МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ХОЛОД Юрій Васильович

Програмний модуль для перегляду зображень / A software module for viewing images

Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КНз-41
Ю.В. Холод

Науковий керівник:
к.т.н. І.М. Майків

Кваліфікаційну роботу допущено до захисту
«___»_____ 2025 р.

В.о. завідувача кафедри
_____ Н.М. Васильків

Тернопіль – 2025

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.М. Васильків
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ХОЛОДУ Юрію Васильовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Програмний модуль для перегляду зображень / Software module for images viewing
керівник роботи к.т.н. І.М. Майків
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 20 грудня 2024 р. № 938.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:

– провести аналіз програмних продуктів призначених для перегляду зображень, та сформулювати набір функціональних вимог до программ даного класу;

– розробити архітектуру програмного модуля, зокрема алгоритмічне та інформаційне забезпечення, модель даних;

– реалізувати програмний модуль, втому числі користувацький інтерфейс із використанням Qt бібліотеки;

– провести тестування розробленого модуля, зокрема корректного завантаження та відображення зображень, а також всіх функціональних мижливостей.

5. Перелік графічного матеріалу в роботі:

– діаграма потоків даних;
– діаграма декомпозиції IDEF0;
– EDR-діанрама;
– діаграма класів;
– блок-схема алгоритму роботи модуля;
– результати експериментальних досліджень.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2025 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2025 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2025 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2025 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2025 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2025 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2025 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2025 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2025 р.	

Студент _____ Ю.В. Холод
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи _____ І.М. Майків
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмний модуль для перегляду зображень» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом у 71 сторінки і містить 19 ілюстрацій, 1 таблиця, 2 додатки та 28 використаних джерела.

Кваліфікаційну роботу присвячено розробці програми для зручного перегляду зображень на персональному компютері користувача під ОС Windows.

У роботі розглянуті існуючі інструменти для перегляду зображень, проведено їх порівняння, зокрема аналіз функціональності та можливостей програмного забезпечення.

Виконано планування розробки програмного модуля, яке включало створення моделі DFD та її декомпозицію, а також розробку ERD діаграм.

Результатом проведеної роботи є створений додаток для перегляду зображень, що забезпечує інтерфейс для перегляду, масштабування зображень та перемикання між зображеннями.

Практичне значення роботи полягає в тому, що розроблений додаток може бути використаний для роботи з графічними файлами, надаючи користувачам інструмент для перегляду зображень на персональному компютері під ОС Windows.

Ключові слова: ПРОГРАМНИЙ МОДУЛЬ, ГРАФІЧНИЙ ФАЙЛ, ОБРОБКА ЗОБРАЖЕННЯ, ПЕРЕГЛЯД, ФОТОГРАФІЯ, ТЕСТУВАННЯ.

ANNOTATION

The bachelor’s qualification thesis titled “Software module for images viewing”, completed to obtain the bachelor’s degree in Computer Science (specialty 122 “Computer Science”, educational program “Computer Science”), consists of 71 pages and includes 19 figures, 1 table, 2 appendices, and 28 references.

The qualification work is dedicated to the development of a program for convenient image viewing on a user's personal computer running the Windows operating system.

The work examines existing tools for image viewing, compares them, and analyzes their functionality and capabilities.

The planning of the software module development has been carried out, which included the creation of a DFD model and its decomposition, as well as the development of ERD diagrams.

The result of the work is the creation of an image viewer application that provides an interface for viewing, scaling images, and switching between images.

The practical significance of the work lies in the fact that the developed application can be used for working with graphic files, providing users with a tool for viewing images on a personal computer under the Windows operating system.

Keywords: SOFTWARE MODULE, GRAPHIC FILE, IMAGE PROCESSING, VIEWING, PHOTOGRAPHY, TESTING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

JPEG - Joint Photographic Experts Group

PNG - Portable Network Graphics

BMP - Bitmap

TIF - Tagged Image File Format

GIF - Graphics Interchange Format

RAW - Raw Image Format

PDF - Portable Document Format

GFL - General Flow Language

WPF - Windows Presentation Foundation

DFD - Data Flow Diagram

ERD - Entity Relationship Diagram

IDE - Integrated Development Environment

ВСТУП

Обробка зображень є невід'ємною складовою багатьох галузей, таких як фотографія, дизайн, маркетинг, наукові дослідження, а також повсякденне використання цифрових пристроїв. Для роботи з графічними файлами користувачам необхідні програми, що дозволяють переглядати зображення різних форматів, здійснювати базові операції над ними, такі як масштабування, навігація між зображеннями.

Попри значну кількість доступних програм для перегляду зображень, не всі вони відповідають запитам користувачів, які прагнуть отримати поєднання простоти інтерфейсу, гнучких функцій управління зображеннями та підтримки різноманітних форматів. Багато з існуючих програм мають обмеження, що стосуються продуктивності, зручності використання або підтримки специфічних функцій, таких як інтерактивне масштабування, навігація по галереях із великою кількістю файлів. Це створює передумови для розроблення нового програмного забезпечення, яке забезпечить зручний і функціональний перегляд графічних даних.

У межах цієї кваліфікаційної роботи розглядається процес створення програмного модуля для перегляду зображень, який дозволяє реалізувати функції завантаження окремих графічних файлів або цілих директорій, навігації між зображеннями, масштабування, а також підтримку різних графічних форматів. Проєктування програми відбувається на основі бібліотеки Qt, що забезпечує розширені можливості створення графічного інтерфейсу користувача, а також кросплатформенність.

Актуальність цієї роботи зумовлена необхідністю створення програмного забезпечення, яке поєднує підтримку різноманітних графічних форматів із базовими функціями роботи з зображеннями, такими як завантаження, навігація та масштабування, а також забезпечує простий і зрозумілий інтерфейс для користувачів.

Метою роботи є розробка програмного модуля для перегляду зображень із функціоналом, який забезпечує базові операції із графічними файлами в середовищі персональних комп’ютерів.

Завдання роботи включають:

1. Проєктування структури та функціоналу програмного модуля.
2. Розроблення програмного коду на основі бібліотеки Qt.
3. Тестування програмного модуля.

Об’єктом проектування є процес розроблення програмного забезпечення для перегляду зображень.

Предметом дослідження є програмний модуль для перегляду графічних файлів, включно з функціональними можливостями завантаження, масштабування, навігації та інтерактивної обробки зображень.

Методи дослідження включають системний аналіз існуючих програмних рішень для роботи з графічними файлами, розробку алгоритмічного та інформаційного забезпечення модуля для перегляду зображень, методи об'єктно-орієнтованого програмування для реалізації функціоналу модуля, а також тестування для перевірки його працездатності.

Результати роботи можуть бути застосовані в різних сферах, таких як побутове використання, навчання або у професійних галузях, де необхідно працювати з графічними файлами

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОГО МОДУЛЯ ПЕРЕГЛЯДУ ЗОБРАЖЕНЬ

1.1 Функціональні вимоги до програм призначених для перегляду зображень

Програмне забезпечення для перегляду зображень використовується для роботи з графічними файлами, забезпечуючи користувачам можливість виконувати базові операції з їх перегляду, управління та редагування. Такі програми можуть виконувати широкий спектр завдань, включаючи завантаження та відображення зображень, зміну їх масштабу, орієнтації та навігацію між файлами.

Основна функція програм для перегляду зображень – забезпечення користувачам доступу до графічної інформації у зручному форматі. Завантаження зображень у таких програмах зазвичай реалізується через вбудовані інтерфейси для вибору файлів із локального пристрою. Відображення зображень відбувається з автоматичним масштабуванням під розміри вікна програми, що дозволяє оптимізувати простір екрана [1].

Зміна масштабу дає можливість користувачам переглядати фрагменти зображень або зменшувати їх для перегляду у повному розмірі. Функція обертання дозволяє змінювати орієнтацію зображень на фіксовані кути, наприклад, на 90 градусів, що є корисним для виправлення помилок у розташуванні графічних файлів.

Для роботи з великою кількістю зображень реалізується навігація між файлами. Програми можуть автоматично визначати список графічних файлів у вибраній директорії, дозволяючи користувачам переглядати їх у послідовному порядку за допомогою кнопок або гарячих клавіш.

Для визначення повного набору функцій програми створено діаграму дерева функцій (рисунок 1.1). Ця діаграма дозволяє структурно подати всі основні процеси, що реалізуються програмою [2]. Верхній рівень дерева представляє основну функцію програми – забезпечення перегляду зображень. На наступних рівнях деталізуються підфункції, зокрема:

– завантаження графічних файлів;

– відображення зображень;
– масштабування;
– адаптація під розмір вікна;
– додавання прокруток для великих зображень;
– обертання;
– навігація між файлами;
– сортування файлів;
– підтримка різних форматів графічних файлів.

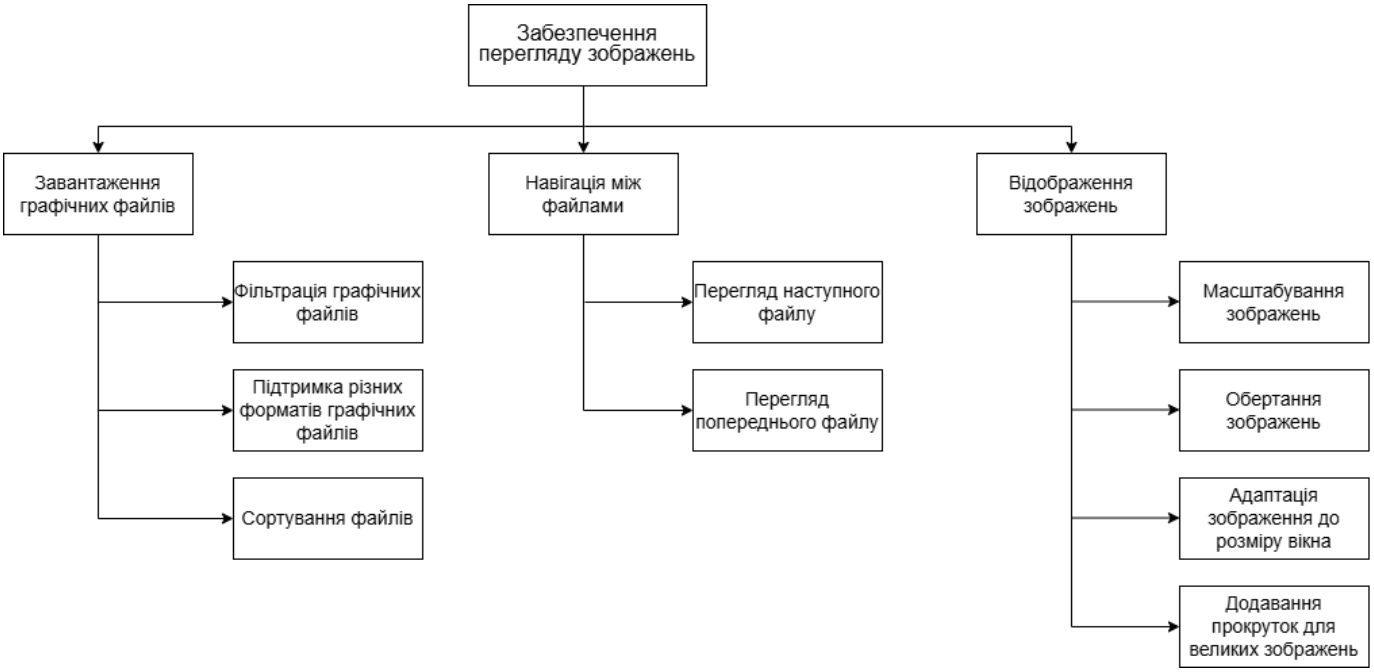

Рисунок 1.1 – Діаграма дерева функцій

Ця діаграма дерева функцій слугує основою для аналізу програмного забезпечення, дозволяючи визначити, які складові функціоналу є важливими для користувачів у процесі роботи із зображеннями. Транзакційна складова в програмі для перегляду зображень включає збір, обробку та збереження кількісних даних, що відображають взаємодію користувача з програмою. Це стосується фіксації всіх основних операцій, виконуваних під час роботи з графічними файлами. Зокрема, до

транзакційної складової відносяться завантаження фотографій, масштабування зображень, обертання та навігація між зображеннями.

Програма фіксує інформацію про завантажені графічні файли, включаючи їх кількість, формати та час завантаження. Це дозволяє визначити, скільки файлів було завантажено протягом певного періоду, а також які формати використовуються найчастіше.

При масштабуванні зображень кожне змінення масштабу реєструється системою. Це включає в себе фіксацію рівня масштабування, коли користувач змінює розмір зображення, як за допомогою кнопок, так і за допомогою прокрутки миші або жестів. Така інформація дозволяє зрозуміти, які значення масштабу найбільш часто використовуються.

Також програма фіксує кожне натискання умовних кнопок "Вперед" та "Назад" для перемикання між зображеннями в каталозі. Ці дані дають змогу відстежити, скільки разів користувач перемикався між файлами, а також яку кількість зображень він переглянув за один сеанс роботи з програмою.

Аналітична складова в програмі для перегляду зображень базується на використанні зібраних транзакційних даних для поглибленого аналізу роботи програми. Ці дані дозволяють отримати кількісні показники, необхідні для подальшого вивчення використання програми в різних аспектах.

Аналіз найбільш популярних форматів зображень дозволяє з'ясувати, які формати зображень використовуються найчастіше. Це дає можливість оцінити популярність таких форматів, як JPEG, PNG, BMP та інших, і визначити, які з них потребують додаткової підтримки або оптимізації в програмі.

Дані про час використання програми дозволяють оцінити, скільки часу користувачі витрачають на перегляд зображень та які функції програми вони використовують найчастіше [6]. Це включає дослідження періодів активності користувачів, кількості сеансів, а також середнього часу перегляду зображень.

1.2 Огляд програм призначених для перегляду зображень

IrfanView - це безкоштовний графічний переглядач, розроблений для операційних систем Windows боснійським програмістом Ірфаном Шкільяном. Програма надає користувачам можливість переглядати, редагувати, конвертувати та обробляти зображення в різних форматах. Першочергово створена для роботи з графічними файлами, вона підтримує й інші спеціалізовані формати, зокрема GIF, TIF, ICO, JPEG, PNG і багато інших. Однією з перших її особливостей була підтримка мультисторінкових TIF файлів та анімованих GIF зображень [3].

Програма забезпечує наступні основні функції:

- перегляд зображень;
- масштабування;
- обертання;
- зміну формату;
- створення слайдшоу;
- обробку пакетних файлів та їх оптимізацію.

Окрім базових функцій перегляду та редагування, програма підтримує роботу з мультимедійними файлами та дозволяє здійснювати сканування і друк зображень.

Інтерфейс програми побудований таким чином, щоб користувач міг швидко отримати доступ до основних функцій, таких як перегляд, конвертація та редагування зображень. Вікно програми (рисунок 1.2) містить елементи для запуску цих операцій, що дозволяє спрощено здійснювати роботу з графічними файлами. Для пакетної обробки зображень застосовуються відповідні функції, що дозволяють обробляти кілька файлів одночасно.

Програма написана мовою програмування C++ [4], що дозволяє забезпечити високу швидкість роботи. Для розробки інтерфейсу та підтримки мультимедійних форматів також використовуються різноманітні бібліотеки та інструменти, такі як DirectX для роботи з відео та мультимедіа. Це дозволяє IrfanView підтримувати широкий спектр форматів зображень і відео файлів.

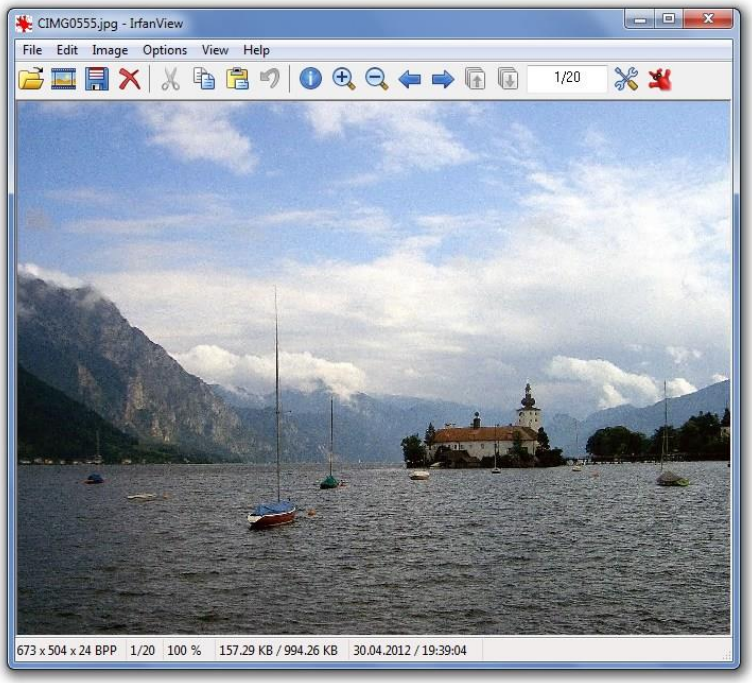

Рисунок 1.2 – Вигляд головного вікна програми IrfanView

XnView - це безкоштовний переглядач та редактор зображень, розроблений компанією XnSoft і вперше випущений в 1998 році. Програма підтримує широкий спектр графічних форматів, включаючи JPEG, TIFF, PNG, GIF, WEBP, JPEG-XL, HEIC, PSD, JPEG2000, OpenEXR, а також формати камери RAW, DNG, CR2 та PDF [5]. Завдяки цьому XnView забезпечує можливість роботи з різними типами зображень і файлів, включаючи професійні формати, що широко використовуються в фотографії та графічному дизайні.

Програма дозволяє користувачам переглядати, редагувати і управляти графічними файлами за допомогою інтуїтивно зрозумілого інтерфейсу, який нагадує файловий менеджер. Головне вікно програми (рисунок 1.3) надає доступ до основних функцій, таких як перегляд зображень, редагування метаданих (IPTC, XMP), зміна розміру зображень, обрізка, коригування кольору, а також можливість здійснювати захоплення екрану. Це дозволяє користувачам виконувати базові операції редагування зображень прямо в програмі без потреби в додаткових інструментах.

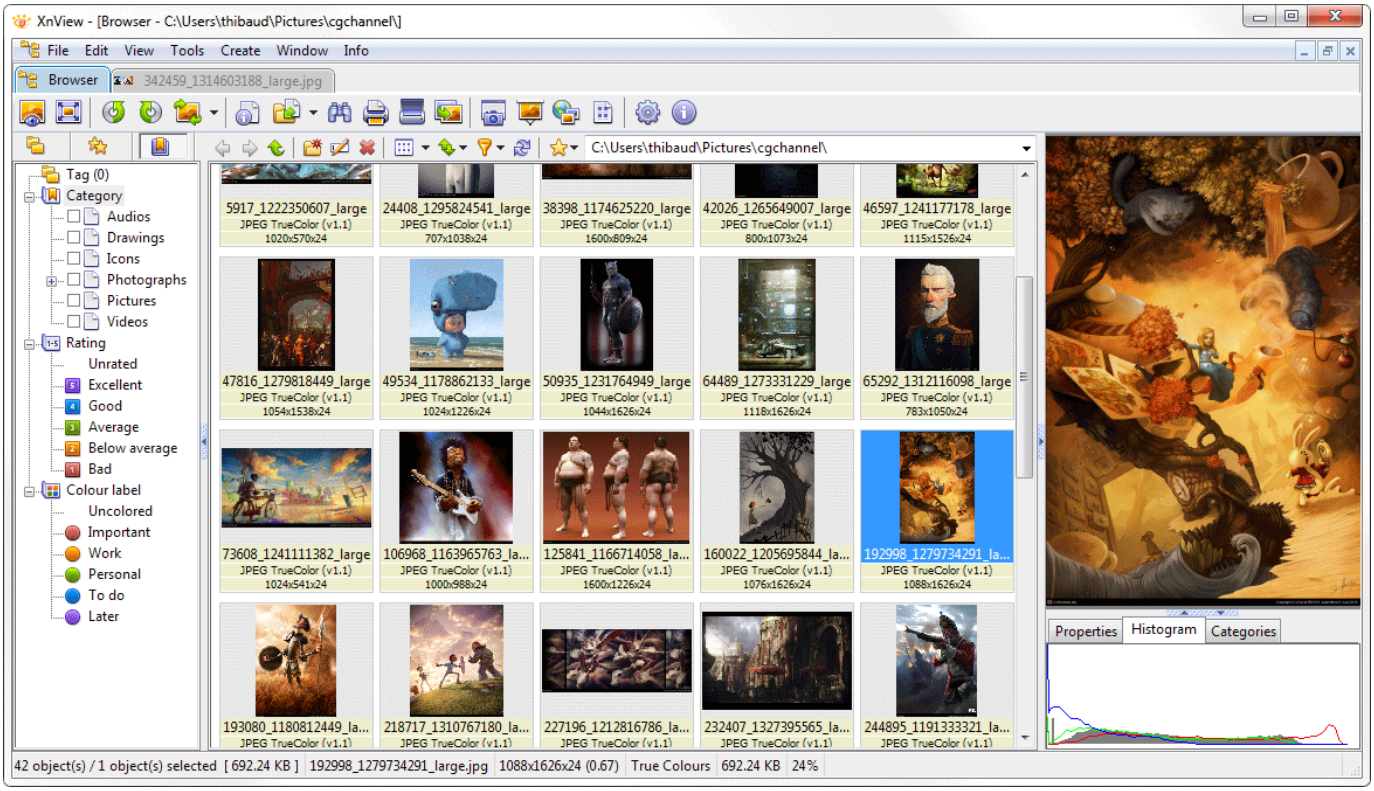

Рисунок 1.3 – Вигляд головного вікна програми XnView

XnView також підтримує функції пакетної обробки файлів, що дозволяє виконувати операції, такі як перейменування, конвертація, зміна розміру, додавання водяних знаків та тексту, а також фільтрація зображень. Для пакетної обробки використовується окремий додаток XnConvert, який дозволяє здійснювати швидку обробку великої кількості файлів одночасно. Для зміни розміру зображень доступна програма XnResize, яка дає змогу змінювати розмір і конвертувати зображення в пакетному режимі.

Для організації роботи з великою кількістю зображень XnView пропонує інструменти для порівняння зображень, пошуку дублікати, а також створення контактних листів і слайдшоу. Такі можливості роблять XnView зручним інструментом не тільки для перегляду, але й для організації зображень.

Програма написана на мовах програмування C і C++ [4], що дозволяє забезпечити сумісностність з різними операційними системами, включаючи Windows, Linux та macOS. Для роботи з графічними файлами XnView використовує

бібліотеку GFL (Graphic File Library) [6], що забезпечує підтримку великої кількості графічних форматів і обробку зображень.

Програма «Фотографії» - вбудований інструмент операційних систем Windows, починаючи з Windows 8. Вона призначена для перегляду, редагування та організації зображень. У програмі реалізовані функції перегляду зображень у таких форматах, як JPEG, PNG, BMP, GIF, TIFF, HEIC та інші [7]. Після відкриття файлу, користувач може виконувати базові операції редагування, зокрема зміну масштабу, обертання, обрізку та коригування параметрів кольору, яскравості, контрасту та насиченості зображень.Інтерфейс програми (рисунок 1.4) спроектовано для забезпечення простоти використання. Зображення можуть бути переглянуті як у повноекранному режимі, так і в вікні програми. Також доступна функція перегляду слайдшоу та переходу між зображеннями в папці або галереї. У програмі реалізовано механізм пошуку зображень, що дозволяє знаходити файли за різними критеріями, такими як тип файлу, дата зйомки чи теги.

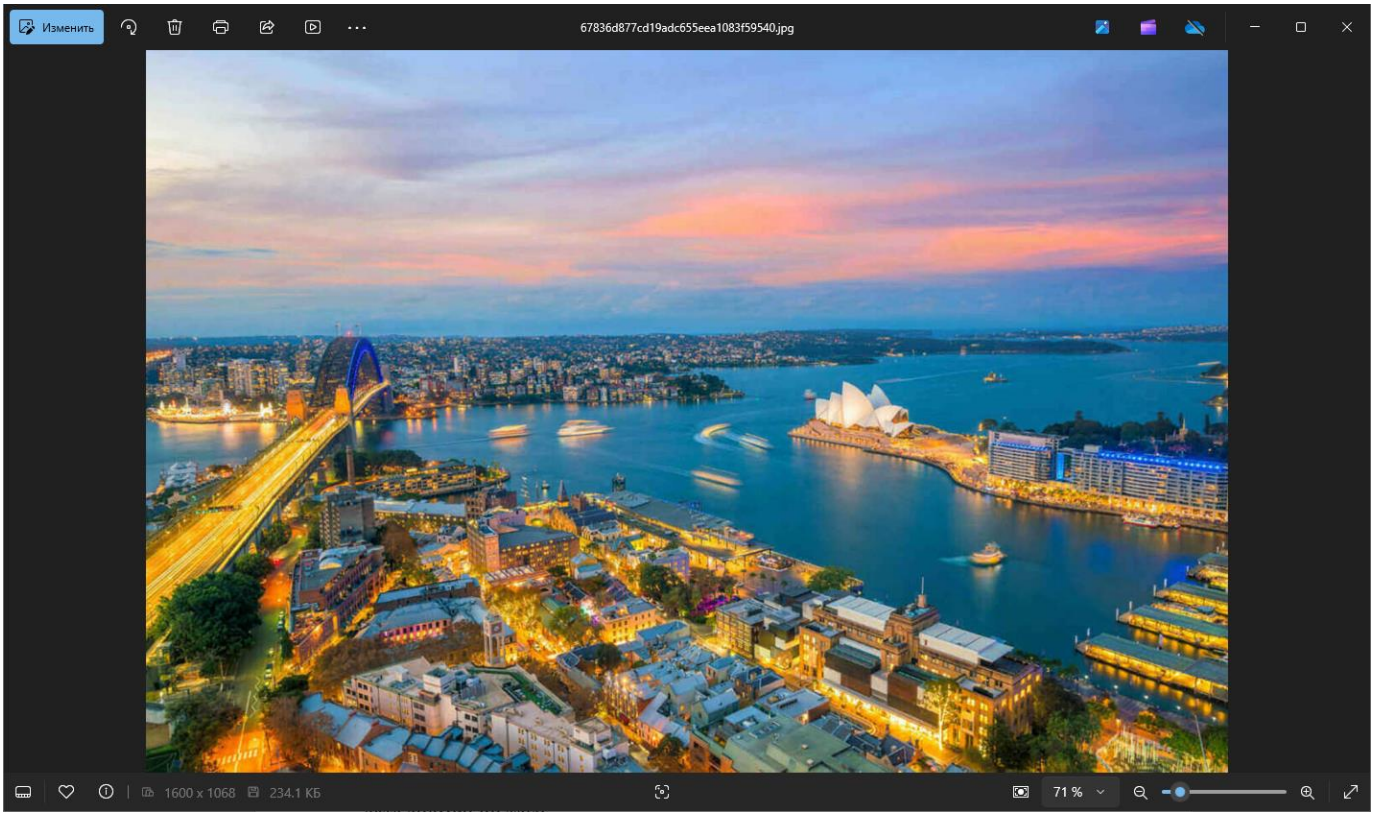

Рисунок 1.4 – Вигляд головного вікна програми «Фотографії»

Програма має інтерфейс для редагування метаданих зображень, зокрема таких, як дата зйомки, геолокація, інформація про камеру та теги. Крім того, доступна синхронізація з OneDrive, що дозволяє зберігати й переглядати зображення на різних пристроях через хмарний сервіс.

Технічно програма «Фотографії» є частиною операційної системи Windows і написана з використанням мов програмування C++ та C#. Для побудови графічного інтерфейсу користувача використано Windows Presentation Foundation (WPF). Взаємодія з іншими компонентами операційної системи здійснюється через WinRT API, що забезпечує інтеграцію з іншими додатками та службами Windows. Оскільки програма є частиною операційної системи, вона безпосередньо взаємодіє з іншими системними компонентами, що дозволяє підтримувати високу швидкість і стабільність при роботі з графічними файлами [8].

Програма підтримує новітні формати зображень, такі як HEIC, що є необхідною функцією для роботи з файлами, створеними на пристроях Apple.

Результати проведеного аналізу різних програмних продуктів представлені в таблиці 1.1 , де проведено порівняння їх основних функціональних можливостей.

З таблиці можна зробити висновок, що кожна програма має свої особливості і обмеження. IrfanView є обмеженим у підтримці мультимедійних форматів і редагуванні метаданих, але забезпечує основні функції для перегляду та редагування зображень. XnView пропонує більш широку підтримку мультимедійних форматів і метаданих, а також підтримку пакетної обробки, що робить його зручним інструментом для обробки великої кількості файлів. Програма «Фотографії» інтегрована з операційною системою Windows, забезпечує базові функції редагування і перегляду зображень, а також підтримує синхронізацію з хмарними сервісами через OneDrive. Вона є найбільш обмеженою у порівнянні з іншими двома програмами, оскільки не підтримує мультимедійні формати і не має таких функцій, як пакетна обробка чи редагування метаданих.

Таблиця 1.1 – Порівняння основних функцій оглянутих програм

№	Критерій	IrfanView	XnView	Програма «Фотографії»
1	Операційні системи	Windows	Windows, Linux, macOS	Windows 8 і новіші
2	Ціна	Безкоштовно	Безкоштовно	Входить в вартість ОС Windows
3	Перегляд зображень	Так	Так	Так
4	Перегляд групи зображень	Так (слайдшоу)	Так	Так (слайдшоу, галерея)
5	Редагування зображень	Масштабування, обертання, обрізка, корекція кольору	Масштабування, обертання, обрізка, корекція кольору, редагування метаданих	Масштабування, обертання, обрізка, корекція кольору
6	Підтримка GIF	Так	Так	Так
7	Підтримка мультимедійних форматів	Обмежено (відео підтримуються частково)	Широка підтримка мультимедіа (відео, анімація)	Немає підтримки мультимедіа

1.3 Формулювання вимог та постановка задачі

Метою роботи є розробка програмного модуля для перегляду зображень з можливістю здійснення базових операцій, таких як завантаження, перегляд, масштабування та перемикання між зображеннями. Програма повинна бути реалізована на мові програмування C++ з використанням бібліотеки Qt для побудови графічного роботи. В процесі виконання роботи потрібно:

1. Проєктування структури та функціоналу програмного модуля:

– загальна структура програмного модуля;

– алгоритмічне забезпечення модуля;

– інформаційне забезпечення модуля.

2. Розроблення програмного коду на основі бібліотеки Qt:

– реалізація методів вибору та завантаження зображень з файлової системи через стандартний діалог.

– реалізація методів відображення зображень у вікні програми з можливістю прокручування при перевищенні розміру зображення.

– реалізація методів які забезпечують перемикання між зображеннями в папці за допомогою кнопок "Вперед" і "Назад".

– реалізація можливості масштабування зображень за допомогою кнопок або прокрутки миші;

– проєктування інтерфейсу для взаємодії користувача з програмою

3. Тестування програмного модуля:

– перевірка коректності завантаження та відображення зображень.

– перевірка роботи функцій перемикання між зображеннями.

– перевірка роботи функцій масштабування та повороту зображень.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ ДЛЯ ПЕРЕГЛЯДУ ЗОБРАЖЕНЬ

2.1 Загальна структура програмного модуля

Для реалізації програмного модуля для перегляду зображень було обрано підхід, що передбачає використання декількох етапів аналізу та моделювання системи. Першим кроком було визначення загальної структури системи, що включає в себе ідентифікацію основних компонентів та їхніх взаємозв'язків. Для цього було застосовано діаграму потоків даних (DFD), а також декомпозицію цього процесу за допомогою діаграми IDEF0 [19].

Діаграма потоків даних (DFD), представлена на рисунку 2.1, описує загальну структуру процесу перегляду зображень [20], що включає в себе основні етапи, від вхідних даних до кінцевого результату.

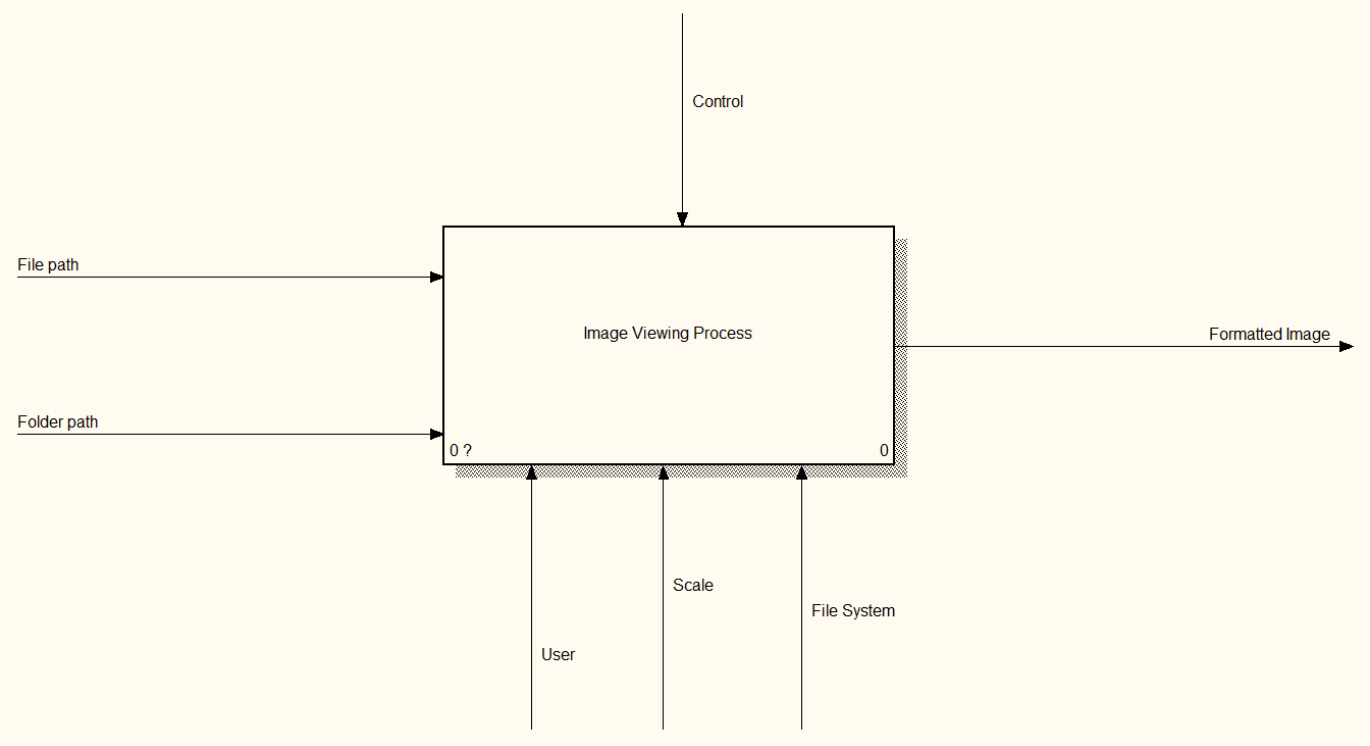

Рисунок 2.1 – Діаграма потоків даних модуля перегляду зображень

У центрі цієї діаграми знаходиться процес «Перегляд зображення», який є основним етапом обробки даних у системі. Вхідними даними для цього процесу є такі параметри, як шлях до файлу, шлях до папки, а також дані користувача, що ініціює операцію перегляду. Крім того, додатковими елементами є файлові системи, які забезпечують доступ до зображень, а також зовнішні фактори, такі як контрольні сигнали, що керують процесом.

В результаті роботи цього процесу на виході формується оброблене зображення, яке подається користувачу.

Для більш детального аналізу було застосовано методологію IDEF0, що дозволяє структурувати і описати процеси з чітким визначенням їх взаємозв'язків. Діаграма IDEF0, представлена на рисунку 2.2, демонструє детальну декомпозицію основного процесу «Перегляд зображення» на три основні модулі: 1) «Завантаження зображення»; 2) «Обробка зображення»; 3) «Форматування зображення».

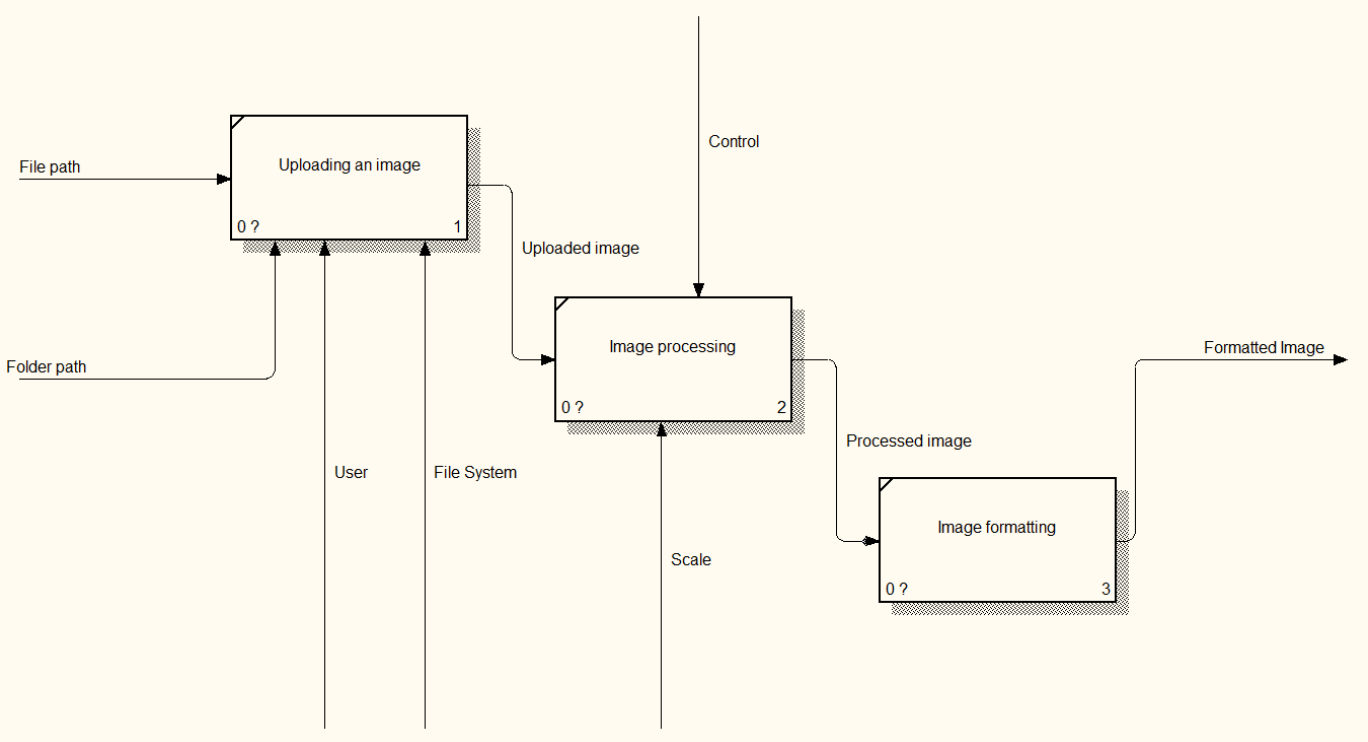

Рисунок 2.2 – Діаграма декомпозиції IDEF0 модуля перегляду зображень

Перший модуль – «Завантаження зображення», отримує вхідні дані, такі як шлях до файлу, шлях до папки, а також дані від користувача і файлової системи. На цьому етапі система завантажує зображення, перевіряючи наявність файлів за вказаними шляхами, і передає його далі для обробки.

Другий модуль – «Обробка зображення», відповідає за застосування операцій обробки, таких як масштабування, обертання чи інші перетворення зображення. Вхідними даними для цього процесу є зображення, передане з попереднього етапу, а також параметри масштабу та контрольні сигнали. Результатом цього етапу є оброблене зображення, яке передається для подальшого форматування.

Третій модуль – «Форматування зображення», приймає оброблене зображення і виконує необхідні перетворення для приведення його до кінцевого вигляду, який буде відображений на екрані користувача. На виході отримується відформатоване зображення, готове до відображення.

Також було побудовано діаграму сутностей та зв’язків (ERD) [25], яка надає візуальне уявлення про основні компоненти програми та їх взаємодії. На рисунку 2.3 представлена ERD, що відображає сутності та зв’язки між ними, які визначають структуру програми для перегляду зображень.

Ця діаграма включає в себе чотири основні сутності: 1) вікно програми; 2) зображення; 3) папка; 4) файлова система. Кожна з цих сутностей має свої атрибути та визначені зв’язки з іншими елементами системи.

Вікно програми є центральною сутністю, яка відповідає за відображення зображення на екрані користувача. Атрибути цієї сутності включають: 1) поточне зображення; 2) коефіцієнт масштабу; 3) розмір вікна. Вікно програми має дві основні дії: 1) відображає зображення, яка виводить зображення на екран; 2) запитує зображення, що забезпечує взаємодію з файловою системою для отримання потрібного файлу.

Зображення - містить атрибути, такі як: 1) шлях в файловій системі; 2) поточний масштаб, Роздільна здатність. Зображення не має дій, але воно взаємодіє з іншими сутностями через зв'язки, визначені в системі.

Папка - містить зображення і має атрибути, такі як шлях в файловій системі Основною дією цієї сутності є зображення, що вказує на те, що папка може зберігати зображення, організовуючи їх за певними шляхами.

Файлова система є контейнером для папок і зображень. Атрибутами цієї сутності є вільний простір. Дія містить папку та показує, що файлову систему можна розглядати як контейнер для папок, кожна з яких може містити зображення.

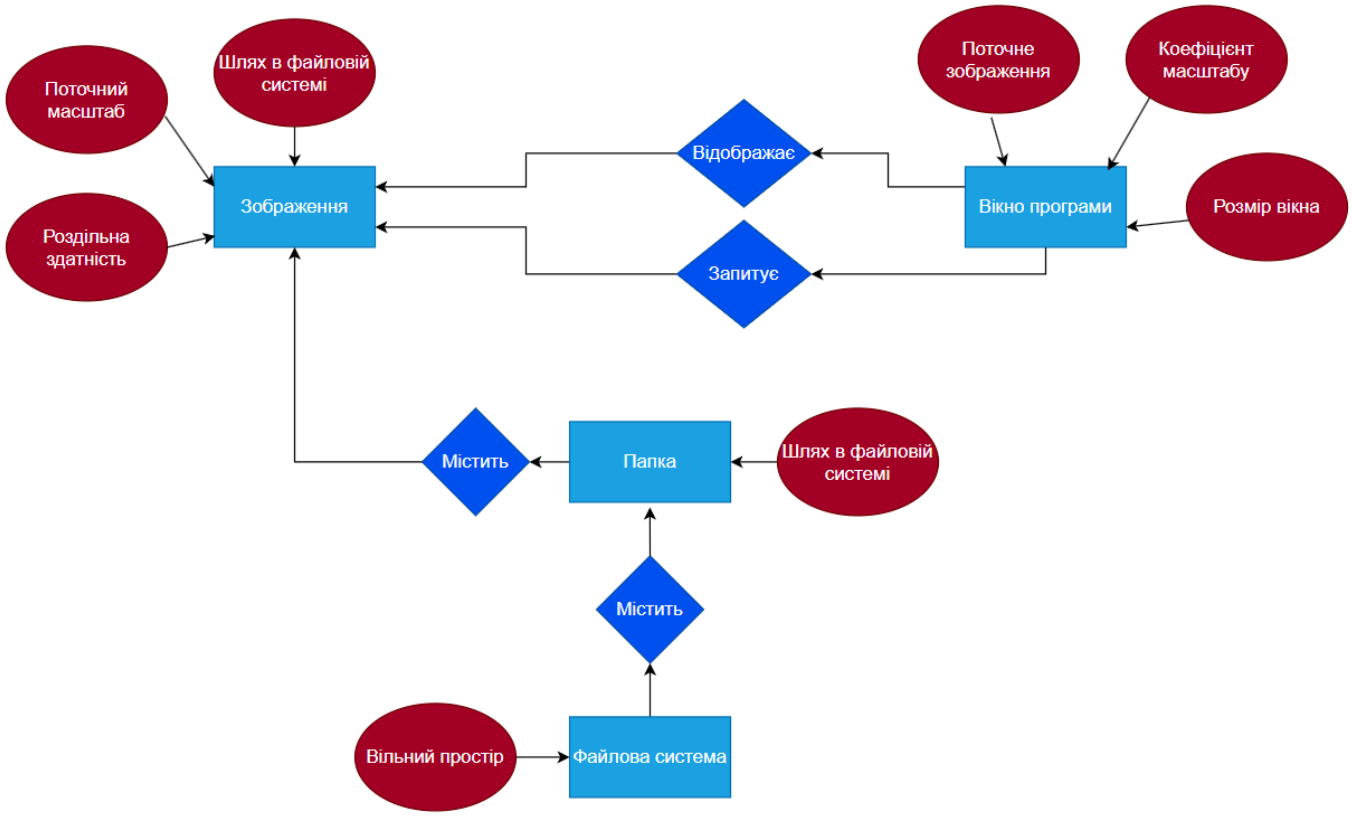

Рисунок 2.3 – ERD діаграма модуля перегляду зображень

Програмний модуль можна представити й у вигляді класів [10], що забезпечують функціонування основних компонентів системи. На рисунку 2.4 представлена діаграма класів, що відображає основні класи, їх атрибути та методи, а також взаємодії між ними.

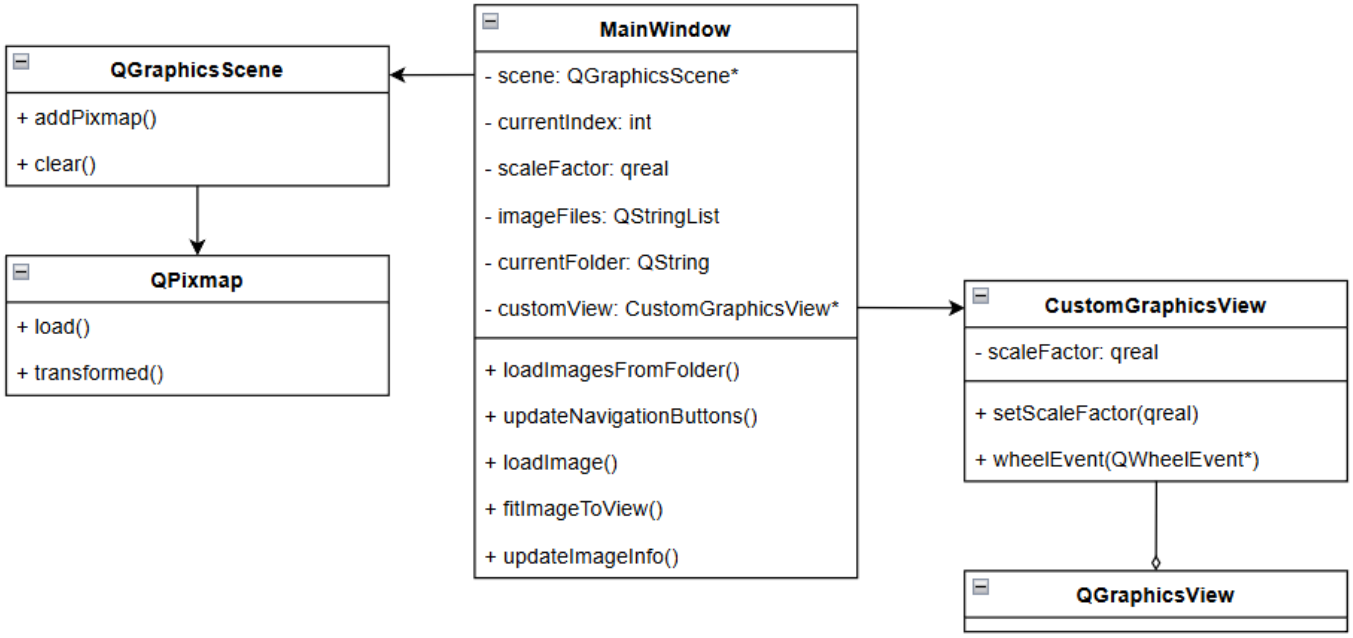

Рисунок 2.4 –Діаграма класів модуля перегляду зображень

Основними класами є MainWindow, CustomGraphicsView, QGraphicsScene та QPixmap. Клас MainWindow є головним компонентом, який керує іншими елементами системи, зокрема, сценою для відображення зображень та спеціалізованим класом CustomGraphicsView, який додає функціональність масштабування та прокручування зображень. Клас MainWindow включає атрибути для збереження поточного зображення, списку зображень, індексу поточного зображення, а також сцени для відображення зображень. Крім того, клас містить методи для завантаження зображень, оновлення навігаційних кнопок та коригування відображення зображень відповідно до розміру вікна.

Клас CustomGraphicsView є підкласом QGraphicsView і реалізує додаткові можливості для зміни масштабу зображення за допомогою анімації при прокручуванні миші. Він містить метод для встановлення коефіцієнта масштабу та обробки подій прокручування.

Клас QGraphicsScene використовується для управління відображенням елементів на сцені, зокрема, для додавання зображень та очищення сцени перед

завантаженням нового зображення. Клас QPixmap відповідає за завантаження та перетворення зображень, які відображаються на сцені.

Зв’язки між класами на діаграмі відображаються у вигляді асоціацій, де, наприклад, клас MainWindow має об’єкти класів CustomGraphicsView та QGraphicsScene, що вказує на їх використання в рамках головного класу. Клас CustomGraphicsView є підкласом QGraphicsView, що розширює функціональність роботи з графічними елементами на сцені.

2.2 Алгоритмічне забезпечення модуля

Алгоритм роботи модуля для перегляду зображень (рисунок 2.5) починається з вибору користувачем варіанту дії — фото або папка. Після цього система перевіряє, чи існує вказана папка в файловій системі. Якщо папка не існує, процес завершується, і користувач отримує повідомлення про помилку. У разі, коли папка існує, наступним кроком перевіряється наявність зображень підтримуваних форматів у вибраній папці. Якщо зображення відсутні або папка порожня, також виводиться відповідне повідомлення про помилку, і алгоритм завершується.

Якщо вибрано конкретне фото, система завантажує всі зображення з папки і на основі індексу вибраного фото відображає саме його. Інші зображення з цієї ж папки також доступні для перегляду через навігаційні кнопки, які дозволяють переміщатися між зображеннями. У разі вибору папки, система завантажує всі зображення з неї і відображає перше зображення, при цьому також активуються кнопки навігації для переміщення по зображеннях.

Після того, як зображення завантажені і відображено перше або вибране фото, на екрані з’являються кнопки навігації. Ці кнопки дозволяють переміщатися вперед і назад між зображеннями. При натисканні кнопки «Далі» відображається наступне зображення в списку, а при натисканні кнопки «Назад» — попереднє. У разі досягнення кінця або початку списку зображень система не дозволяє переміщатися далі або назад і надає користувачу відповідну інформацію.

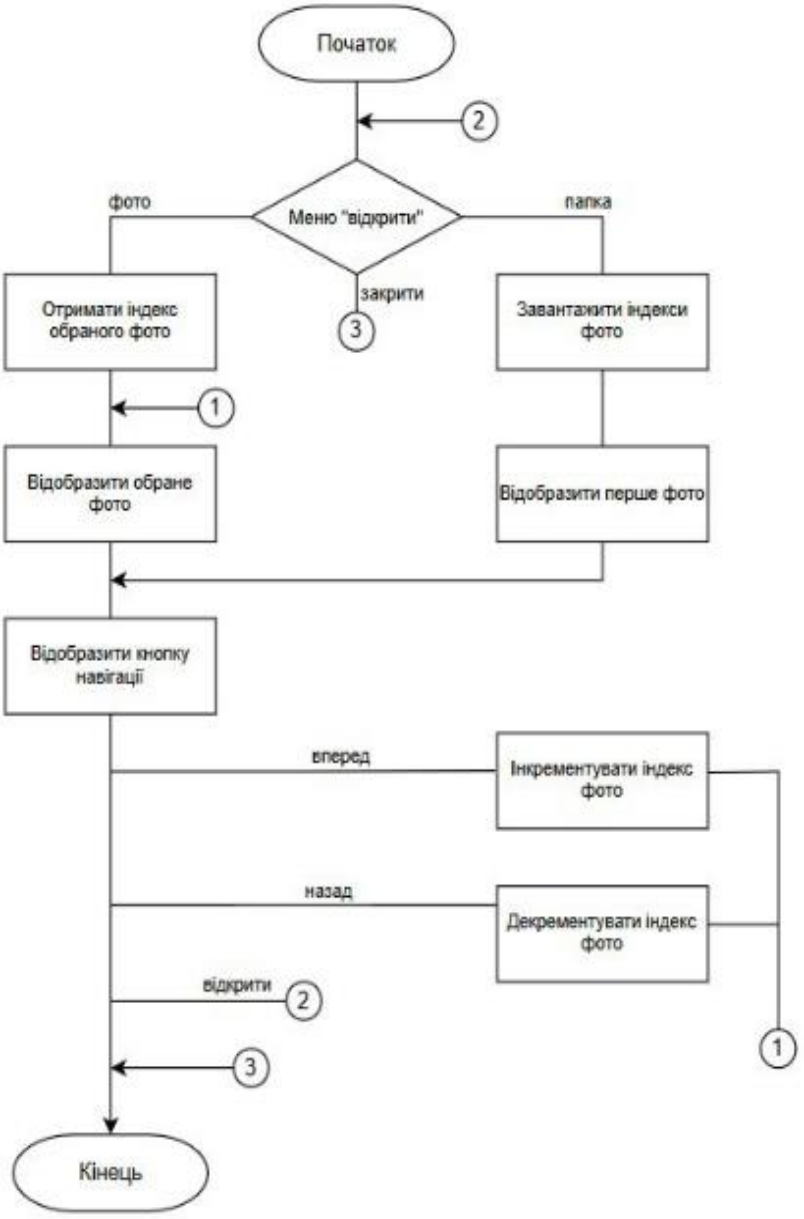

Рисунок 2.5 –Схема роботи модуля перегляду зображень

Алгоритм також враховує дії користувача, які можуть включати зміни масштабу зображення або закриття програми. У разі зміни масштабу система коригує розміри зображення відповідно до нових параметрів. Алгоритм завершується, коли користувач закриває програму або виконує інші операції, такі як зміна вибору фото чи папки для перегляду.

2.3 Інформаційне забезпечення модуля

Модуль для перегляду зображень обробляє вхідну інформацію, яка складається з файлів зображень, що знаходяться в обраній папці, а також самих параметрів вибору фото чи папки, які визначає користувач. Вхідні дані включають в себе наступну інформацію:

– шлях до папки або шлях до конкретного фото, обраний користувачем для перегляду;

– список файлів зображень, які знаходяться в обраній папці;

– індекс вибраного зображення, що визначає, яке саме зображення має бути відображено першим.

Процес обробки даних здійснюється через завантаження файлів зображень у систему, перевірку їх на відповідність підтримуваним форматам, а також перевірку їх наявності в обраній папці.

Вихідною інформацією є:

– зображення, що відображається на екрані, яке обирається відповідно до індексу вибраного фото або в разі вибору папки — перше зображення з цієї папки;

– кнопки навігації, що дозволяють користувачу переміщатися між зображеннями вперед і назад;

– повідомлення про помилки, якщо папка не існує, не містить зображень або має інші проблеми, що унеможливлюють подальшу роботу.

Глобальна даталогічна модель даних для модуля перегляду зображень включає елементи, що описують взаємодію між компонентами програми. Модель передбачає взаємодію таких сутностей, як «Фото», «Папка» та «Індекс фото».

Сутність «Фото» містить дані про конкретне зображення, включаючи його розмір, формат, шлях до файлу та інші характеристики. Сутність «Папка» містить дані про каталог, що містить зображення, включаючи шлях до папки і перелік файлів, що знаходяться в ній. Сутність «Індекс» фото зберігає інформацію про поточний вибір зображення для перегляду.

Модель даних виглядатиме наступним чином:

1. Папка має зв'язок "has-a" з Фото, тобто папка містить зображення.

2. Фото пов'язане з «Індексом фото», де індекс визначає, яке зображення відображається в даний момент.

Цей зв'язок візуалізовано за допомогою діаграми класів на рисунку 2.4.

3 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЯ

3.1 Реалізація програмного модуля

Програмний модуль для перегляду та обробки зображень розробляється на мові програмування C++ з використанням бібліотеки Qt. Qt є інструментом для створення графічних інтерфейсів та забезпечення взаємодії з користувачем [26]. Для розробки програмного забезпечення використовуватиметься інтегроване середовище розробки (IDE) Qt Creator [20], яке надає всі необхідний інструментарій для написання, налагодження та компіляції програм.

Використання C++ дозволить забезпечити продуктивність програми, а бібліотека Qt створити інтерфейс. Вона надає готові рішення для роботи з графічними елементами, файловими операціями, обробкою подій і створенням віконних додатків.

Qt Creator, в свою чергу, є інструментом для розробки на C++ з підтримкою Qt, що включає редактор лістингу, інструменти для дебаггінгу, профілювання та керування проектами.

Реалізація програмного інтерфейсу розпочинається зі створення класу CustomGraphicsView, який є похідним від базового класу QGraphicsView [17]. Метою створення цього класу є забезпечення можливості перевизначення стандартних методів роботи з графічними елементами, зокрема методу wheelEvent [23], для додавання функціональності масштабування зображень за допомогою колеса прокрутки миші.

Використання власного класу дозволяє реалізувати поведінку, якої не підтримує стандартна реалізація QGraphicsView. Це забезпечує контроль за масштабуванням графічних елементів сцени, реалізацію плавного анімованого переходу між різними рівнями масштабу та інтеграцію цих функцій з іншими компонентами графічного інтерфейсу додатку.

Для цього створюється заголовковий файл класу CustomGraphicsView, у якому визначаються публічні та приватні методи, приватні змінні, а також сигнали,

необхідні для взаємодії з іншими компонентами програми. Лістинг програми наведений у дадатку А (лістинг А.1). Заголовковий файл customgraphicsview.h містить наступні фенкції та методи:

– змінна scaleFactor, що зберігає поточний коефіцієнт масштабу;

– конструктор класу для ініціалізації об'єкта;

– метод setScaleFactor для встановлення нового коефіцієнта масштабу;

– властивість scaleFactor, яка використовується для інтеграції з анімаціями QPropertyAnimation;

– сигнал scaleFactorChanged, що повідомляє про зміну масштабу;

– перевизначення методу wheelEvent, що дозволяє реалізувати специфічну поведінку при обробці подій прокручування миші.

Методи setScaleFactor та перевизначений wheelEvent реалізуються у файлі customgraphicsview.cpp, який є файлом реалізації для класу CustomGraphicsView. У цьому файлі міститься функціональний лістинг, що забезпечує виконання визначених у заголовковому файлі методів і логіку роботи класу. Повний текст файлу customgraphicsview.cpp наведено у додатку А (лістинг А.2).

Метод wheelEvent перевизначено для додавання можливості обробки прокручування миші в поєднанні з клавішею Ctrl. Цей метод забезпечує функціональність масштабування зображень у графічному інтерфейсі. Лістинг методу wheelEvent для класу CustomGraphicsView наведено в лістингу 3.1.

Метод починається з перевірки, чи натиснута клавіша Ctrl, за допомогою виклику event->modifiers() == Qt::ControlModifier [21]. Якщо ця умова виконується, викликається event->ignore(), що вказує системі, що стандартна обробка події не потрібна. Це дозволяє реалізувати власну логіку для події прокручування.

Далі метод отримує величину прокручування в одиницях "кліків миші" за допомогою event->angleDelta().y(). Це значення використовується для обчислення коефіцієнта масштабування factor, який приводить величину прокручування до стандартного діапазону для одного кроку колеса миші.

Лістинг 3.1 – Реалізація методу обробки події прокручування миші в CustomGraphicsView

```
void CustomGraphicsView::wheelEvent(QWheelEvent *event)
{
    if (event->modifiers() == Qt::ControlModifier) {
        event->ignore();

        qreal angleDelta = event->angleDelta().y();
        qreal factor = 1.0 + (angleDelta / 120.0) * 0.1;

        qreal currentScale = transform().m11();

        qreal newScale = currentScale * factor;

        QPropertyAnimation *animation =
        new QPropertyAnimation(this, "scaleFactor");
        animation->setDuration(50);
        animation->setStartValue(currentScale);
        animation->setEndValue(newScale);
        animation->setEasingCurve(QEasingCurve::InOutQuad);
        animation->start();

        event->accept();
    } else {
        QGraphicsView::wheelEvent(event);
    }
}
```

Поточний масштаб отримується з матриці трансформації transform() за допомогою виклику m11(). Цей метод повертає значення коефіцієнта масштабування по осі X. Потім обчислюється новий масштаб шляхом множення поточного масштабу на коефіцієнт factor: qreal newScale = currentScale * factor.

Для плавного переходу між значеннями масштабу використовується клас QPropertyAnimation [19]. Об'єкт animation ініціалізується як нова анімація для властивості "scaleFactor". Далі встановлюються її параметри:

– animation->setDuration(50) задає тривалість анімації в мілісекундах.

– animation->setStartValue(currentScale) визначає початкове значення для анімації.

– animation->setEndValue(newScale) визначає кінцеве значення.

– animation->setEasingCurve(QEasingCurve::InOutQuad) задає криву сповільнення та прискорення для плавності переходу.

Анімація запускається викликом animation->start().

Подія завершується методом event->accept(), який вказує системі, що подія оброблена і подальша її обробка не потрібна.

Якщо клавіша Ctrl не натиснута, викликається базова реалізація QGraphicsView::wheelEvent(event), яка забезпечує стандартну обробку події прокручування миші.

Для додавання класу CustomGraphicsView у проєкт необхідно повідомити компілятору про місцезнаходження його файлів. Це здійснюється шляхом указання директорії проєкту, де розміщено файли з вихідним лістингом.

Вказівка директорії здійснюється у файлі CMakeLists.txt через команду include_directories(${CMAKE_CURRENT_SOURCE_DIR}). Ця команда додає поточну директорію проєкту, задану змінною ${CMAKE_CURRENT_SOURCE_DIR}, до списку шляхів, у яких компілятор шукатиме заголовкові файли.

Основні методи для роботи додатку реалізуються у класі MainWindow. Клас MainWindow є основним класом графічного інтерфейсу додатку, який відповідає за створення головного вікна, управління сценами, відображення зображень і реалізацію базової логіки взаємодії користувача з програмою. Цей клас наслідує QMainWindow [18].

Лістинг заголовкового файлу mainwindow.h наведено у додатку А (лістинг А.3)

Файл містить включення заголовкових файлів QMainWindow, QGraphicsScene та customgraphicsview.h, які забезпечують доступ до функціональності відповідних класів Qt. Простір імен Ui використовується для згенерованого інтерфейсу, який створюється за допомогою засобів Qt Designer.

Клас MainWindow визначений як похідний від класу QMainWindow. Цей клас містить опис змінних і методів, які забезпечують роботу програми. Зокрема, вказівник Ui::MainWindow відповідає за доступ до інтерфейсу. Для створення сцени

використовується об'єкт QGraphicsScene [13]. Рядок currentFolder зберігає шлях до поточної папки. Список QStringList imageFiles містить файли із зображеннями, доступні для перегляду. Індекс поточного зображення зберігається у змінній currentIndex. Масштаб зображення задається параметром scaleFactor. Для інтеграції кастомізованого переглядача використовується вказівник CustomGraphicsView.

Конструктор MainWindow із параметром QWidget ініціалізує всі необхідні елементи, тоді як деструктор забезпечує звільнення динамічно виділеної пам'яті.

Метод resizeEvent перевизначає обробку зміни розміру вікна, що дозволяє автоматично підлаштовувати елементи інтерфейсу до нового розміру.

Приватні методи включають fitImageToView, який забезпечує підгонку зображення під розміри вікна, та loadImagesFromFolder, що виконує завантаження зображень із заданої папки. Метод updateNavigationButtons реалізує оновлення кнопок для переходу між зображеннями, а setScrollBarsVisible дозволяє керувати видимістю смуг прокручування. Оновлення інформації про поточне зображення виконується за допомогою методу updateImageInfo.

Слоти обробляють події, такі як завантаження зображень, натискання кнопок «Назад», «Вперед», обертання зображення вліво чи вправо, а також зміну вибраного елемента в комбінованому списку.

Реалізація методів класу MainWindow здійснюється у файлі mainwindow.cpp. Початок цього файлу містить конструктор класу MainWindow, який відповідає за ініціалізацію елементів графічного інтерфейсу та початкових параметрів програми. Код конструктора наведено в лістингу 3.2.

У конструкторі спочатку викликається базовий конструктор QMainWindow з переданим батьківським об'єктом parent. Ініціалізується об'єкт ui для доступу до елементів інтерфейсу, автоматично створених класом Ui::MainWindow. Створюється новий об'єкт QGraphicsScene, який задає область для розміщення графічних елементів. Ця сцена прив'язується до кастомізованого виджета CustomGraphicsView через вказівник customView.

Лістинг 3.2 – Конструктор класу MainWindow

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
    , scene(new QGraphicsScene(this))
    , currentIndex(0)
    , scaleFactor(1.0)
{
    ui->setupUi(this);

    customView = ui->graphicsView;
    customView->setScene(scene);
    customView->setRenderHint(QPainter::Antialiasing);
    customView->setRenderHint(QPainter::SmoothPixmapTransform);
    customView->setDragMode(QGraphicsView::ScrollHandDrag);
    customView->
    setTransformationAnchor(QGraphicsView::AnchorUnderMouse);
}
```

Виклик ui->setupUi(this) забезпечує ініціалізацію графічного інтерфейсу, створеного в Qt Designer. Метод setScene асоціює об'єкт scene із графічним видом customView. Установлюються параметри відображення, зокрема згладжування контурів за допомогою QPainter::Antialiasing та плавна трансформація пікселів через QPainter::SmoothPixmapTransform. Метод setDragMode задає режим перетягування елементів, а setTransformationAnchor встановлює якір трансформації у положенні курсора миші.

Ініціалізація відкриття файлу або папки здійснюється через вибір відповідної опції в комбінованому списку comboBox. Для цього реалізовано обробник події, що спрацьовує при зміні індексу вибраного елементу в comboBox. Логіка обробки вибору елементів описана у методі on_comboBox_currentIndexChanged(int index), який наведено в лістингу 3.3.

При виборі першого елемента "Відкрити..." не виконується жодних дій, оскільки цей пункт лише відображає початковий стан. У разі вибору другого елемента "Фото" відкривається діалогове вікно для вибору окремого зображення за допомогою QFileDialog::getOpenFileName. Якщо файл був вибраний, визначається шлях до файлу та шлях до папки, в якій він знаходиться, через QFileInfo. Далі

викликається метод loadImagesFromFolder(folderPath), який завантажує всі зображення з обраної папки.

Лістинг 3.3 – Обробник зміни індексу вибраного елементу в comboBox

```
void MainWindow::on_comboBox_currentIndexChanged(int index)
{
    switch (index) {
    case 0:
        break;
    case 1:
    {
        QString selectedFile =
        QFileDialog::getOpenFileName(this, "Виберіть фото",
        currentFolder, "Images (*.png *.jpg *.jpeg *.bmp)");
        if (!selectedFile.isEmpty()) {
            QFileInfo fileInfo(selectedFile);
            QString folderPath = fileInfo.absolutePath();
            loadImagesFromFolder(folderPath);
            currentIndex = imageFiles.
            indexOf(fileInfo.fileName());
            if (currentIndex == -1) {
                currentIndex = 0;
            }
            loadImage();
        }
        updateNavigationButtons();
        break;
    }
    case 2:
    {
        QString folderPath = QFileDialog::getExistingDirectory
        (this, "Виберіть папку з зображеннями", currentFolder);
        if (!folderPath.isEmpty()) {
            loadImagesFromFolder(folderPath);
        }
        updateNavigationButtons();
        break;
    }
    default:
        break;
    }
    ui->comboBox->setCurrentIndex(0);
}
```

Після завантаження списку зображень визначається індекс вибраного зображення в списку за допомогою методу indexOf. Якщо файл не знайдений,

встановлюється перше зображення за замовчуванням. Після цього викликається метод loadImage(), що завантажує та відображає вибране зображення.

У разі вибору третього елемента "Папка" відкривається діалогове вікно для вибору директорії через QFileDialog::getExistingDirectory. Якщо папка була обрана, викликається той самий метод loadImagesFromFolder, який завантажує зображення з нової директорії. Після кожної операції оновлюється стан кнопок навігації між зображеннями за допомогою updateNavigationButtons(). По завершенні обробки вибору скидається індекс comboBox на початкове значення, що дозволяє користувачу повторно обрати іншу опцію. Код, згаданого вище методу loadImagesFromFolder наведено в лістингу 3.4.

Лістинг 3.4 – Лістинг методу завантаження зображень з вказаної папки

```
void MainWindow::loadImagesFromFolder(const QString &folderPath)
{
    imageFiles.clear();
    currentFolder = folderPath;
    QDir dir(currentFolder);
    imageFiles = dir.entryList(
    QStringList() << "*.png" << "*.jpg" << "*.jpeg" << "*.bmp",
    QDir::Files);
    if (!imageFiles.isEmpty()) {
        currentIndex = 0;
        loadImage();
    }
    updateNavigationButtons();
}
```

Тут, спочатку очищається список imageFiles, щоб видалити попередньо завантажені файли. Далі оновлюється поточний шлях до папки через змінну currentFolder, що містить шлях до переданої папки.

Для отримання списку зображень у папці використовується клас QDir. За допомогою методу entryList зчитуються всі файли з розширеннями .png, .jpg, .jpeg та .bmp, що знаходяться в директорії. Ці файли додаються до списку imageFiles, який містить всі доступні зображення для подальшого використання.

Після отримання списку файлів перевіряється, чи є в папці зображення. Якщо зображення знайдені, індекс currentIndex встановлюється на 0, що означає перше зображення в списку. Потім викликається метод loadImage(), код якого наведено в лістингу 3.5, для завантаження та відображення першого зображення.

Лістинг 3.5 – Лістинг методу для завантаження зображення

```
void MainWindow::loadImage()
{
    if (currentIndex >= 0 && currentIndex < imageFiles.size()) {
        QString filePath = currentFolder + "/" +
        imageFiles[currentIndex];
        QPixmap pixmap(filePath);
        if (!pixmap.isNull()) {
            scene->clear();
            scene->addPixmap(pixmap);
            fitImageToView();
            updateImageInfo(filePath);
        }
    }
}
```

Як видно з лістингу 3.5, метод loadImage() виконується при умові, що значення currentIndex знаходиться в межах допустимого діапазону, тобто більше або дорівнює 0 та менше розміру списку imageFiles. Це забезпечує правильність індексації файлів у списку зображень.

Якщо індекс є валідним, складається повний шлях до файлу зображення через з'єднання змінної currentFolder з ім'ям файлу, яке міститься у imageFiles[currentIndex]. Створюється об'єкт QPixmap на основі цього шляху, який містить зображення.

Далі перевіряється, чи зображення завантажилось коректно за допомогою методу isNull(). Якщо зображення було завантажене успішно, виконується кілька операцій. Спочатку очищається сцена scene за допомогою методу clear(), щоб попереднє зображення було видалено перед додаванням нового. Потім зображення додається на сцену за допомогою addPixmap(pixmap).

Після цього викликається метод fitImageToView(), який підганяє розміри зображення до розмірів вікна перегляду, забезпечуючи правильне відображення в межах доступної області. Наприкінці викликається метод updateImageInfo(filePath), який оновлює інформацію про поточне зображення, таку як його розмір, роздільна здатність та інші параметри. Реалізація цих двох методів: fitImageToView та updateImageInfo наведено в лістингу 3.6.

Лістинг 3.6 – Реалізація методів fitImageToView та updateImageInfo

```
void MainWindow::fitImageToView()
{
    if (!scene->items().isEmpty()) {
        QRectF sceneRect = scene->itemsBoundingRect();
        customView->setSceneRect(sceneRect);
        customView->fitInView(sceneRect, Qt::KeepAspectRatio);
        scaleFactor = 1.0; // Скидаємо масштаб
    }
}
void MainWindow::updateImageInfo(const QString &filePath)
{
    QFileInfo fileInfo(filePath);
    QString fileName = fileInfo.fileName();
    ui->labelTop->setText(fileName);
    QPixmap pixmap(filePath);
    QString resolution = QString::number(pixmap.width()) + " x " +
    QString::number(pixmap.height());
    ui->labelResolutionText->setText(resolution);
    qint64 fileSizeBytes = fileInfo.size();
    QString fileSize;
    if (fileSizeBytes < 1024 * 1024) {
        fileSize = QString::number(
        fileSizeBytes / 1024.0, 'f', 2) + " KB";
    } else {
        fileSize = QString::number(
        fileSizeBytes / (1024.0 * 1024.0), 'f', 2) + " MB";
    }
    ui->labelSizeText->setText(fileSize);
}
```

Метод fitImageToView перевіряє, чи містить сцена хоча б один елемент, викликаючи scene->items().isEmpty(). Якщо сцена не порожня, за допомогою методу itemsBoundingRect() отримується прямокутник, що охоплює всі елементи на сцені.

Цей прямокутник встановлюється як область перегляду за допомогою setSceneRect(sceneRect), що забезпечує правильне відображення вікна для сцени.

Далі виконується підгонка зображення до видимої області вікна через метод fitInView(sceneRect, Qt::KeepAspectRatio), що зберігає пропорції зображення при підгонці. В кінці скидається значення змінної scaleFactor до 1.0, що свідчить про відсутність масштабування.

Метод updateImageInfo оновлює інформацію про поточне зображення. Спочатку, за допомогою QFileInfo, отримується ім'я файлу, яке встановлюється у відповідну мітку інтерфейсу користувача через ui->labelTop->setText(fileName). Потім завантажується зображення за допомогою QPixmap для отримання його роздільної здатності. Роздільна здатність виводиться на екран у вигляді ширини та висоти зображення.

Для отримання розміру файлу використовується метод size() класу QFileInfo. Розмір файлу конвертується у кілобайти або мегабайти, залежно від величини, і виводиться у відповідній мітці через ui->labelSizeText->setText(fileSize).

Також реалізовано функціонал повороту зображень на 90 градусів за/проти годинникової стрілки через обробники подій для натискання відповідних кнопок. Код кожного з цих обробників подій наведено в лістингу 3.7.

Метод on_pushButtonRotateLeft_clicked виконується при натисканні кнопки для повороту зображення проти годинникової стрілки. Спочатку перевіряється, чи є елементи на сцені. Якщо елемент є, за допомогою dynamic_cast здійснюється приведення першого елемента сцени до типу QGraphicsPixmapItem, що дозволяє працювати з зображенням. Далі отримується поточне зображення через pixmapItem->pixmap(), після чого застосовується трансформація через QTransform, де задається поворот на 90 градусів проти годинникової стрілки. Зображення після повороту зберігається в нову змінну rotatedPixmap. Після цього оновлюється зображення на сцені за допомогою pixmapItem->setPixmap(rotatedPixmap). Метод fitImageToView викликається для підгонки розміру зображення під видиму область.

Лістинг 3.7 – Реалізація методів для повороту зображення

```
void MainWindow::on_pushButtonRotateLeft_clicked()
{
    if (!scene->items().isEmpty()) {
        QGraphicsPixmapItem *pixmapItem = dynamic_cast
        <QGraphicsPixmapItem *>(scene->items().first());
        if (pixmapItem) {
            QPixmap pixmap = pixmapItem->pixmap();
            QTransform transform;
            transform.rotate(-90);
            QPixmap rotatedPixmap = pixmap.transformed(
            transform, Qt::SmoothTransformation);
            pixmapItem->setPixmap(rotatedPixmap);
            fitImageToView();
        }
    }
}
void MainWindow::on_pushButtonRotateRight_clicked()
{
    if (!scene->items().isEmpty()) {
        QGraphicsPixmapItem *pixmapItem = dynamic_cast
        <QGraphicsPixmapItem *>(scene->items().first());
        if (pixmapItem) {
            QPixmap pixmap = pixmapItem->pixmap();
            QTransform transform;
            transform.rotate(90);
            QPixmap rotatedPixmap = pixmap.transformed(
            transform, Qt::SmoothTransformation);
            pixmapItem->setPixmap(rotatedPixmap);
            fitImageToView();
        }
    }
}
```

Метод on_pushButtonRotateRight_clicked виконується аналогічно, але при натисканні кнопки для повороту зображення за годинниковою стрілкою. У цьому випадку трансформація задає поворот на 90 градусів за годинниковою стрілкою. Все інше виконується за аналогією з попереднім методом.

Лістинг файлу для ініціалізації додатку, вибору локалізації та запуску основного вікна розміщений у файлі main.cpp, у додатку А (лістинг А.4). Оскільки цей лістинг створений автоматично середовищем розробки, він не розглядається окремо.

3.2 Проектування інтерфейсу користувача

Розробка графічного інтерфейсу здійснюється за допомогою режиму дизайнера в Qt Creator. Основним віджетом є MainWindow, який має горизонтальну компоновку і містить три основні віджети:

- Віджет для ComboBox та Label, який використовується для виведення назви поточного зображення. Цей віджет має горизонтальну компоновку;
- віджет для CustomGraphicsView (де відображається зображення) та кнопок для перемикання між зображеннями. Віджет має вертикальну компоновку;
- віджет для Label, який виводить інформацію про зображення (розмір, роздільна здатність) та кнопки для повороту зображення. Цей віджет також має вертикальну компоновку.

Повний текст файлу mainwindow.ui наведено у додатку А (лістинг А.5).

Застосування стилів до окремих елементів інтерфейсу здійснюється за допомогою QSS (Qt Style Sheets). Вони дозволяють налаштовувати вигляд віджетів у Qt, використовуючи синтаксис, схожий на CSS, але з деякими обмеженнями. QSS дозволяє змінювати кольори, шрифти, межі, фони та інші властивості елементів інтерфейсу [22].

В лістингу 3.8 наведена стилізація таких елементів, як головне вікно (QMainWindow), лейбли для назви фотографії та додаткової інформації про фото (QLabel), а також для випадаючого списку (QComboBox).

Для кнопок, які відповідають за перемикання між фотографіями та їх поворот, використовуються власні іконки. Щоб це працювало, необхідно розмістити іконки в директорії проекту та створити файл resources.qrc [11], в якому визначається маршрут до папки з іконками /images. Після цього потрібно підключити іконки до ресурсів проекту, щоб вони могли бути використані в додатку.

Лістинг 3.8 – Стилізація елементів інтерфейсу

```
QMainWindow {
    background: rgb(32, 32, 32);
}

QLabel#labelTop {
    qproperty-alignment: 'AlignHCenter | AlignVCenter';
    color: rgb(255, 255, 255);
}

QLabel#labelSize {
    color: rgb(255, 255, 255);
}

QComboBox {
    color: white;
    border: 1px solid grey;
    border-radius: 3px;
    padding: 1px 18px 1px 3px;
    min-width: 6em;
}

QComboBox:editable {
    background: white;
}

QComboBox:!editable, QComboBox::drop-down:editable {
    background: #4399DB;
}

QComboBox:!editable:on, QComboBox::drop-down:editable:on {
    background: #4399DB;
}

QComboBox:on {
    padding-top: 3px;
    padding-left: 4px;
}
QComboBox::drop-down {
    subcontrol-origin: padding;
    subcontrol-position: top right;
    width: 15px;
    border-left-width: 1px;
    border-left-color: darkgray;
    border-left-style: solid;
    border-top-right-radius: 3px;
    border-bottom-right-radius: 3px;
}

QComboBox QAbstractItemView {
```

```
    color: rgb(85, 170, 255);
    background-color: #373e4e;
    padding: 10px;
    selection-background-color: rgb(39, 44, 54);
}
```

Файл resources.qrc містить посилання на ресурси та вказує, де знаходяться відповідні файли. Вигляд вмісту файлу resources.qrc, коли він відкривається в режимі редактора ресурсів Qt Creator наведено на рисунку 3.1.

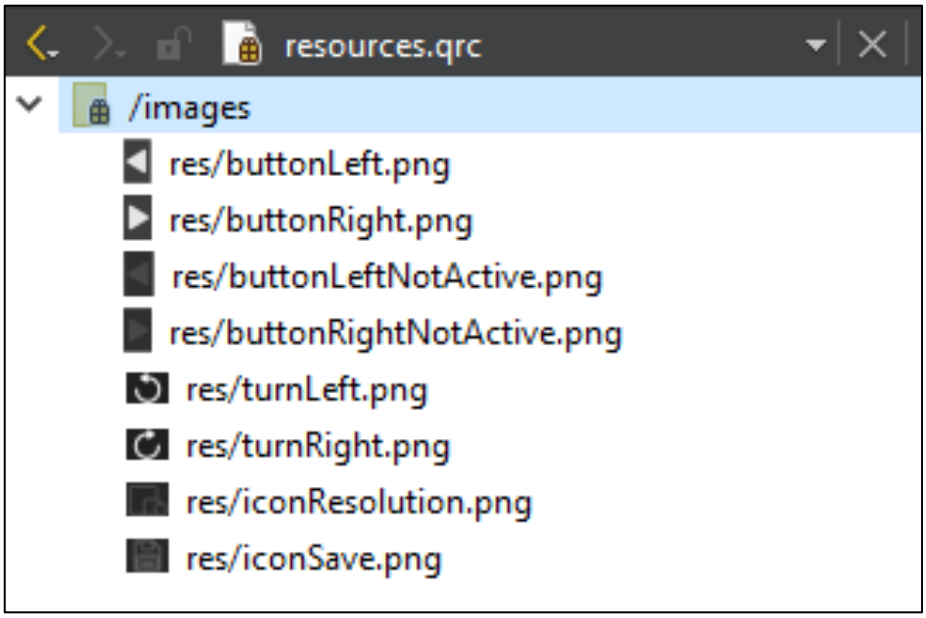

Рисунок 3.1 – Вміст файлу resources.qrc в редакторі ресурсів

Після створення файлу ресурсів, його можна використовувати для стилізації елементів інтерфейсу за допомогою QSS [9]. В лістингу 3.9 наведені стилі для кнопок, які відповідають за перемикання між фотографіями та їх повороти, із застосуванням ресурсних файлів для встановлення іконок на кнопки та інших стилів для цих елементів.

Лістинг 3.9 – Код стилів для кнопок

```
QPushButton#pushButtonBack {
    background-image: url(:/images/res/buttonLeftNotActive.png);
    border-radius: 7px;
}
QPushButton#pushButtonBack:hover {
    background-image: url(:/images/res/buttonLeft.png);
}
QPushButton#pushButtonBack:pressed {
    border-width: 1px;
    border-style: solid;
    border-color: white;
}
QPushButton#pushButtonForward {
    background-image: url(:/images/res/buttonRightNotActive.png);
    border-radius: 7px;
}
QPushButton#pushButtonForward:hover {
    background-image: url(:/images/res/buttonRight.png);
}
QPushButton#pushButtonForward:pressed {
    border-width: 1px;
    border-style: solid;
    border-color: white;
}
QPushButton#pushButtonRotateLeft {
    background-image: url(:/images/res/turnLeft.png);
    border-radius: 3px;
}
QPushButton#pushButtonRotateLeft:pressed {
    border-width: 1px;
    border-style: solid;
    border-color: white;
}
QPushButton#pushButtonRotateRight {
    background-image: url(:/images/res/turnRight.png);
    border-radius: 3px;
}
QPushButton#pushButtonRotateRight:pressed {
    border-width: 1px;
    border-style: solid;
    border-color: white;
}
```

У цих стилях для кнопок перемикання між зображеннями по замовчуванню використовуються зображення buttonLeftNotActive.png для кнопки повернення назад і buttonRightNotActive.png для кнопки переходу вперед, за допомогою параметрів

hover та background-image [24]. При наведенні на кнопку ці зображення змінюються на активні варіанти (buttonLeft.png та buttonRight.png). Змінюється також стилізація для кнопок при натисканні, додаючи білу обведену межу.

3.3 Тестування програмного модуля

Тестування розробленої програми полягає у перевірці коректності написання лістингу та успішної компіляції проекту. Під час тестування перевіряються всі основні функціональні можливості програми, зокрема, взаємодія з користувачем, коректне відображення зображень, обробка подій, таких як поворот зображень та перемикання між файлами. Також перевіряється правильність роботи стилізації інтерфейсу за допомогою QSS, в тому числі відображення кнопок та лейблів з відповідними стилями.

Після успішного компілювання програми відбувається запуск додатку, в результаті чого з'являється головне вікно програми, яке відповідає за відображення зображень та виконання навігаційних операцій. Зовнішній вигляд цього вікна продемонстровано на рисунку 3.2.

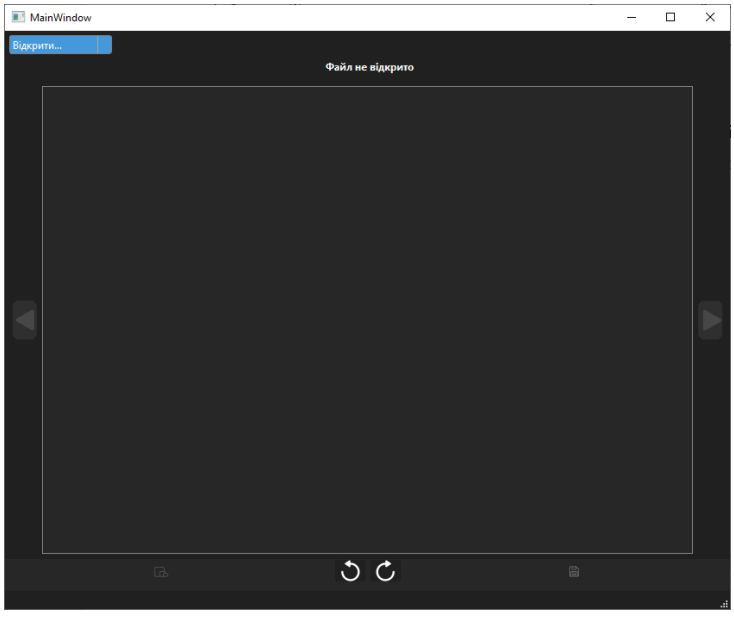

Рисунок 3.2 – Зовнішній вигляд додатку при початковому запуску

Наступним етапом є перевірка застосування стилів до елементів інтерфейсу. Одним з таких елементів є випадаючий список, до якого було застосовано відповідний стиль через QSS. Зовнішній вигляд випадаючого списку після застосування стилю продемонстровано на рисунку 3.3.

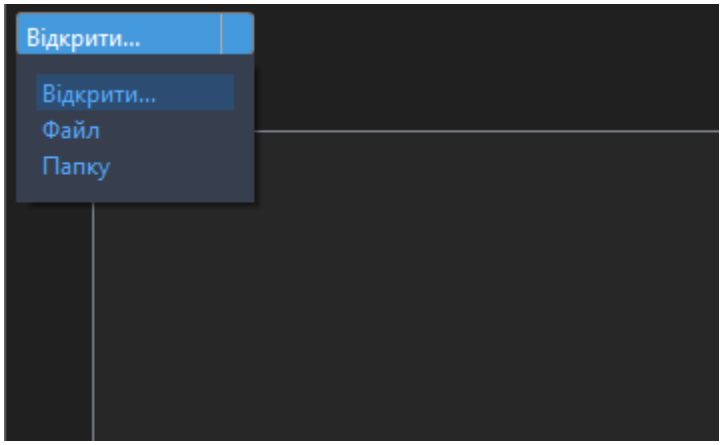

Рисунок 3.3 – Результат застосування стилів до випадающого списку

Також тестується функціональність відкриття конкретно вибраної фотографії. Після вибору фотографії з випадаючого списку, програма повинна завантажити та відобразити зображення в головному вікні. Результат цього процесу показано на рисунку 3.4.

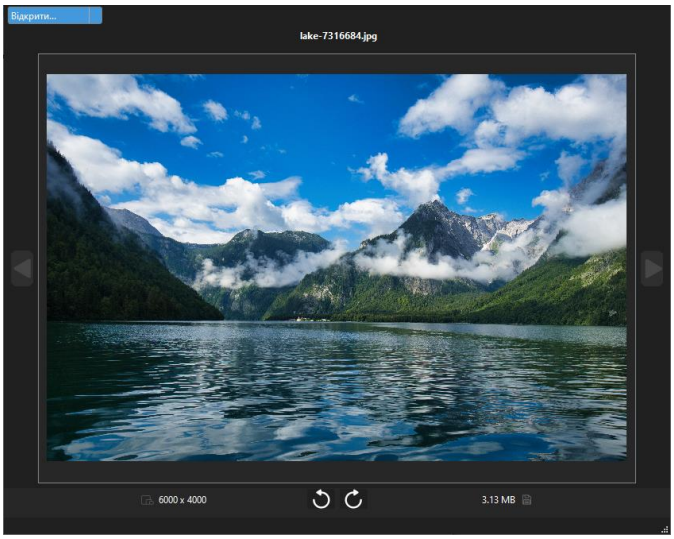

Рисунок 3.4 – Результат відкриття фотографії у додатку

Однією з задач було забезпечити адаптацію зображення до розміру вікна. Для цього треба перевірити, як програма обробляє відкриття вертикального зображення, яке має інші пропорції порівняно з горизонтальними. Результат цього тесту продемонстровано на рисунку 3.5.

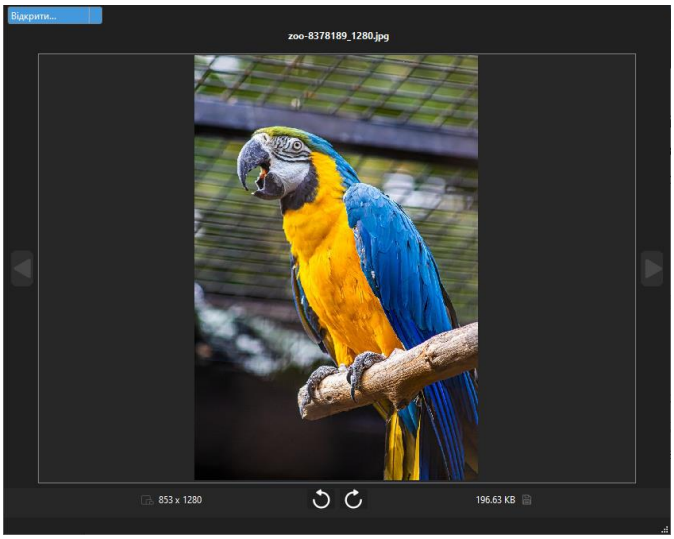

Рисунок 3.5 – Результат відкриття вертикального зображення

Також перевіряється, як зображення реагує на зміну розміру вікна. При зміні розміру вікна зображення автоматично підлаштовуватися під нові розміри, зберігаючи правильні пропорції. Результат цієї перевірки наведено на рисунку 3.6.

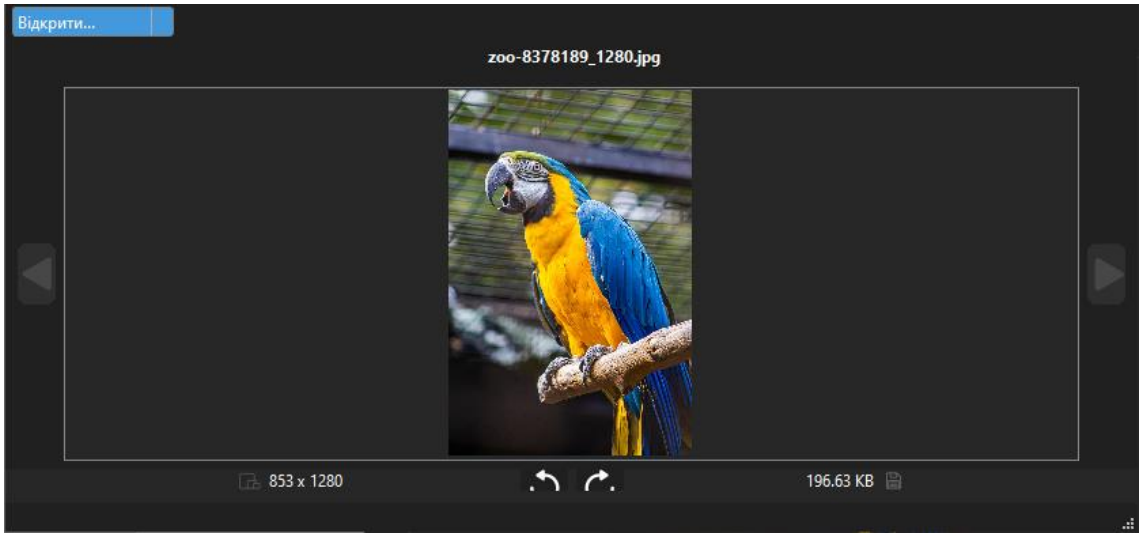

Рисунок 3.6 – Адаптація зображення під розміри вікна

Далі перевіряється функціонал повороту зображення на 90 градусів. Результат повороту зображення проти годинникової стрілки продемонстровано на рисунку 3.7. Цей тест перевіряє, як зображення змінюється після застосування обробника події для повороту на 90 градусів вліво.

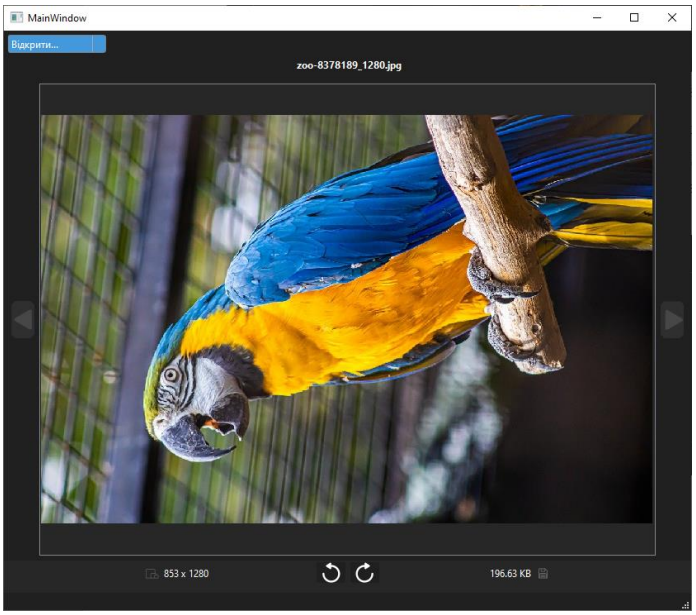

Рисунок 3.7 – Результат повороту зображення на 90 градусів вліво

На рисунку 3.8 продемонстровано дію від натискання кнопки «Назад». Цей тест перевіряє, як програма реагує на натискання кнопки для перемикання до попереднього зображення.

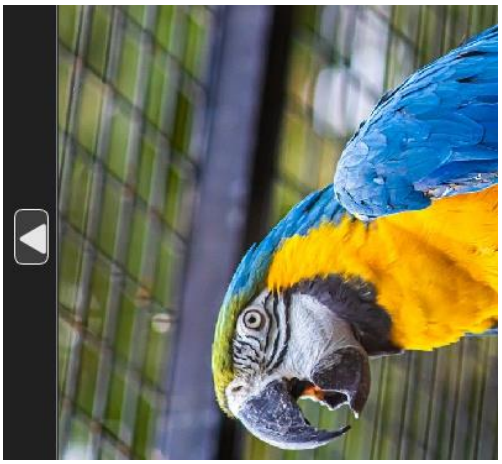

Рисунок 3.8 – Результат від натискання на кнопку «Назад»

Результат перемикання на попередню фотографію наведений на рисунку 3.9.

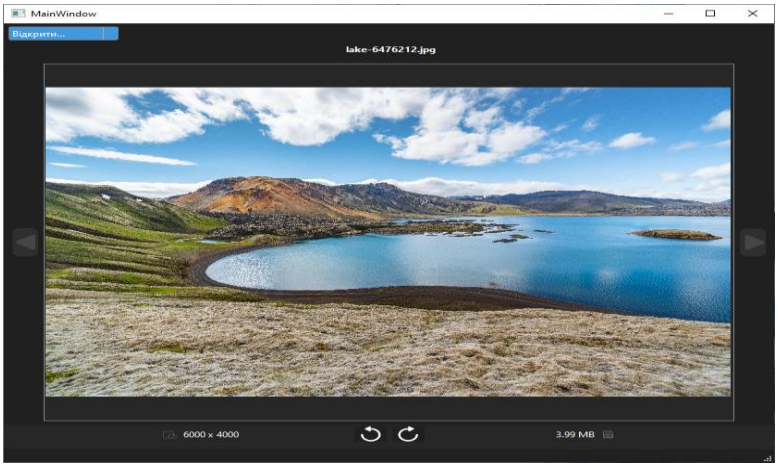

Рисунок 3.9 – Результат переключення на попереднє фото в директорії

На рисунку 3.10 продемонстровано перевірку роботу зуму, коли при утриманні клавіші Ctrl і прокручуванні колеса миші зображення збільшується збільшуватися.

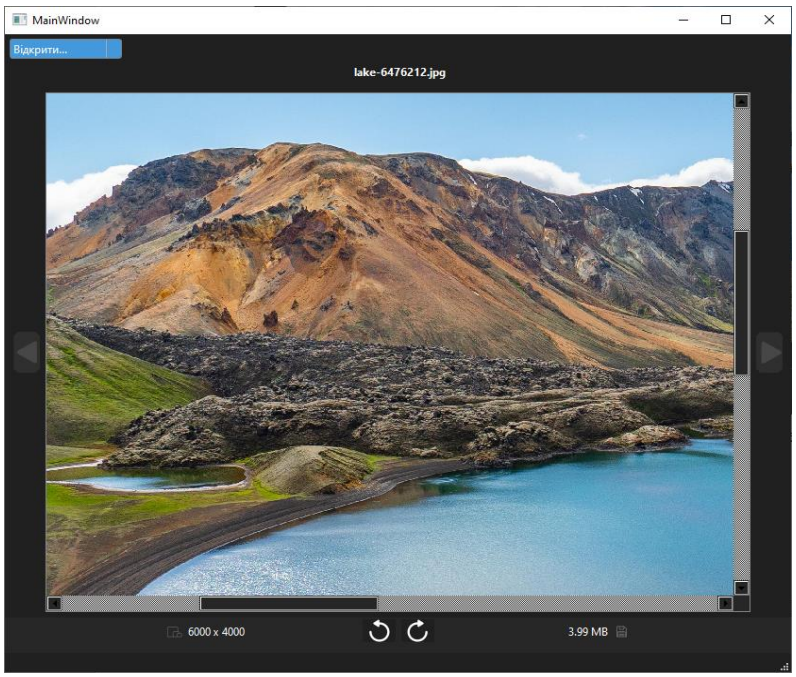

Рисунок 3.10 – Результат збільшення фотографії

Результати роботи опубліковані та представлені на міжнародній студентській конференції [27].

ВИСНОВКИ

Кваліфікаційна робота присвячена с створенню програмного модуля для перегляду зображень. Актуальність даної розробки зумовлена зростаючою потребою в простих та одночас ефективних програмних продуктах, що дозволяють переглядати зображення різних форматів, здійснювати базові операції над ними, такі як масштабування, навігація між зображеннями. Результатом розробки стала програма для перегляду зображень з підтримкою базових функцій навігації та управління зображеннями. Створений програмний модуль дозволяє користувачам швидко переглядати зображення, змінювати їхній вигляд за допомогою функцій зуму та повороту, а також переходити між зображеннями за допомогою кнопок навігації.

В процесі виконання кваліфікаційної роботи вирішено наступні завдання:

1. Проведено аналіз предметної області, що включає огляд представлених на ринку програмних продуктів провести аналіз програмних продуктів призначених для перегліду зображень, та сформулювати набір функціональних вимог до программ даного класу.

2. Розроблено архітектуру програмного модуля, зокрема алгоритмічне та інформаційне забезпечення, модель даних.

3. Реалізовано програмний модуль, втому числі користувацький інтерфейс із використанням Qt бібліотеки.

4. Проведене тестування розробленого модуля, зокрема корректного завантаження та відображення зображень, а також всіх функціональних мижливостей.

Результати роботи можуть бути використані в якості основи для подальшого вдосконалення додатків для перегляду зображень, зокрема для додавання нових функцій, таких як редагування зображень або підтримка додаткових форматів файлів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Браун Ф. Модель діаграми зв’язків сутностей. Guru99. URL: https://www.guru99.com/uk/er-diagram-tutorial-dbms.html (дата звернення: 15.03.2025).
2. Дерево деталізації функцій. Stud. URL: https://stud.com.ua/77210/informatika/derevo_detalizatsiyi_funktsiy (дата звернення: 15.03.2025).
3. Дідковська М. Проектування програмного забезпечення засобами UML : Навчальний посібник. Київ, 2020.
4. Засім М. Діаграми потоків даних. Maxym Zosym. URL: https://www.maxzosim.com/data-flow-diagrams/ (дата звернення: 15.03.2025).
5. Методологія idef0. Stud. URL: https://stud.com.ua/87184/ekonomika/metodologiya_idef0 (дата звернення: 15.03.2025).
6. Політика конфіденційності – Конфіденційність і умови. Google. URL: https://policies.google.com/privacy?hl=uk (дата звернення: 15.03.2025).
7. Швачич Г. Побудова блок - схем : Навчальний посібник. Дніпро, 2004.
8. C++/WinRT. Microsoft Learn. URL: https://learn.microsoft.com/en-us/windows/uwp/cpp-and-winrt-apis/ (дата звернення: 15.03.2025).
9. How to display a picture. Qt Forum. URL: https://forum.qt.io/topic/71836/how-to-display-a-picture (дата звернення: 15.03.2025).
10. Image Viewer XnView. XnView Software. URL: https://www.xnview.com/en/ (дата звернення: 19.12.2024).
11. Images | Qt Design Studio Documentation 4.6.2. Qt Documentation. URL: https://doc.qt.io/qtdesignstudio/quick-images.html (дата звернення: 16.12.2024).
12. IrfanView. Encyclopedia MDPI. URL: https://encyclopedia.pub/entry/31916 (дата звернення: 12.12.2024).
13. Johnson D. Simple QGraphicsView Example for QT. Qt Centre Forum. URL: https://www.qtcentre.org/threads/67155-Very-Simple-QGraphicsView-Example-for-QT5-7-QtCreator-mainwindow-s-main-cpp (дата звернення: 15.03.2025).

14. Microsoft Photos app. Microsoft Support. URL: https://support.microsoft.com/en-us/windows/get-help-with-microsoft-photos-app-702104c6-7541-6757-d624-c0846b3bdbd3 (дата звернення: 25.03.2025).

15. Overview XnView Wiki. XnSoft. URL: https://www.xnview.com/wiki/index.php?title=Overview (дата звернення: 10.12.2024).

16. Programming Language. XnView Software. URL: https://newsgroup.xnview.com/viewtopic.php?t=3168&p=14921#p14921 (дата звернення: 02.12.2024).

17. QGraphicsView Class. Doc Qt. URL: https://doc.qt.io/qt-6/qgraphicsview.html (дата звернення: 18.12.2024).

18. QMainWindow Class. Massachusetts Institute of Technology. URL: https://web.mit.edu/~firebird/arch/i386_rh9/doc/html/qmainwindow.html (дата звернення: 17.12.2024).

19. QPropertyAnimation Class. Doc Qt. URL: https://doc.qt.io/qt-6/qpropertyanimation.html (дата звернення: 18.12.2024).

20. Qt creator development tools. Qt. URL: https://www.qt.io/product/development-tools (дата звернення: 18.12.2024).

21. Qt Namespace. Doc Qt. URL: https://doc.qt.io/qt-6/qt.html (дата звернення: 17.12.2024).

22. QT Style Sheets. Packt. URL: https://www.packtpub.com/en-us/learning/how-to-tutorials/qt-style-sheets?srsltid=AfmBOopetO_ZaThUyKxhN3vDaxrXEKVP_S3HrOyYNRHzeX68yU-2ehEF (дата звернення: 18.12.2024).

23. QWheelEvent Class. Doc Qt. URL: https://doc.qt.io/qt-6/qwheelevent.html (дата звернення: 17.12.2024).

24. Scaled background image using stylesheet. Qt Forum. URL: https://forum.qt.io/topic/40151/solved-scaled-background-image-using-stylesheet (дата звернення: 15.03.2025).

25. Shelly, Gary B.; Vermaat, Misty E. (2011). Discovering Computers 2011: Living in a Digital World, Complete. Cengage Learning. ISBN 978-1-4390-7926-3.

26. Shrestha U. Introduction to QT using C++. Medium. URL: https://medium.com/geekculture/visualize-sorting-algorithm-with-qt-introduction-to-qt-using-c-fe49c6df4125 (дата звернення: 17.12.2024).

27. Холод Ю.В. Програиний модуль для перегляду зображень. *Природничі та гуманітарні науки*: матеріали VIII Міжнародної студентської науково - технічної конференції. м. Тернопіль, 24-25 квітня 2025. С. 68-70

28. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

Додаток А

Лістинг програмного коду

А.1 – Лістинг файлу customgraphicsview.h

```
#ifndef CUSTOMGRAPHICSVIEW_H
#define CUSTOMGRAPHICSVIEW_H

#include <QGraphicsView>
#include <QWheelEvent>
#include <QPropertyAnimation>

class CustomGraphicsView : public QGraphicsView
{
    Q_OBJECT

private:
    qreal scaleFactor = 1.0;

public:
    explicit CustomGraphicsView(QWidget *parent = nullptr);

    qreal getScaleFactor() const { return scaleFactor; }

    void setScaleFactor(qreal scale);

    Q_PROPERTY(qreal scaleFactor READ getScaleFactor WRITE
setScaleFactor NOTIFY scaleFactorChanged)

signals:
    void scaleFactorChanged(qreal scale);

protected:
    void wheelEvent(QWheelEvent *event) override;
};

#endif
```

А.2 – Лістинг файлу customgraphicsview.cpp

```
#include "customgraphicsview.h"
#include <QDebug>
#include <QEasingCurve>
#include <QPropertyAnimation>

CustomGraphicsView::CustomGraphicsView(QWidget *parent)
    : QGraphicsView(parent)
{
}


void CustomGraphicsView::setScaleFactor(qreal scale) {
    if (scaleFactor != scale) {
        scaleFactor = scale;
        resetTransform();
        QGraphicsView::scale(scaleFactor, scaleFactor);

        emit scaleFactorChanged(scaleFactor);
    }
}

void CustomGraphicsView::wheelEvent(QWheelEvent *event)
{
    if (event->modifiers() == Qt::ControlModifier) {
        event->ignore();

        qreal angleDelta = event->angleDelta().y();
        qreal factor = 1.0 + (angleDelta / 120.0) * 0.1;

        qreal currentScale = transform().m11();

        qreal newScale = currentScale * factor;

        QPropertyAnimation *animation = new QPropertyAnimation(this,
"scaleFactor");
        animation->setDuration(50);
        animation->setStartValue(currentScale);
        animation->setEndValue(newScale);
        animation->setEasingCurve(QEasingCurve::InOutQuad);
        animation->start();

        event->accept();
    } else {
        QGraphicsView::wheelEvent(event);
    }
}
```

А.3 – Лістинг файлу mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QGraphicsScene>
#include "customgraphicsview.h"

QT_BEGIN_NAMESPACE
namespace Ui {
class MainWindow;
}
QT_END_NAMESPACE

class CustomGraphicsView; // Оголошення перевизначеного класу

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

protected:
    //void wheelEvent(QWheelEvent *event) override; // Обробка
прокрутки миші
    void resizeEvent(QResizeEvent *event) override; // Обробка зміни
розміру вікна

private:
    Ui::MainWindow *ui;
    QGraphicsScene *scene; // Сцена для QGraphicsView
    QString currentFolder; // Поточна папка
    QStringList imageFiles; // Список зображень у папці
    int currentIndex;       // Індекс поточного зображення
    double scaleFactor;     // Масштаб зображення

    CustomGraphicsView *customView;

    void fitImageToView(); // Підганяє зображення під розмір вікна
    void loadImagesFromFolder(const QString &folderPath); //
Завантажити всі зображення з папки
    void updateNavigationButtons(); // Оновити кнопки "Вперед" і
"Назад"
    void setScrollBarsVisible(bool visible);
    void updateImageInfo(const QString &filePath);
```

```
private slots:
    void loadImage(); // Завантажити одне зображення
    void on_pushButtonBack_clicked();
    void on_pushButtonForward_clicked();
    void on_pushButtonRotateLeft_clicked();
    void on_pushButtonRotateRight_clicked();
    void on_comboBox_currentIndexChanged(int index);
};

#endif // MAINWINDOW_H
```

А.4 – Лістинг файлу main.cpp

```
#include "mainwindow.h"

#include <QApplication>
#include <QLocale>
#include <QTranslator>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    QTranslator translator;
    const QStringList uiLanguages = QLocale::system().uiLanguages();
    for (const QString &locale : uiLanguages) {
        const QString baseName = "PhotoViewer_" +
QLocale(locale).name();
        if (translator.load(":/i18n/" + baseName)) {
            a.installTranslator(&translator);
            break;
        }
    }
    MainWindow w;
    w.show();
    return a.exec();
```

А.5 – Лістинг файлу mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>839</width>
    <height>669</height>
   </rect>
  </property>
  <property name="sizePolicy">
   <sizepolicy hsizetype="Minimum" vsizetype="Fixed">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <property name="styleSheet">
   <string notr="true">QMainWindow {
    background:  rgb(32, 32, 32);
}</string>
  </property>
  <property name="toolButtonStyle">
   <enum>Qt::ToolButtonStyle::ToolButtonIconOnly</enum>
  </property>
  <widget class="QWidget" name="centralwidget">
   <property name="styleSheet">
    <string notr="true"/>
   </property>
   <layout class="QVBoxLayout" name="verticalLayout_2"
stretch="1,15,1">
    <property name="spacing">
     <number>6</number>
    </property>
    <property name="sizeConstraint">
     <enum>QLayout::SizeConstraint::SetNoConstraint</enum>
    </property>
    <property name="leftMargin">
     <number>0</number>
    </property>
    <property name="topMargin">
     <number>0</number>
    </property>
    <property name="rightMargin">
```

```
     <number>0</number>
    </property>
    <property name="bottomMargin">
     <number>0</number>
    </property>
    <item>
     <widget class="QWidget" name="widget" native="true">
      <layout class="QVBoxLayout" name="verticalLayout">
       <property name="leftMargin">
        <number>5</number>
       </property>
       <property name="topMargin">
        <number>5</number>
       </property>
       <property name="rightMargin">
        <number>0</number>
       </property>
       <property name="bottomMargin">
        <number>0</number>
       </property>
       <item>
        <widget class="QComboBox" name="comboBox">
         <property name="sizePolicy">
          <sizepolicy hsizetype="MinimumExpanding" vsizetype="Fixed">
           <horstretch>0</horstretch>
           <verstretch>0</verstretch>
          </sizepolicy>
         </property>
         <property name="maximumSize">
          <size>
           <width>100</width>
           <height>16777215</height>
          </size>
         </property>
         <property name="styleSheet">
          <string notr="true">QComboBox {
  color: white;
    border: 1px solid grey;
    border-radius: 3px;
    padding: 1px 18px 1px 3px;
    min-width: 6em;
}
QComboBox:editable {
    background: white;
}
QComboBox:!editable, QComboBox::drop-down:editable {
     background: #4399DB;
}
QComboBox:!editable:on, QComboBox::drop-down:editable:on {
    background: #4399DB;
}
```

```
QComboBox:on {
    padding-top: 3px;
    padding-left: 4px;
}
QComboBox::drop-down {
    subcontrol-origin: padding;
    subcontrol-position: top right;
    width: 15px;
    border-left-width: 1px;
    border-left-color: darkgray;
    border-left-style: solid;
    border-top-right-radius: 3px;
    border-bottom-right-radius: 3px;
}
QComboBox QAbstractItemView {
  color: rgb(85, 170, 255);
  background-color: #373e4e;
  padding: 10px;
  selection-background-color: rgb(39, 44, 54);
}</string>
         </property>
         <item>
          <property name="text">
           <string>Відкрити...</string>
          </property>
         </item>
         <item>
          <property name="text">
           <string>Файл</string>
          </property>
         </item>
         <item>
          <property name="text">
           <string>Папку</string>
          </property>
         </item>
        </widget>
       </item>
       <item>
        <widget class="QLabel" name="labelTop">
         <property name="sizePolicy">
          <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
           <horstretch>0</horstretch>
           <verstretch>0</verstretch>
          </sizepolicy>
         </property>
         <property name="font">
          <font>
           <bold>true</bold>
          </font>
         </property>
```

```
         <property name="layoutDirection">
          <enum>Qt::LayoutDirection::LeftToRight</enum>
         </property>
         <property name="styleSheet">
          <string notr="true">QLabel#labelTop {
    qproperty-alignment: 'AlignHCenter | AlignVCenter'; /* Центрування
тексту */
    color: rgb(255, 255, 255); /* Білий текст */
}
</string>
         </property>
         <property name="text">
          <string>Файл не відкрито</string>
         </property>
        </widget>
       </item>
      </layout>
     </widget>
    </item>
    <item>
     <widget class="QWidget" name="widget_2" native="true">
      <layout class="QHBoxLayout" name="horizontalLayout_2">
       <property name="bottomMargin">
        <number>0</number>
       </property>
       <item>
        <widget class="QPushButton" name="pushButtonBack">
         <property name="maximumSize">
          <size>
           <width>30</width>
           <height>45</height>
          </size>
         </property>
         <property name="styleSheet">
          <string notr="true">QPushButton#pushButtonBack {
   background-image: url(:/images/res/buttonLeftNotActive.png);
   border-radius: 7px;
}

QPushButton#pushButtonBack:hover {
    background-image: url(:/images/res/buttonLeft.png);
}

QPushButton#pushButtonBack:pressed {
    border-width: 1px;
    border-style: solid;
    border-color: white;
}</string>
         </property>
         <property name="text">
          <string/>
```

```
         </property>
         <property name="iconSize">
          <size>
           <width>16</width>
           <height>16</height>
          </size>
         </property>
        </widget>
       </item>
       <item>
        <widget class="CustomGraphicsView" name="graphicsView">
         <property name="styleSheet">
          <string notr="true">QWidget {
    background-color: rgb(39, 39, 39);
}</string>
         </property>
        </widget>
       </item>
       <item>
        <widget class="QPushButton" name="pushButtonForward">
         <property name="maximumSize">
          <size>
           <width>30</width>
           <height>45</height>
          </size>
         </property>
         <property name="styleSheet">
          <string notr="true">QPushButton#pushButtonForward {
   background-image: url(:/images/res/buttonRightNotActive.png);
   border-radius: 7px;
}

QPushButton#pushButtonForward:hover {
    background-image: url(:/images/res/buttonRight.png);
}

QPushButton#pushButtonForward:pressed {
    border-width: 1px;
    border-style: solid;
    border-color: white;
}</string>
         </property>
         <property name="text">
          <string/>
         </property>
        </widget>
       </item>
      </layout>
     </widget>
    </item>
    <item>
```

```
     <widget class="QWidget" name="widget_3" native="true">
      <property name="styleSheet">
       <string notr="true">QWidget {
    background-color: rgb(39, 39, 39);
}</string>
      </property>
      <layout class="QHBoxLayout" name="horizontalLayout_3">
       <property name="topMargin">
        <number>0</number>
       </property>
       <property name="bottomMargin">
        <number>9</number>
       </property>
       <item>
        <widget class="QLabel" name="labelResolution">
         <property name="maximumSize">
          <size>
           <width>200</width>
           <height>26</height>
          </size>
         </property>
         <property name="styleSheet">
          <string notr="true">QLabel#labelResolution {
    color: rgb(255, 255, 255);
}
</string>
         </property>
         <property name="text">
          <string/>
         </property>
         <property name="pixmap">
          <pixmap
resource="resources.qrc">:/images/res/iconResolution.png</pixmap>
         </property>
         <property name="alignment">

<set>Qt::AlignmentFlag::AlignRight|Qt::AlignmentFlag::AlignTrailing|Qt:
:AlignmentFlag::AlignVCenter</set>
         </property>
        </widget>
       </item>
       <item>
        <widget class="QLabel" name="labelResolutionText">
         <property name="layoutDirection">
          <enum>Qt::LayoutDirection::LeftToRight</enum>
         </property>
         <property name="styleSheet">
          <string notr="true">QLabel#labelResolutionText {
    color: rgb(255, 255, 255);
}
</string>
```

```
          </property>
          <property name="text">
           <string/>
          </property>
          <property name="alignment">

<set>Qt::AlignmentFlag::AlignLeading|Qt::AlignmentFlag::AlignLeft|Qt::A
lignmentFlag::AlignVCenter</set>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton" name="pushButtonRotateLeft">
          <property name="maximumSize">
           <size>
            <width>35</width>
            <height>25</height>
           </size>
          </property>
          <property name="styleSheet">
           <string notr="true">QPushButton#pushButtonRotateLeft {
   background-image: url(:/images/res/turnLeft.png);
   border-radius: 3px;
}

QPushButton#pushButtonRotateLeft:pressed {
    border-width: 1px;
    border-style: solid;
    border-color: white;
}</string>
          </property>
          <property name="text">
           <string/>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton" name="pushButtonRotateRight">
          <property name="sizePolicy">
           <sizepolicy hsizetype="Minimum" vsizetype="Fixed">
            <horstretch>0</horstretch>
            <verstretch>0</verstretch>
           </sizepolicy>
          </property>
          <property name="maximumSize">
           <size>
            <width>35</width>
            <height>25</height>
           </size>
          </property>
          <property name="styleSheet">
```

```
          <string notr="true">QPushButton#pushButtonRotateRight {
   background-image: url(:/images/res/turnRight.png);
   border-radius: 3px;
}

QPushButton#pushButtonRotateRight:pressed {
    border-width: 1px;
    border-style: solid;
    border-color: white;
}</string>
         </property>
         <property name="text">
          <string/>
         </property>
        </widget>
       </item>
       <item>
        <widget class="QLabel" name="labelSizeText">
         <property name="styleSheet">
          <string notr="true">QLabel#labelSizeText {
    color: rgb(255, 255, 255);
}
</string>
         </property>
         <property name="text">
          <string/>
         </property>
         <property name="alignment">

<set>Qt::AlignmentFlag::AlignRight|Qt::AlignmentFlag::AlignTrailing|Qt:
:AlignmentFlag::AlignVCenter</set>
         </property>
        </widget>
       </item>
       <item>
        <widget class="QLabel" name="labelSize">
         <property name="maximumSize">
          <size>
           <width>200</width>
           <height>26</height>
          </size>
         </property>
         <property name="styleSheet">
          <string notr="true">QLabel#labelSize {
    color: rgb(255, 255, 255);
}
</string>
         </property>
         <property name="text">
          <string/>
         </property>
```

```
         <property name="pixmap">
          <pixmap
resource="resources.qrc">:/images/res/iconSave.png</pixmap>
         </property>
         <property name="alignment">

<set>Qt::AlignmentFlag::AlignLeading|Qt::AlignmentFlag::AlignLeft|Qt::A
lignmentFlag::AlignVCenter</set>
         </property>
        </widget>
       </item>
      </layout>
     </widget>
    </item>
   </layout>
  </widget>
  <widget class="QMenuBar" name="menubar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>839</width>
     <height>22</height>
    </rect>
   </property>
  </widget>
  <widget class="QStatusBar" name="statusbar"/>
 </widget>
 <customwidgets>
  <customwidget>
   <class>CustomGraphicsView</class>
   <extends>QGraphicsView</extends>
   <header location="global">customgraphicsview.h</header>
  </customwidget>
 </customwidgets>
 <resources>
  <include location="resources.qrc"/>
 </resources>
 <connections/>
</ui>
```

Додаток Б

Публікація за темою кваліфікаційної роботи

Міністерство освіти і науки України
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка,
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет,
Чернівецький національний університет
імені Юрія Федьковича,
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

VIII МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ.

АКТУАЛЬНІ ПИТАННЯ"

24-25 квітня 2025 р.

(збірник тез конференції)

Тернопіль 2025

ББК 72+34 (Укр)
М34

Матеріали VIII Міжнародної студентської науково - технічної конференції / Тернопіль: Тернопільський національний технічний університет ім. І.Пулюя (м. Тернопіль, 24-25 квітня 2025 р.), 2025.- 391 с.

В збірнику друкуються матеріали VIII Міжнародної студентської науково-технічної конференції. Тернопіль. – ТНТУ ім. І. Пулюя (24-25 квітня 2025 р.) за наступними науковими спеціальностями:
культура і мистецтво; гуманітарні науки; соціальні та поведінкові науки; управління та адміністрування; природничі науки; математика та статистика; інформаційні технології; механічна інженерія; електрична інженерія; автоматизація та приладобудування; хімічна та біоінженерія; електроніка та телекомунікації; виробництво та технології; архітектура та будівництво; аграрні науки та продовольство; сфера обслуговування; транспорт.

Комп'ютерний набір, верстка та редагування:
науковий секретар Ігор Окіпний

Адреса конференції:
46001, м. Тернопіль, вул. Руська, 56
Тернопільський національний технічний університет ім. Івана Пулюя
e-mail: snt@tntu.edu.ua
Тернопільський національний технічний університет ім. Івана Пулюя

УДК 621.3.06
Холод Ю. – ст. гр. КН-43
Західноукраїнський національний університет

ПРОГРАМНИЙ МОДУЛЬ ДЛЯ ПЕРЕГЛЯДУ ЗОБРАЖЕНЬ

Науковий керівник: к.т.н. Майків І.М.

Kholod Y.
West Ukrainian National University

SOFTWARE MODULE FOR IMAGE VIEWING

Supervisor: Ph.D. Maikiv I.M.

Ключові слова: дизайн, функціонал, масштабування, інтерфейс, файл
Keywords: design, functionality file, scaling, interface.

У цифрову епоху робота з графічними файлами стала звичним елементом як професійної діяльності (фотографія, дизайн, дослідження), так і повсякденного користування. У цьому контексті особливе значення має наявність ефективного інструмента для перегляду зображень. Попри те, що на ринку вже існує чимало програм для цієї мети, багато з них мають обмежену функціональність, складний інтерфейс або не підтримують потрібні формати. Особливої уваги заслуговує проблема інтерактивної взаємодії користувача із програмою, зокрема зручність масштабування, перемикання між зображеннями, підтримка галерей та широкого спектра форматів. Це створює передумови для розробки нового, простого у використанні, але водночас функціонально повного програмного забезпечення.

Розробка програмного модуля передбачала побудову структурної моделі, зокрема DFD та IDEF0-діаграм, що деталізують процеси завантаження, обробки та форматування зображень [1]. Окремо створено ERD-діаграму (див. рис. 1) [2] для візуалізації взаємодії між основними сутностями системи: вікном програми, зображенням, папкою, файловою системою. Програмна реалізація виконана з використанням класів Qt, серед яких головну роль відіграє MainWindow, що координує роботу з інтерфейсом, та CustomGraphicsView, який дозволяє масштабувати зображення прокручуванням миші. Також реалізовано обробку подій, пов'язаних із завантаженням фото, навігацією, обертанням зображень, оновленням метаінформації (розмір, роздільна здатність).Інтерфейс програмного модуля було створено з використанням інструменту Qt Designer, що дозволило реалізувати логічну, інтуїтивно зрозумілу структуру вікна користувача. Основний екран додатку складається з комбінованого списку для вибору режиму завантаження (фото чи папка), області перегляду зображення, а також панелі з кнопками навігації (вперед/назад) та обертання. Всі елементи інтерфейсу розміщено за принципом вертикальної та горизонтальної компоновки з урахуванням сучасних вимог до зручності користування. Для покращення візуального сприйняття програми були застосовані стилі QSS, які визначають кольори фону, шрифти, межі, тіні та поведінку елементів у різних станах - наприклад, активному чи неактивному. Це надало програмі стильного, сучасного вигляду з темною кольоровою палітрою, комфортною для зору.

Рисунок 1 - ERD діаграма модуля перегляду зображень

Після завершення етапу реалізації основного функціоналу — завантаження зображень, їх масштабування, перегляду та обертання — було проведено повноцінне тестування додатку. Тестуванню підлягали як стандартні сценарії (відкриття одного файлу або всієї директорії, перемикання між зображеннями), так і виняткові ситуації: наприклад, вибір папки без зображень або введення неправильного шляху. Усі такі ситуації обробляються відповідними повідомленнями, що дозволяє користувачу розуміти причину проблеми та уникати збоїв у роботі програми.

Окрім цього, в модулі реалізовано автоматичне визначення індексу зображення, яке користувач обирає з папки — завдяки чому програма не просто відкриває каталог із зображеннями, а й відразу відображає потрібний файл, дозволяючи продовжити перегляд у зручному порядку. Програмний модуль підтримує інтерактивне масштабування, що супроводжується анімацією плавного збільшення або зменшення масштабу, створюючи відчуття «живої» взаємодії. Обертання зображення реалізовано як вліво, так і вправо (на 90 градусів), і після кожної операції зображення автоматично підлаштовується під розмір вікна.

Таким чином, програмний модуль не лише виконує поставлені задачі, а й забезпечує користувачеві комфортну, стабільну та візуально привабливу взаємодію, що потрібно для роботи із зображеннями у повсякденному чи професійному контексті. Практична цінність модуля полягає в його універсальності та простоті, що дає змогу застосовувати його як у побутових умовах, так і в навчальних або професійних задачах, пов'язаних із роботою з графічними файлами. Проект має потенціал для подальшого розширення функціональності, наприклад, додавання фільтрів, редагування метаданих або інтеграції з хмарними сховищами [3].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методологія IDEF0. [Електронний ресурс]
URL: https://stud.com.ua/87184/ekonomika/metodologiya_idef0
2. Браун Ф. Модель діаграми зв'язків сутностей. Guru99. [Електронний ресурс]:
URL: https://www.guru99.com/uk/er-diagram-tutorial-dbms.html
3. Дідковська М. Проектування програмного забезпечення засобами UML: Навчальний посібник. Харьків, 2020. №2, Том 1, 2007, ст.117-122