МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ АГРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ
ВІДДІЛЕННЯ «ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП'ЮТЕРНІ НАУКИ»
ЦИКЛОВА КОМІСІЯ СПЕЦІАЛЬНОСТІ «КОМП'ЮТЕРНІ НАУКИ»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи
освітньо-професійний ступінь «фаховий молодший бакалавр»
на тему:

Розробка програми «Сортування фото та відео по теках»

Виконав студент (ка) _4_ курсу, групи П-41
спеціальності 122 «Комп'ютерні науки»

Степанець Роман Степанович

Керівник Устименко Ярослав Іванович

Рецензент _____

Житомир – 2024 року

Зміст

РЕФЕРАТ

Записка: 50 стор., 26 рис., 1 додаток, 20 джерел.

Ключові слова: СОРТУВАННЯ ФОТО ТА ВІДЕО, ЦИФРОВІ ДАНІ, ЕФЕКТИВНІСТЬ ОРГАНІЗАЦІЇ, АВТОМАТИЗАЦІЯ ПРОЦЕСУ, МУЛЬТИМЕДІЙНІ ФАЙЛИ, WINDOWS PRESENTATION FOUNDATION (WPF), ІНТЕРФЕЙС КОРИСТУВАЧА, АЛГОРИТМИ СОРТУВАННЯ, ПОЧАТКОВА ТЕКА, ТИП ФАЙЛІВ.

Розробка програмного забезпечення для сортування фото та відео по теках є актуальною проблемою в сучасному цифровому світі, де кількість зображень та відеоматеріалів, які виробляються та зберігаються, постійно зростає. Завдяки розвитку цифрової фотографії та відеозапису, люди зберігають великі обсяги медіа-файлів на своїх пристроях та комп'ютерах. Проте, зберігати і організовувати ці файли може бути складною задачею, особливо коли розмір колекції стає значним.

Метою даної дипломної роботи є розробка програмного забезпечення, яке допоможе користувачам організовувати та сортувати свої фотографії та відео шляхом автоматичного розподілу їх за різними критеріями, такими як дата, час, місцезнаходження та інші. Дана програма буде надавати зручний та ефективний інтерфейс користувача, щоб спростити процес організації та управління великими обсягами медіа-файлів.

У ході роботи буде розглянуто різні аспекти розробки програмного забезпечення, включаючи вибір технологій, проектування інтерфейсу користувача, розробку функціональності, тестування та впровадження. Для досягнення поставленої мети буде використано сучасні технології розробки програмного забезпечення, такі як мова програмування C#, Windows Presentation Foundation (WPF), база даних SQLite, система керування версіями GitHub та середовище розробки Visual Studio.

Результатом цієї роботи буде створення функціонального та ефективного програмного забезпечення, яке допоможе користувачам

ефективно організовувати та управляти своїми медіа-колекціями. Така програма матиме потенціал стати корисним інструментом для широкого кола користувачів, які зіткнулися з проблемою управління великими обсягами фотографій та відео.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

GUI - Графічний інтерфейс користувача.

API - Інтерфейс програмування додатків.

UI - Інтерфейс користувача.

IDE - Інтегроване середовище розробки.

.NET - Платформа розробки програмного забезпечення Microsoft.

WPF - Windows Presentation Foundation.

SQL - Мова структурованих запитів.

CRUD - Create, Read, Update, Delete (створення, читання, оновлення, видалення).

MVVM - Model-View-ViewModel.

ORM - Об'єктно-реляційне відображення.

VCS - Система керування версіями.

API - Інтерфейс програмування додатків.

URI - Уніфікований ідентифікатор ресурсу.

SDK - Набір розробника програмного забезпечення.

JPEG - Група експертів з фотографії.

PNG - Портативна мережева графіка.

MVC - Модель-вид-контролер.

CDN - Мережа доставки контенту.

XML - Розширюваний мовний опис.

URI - Уніфікований ідентифікатор ресурсу.

ВСТУП

У наш час інформаційних технологій, коли цифрові дані стають неухильною частиною нашого повсякденного життя, проблема ефективного управління цими даними набуває особливого значення. Особливо це стосується мультимедійних файлів, таких як фотографії та відеозаписи, які займають все більше місця в наших цифрових архівах.

З надлишком фотографій та відео на наших пристроях зберігання, часто виникає проблема організації та управління цими медіа-ресурсами. Необхідність у системі, яка допомагатиме структурувати та розміщувати ці дані за критеріями, стає все більш актуальною.

Мета цієї дипломної роботи полягає в розробці програмного забезпечення, яке автоматизує процес сортування та організації фотографій та відео на основі їх характеристик. Основними завданнями є дослідження методів класифікації мультимедійних файлів, розробка алгоритмів сортування та реалізація програмного рішення.

Предметом розробки є процес автоматизації сортування мультимедійних файлів (фото та відео) у відповідні теки на основі їх типу (розширення файлів). Це включає в себе алгоритми для виявлення файлів, визначення їх типу, створення структур папок та переміщення файлів у відповідні папки.

Об'єктом розробки є програмне забезпечення (десктопна програма), яке реалізовано з використанням Windows Presentation Foundation (WPF) для операційної системи Windows. Це програмне забезпечення має зручний інтерфейс користувача, який дозволяє користувачам вибирати папки з файлами, запускати процес сортування та переглядати статус виконання операцій.

Розробка програми "Сортування фото та відео по теках" проводиться у середовищі Visual Studio з використанням платформи Windows Presentation Foundation (WPF). Основні етапи розробки включають:

Проектування інтерфейсу користувача (UI) з елементами для вибору папки, перегляду файлів і запуску процесу сортування.

Реалізація логіки сортування, що включає виявлення файлів у вибраній папці, визначення їх типу на основі розширення, створення відповідних папок та переміщення файлів.

Тестування програми для забезпечення її стабільної роботи та виправлення можливих помилок.

Результатом роботи є десктопна програма, яка дозволяє користувачам легко та швидко сортувати свої фото та відео по відповідних теках. Програма створює у вибраній користувачем папці дві підпапки: "Photos" та "Videos", і переміщує відповідні файли до цих папок. Це дозволяє значно зменшити час та зусилля, необхідні для організації мультимедійних архівів.

Програма "Сортування фото та відео по теках" є ефективним інструментом для автоматизації процесу організації мультимедійних файлів. Використання сучасних технологій, таких як WPF, дозволяє забезпечити зручний інтерфейс користувача та високу продуктивність програми. Розроблене програмне забезпечення може бути корисним як для звичайних користувачів, так і для професіоналів, що працюють з великими обсягами цифрових даних.

Дослідження цієї проблеми має важливе значення для практичного застосування в сучасному цифровому середовищі. Потенційні користувачі програми включають як звичайних користувачів, що шукають зручний спосіб організації своїх медіа-колекцій, так і професіоналів, які потребують ефективного управління великими обсягами фотографій та відео.

У даному вступі буде здійснено аналіз актуальності проблеми, визначено мету та завдання дипломної роботи, охарактеризовано предмет та об'єкт дослідження, а також надано короткий огляд структури подальшого викладу матеріалу.

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ

1.1. Постановка основної задачі розробки

Постановка основної задачі розробки передбачає визначення конкретної мети, яку необхідно досягти у процесі розробки програмного забезпечення "Сортування фото та відео по теках". Основна задача розробки полягає у створенні ефективного та надійного інструменту, що автоматизує процес сортування мультимедійних файлів за їх типом та іншими характеристиками. Конкретні аспекти постановки основної задачі розробки включають:

Аналіз вимог: Детальний аналіз потреб користувачів та вимог до програмного забезпечення для визначення функціональних та нефункціональних вимог.

Розробка алгоритмів: Розробка алгоритмів сортування, які враховують різні типи медіа-файлів та їх характеристики.

Створення програми: Реалізація програмного забезпечення з використанням сучасних технологій розробки програм.

Тестування та валідація: Проведення тестів для перевірки коректності роботи програми та відповідності вимогам.

Оптимізація та підтримка: Забезпечення оптимізації програми для забезпечення її ефективності та підтримки для користувачів у разі виникнення проблем.

Ці кроки в постановці основної задачі розробки допоможуть забезпечити успішну реалізацію програмного забезпечення та задовольнення потреб користувачів у сфері організації цифрових медіа-файлів.

1.2. Огляд та порівняння аналогічних систем.

Огляд та порівняння аналогічних систем є важливим етапом при розробці програмного забезпечення "Сортування фото та відео по теках", оскільки це дозволяє краще зрозуміти потреби користувачів та визначити переваги та недоліки існуючих рішень.

Adobe Lightroom: Ця програма від Adobe пропонує широкий спектр інструментів для редагування та організації фотографій. Вона має потужні функції сортування за різними критеріями, але вона орієнтована на професійних фотографів та має високу вартість.

Google Photos: Це безкоштовна онлайн-платформа для зберігання та організації фото та відео. Google Photos має автоматизовану систему сортування та розпізнавання облич, а також інтегровану з різними пристроями. Однак, у нього є обмеження на обсяг зберігання фотографій у високій якості.

Apple Photos: Ця програма для macOS та iOS пропонує інтегрований досвід зберігання та організації фото та відео для користувачів пристроїв Apple. Вона автоматично організовує фотографії за часом та місцезнаходженням, а також має функції розпізнавання облич та об'єктів.

ACDSee: Ця програма для Windows пропонує широкий спектр функцій для організації та редагування фотографій. Вона має простий інтерфейс та швидку роботу, але може бути менш потужною за Adobe Lightroom.

Flickr: Це інтернет-платформа для зберігання та обміну фото та відео. Flickr має ряд функцій сортування та тегування фотографій, а також можливість обміну з іншими користувачами. Однак, це не така потужна система, як Adobe Lightroom чи ACDSee.

Порівнявши ці системи, можна виділити, що кожна з них має свої переваги та недоліки. Для розробки програмного забезпечення "Сортування фото та відео по теках" важливо врахувати ці особливості та забезпечити програму якісними та конкурентноздатними функціями.

Давайте порівняємо п'ять аналогічних систем для організації та сортування фото та відео:

Adobe Lightroom:

Переваги: Потужні функції редагування та організації, широкі можливості роботи з RAW-файлами, інтеграція з іншими Adobe-продуктами.

Недоліки: Висока вартість, великі вимоги до обладнання та ресурсів комп'ютера, складний інтерфейс для новачків.

Google Photos:

Переваги: Безкоштовне зберігання для фото та відео з обмеженнями, автоматизоване сортування та розпізнавання облич.

Недоліки: Обмежена можливість редагування, компресія фотографій при зберіганні у високій якості.

Apple Photos:

Переваги: Інтегрований з пристроями Apple, автоматична організація за часом та місцезнаходженням, розпізнавання облич та об'єктів.

Недоліки: Обмежені можливості на Windows, обмежені функції редагування порівняно з іншими програмами.

ACDSee:

Переваги: Простий та швидкий інтерфейс, багато функцій для організації та редагування фотографій.

Недоліки: Менш потужна у порівнянні з Adobe Lightroom, деякі функції можуть бути менш ефективними.

Flickr:

Переваги: Можливість обміну фотографіями з іншими користувачами, деякі функції сортування та тегування.

Недоліки: Менша потужність у порівнянні з іншими програмами, обмежена кількість функцій для організації.

Порівнюючи ці системи, можна сказати, що кожна має свої переваги та недоліки. Вибір системи залежить від потреб користувача: від професіоналів,

які шукають високу якість обробки та організації, до звичайних користувачів, яким важлива простота використання та доступність.

1.3. Вибір та обґрунтування технологій розробки

Для розробки програмного забезпечення "Сортування фото та відео по теках" важливо обрати технології, які найкраще відповідають вимогам проекту, забезпечують швидкий розвиток та підтримку, а також сприяють створенню високоякісного та ефективного продукту. Ось деякі обґрунтування вибору конкретних технологій:

Мова програмування: Використання мови програмування C# обґрунтоване тим, що це потужна та широко використовувана мова програмування, яка підтримується платформою .NET Framework. C# забезпечить ефективність розробки та можливість інтеграції з іншими технологіями Microsoft.

Windows Presentation Foundation (WPF): Використання WPF для розробки графічного інтерфейсу користувача обґрунтоване його потужними можливостями створення багатофункціональних та естетичних інтерфейсів. WPF забезпечить швидку та ефективну розробку UI, а також можливість створення анімацій та перехідних ефектів. Windows Presentation Foundation (WPF) - це технологія розробки програмного забезпечення, розроблена компанією Microsoft для створення багатофункціональних та графічно збагачених додатків для операційної системи Windows. Ось деякі ключові характеристики WPF:

Графічна мережа: WPF використовує графічну мережу для створення графічного інтерфейсу користувача (GUI), що дозволяє розробникам створювати складні та естетично привабливі додатки.

Декларативна розмітка і стиль: WPF використовує XAML (Extensible Application Markup Language) для опису структури та вигляду UI елементів. Це дозволяє відокремити дизайн від логіки програми та спрощує розробку.

Вбудовані анімації та ефекти: WPF надає можливість створення анімацій та перехідних ефектів безпосередньо в XAML. Це дозволяє створювати динамічні та привабливі візуальні ефекти для додатків.

Відмінна підтримка векторної графіки: WPF підтримує векторну графіку, що дозволяє створювати високоякісні та масштабовані зображення, які зберігають якість при будь-якому масштабуванні.

Динамічні ресурси та стилі: WPF дозволяє використовувати динамічні ресурси та стилі, що полегшує створення змінних та легко змінюваних елементів дизайну.

Інтеграція зі средою Visual Studio: WPF інтегрований зі середою розробки Visual Studio, що спрощує розробку, відлагодження та підтримку додатків.

Загалом, WPF є потужною технологією для розробки графічних додатків для операційної системи Windows, яка дозволяє створювати сучасні та естетичні інтерфейси користувача з великою гнучкістю та продуктивністю.

Висновки до розділу 1

У розділі "Обгрунтування вибору технологій розробки" було проведено аналіз різних аспектів технологій, які використовуються для створення програмного забезпечення "Сортування фото та відео по теках". На основі цього аналізу можна зробити декілька висновків:

Ефективність використання: Вибрані технології (мова програмування C#, Windows Presentation Foundation (WPF), база даних SQLite, GitHub для керування версіями та Visual Studio для розробки) є ефективними для реалізації функціональності програмного забезпечення з урахуванням вимог проекту.

Спрощення розробки: Обрані технології сприяють спрощенню процесу розробки, оскільки вони надають широкий набір інструментів, готових бібліотек та ресурсів для розробників.

Інтеграція та підтримка: Використання відомих та популярних технологій також сприяє легкості інтеграції з іншими системами та забезпечує підтримку та оновлення у майбутньому.

Безпека та стабільність: Врахування вибору технологій, таких як SQLite та GitHub, сприяє забезпеченню безпеки та стабільності програмного забезпечення.

Отже, вибір обгрунтованих технологій є доцільним та відповідає вимогам та цілям проекту, що сприятиме успішній реалізації програми "Сортування фото та відео по теках" з високою якістю та продуктивністю.

РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА

2.1. Вибір алгоритмів та їх ефективність

Програма використовує бібліотеку ExifLibNet для .NET, яка дозволяє працювати з метаданими Exif у зображеннях. Вона надає зручні методи для читання та запису метаданих Exif, що зберігаються у файлах зображень. Exif (Exchangeable Image File Format) — це стандарт, що визначає формат метаданих для зображень, записаних цифровими камерами та іншими пристроями. Бібліотека працює з різними форматами зображень, включаючи JPEG та TIFF.

Бібліотека дозволяє зчитувати різноманітні метадані Exif, такі як дата та час створення зображення, геолокаційні дані, налаштування камери (витримка, діафрагма, ISO), і багато іншого.

Крім читання, ExifLibNet дозволяє змінювати та записувати метадані Exif у файлах зображень.

ExifLibNet надає простий і зручний API для доступу до метаданих. API включає класи та методи для отримання та встановлення значень Exif тегів.

ExifLibNet є потужним інструментом для розробників, які потребують доступу до метаданих зображень у своїх .NET додатках. Завдяки цій бібліотеці можна легко отримувати і маніпулювати Exif даними, що зберігаються в цифрових фотографіях

Також програма використовує бібліотеку **Ookii.Dialogs.Wpf** для платформи .NET, яка надає зручні класи для відображення стандартних діалогових вікон Windows у WPF (Windows Presentation Foundation) додатках. Вона є розширенням для бібліотеки **Ookii.Dialogs**, призначеної для роботи з Windows Forms, і забезпечує підтримку WPF.

Підтримує такі діалогові вікна, як вибір файлу, вибір папки, повідомлення, прогрес і багато інших. Забезпечує просту інтеграцію з WPF додатками, дозволяючи використовувати стандартні діалоги Windows у сучасних WPF інтерфейсах. Автоматичне визначення теми діалогу (світла/темна), підтримка асинхронних операцій, кастомізація і т.д.

Основні класи бібліотеки:

- VistaOpenFileDialog – Використовується для відкриття файлу. Підтримує розширені можливості вибору файлів, такі як вибір кількох файлів, фільтрація за типом файлу тощо.
- VistaFolderBrowserDialog – Використовується для вибору папки. Дозволяє користувачу вибрати папку з файлової системи.
- TaskDialog – Потужний клас для створення діалогових вікон з багатими можливостями, такими як кнопки, радіо-кнопки, прапорці, прогрес-бар і т.д.

Для робити з форматом JSON я використав бібліотеку Newtonsoft.Json, також відома як Json.NET, — це популярна бібліотека для роботи з JSON у .NET. Вона забезпечує простий і ефективний спосіб серіалізації та десеріалізації об'єктів у формат JSON, а також надає розширені можливості для роботи з JSON-даними. Бібліотека дозволяє виконувати серіалізацію та десеріалізацію об'єктів .NET у JSON та назад. Дозволяє працювати з JSON як з об'єктною моделлю за допомогою LINQ. Можливість налаштовувати формат JSON, включаючи відступи, культуру, формат дати та інше. Використання атрибутів для контролю над тим, як об'єкти серіалізуються та десеріалізуються.

Основні класи бібліотеки:

- JsonConvert – Основний клас для серіалізації та десеріалізації JSON.
- JObject – Клас для роботи з JSON-об'єктами. Надає можливість динамічного доступу до властивостей.

Бібліотека Newtonsoft.Json є однією з найбільш популярних бібліотек для роботи з JSON у .NET завдяки своїй простоті, гнучкості та широким можливостям.

2.2. Опис класів

Програма використовує об'єктно-орієнтоване програмування. Для роботи були написані класи (Рисунок 2.2.1).

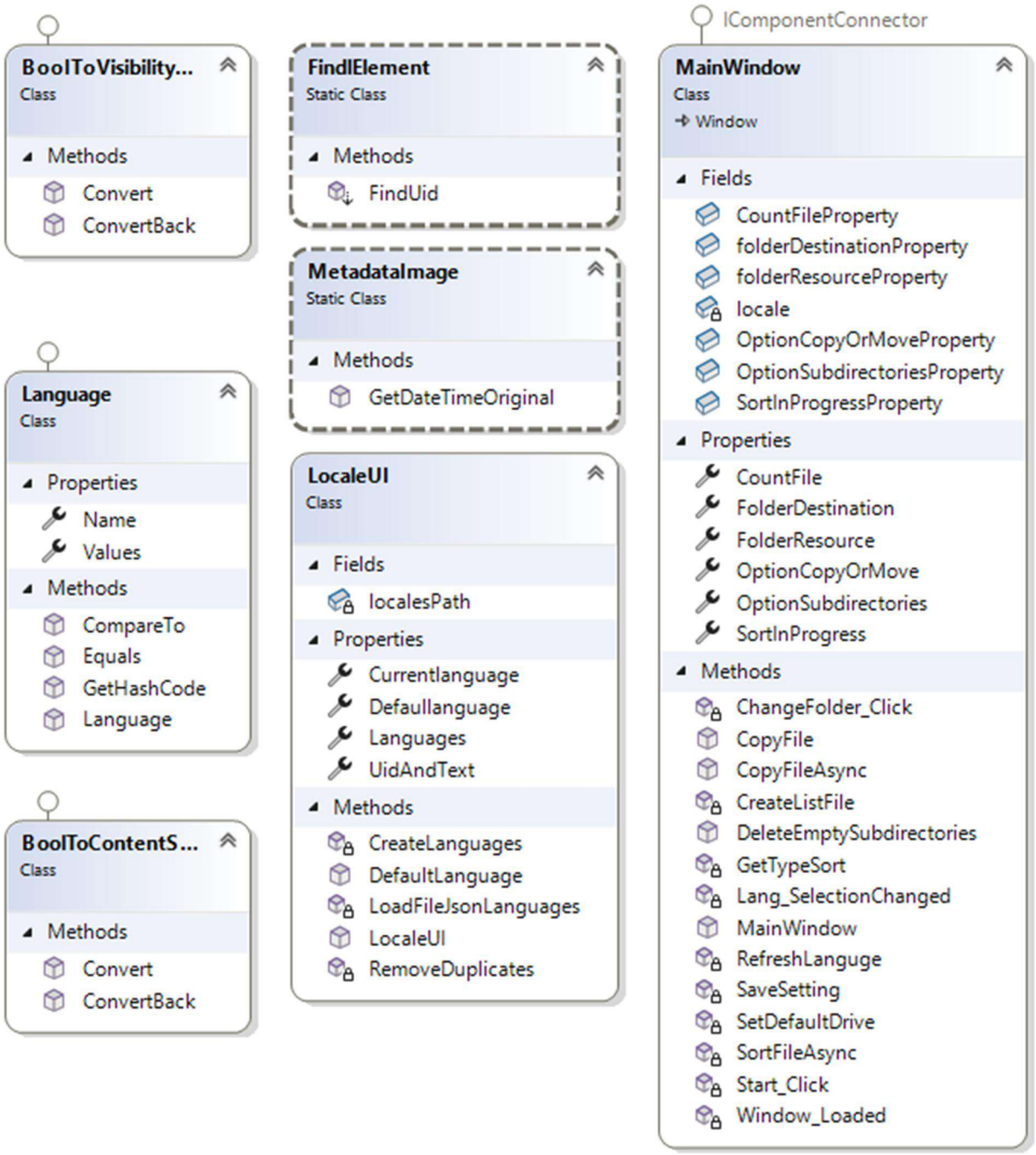
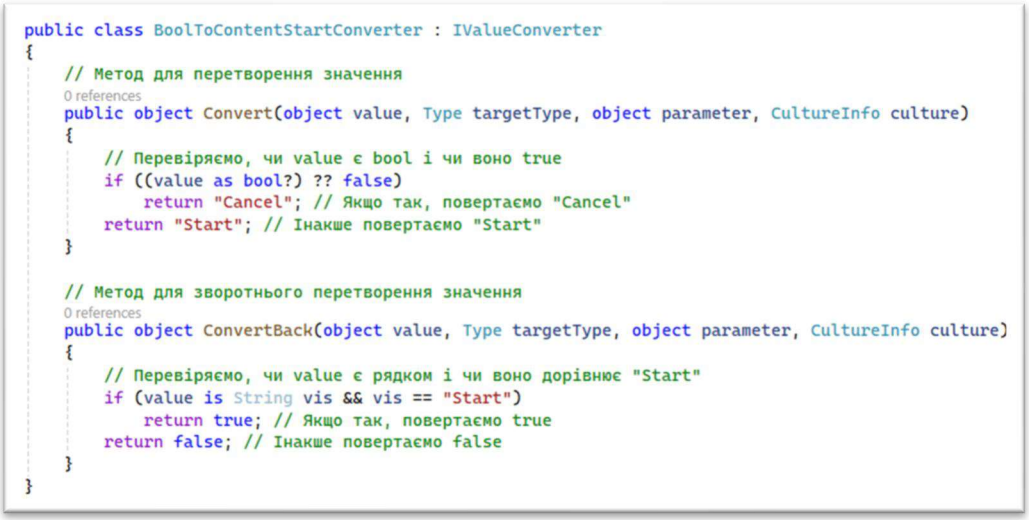


Рисунок 2.2.1 – Діаграма класів програми

Цей клас BoolToContentStartConverter (Рисунок 2.2.2) реалізує інтерфейс IValueConverter, який використовується для перетворення значень в рамках прив'язки даних в WPF. Він має два методи: Convert та ConvertBack.

Метод Convert перевіряє, чи передане значення (value) є булевим і чи воно true. Якщо так, то повертається рядок "Cancel", в іншому випадку - "Start".

Метод ConvertBack здійснює зворотнє перетворення: якщо передане значення (value) є рядком "Start", то повертається true, в іншому випадку - false.



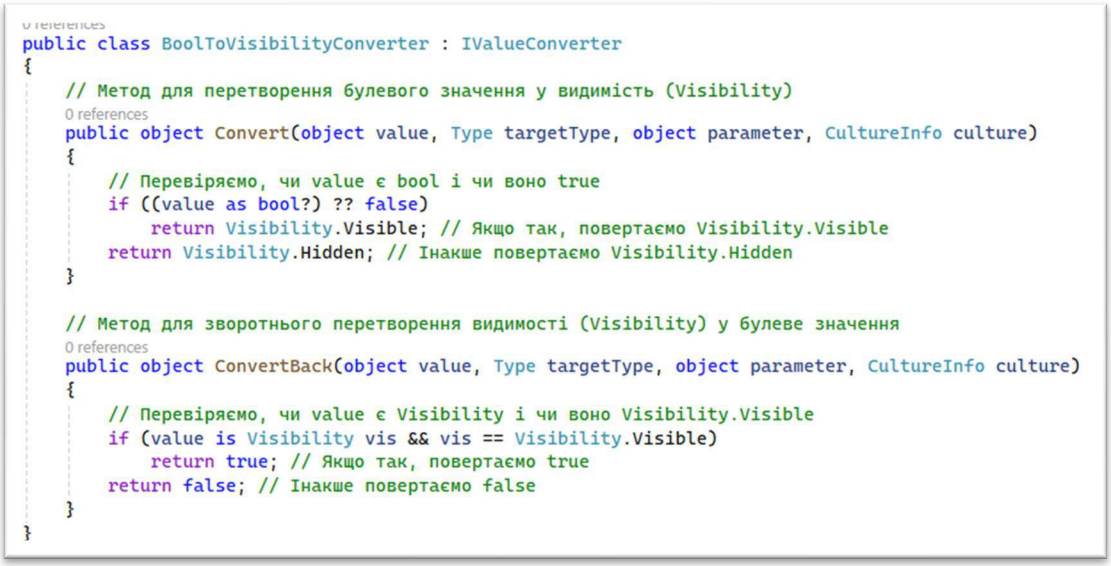
```
public class BoolToContentStartConverter : IValueConverter
{
    // Метод для перетворення значення
    0 references
    public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
    {
        // Перевіряємо, чи value є bool і чи воно true
        if ((value as bool?) ?? false)
            return "Cancel"; // Якщо так, повертаємо "Cancel"
        return "Start"; // Інакше повертаємо "Start"
    }

    // Метод для зворотнього перетворення значення
    0 references
    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
    {
        // Перевіряємо, чи value є рядком і чи воно дорівнює "Start"
        if (value is String vis && vis == "Start")
            return true; // Якщо так, повертаємо true
        return false; // Інакше повертаємо false
    }
}
```

Рисунок 2.2.2 – Клас BoolToContentStartConverter

Клас BoolToVisibilityConverter (Рисунок 2.2.3) реалізує інтерфейс IValueConverter, який використовується для перетворення булевих значень у видимість елементів у WPF. Він містить два методи: Convert та ConvertBack.

Метод Convert перевіряє, чи передане значення (value) є булевим і чи воно true. Якщо так, то повертається Visibility.Visible, в іншому випадку - Visibility.Hidden.

Метод ConvertBack здійснює зворотнє перетворення: якщо передане значення (value) є Visibility.Visible, то повертається true, в іншому випадку - false



```
0 references
public class BoolToVisibilityConverter : IValueConverter
{
    // Метод для перетворення булевого значення у видимість (Visibility)
    0 references
    public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
    {
        // Перевіряємо, чи value є bool і чи воно true
        if ((value as bool?) ?? false)
            return Visibility.Visible; // Якщо так, повертаємо Visibility.Visible
        return Visibility.Hidden; // Інакше повертаємо Visibility.Hidden
    }

    // Метод для зворотнього перетворення видимості (Visibility) у булеве значення
    0 references
    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
    {
        // Перевіряємо, чи value є Visibility і чи воно Visibility.Visible
        if (value is Visibility vis && vis == Visibility.Visible)
            return true; // Якщо так, повертаємо true
        return false; // Інакше повертаємо false
    }
}
```

Рисунок 2.2.3 – Клас BoolToVisibilityConverter

Статичний клас FindElement (Рисунок 2.2.4) містить метод розширення FindUid, який дозволяє знаходити елемент користувацького інтерфейсу (UIElement) за його унікальним ідентифікатором (UID) у дереві логічних елементів WPF. Метод FindUid приймає два параметри: батьківський елемент (parent) та шуканий UID (uid). Використовуючи LogicalTreeHelper.GetChildren, метод проходить по всім дочірнім елементам типу UIElement. Якщо UID поточного елемента збігається з шуканим UID, цей елемент повертається. Якщо ні, то метод рекурсивно викликається для кожного дочірнього елемента. Якщо жоден елемент з шуканим UID не знайдено, метод повертає null.

```
1 reference
public static class FindlElement
{
    // Метод для пошуку елемента за UID в дереві логічних елементів
    2 references
    public static UIElement FindUid(this DependencyObject parent, string uid)
    {
        // Проходимо по всім дочірнім елементам типу UIElement
        foreach (UIElement element in LogicalTreeHelper.GetChildren(parent).OfType<UIElement>())
        {
            // Якщо UID елемента співпадає з шуканим UID, повертаємо цей елемент
            if (element.Uid == uid)
                return element;

            // Рекурсивно шукаємо в дочірніх елементах
            UIElement resultChildren = FindUid(element, uid);
            if (resultChildren != null)
                return resultChildren;
        }
        // Якщо не знайдено жодного елемента з шуканим UID, повертаємо null
        return null;
    }
}
```

Рисунок 2.2.4 – Клас FindElement

Клас Language (Рисунок 2.2.5) реалізує інтерфейси IEquatable<Language> та IComparable<Language>, що дозволяє порівнювати об'єкти цього класу та перевіряти їх на рівність.

Поля класу:

- Name – назва мови, визначається при створенні об'єкта і є доступною тільки для читання.
- Values – словник, де ключем є назва елемента для перекладу, а значенням — перекладений текст.

Методи класу:

- CompareTo(Language other) – порівнює назви мов для сортування.

- Equals(Language other) – перевіряє рівність об'єктів на основі їх назв.
- GetHashCode() – повертає хеш-код назви мови для використання в хеш-таблицях.

```
13 references
public class Language : IEquatable<Language>, IComparable<Language>
{
    // Назва мови
    7 references
    public string Name { get; private set; }
    // Словник з написами цією мовою
    // ключ - назва елемента, що перекладається
    // значення - текст
    3 references
    public Dictionary<string, string> Values { get; private set; }
    // Конструктор для ініціалізації мови та її значень
    1 reference
    public Language(string name, Dictionary<string, string> values)...
    // Метод порівняння для сортування за назвою мови
    1 reference
    public int CompareTo(Language other)...
    // Метод для перевірки рівності двох об'єктів Language
    0 references
    public bool Equals(Language other)...
    // Перевизначення GetHashCode для коректної роботи з хеш-таблицями
    0 references
    public override int GetHashCode()...
}
```

Рисунок 2.2.5 – Клас Language

Цей клас LocaleUI (Рисунок 2.2.6) відповідає за керування мовними налаштуваннями інтерфейсу користувача. Він містить список мов, поточну мову та мову за замовчуванням, а також шлях до файлу з мовними налаштуваннями. Методи класу дозволяють завантажувати JSON файл з мовами, створювати список мов з JSON, встановлює мову за замовчуванням, видаляє дублікати мов. Містить статичний словник з текстами за замовчуванням для елементів інтерфейсу.

Рисунок 2.2.6 – Клас LocaleUI

Клас MetadataImage надає статичний метод GetDateTimeOriginal, який дозволяє отримувати властивість Exif зображення за вказаним тегом exifTag та шляхом до файлу filePath (Рисунок 2.2.7). Метод GetDateTimeOriginal повертає ExifProperty, якщо метадані успішно витягнуті, або null, якщо виникла помилка. Використовується блок try-catch для обробки можливих винятків при завантаженні файлу або отриманні метаданих. У разі виникнення винятку повідомлення про помилку записується у відладку за допомогою Debug.WriteLine.

```
1 reference
public static class MetadataImage
{
    // Метод для отримання властивості Exif за тегом exifTag з вказаного файлу
    1 reference
    public static ExifProperty GetDateTimeOriginal(ExifTag exifTag, string filePath)
    {
        ExifProperty exifProperty = null;
        try
        {
            // Витягуємо Exif метадані з файлу
            var file = ImageFile.FromFile(filePath);
            exifProperty = file.Properties.Get(exifTag);
        }
        catch (Exception ex)
        {
            Debug.WriteLine(ex.Message);
        }

        return exifProperty;
    }
}
```

Рисунок 2.2.7 – Клас MetadataImage

2.3. Програмна реалізація

Програма має чітку структуру. Класи розміщено в папку Clases. Файл для з набором мови інтерфейсу збережено до папки Lang. Головне вікно зберігається у папці Window.

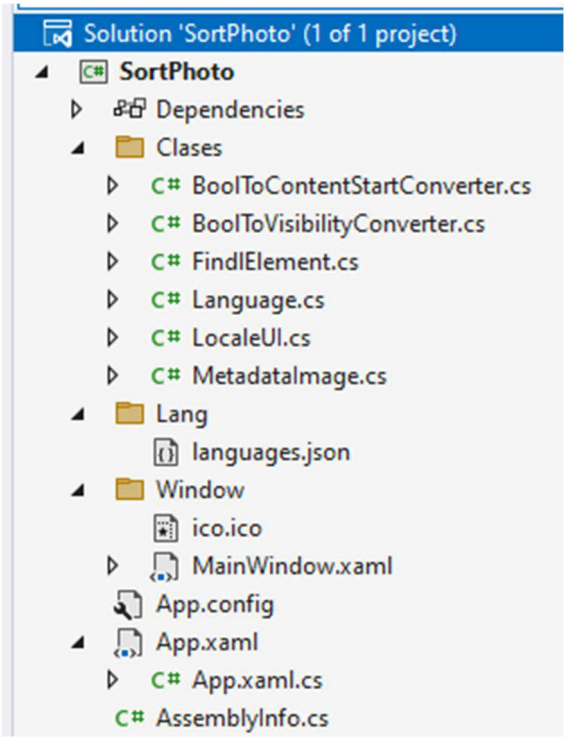


Рисунок 2.3.1 – Структура проєкту

Головне вікно використовує розмітку XAML()

```
Window (Window_1)                                                    Window (Window_1)
1   <Window
2       x:Class="SortPhoto.MainWindow"
3       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5       xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
6       xmlns:local="clr-namespace:SortPhoto"
7       xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8       x:Uid="Window_1"
9       Title="SortPhoto"
10      Width="640"
11      Height="380"
12      MinWidth="640"
13      MinHeight="380"
14      MaxHeight="380"
15      FontFamily="Segoe UI"
16      WindowStartupLocation="CenterScreen"
17      WindowStyle="SingleBorderWindow"
18      mc:Ignorable="d" Loaded="Window_Loaded" Icon="/Window/ico.ico">
19      <Window.Resources>
20          <local:BoolToVisibilityConverter x:Key="BoolToVisConverter" x:Uid="BoolToVisibilityConverter" />
21          <local:BoolToContentStartConverter x:Key="BoolToContStartConverter" x:Uid="BoolToContStartConverter" />
22      </Window.Resources>
23      <Grid x:Uid="Grid_1">
24          <Grid.RowDefinitions>
25              <RowDefinition x:Uid="RowDefinition_1" Height="auto" />
26              <RowDefinition x:Uid="RowDefinition_5" Height="auto" />
27              <RowDefinition x:Uid="RowDefinition_2" Height="auto" />
28              <RowDefinition x:Uid="RowDefinition_3" Height="auto" />
29              <RowDefinition x:Uid="RowDefinition_4" Height="auto" />
30              <RowDefinition x:Uid="RowDefinition_6" Height="auto" />
```

Рисунок 2.3.2 – Розмітка головного вікна

Вигляд головного вікна в конструкторі представлено на рисунку 2.3.3

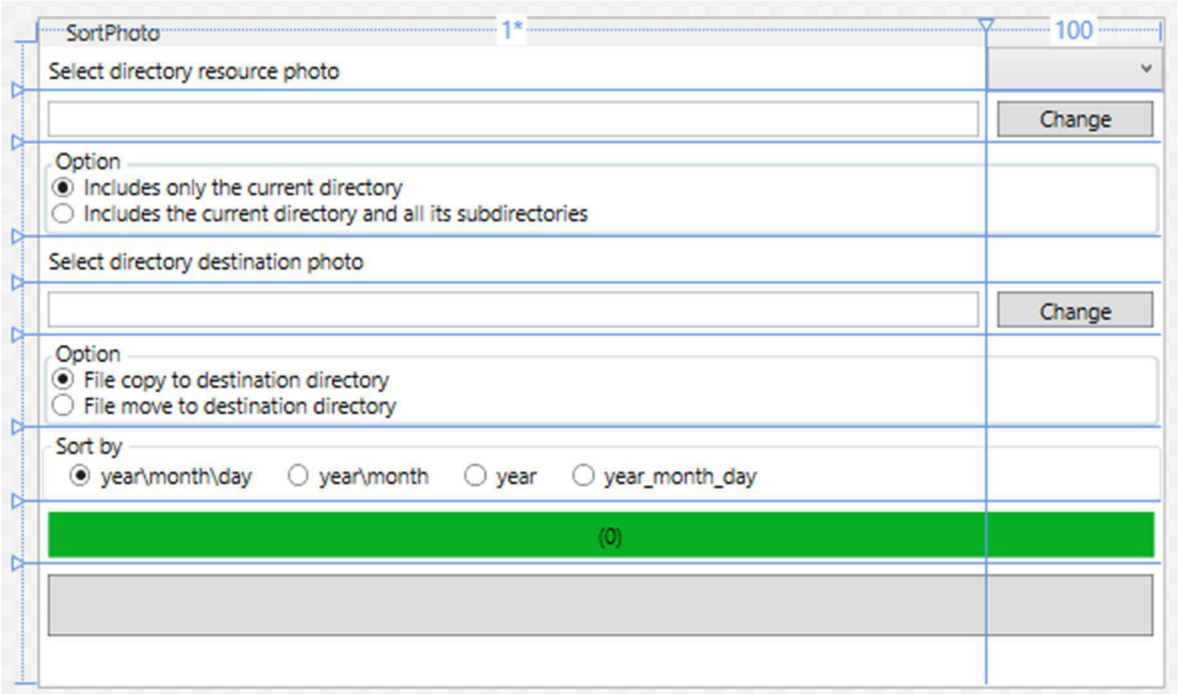


Рисунок 2.3.3 – Вигляд головного вікна в конструкторі

Для зручного перекладу інтерфейсу використовується файл languages.json (Рисунок 2.3.4). Його можна дописати на будь яку мову світу без перекомпіляції програми.

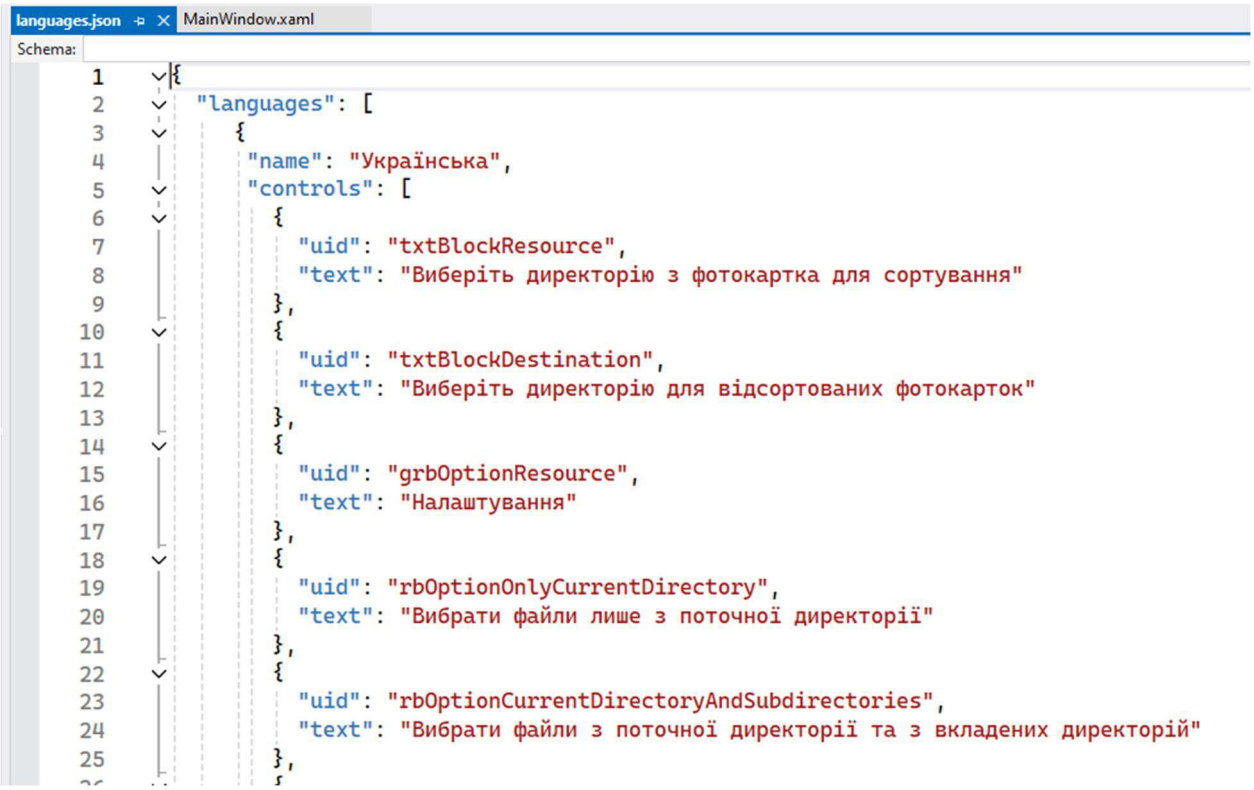


Рисунок 2.3.4 – Структура файлу languages.json

2.4. Тестування програми

Програма була ретельно протестована на операційних системах Windows та Ubuntu, що забезпечує її стабільну роботу на цих платформах. Завдяки використанню технологій .NET програма виконує свої функції ефективно та надійно, забезпечуючи високу продуктивність та сумісність з різними версіями операційних систем.

Інтерфейс програми розроблений таким чином, щоб уникнути помилок користувача. Наприклад, користувач не може вибрати неіснуючий шлях до теки, оскільки інтерфейс автоматично перевіряє введені дані та вказує на будь-які помилки. Це забезпечує зручність та надійність використання програми.

Програма здатна зчитувати інформацію про дату зйомки з файлу, використовуючи метадані Exif, що зберігаються в зображеннях. Однак, не всі файли містять цю інформацію. У випадку, коли метадані Exif відсутні, програма автоматично використовує дату створення файлу, що дозволяє зберегти коректний порядок сортування.

Програма призначена для сортування різних типів файлів, включаючи зображення, відео та інші мультимедійні файли. Вона аналізує метадані файлів та організовує їх відповідно до вказаних критеріїв, таких як дата створення або дата зйомки. Це робить програму універсальним інструментом для впорядкування медіафайлів у будь-якому каталозі.

Додатково, програма підтримує різні режими сортування, що дозволяє користувачам налаштовувати її під свої потреби. Наприклад, можна сортувати файли за роком, місяцем, днем або іншими параметрами. Це забезпечує гнучкість та зручність у використанні, дозволяючи користувачам легко знаходити необхідні файли та організовувати свою медіатеку в оптимальний спосіб.

Висновки до розділу 2

У цьому розділі було детально розглянуто процес проектування та розробки програми для сортування файлів, яка має на меті забезпечити зручне та ефективне впорядкування медіафайлів. Ми розглянули ключові аспекти проектування інтерфейсу користувача, архітектури програми, а також основні функціональні можливості, які дозволяють користувачам з легкістю організовувати свої файли.

Програма була успішно протестована на операційних системах Windows та Ubuntu. Завдяки використанню технологій .NET, забезпечується висока продуктивність та стабільність роботи на цих платформах.

Інтерфейс програми розроблений з урахуванням принципів зручності та інтуїтивної зрозумілості. Було забезпечено можливість вибору лише існуючих шляхів до тек, що зменшує кількість помилок користувача та підвищує надійність роботи програми.

Програма здатна зчитувати інформацію про дату зйомки з метаданих Exif у файлах зображень. У випадку, коли ці дані відсутні, використовується дата створення файлу. Це забезпечує точність сортування навіть у відсутності повних метаданих.

Програма не обмежується лише зображеннями, вона також підтримує сортування відео та інших мультимедійних файлів, що робить її універсальним інструментом для впорядкування медіафайлів.

Користувачі можуть налаштовувати параметри сортування відповідно до своїх потреб, вибираючи різні критерії, такі як рік, місяць, день та інші. Це забезпечує високу гнучкість та адаптивність програми.

Процес проектування та розробки цієї програми показав важливість комплексного підходу до створення інструментів для роботи з файлами. Була досягнута головна мета – створити ефективну та зручну програму, яка відповідає потребам користувачів у впорядкуванні медіафайлів. Усі ключові функції, такі як обробка метаданих, підтримка різних типів файлів та надійність інтерфейсу, були успішно реалізовані.

Цей розділ описує фундаментальні етапи розробки, починаючи від початкового проектування і до завершення розробки функціональних можливостей програми. Отримані результати можуть бути основою для подальших удосконалень та розвитку програмного продукту, враховуючи зворотний зв'язок від користувачів та нові технологічні можливості.

РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАННЯ ПРОГРАМОЮ

3.1. Мінімальні вимоги до системи

Для запуску програми необхідно враховувати мінімальні вимоги до комп'ютера. Ці вимоги можуть варіюватися залежно від конкретних функціональних можливостей програми та її ресурсомісткості

Для архітектура x64 (рекомендовано процесори Intel або AMD з мінімальною тактовою частотою 1.6 ГГц).

Мінімум 2 ГБ оперативної пам'яті (рекомендовано 4 ГБ або більше для кращої продуктивності).

Мінімум 1 ГБ вільного місця на диску для встановлення .NET 8 та необхідних компонентів програми.

Відеокарта з підтримкою DirectX 9 або новіше з драйвером WDDM 1.0 або новіше (для WPF необхідна підтримка апаратного прискорення).

Windows 10 версії 1607 (Anniversary Update) або новіше може бути Ubuntu 20.04 LTS або новіше (для роботи на Linux).

.NET 8 Runtime (необхідно завантажити та встановити відповідну версію .NET Runtime з офіційного сайту .NET).

Роздільна здатність дисплея не менше 1280x720 для коректного відображення інтерфейсу програми.

Виконання цих мінімальних вимог забезпечить належну продуктивність та стабільність роботи програми на більшості сучасних комп'ютерів. Важливо регулярно оновлювати операційну систему та інші програмні компоненти для підтримки безпеки та сумісності.

3.2. Інтерфейс користувача

Програма запускається за допомогою файлу SortPhoto.exe з папки . Для роботи програми потрібно мати додаткові бібліотеки(Рисунок 3.2.1)

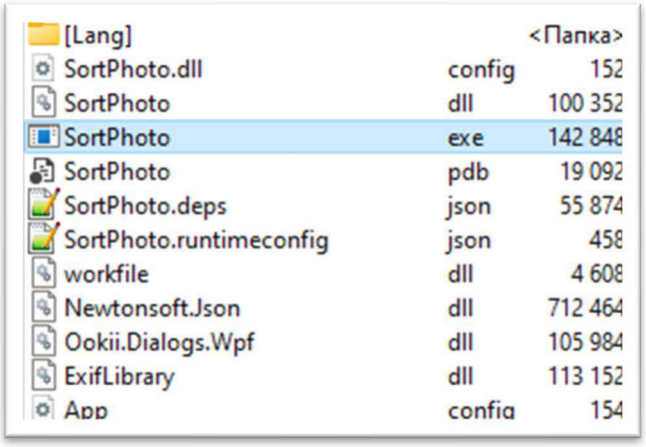


Рисунок 3.2.1 – Вміст папки з програмою

Після запуску програма має український інтерфейс (Рисунок 3.2.2).

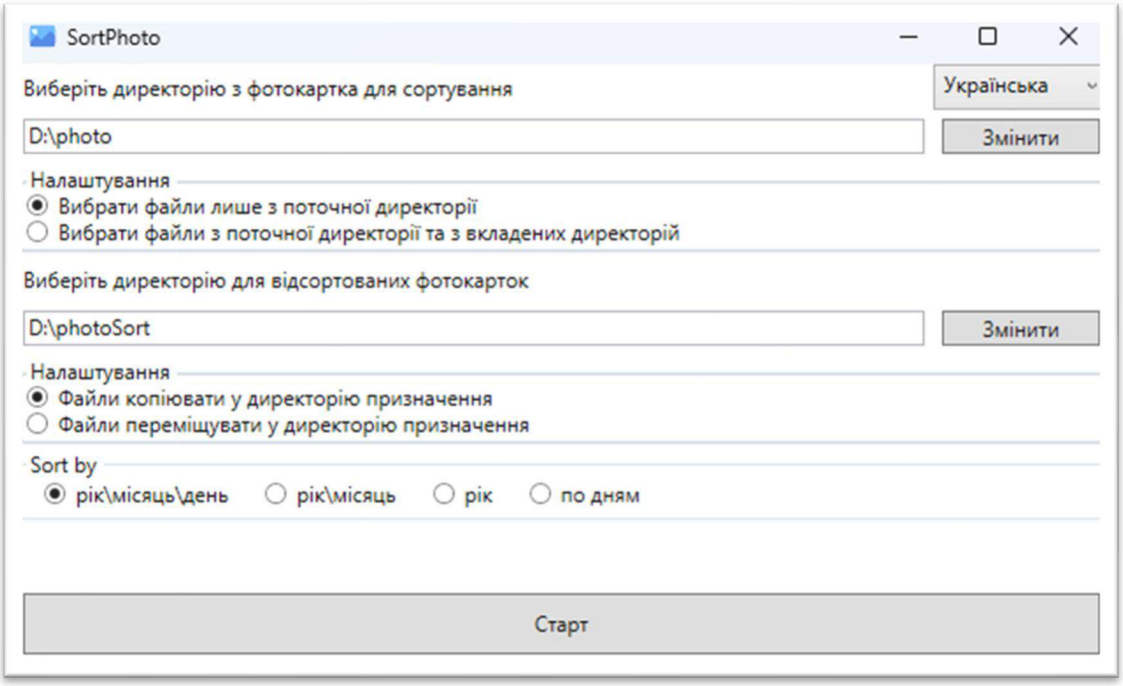


Рисунок 3.2.2 – Інтерфейс програми

Програма має можливість вибору мови інтерфейсу за допомогою списку що розкривається (Рисунок 3.2.2).

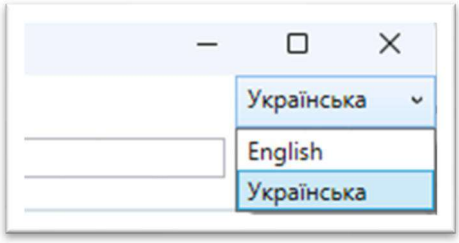


Рисунок 3.2.3 – Вибір мови інтерфейсу програми

При виборі англійської мови програма змінює весь інтерфейс на англійську мову (Рисунок 3.2.4).

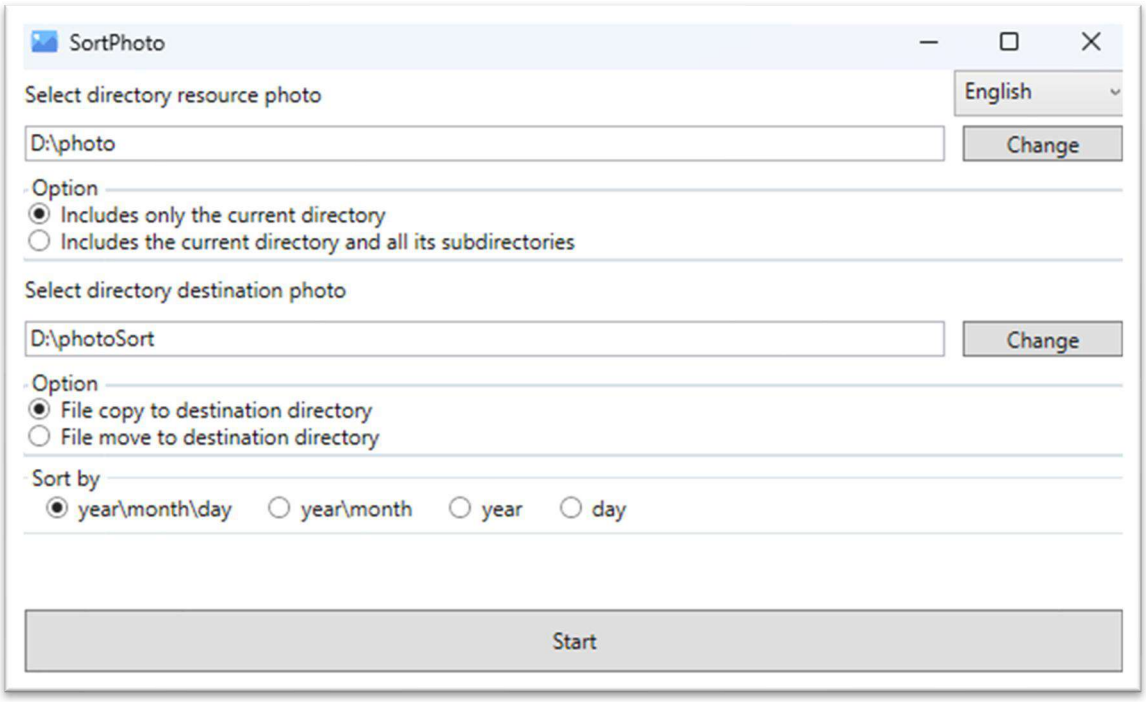


Рисунок 3.2.4 – Інтерфейс програми англійською мовою

Після успішного запуску програми потрібно вибрати теку з якої будуть оброблятись фотографії (Рисунок 3.2.5). Програма має можливість обробляти поточну теку або поточну з вкладеними теками. Для вибору скористайтесь перемикачем (Рисунок 3.2.6).

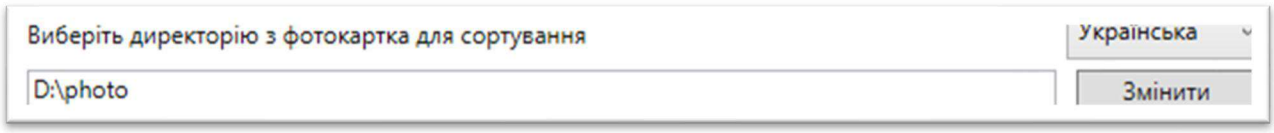


Рисунок 3.2.5 – Вибір директорії з фотографіями

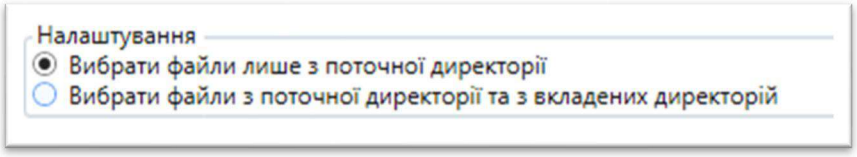


Рисунок 3.2.6 – Налаштування глибини перегляду теки

Важливо вибрати теку для збереження відсортованих фотографій (Рисунок 3.2.7).

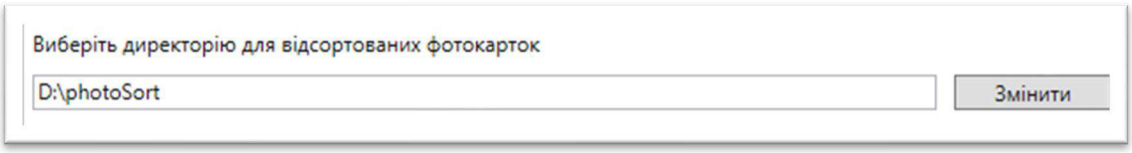


Рисунок 3.2.7 – Вибір теки в яку будемо сортування фото

Файли можуть копіюватися або переміщуватися під час сортування з вибраної теки. Це можна налаштувати за допомого перемикача (Рисунок 3.2.8).

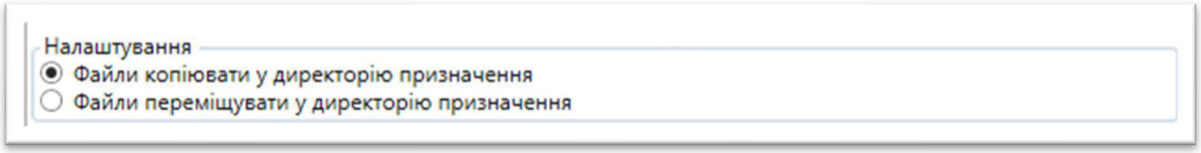


Рисунок 3.2.8 – Налаштування копіювання або переміщення

Сортувати фотографії можна за різним правилом. Правило дозволяє робити певну вкладеність тек. Правила перемикають за допомогою перемикача(Рисунок 3.2.9).

Правила:

- рік\місяць\день – рік, всередині року буде папка місяця, а у місяці тека з назвою дня місяця. Приклад організації тек можна переглянути на рисунку(Рисунок 3.2.12);
- рік\місяць – рік, всередині року буде папка місяця. Приклад організації тек можна переглянути на рисунку (Рисунок 3.2.13);
- рік– буде створено папка року фото і файл записуватиметься у неї. Приклад організації тек можна переглянути на рисунку (Рисунок 3.2.14);

- по дням – створюються папки на кожен день фотографії. Приклад організації тек можна переглянути на рисунку (Рисунок 3.2.15).

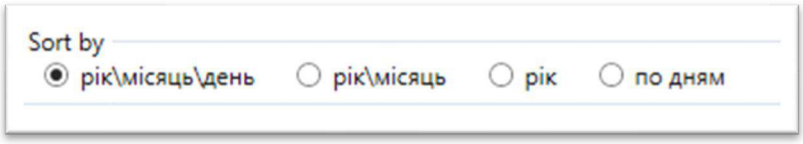


Рисунок 3.2.9 – Вибір вкладень тек для сортування

Після налаштування програми потрібно натиснути кнопку «Старт» (Рисунок 3.2.10).

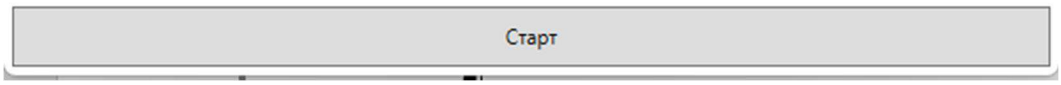


Рисунок 3.2.10 – Кнопка старту сортування

Програма може довго обробляти велику кількість файлів. Прогрес виконання можна спостерігати в інтерфейсі програми, де написано скільки оброблено і скільки всього файлів (Рисунок 3.2.11).

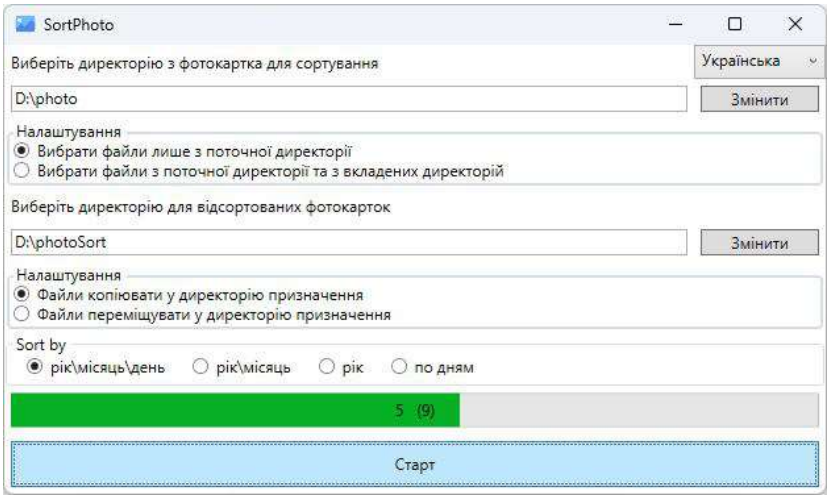


Рисунок 3.2.11 – Процес обробки зображень

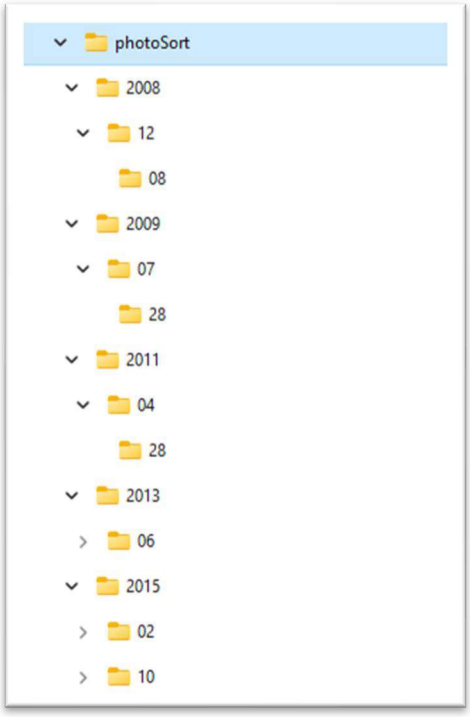


Рисунок 3.2.12 – Режим сортування рік\місяць\день

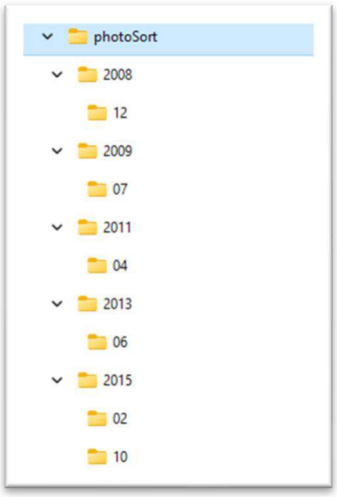


Рисунок 3.2.13 – Режим сортування рік/місяць

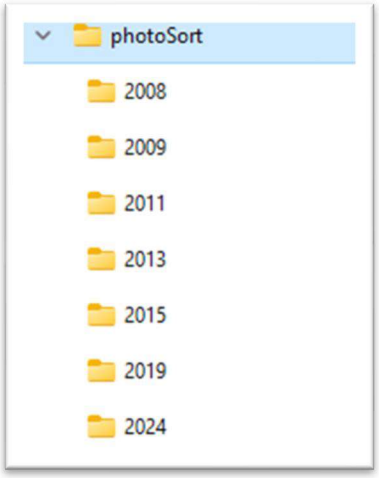


Рисунок 3.2.14 – Режим сортування за роком

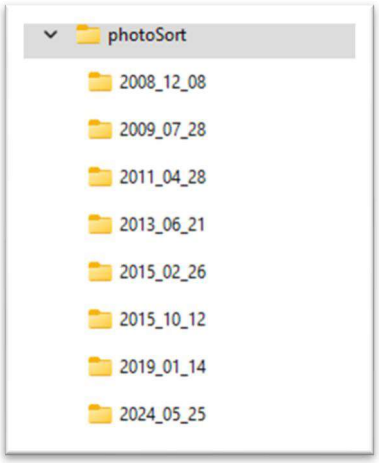


Рисунок 3.2.15 – Режим сортування за датою

Висновки до розділу 3

Інтерфейс програми для сортування файлів був розроблений з урахуванням найкращих практик юзабіліті, що забезпечує винятковий користувацький досвід. Програма відзначається інтуїтивно зрозумілим та приємним дизайном, який значно полегшує процес сортування та організації файлів.

Інтерфейс програми спроектовано так, щоб навіть користувачі без технічного досвіду могли швидко та легко зрозуміти його функціонал. Всі основні функції розташовані на видному місці, що мінімізує час на пошук необхідних опцій.

Програма надає можливість вибору та підтвердження дій через зрозумілі та зручні діалогові вікна. Користувачі можуть з легкістю обирати необхідні папки та файли, а також налаштовувати параметри сортування.

Чистий та сучасний дизайн інтерфейсу робить роботу з програмою приємною. Використання зрозумілих іконок та логічних кольорових схем сприяє зниженню візуального навантаження та підвищенню продуктивності користувачів.

Інтерфейс надає всі необхідні інструменти для ефективного сортування файлів, включаючи підтримку різних типів файлів, таких як зображення, відео та інші мультимедійні файли. Користувачі можуть налаштовувати сортування за різними критеріями, такими як дата створення, тип файлу та інші параметри.

Інтерфейс програми виключає можливість помилок користувача, таких як вибір неіснуючого шляху до теки, завдяки вбудованій перевірці введених даних. Це забезпечує високу надійність та зручність використання.

Програма добре працює на різних роздільних здатностях екрану та підтримує як світлу, так і темну теми, що дозволяє користувачам налаштовувати інтерфейс під свої потреби.

ВИСНОВКИ

Інтерфейс програми для сортування файлів демонструє високу якість та зручність використання. Його інтуїтивність, естетична привабливість, багатофункціональність та безпека роблять його ідеальним вибором для користувачів, які прагнуть ефективно організовувати свої файли. Завдяки ретельному підходу до проектування, програма забезпечує плавний та приємний користувацький досвід, задовольняючи потреби найвибагливіших користувачів.

У даній дипломній роботі була розглянута проблема сортування фотографій та відео за допомогою розробленої програми "Сортування фото та відео по теках". Під час проведення дослідження було виявлено, що існує значна потреба у зручному та ефективному інструменті для організації цифрових медіа-файлів, який допомагав би користувачам з легкістю управляти великими обсягами фотографій та відео.

В результаті аналізу були обрані технології розробки, такі як мова програмування C#, Windows Presentation Foundation (WPF), база даних SQLite, GitHub для керування версіями та Visual Studio для розробки. Ці технології були обрані на основі їхньої ефективності, зручності використання та можливості інтеграції.

В рамках дипломної роботи було розроблено програму "Сортування фото та відео по теках", яка дозволяє користувачам автоматично сортувати та організовувати свої медіа-колекції за різними критеріями, такими як дата, час, місцезнаходження та інші. Програма також має можливість редагування та анотування фотографій та відео, що робить її універсальним інструментом для організації медіа-колекцій.

У цій роботі було розглянуто і обґрунтовано кожен етап процесу розробки програмного забезпечення, від постановки задачі до вибору та застосування технологій розробки. Отримані результати свідчать про

успішність вибраних підходів та технологій у створенні ефективного та корисного інструменту для управління медіа-файлами.

Загалом, дана дипломна робота відображає важливість розробки програмного забезпечення для вирішення актуальних проблем цифрового суспільства та демонструє успішний приклад застосування сучасних технологій для реалізації практичних рішень.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Microsoft Developer Network (MSDN). URL: https://docs.microsoft.com/en-us/dotnet/ (дата звернення: 04.06.2024).
2. Windows Presentation Foundation (WPF) Documentation. Microsoft. URL: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/?view=netdesktop-6.0 (дата звернення: 04.06.2024).
3. SQLite Documentation. SQLite. URL: https://www.sqlite.org/docs.html (дата звернення: 04.06.2024).
4. GitHub Guides. GitHub. URL: https://guides.github.com/ (дата звернення: 04.06.2024).
5. Visual Studio Documentation. Microsoft. URL: https://docs.microsoft.com/en-us/visualstudio/ (дата звернення: 04.06.2024).
6. "Pro C# 9 with .NET 5" by Andrew Troelsen, Philip Japikse. Apress, 2020. ISBN: 978-1484268308.
7. "Professional WPF Programming: .NET Development with the Windows Presentation Foundation" by Chris Anderson, Chris Sells. Wrox, 2007. ISBN: 978-0470041806.
8. "Programming C# 9: Dive Deep into C# 9 and .NET 5" by Ian Griffiths. O'Reilly Media, 2021. ISBN: 978-1492082129.
9. "SQLite Database System: Design and Implementation" by Sibsankar Haldar. Springer, 2015. ISBN: 978-9812876635.
10. "GitHub For Dummies" by Sarah Guthals, Phil Haack, and others. Wiley, 2020. ISBN: 978-1119718199.
11. "Visual Studio 2019 In Depth" by Ovais Mehboob Ahmed Khan. Packt Publishing, 2020. ISBN: 978-1839213562.
12. "WPF 4.5 Unleashed" by Adam Nathan. Sams Publishing, 2013. ISBN: 978-0672336973.
13. "C# 9.0 in a Nutshell: The Definitive Reference" by Joseph Albahari, Ben Albahari. O'Reilly Media, 2021. ISBN: 978-1492051132.

14. "Learning SQLite for iOS" by Gene Da Rocha. Packt Publishing, 2016. ISBN: 978-1783555823.
15. "GitHub Essentials" by Achilleas Pipinellis. Packt Publishing, 2015. ISBN: 978-1783553713.
16. "Programming Windows Presentation Foundation" by Chris Sells, Ian Griffiths. O'Reilly Media, 2005. ISBN: 978-0596101138.
17. "C# 9 and .NET 5 - Modern Cross-Platform Development" by Mark J. Price. Packt Publishing, 2021. ISBN: 978-1800568106.
18. "C# 9.0 Pocket Reference: Instant Help for C# 9.0 Programmers" by Joseph Albahari, Ben Albahari. O'Reilly Media, 2021. ISBN: 978-1492082785.
19. "GitHub For Teens" by Christopher Tozier. For Dummies, 2022. ISBN: 978-1119954031.
20. "WPF Recipes in C# 2008: A Problem-Solution Approach" by Sam Bourton, Allen O'Neill. Apress, 2008. ISBN: 978-1430210845.

ДОДАТКИ

ДОДАТОК А.

Програмний код модулів

```
namespace SortPhoto
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        LocaleUI locale;
       // private string key;

        //folderresource
        public string FolderResource
        {
            get { return (string)GetValue(folderResourceProperty); }
            set { SetValue(folderResourceProperty, value); }
        }
        public static readonly DependencyProperty folderResourceProperty =
                DependencyProperty.Register("FolderResource", typeof(string),
typeof(MainWindow), new PropertyMetadata(@"e:\ALL"));

        //folderDestination
        public string FolderDestination
        {
            get { return (string)GetValue(folderDestinationProperty); }
            set { SetValue(folderDestinationProperty, value); }
        }
        public static readonly DependencyProperty folderDestinationProperty =
```

```
            DependencyProperty.Register("FolderDestination",       typeof(string),
typeof(MainWindow), new PropertyMetadata(@"e:\res"));

        //folderDestination
        public bool OptionSubdirectories
        {
           get { return (bool)GetValue(OptionSubdirectoriesProperty); }
           set { SetValue(OptionSubdirectoriesProperty, value); }
        }
        public static readonly DependencyProperty OptionSubdirectoriesProperty =
            DependencyProperty.Register("OptionSubdirectories",    typeof(bool),
typeof(MainWindow), new PropertyMetadata(false));

        // OptionCopyOrMove
        public bool OptionCopyOrMove
        {
           get { return (bool)GetValue(OptionCopyOrMoveProperty); }
           set { SetValue(OptionCopyOrMoveProperty, value); }
        }
        public static readonly DependencyProperty OptionCopyOrMoveProperty =
            DependencyProperty.Register("OptionCopyOrMove",     typeof(bool),
typeof(MainWindow), new PropertyMetadata(false));

        // CountFile
        public Double CountFile
        {
           get { return (Double)GetValue(CountFileProperty); }
           set { SetValue(CountFileProperty, value); }
        }
        public static readonly DependencyProperty CountFileProperty =
```

```
            DependencyProperty.Register("CountFile", typeof(double),
typeof(MainWindow), new PropertyMetadata(0.0));

        // SortInProgress
        public bool SortInProgress
        {
            get { return (bool)GetValue(SortInProgressProperty); }
            set { SetValue(SortInProgressProperty, value); }
        }
        public static readonly DependencyProperty SortInProgressProperty =
            DependencyProperty.Register("SortInProgress", typeof(bool),
typeof(MainWindow), new PropertyMetadata(false));

        public MainWindow()
        {
            locale = new LocaleUI();
            //     BoolToContentStartConverter.Main = this;
            //   BoolToContentStartConverter.lang = locale.Currentlanguage;
            InitializeComponent();
            DataContext = this;

        }
        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            SetDefaultDrive();
            RefreshLanguge();
            Lang.ItemsSource = locale?.Languages?.Select(d => d.Name);
            TestTime.TestNow(1725207616);
            var result = ConfigurationManager.AppSettings["IndexLanguage"];
            Lang.SelectedIndex = int.Parse(result);
```

```
            //var g=ConfigurationManager.AppSettings
            //    .AllKeys.Select(x => $"{x} {appSettings[x]}").ToList();
            //  string result = appSettings["IndexLanguage"] ?? "5";
            //  Lang.SelectedIndex = int.Parse(result);

        }

        private void SetDefaultDrive()
        {
            DriveInfo[] allDrives = DriveInfo.GetDrives();
            //test
            //FolderDestination = "d:\\tttt\\";
            //FolderResource = "d:\\testPhoto\\захист курсових\\";
            //ver 2
            FolderResource = allDrives.FirstOrDefault()?.Name;
            FolderDestination = allDrives.LastOrDefault()?.Name;
            //ver 1
            // FolderResource = System.IO.Directory.GetCurrentDirectory();
            // FolderDestination = System.IO.Directory.GetCurrentDirectory();
        }

        private void ChangeFolder_Click(object sender, RoutedEventArgs e)
        {
            if (sender is Button button)
            {
                VistaFolderBrowserDialog browserDialog = new
VistaFolderBrowserDialog();
                if (browserDialog.ShowDialog() ?? false)
                {
                    if (button.Tag.ToString() == "FolderResourse")
```

```
                FolderResource = browserDialog.SelectedPath;
            else
                FolderDestination = browserDialog.SelectedPath;


        }
    }
}
private void RefreshLanguge()
{
    //  BoolToContentStartConverter.lang = locale.Currentlanguage;
    foreach (var dic in locale.Defaullanguage.Values)
    {
        var ui = FindlElement.FindUid(this, dic.Key);
        if (!locale.Currentlanguage.Values.TryGetValue(dic.Key, out string newVallocal))
        {
            newVallocal = dic.Value;
        }
        switch (ui)
        {
            case GroupBox gb:
                gb.Header = newVallocal; break;
            case Button bt:
                {
                    string text = newVallocal;
                    if (!string.IsNullOrEmpty(text) && text.Contains("|", StringComparison.CurrentCulture))
                    {
                        text = text.Substring(0, text.IndexOf("|", StringComparison.CurrentCulture));
```

```
                }

                bt.Content = text;
            }
            break;
        case RadioButton rb:
            rb.Content = newVallocal; break;
        case TextBlock tb:
            tb.Text = newVallocal; break;
    }
  }

}

private async void Start_Click(object sender, RoutedEventArgs e)
{
  //  Button bt = sender as Button;
    if (!SortInProgress)
    {
        if (!Directory.Exists(FolderResource))
        {
            MessageBox.Show("Error on path to folder resource: " +
FolderResource);
            return;
        }
        if (!Directory.Exists(FolderDestination))
        {
            MessageBox.Show("Error path folder destination: " +
FolderDestination);
            return;
```

```
            }
          //  bt.IsEnabled = false;
            List<string>  listFile  =  CreateListFile(OptionSubdirectories  ?
SearchOption.AllDirectories : SearchOption.TopDirectoryOnly);
            //  bt.IsEnabled = true;
          //   listFile = listFile.Skip(2829).ToList();
            //    .2830
            if (listFile != null)
            {
               SortInProgress = true;
               ProgrSort.Maximum = listFile.Count;
               await SortFileAsync(listFile).ConfigureAwait(true);
               if (OptionCopyOrMove)
                  DeleteEmptySubdirectories(FolderResource);
               SortInProgress = false;
            }
         }
         else
         {
            SortInProgress = false;
         }

      }
      private List<string> CreateListFile(SearchOption option)
      {
         return Directory.GetFiles(FolderResource, "*.*", option).ToList<string>();
      }
      private string GetTypeSort()
      {
```

```
        return SortingSelection.Children.OfType<RadioButton>().Where(rb => rb.IsChecked == true).First().Tag.ToString();


    }
    private async Task SortFileAsync(List<string> listFile)
    {
        string TypeSort = GetTypeSort();
        foreach (var fileName in listFile)
        {
            await CopyFile(fileName, TypeSort).ConfigureAwait(true);
            CountFile++;
            if (!SortInProgress)
                break;
        }
        CountFile = 0;
    }
    public async Task CopyFile(string sourceFile, string TypeSort = "yyyy_MM_dd")
    {
        string ext = System.IO.Path.GetExtension(sourceFile);
        string destinationFile = String.Empty;
        string nameFolder = FolderDestination + "\\";

        ExifProperty exifProperty = MetadataImage.GetDateTimeOriginal(ExifTag.DateTimeOriginal, sourceFile);
        DateTime dateFile;
        if (exifProperty != null && exifProperty.Value is DateTime dateTimeOriginal)
            dateFile = dateTimeOriginal;
        else
```

```
            dateFile = File.GetLastWriteTime(sourceFile);

            destinationFile = $"{dateFile:yyyy_MM_dd_HH_mm_ss}{ext}";
            nameFolder += dateFile.ToString(TypeSort);
            if (!Directory.Exists(nameFolder))
            destinationFile = duplicateDestinationFile;
            await CopyFileAsync(sourceFile, destinationFile,
OptionCopyOrMove).ConfigureAwait(true);
        }
        public static async Task CopyFileAsync(string sourceFile, string
destinationFile, bool move = false)
        {
            try
            {
                if (move)
                {
                    await Task.Run(
                        () => File.Move(sourceFile, destinationFile)
                        ).ConfigureAwait(true);
                }
                else
                {
                    await Task.Run(
                         () => File.Copy(sourceFile, destinationFile)
                         ).ConfigureAwait(true);
                }
            }
            catch (Exception ex)
            {
                 DeleteEmptySubdirectories(directory);
```

```
                if
(!System.IO.Directory.EnumerateFileSystemEntries(directory).Any())
                    System.IO.Directory.Delete(directory, false);
        private void SaveSetting(int value)
        {
            try
            {
                string key = "IndexLanguage";
                var                                    configFile                                    =
ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
                var settings = configFile.AppSettings.Settings;
                if (settings[key] == null)
                {
                    settings.Add(key, "0");
                }
                else
                {
                    settings[key].Value = "" + value;
                }
                configFile.Save(ConfigurationSaveMode.Modified);

ConfigurationManager.RefreshSection(configFile.AppSettings.SectionInformation
.Name);
            }
            catch (ConfigurationErrorsException)
            {
              Title="Error writing app settings";
    }
}
```