МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гродненский государственный университет имени Янки Купалы»

Факультет математики и информатики

Кафедра современных технологий программирования

ВЫСКВАРКО ГЛЕБ ИГОРЕВИЧ

**Разработка приложения «Cистема поиска файлов-дубликатов»**

Курсовой проект

по дисциплине «Конструирование программного обеспечения»

студент 2 курса специальности

6-05-0612-01 «Программная инженерия»

дневной формы получения образования

Научный руководитель

Урбан Ольга Ивановна,

старший преподаватель кафедры современных

технологий программирования,

магистр физико-математических наук

Гродно, 2024

**РЕЗЮМЕ**

Выскварко Глеб Игоревич

**Разработка приложения «Cистема поиска файлов-дубликатов»**

Работа содержит: 37 страницы, 7 рисунков, 5 использованных источников литературы.

Ключевые слова: WPF, С#, .NET framework.

Цель курсовой работы – Разработка и исследование системы поиска файлов-дубликатов, способной эффективно анализировать набор данных, выявлять повторяющиеся файлы на основе содержимого или метаданных, а также предлагать методы их оптимального управления для снижения занимаемого объема памяти и повышения производительности работы с данными.

Предусмотрено решение следующих задач:

1. Выполнить сравнительный анализ ПО аналогов.
2. Проектирование приложения на языке программирования C#.
3. Разработка и программная реализация интерфейса приложения:
   * Выявление ключевых функций, которые должны быть доступны через интерфейс (поиск файлов-дубликатов, просмотр результатов, удаление/перемещение файлов и т. д.).
   * Создание сценариев использования (user stories) для различных случаев взаимодействия пользователей с приложением.

**RESUME**

**Vysskvarka Hleb Igorevich**  
**Development of the Application "Duplicate File Search System"**

The work includes: 37 pages, 7 figures, 5 referenced sources.  
 **Keywords**: WPF, C#, .NET Framework.

**The goal of the course project** is to develop and study a duplicate file search system capable of efficiently analyzing datasets, identifying duplicate files based on content or metadata, and proposing methods for optimal management to reduce occupied memory and improve data handling performance.

The following tasks are addressed:

1. Perform a comparative analysis of similar software.
2. Design the application using the C# programming language.
3. Develop and implement the application's user interface:
   * Identify key functions to be accessible through the interface (duplicate file search, result preview, file deletion/movement, etc.).
   * Create user stories for various interaction scenarios between users and the application.

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ](#_Toc185156059) 5

[ГЛАВА 1 7](#_Toc185156060)

[АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc185156061)

[1.1. Постановка задач для реализации приложения C**истема поиска файлов-дубликатов** 7](#_Toc185156062)

[1.2. Обзор существующих решений приложений](#_Toc185156063) 9

[1.3. Выводы по главе 1 1](#_Toc185156067)6

[ГЛАВА 2 1](#_Toc185156068)9

[ПРОЕКТИРОВАНИЕ СИСТЕМЫ 1](#_Toc185156069)9

[2.1. Проектирование функций приложения](#_Toc185156070) 19

[2.2. Построение графического интерфейса приложения](#_Toc185156071) 20

[2.3. Вывод по главе 2](#_Toc185156072) 22

[ГЛАВА 3](#_Toc185156073) 26

[ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ 2](#_Toc185156074)6

[3.1. Выбор программных средств 2](#_Toc185156075)6

[3.2. Описание структуры приложения 2](#_Toc185156076)8

[3.3. Работа с приложением](#_Toc185156077) 31

[3.4. Выводы по главе 3](#_Toc185156078) 35

[ЗАКЛЮЧЕНИЕ](#_Toc185156079) 36

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 3](#_Toc185156080)7

**ВВЕДЕНИЕ**

Современные цифровые устройства и компьютерные системы ежедневно создают и обрабатывают огромные объемы данных. В процессе работы с файлами часто возникают ситуации, когда на одном устройстве или в одной файловой системе сохраняются дубликаты данных. Это могут быть одинаковые фотографии, документы, музыкальные файлы или другие типы данных, которые занимают ценное дисковое пространство.

Проблема усугубляется с ростом объемов хранения и усложнением файловой структуры, что затрудняет ручное управление файлами и их оптимизацию. Избыточное хранение одинаковых данных может негативно сказываться на производительности системы, вызывать сложности при поиске нужной информации и увеличивать расходы на хранилища данных.

Для решения этой проблемы необходимо специализированное программное обеспечение, которое позволит автоматизировать поиск и идентификацию дублирующихся файлов.

Проблема дублирования файлов связана с несколькими основными аспектами:

1. **Переполнение дискового пространства**  
   Избыточное количество дубликатов увеличивает потребление дискового пространства, особенно на системах с ограниченными ресурсами. Это приводит к необходимости расширения хранилища, что может быть дорогостоящим.
2. **Сложности управления файлами**  
   Дубликаты создают путаницу в структуре файловой системы, увеличивая время на поиск нужных данных. Пользователь может случайно работать с неверной версией файла, что чревато потерей или искажением данных.
3. **Затруднения в резервном копировании**  
   Системы резервного копирования часто копируют файлы без анализа их уникальности. Дублирование увеличивает время и объемы копирования, снижая эффективность резервного процесса.
4. **Ухудшение производительности системы**  
   Увеличение общего объема данных может снижать производительность файловой системы, так как возрастает время доступа к информации.
5. **Сложность в идентификации дубликатов**  
   Дублирующиеся файлы могут иметь различные имена, временные метки или быть сохранены в разных папках, что затрудняет их обнаружение.

Создание приложения для автоматизированного поиска файлов-дубликатов позволит решить эти задачи, обеспечив эффективное управление ресурсами и повышение производительности системы. Разрабатываемая система будет оснащена удобным интерфейсом для удаления файлов дубликатов.

ГЛАВА 1

АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Постановка задач для реализации приложения **система поиска файлов-дубликатов**

В рамках курсовой работы будет разработано приложение для поиска файлов-дубликатов с минимально необходимым функционалом. Основное внимание уделяется созданию графического интерфейса, удобного для взаимодействия с пользователем, и реализации ключевых функций поиска и отображения результатов. Логирование и экспорт данных не входят в состав задач.

#### 1.1. Исследование пользовательских потребностей

* Определить минимальный набор функций для поиска дубликатов: сканирование, идентификация и управление результатами.
* Уточнить, какие типы файлов будут обрабатываться (например, изображения, текстовые документы).

#### 1.2. Проектирование архитектуры

* Разработать архитектуру, включающую модуль обработки файловой системы, модуль поиска дубликатов и графический интерфейс.
* Предусмотреть возможность добавления новых функций в будущем

2. Реализация основных функций

#### 2.1. Обход файловой системы

* Реализовать функцию сканирования выбранной пользователем директории с возможностью рекурсивного обхода.
* Добавить базовые фильтры, такие как тип и размер файла, для ограничения анализируемых данных.

#### 2.2. Поиск дубликатов

* Использовать алгоритмы предварительного отбора дубликатов на основе размеров и метаданных файлов.
* Реализовать вычисление хэшей файлов (например, SHA-256) для точного сравнения содержимого.
* Ориентироваться на обработку файлов среднего размера для упрощения реализации.

#### 2.3. Отображение результатов

* Разработать графический интерфейс для отображения найденных дубликатов, сгруппированных по наборам.
* Реализовать функции удаления или перемещения выбранных дубликатов.

#### 2.4. Обработка ошибок

* Обеспечить обработку ошибок, связанных с доступом к файлам, с выводом соответствующих уведомлений в интерфейсе.
* Предусмотреть игнорирование недоступных файлов для обеспечения стабильности работы.

#### 3.1. Дизайн интерфейса

* Создать простой и интуитивно понятный графический интерфейс, который позволит:
  + Выбирать папку для анализа.
  + Настраивать параметры поиска.
  + Просматривать результаты и выполнять действия с дубликатами.

#### 3.2. Основные элементы интерфейса

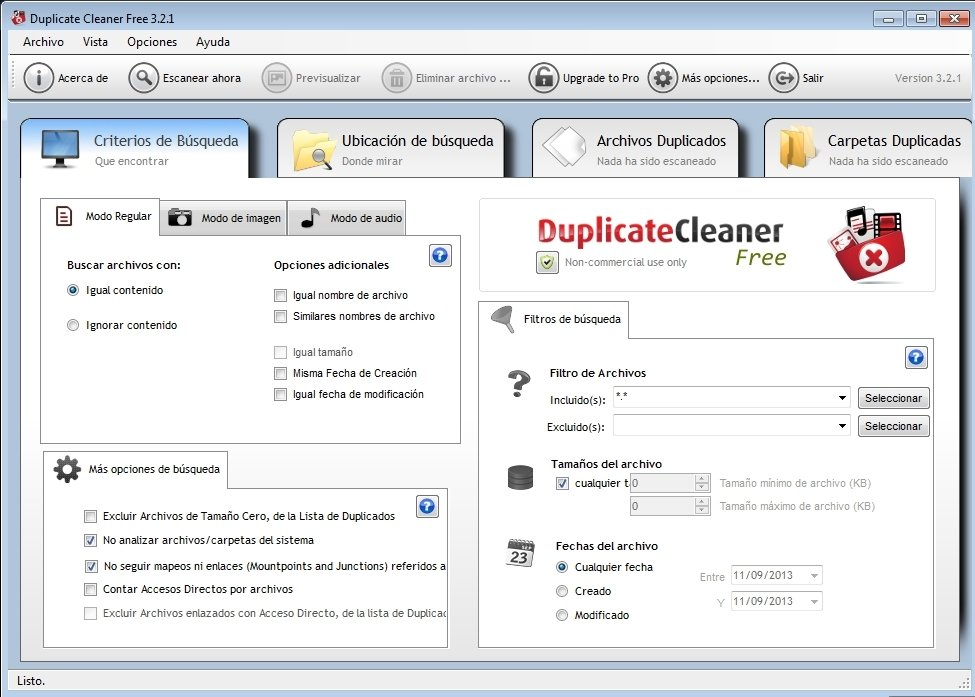
* Главное окно с кнопками для запуска анализа и выбора директории.
* Список результатов с возможностью выделения групп дубликатов.
* Элементы управления действиями над дубликатами (удаление, перемещение).

#### 4.1. Тестирование

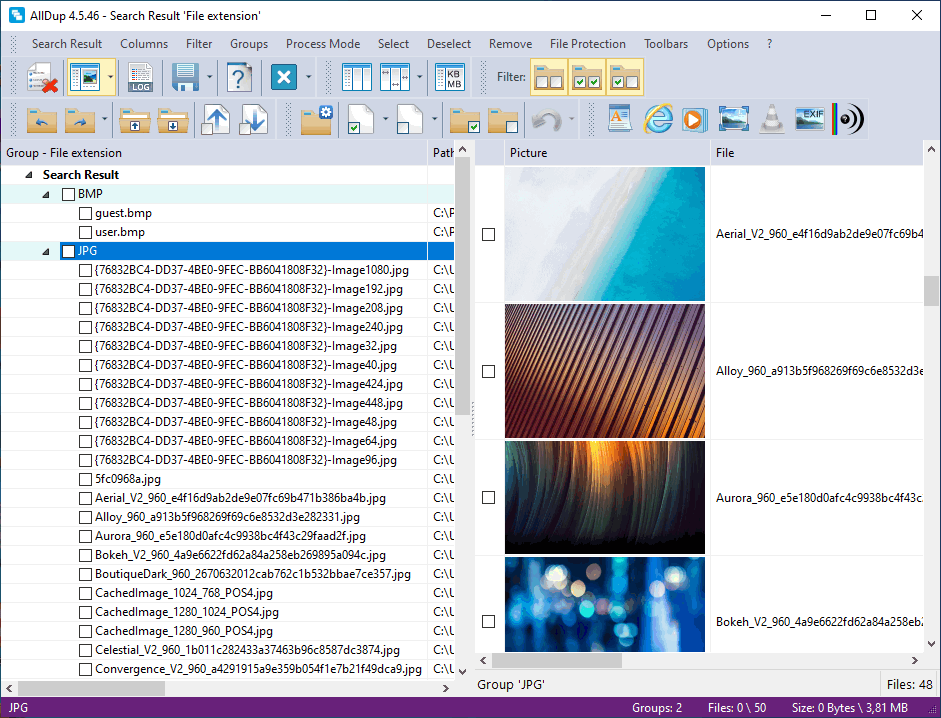
* Провести функциональное тестирование на наборах данных с реальными дубликатами и уникальными файлами.
* Проверить корректность обработки ошибок (например, недоступных или защищенных файлов).

По итогам курсовой работы будет создано приложение с графическим интерфейсом, позволяющее пользователю находить и управлять файлами-дубликатами. Основное внимание будет уделено удобству использования и корректности выполнения ключевых операций.

**1.2. Обзор существующих решений**

Рисунок 1.1(**Duplicate Cleaner Free**)

#### 1.1. ****Duplicate Cleaner Free****

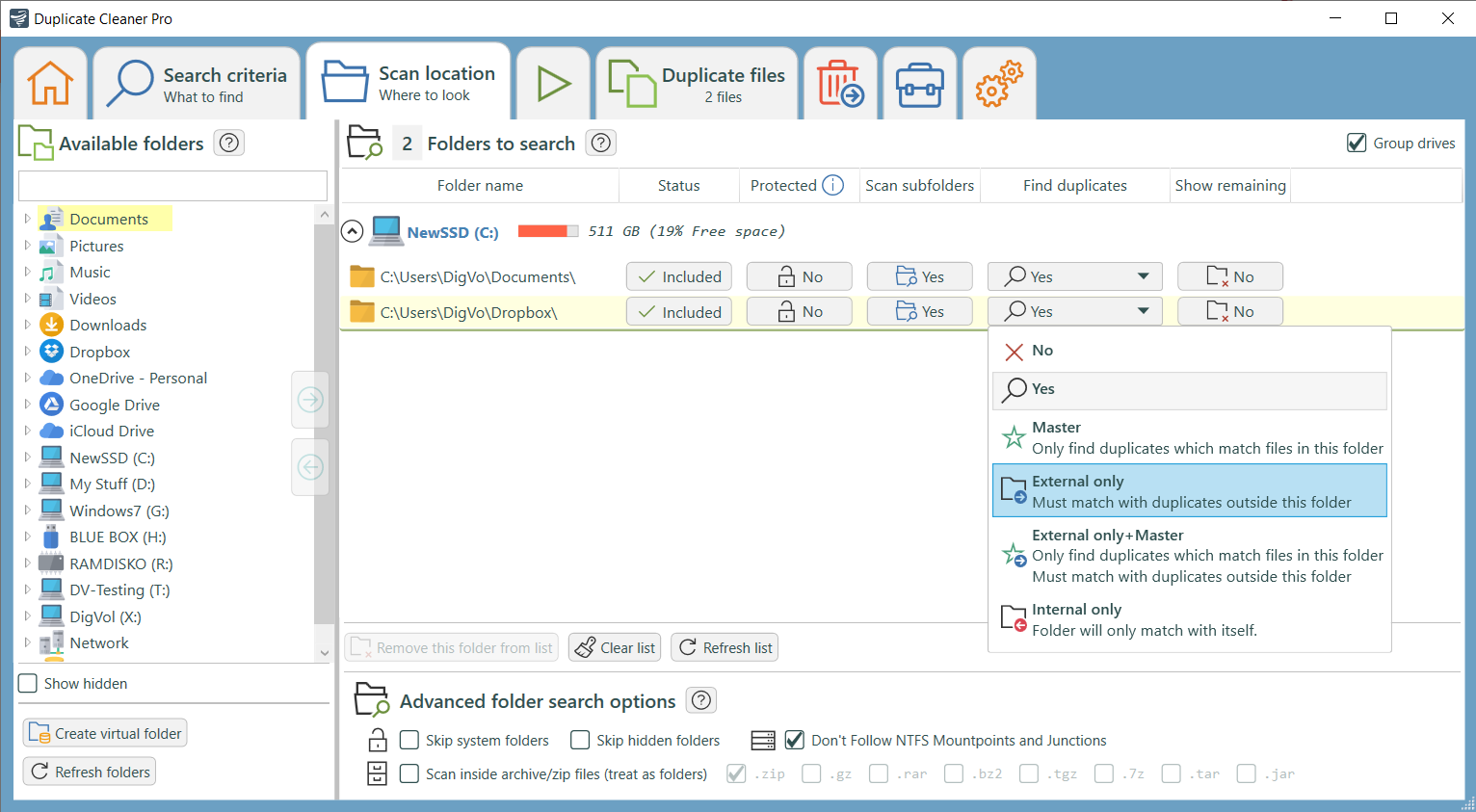
* **Описание:** Популярное приложение с базовым набором функций для поиска дубликатов.
* **Функции:**
  + Сканирование указанных папок.
  + Поиск дубликатов по содержимому, имени, размеру и другим критериям.
  + Фильтрация файлов по типу.
* **Преимущества:**
  + Простота использования.
  + Бесплатность.
* **Недостатки:**
  + Ограниченные возможности в бесплатной версии.
  + Устаревший пользовательский интерфейс.  
    Рисунок 1.2(**AllDup**)

#### 1.2. ****AllDup****

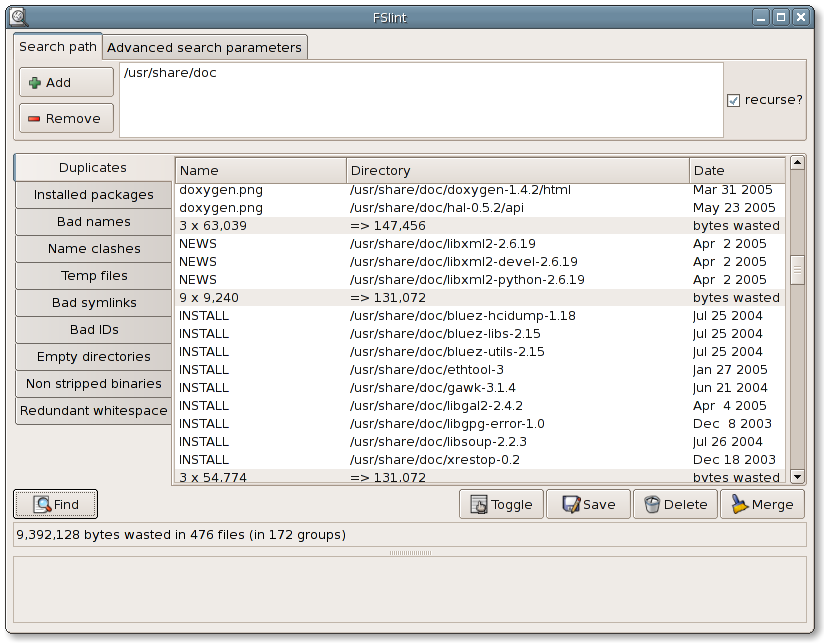
* **Описание:** Бесплатная программа для поиска дубликатов с широким функционалом.
* **Функции:**
  + Поиск по содержимому, имени и атрибутам файла.
  + Поддержка больших файловых систем.
  + Гибкие фильтры для исключения файлов.
* **Преимущества:**
  + Подробные настройки.
  + Возможность предварительного просмотра.
* **Недостатки:**
  + Сложность интерфейса для неопытных пользователей.

#### **2.1. **Duplicate Cleaner Pro****

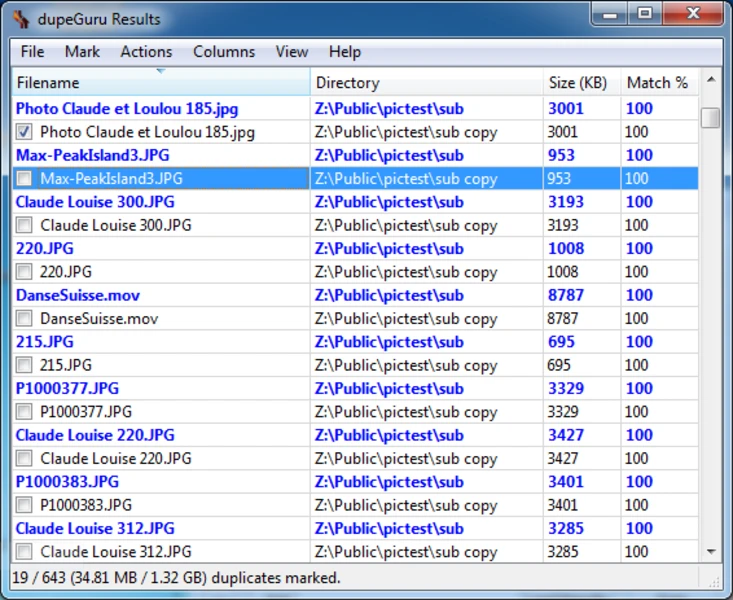
* **Описание:** Расширенная версия Duplicate Cleaner с дополнительными функциями.
* **Функции:**
  + Углубленный анализ мультимедийных файлов.
  + Группировка и сортировка дубликатов.
  + Поддержка обработки сетевых дисков.
* **Преимущества:**
  + Поддержка большого числа форматов.
  + Профессиональные инструменты для управления дубликатами.
* **Недостатки:**
  + Высокая стоимость.
  + Требует лицензии для полного доступа.

Рисунок 1.3(**CCleaner (Duplicate Finder)**)

#### 2.2. ****CCleaner (Duplicate Finder)****

* **Описание:** Инструмент для поиска дубликатов, встроенный в популярное приложение для очистки системы.
* **Функции:**
  + Поиск дубликатов по основным параметрам (размер, имя, содержимое).
  + Интеграция с функциями очистки системы.
* **Преимущества:**
  + Удобство использования в рамках комплексного инструмента.
  + Интуитивно понятный интерфейс.
* **Недостатки:**
  + Ограниченный функционал по сравнению с другими решениями.  
    Рисунок 1.4(**FSlint**)

#### 3.1. ****FSlint****

* **Описание:** Бесплатное приложение для поиска дубликатов, доступное на Linux.
* **Функции:**
  + Поиск по содержимому и метаданным.
  + Удаление или перемещение дубликатов.
* **Преимущества:**
  + Простота и легкость.
  + Подходит для Linux-пользователей.
* **Недостатки:**
  + Устаревший интерфейс.
  + Отсутствие поддержки других платформ.  
    Рисунок 1.5(**dupeGuru**)

#### 3.2. ****dupeGuru****

* **Описание:** Открытое приложение с поддержкой Windows, macOS и Linux.
* **Функции:**
  + Гибкие настройки поиска (по содержимому, имени, тегам).
  + Поддержка музыкальных и графических файлов.
* **Преимущества:**
  + Кроссплатформенность.
  + Открытый исходный код.
* **Недостатки:**
  + Ограниченная производительность на больших объемах данных.

### **Выводы**

Существующие решения для поиска файлов-дубликатов предлагают широкий спектр функций и интерфейсов, от минималистичных инструментов до профессиональных систем. Основные аспекты, на которых фокусируются разработчики:

1. **Алгоритмы идентификации дубликатов:** Сравнение метаданных, содержимого или использование хэш-функций.
2. **Интерфейс:** Простые и интуитивно понятные решения чаще выбираются массовыми пользователями, тогда как профессиональные инструменты предоставляют больше настроек.
3. **Кроссплатформенность:** Большинство решений ограничены одной операционной системой, что может быть препятствием для части пользователей.

Для курсового проекта важно учитывать успешные подходы из существующих решений, такие как:

* Использование хэш-функций для точного сравнения файлов.
* Разработка интуитивного интерфейса, ориентированного на основные сценарии пользователя.
* Сосредоточение на ключевых функциях, таких как сканирование и управление результатами, без перегрузки функционала.

1.3. Выводы по главе 1

Анализ предметной области выявил ключевые аспекты и задачи для разработки приложения для поиска файлов-дубликатов, а также позволил оценить существующие решения, выделив их сильные и слабые стороны. Основные выводы и рекомендации:

1. **Необходимость минимального функционала**  
   Для обеспечения удобства пользователя приложение должно быть сосредоточено на реализации базовых функций: сканировании файловой системы, идентификации дубликатов и управлении результатами. Дополнительные возможности, такие как логирование или экспорт данных, можно добавить в будущем, но в рамках данной работы они исключены, чтобы сконцентрироваться на качестве реализации основных задач.
2. **Фокус на графическом интерфейсе**  
   Одной из ключевых особенностей приложения станет интуитивно понятный графический интерфейс. Пользовательский опыт играет важную роль, особенно для массовой аудитории. Интерфейс должен позволять:
   * Легко выбирать директорию для анализа.
   * Настраивать параметры поиска.
   * Просматривать результаты и управлять дубликатами (удалять или перемещать).
3. **Архитектура и модульность**  
   Предложенная архитектура приложения включает три основных компонента:
   * Модуль обработки файловой системы.
   * Модуль поиска дубликатов.
   * Графический интерфейс. Эта структура обеспечивает возможность дальнейшего расширения функциональности, например, добавления поддержки новых алгоритмов или форматов файлов.
4. **Использование эффективных алгоритмов**  
   Для точной идентификации дубликатов планируется использование хэш-функций (например, SHA-256). Этот метод зарекомендовал себя как надёжный способ сравнения содержимого файлов. На первом этапе для ускорения работы будет реализован предварительный отбор файлов по размерам и метаданным.
5. **Сравнение с существующими решениями**  
   Анализ аналогов, таких как Duplicate Cleaner Free, AllDup, CCleaner, FSlint, и dupeGuru, выявил следующие особенности:
   * Простые инструменты, ориентированные на массового пользователя, предлагают базовый функционал, но часто имеют устаревший интерфейс или ограниченные возможности.
   * Профессиональные решения, такие как Duplicate Cleaner Pro, предоставляют расширенный функционал, но требуют значительных затрат или сложны в использовании для неподготовленных пользователей.
   * Приложения с открытым исходным кодом (например, dupeGuru) выделяются гибкостью и кроссплатформенностью, но могут быть менее производительными на больших объёмах данных.
6. **Учет пользовательских потребностей**  
   Важным выводом из исследования стало понимание потребностей пользователей. Наибольшую ценность представляют:
   * Простота использования и минимализм интерфейса.
   * Высокая точность поиска дубликатов.
   * Возможность настроек для фильтрации и управления результатами.
7. **Особенности тестирования и обработки ошибок**  
   Для обеспечения стабильности приложения необходимо предусмотреть:
   * Обработку ошибок, связанных с доступом к файлам, с выводом соответствующих уведомлений.
   * Игнорирование недоступных или защищённых файлов, чтобы избежать остановки процесса.

Разработка приложения для поиска файлов-дубликатов должна учитывать успешные подходы из существующих решений, но с фокусом на минимализм, интуитивный интерфейс и высокую точность работы. Главным результатом станет удобное и функциональное средство для массовой аудитории, обеспечивающее базовые сценарии работы с файлами-дубликатами.

ГЛАВА 2

ПРОЕКТИРОВАНИЕ СИСТЕМЫ

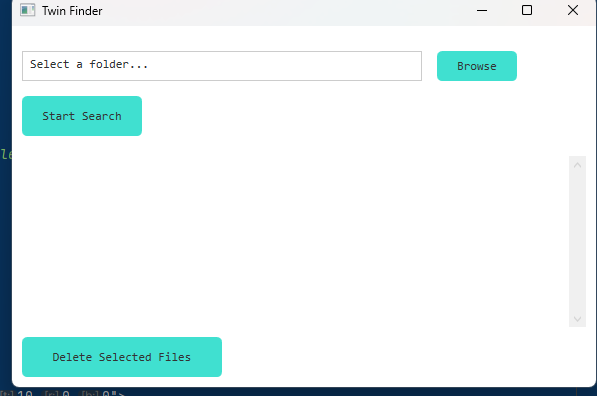
2.1. Проектирование функций приложения

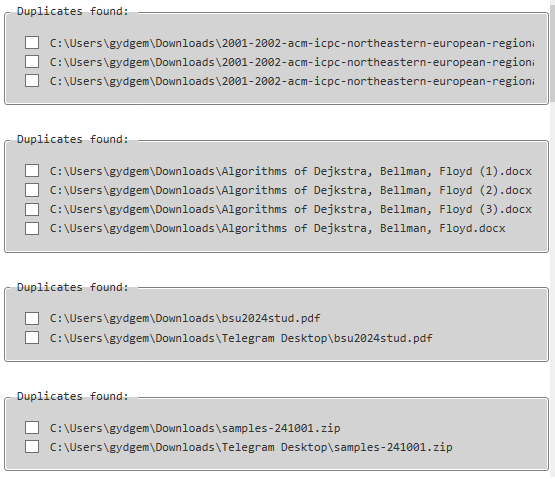
Для реализации приложения предусмотрены следующие функции:

1. **Выбор директории для анализа**
   * Возможность указать папку, в которой будет производиться поиск файлов-дубликатов.
   * Поддержка как ручного ввода пути, так и выбора папки через системный диалог.
2. **Запуск процесса поиска**
   * Функция сканирования указанной директории с возможностью рекурсивного обхода вложенных папок.
   * Выявление дубликатов на основе предварительного отбора по размеру и метаданным, а также с использованием хэш-функции (SHA-256) для точного сравнения содержимого файлов.
3. **Отображение результатов анализа**
   * Формирование списка найденных дубликатов, сгруппированных по наборам.
   * Включение информации о файлах, такой как имя, размер и путь.
4. **Удаление выбранных файлов-дубликатов**
   * Возможность выбора конкретных дубликатов из результатов поиска.
   * Функция удаления выбранных файлов для освобождения места на диске.
5. **Обработка больших объемов данных**
   * Обеспечение корректной работы с длинными списками результатов, включая возможность их группировки и сортировки.
6. **Обработка ошибок**
   * Игнорирование недоступных файлов с соответствующими уведомлениями для пользователя.
   * Обеспечение стабильности работы приложения при возникновении ошибок, связанных с доступом к файлам.

Эти функции обеспечивают выполнение ключевых задач приложения по поиску и управлению файлами-дубликатами.

2.2. Построение графического интерфейса приложения

Рисунок 2.1(основной вид приложения)

1. **Основное окно**:
2. **Поле выбора папки**:
   * Содержит текстовое поле (TextBox), где будет отображаться путь к выбранной папке. Оно доступно только для чтения, чтобы пользователь не изменял текст вручную.
   * Кнопка Browse, которая открывает окно выбора папки, для удобства пользователя. При нажатии на кнопку срабатывает событие BrowseButton\_Click, которое открывает диалоговое окно для выбора директории.
3. **Кнопка запуска поиска**:
   * Кнопка StartButton инициирует процесс поиска дубликатов файлов в выбранной папке. Она расположена в первой строке сетки и активируется только после выбора директории.
   * При нажатии на кнопку выполняется поиск, а результаты отображаются в следующем блоке.
4. Рисунок 2.2 (вид результата)
5. **Блок с результатами поиска**:
   * Результаты поиска дубликатов отображаются внутри ScrollViewer с вертикальной прокруткой. Внутри будет StackPanel, куда динамически добавляются элементы, представляющие найденные дубликаты.
6. **Кнопка удаления выбранных файлов**:
   * Кнопка DeleteButton предназначена для удаления выбранных дубликатов. Она будет доступна только в случае, если найдены дубликаты (с помощью параметра IsEnabled="False").
   * При нажатии на кнопку выполняется удаление файлов.

* **TextBox** — для отображения выбранного пути.
* **Button** — для взаимодействия с пользователем (выбор папки, запуск поиска, удаление файлов).
* **ScrollViewer** и **StackPanel** — для динамического отображения результатов поиска.

Этот интерфейс легко расширяется в зависимости от функциональных требований, например, можно добавить индикатор загрузки или более подробную информацию о каждом дубликате.

2.3. Вывод по главе 2

Проектирование приложения для поиска и удаления дубликатов файлов включает в себя важные аспекты, которые охватывают как функциональные, так и интерфейсные требования. Обеспечение удобства использования, точности выполнения операций и безопасности данных — ключевые задачи при создании такого приложения. Важно, чтобы приложение не только выполняло свою основную задачу эффективно, но и предоставляло пользователю комфортное взаимодействие с системой, минимизируя возможные ошибки и предлагая ясные инструкции и уведомления.

Одним из важнейших функциональных аспектов является возможность выбора директории для анализа. Пользователь должен иметь гибкость в выборе папки: как через ручной ввод пути, так и с помощью удобного диалога выбора папки, который упрощает процесс. Это позволяет работать как с заранее известными путями, так и с динамически выбранными папками. Для большей гибкости система должна поддерживать возможность выбора нескольких директорий для анализа, что может быть полезно для пользователей, желающих выполнить поиск дубликатов на нескольких дисках или в разных папках.

Следующий ключевой момент — это точность поиска дубликатов. Приложение должно не только проверять файлы на основе их размера и метаданных (например, времени последнего изменения или типа), но и использовать хэш-функции, такие как SHA-256, для точного сравнения содержимого файлов. Это необходимо для предотвращения ложных срабатываний и обеспечения того, чтобы только идентичные файлы, включая их содержимое, были помечены как дубликаты. Важно также учесть необходимость рекурсивного поиска по вложенным папкам, что позволит пользователю находить дубликаты не только в основной директории, но и во всех её подкаталогах.

Отображение результатов поиска должно быть организовано таким образом, чтобы пользователь мог легко понять, какие файлы являются дубликатами и какие действия нужно предпринять с ними. Результаты поиска должны быть группированы по наборам дубликатов, а также содержать информацию о каждом файле, такую как имя, размер, путь и, возможно, дополнительные метаданные, такие как дата последнего изменения. Это предоставит пользователю всю необходимую информацию для принятия решений о дальнейших действиях.

Удаление выбранных дубликатов должно быть безопасным и простым процессом. Пользователь должен иметь возможность выделить конкретные файлы и удалить их для освобождения места на диске. Важно предусмотреть систему подтверждения удаления файлов, чтобы предотвратить случайные ошибки. Также необходимо учитывать возможность восстановления удаленных файлов, например, путём перемещения их в корзину или создание резервной копии перед удалением.

Интерфейс приложения должен быть интуитивно понятным и удобным для пользователей с разным уровнем технической подготовки. Основное окно должно быть организовано таким образом, чтобы все важные функции были легко доступны и логично сгруппированы. Например, поле для выбора директории должно быть расположено в верхней части окна, рядом с кнопкой для запуска поиска, что позволяет пользователю быстро начать работу. Кнопка для выбора папки должна быть явно видна, а текстовое поле для отображения пути должно быть доступным для быстрого ознакомления с выбранной директорией.

Кроме того, интерфейс должен предусматривать возможность динамического отображения результатов поиска. Использование компонентов, таких как ScrollViewer и StackPanel, позволяет эффективно работать с большими объемами данных, обеспечивая удобную прокрутку результатов. Важно, чтобы результаты были представлены в виде списков или таблиц, что позволит пользователю легко сравнивать файлы и оценивать, какие из них являются дубликатами. Каждый элемент в списке должен содержать достаточную информацию, чтобы пользователь мог легко распознать файл, а также иметь возможность выделить несколько элементов для последующих действий (например, удаления).

Отдельное внимание следует уделить кнопке удаления файлов. Эта кнопка должна быть доступна только после того, как пользователем будут найдены дубликаты. Важный момент — кнопка удаления должна быть отключена (с помощью свойства IsEnabled="False") до тех пор, пока пользователь не выберет хотя бы один файл для удаления, чтобы избежать случайных действий. После того как файлы выбраны, кнопка должна стать активной, и при нажатии на неё будет происходить удаление.

Также важно предусмотреть дополнительные механизмы, такие как индикатор загрузки или прогресса, который будет отображаться в процессе поиска, чтобы пользователь мог видеть, что приложение работает и насколько быстро происходит процесс сканирования файлов. Это особенно важно при работе с большими объемами данных, когда поиск может занять продолжительное время.

Немаловажным аспектом является качественная обработка ошибок. В процессе работы с файлами может возникать множество ситуаций, когда файлы будут недоступны для чтения или обработки, например, из-за отсутствия прав доступа или файловой системы. В таких случаях приложение должно информировать пользователя о возникшей ошибке и игнорировать такие файлы, чтобы не нарушать выполнение остальной части программы. Важно, чтобы сообщения об ошибках были понятными и не создавали дополнительных сложностей для пользователя. Например, при недоступности файла должно выводиться уведомление о том, что файл не может быть прочитан, а не общее сообщение о сбое.

Кроме того, необходимо предусмотреть систему логирования ошибок, которая будет фиксировать возникающие проблемы в процессе работы приложения. Это поможет не только пользователю понять, что произошло, но и разработчику — для диагностики и исправления возможных багов в будущем.

Гибкость интерфейса и архитектуры приложения — это важные аспекты, которые позволяют легко добавлять новые функции в будущем. Например, можно интегрировать функционал для анализа не только файлов, но и папок с возможностью их удаления или перемещения. Также могут быть добавлены дополнительные методы фильтрации и сортировки результатов поиска, такие как фильтрация по типу файлов, дате создания или изменению.

Приложение также может быть расширено с помощью интеграции с облачными хранилищами, что позволило бы искать дубликаты не только на локальных дисках, но и в облаке. Это потребует создания дополнительных механизмов для работы с облачными сервисами и обеспечения безопасности данных.

Кроме того, можно внедрить возможности для создания отчетов о проделанных действиях, что позволит пользователю видеть, сколько места было освобождено, какие файлы были удалены, а также получить более подробную информацию о каждом действии.

ГЛАВА 3

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

ПРИЛОЖЕНИЯ ДЛЯ СИСТЕМА ПОИСКА ФАЙЛОВ-ДОБЛИКАТОВ

3.1. Выбор программных средств

В качестве основной платформы разработки была выбрана современная технология .NET Core 8.0, выпущенная компанией Microsoft в 2023 году. .NET Core представляет собой кроссплатформенную, высокопроизводительную среду разработки с открытым исходным кодом, которая позволяет создавать современные приложения для различных операционных систем.

Для реализации пользовательского интерфейса был выбран фреймворк Windows Presentation Foundation (WPF). WPF – это система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем. WPF предоставляет широкий спектр элементов управления и использует язык разметки XAML (eXtensible Application Markup Language) для описания пользовательского интерфейса, что позволяет эффективно отделить логику приложения от его представления. Данная технология была выбрана благодаря её богатым возможностям для создания современного и отзывчивого интерфейса кассы самообслуживания.

В качестве языка программирования используется C# – современный объектно-ориентированный язык программирования, разработанный компанией Microsoft. C# версии 12, которая поставляется с .NET 8.0, предоставляет множество современных возможностей, таких как паттерн-матчинг, записи (records), nullable reference types и другие функции, которые позволяют писать более надежный и поддерживаемый код.

Для работы с данными в приложении используется формат CSV (Comma-Separated Values). Этот формат был выбран в качестве легковесной альтернативы традиционным системам управления базами данных, что особенно актуально для автономной работы касс самообслуживания. CSV-файлы обеспечивают простое и эффективное хранение табличных данных, таких как каталог товаров, цены и информация о транзакциях.

. Выбранный стек технологий обеспечивает надежную работу системы кассы самообслуживания, предоставляя все необходимые инструменты для реализации требуемого функционала: от создания удобного пользовательского интерфейса до формирования отчетной документации и управления данными. Современная платформа .NET Core 8.0 гарантирует высокую производительность и масштабируемость решения, а использование проверенных временем библиотек обеспечивает стабильность работы системы.

3.2. Описание структуры приложения

Структура приложения для поиска и удаления дубликатов файлов была разработана с целью разделить функциональность и улучшить модульность системы. Проект разбит на два основных компонента: **TwinFinder** и **TwinFinderWPF**. Каждый из этих компонентов отвечает за свою часть функционала, что способствует чистоте кода, его повторному использованию и удобству дальнейшего расширения.

#### 1. ****TwinFinder (логика приложения)****

Компонент **TwinFinder** представляет собой основную логику приложения, которая реализует функциональность поиска и удаления дубликатов файлов. Это ядро приложения, которое выполняет основную работу и предоставляет сервисы для обработки файлов, вычисления хешей, поиска дубликатов и других операций.

Основные задачи компонента **TwinFinder**:

* **Поиск дубликатов**: Сканирует выбранные директории и ищет дубликаты файлов, сначала по размеру, затем с использованием хэш-функции SHA-256 для точного сравнения содержимого.
* **Удаление дубликатов**: Обрабатывает удаление выбранных файлов или перемещение их в корзину. Это может включать в себя запросы на подтверждение удаления и работу с файловой системой.
* **Обработка ошибок**: Логика работы с ошибками и недоступными файлами. Например, если файл не удается прочитать, компонент сообщает об этом и исключает его из дальнейшего процесса.
* **Обработка больших объемов данных**: Оптимизация работы с большими папками и файлами, включая возможность группировки, фильтрации и сортировки результатов.

Компонент **TwinFinder** не имеет зависимости от пользовательского интерфейса. Он может быть использован в различных приложениях или сервисах для поиска и обработки дубликатов файлов.

**Основные классы и компоненты:**

* **FileScanner** — отвечает за сканирование директорий и анализ файлов.
* **DuplicateFinder** — ищет дубликаты на основе хэширования файлов.
* **FileDeleter** — управляет удалением или перемещением файлов в корзину.
* **ErrorHandler** — обрабатывает ошибки, связанные с доступом к файлам и прочие исключения.

#### 2. ****TwinFinderWPF (графический интерфейс пользователя)****

Компонент **TwinFinderWPF** отвечает за взаимодействие с пользователем и визуализацию данных. Это приложение с графическим интерфейсом, которое предоставляет удобный способ для пользователей взаимодействовать с функциональностью, реализованной в **TwinFinder**.

Основные задачи компонента **TwinFinderWPF**:

* **Интерфейс пользователя**: Обеспечивает интерфейс для выбора директорий, отображения результатов поиска дубликатов, управления списками файлов и их удалением.
* **Визуализация результатов**: Отображает результаты поиска дубликатов в виде списка или таблицы, с возможностью сортировки и фильтрации.
* **Обработка пользовательских событий**: Реализует обработку кликов на кнопки, выбор папок, запуск поиска, отображение сообщений и подтверждений.
* **Интеграция с компонентом TwinFinder**: Взаимодействует с логикой приложения через вызовы сервисов и обновляет интерфейс в зависимости от результатов.

**Основные классы и компоненты:**

* **MainWindow.xaml** — основной экран приложения, где пользователи могут выбирать директории и управлять процессом поиска дубликатов.
* **SearchResultsView** — отображает результаты поиска дубликатов в удобном виде.
* **DeleteConfirmationDialog** — диалоговое окно для подтверждения удаления файлов.
* **ProgressBar** — отображает прогресс выполнения поиска или удаления.

#### 3. ****Интеракция между TwinFinder и TwinFinderWPF****

Главной задачей является обеспечение взаимодействия между компонентами **TwinFinder** и **TwinFinderWPF**. Логика работы приложения и пользовательский интерфейс должны работать синхронно, обеспечивая пользователю функциональные возможности и приятный опыт.

* **Запуск поиска**: Когда пользователь нажимает на кнопку "Start Search" в интерфейсе **TwinFinderWPF**, компонент вызывает сервисы из **TwinFinder**, которые начинают сканирование файлов и поиск дубликатов.
* **Обновление интерфейса**: После того как поиск завершен, компонент **TwinFinderWPF** получает результаты из **TwinFinder** и отображает их в пользовательском интерфейсе. Это может включать в себя динамическое добавление новых элементов в список результатов, а также отображение ошибок или уведомлений.
* **Удаление дубликатов**: Когда пользователь выбирает дубликаты для удаления, **TwinFinderWPF** отправляет команду компоненту **TwinFinder**, который обрабатывает удаление или перемещение файлов в корзину.

#### 4. ****Преимущества разделения на два компонента****

1. **Модульность**: Разделение приложения на два компонента позволяет сосредоточиться на каждой части отдельно, улучшая читаемость и поддержку кода. Логика поиска и удаления дубликатов не зависит от графического интерфейса.
2. **Переиспользуемость**: Логика из компонента **TwinFinder** может быть использована в других приложениях или даже в командных скриптах, без необходимости зависеть от интерфейса.
3. **Упрощение тестирования**: Компонент **TwinFinder** можно протестировать без привязки к UI, что облегчает процесс юнит-тестирования.
4. **Расширяемость**: В будущем можно легко обновлять или расширять функциональность графического интерфейса в **TwinFinderWPF**, не затрагивая логику работы с файлами и дубликатами.

3.3. Работа с приложением

Работа с приложением для поиска и удаления дубликатов файлов должна быть интуитивно понятной и удобной, обеспечивая пользователю максимально простой и эффективный процесс взаимодействия. Рассмотрим основные этапы работы с приложением, от его запуска до выполнения поиска и удаления дубликатов.

#### 1. ****Запуск приложения****

После запуска приложения, пользователь видит главное окно, которое содержит все необходимые элементы управления для работы с программой. Интерфейс должен быть максимально простым и функциональным, с четким распределением всех функций. Основные элементы интерфейса включают поле для выбора папки, кнопку для выбора папки, кнопку для начала поиска, блок для отображения результатов и кнопку для удаления выбранных дубликатов.

#### 2. ****Выбор директории для анализа****

На первом этапе работы с приложением пользователь должен выбрать директорию, в которой будет производиться поиск дубликатов файлов. Для этого предусмотрено поле ввода, в котором будет отображаться путь к выбранной директории, а также кнопка **"Browse"** для удобного выбора папки через стандартный диалог выбора директории.

Когда пользователь нажимает на кнопку **"Browse"**, открывается диалоговое окно, которое позволяет выбрать папку на локальном диске или в сетевых ресурсах. После выбора папки путь к директории автоматически отображается в текстовом поле.

**Пользовательский процесс:**

* Нажать на кнопку **"Browse"**.
* Открывается диалоговое окно для выбора папки.
* Выбрать нужную папку и нажать **OK**.
* Путь к папке отобразится в текстовом поле.

#### 3. ****Запуск процесса поиска дубликатов****

После того как пользователь выбрал папку для анализа, он может запустить процесс поиска дубликатов, нажав на кнопку **"Start Search"**.

При нажатии на кнопку программа начинает сканировать выбранную директорию и все вложенные папки, ища дубликаты файлов. Процесс поиска может занять некоторое время в зависимости от количества файлов и их размеров. На этом этапе пользователю стоит показать индикатор прогресса (например, через индикатор загрузки или проценты выполнения), чтобы он мог следить за процессом.

Поиск дубликатов осуществляется с использованием фильтров: сначала программа проверяет файлы на основе их размера и метаданных, а затем применяет хэш-функцию SHA-256 для точного сравнения содержимого файлов. Все файлы, которые окажутся идентичными по содержимому, будут сгруппированы в наборы дубликатов.

**Пользовательский процесс:**

* Нажать на кнопку **"Start Search"**.
* Дождаться завершения процесса сканирования.
* Видеть индикатор прогресса выполнения поиска.

#### 4. ****Отображение результатов поиска****

После завершения поиска результаты отображаются в основной части окна. Все найденные дубликаты сгруппированы по набору файлов, которые являются одинаковыми. В каждой группе дубликатов отображаются основные данные о файлах: имя, размер, путь к файлу, и, возможно, дополнительная информация (например, дата последнего изменения).

Каждый набор дубликатов представляется в виде списка файлов, которые пользователь может рассматривать и решать, какие из них оставить, а какие удалить. Также для удобства могут быть добавлены опции сортировки и фильтрации (например, по имени, размеру или дате изменения), что поможет пользователю быстрее ориентироваться среди большого количества файлов.

**Пользовательский процесс:**

* Просмотреть найденные дубликаты в области результатов.
* Изучить информацию о каждом файле: имя, размер, путь.
* Отсортировать или фильтровать результаты по различным критериям (если это предусмотрено).

#### 5. ****Удаление выбранных дубликатов****

Когда пользователь решит удалить дубликаты, он может выбрать файлы из списка. Для этого приложение должно предоставить удобные средства для выбора нескольких файлов, например, с помощью флажков или многократного выделения файлов с клавишей Ctrl или Shift.

После того как дубликаты выбраны, кнопка **"Delete Selected Files"** становится доступной для использования. Нажав на эту кнопку, пользователь может удалить выбранные файлы. Важно, чтобы приложение предупредило пользователя о том, что выбранные файлы будут удалены, и запросило подтверждение действия, чтобы избежать случайных удалений.

Удаленные файлы могут быть перемещены в корзину, чтобы их можно было восстановить в случае ошибки. Также можно предусмотреть возможность создания резервных копий перед удалением, чтобы обеспечить дополнительную безопасность.

**Пользовательский процесс:**

* Выбрать дубликаты для удаления.
* Нажать на кнопку **"Delete Selected Files"**.
* Подтвердить удаление файлов.
* Файлы будут удалены или перемещены в корзину.

#### 6. ****Обработка ошибок и уведомления****

Во время работы с приложением могут возникать ошибки, например, если некоторые файлы недоступны для чтения или не могут быть обработаны по другим причинам. В таких случаях приложение должно уведомлять пользователя о возникшей ошибке, не прерывая работу всего процесса. Например, если файл недоступен из-за недостатка прав, программа должна вывести уведомление с кратким описанием ошибки.

Важно, чтобы такие уведомления были информативными и понятными для пользователя. Приложение должно быть стабильным и не допускать критических сбоев в процессе работы.

**Пользовательский процесс:**

* В случае возникновения ошибки, пользователю будет выведено уведомление с описанием проблемы.
* Приложение продолжит работать, исключая недоступные файлы из обработки.

3.4. Выводы по главе 3

Приложение для поиска и удаления дубликатов файлов представляет собой удобный и функциональный инструмент, обеспечивающий простой и эффективный процесс работы с файлами. Разделение на два компонента — **TwinFinder** для логики работы с файлами и **TwinFinderWPF** для графического интерфейса — позволяет достичь высокой модульности и гибкости системы. Каждый этап работы с приложением, от выбора директории до удаления дубликатов, продуман с целью максимального удобства для пользователя, при этом обеспечивая надежную обработку ошибок и возможность подтверждения действий. В результате приложение предлагает пользователю интуитивно понятный интерфейс и высокую степень безопасности при выполнении операций с файлами.

ЗАКЛЮЧЕНИЕ

В результате разработки приложения для поиска и удаления дубликатов файлов, был создан эффективный инструмент, способный решать важные задачи, связанные с оптимизацией хранения данных и управлением файловыми системами. Проблема дублирования файлов, как переполнение дискового пространства, сложность управления файлами и ухудшение производительности, приобретает особую актуальность в условиях постоянно растущих объемов данных.

Разработанная система предоставляет пользователю интуитивно понятный графический интерфейс и реализует базовый функционал, направленный на сканирование, идентификацию и удаление дубликатов. Система использует современный подход, разделяя логику работы и пользовательский интерфейс, что улучшает модульность и расширяемость приложения.

Рассмотренные выводы по главе 1, такие как необходимость минимализма в функционале, фокус на удобстве интерфейса и модульности архитектуры, позволяют наметить успешный путь для дальнейшей разработки. В будущем можно будет добавить дополнительные возможности, такие как логирование или экспорт данных, но на данный момент основное внимание сосредоточено на качестве реализации ключевых задач.

Таким образом, данное приложение не только помогает пользователю оптимизировать использование дискового пространства, но и значительно упрощает процесс работы с большими объемами данных, снижая нагрузку на файловую систему и обеспечивая более эффективную работу с файлами.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. <https://doss.microsoft.com/ru-ru> - документация Mincrosoft [Электронный ресурс] – Режим доступа: <https://dosc.mizrosoft.com/ru-ru> - Дата доступа 01.12.2024
2. https://habr.com/ru/companies/pvs-studio/articles/773896/ [Электронный ресурс] – Что нового в .NET 8? – Режим доступа: https://habr.com/ru/companies/pvs-studio/articles/773896/ Дата доступа: 22.11.2021
3. https://metanit.com/sharp – C# и .NET [Электронный ресурс] – Режим доступа: https://metanit.com/sharp – Дата доступа: 01.11.2024
4. <https://habr.com/ru/articles/647259/> – Джентльменский набор для создания WPF-приложений [Электронный ресурс] – Режим доступа: <https://habr.com/ru/articles/647259/> – Дата доступа: 01.11.2024
5. <https://github.com/dotnet/wpf/tree/main> - документация WPF на гитхабе [Электронный ресурс] – Режим доступа: <https://github.com/dotnet/wpf/tree/main> - Дата доступа 01.12.2024