную в курсовом проекте программу можно использовать как простой пример программы конвертации пакетов изображений в формат JPG в целях обучения и изучения.

1 Нормативные ссылки

В данном курсовом проекте использованы следующие нормативные ссылки:

ГОСТ Р 1.5-2004 Стандартизация в Российской Федерации. Стандарты национальные Российской Федерации. Правила построения, изложения, оформления и обозначения.

ГОСТ Р 1.12-2004 Стандартизация в Российской Федерации. Термины и определения.

ГОСТ Р 7.0.5-2008 СИБИД. Библиографическая ссылка. Общие требования и правила составления.

ГОСТ Р ИСО 9000-2008 Системы менеджмента качества. Основные положения и словарь.

2 Исследование предметной области

2.1 Графические форматы

Графический формат – это способ записи графической информации. Графические форматы файлов предназначены для хранения изображений, таких как фотографии и рисунки.

Графические форматы делятся на векторные и растровые. Большинство графических форматов реализуют сжатие данных (одни – с потерями, другие – без).

2.2 Растровая Графика

Растровое изображение (лат. Rastrum – скребок, грабли) – изображение, представляющее собой сетку (мозаику) пикселей – цветных точек (обычно прямоугольных) на мониторе, бумаге и других отображающих устройствах.

Важными характеристиками изображения являются:

– Размер изображения в пикселях – может выражаться в видео количества пикселей по ширине и по высоте (800 × 600 px, 1024 × 768 px, 1600 × 1200 px и т. д.) или же в виде общего количества пикселей (так, изображение размером 1600 × 1200 px состоит из 1 920 000 точек, то есть примерно из двух мегапикселей);

– Количество используемых цветов или глубина цвета (эти характеристики имеют следующую зависимость: $N = 2^k$, где N — количество цветов, k – глубина цвета);

– Цветовое пространство (цветовая модель) – RGB, CMYK, XYZ, YCbCr и др.;

– Разрешение изображения – величина, определяющая количество точек (элементов растрового изображения) на единицу площади (или единицу длины).

Растровую графику редактируют с помощью растровых графических редакторов. Создаётся растровая графика фотоаппаратами, сканерами, непосредственно в растровом редакторе, а также путём экспорта из векторного редактора или в виде снимков экрана.

Преимущества растровой графики:

– растровая графика позволяет создать практически любой рисунок, вне зависимости от сложности, в отличие, например, от векторной, где невозможно точно передать эффект перехода от одного цвета к другому без потерь в размере файла;

– распространённость – растровая графика используется сейчас практически везде: от маленьких значков до плакатов;

– высокая скорость обработки сложных изображений, если не нужно масштабирование;

– растровое представление изображения естественно для большинства устройств ввода-вывода графической информации, таких как мониторы (за исключением векторных устройств вывода), матричные и струйные принтеры, цифровые фотоаппараты, сканеры, а также сотовые телефоны.

Недостатки:

– большой размер файлов у простых изображений из большого количества точек;

– невозможность идеального масштабирования;

– невозможность вывода на печать на векторный графопостроитель.

Из-за этих недостатков для хранения простых рисунков рекомендуют вместо даже сжатой растровой графики использовать векторную графику.

Растровые изображения обычно хранятся в сжатом виде. В зависимости от типа сжатия может быть возможно или невозможно восстановить изображение в точности таким, каким оно было до сжатия

(сжатие без потерь или сжатие с потерями соответственно). Также в графическом файле могут храниться дополнительные данные: об авторе файла, фотокамере и её настройках, количестве точек на дюйм при печати, место съёмки (если изображение – снимок), программное обеспечение, использованное для подготовки, и др.

– использует алгоритмы сжатия, основанные на уменьшении избыточности информации (сжатие без потерь);

– BMP или Windows Bitmap – обычно используется без сжатия, хотя возможно использование алгоритма RLE;

– GIF (Graphics Interchange Format) – устаревающий формат, поддерживающий не более 256 цветов одновременно. Всё ещё популярен из-за поддержки анимации, которая отсутствует в чистом PNG, хотя ПО начинает поддерживать APNG;

– PCX – устаревший формат;

– PNG (Portable Network Graphics) – растровый формат, в основе которого алгоритм сжатия Deflate;

– JPEG-LS в режиме сжатия без потерь – алгоритм использует адаптивное предсказание значения текущего пиксела по окружению, включающему уже закодированные пикселы;

– Lossless JPEG – быстрый, но малоэффективный алгоритм сжатия, использующий (при обходе изображения попиксельно слева направо, сверху вниз) простое неадаптивное предсказание значения текущего пиксела по значениям верхнего, левого и верхнего левого пикселов.

Основано на отбрасывании части информации, как правило, наименее воспринимаемой глазом (сжатие с потерями).

JPEG – очень широко используемый формат изображений. Сжатие использует разбиение изображения на блоки, квантование пространственных спектральных компонент в каждом блоке изображения с последующим их энтропийным кодированием. При детальном рассмотрении сильно сжатого

изображения заметно размытие резких границ и характерный муар вблизи них. При невысоких степенях сжатия восстановленное изображение визуально неотличимо от исходного.

Информация в данном разделе найдена на электронном ресурсе [2].

2.3 Формат изображения JPEG

JPEG (произносится «джейпег», англ. Joint Photographic Experts Group, по названию организации-разработчика) — один из популярных растровых графических форматов, применяемый для хранения фотографий и подобных им изображений. Файлы, содержащие данные JPEG, обычно имеют расширения (суффиксы) .jpg, .jfif, .jpe или .jpeg.

Алгоритм JPEG позволяет сжимать изображение как с потерями, так и без потерь (режим сжатия lossless JPEG). Поддерживаются изображения с линейным размером не более 65535 × 65535 пикселов.

Все знают и часто встречают данный формат формат изображения – JPEG. Однако, иногда на практике можно увидеть и картинки с похожим форматом JPG.

Отличие межу JPG и JPEG заключается только в букве «е». На практике при работе с файлами этих форматов никаких отличий нет. Это абсолютно одинаковые форматы изображений.

Формат Jpg появился из-за того, что в старых операционных системах нельзя было дать расширению файла значение, содержащее более чем три символа. В связи с этим, расширение JPEG сократили до JPG. В новых же версиях операционных систем, расширение может содержать и четыре, и пять символов, и даже больше. Поэтому было решено вернуть картинкам букву «е» и на практике стали применять вариант JPEG. Однако традиция записывать формат в трёхбуквенном варианте до сих пор осталась, поэтому и сегодня в

новых операционных системах всё ещё можно встретить написание JPG, что на деле идентично формату JPEG.

2.3.1 Область применения

Алгоритм JPEG наиболее эффективен для сжатия фотографий и картин, содержащих реалистичные сцены с плавными переходами яркости и цвета. Наибольшее распространение JPEG получил в цифровой фотографии и для хранения и передачи изображений с использованием Интернета.

Формат JPEG в режиме сжатия с потерями малопригоден для сжатия чертежей, текстовой и знаковой графики, где резкий контраст между соседними пикселами приводит к появлению заметных артефактов. Такие изображения целесообразно сохранять в форматах без потерь, таких как JPEG-LS, TIFF, GIF, PNG, либо использовать режим сжатия Lossless JPEG.

JPEG (как и другие форматы сжатия с потерями) не подходит для сжатия изображений при многоэтапной обработке, так как искажения в изображения будут вноситься каждый раз при сохранении промежуточных результатов обработки.

JPEG не должен использоваться и в тех случаях, когда недопустимы даже минимальные потери, например при сжатии астрономических или медицинских изображений. В таких случаях может быть рекомендован предусмотренный стандартом JPEG режим сжатия Lossless JPEG (который, однако, не поддерживается большинством популярных кодеков) или стандарт сжатия JPEG-LS.

2.3.2 Сжатие

При сжатии изображение преобразуется из цветового пространства RGB в YCbCr. Стандарт JPEG (ISO/IEC 10918-1) не регламентирует выбор именно YCbCr, допуская и другие виды преобразования (например, с числом

компонентов, отличным от трёх), и сжатие без преобразования (непосредственно в RGB), однако спецификация JFIF (JPEG File Interchange Format, предложенная в 1991 году специалистами компании C-Cube Microsystems, и ставшая в настоящее время стандартом де-факто) предполагает использование преобразования RGB->YCbCr.

После преобразования RGB->YCbCr для каналов изображения Cb и Cr, отвечающих за цвет, может выполняться «прореживание» (subsampling), которое заключается в том, что каждому блоку из 4 пикселей (2x2) яркостного канала Y ставятся в соответствие усреднённые значения Cb и Cr (схема прореживания «4:2:0»). При этом для каждого блока 2x2 вместо 12 значений (4 Y, 4 Cb и 4 Cr) используется всего 6 (4 Y и по одному усреднённому Cb и Cr). Если к качеству восстановленного после сжатия изображения предъявляются повышенные требования, прореживание может выполняться лишь в каком-то одном направлении – по вертикали (схема «4:4:0») или по горизонтали («4:2:2»), или не выполняться вовсе («4:4:4»).

Стандарт допускает также прореживание с усреднением Cb и Cr не для блока 2x2, а для четырёх расположенных последовательно (по вертикали или по горизонтали) пикселей, то есть для блоков 1x4, 4x1 (схема «4:1:1»), а также 2x4 и 4x2 (схема «4:1:0»). Допускается также использование различных типов прореживания для Cb и Cr, но на практике такие схемы применяются исключительно редко.

Далее яркостный компонент Y и отвечающие за цвет компоненты Cb и Cr разбиваются на блоки 8x8 пикселей. Каждый такой блок подвергается дискретному косинусному преобразованию (ДКП). Полученные коэффициенты ДКП квантуются (для Y, Cb и Cr в общем случае используются разные матрицы квантования) и пакуются с использованием кодирования серий и кодов Хаффмана. Стандарт JPEG допускает также использование значительно более эффективного арифметического кодирования, однако из-за патентных ограничений (патент на описанный в

стандарте JPEG арифметический QM-кодер принадлежит IBM) на практике оно используется редко. В популярную библиотеку libjpeg последних версий включена поддержка арифметического кодирования, но с просмотром сжатых с использованием этого метода изображений могут возникнуть проблемы, поскольку многие программы просмотра не поддерживают их декодирование.

Матрицы, используемые для квантования коэффициентов ДКП, хранятся в заголовочной части JPEG-файла. Обычно они строятся так, что высокочастотные коэффициенты подвергаются более сильному квантованию, чем низкочастотные. Это приводит к огрублению мелких деталей на изображении. Чем выше степень сжатия, тем более сильному квантованию подвергаются все коэффициенты.

При сохранении изображения в JPEG-файле кодеру указывается параметр качества, задаваемый в некоторых условных единицах, например, от 1 до 100 или от 1 до 10. Большее число обычно соответствует лучшему качеству (и большему размеру сжатого файла). Однако, в самом JPEG-файле такой параметр отсутствует, а качество восстановленного изображения определяется матрицами квантования, типом прореживания цветоразностных компонентов и точностью выполнения математических операций как на стороне кодера, так и на стороне декодера. При этом даже при использовании наивысшего качества (соответствующего матрице квантования, состоящей из одних только единиц, и отсутствию прореживания цветоразностных компонентов) восстановленное изображение не будет в точности совпадать с исходным, что связано как с конечной точностью выполнения ДКП, так и с необходимостью округления значений Y, Cb, Cr и коэффициентов ДКП до ближайшего целого. Режим сжатия Lossless JPEG, не использующий ДКП, обеспечивает точное совпадение восстановленного и исходного изображений, однако его малая эффективность (коэффициент сжатия редко

превышает 2) и отсутствие поддержки со стороны разработчиков программного обеспечения не способствовали популярности Lossless JPEG.

2.3.3 Разновидности схем сжатия JPEG

Стандарт JPEG предусматривает два основных способа представления кодируемых данных.

Наиболее распространённым, поддерживаемым большинством доступных кодеков, является последовательное (sequential JPEG) представление данных, предполагающее последовательный обход кодируемого изображения разрядностью 8 бит на компоненту (или 8 бит на пиксель для чёрно-белых полутоновых изображений) поблочно слева направо, сверху вниз. Над каждым кодируемым блоком изображения осуществляются описанные выше операции, а результаты кодирования помещаются в выходной поток в виде единственного «скана», то есть массива кодированных данных, соответствующего последовательно пройденному («просканированному») изображению. Основной или «базовый» (baseline) режим кодирования допускает только такое представление (и хаффмановское кодирование квантованных коэффициентов ДКП). Расширенный (extended) режим наряду с последовательным допускает также прогрессивное (progressive JPEG) представление данных, кодирование изображений разрядностью 12 бит на компоненту/пиксель (сжатие таких изображений спецификацией JFIF не поддерживается) и арифметическое кодирование квантованных коэффициентов ДКП.

В случае progressive JPEG сжатые данные записываются в выходной поток в виде набора сканов, каждый из которых описывает изображение полностью с всё большей степенью детализации. Это достигается либо путём записи в каждый скан не полного набора коэффициентов ДКП, а лишь какой-то их части: сначала – низкочастотных, в следующих сканах – высокочастотных (метод «spectral selection» то есть спектральных выборок), либо путём последовательного, от скана к скану, уточнения коэффициентов

ДКП (метод «successive approximation», то есть последовательных приближений). Такое прогрессивное представление данных оказывается особенно полезным при передаче сжатых изображений с использованием низкоскоростных каналов связи, поскольку позволяет получить представление обо всём изображении уже после передачи незначительной части JPEG-файла.

Обе описанные схемы (и sequential, и progressive JPEG) базируются на ДКП и принципиально не позволяют получить восстановленное изображение абсолютно идентичным исходному. Однако стандарт допускает также сжатие, не использующее ДКП, а построенное на основе линейного предсказателя (lossless, то есть «без потерь», JPEG), гарантирующее полное, бит-в-бит, совпадение исходного и восстановленного изображений. При этом коэффициент сжатия для фотографических изображений редко достигает 2, но гарантированное отсутствие искажений в некоторых случаях оказывается востребованным. Заметно большие степени сжатия могут быть получены при использовании не имеющего, несмотря на сходство в названиях, непосредственного отношения к стандарту JPEG ISO/IEC 10918-1 (ITU T.81 Recommendation) метода сжатия JPEG-LS, описываемого стандартом ISO/IEC 14495-1 (ITU T.87 Recommendation).

2.3.4 Достоинства и недостатки формата

К недостаткам сжатия по стандарту JPEG следует отнести появление на восстановленных изображениях при высоких степенях сжатия характерных артефактов: изображение рассыпается на блоки размером 8×8 пикселей (этот эффект особенно заметен на областях изображения с плавными изменениями яркости), в областях с высокой пространственной частотой (например, на контрастных контурах и границах изображения) возникают артефакты в виде шумовых ореолов. Стандарт JPEG (ISO/IEC 10918-1, Annex K, п. K.8) предусматривает использование специальных фильтров для подавления блоковых артефактов, но на практике подобные фильтры, несмотря на их высокую эффективность, практически не используются.

Однако, несмотря на недостатки, JPEG получил очень широкое распространение из-за достаточно высокой (относительно существовавших во время его появления альтернатив) степени сжатия, поддержке сжатия полноцветных изображений и относительно невысокой вычислительной сложности.

Информация в данном разделе найдена на электронных ресурсах [3] и [4].

3 Технические требования

3.1 Общие сведения о программном продукте

«Package Image Converter Jpeg» – десктоп-приложение, разработанное при помощи языка программирования C#. Приложение позволяет за раз конвертировать несколько файлов с изображениями в формат JPG. Конвертация происходит при помощи специальных методов для работы с файлами изображений.

Программа разрабатывается для использования в практическом и лабораторном курсе дисциплин кафедры ИСП. Программа предоставляет только самые необходимые функции для среднестатистического пользователя.

3.2 Требования к функциональным характеристикам

Программа должна предоставлять пользователю следующий функционал:

- выбор файлов, которые необходимо конвертировать;
- просмотр списка файлов, подлежащих конвертации;
- редактирование списка файлов, подлежащих конвертации;
- сохранение сконвертированных изображений в указанный пользователем путь.

3.3 Требования к надежности

Программа должна соответствовать следующим требованиям к надежности:

– ПО должно иметь защиту от некорректных действий пользователей и ошибочных исходных данных;

– ПО не должно во время работы модифицировать свой код или код других программ;

– ПО должно корректно считывать данные файлов.

3.5 Требования к составу и параметрам технических средств

Для нормального функционирования кодировщика необходима следующая минимальная конфигурация ПК:

– 32-разрядный или 64-разрядный процессор с тактовой частотой 1 ГГц или выше;

– 1 ГБ ОЗУ для 32-разрядного процессора или 2 ГБ ОЗУ для 64-разрядного процессора;

– 16 ГБ для 32-рязрядной системы или 20 ГБ для 64-разрядной системы свободного места на жестком диске;

– операционную систему Windows 7 или новее.

4 Проектирование программного обеспечения

4.1 Диаграмма вариантов использования

Вариант использования (use case) представляет собой последовательность действий, выполняемых системой в ответ на событие, инициализируемое некоторым внешним объектом.

Диаграмма вариантов использования отображена на рисунке 1, она отражает все возможные взаимодействия пользователя с программой, а именно:

- выбор файлов, которые необходимо конвертировать;
- просмотр списка файлов, подлежащих конвертации;
- редактирование списка файлов, подлежащих конвертации;
- сохранение сконвертированных изображений в указанный пользователем путь.

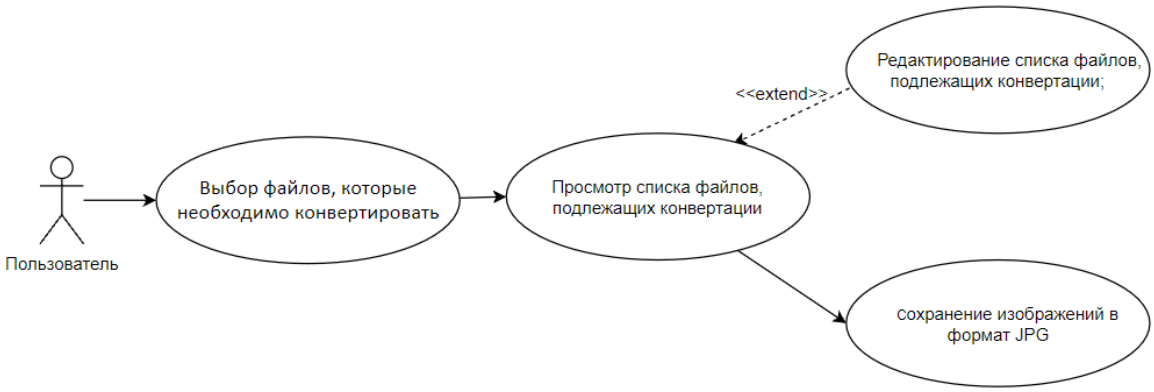

Рисунок 1 – Диаграмма вариантов использования

Данная диаграмма была построена при помощи ресурса [5].

4.2 Диаграмма состояний

Диаграммы состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий.

Диаграмма состояний изображена на рисунке 2. На ней расписаны все возможные процессы работы программы и переходы между ними.

После запуска программы она имеет свое начальное состояние бездействия.

Если пользователь нажмет кнопку «Добавить изображение», программа откроет окно выбора файлов изображений, которые нужно будет сконвертировать. Выбрав необходимые файлы, пользователь нажимает на кнопку «Открыть», и переходит на главную форму с обновлённым списком файлов подлежащих конвертации, и сообщением о повторяющихся файлах, если таковые имеются.

Если пользователь передумает конвертировать один из файлов, то ему необходимо будет нажать на кнопку «Удалить изображение». После нажатия, откроется форма, куда пользователь должен будет ввести путь к файлу, который уже не надо будет конвертировать, и нажать кнопу «Удалить из списка». После этого форма удаления записи закрывается, и пользователь попадает обратно на главную форму, и наблюдает обновившийся список.

Если пользователь будет готов к конвертации, ему надо будет ввести в поле «Куда конвертировать (путь)» путь куда будут сохранены конвертированные в формат JPG файлы изображения, и нажать кнопку «Конвертировать изображения».

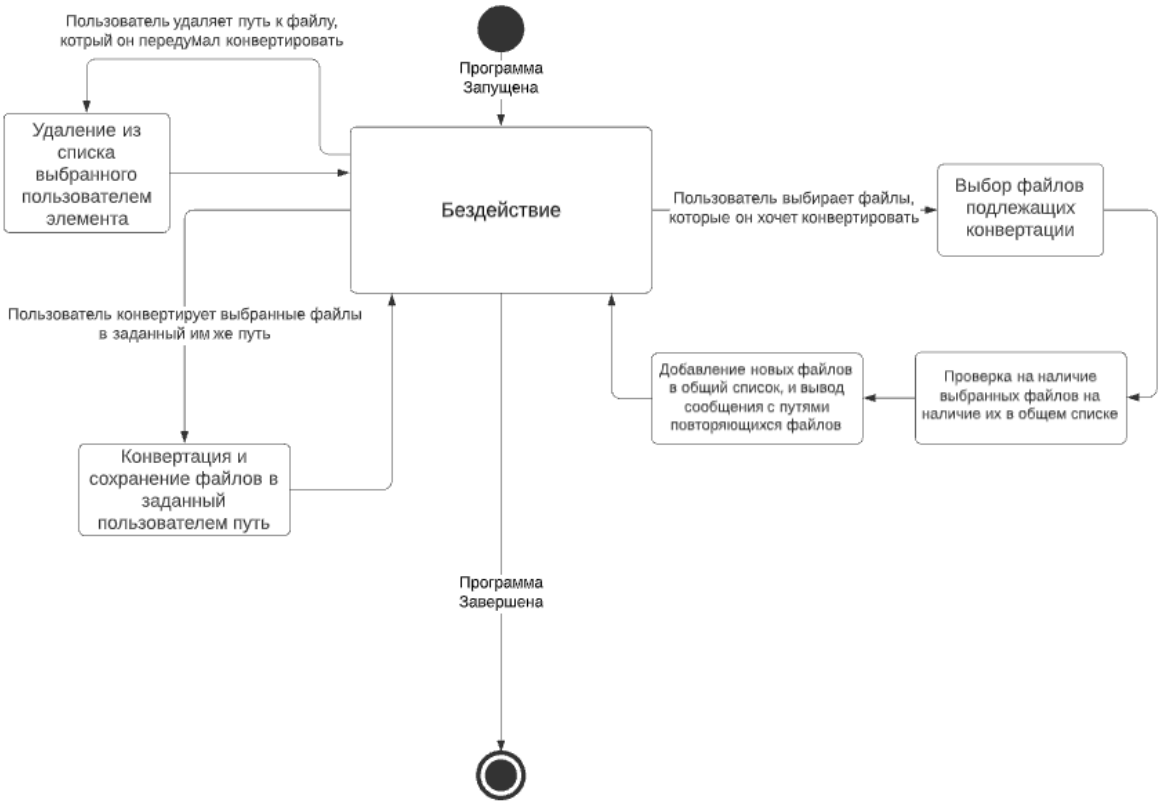

Рисунок 2 – Диаграмма состояний

Данная диаграмма была построена при помощи ресурса [6].

4.3 Диаграмма классов

Диаграмма классов – демонстрирует общую структуру иерархии классов системы, их кооперацией, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними.

Диаграмма классов изображена на рисунке 3. На данной диаграмме (рисунок 3) расписаны все возможные классы и методы, реализованные в программе.

Класс Program – это основной класс, в котором находится метод Main – точка входа в программу.

Класс GeneralForm наследуется от Form. GeneralForm использует структуру List, для хранения списка путей на исходные файлы. GeneralForm содержит методы для обработки событий, таких как нажатие на кнопку. А

также методы для обновления данных таблицы ShowList(), и кнопки добавления исходных изображений, редактирования списка путей к исходным файлам и удаления и сохранения изображений.

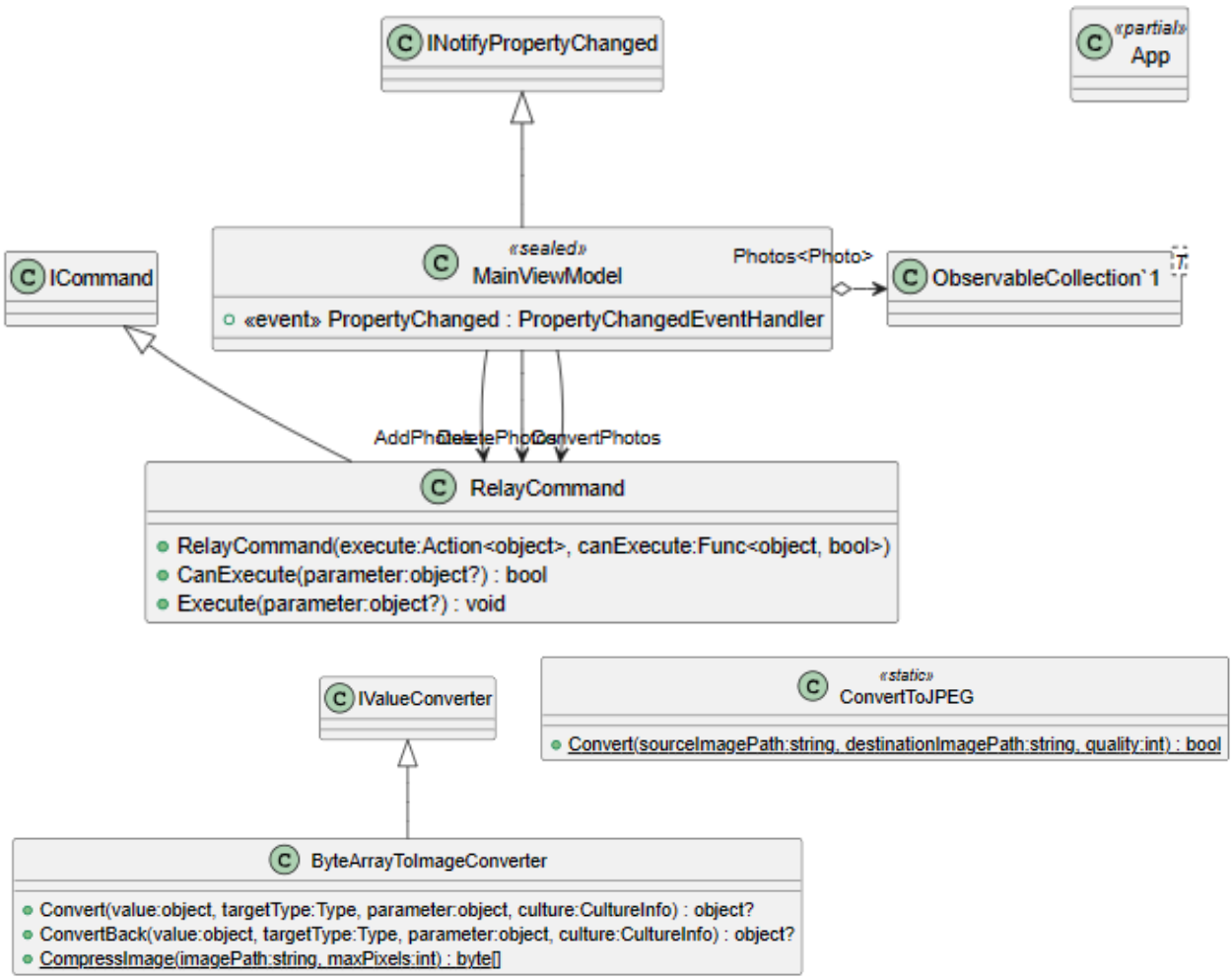

Рисунок 3 – Диаграмма классов

Данная диаграмма была построена при помощи ресурса [5].

4.3 Проектирование интерфейса приложения

Для интуитивно понятного взаимодействия пользователя с программой необходимо грамотно спроектировать интерфейс приложения. Макет интерфейса программы изображен на рисунке 4. На макете изображено в

центральной части, заполняющей практически всю форму, список ссылок на исходные файлы. Под ним три кнопки: кнопка добавления, кнопка редактирования и кнопка сохранения. После нажатия на кнопку добавления, пользователь выбирает нужные ему файлы, и ссылки на них тут же добавляются в список, находящийся в центральной части формы. При нажатии кнопки удаления из общего списка удалится выбранная строка. При нажатии на кнопку сохранения появляется окно для выбора пути, куда будут конвертированы и сохранены изображений.

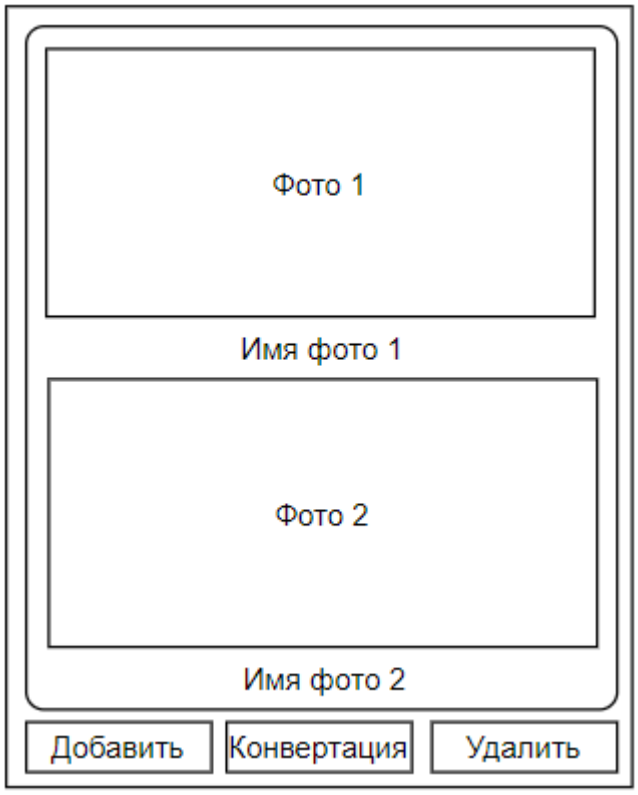

Рисунок 4 – Макет интерфейса программы

5 Реализация программного продукта

5.1 Построение интерфейса приложения

Для построения интерфейса приложения использовалась технология WPF (Windows Presentation Foundation) встроенная в интегрированную среду разработки. Конечный вариант пользовательского интерфейса в конструкторе изображен на рисунке 5-8. В качестве таблицы был взят элемент ListView. Для кнопок был взят элемент Button.

На данном интерфейсе (рисунок 5) существует две области: таблица со списком выбранных изображений и три кнопки: «Добавить», «Конвертировать», «Удалить выбранные».

При нажатии кнопки «Добавить» открывается форма для выбора файлов изображений, которые надо будет конвертировать (рисунок 6).

При нажатии кнопки «Удалить выбранные» будут удалены выбранные фото из списка.

При нажатии кнопки «Конвертироват» появляется объект класса FolderBrowserDialog, в котором пользователь выбирает папку, куда будут сохранены конвертированные изображения (рисунок 7). После чего высвечивается объект класса MessageBox с текстом о том, в какую папку были сохранены изображения (рисунок 8).

Рисунок 5 – Интерфейс программы

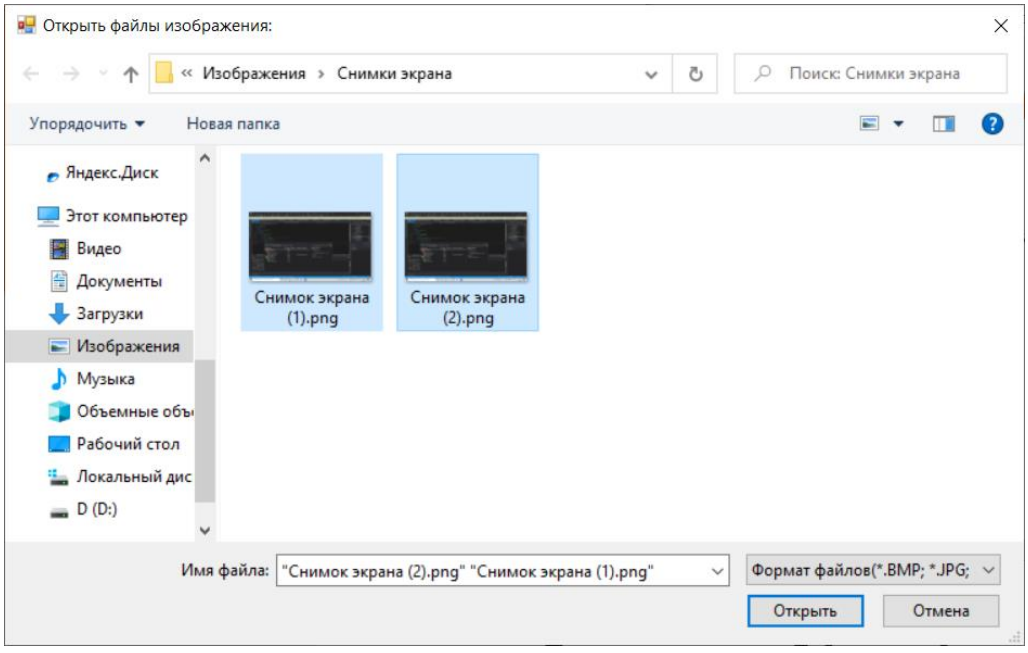

Рисунок 6 – форма для выбора изображений

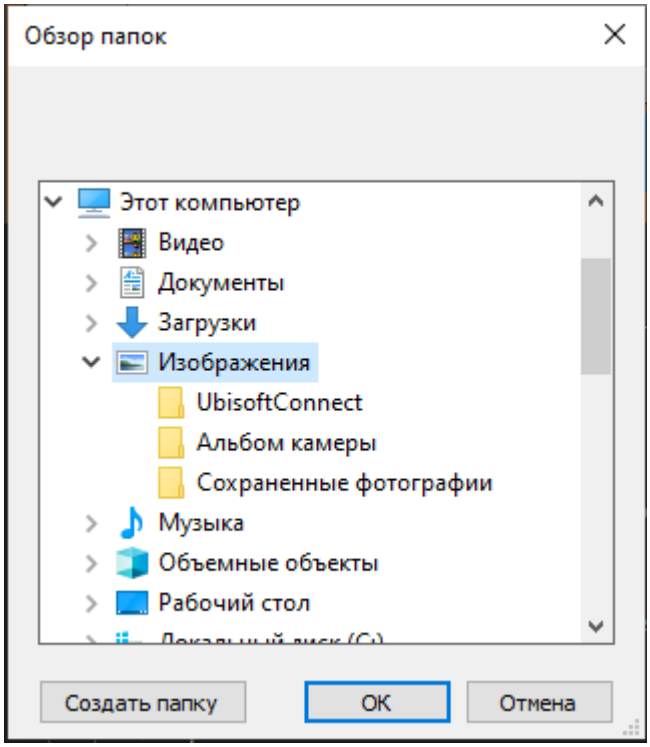

Рисунок 7 – Выбор пути для конвертации

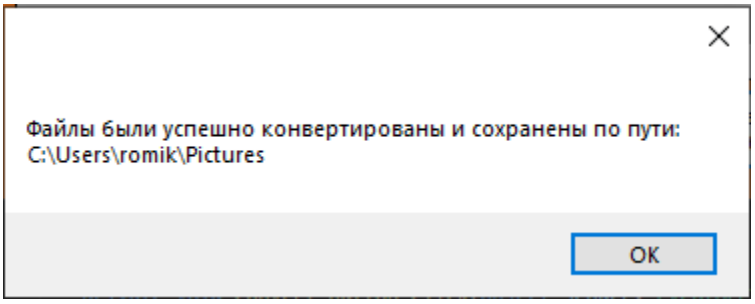

Рисунок 8 – Результат конвертации

5.2 Описание методов программирования

Данная программа разрабатывалась на основе паттерна MVVM (Model-Model-View-ViewModel). Данный паттерн содержит три основных компонента: модель, представление и модель представления. Каждый из них служит определённой цели.

На высоком уровне представление "знает о" модели представления, а модель представления "знает о" модели, но модель не знает о модели

представления, а модель представления не знает о представлении. Таким образом, модель представления изолирует представление от модели и позволяет модели развиваться независимо от представления.

Преимущества использования шаблона MVVM:

- Если существующая реализация модели инкапсулирует существующую бизнес-логику, изменить ее может быть сложно или рискованно. В этом сценарии модель представления выступает в качестве адаптера для классов модели и не позволяет вносить существенные изменения в код модели.
- Разработчики могут создавать модульные тесты для модели представления и модели без использования представления. Модульные тесты для модели представления могут выполнять те же функции, что и в представлении.
- Пользовательский интерфейс приложения можно перепроектировать, не касаясь модели представления и кода модели, при условии, что представление полностью реализовано в XAML или C#. Поэтому новая версия представления должна работать с существующей моделью представления.
- Дизайнеры и разработчики могут работать над компонентами независимо и параллельно во время разработки. Конструкторы могут сосредоточиться на представлении, в то время как разработчики могут работать с моделью представления и компонентами модели.

Для получения нескольких файлов изображения динамически создаётся объект класса OpenFileDialog и изменение необходимых нам параметров у этого объекта (листинг 5.1).

Листинг 5.1 –Динамическое создание объекта класса OpenFileDialog

```
OpenFileDialog openFileDialog = new OpenFileDialog(); // создание
объекта диалоговой формы
openFileDialog.Multiselect = true; // свойство для выбора
нескольких файлов
```

```
openFileDialog.Title = "Открыть файлы изображения:"; //
изменение свойства заголовка
openFileDialog.Filter = "Формат файлов(*.BMP; *.JPG; *.PNG;
*.GIF) | *.BMP; *.JPG; *.PNG; *.GIF| All files (*.*)|*.*"; //
фильтрация файлов
```

Информация по работе с данным классом найдена на электронном ресурсе [7].

Для сохранения ссылок на файлы был создан список, в который добавляются объекты, хранящие всю необходимую информацию о файлах. Также был прописан алгоритм для выявления ссылок, которые уже присутствуют в списке, недопущения их дублирования. (листинг 5.2).

Листинг 5.2 – Заполнение списка с путями к файлам

```
try
{
    if (openFileDialog.ShowDialog() == true) // если была нажата
кнопка ОК
        foreach (string file_path in openFileDialog.FileNames)
// Проверяем каждый выбранный файл по пути
            if (!Photos.Any(photo => photo.Path == file_path))
// Проверяем, существует ли файл в списке объектов Photo
                Photos.Add(new
Photo(Path.GetFileName(file_path), new BitmapImage(new
Uri(file_path)), file_path)); // Добавляем фото, если ещё его
нет
}
catch
{
    MessageBox.Show("Невозможно открыть данный файл", "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}
```

В стабильной постоянной синхронизации помогает паттерн MVVM и реализация интерфейса INotifyPropertyChanged (листинг 5.3).

Листинг 5.3 – Реализация интерфейса INotifyPropertyChanged

```
public event PropertyChangedEventHandler PropertyChanged;

private void OnPropertyChanged([CallerMemberName] string
propertyName = null)
{
```

```
    PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
}

private bool SetField<T>(ref T field, T value,
[CallerMemberName] string propertyName = null)
{
    if (EqualityComparer<T>.Default.Equals(field, value)) return
false;
    field = value;
    OnPropertyChanged(propertyName);
    return true;
}
```

Для открытия формы удаления было прописано событие btDelete_OnClick (листинг 5.4).

Листинг 5.4 – Событие btDelete_OnClick

```
private RelayCommand _deletePhotos;

public RelayCommand DeletePhotos => _deletePhotos ??= new
RelayCommand(btDelete_OnClick);

private void btDelete_OnClick(object o)
{
    ListView photos = (ListView)o;
    List<Photo> photosToRemove = new
List<Photo>(photos.SelectedItems.Cast<Photo>());
    if (photosToRemove != null)
        foreach (Photo photo in photosToRemove)
            Photos.Remove(photo);
}
```

Для реализации конвертирования и сохранения файлов изображения было прописано событие btConvert_OnClick (листинг 5.5).

Листинг 5.5 – Событие btConvert_OnClick

```
private RelayCommand _convertPhotos;

public RelayCommand ConvertPhotos => _convertPhotos ??= new
RelayCommand(btConvert_OnClick);

private void btConvert_OnClick(object o)
{
    try
    {
```

```
        using var dialog = new FolderBrowserDialog();
        DialogResult result = dialog.ShowDialog();
        if (result == DialogResult.OK)
        {
            int i = 0;
            string savePath = dialog.SelectedPath; // Получаем
выбранную папку
            foreach (var item in Photos)
            {
                string filePath = Path.Combine(savePath, $"New
image{++i}.jpg");
                ConvertToJPEG.Convert(item.Path, filePath);
            }

            MessageBox.Show($"Файлы были сохранены по пути
{savePath}");
        }
        else
        {
            MessageBox.Show("Отменено");
        }
    }
    catch
    {
        MessageBox.Show("Операция конвертации сорвалась",
"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
```

Информация о том, как конвертировать и сохранять файлы изображения была найдена на электронных ресурсах [7], [8], [9], [10]

Весь код программы представлен в листинге (приложение А).

Заключение

В результате выполнения курсового проекта по дисциплине «Компьютерные технологии» было спроектировано и после чего разработано приложение Package Image Converter Jpeg. Приложение позволяет выбирать файлы для конвертации, конвертировать их и сохранять в выбранной пользователем директории.

В ходе работы были выполнены следующие задачи:

– произведено изучение предметной области, и проведён анализ существующих расширений;

– было подробно изучен формат JPG;

– спроектирован программный продукт;

– реализован программный продукт;

– составлена пояснительная записка проекта.

Благодаря детальному разбору проекта при помощи диаграмм проектирования, полученных в процессе разработки, можно с уверенностью сказать, что полученное ПО полностью способно выполнять задуманные задачи, полностью соответствуя современным стандартам ПО.

Были получены важные знания и практические навыки как в области использования объектно-ориентированных языков программирования в целом, так и в области построения диаграмм проектирования, отображающих поведение различных организационных структур.

Список используемых источников

1. Графические форматы [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Графические_форматы (дата обращения 18.12.2021).
2. Растровая графика [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Растровая_графика (дата обращения 18.12.2021).
3. Чем отличается Jpg от Jpeg на практике [Электронный ресурс] URL: https://system-blog.ru/chem-otlichaetsya-jpg-ot-jpeg (дата обращения 18.12.2021).
4. JPEG [Электронный ресурс] https://ru.wikipedia.org/wiki/JPEG (дата обращения 18.12.2021).
5. Flowchart Maker & Online Diagram Software [Электронный ресурс] URL: https://rt.draw.io/ (дата обращения 18.12.2021)
6. Lucid [Электронный ресурс] URL: https://lucid.app (дата обращения 18.12.2021).
7. FileStream. Чтение и запись файла [Электронный ресурс] URL: https://metanit.com/sharp/tutorial/5.4.php (дата обращения 18.12.2021).
8. Чтение и запись текстовых файлов. StreamReader и StreamWriter [Электронный ресурс] URL: https://metanit.com/sharp/tutorial/5.5.php (дата обращения 18.12.2021).
9. Работа с файлами. Классы File и FileInfo [Электронный ресурс] URL: https://metanit.com/sharp/tutorial/5.3.php (дата обращения 18.12.2021).
10. Image и работа с изображениями [Электронный ресурс] URL: https://metanit.com/sharp/tutorial/5.3.php (дата обращения 18.12.2021).

Приложение А

Листинг

А.1 Файл MainWindow.xaml

```
<Window x:Class="PackageImageConverterJpeg.View.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        xmlns:app="clr-namespace:PackageImageConverterJpeg"
        xmlns:viewModel="clr-
namespace:PackageImageConverterJpeg.ViewModel"
        mc:Ignorable="d"
        Title="Package Image Converter to Jpeg" Height="500"
Width="450" Background="DarkBlue">
    <Window.Resources>
        <app:ByteArrayToImageConverter
x:Key="ByteArrayToImageConverter" />
        <viewModel:MainViewModel x:Key="ViewModel" />
        <Style x:Key="StandartFont">
            <Setter Property="Control.Foreground" Value="White"
/>
            <Setter Property="Control.HorizontalAlignment"
Value="Center" />
            <Setter Property="Control.VerticalAlignment"
Value="Center" />
            <Setter Property="Control.FontSize" Value="12" />
            <Setter Property="Control.Margin" Value="5, 0" />
        </Style>
        <Style x:Key="StandartButton">
            <Setter Property="Control.Height" Value="25" />
            <Setter Property="Control.Width" Value="115" />
        </Style>
    </Window.Resources>
    <Window.DataContext>
        <viewModel:MainViewModel />
    </Window.DataContext>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="20*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>

        <Grid Grid.Row="0" Grid.Column="1">
            <StackPanel>
```

```
                <GroupBox Grid.Column="0" Grid.ColumnSpan="2"
Margin="10" Header="Таблица хранимых единиц" Foreground="White"
Height="400">
                    <ListView Name="PhotosListView"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch"
ItemsSource="{Binding Photos}">
                        <ListView.View>
                            <GridView>
                                <!-- Создаем столбец для
изображений и подписей -->
                                <GridViewColumn
Header="Изображения" Width="370">

<GridViewColumn.CellTemplate>
                                        <DataTemplate>
                                            <StackPanel
Orientation="Vertical" HorizontalAlignment="Center">
                                                <!-- Изображение
-->
                                                <Image
Source="{Binding ImageSource}" Height="250"
HorizontalAlignment="Center"/>
                                                <!-- Подпись
снизу -->
                                                <TextBlock
Text="{Binding Name}" VerticalAlignment="Center"
HorizontalAlignment="Center"/>
                                            </StackPanel>
                                        </DataTemplate>

</GridViewColumn.CellTemplate>
                                </GridViewColumn>
                            </GridView>
                        </ListView.View>
                    </ListView>
                </GroupBox>
                <Grid>
                    <Button Name="btAdd"
HorizontalAlignment="Left"  Style="{StaticResource
StandartButton }"
                            Margin="20,0,0,0" Content="Добавить"
                            Command="{Binding AddPhotos }"/>
                    <Button Name="btConvert"
HorizontalAlignment="Center"  Style="{StaticResource
StandartButton }"
                            Content="Конветировать"
                            Command="{Binding ConvertPhotos }"
/>
                    <Button Name="btDelete"
HorizontalAlignment="Right" Style="{StaticResource
StandartButton }"
                            Margin="20,10" Content="Удалить
выбранные"
```

```
                            Command="{Binding DeletePhotos }"
                            CommandParameter="{Binding
ElementName=PhotosListView}" />
                </Grid>
            </StackPanel>
        </Grid>
    </Grid>
</Window>
```

A.2 Файл MainWindow.xaml.cs

```
using PackageImageConverterJpeg.ViewModel;

namespace PackageImageConverterJpeg.View
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow
    {
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

A.3 MainViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.IO;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Windows;
using System.Windows.Forms;
using System.Windows.Media.Imaging;
using PackageImageConverterJpeg.Model;
using MessageBox = System.Windows.MessageBox;
using OpenFileDialog = Microsoft.Win32.OpenFileDialog;
using ListView = System.Windows.Controls.ListView;

namespace PackageImageConverterJpeg.ViewModel;

public sealed class MainViewModel : INotifyPropertyChanged
{
    public ObservableCollection<Photo> Photos { get; set; } =
new ObservableCollection<Photo>();

    private RelayCommand _addPhotos;
```

```
    private RelayCommand _convertPhotos;
    private RelayCommand _deletePhotos;

    public RelayCommand AddPhotos => _addPhotos ??= new
RelayCommand(btAdd_OnClick);
    public RelayCommand ConvertPhotos => _convertPhotos ??= new
RelayCommand(btConvert_OnClick);
    public RelayCommand DeletePhotos => _deletePhotos ??= new
RelayCommand(btDelete_OnClick);

    private void btAdd_OnClick(object o)
    {
        // запуск формы для выбора файлов
        OpenFileDialog openFileDialog = new OpenFileDialog(); //
создание объекта диалоговой формы
        openFileDialog.Multiselect = true; // свойство для
выбора нескольких файлов
        openFileDialog.Title = "Открыть файлы изображения:"; //
изменение свойства заголовка
        openFileDialog.Filter =
            "Формат файлов(*.BMP; *.JPG; *.PNG; *.GIF) | *.BMP;
*.JPG; *.PNG; *.GIF| All files (*.*)|*.*"; // фильтрация файлов

        try
        {
            if (openFileDialog.ShowDialog() == true) // если
была нажата кнопка ОК
                foreach (string file_path in
openFileDialog.FileNames) // Проверяем каждый выбранный файл по
пути
                    if (!Photos.Any(photo => photo.Path ==
file_path)) // Проверяем, существует ли файл в списке объектов
Photo
                        Photos.Add(new
Photo(Path.GetFileName(file_path), new BitmapImage(new
Uri(file_path)), file_path)); // Добавляем фото, если ещё его
нет
        }
        catch
        {
            MessageBox.Show("Невозможно открыть данный файл",
"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    private void btConvert_OnClick(object o)
    {
        try
        {
            using var dialog = new FolderBrowserDialog();
            DialogResult result = dialog.ShowDialog();
            if (result == DialogResult.OK)
            {
```

```
                int i = 0;
                string savePath = dialog.SelectedPath; //
Получаем выбранную папку
                foreach (var item in Photos)
                {
                    string filePath = Path.Combine(savePath,
$"New image{++i}.jpg");
                    ConvertToJPEG.Convert(item.Path, filePath);
                }

                MessageBox.Show($"Файлы были сохранены по пути
{savePath}");
            }
            else
            {
                MessageBox.Show("Отменено");
            }
        }
        catch
        {
            MessageBox.Show("Операция конвертации сорвалась",
"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    private void btDelete_OnClick(object o)
    {
        ListView photos = (ListView)o;
        List<Photo> photosToRemove = new
List<Photo>(photos.SelectedItems.Cast<Photo>());
        if (photosToRemove != null)
            foreach (Photo photo in photosToRemove)
                Photos.Remove(photo);
    }

    public event PropertyChangedEventHandler PropertyChanged;

    private void OnPropertyChanged([CallerMemberName] string
propertyName = null)
    {
        PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
    }

    private bool SetField<T>(ref T field, T value,
[CallerMemberName] string propertyName = null)
    {
        if (EqualityComparer<T>.Default.Equals(field, value))
return false;
        field = value;
        OnPropertyChanged(propertyName);
        return true;
```

```
    }
}
```

A.4 ByteArrayToImageConverter.cs

```
using System;
using System.Globalization;
using System.IO;
using System.Windows.Data;
using System.Windows.Media;
using System.Windows.Media.Imaging;

namespace PackageImageConverterJpeg;

public class ByteArrayToImageConverter : IValueConverter
{
    public object? Convert(object value, Type targetType, object
parameter, CultureInfo culture)
    {
        var byteArray = value as byte[];

        if (byteArray == null || byteArray.Length == 0)
            return null;

        var image = new BitmapImage();
        using (var mem = new MemoryStream(byteArray))
        {
            mem.Position = 0;
            image.BeginInit();
            image.CreateOptions =
BitmapCreateOptions.PreservePixelFormat;
            image.CacheOption = BitmapCacheOption.OnLoad;
            image.UriSource = null;
            image.StreamSource = mem;
            image.EndInit();
        }

        image.Freeze();

        return image;
    }

    public object? ConvertBack(object value, Type targetType,
object parameter, CultureInfo culture)
    {
        var image = value as BitmapImage;

        if (image == null)
            return null;

        using var mem = new MemoryStream();
        var encoder = new PngBitmapEncoder();
        encoder.Frames.Add(BitmapFrame.Create(image));
```

```
        encoder.Save(mem);
        return mem.ToArray();
    }

    public static byte[] CompressImage(string imagePath, int
maxPixels)
    {
        var bitmap = new BitmapImage(new Uri(imagePath));

        var originalWidth = bitmap.PixelWidth;
        var originalHeight = bitmap.PixelHeight;

        if (originalWidth * originalHeight <= maxPixels)
            // Изображение уже удовлетворяет заданному
количеству пикселей, возвращаем оригинальные данные
            return File.ReadAllBytes(imagePath);

        var targetWidth = (int)Math.Sqrt(maxPixels *
(originalWidth / (double)originalHeight));
        var targetHeight = (int)(maxPixels /
(double)targetWidth);

        var transformedBitmap = new TransformedBitmap(bitmap,
            new ScaleTransform(targetWidth /
(double)originalWidth, targetHeight / (double)originalHeight));

        BitmapEncoder encoder = new JpegBitmapEncoder();

encoder.Frames.Add(BitmapFrame.Create(transformedBitmap));

        using var memoryStream = new MemoryStream();
        encoder.Save(memoryStream);
        return memoryStream.ToArray();
    }
}
```

A.5 ConvertToJPEG.cs

```
using System;
using System.IO;
using System.Windows.Media.Imaging;

namespace PackageImageConverterJpeg;

public static class ConvertToJPEG
{
    public static bool Convert(string sourceImagePath, string
destinationImagePath, int quality = 90)
    {
        try
        {
            // Открываем исходное изображение
            BitmapImage sourceImage = new BitmapImage(new
```

```
Uri(sourceImagePath));

            // Создаем кодер JPEG
            JpegBitmapEncoder jpegEncoder = new
JpegBitmapEncoder();
            jpegEncoder.QualityLevel = quality; // Устанавливаем
качество сжатия (0-100)

            // Конвертируем изображение в JPEG

jpegEncoder.Frames.Add(BitmapFrame.Create(sourceImage));

            // Сохраняем конвертированное изображение
            using (FileStream stream = new
FileStream(destinationImagePath, FileMode.Create))
            {
                jpegEncoder.Save(stream);
            }

            return true;
        }
        catch (Exception)
        {
            // Если произошла ошибка при конвертации, возвращаем
false
            return false;
        }
    }
}
```

A.6 RelayCommand.cs

```
using System;
using System.Windows.Input;

namespace PackageImageConverterJpeg;

public class RelayCommand : ICommand
{
    private Action<object> execute;
    private Func<object, bool> canExecute;

    public event EventHandler CanExecuteChanged
    {
        add { CommandManager.RequerySuggested += value;}
        remove { CommandManager.RequerySuggested -= value; }
    }

    public RelayCommand(Action<object> execute, Func<object,
bool> canExecute = null)
    {
        this.execute = execute;
        this.canExecute = canExecute;
```

```
    }

    public bool CanExecute(object? parameter)
    {
        return this.canExecute == null ||
this.canExecute(parameter);
    }

    public void Execute(object? parameter)
    {
        this.execute(parameter);
    }
}
```

A.7 Photo.cs

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Windows.Media.Imaging;

namespace PackageImageConverterJpeg.Model;

public class Photo : INotifyPropertyChanged
{
    public Photo(string name, BitmapImage imageSource, string
path)
    {
        Name = name;
        ImageSource = imageSource;
        Path = path;
    }

    private string _name;
    private BitmapImage _imageSource;
    private string _path;

    public string Name
    {
        get => _name;
        set
        {
            if (value == _name) return;
            _name = value;
            OnPropertyChanged();
        }
    }

    public BitmapImage ImageSource
    {
        get => _imageSource;
        set
        {
```

```
            if (value == _imageSource) return;
            _imageSource = value;
            OnPropertyChanged();
        }
    }

    public string Path
    {
        get => _path;
        set
        {
            if (value == _path) return;
            _path = value;
            OnPropertyChanged();
        }
    }

    public event PropertyChangedEventHandler PropertyChanged;

    protected virtual void OnPropertyChanged([CallerMemberName]
string propertyName = null)
    {
        PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
    }

    protected bool SetField<T>(ref T field, T value,
[CallerMemberName] string propertyName = null)
    {
        if (EqualityComparer<T>.Default.Equals(field, value))
return false;
        field = value;
        OnPropertyChanged(propertyName);
        return true;
    }
}
```

Приложение Б

Проверка на антиплагиат