МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет           Информационных Технологий  

Кафедра               Информационных систем и технологий

Специальность 1-40 05 01 Информационные системы и технологии          

Направление специализация 1-40 05 0103 Информационные системы и технологии (издательско-полиграфический комплекс)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

КУРСОВОГО ПРОЕКТА:

по дисциплине **«**Объектно-ориентированное программирование»

Тема «Программное средство «Файловый менеджер»»

Исполнитель

студент 2 курса группы 1 Муругин В.А.

(Ф. И. О.)

Руководитель работы                                  ассистент Панченко О. Л.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующий кафедрой ПИ

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Н.В. Пацей

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Объектно-ориентированное программирование"

|  |  |
| --- | --- |
| Специальность: 1-40 05 01-03 Информационные системы и технологии  Студент: Муругин В.А. | Группа:\_\_1\_\_ |
| **Тема: Программное средство «Файловый менеджер»** | |

**1. Срок сдачи студентом законченной работы**: "20 мая 2021 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Создание, открытие файлов и папок
* Перемещение файлов и папок
* Переименование файлов и папок
* Копирование файлов и папок
* Удаление файлов и папок
* Архивирование файлов и папок
* Поиск файлов и папок

**2.2.** При выполнении курсового проекта необходимо использовать принципы проектирования ООП. Приложение разрабатывается под ОС Windows и представляет собой настольное приложение (desktop). Отображение, бизнес логика должны быть максимально независимы друг от друга для возможности расширения. Диаграммы вариантов использования, классов реализации задачи, взаимодействия разработать на основе UML. Язык разработки проекта – C#. Управление программой должно быть интуитивно понятным и удобным. При разработке использовать несколько наиболее подходящих шаблонов проектирования ПО.

**3. Содержание расчетно-пояснительной записки**

(перечень вопросов, подлежащих разработке)

* Введение
* Постановка задачи и обзор литературы (алгоритмы решения, обзор прототипов, актуальность задачи)
* Проектирование архитектуры проекта (структура модулей, классов).
* Разработка функциональной модели и модели данных ПС (выполняемые функции)
* Тестирование
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненной курсовой работы:**

* + Теоретическая часть курсового проекта должны быть представлены в формате docx. Оформление записки должно быть согласно выданным правилам.
  + Листинги программы представляются частично в приложении.
  + Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить диск, указанный преподавателем.

***Календарный план***

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2021 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2021 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2021 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 2.04.2021 |  |
| 5 | Кодирование программного средства | 23.04.2021 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2021 |  |
| 7 | Оформление пояснительной записки | 7.05.2021 |  |
| 9 | Сдача проекта | 20.05.2021 |  |

**5. Дата выдачи задания \_\_\_\_**12.02.2021**\_\_\_\_**

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *О.Л. Панченко*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

Содержание

[ВВЕДЕНИЕ 5](#_Toc73081531)

[1 Аналитический обзор литературы 6](#_Toc73081532)

[1.1 Анализ прототипов 6](#_Toc73081533)

[1.2 Формирование требований к проектируемому программному средству 8](#_Toc73081534)

[2 Анализ требований к программному средству и разработка функциональных требований 9](#_Toc73081535)

[3 Проектирование программного средства 10](#_Toc73081536)

[3.1 Проектирование структуры проекта 10](#_Toc73081537)

[3.2 Проектирование дизайна приложения 12](#_Toc73081538)

[4 Создание программного средства 14](#_Toc73081539)

[4.1 Описание функций, реализуемых программой 14](#_Toc73081540)

[4.2 Описание вспомогательных классов 20](#_Toc73081541)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 23](#_Toc73081542)

[6 Руководство по установке и использованию 24](#_Toc73081543)

[ЗАКЛЮЧЕНИЕ 27](#_Toc73081544)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 28](#_Toc73081545)

# ВВЕДЕНИЕ

Файловые менеджеры – это класс программ, служащих для всевозможной работы с файлами, включая поддержку таких операций, как создание, редактирование, копирование и удаление, для обеспечения гибкого и удобного запуска других программ, предназначенных для работы с этими файлами. Нередко они сопровождаются дополнительными утилитами, облегчающими жизнь пользователя. Для многих пользователей любимый файловый менеджер зачастую выступает в роли оболочки, заменяя часть стандартных средств работы с файлами, имеющихся в операционной системе.

Надо отметить, что за последние годы изменилась и потребность в подобных программах. Если раньше файловые менеджеры были чуть ли не единственной возможностью удобно работать с операционной системой, то постепенно, благодаря стараниям Microsoft, большинству пользователей оказалось достаточно тех средств, которые доступны в Windows. Другая же часть пользователей, потребности которой продолжали возрастать, стала, наоборот, предъявлять новые, повышенные требования к функционалу и внешнему виду файловых менеджеров.

Менеджеры бывают двух типов: навигационные и двухпанельные. Классическая утилита Windows «Проводник» – пример навигационного менеджера. Сторонние утилиты обычно двухпанельные, хотя существуют и интерфейсы, которые не разделены на несколько частей или же подразумевают включение одной или нескольких панелей в настройках.

Можно с уверенностью утверждать, что файловые менеджеры сегодня превратились в важный инструмент профессиональных пользователей, обеспечивающий высокую эффективность работы. А какую конкретно программу этого класса использовать, каждый пользователь решает сам. Одним важнее тесная интеграция всех модулей в удобном современном интерфейсе, другим – наличие дополнительных плагинов на все случаи жизни.

Впечатление от графического интерфейса операционной системы во многом определяется удобством файлового менеджера. Именно с его помощью выполняется поиск, копирование и перемещение, переименование и удаление файлов, доступ к архивам. Не случайно для многих любимой оболочкой остается файловый менеджер с классическим двухоконным интерфейсом. Необходимость двух панелей продиктована удобством выполнения двуместных операций, а наличие дополнительных функций наподобие просмотра текстовых и графических файлов, работы с FTP-серверами, непосредственного доступа к архивам распространенных форматов обеспечивает современному файловому менеджеру законное место в обязательном наборе программ.

# Аналитический обзор литературы

Перед определением окончательных целей курсового проекта необходимо проанализировать прототипы программных средств других производителей приложений выбранной темы.

## Анализ прототипов

Первый аналог – *Explorer++*. Полностью портативная альтернатива «Проводнику», которая работает без установки как из любой папки, так и с внешнего носителя. Вкладочный интерфейс, древовидная панель папок, выделение по шаблону, неплохой поиск, множество параметров сортировки файлов.

Плюсы:

– быстрота;

– небольшой размер;

– простой и понятный интерфейс;

– наличие вкладок.

Минусы:

– по сравнению с другими менеджерами имеет меньший функционал.

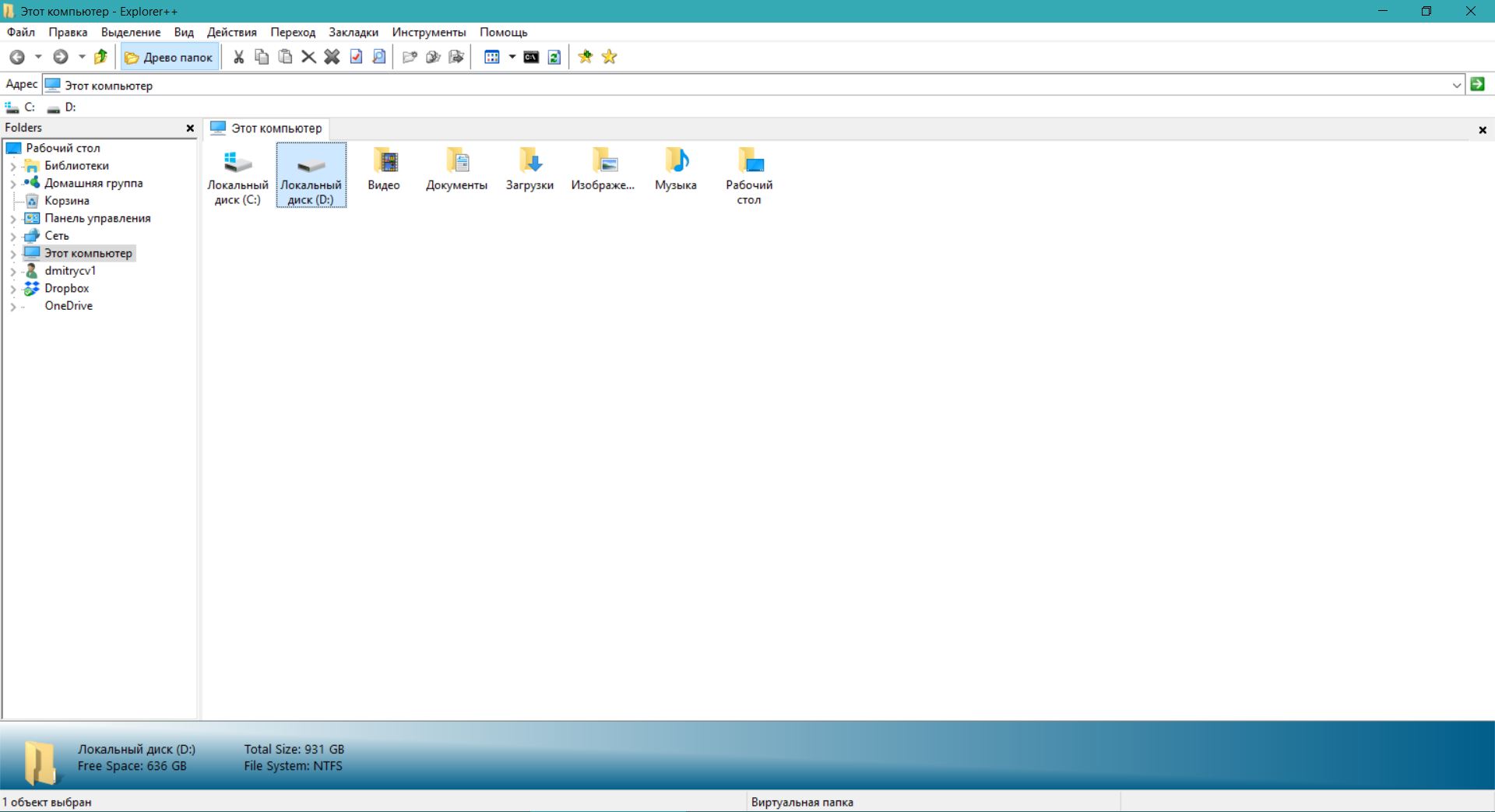


Рисунок 1.1 – Интерфейс Explorer++

Второй аналог – XYplorer. Данный файловый менеджер похож на предыдущий (Explorer++), но разработан как для опытных, так и для начинающих пользователей. Начиная с пользовательского интерфейса, вы получаете удобные кнопки и функции везде. Так же, как и в Explorer++, вы получаете опцию вкладок для интерфейса.

Плюсы:

– наличие вкладок;

– удобство использования.

Минусы:

– непривычный и сложный интерфейс;

– не всегда стабильная работа.

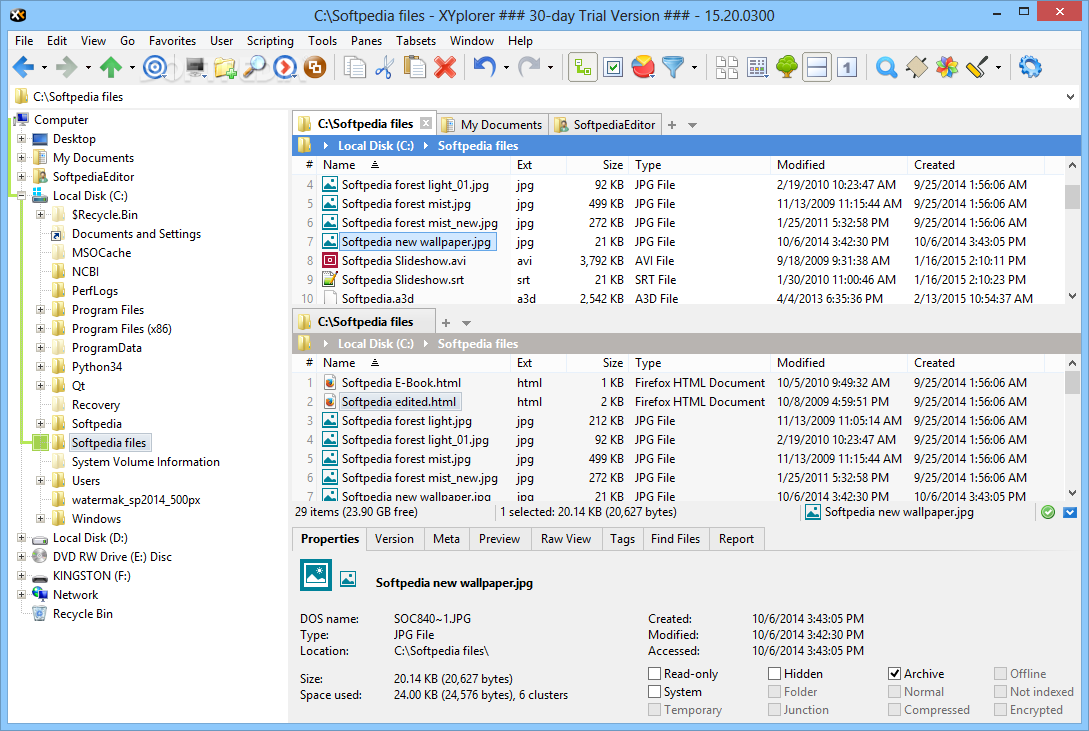


Рисунок 1.2 – Интерфейс XYplorer

Третий аналог – *Total Commander*. Это ещё один сторонний файловый менеджер для Windows, который существует с начала 90-х годов. Он имеет минималистский интерфейс с двумя панелями, который прост в освоении для обычного пользователя. Total Commander нельзя назвать самой функциональной, самой красивой или самой быстрой программой данного класса. Причина популярности файлового менеджера кроется в сбалансированности всех составляющих.

Плюсы:

– работа с архивами;

– широкий функционал с большим количеством дополнительных возможностей;

– наличие вкладочного интерфейса, как в браузере.

Минусы:

– платное распространение;

– не всегда удобный интерфейс.

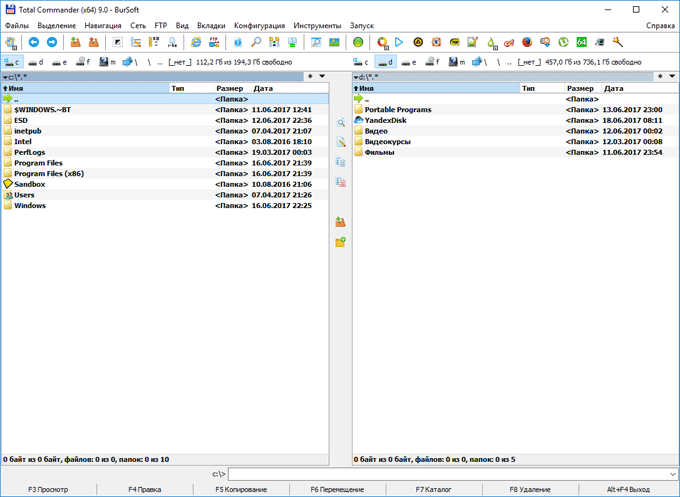


Рисунок 1.3 – Интерфейс Total Commander

## Формирование требований к проектируемому программному средству

В курсовом проекте должна иметься возможность с помощью графического интерфейса манипулировать файлами на компьютере.

Приложение должно содержать следующие функции:

– создание файлов и папок;

– открытие файлов и папок;

– удаление файлов и папок;

– создание архивов (сжатие файлов и папок);

– копирование в буфер обмена файлов и папок;

– вставка из буфера обмена файлов и папок;

– переименование файлов и папок;

– поиск файлов и папок по имени;

– перемещение папок и файлов в другие папки.

# Анализ требований к программному средству и разработка функциональных требований

При разработке курсового проекта было принято решение реализовать вкладочный интерфейс, на подобие браузера, где в каждой вкладке может быть открыта любая папка. Будет возможность добавлять и удалять вкладки. UML-диаграмма вариантов использования программного средства представлена на рисунке 2.1.

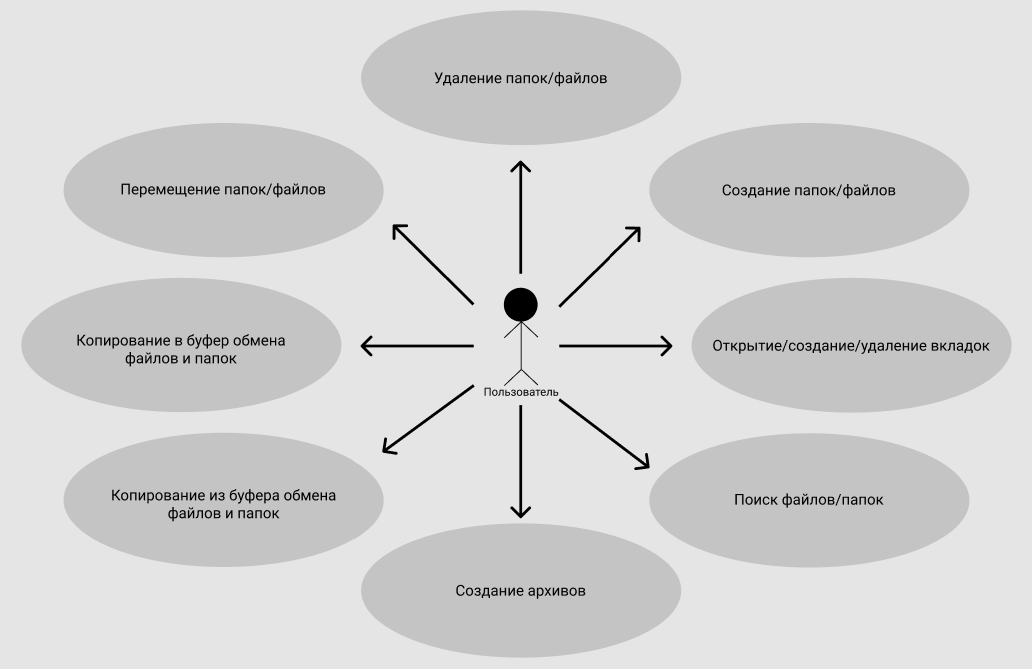


Рисунок 2.1 – UML-диаграмма вариантов использования

В дальнейшем данные функции будут проверены на правильность реализации с помощью специально разработанных тестов.

# Проектирование программного средства

## Проектирование структуры проекта

На рисунке 3.1 представлена структура проекта. Программное средство реализует паттерн MVVM(Model-View-ViewModel), который позволяет определить особую логику взаимодействия визуальной части приложения и бизнес-логики.

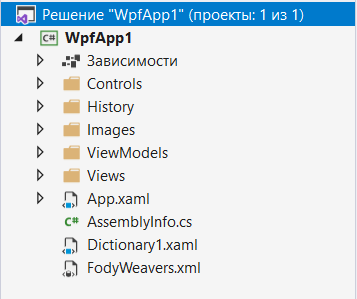


Рисунок 3.1 – Структура проекта

Рассмотрим структуру и назначение папок и файлов:

– Controls –– содержит пользовательские элементы управления;

– History –– содержит интерфейс и класс, реализующие логику перемещения по папкам «вперёд» и «назад» на основе истории посещений, которая работает как двусвязный список;

– Images – содержит изображения, которые применяются в приложении;

– ViewModels – содержит модель представления, которая связывает модель и представление через механизм привязки;

– Views – содержит представления, определяющие внешний вид приложения;

– App.xaml – Файл класс Application;

– Dictionary1.xaml – словарь ресурсов приложения;

– FodyWeavers.xml – файл из загружаемого пакета, который автоматически добавляет уведомления об изменении свойств ко всем классам, реализующим интерфейс INotifyPropertyChanged.

Диаграмма классов служит для представления системы классов объектно-ориентированного программирования. Диаграмма классов описывает составные части системы и может отражать различные взаимосвязи между ними, а также описывает их строение и типы отношений. На рисунке 3.2 представлена UML-диаграмма классов, созданного приложения.

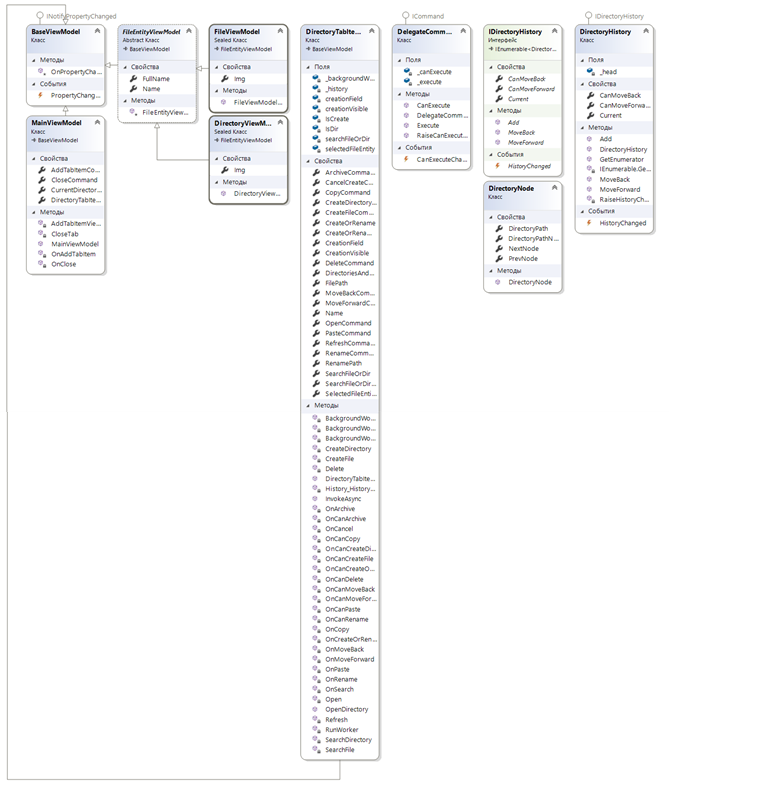


Рисунок 3.2 – Диаграмма классов

Класс BaseViewModel – базовый класс модели представления, реализующий интерфейс INotifyPropertyChanged, от которого наследуются все остальные классы моделей представления.

Класс MainViewModel – модель представления, отвечающая за работу с вкладками приложения, реализует команды создания и удаления вкладки.

Класс FileEntityViewModel – определяет модель представления сущности, с которой может работать файловый менеджер.

Класс FileViewModel – наследник класса FileEntityViewModel, представляет сущность файла.

Класс DirectoryViewModel – наследник класса FileEntityViewModel, представляет сущность папки.

Класс DirectoryTabItemViewModel – модель представления, отвечающая за каждую отдельно взятую вкладку. Хранит в себе список файлов и папок, а также команды, отвечающие за работу с файлами и папками, например, открытие, удаление, создание и т.д.

Класс DelegateCommand – класс, реализующий интерфейс ICommand. Конструктор данного класса принимает в себя метод, отвечающий за выполнение конкретной задачи. Также может принимать метод, проверяющий возможность выполнения команды в определённый момент времени.

Интерфейс IDirectoryHistory – задаёт основной функционал, необходимый для перемещения вперёд/назад по истории посещения папок в конкретной вкладке. История реализована в виде узлов, связанный в двусвязный список.

Класс DirectoryHistory – реализует интерфейс IDirectiryHistory, описывает логику перехода между ранее посещёнными папками.

Класс DirectoryNode – узел в истории посещения папок, содержит ссылки на предыдущий и следующий элемент двусвязного списка истории посещения папок.

## Проектирование дизайна приложения

Основные цвета выбирались исходя из предпочтений пользователей, поэтому было принято решение использовать тёмную тему для приложения. Основными цветами будут выступать оттенки серого. Цветовая схема представлена на рисунке 3.3.

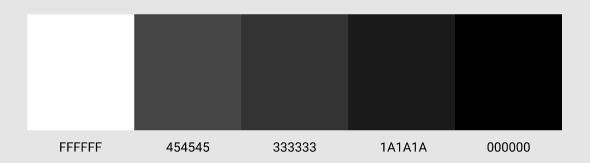


Рисунок 3.3 – Цветовая схема

Была разработана иконка, которая ассоциируется с тематикой приложения. Изображение данной иконки представлено на рисунке 3.4.

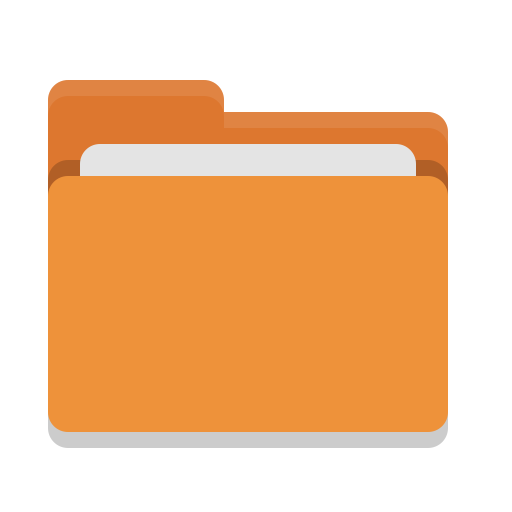


Рисунок 3.4 – Иконка приложения

# Создание программного средства

## Описание функций, реализуемых программой

Рассмотрим основные функции, которые были описаны в требованиях к приложению, а также их реализацию.

Реализация создания папок и файлов указано на рисунке 4.1.

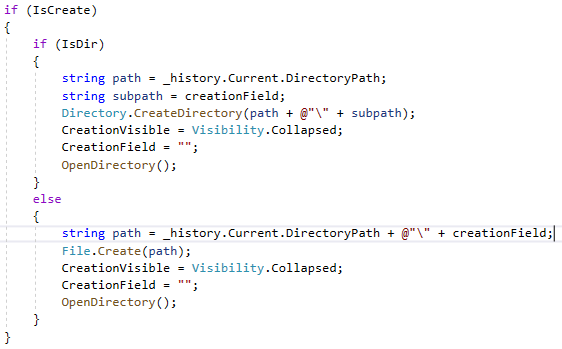


Рисунок 4.1 – Создание папки\файла

Реализация открытия файлов и папок показано на рисунке 4.2.

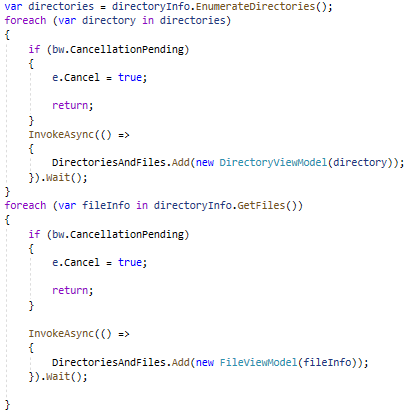


Рисунок 4.2 – Открытие файлов и папок

Так как процесс добавления в коллекцию файлов и папок может оказаться достаточно трудоёмким в случае их большого количества, используется класс BackgroundWorker. Он позволяет выполнять трудоёмкие задачи в отдельном потоке для того, чтобы избежать зависания приложения. Также здесь используется класс Dispatcher для тех же целей. Он представляет службы для управления очередью рабочих элементов для потока.

Реализация перемещения файлов и папок в другую папку показана на рисунке 4.3.

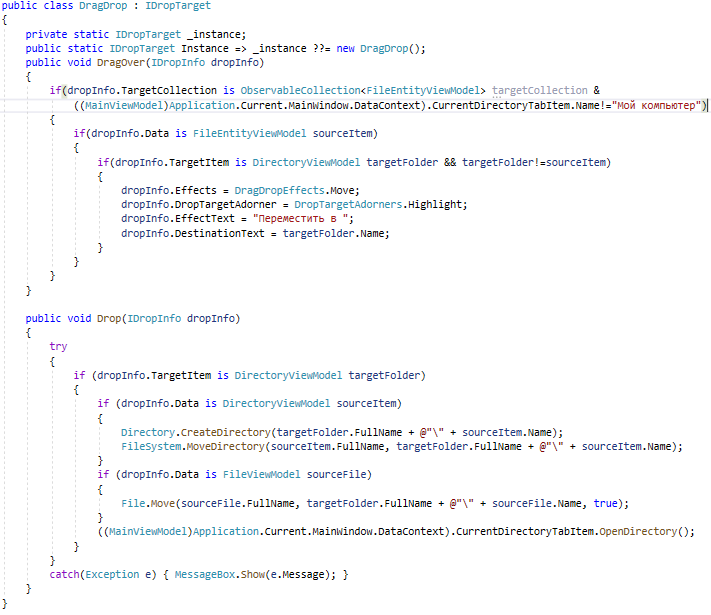


Рисунок 4.3 – Перемещение файлов и папок

Перемещение выполняется с помощью drag-and-drop механизма, который реализуется подключаемой библиотекой GongSolutions.Wpf.DragDrop. Для этого нужно создать класс, который реализует методы DragOver и Drop интерфейса IDropTarget, также этот класс реализует паттерн Singleton. Метод DragOver задаёт правила, по которым можно либо нельзя перетаскивать те или иные объекты на другие. Метод Drop описывает действие, которое выполняется после того, как будет выполнено перетаскивание объекта. В данном случае это будет перемещение файла или папки в другую папку. Также в xaml коде нужно для listbox’а, в который помещаются папки и файлы, указать некоторые свойства, показанные на рисунке 4.4.

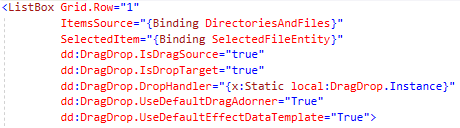


Рисунок 4.4 – Свойства библиотеки gong-wpf-dragdrop

Реализация переименования файлов и папок показана на рисунке 4.5.

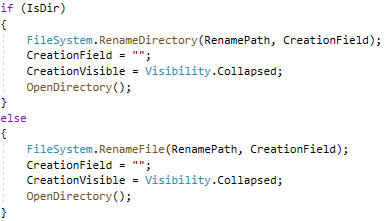


Рисунок 4.5 – Переименование файлов и папок

В данном случае была подключена библиотека Microsoft.VisualBasic.FileIO и использован её статический класс FileSystem для работы с файловой системой. Её использование оправдывается тем, что в этом случае код становится более компактным и читаемым. Используются методы RenameDirectory и RenameFile.

Реализация копирования файлов и папок в буфер обмена показана на рисунке 4.6.

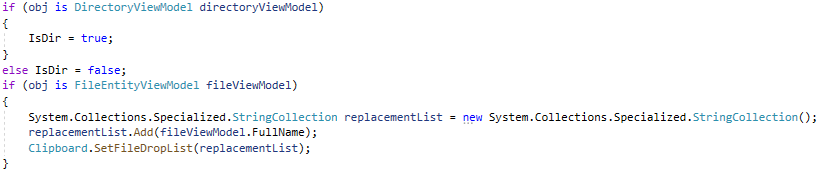


Рисунок 4.6 – Копирование файла либо папки в буфер обмена

Для работы с буфером обмена используется статический класс Clipboard из библиотеки System.Windows.

Реализация извлечения файлов и папок из буфера обмена показана на рисунке 4.7.



Рисунок 4.7 – Извлечение файлов и папок из буфера обмена

Реализация удаления файлов и папок показана на рисунке 4.8.

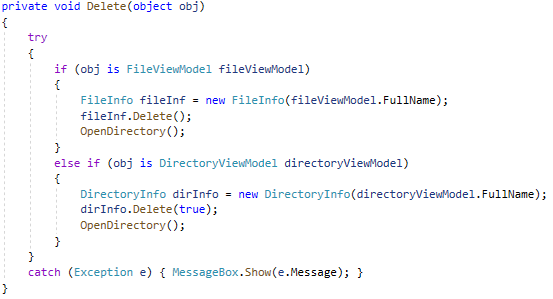


Рисунок 4.8 – Удаление файлов и папок

Реализация архивирования файлов и папок показана на рисунке 4.9.

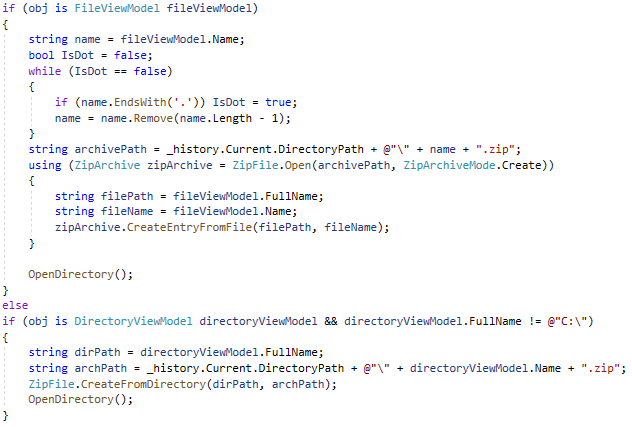


Рисунок 4.9 – Архивирование файлов и папок

Для архивирования используется статический класс ZipFile из библиотеки System.IO.Compression и его методы CreateFromDirectory и CreateFromFile.

Реализация поиска файлов и папок в текущей директории показана на рисунке 4.10.

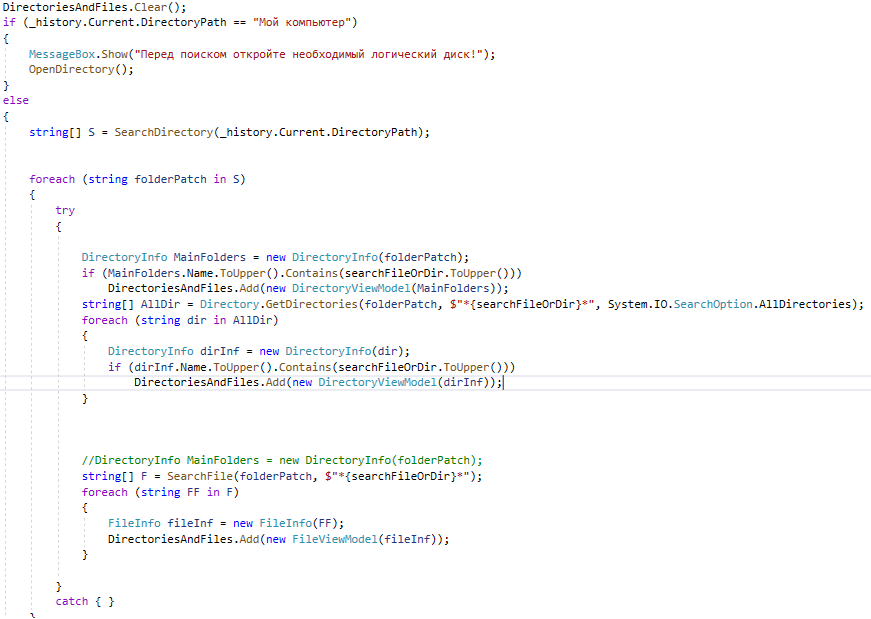


Рисунок 4.10 – Поиск файлов и папок

## Описание вспомогательных классов

Для реализации перехода между ранее посещёнными папками были разработаны некоторые вспомогательные классы. Изначально был создан интерфейс IDirectoryHistory. Код данного интерфейса указан на рисунке 4.11.

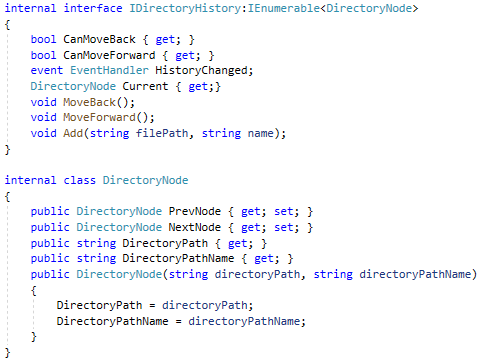
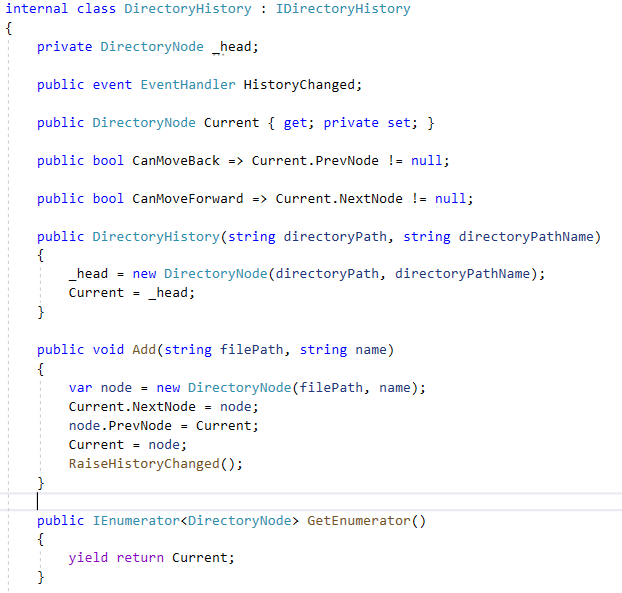


Рисунок 4.11 – Интерфейс IDirectoryHistory

Кроме методов, позволяющих перемещаться между папками данный интерфейс содержит класс DirectoryNode, который представляет собой узел двусвязного списка. Реализацией данного интерфейса является класс DirectoryHistory. Код данного класса показан на рисунке 4.12.



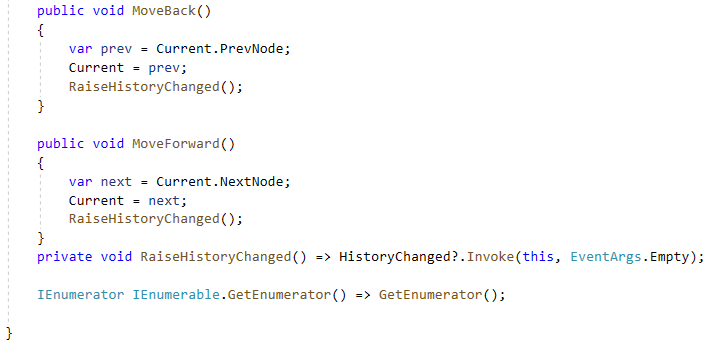


Рисунок 4.12 – Класс DirectoryHistory

# Тестирование, проверка работоспособности и анализ полученных результатов

Программное средство проходило тестирование в три этапа. На первом этапе происходила отладка программы в процессе написания кода, т.е. для правильной работы функций. Второй этап был после завершения основного кода, во время тестирования ПО на работоспособность, и при наличии ошибок, исправлялся. Третий этап включал в себе участие постороннего человека: архив с программой отправлялся добровольцу, а тот запускал приложение следуя инструкции. Таким образом добивалась максимальная работоспособность программы.

После выполнения всех этапов готовое приложение:

**–** не позволяет выполнять операции над системными файлами, которые могут привести к их повреждению;

– не позволяет перемещать папки и файлы в другие файлы;

– не позволяет выполнять команды, если не соблюдены определённые условия для их выполнения. Это достигается блокированием кнопки, отвечающей за данную операцию;

– не позволять выполнять команды над логическими дисками, кроме их открытия.

Примеры результатов данных тестов показаны на рисунке 5.1.

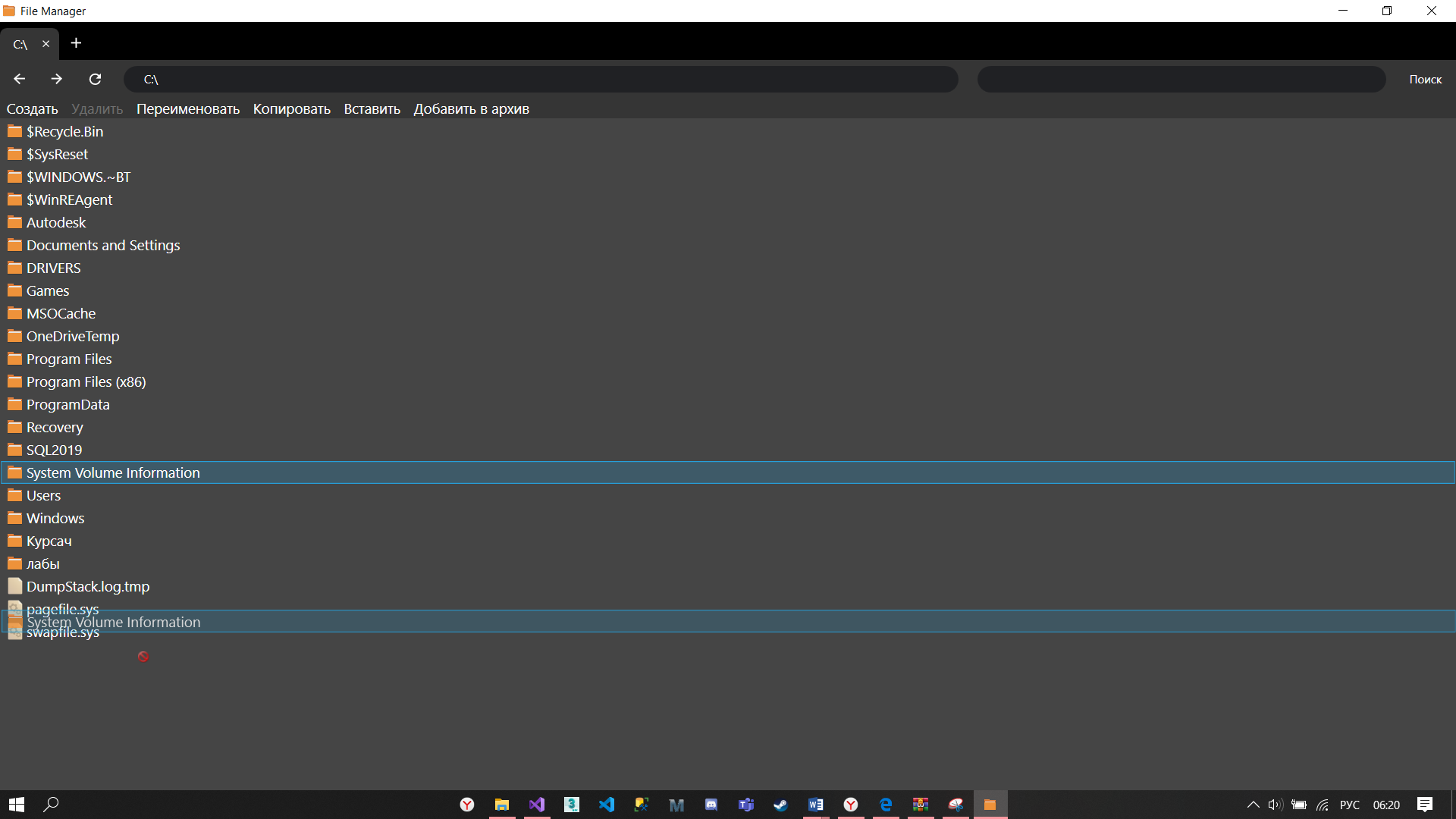
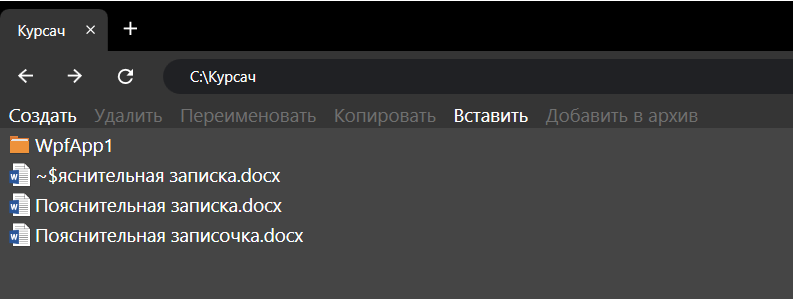
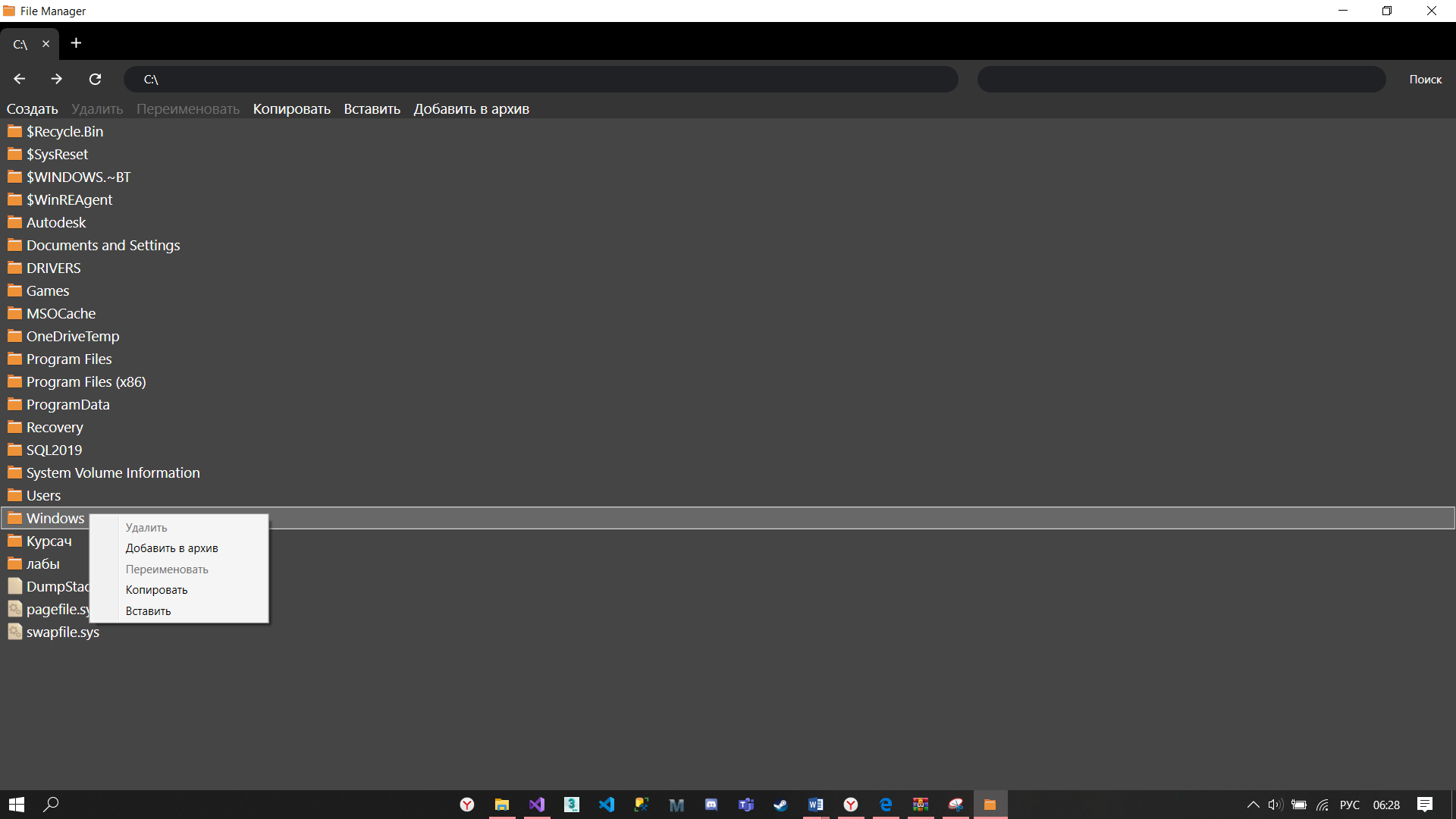
 

Рисунок 5.1 – Примеры результатов тестов

# Руководство по установке и использованию

Для открытия папки\логического диска\файла необходимо сделать двойной клик по элементу списка. Внешний вид списка указан на рисунке 6.1.

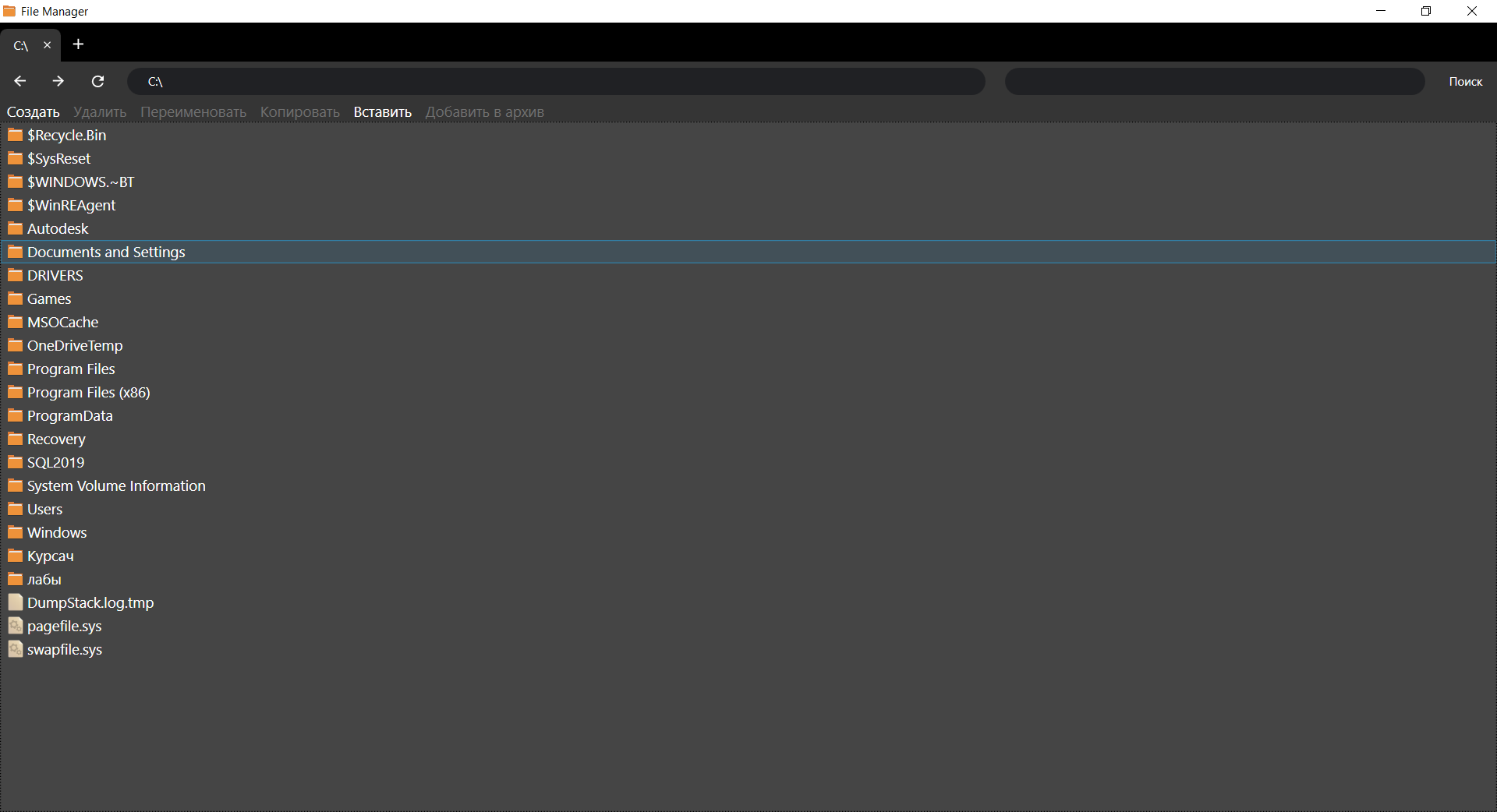


Рисунок 6.1 – Дизайн приложения

Приложение предоставляет два способа вызова операций над файлами и папками:

– из меню в верхней части экрана;

– вызов контекстного меню нажатием правой кнопки мыши по элементу списка файлов.

Чтобы воспользоваться первым способом необходимо один раз нажать на папку или файл, чтобы он был выбранным. В таком случае будут видны операции, которые можно выполнить над данным файлом или папкой.

Пример вызова контекстного меню показан на рисунке 6.2.

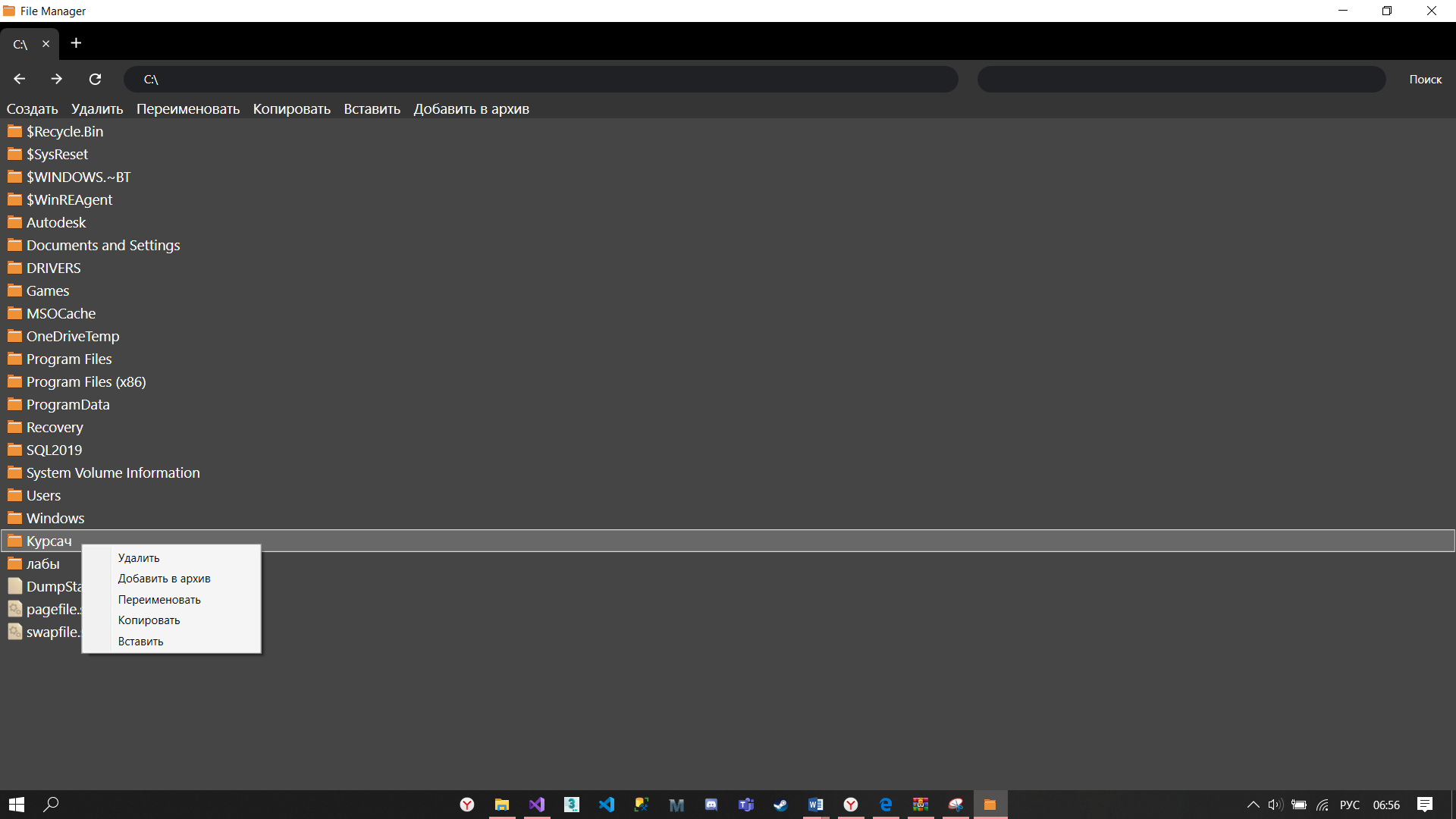


Рисунок 6.2 – Вызов контекстного меню

Создание файла либо папки.

Для создания файла либо папки необходимо зайти в папку, где вы хотите создать файл\папку и нажать кнопку **Создать**, после чего появится подменю, где можно выбрать что вы хотите создать. Данный шаг показан на рисунке 6.3.

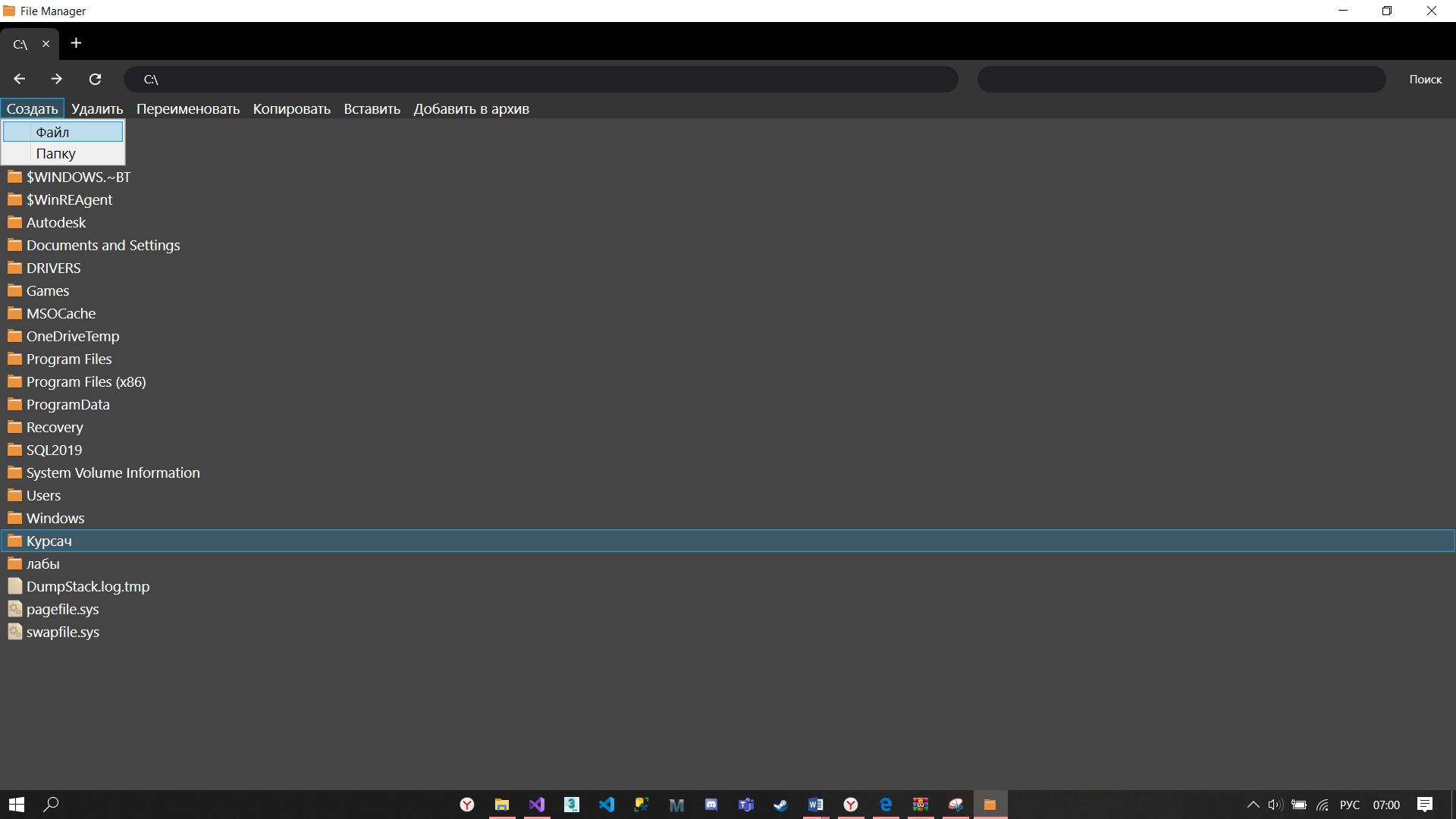


Рисунок 6.3 – Создание файла\папки

После выбора того, что вы хотите создать, появится текстовое поле, в которое нужно ввести имя нового файла\папки, а также 2 кнопки **Создать** и **Отмена.**

Если введённое имя будет удовлетворять заданным условиям проверки на правильность, например, нельзя создать папку\файл с именем, которое уже используется в папке назначения, то кнопка **Создать** станет доступной. Иллюстрация данного процесса показана на рисунке 6.4.

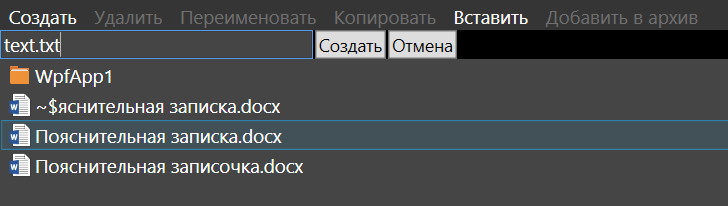


Рисунок 6.4 – Создание файла

Переименование.

Процесс переименования очень похож на ранее описанный процесс создания папки\файла, т.к. используется всё то же текстовое поле, однако вместо книпки **Создать** будет кнопка **Переименовать**. Пример переименования файла показана на рисунке 6.5.

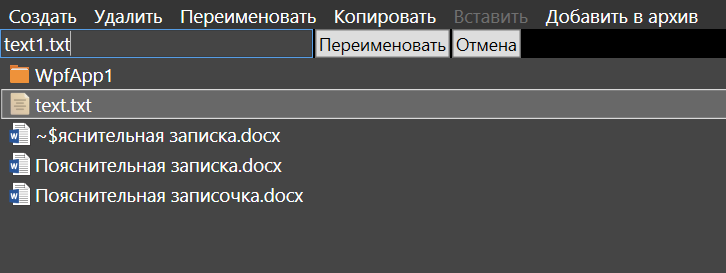


Рисунок 6.5 – Переименование файла\папки

Создание архива.

Для создания архива необходимо один раз нажать левой кнопкой мыши на файл\папку, а затем либо в верхней меню, либо в контекстном меню нажать кнопку **Добавить в архив,** после чего в текущей папке будет создан архив с расширением .zip, имя которого будет совпадать с именем файла\папки.

Перемещение файла\папки.

Для перемещения файла/папки в другую папку необходимо схватить его левой кнопкой мыши и начать тащить на папку, в которую вы хотите переместить данный файл. При этом будет запрещено помещать файл/папку в файл. Пример перемещения файла показан на рисунке 6.6.

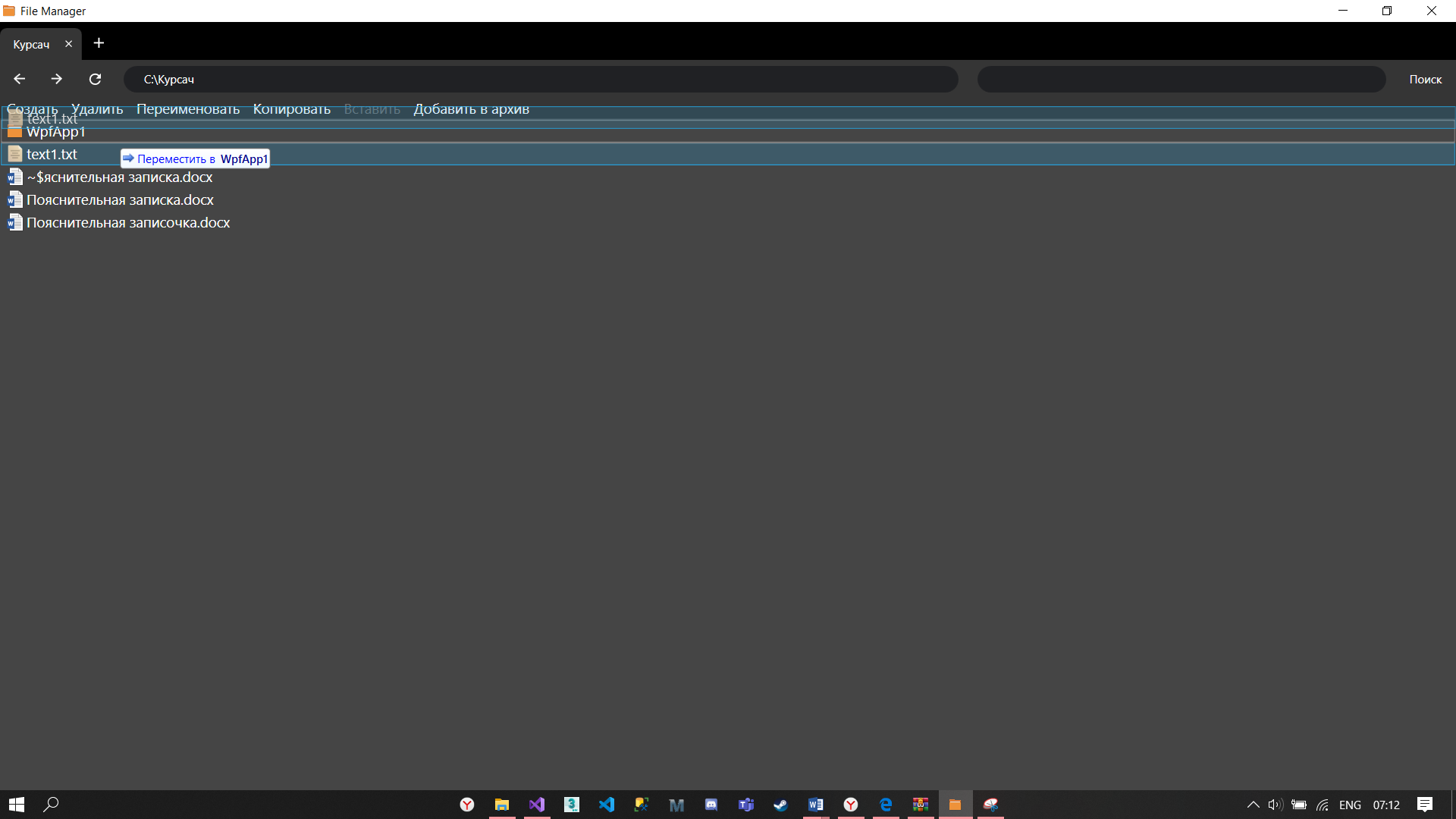


Рисунок 6.6 – Перемещение файла в папки

# ЗАКЛЮЧЕНИЕ

Итогом курсового проекта является рабочее приложение для работы с файлами на своём компьютере. В результате выполнения были разработаны все функции, прописанные в листе заданий, а также добавлен дополнительный функционал, способствующий обогащению функционала программного средства

За период выполнения курсового проекта по созданию приложения «File Manager» были изучены принципы создания декстопного приложения с использованием .NET и WPF, изучен новый язык верстки XAML для построения интерфейса и работы с ним.

Также в процессе выполнения данного курсового проекта были закреплены навыки программирования на языке C#, создания приложений на WPF, работы с архитектурным паттерном MVVM, Singleton.

Также в процессе разработки проекта были использованы дополнительные библиотеки и пакеты, например, GongSolutions.Wpf.DragDrop, Microsoft.VisualBasic.FileIO, PropertyChanged.Fody, которые были добавлены с помощью редактора пакетов NuGet.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Docs.Microsoft [Электронный ресурс]. – BackgroundWorker Класс. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/api/system.componentmodel.backgroundworker?view=netframework-4.8.

2. Docs.Microsoft [Электронный ресурс]. – Dispatcher Класс. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.threading.dispatcher?view=netframework-4.8.

3. Metanit [Электронный ресурс]. – Руководство по WPF. – Режим доступа: <https://metanit.com/sharp/wpf/>

**ПРИЛОЖЕНИЕ 1**

Диаграмма классов

