Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Пояснювальна записка

до магістерської кваліфікаційної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему: Розробка методів і засобів колористичного оброблення зображень та системи на їх основі

Виконав: студент II курсу
групи 1ПІ-18 м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Ковбасюк О. В.
(прізвище та ініціали)

Керівник: к. т. н., доц. Коваленко О. О.
(прізвище та ініціали)

Рецензент: к. т. н., проф. Савчук Т. О.
(прізвище та ініціали)

ВНТУ – 2019

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Освітньо-кваліфікаційний рівень – магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
"____"_____2019 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Ковбасюку Олександру Вікторовичу

1. Тема роботи: «Розробка методів і засобів колористичного оброблення зображень та системи на їх основі», керівник роботи: Коваленко Олена Олексіївна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від "__"_____2019 року №___

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи: зображення в просторі Truecolor, розмір екрану – 1280x1024, вихідний метод для модифікації – кластеризація методом к-середніх.

4. Зміст розрахунково-пояснювальної записки: вступ, аналіз предметної області, розробка модифікації методу кластеризації к-середніх для підвищення ефективності визначення домінуючої палітри кольорів зображення, розробка програмних засобів реалізації розробленого методу визначення домінуючої палітри кольорів зображення, тестування системи, економічна частина, висновки, список використаної джерел, додатки.

5. Перелік графічного матеріалу: актуальність досліджуваної теми; метод кластеризації к-середніх для визначення домінантної палітри кольорів; модель врахування людського сприйняття кольорів; візуалізації кластеризації зображення; загальна структурна системи; інтерфейс розробленого програмного додатку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Коваленко О. О., к.т.н., доцент кафедри ПЗ		
5	Бальзан М. В., к.е.н., доцент кафедри ЕПВМ		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	04.09.2019 – 14.09.2019	Вик.
2	Розробка модифікації методу кластеризації к-середніх для підвищення ефективності визначення домінуючої палітри кольорів зображення	15.09.2019 – 20.10.2019	Вик.
3	Розробка програмних засобів реалізації розробленого методу визначення домінуючої палітри кольорів зображення	21.10.2019 – 15.11.2019	Вик.
4	Тестування системи	16.11.2019 – 21.11.2019	Вик.
5	Економічна частина	22.11.2019 – 01.12.2019	Вик.

Студент _____ **Ковбасюк О.В.**

(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Коваленко О.О.**

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У магістерській кваліфікаційній роботі «Розробка методів і засобів колористичного оброблення зображень та системи на їх основі» розроблено модифікацію методу кластеризації к-середніх, призначену для підвищення швидкодії та точності аналізу зображень при визначенні домінантної палітри кольорів, а також програмну систему, у якій впроваджено розроблену модифікацію, і яка може бути інтегрована до вже існуючих програмних систем.

В ході виконання роботи було проаналізовано існуючі методи визначення домінантної палітри кольорів та існуючі аналоги. Обґрунтовано вибір засобів для розробки програмного забезпечення: мова програмування C#, фреймворки Entity Framework, Emgu CV, Windows Presentation Foundation. Розроблено архітектуру системи, програмні модулі, протестовано коректність роботи системи.

ABSTRACT

In master's qualification thesis on "Development of methods and software for colour image processing and system based on them" developed a modification of the k-means clustering method, designed to improve the speed and accuracy of image processing in determining the dominant colour palette, as well as a software system that incorporates the developed modification and which can be integrated with existing software systems.

During the execution, existing methods of determining the dominant colour palette and existing analogues were analysed. The following software development tools are chosen: C# programming language, Entity Framework, Emgu CV, Windows Presentation Foundation. The system architecture and software modules were developed; the correctness of the system operation was tested.

ЗМІСТ

ВСТУП

Актуальність роботи. В сучасному світі, розвиток технологій відкрив новий простір для застосування сучасних досягнень в науці та техніці у вирішенні проблем, які раніше було неможливо або було вкрай важко вирішити. Одне з досягнень є можливість здійснення колористичного оброблення зображень з визначенням домінуючої палітри кольорів [1]. Така можливість, в сучасному світі, що стрімко розвивається, є доволі затребуваною.

Було проведено безліч досліджень, які показали важливість кольору в житті людини [2]. Так, наприклад, згідно досліджень CCICOLOR – Institute for Color Research, люди роблять підсвідому оцінку предмету, навколишнього середовища, іншої людини впродовж 90 секунд перегляду, і що від 62% до 90% їхнього оціночного судження базується саме на кольорі. В результаті дослідження проведеного університетом Лойоли, штат Меріленд – колір підвищує впізнаваність бренду до 80%. Згідно з опублікованою статтею в журналі Experimental Psychology: Learning, Memory and Cognition видавництва Американської психологічної асоціації, колір стимулює мозкову діяльність людини, чим допомагає аналізувати та запам’ятовувати зображення більш ефективно, ніж якщо б вони були безкольорові (чорно-біле зображення).

Отже, у зв’язку з великим потенціалом та поширеним використанням колористичного аналізу є актуальною проблема створення системи колористичного оброблення зображень для отримання домінантної палітри кольорів.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалась згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення продуктивності та точності пошуку домінантних кольорів при аналізі зображень за рахунок розробки нових та модифікації існуючих методів та засобів колористичного аналізу.

Основними задачами дослідження є:

- аналіз існуючих методів колористичного аналізу зображень задля пошуку домінуючої палітри кольорів;
- розробка вдосконаленого методу, що буде характеризуватись підвищеним рівнем швидкодії та точністю пошуку домінуючої палітри кольорів;
- розробка програмної системи, на основі якої буде можливе впровадження розробленого методу колористичного оброблення зображень.

Об’єкт дослідження – процес розробки та впровадження вдосконаленого методу пошуку домінуючої палітри кольорів.

Предмет дослідження – методи та засоби колористичного оброблення зображень.

Методи дослідження. У процесі досліджень використовувались: метод кластеризації к-середніх, методи математичного аналізу та методи статистичного дослідження для підвищення продуктивності та точності пошуку домінантних кольорів при аналізі зображень; методи проектування баз даних, метод моделювання архітектури програмного забезпечення та методи розробки графічних інтерфейсів клієнтських додатків для розробки засобі колористичного оброблення зображень.

Наукова новизна отриманих результатів.

- Подальшого розвитку набув метод аналізу зображення, який базується на методі кластеризації, який на відміну від існуючих, визначає залежності між якістю аналізу та розмірами зображення шляхом формування відповідності розмірів зображення та показників втрати точності аналізу для пошуку домінантних кольорів зображення, що дозволяє збільшити продуктивність та покращити точність пошуку домінантних кольорів.
- Подальшого розвитку отримав метод аналізу зображень, який в порівнянні з існуючими, враховує параметри людського сприйняття кольорів на основі їх фільтрації за яскравістю, а також містить додаткову опцію в налаштуванні аналізу безпосередньо користувачем, що дозволяє розширити

функції налаштування, а саме здійснювати вибір виду аналізу – прямий або з урахуванням людського сприйняття.

- Запропоновано нові функції процесу аналізу набору зображень, які, на відміну від існуючих, дозволяють реалізувати виконання паралельних процесів розпізнавання та візуалізації кластеризації зображення, що дає можливість підвищити швидкодію методу та визначення належності окремого пікселя до визначеного кластеру.

Практична цінність отриманих результатів полягає в тому, що на основі проведених теоретичних досліджень й отриманих наукових результатів розроблено комплекс програмних засобів для колористичного оброблення зображень з визначенням домінуючої палітри кольорів, виконана програмна реалізація окремих модулів, що можуть бути застосовані на підприємствах з розробки дизайну приміщень та сайтів, обробки зображень та картин, автоматизованого виробництва та сортування окремих видів продукції.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати:

- проведення дослідження існуючих методів колористичного оброблення зображень [3];
- проведення порівняльної характеристики існуючих аналогів колористичного оброблення зображень задля пошуку домінуючої палітри кольорів [4].

Апробація матеріалів. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на XII міжнародній науково-практичній конференції «Інформаційні технології і автоматизація –2019» (Одеса, 2019).

Результати дослідження подані до колективної монографії «Інформаційні технології та автоматизація» ОНАПТ, Одеса, 2019 р.

Публікації. Основні результати досліджень опубліковано в 2 наукових працях, у тому числі 1 – в матеріалах конференції й 1 – в монографічному вигляді.

Структура та обсяг роботи. Магістерська кваліфікаційна робота складається зі вступу, п’ятьох розділів, висновків, списку використаних джерел, що містить 22 найменування, 3 додатків. Робота містить 39 ілюстрацій, 8 таблиць.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ КОЛОРИСТИЧНОГО ОБРОБЛЕННЯ ЗОБРАЖЕНЬ

1.1 Аналіз існуючих методів визначення домінуючої палітри кольорів

Існує декілька методів обробки зображень для визначення домінуючої палітри кольорів.

Одним з таких способів є використання колірної моделі HSV (Hue Saturation, Value) для числового представлення кольору, обчисленні цього значення для кожного окремо взятого пікселя та обрахунку отриманого значення на предмет належності до завчасно визначеного переліку кольорів з усередненням отриманих значень в якості кінцевого кроку.

Вказана колірна модель є перетвореною моделлю RGB (рисунок 1.1).

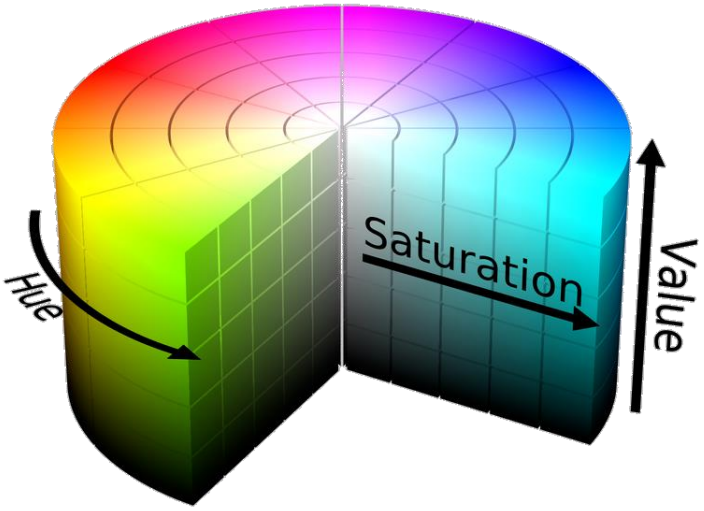

Рисунок 1.1 – Циліндричне представлення моделі HSV

Ця колірна модель в якості властивостей для вираження кольору має:

- Hue – колірний тон, що варіюється в межах 0-360°, але в окремих випадках діапазон є в межах 0-100 або 0-1;
- Saturation – насиченість, що варіюється в межах 0-100 або 0-1, чим більше цей параметр тим «чистіше» виглядає колір, а при наближені до 0 колір набуває сіруватого відтінку;

- Value (Brightness) – значення кольору або його яскравість. Варіюється в межах 0-100 або 0-1. При наближення значення параметру до 0, колір набуває темного забарвлення, а до 1 – білого.

Для обрахунку належності певного кольору до завчасно визначеної множини кольорів використовується шкала відтінків колірного тону (рисунок 1.2) з врахуванням значень насиченості та яскравості.

Рисунок 1.2 – Шкала відтінків колірного тону

Останнім кроком є усередненням отриманого підрахунку кольорів, що належать до визначеного кольору з початкової множини.

Для здійснення вказаного кроку, отримані кольори зручно перевести в колірну модель RGB.

Варто зауважити, що при перетворені з колірної схеми HSV в RGB варто застосовувати саме конічне представлення моделі HSV (рисунок 1.3).

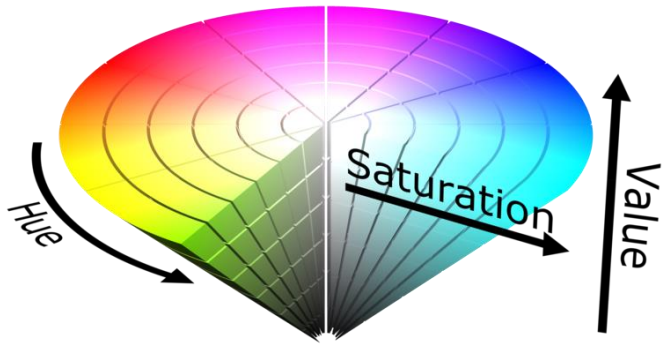

Рисунок 1.3 – Конічне представлення моделі HSV

Це зумовлено ти, що при використанні циліндричної моделі представлення, з’являються суттєві помилки округлення при перетворенні параметрів Saturation (насиченість) та Value (яскравість), якщо вони мають малі значення.

Після здійсненого перетворення виконується усереднення шляхом підрахунку суми отриманих окремих значень компонентів RGB поділеної на кількість належних до вказаного кольору пікселів:

$$C_{RGB} = \frac{1}{\sum_{p \in C} p} \sum_{p \in C_{RGB}} p_{RGB}$$

де

C_{RGB} – значення відповідного простору кінцевого кольору в форматі RGB;

p_{RGB} – значення відповідного простору пікселя в форматі RGB;

C – кінцевий колір;

p – окремо взятий піксель зображення.

Після здійснення обрахунків кожного з просторів формату RGB, ми отримуємо домінантний колір зображення.

Перевагою використання є одиничне проходження по окремо взятим пікселям зображення без необхідності здійснення повторних обрахунків.

Недоліками є необхідність визначити початковий набір кольорів, до яких буде здійснюватися обрахунок належності пікселів зображення від чого результат обробки може змінюватись відповідно до вказаного вхідного набору, також, в окремих випадках отриманий результат не є достатньо точним.

Іншим методом обробки зображень є використання кластеризації методом к-середніх.

Стандартний алгоритм був вперше запропонований Стюардом Лойдом у 1957 році, хоча «к-середніх» був уперше вжитий Джеймсом МакКвіном у 1967 році [5].

Вказаний метод є популярним методом кластеризації, задача якої є розбиття заданої вибірки об’єктів (ситуацій) на підмножини, які називаються кластерами, так, щоб кожен кластер складався з схожих об’єктів, а об’єкти різних кластерів істотно відрізнялися. Спектр застосувань кластерного аналізу дуже широкий: його використовують в археології, антропології, медицині, психології, хімії, біології, державному управлінні, філології тощо [6].

Мета методу – розділити n спостережень на k кластерів, так щоб кожне спостереження належало до кластера з найближчим до нього середнім значенням. Метод базується на мінімізації суми квадратів відстаней між кожним спостереженням та центром його кластера, тобто функції:

$$\sum_{i=1}^{N} d(x_i, m_j(x_i))^2$$

де

d – метрика;

x_i – і-ий об’єкт даних;

$m_i(x_i)$ – центр кластера, якому на j-ій ітерації приписаний елемент (x_i).

Алгоритм кластеризації методом к-середніх має наступний вигляд:

1. Визначення дослідником кількості кластерів, які необхідно створити;

2. Випадковим чином обирається k кількість спостережень, які вважаються центрами кластерів;

3. Кожне спостереження приписується до н-го кластера, відстань до центру якого є найменшою;

4. Визначається новий центр кластера шляхом підрахунку середнє арифметичного ознак об’єктів, що входять в цей кластер;

5. Перевірка чи центр кластера є стійким (тобто при кожній ітерації в кожному кластері опинятимуться одні й ті самі об’єкти), якщо трапляється

зворотне – відбувається така кількість ітерацій кроку 3 та 4 допоки центри кластерів не стануть стійкими.

Рисунок 1.4 – Приклад роботи алгоритму з 3-ма кластерами

На рисунку 1.4 зображено принцип роботи алгоритму відповідно до вказаних кроків.

Принцип алгоритму полягає в пошуку центрів кластерів та елементів, що належать цим кластерам, при наявності деякої функції, що виражає якість поточного розбиття множини на k кластерів, при найменшому сумарному квадратичному відхилені елементів кластерів від центрів цих кластерів:

$$V = \sum_{i=1}^{k} \sum_{x_j \epsilon S_i} (x_j - \mu_i)^2$$

де

k – число кластерів;

S_i – отримані кластери;

i – знаходиться в межах від 1 до k;

μ_i – центри мас векторів $x_j \epsilon S_i$.

Перевагами використання такого алгоритму є його швидкість. Також, метод к-середніх є зручнішим для кластеризації великої кількості спостережень (інформаційних точок), ніж метод ієрархічного кластерного аналізу, в якому дендрограми (графи без циклів побудовані за допомогою матриці заходів близькості) стають перевантаженими і втрачають наочність.

Недоліками такого методу є те, що кількість кластерів повинна бути заздалегідь відома; не гарантується досягнення глобального мінімуму сумарного квадратного відхилення V, а лише його один з його локальних мінімумів; результат залежить від вибору початкових центрів кластерів.

Отже, після проведеного дослідження існуючих методів, виявлено перелік їхніх основних недоліків, які виражаються в недостатній точності аналізу, недостатній швидкості обробки зображень, потребі в завчасно визначених вхідних даних. Саме тому є актуальною потреба в розробці власного або модифікації вже існуючого методу обробки зображень для визначення домінантної палітри кольорів.

1.2 Порівняльний аналіз аналогів

Розглянемо одні з найбільш популярних сервісів, що надають можливість визначення домінантної палітри кольорів зображення.

В якості першого існуючого аналога взято онлайн сервіс IMGonline [7] (рисунок 1.5).

Вказаний сервіс здійснює аналіз вхідного зображення через його завантаження на свою серверну частину, при цьому присутня можливість налаштування аналізу зображення у вигляді вибору вихідної кількості домінуючих кольорів та стисненні вихідного зображення до формату JPEG.

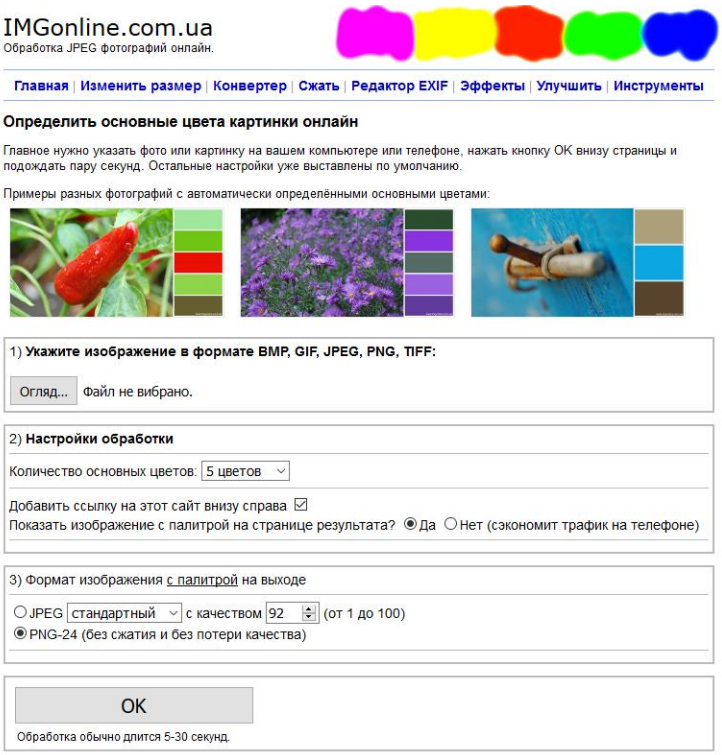

Рисунок 1.5 – Онлайн сервіс «IMGonline»

Перевагами використання цього сервісу є його онлайн платформа та можливість налаштування вихідної кількості домінуючих кольорів, швидкодія при аналізі невеликих за розмірами зображень.

Результат роботи цього сервісу зображено на рисунку 1.6 та 1.7

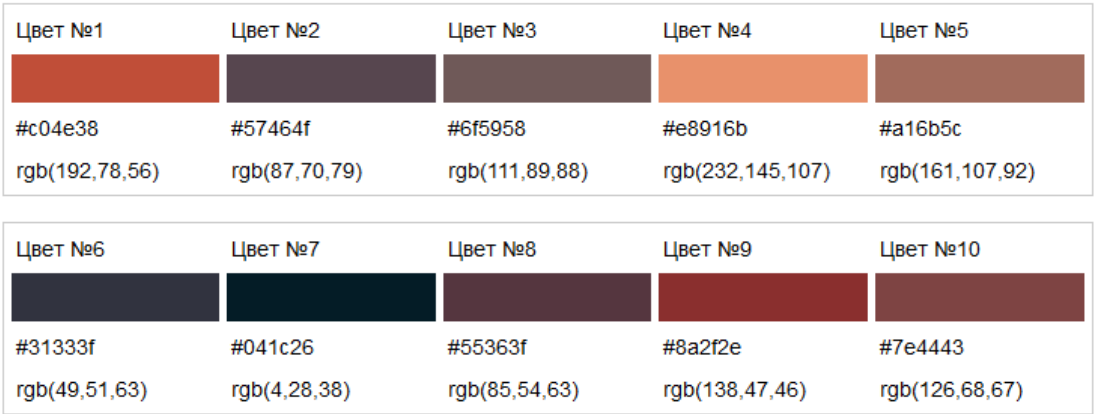

Рисунок 1.6 – Перелік домінантних кольорів у вікні результату

Рисунок 1.7 – Результат роботи сервісу «IMGonline»

Недоліками цього сервісу є: неможливість здійснення аналізу групи зображень (кожне зображення необхідно завантажувати окремо), недостатньо точний результат при обробленні зображень з яскравими кольорами, необхідність завантаження зображення на сервер третьої сторони для його оброблення, неможливість створення вибірки з раніше проаналізованих зображень, втрата швидкодії в обробленні зображення при обробці об’ємних зображень та через використання клієнт-серверної архітектури.

Наступним аналогом є онлайн сервіс «TinEye» [8].

Перевагами цього сервісу є простота та зручність інтерфейсу, можливість виключення фонового кольору з отриманої палітри кольорів, швидка обробка відносно невеликих зображень (рисунок 1.8).

Недоліки цього сервісу є аналогічними до недоліків попереднього аналога, також відсутня можливість налаштування процесу аналізу зображення.

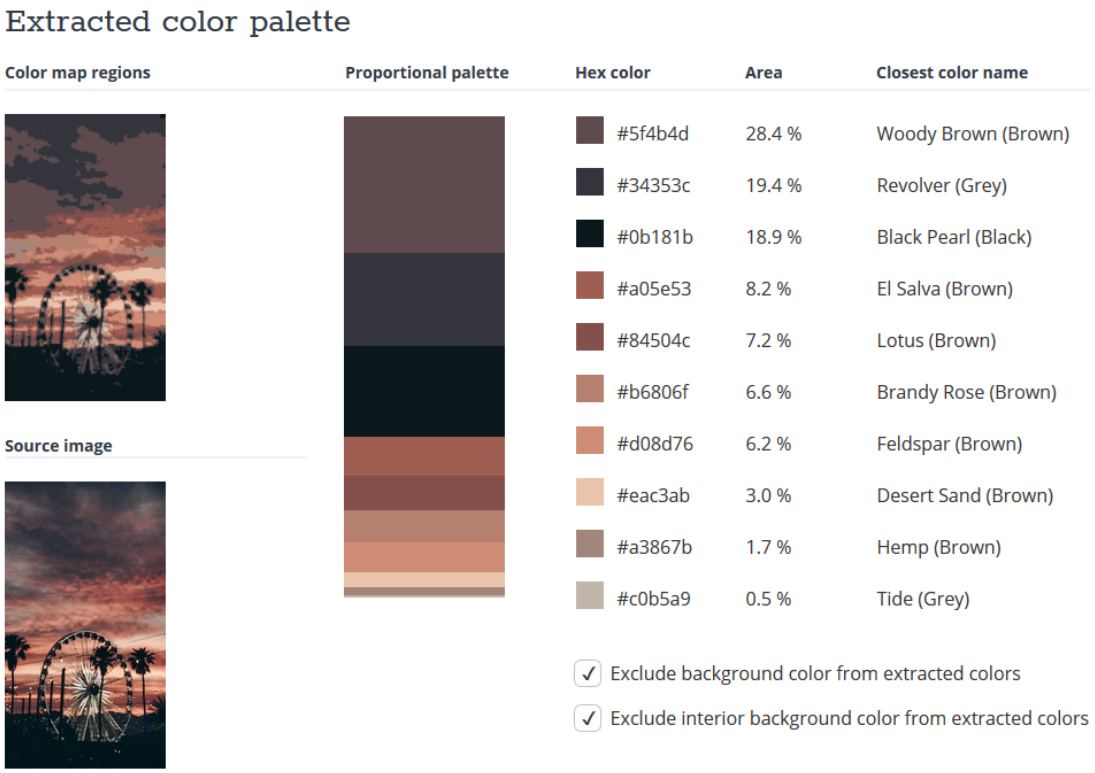

Рисунок 1.8 – Онлайн сервіс «TinEye»

Відповідно до здійсненого аналізу, були виявлені такі основні недоліки вказаних аналогів:

- повільна обробка об’ємних зображень;
- відсутність можливості відокремлення кольорів, до яких людське сприйняття є більш чутливим;
- обмежений функціонал у вигляді неможливості одночасного аналізу набору зображень;
- побудови вибірки з раніше проаналізованих;
- використання клієнт-серверної архітектури (що зменшує швидкодію аналізу).

Крім того, недоліком є необхідність завантаження зображення на сервер третьої сторони, що в окремих випадках не є бажаним так як зображення можуть містити чутливу інформацію.

1.3 Постановка задачі

Оскільки проаналізовані методи та аналоги мають низку недоліків в колористичній обробці зображень, перед початком розробки були поставлені наступні задачі:

- розробити модифікацію методу колористичної обробки зображень з підвищеною швидкодією обробки та покращеним визначенням домінантної палітри кольорів;
- розробити програмну систему для впровадження методу;
- розробити можливість експорту результатів обробки у вигляді XML файлів;
- розробити засоби візуалізації кластеризації зображення;
- розробити засоби паралельної обробки вхідного набору зображень;
- провести тестування системи, виправити виявлені помилки.

Також, необхідно забезпечити можливість збереження вхідних та вихідних даних для їхнього подальшого зберігання та аналізу шляхом створення нормалізованої та структурованої бази даних. Окрім цього, потрібно забезпечити цілісність вихідних даних, створивши механізм, що буде контролювати процес обробки зображень та отримання відповідного результату.

1.4 Висновки

Отже, після здійсненого аналізу існуючих методів колористичного оброблення зображень для визначення домінуючої палітри кольорів та існуючих програмних реалізацій, було виявлено ряд недоліків, що свідчить про актуальність розробки відповідних методів та засобів, що, окрім вирішення вже існуючих недоліків в швидкодії та точності аналізу, розширювали б можливості вже існуючих реалізацій шляхом додання нових способів взаємодії з вхідними та вихідними даними через візуалізацію кластеризації зображення, здатності виконувати паралельну обробку цілого набору зображень та зберіганні отриманих результатів для їх подальшої обробки та аналізу.

2 РОЗРОБКА МОДИФІКАЦІЇ МЕТОДУ КЛАСТЕРИЗАЦІЇ К-СЕРЕДНІХ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ВИЗНАЧЕННЯ ДОМІНУЮЧОЇ ПАЛІТРИ КОЛЬОРІВ ЗОБРАЖЕННЯ

2.1 Розробка модифікації алгоритму кластеризації методом к-середніх для підвищення швидкодії обробки зображень

Метод кластеризації к-середніх, як алгоритм кластерного аналізу, має на меті розбиття заданої вибірки об’єктів на підмножини (кластери) так, щоб кожен кластер складався з елементів максимально подібних між собою, а об’єкти різних кластерів – істотно відрізнялися.

При здійснені колористичного оброблення зображення необхідно визначити початкові дані. Так як вхідними даними є зображення, процес кластеризації потрібно виконувати на основі його ключових характеристик – матриці пікселів та колірного забарвлення окремо взятого пікселя.

Матриця пікселів – декартова система координати, яка знаходиться в межах розмірів зображення (його висоти ти ширини) і представляє набір пікселів цього зображення.

Кожен окремо взятий піксель, окрім власних координат, має своє представлення кольору, яке можна описати використовуючи такі колірні моделі як RGB, CMYK, HSV тощо [9].

Маючи вказані характеристики, процес кластеризації задля колористичного оброблення зображення, буде ґрунтуватись на використанні колірного забарвлення окремо взятого пікселя як частини вхідної вибірки даних.

Для опису цього окремо взятого пікселя буде використано колірну модель RGB, як є адитивною колірною моделлю, що описує спосіб синтезу кольору, за яким червоне, зелене та синє світло накладаються разом, змішуючись у різноманітні кольори.

Перевагами використання цієї моделі є:

- апаратна близькість із моніторами, сканерами, проекторами та іншими пристроями;

- велика колірна гама, яка є дуже близькою до можливостей людського ока;
- невеликий (порівняно з моделлю CMYK) обсяг, проте ширший спектр кольорів;

У вказаній моделі колір кодується градаціями складових каналів (Red, Green, Blue). Тому за збільшення величини градації котрогось каналу – зростає його інтенсивність під час синтезу.

Кількість градацій кожного каналу залежить від розрядності бітового значення RGB. Зазвичай використовують 24-бітну модель, у котрій визначається по 8 біт на кожен канал, і тому кількість градацій дорівнює 256 [10].

Відповідно до вказаної особливості колірного простору RGB, колірне забарвлення кожного пікселя можна представити у вигляді трьох цифрових значень, кожне з яких знаходиться в межах від 0 до 255 (рисунок 2.2).

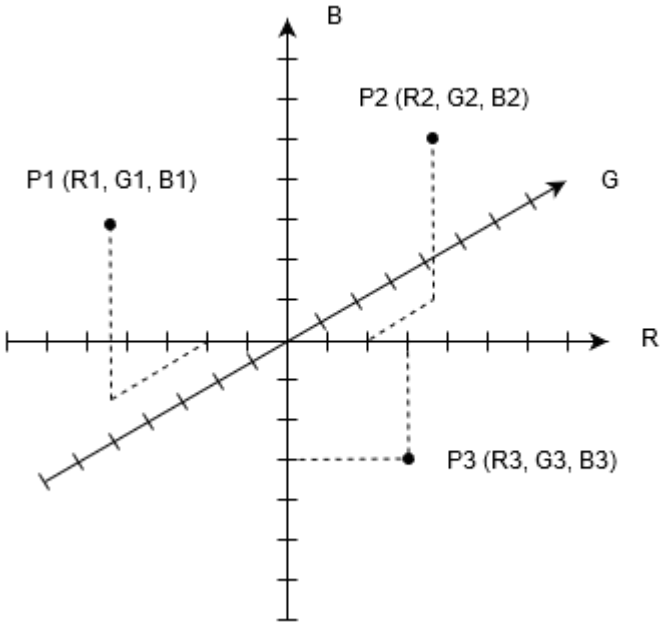

Рисунок 2.2 – Колірне представлення окремого пікселя в RGB просторі

Визначивши початкові дані для здійснення процесу кластеризації, необхідно їх адаптувати відповідно до кроків алгоритму.

Для першого кроку, необхідно визначити початкову кількість кластерів, яка буде використовуватись при здійснені аналізу методом к-середніх.

Цей крок, є доволі важливим так як він є початковою точкою для здійснення аналізу. Окрім цього, визначивши необхідну кількість кластерів, можна вирішити один з недоліків цього алгоритму. Так як, модифікація алгоритму націлена на колористичний аналіз зображення, кількість кластерів повинна відповідати бажаній кількості очікуваних домінантних кольорів.

Оптимальна кількість кластерів для розроблювального методу залежить від бажаного результату, але в якості оптимального рівня взято 10 кластерів. Саме така кількість кластерів дозволяє отримати деталізований результат аналізу без використанням великого об’єму ресурсів та часу.

Наступним кроком, випадковим чином обирається центри початкових кластерів. Вибір здійснюється серед кольорів, що присутні на зображені.

Після цього, здійснюється аналіз кожного елемента спостереження. У випадку з колористичним аналізом, елементом спостереження є колірна складова одиничного пікселя. Сам аналіз будується на знаходженні найменшої відстані між центром кластера й кольором пікселя. Для його здійснення використовується формула Евклідової відстані:

$$d(S,p) = \sqrt{(S_R - p_R)^2 + (S_G - p_G)^2 + (S_B - p_B)^2}, \quad (2.1)$$

де

$d(S,p)$ – відстань між кластером та елементом спостереження;

S – центр кластеру;

p – піксель, що аналізується;

S_R, S_G, S_B – колірне представлення кластеру в просторі RGB;

p_R, p_G, p_B – колірне представлення пікселя в просторі RGB;

Аналіз триває допоки кожен піксель зображення не буде відноситись до певного кластера.

Наступний крок полягає у перерахунку центрів кластерів відповідно до здійсненого аналізу. Для цього використовується наступна формула:

$$m_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j,$$

де

$m_i^{(t+1)}$ – новий центр кластера для наступної ітерації;

$S_i^{(t)}$ – і-ий кластер поточної ітерації аналізу;

x_j – окремо взяте значення елементу спостереження відповідного кластера.

Перерахунок здійснюється для усіх кластерів. Після отримання набору кластерів для наступної ітерації, здійснюється порівняння з відповідним кластером на попередній ітерації. Для цього також використовується формула Евклідової відстані подібна до формули 2.1, лише вхідні параметри – це центри кластерів на поточній та наступній ітерації аналізу.

Якщо отриманий результат порівняння є стійким (дорівнює нулю або встановленій межі точності) – процес аналізу зображення рахується завершеним і на виході отримуємо кінцеві кластери, що представляють собою домінантні кольори аналізованого зображення.

Враховуючи вказаний алгоритм обробки зображення, розглянемо наступний шлях до його модифікації.

Так як при здійснені аналізу, необхідно виконати обрахунки для кожного окремо взятого елемента спостереження протягом певної кількості ітерацій, існує пряма залежність між розміром вхідного зображення і швидкістю вказаних обрахунків. Так як елементом спостереження є колірне забарвлення пікселя, процес аналізу деталізованих зображень у форматі Full HD або Full HD+ стає вкрай тривалим та таким, що потребує великого об’єму ресурсів. Це пояснюється тим, що мінімальний розмір вхідного зображення є 1920 на 1080 пікселів, тобто 2073600 елементів спостереження, які підлягають окремій обробці та аналізу

протягом певної кількості ітерацій. Слід зауважити, що аналіз зображень, які не досягають деталізації вказаних форматів також є об’ємним, так як на окремо взятому зображені може бути більше одного мільйона елементів спостереження і саме такий формат є найбільш розповсюдженим у повсякденному використанні.

Враховуючи вказаний недолік, пропонується зменшити розміри зображення для підвищення швидкодії обробки шляхом зменшення кількості операцій з обробки окремих елементів спостереження. Так як кількість елементів спостереження є пропорційним до розмірів зображення, зменшивши його до певних розмірів можна отримати вказане підвищення в швидкодії без суттєвих втрат в точності обробки. Слід зауважити, що такий підхід може застосовуватись лише для зображень, які дозволяють вказане зменшення розмірів, тому що у випадку вкрай малих зображень похибка обрахунків елементів спостереження є занадто високою, що є неприйнятним для кінцевого результату.

Для знаходження оптимального розміру зображення для його зменшення, здійснимо обрахунку розроблювальним методом, використовуючи в якості вхідних даних зображення розміром 736 на 1308 пікселів (962688 елементів спостереження) та кількості кластерів рівне десяти (таблиця 2.1).

Таблиця 2.1

Знаходження оптимального розміру зображення

Відсоток від початкових розмірів, %	Розмір	Швидкість, секунд	К-ть ітерацій	Відхилення
100%	736×1308	18,9984839	7	0
67%	490×872	8,4859505	7	3,31
50%	368×654	5,9037977	9	2,56
40%	294×523	2,1571322	5	4,77
33,3%	245×436	1,4734666	5	4,32
28,6%	210×373	1,0905479	5	4,23
25%	184×327	2,897903	16	8,84
22,22%	163×290	2,0792058	16	8,61
20%	147×261	0,8768856	8	11,1
18,18%	133×237	0,4657755	5	15,56

Відповідно до отриманих результатів таблиці 2.1, оптимальним розміром до якого можна зменшити зображення є 33.3-28.6% від оригінального розміру, тобто до розмірів просторів зображення не менше 200×200. В свою чергу, це дозволяє зменшити час на обробки зображення з майже 19 секунд до приблизно 1, що майже в 19 разів швидше за обробку початкового зображення без його зменшення. При цьому слід зауважити, що параметр відхилення обраховувався як середнє значення відхилення кожного з параметрів кольору в просторі RGB від отриманого кольору при обробці 100% зображення. Враховуючи це, відхилення в 5-6 позначок не є помітними так як зміна зображення знаходиться в межах тону визначеного кольору, та не є помітним людському оку. При цьому, середнє відхилення більше ніж 5-6 позначок означає про неточність результату та про помітну втрату в точності визначення домінуючої палітри кольорів принаймні в одному з визначених кластерів.

Отже, запропонований підхід обробки зображень, що передбачає зменшення кількості елементів спостереження до оптимальної кількості на початковій фазі аналізу, а саме до 33.3-28.6% від початкових розмірів зображення, дозволяє збільшити швидкість обробки зображення в 19 разів. Вказаний результат вдалось досягти у випадку обробки зображення розміром 736 на 1308 пікселів (962688 елементів спостереження) та кількості кластерів рівне десяти. У випадках з більш деталізованими зображеннями, швидкість обробки збільшиться відповідно до розмірів зображення.

2.2 Розробка покращеного методу визначення домінуючої палітри кольорів для покращення результату обробки зображень

Результатом виконання розробленої модифікації алгоритму кластеризації методом к-середніх буде набір кольорів, що переважає на заданому вхідному зображені.

Недоліком цього методу є те, що при обрахуванні домінуючої палітри кольорів не враховується людське сприйняття вказаних кольорів.

Наприклад, при обробці зображення, котре складається з переважно сірих, чорних та білих кольорів, але при цьому використання вказаних кольорів повинно підкреслити інші, більш яскраві та насичені кольори на зображені, використання вказаного алгоритму в якості вихідної палітри домінуючих кольорів дасть саме вказані сірі, білі та чорні кольори, так як сумарна кількість належних до вказаних кольорів пікселів буде переважати кількість пікселів, що належить до кольорів, на яких був зроблений акцент.

Через вказаний недолік, пропонується: до кроку з визначенням початкових значень кластерів досліджуваного зображення, додати фільтрацію вказаних кольорів, що повинно в свою чергу забезпечити результат, який підлягає кращому сприйняттю людським оком.

Так як вхідне зображення є у форматі Truecolor, окремо взяті елементи спостереження (пікселі) знаходяться у RGB просторі і представлені у відповідній комбінації червоного, зеленого та синього кольорів (рисунок 2.3) [11].

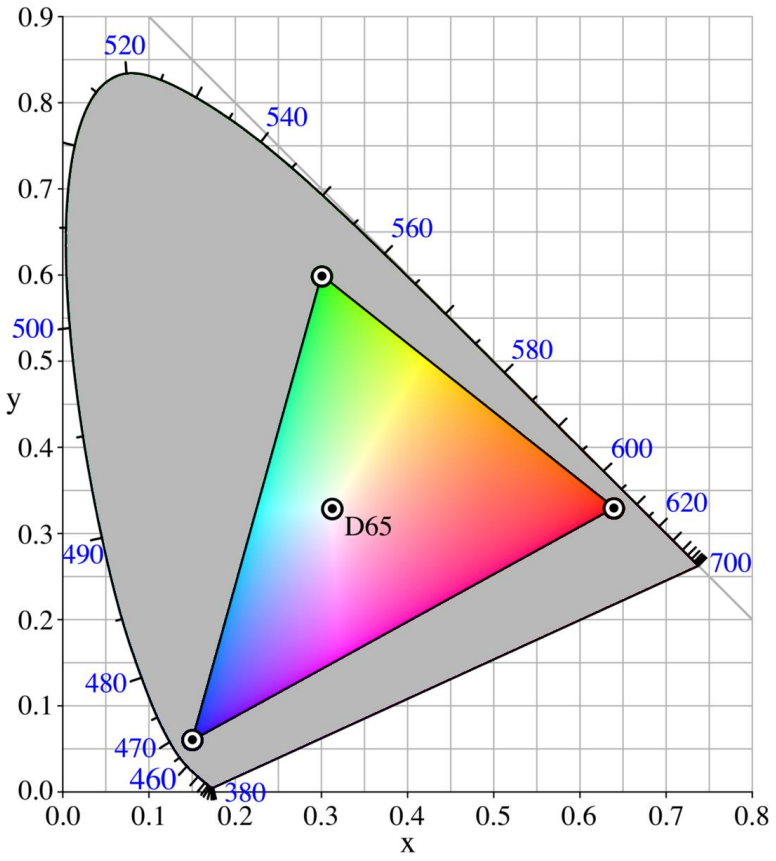

Рисунок 2.3 – Трикутник кольорів утворених за допомогою формату RGB

Саме тому, відповідно до вказаного представлення кольору, пропонується здійснювати фільтрацію кольорів, використовуючи формулу подібну до формули 2.1 на основі такого правила: окремо взятий початковий центр кластеру, повинен мати отриману відстань до білого та чорного кольорів рівну або більше 200-та позначок. Здійснювати вказану фільтрації потрібно на першому кроці ініціалізації центрів кластерів досліджуваного зображення. Саме таке правило, надає необхідне покращення у визначенні домінантної палітри кольорів.

Отже, запропонований підхід визначення домінуючої палітри кольорів зображення дозволяє отримати результат, котрий буде зосереджений на особливостях людського сприйняття кольорів, тим самим покращуючи сприйняття останньої отриманих результатів.

2.3 Висновки

Отже, в ході виконання розробки модифікації методу кластеризації к-середніх, було здійснено: дослідження відповідного алгоритму кластеризації; розроблено його модифікацію, яка може бути застосована для обробки зображень; запропоновано модифікацію, котра базується на зменшенні розмірів вхідного зображення та кількості досліджуваних елементів, що покликано зменшити кількість часу та об’єм ресурсів необхідних для виконання кінцевої обробки; запропоновано покращення методу визначення домінуючої палітри кольорів, котре передбачає здійснення фільтрації кольорів кластерів досліджуваного зображення на їхньому початковому кроці ініціалізації шляхом обчислення їхньої відстані до білого та чорного кольорів відповідно.

3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ МЕТОДУ ВИЗНАЧЕННЯ ДОМІНУЮЧОЇ ПАЛІТРИ КОЛЬОРІВ ЗОБРАЖЕННЯ

3.1 Варіативний аналіз та обґрунтування вибору програмних засобів вирішення задач магістерської кваліфікаційної роботи

Серед багатьох технологій, які сьогодні доступні розробникам програмного забезпечення, для поточного проекту потрібно обрати систему керування базами даних, та технологію створення інтерфейсу користувача. Серед різноманіття доступних варіантів для подальшого вибору були обрані наступні:

1. Windows Presentation Foundation.
2. Windows Forms.
3. Microsoft SQL Server.
4. MySQL.

Windows Presentation Foundation – це система для побудови клієнтських додатків Windows з візуально привабливими можливостями взаємодії з користувачем, графічна підсистема у складі .NET Framework, яка використовую мову XAML. З допомогою Windows Presentation Foundation можна створювати звичайні програмні додатки, так і додатки, які запускаються у браузері. В основі WPF лежить векторна система візуалізації, яка не залежить від роздільної здатності пристрою виводу і створена з урахуванням можливостей сучасного графічного обладнання. Графічною технологією, що лежить в основі WPF є DirectX, на відміну від Windows Forms, де використовується GDI/GDI+.

Особливостями цієї системи є гнучкість у створенні зовнішнього вигляду інтерфейсу користувача, можливість описувати інтерфейс користувача у декларативній манері за допомогою мови XAML, наявність технології «зв’язування даних», яка дозволяє блискавично оновлювати стан інтерфейсу користувача у відповідь на зміну властивостей, до яких виконується прив’язування [12].

Windows Forms – це інтерфейс програмування додатків, який відповідає за графічний інтерфейс користувача та є частиною .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за рахунок створення обгортки для існуючого Win32 API в керованому коді. Особливістю цієї технології є простота у використання. Особливістю цієї технології є простота у використання, але при цьому їй не вистачає гнучкості та декларативної манери опису інтерфейсу користувача. Крім того, сьогодні, ця технологія вже вважається застарілою, а сама компанія Microsoft припинила розробку оновлень, але залишила виправлення помилок [13].

Microsoft SQL Server – це система керування реляційними базами даних, розроблена компанією Microsoft. Основна мова запитів, яка використовується – T-SQL. Така система керування підходить для використання як в невеликих та середніх за розміром баз даних так і в великих базах даних масштабу цілого підприємства. Особливостями цього програмного забезпечення є тісна інтеграція з ОС Windows та власна мова запитів T-SQL, яка була доповнена процедурними розширеннями, що збільшило її можливості [14].

MySQL – це вільна реляційна система керування базами даних. Розробку та підтримку проводить компанія Oracle. Основною особливістю цього програмного забезпечення є вільне розповсюдження [15].

Отже, в результаті аналізу особливостей технологій, які були розглянуті, прийнято рішення обрати для розробки програмних модулів додатку WPF та Microsoft SQL Server.

3.2 Вибір середовища програмування

Для розробки програмного забезпечення, потрібно обрати інтегроване середовище розробки (IDE − Integrated Development Environment). IDE – це програма, що допомагає розробнику створювати нове програмне забезпечення та вдосконалювати вже існуюче. Для порівняння було обрано Microsoft Visual Studio, Visual Studio Code та JetBrains Rider.

Microsoft Visual Studio – це серія продуктів фірми Microsoft, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Presentation Foundation, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight. Перевагами є інтуїтивно зрозумілий інтерфейс, підсвічування синтаксису, покрокове виконання коду, вбудована підтримка Git. До недоліків можна віднести відсутність підказки про потребу підключення необхідних для використання окремих класів бібліотек .

Visual Stduio Code – це засіб для створення, редагування та зневадження сучасних веб-застосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X. Редактор містить вбудований зневаджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціонується як легковаге рішення, що дозволяє обійтися без повного інтегрованого середовища розробки. Перевагою є невеликий розмір програми. Недоліком є відсутність багатьох функцій повноцінних IDE.

JetBrains Rider – це кросплатформне інтегроване середовище розробки програмного забезпечення для платформи .NET, розроблюване компанією JetBrains. Підтримуються мови програмування C#, VB.NET та F#. Проект було анонсовано у січні 2016 року. У його основі лежить інший продукт JetBrains – ReSharper. Середовище підтримує платформи .NET Framework, .NET Core та Mono. Працює на операційних системах Windows, MacOS, Linux. Перевагою є кросплатформність. Недоліком є невідшліфованість нового продукту.

Після дослідження переваг та недоліків наведених вище IDE було прийнято рішення використовувати для розробки Microsoft Visual Studio. Дана

програма має багато зручних функцій, які значно полегшують процес розробки та роблять його швидшим.

3.3 Розробка загальної структури системи

Для створення системи було вирішено застосувати архітектуру, в якій уся інформація про зображення (вмісту зображення, перелік палітри кольорів, час аналізу тощо) буде зберігатися в базі даних, доступ до якої забезпечує спеціальний програмний модуль (рисунок 3.1). Окрім цього реалізована можливість для введення вхідних даних безпосередньо через обрання декількох зображення (для цього дані повинні відповідати визначеному формату). Це спрощує взаємодію з функціоналом розроблювальної програмної системи, оскільки через відсутність складних послідовностей дій, використання цих даних не є проблематичним.

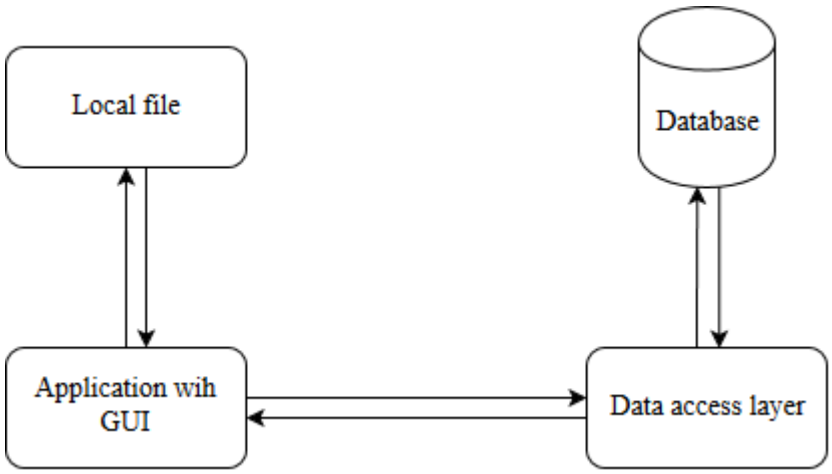

Рисунок 3.1 – Загальна структура системи

Така архітектура дозволяє зберігати великі обсяги інформації в базі даних, що є доволі зручним, оскільки дозволяє окрім самого збереження також вільно оперувати наявними даними (наприклад статистикою часу аналізу). Це проявляється в змозі швидко та без труднощів переміщувати бази даних між різними користувачами та надає можливість для створення централізованої статистики на основі використання програмної системи групою користувачів.

Також, це дає можливість використовувати отримані результати й в інших програмних додатках, оскільки уся інформація може бути отримана через використання SQL запитів та відповідних фреймворків.

Окрім цього, присутність локального файлу надає змогу працювати з програмною системою й при відсутності бази даних у користувача. Так, він зможе здійснювати ініціалізацію даних для здійснення аналізу, а після нього – переглядати статистику. Однак, відсутність бази даних зменшує можливість для порівняльного аналізу даних після проведеного аналізу оскільки буде відсутня статистична інформація з попередніх аналізів, по якій й здійснюється аналіз.

3.4 Розробка структури бази даних

Темою магістерської кваліфікаційної роботи є розробка методів і засобів колористичного оброблення зображень та системи на їх основі, і на підставі порівняльного аналізу, було виявлено, що більша частина розглянутих аналогів має обмежений функціонал, не бере до уваги чутливості людського сприйняття, від чого страждає точність оцінки, не має можливості аналізу більше одного зображення за раз, що не є зручним при роботі з великими обсягами даних. Тому, перш за все, в базі даних потрібно зберігати таку інформацію як наприклад: вміст зображення, час за який було виконано аналіз, отриманий результат аналізу тощо. Окрім цього потрібно розуміти, що окрім статистичних властивостей потрібно також зберігати й зображення, що аналізується, оскільки це дасть змогу здійснювати більш точні та детальні аналізи і спростить пошук необхідної палітри домінантних кольорів для майбутніх досліджень.

Далі, необхідно розробити універсальне відношення.

Універсальне відношення – це відношення, яке містить в собі усі атрибути, які будуть розміщені в базі даних в майбутньому. Вони мають визначену структуру, в якій кожне значення є атомарним (тобто неперервним) та не пустим.

Відношення може бути подане у вигляді файлу або таблиці, стовпці котрих – елементи доменів, а рядки – кортежі. Кожен кортеж відображає один екземпляр інформаційного об'єкта. Імена стовпців (поле запису) називаються атрибутами, а

індивідуальні значення елементів – значеннями атрибутів. Кожен атрибут відображає відповідну характеристику інформаційного об'єкта. Число стовпців у відношенні називається ступенем відношення, а число кортежів – потужністю відношення. У процесі експлуатації бази даних ступінь відношення змінюється значно рідше, ніж його потужність.

Атрибут або набір атрибутів, котрий можна використовувати для однозначної ідентифікації конкретного кортежу, називається початковим ключем (у випадку набору атрибутів – складений ключ). Можливі випадки, коли відношення може вміщувати декілька унікальних ключів. Тоді один з них вибирається як головний початковий, а інші отримують назву початкових ключів.

Атрибути, що представляють копії ключів інших відношень, називаються зовнішніми ключами.

Атрибут або набір атрибутів, що використовуються для більш швидкого пошуку, називається другорядним індексом [16].

При використанні універсального відношення виникає декілька проблем:

- надмірність – дані практично усіх стовпців повторюються;
- потенційна суперечність – при оновленні даних необхідно переглядати усю таблицю для знаходження та заміни необхідних рядків;
- аномалії включення – неможливо додати нові дані, якщо при цьому створюються рядки з невизначеним станом;
- аномалії видалення – аналогічна аномалії включення, але виникає при необхідності видалення рядків.

Відповідно до проведеного аналізу, до універсального відношення потрібно додати атрибути, які будуть описувати такі інформаційні об’єкти як:

- вхідне зображення;
- вихідне зображення.

Для цих інформаційних об’єктів були створенні такі атрибути:

- вхідне зображення – ідентифікатор вхідного зображення, розмір зображення, розширення зображення, вміст зображення;
- вихідне зображення – ідентифікатор вихідного зображення, час аналізу, перелік домінантних кольорів, вміст проаналізованого зображення.

В таблиці 3.1 наведено перелік атрибутів універсального відношення.

Таблиця 3.1

Перелік атрибутів універсального відношення

№	Назва атрибута	Ім’я поля	Коментар
1	Ідентифікатор вхідного зображення	InputImgID	Ідентифікатор вхідного зображення
2	Розмір зображення	ImgSize	Розмір зображення
3	Розширення зображення	ImgExtension	Розширення зображення
4	Вміст зображення	ImgContent	Вміст вхідного зображення
5	Ідентифікатор вихідного зображення	AnalyzedImgID	Ідентифікатор вихідного зображення
6	Час аналізу	AnalysisTime	Час аналізу зображення
7	Перелік домінантних кольорів	DominantColors	Перелік домінантних кольорів
8	Вміст проаналізованого зображення	AnalyzedImgContent	Вміст проаналізованого зображення

Атрибути вказані в таблиці 3.1 є незалежними, це означає, що їхні значення не можуть бути обчислені значеннями інших атрибутів і вони можуть бути включені в склад універсального відношення.

Відповідно до кількості атрибутів в таблиці 3.1 потужність універсального відношення дорівнює 8 і має такий вигляд: R(Ідентифікатор вхідного зображення, розмір зображення, розширення зображення, вміст зображення, ідентифікатор вихідного зображення, час аналізу, перелік домінантних кольорів, вміст проаналізованого зображення).

Наступний крок полягає у розробці ER-діаграми.

Це є заключна фаза аналізу предметної області і полягає вона у розробці концептуальної схеми, що базується на моделі «суть-зв’язок».

ER-модель або модель «сутність-зв’язок» – це модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків (сутності та зв’язки). Така модель слугує засобом опису моделей даних. Сутність – деяка абстракція реально існуючого об’єкта. Кожна не слабка сутність повинна мати мінімальний набір унікальних атрибутів (первинний ключ). Сутність фактично є множиною атрибутів, які описують визначений об’єкт. Кожен інформаційний об’єкт є екземпляром сутності.

Між сутями (інформаційними об’єктами), а також між суттю і її атрибутами з’являються деякі асоціації, що називаються зв’язками. При цьому зв’язки можуть бути різних властивостей, характеру та вибірності. Звичайно зв’язок виражається дієсловом. Зв’язки як і суті іменуються, при цьому екземпляр кожного окремого зв’язку специфікується лінією між тими двома екземплярами суті, котрі цей зв’язок з’єднують. Атрибут (або набір атрибутів), що використовується для ідентифікації суті, називається ключем суті.

Суті можуть з’єднуватися зв’язками різної ступені. Ці ступені бувають:

- 1:1 – ступінь, що свідчить про те, що один екземпляр однієї суті може бути зв’язаним не більше ніж з одним екземпляром іншої суті;
- 1:N – ступінь, що свідчить про те, що один екземпляр однієї суті може бути зв’язаним з декількома екземплярами іншої суті, але не навпаки;
- N:M – ступінь, що свідчить про те, що кожен екземпляр однієї суті може бути зв’язаним з декількома екземплярами іншої суті.

На основі інформаційних об’єктів бази даних, в ER-моделі можна виділити такі сутності та ключі:

- Вхідне зображення (Ідентифікатор вхідного зображення);
- Вихідне зображення (Ідентифікатор вихідного зображення).

Зв’язок сутностей «Вхідне зображення-Вихідне зображення» характеризується ступенем 1:N та класом належності «необов’язковий-обов’язковий». Вхідне зображення може бути проаналізоване за різних вхідних параметрів, тому може входити в декілька вихідних зображень. Клас належності вказує на те, що вхідне зображення може не входити в жодне вихідне зображення, але вихідне зображення обов’язково повинен мати в собі вхідне зображення. Діаграму зображено на рисунку 3.2.

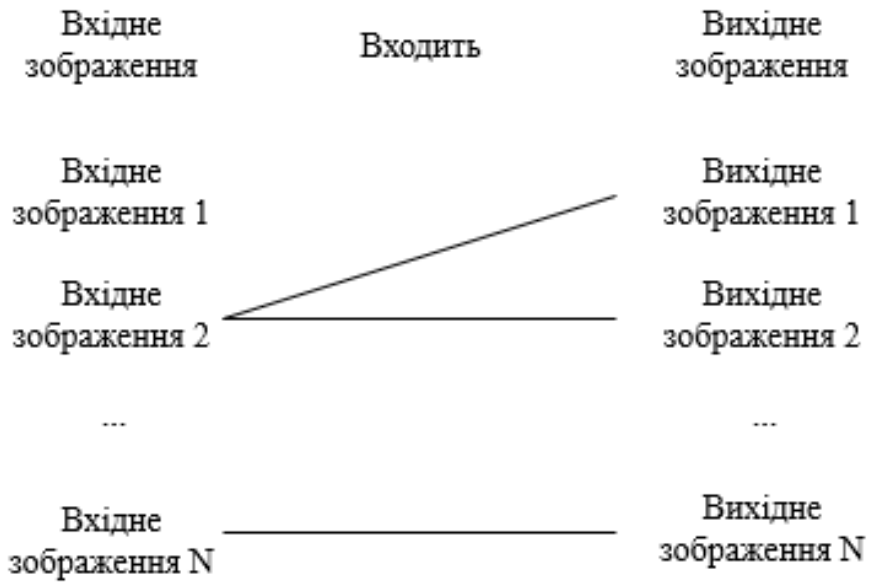

Рисунок 3.2 – ER-діаграма екземплярів сутностей «Вхідне зображення-Вихідне зображення»

Модель ER-діаграм екземплярів сутностей представлено у таблиці 3.2.

Таблиця 3.2

Зв’язки екземплярів сутностей ER-моделі бази даних

Ім’я сутності 1	Ім’я сутності 2	Назва зв’язку	Тип зв’язку	Клас належності
Вхідне зображення	Вихідне зображення	Входить	1:N	Необов.: обов.

На основі даних таблиці 3.2 можна побудувати ER-діаграму типів предметної області (рис. 3.3).

Оскільки загальноприйнятого формату для графічного подання ER-діаграми немає, було використано спосіб запропонований Дж. Ульманом. Але розбіжності, при використанні різних форматів, як правило, несуттєві.

Рисунок 3.3 – ER-діаграма типів предметної області

Вказана діаграма повністю відображає зв’язки між типами сутностей.

Наступний крок полягає у проектуванні нормалізованих відношень.

Нормалізація – це покроковий процес розбиття одного відношення (таблиці) відповідно до алгоритму нормалізації на декілька відношень на базі функціональних залежностей. Нормалізація дозволяє забезпечити захист даних та робить базу даних більш гнучкою, усуваючи надмірність.

Надмірність даних призводить до непродуктивного керування вільною пам’яттю комп’ютера, забираючи її під дублювання однієї й тієї ж інформації. Окрім цього, щоб зробити зміни в базі даних, доведеться змінювати необхідний рядок інформації всюди де він дублюється.

Відношення знаходиться в першій нормальній формі (1НФ) тоді і тільки тоді, коли в будь-якому допустимому значенні відносини кожен його кортеж містить тільки одне значення для кожного з атрибутів [17].

Відношення знаходиться в другій нормальній формі (2НФ) тоді і тільки тоді, коли воно знаходиться в першій нормальній формі і кожен неключових атрибут функціонально повно залежить від її потенційного ключа.

Відношення знаходиться в третій нормальній формі (3НФ) тоді і тільки тоді, коли воно знаходиться в другій нормальній формі, і відсутні транзитивні функціональні залежності неключових атрибутів від ключових.

Взагалі, транзитивні залежності являються коректними, але через надлишковість, їх використання у процесі проектування не потрібне. В зв'язку з цим транзитивні залежності потрібно виключати з набору перед початком проектування.

Відношення знаходиться в нормальній формі Бойса-Кодда (НФБК), якщо між ключами-кандидатами немає функціональної залежності.

Для проведення нормалізації, розбиваємо універсальне відношення R на такі відношення:

- R1 (<Ідентифікатор вхідного зображення>, розмір зображення, розширення зображення, вміст зображення).
- R2 (<Ідентифікатор вихідного зображення>, час аналізу, перелік домінантних кольорів, вміст проаналізованого зображення).

Оскільки кортежі не повторюються, всі атрибути атомарні (1НФ), всі ключі прості (2НФ), відсутні транзитивні залежності (3НФ), немає функціональних залежностей між ключами-кандидатами, то всі відношення знаходяться у НФБК.

Наступний крок – одержання попередніх відношень методом «суть-зв’язок».

Зв’язки будуються за певними правилами, залежно від класу належності відношень у зв’язку та залежно від ступені зв’язку:

1. Якщо ступінь бінарного зв’язку дорівнює 1:1 і клас належності обох сутностей є обов’язковим, то потрібно лише одне відношення, первинним ключем якого може бути ключ будь-якого з двох відношень.

2. Якщо ступінь бінарного зв’язку дорівнює 1:1 і клас належності однієї сутності є обов’язковим («О»), а другої не обов’язковим («НО»), то необхідно побудувати два відношення. На кожну сутність потрібно виділити одне відношення, при цьому ключ сутності повинен слугувати первинним

ключем для відповідного відношення. Крім того, ключ сутності, для якої клас належності є «НО», додається в якості атрибута у відношення, що виділене для сутності з обов'язковим класом належності.

3. Якщо ступінь бінарного зв'язку дорівнює 1:1 і клас належності обох сутностей є «НО», то необхідно використовувати три відношення: по одному для кожної сутності, ключі яких слугуватимуть в якості первинних у відповідних відношеннях та одного відношення для зв'язку. Відношення, що виділяється для зв'язку буде мати по одному ключу сутності від кожної сутності.

4. Якщо ступінь бінарного зв'язку дорівнює 1:N та клас належності N-зв'язної сутності є «О», то достатнім є використання двох відношень по одному на кожну сутність за умови, що ключ кожної сутності слугує в якості первинного ключа для відповідного відношення. Додатково ключ однозв'язної сутності повинен бути доданий, як атрибут у відношення, що відводиться для N-зв'язної сутності.

5. Якщо ступінь бінарного зв'язку 1:N і клас належності багатозв'язної сутності є необов'язковим, то необхідно формувати три відношення: по одному для кожної сутності з відповідними ключами і одне для зв'язку аналогічно правилу №3.

6. Якщо ступінь бінарного зв'язку M:N, то для зберігання даних потрібні три відношення: по одному для кожної суті, причому ключ кожної суті служить як первинний ключ для відповідного відношення, та одного відношення для зв'язку. Зв'язок повинен мати серед своїх атрибутів ключі кожної зі зв'язних сутей.

Відповідно до рисунка 3.3, усі відношення відповідають правилу 4.

Використовуючи множину атрибутів з універсального відношення, що знаходяться в таблиці 3.1, ER-діаграму типів (рисунок 3.3) та зазначені правила, представимо попередні відношення (таблиця 3.3).

Таблиця 3.3

Попередні відношення

Назва зв’язку	№ правила	Попередні відношення	Додаткові атрибути
Входить	4	R1 <Ідентифікатор вхідного зображення>	розмір зображення, розширення зображення, вміст зображення
		R2 <Ідентифікатор вихідного зображення>	час аналізу, перелік домінантних кольорів, вміст проаналізованого зображення, ідентифікатор вхідного зображення

Відношення після застосування правила №4:

- R1 (<Ідентифікатор вхідного зображення>, розмір зображення, розширення зображення, вміст зображення);
- R2 (<Ідентифікатор вихідного зображення>, час аналізу, перелік домінантних кольорів, вміст проаналізованого зображення, ідентифікатор вхідного зображення).

Отже, попередні відношення відповідають правилам методу «суть-зв’язок».

Кінцеві відношення, що описують базу даних, подані таким набором:

- Вхідне зображення (Ідентифікатор вхідного зображення, розмір зображення, розширення зображення, вміст зображення);
- Вихідне зображення (Ідентифікатор вихідного зображення, час аналізу, перелік домінантних кольорів, вміст проаналізованого зображення, ідентифікатор вхідного зображення).

Наступний крок полягає у нормалізації відношення методом декомпозиції.

Цей метод часто використовується для проектування великих баз даних.

Метод декомпозиції має наступний алгоритм:

1. Створення універсального відношення бази даних;
2. Визначення способів і типів зв’язку між атрибутами універсального відношення;
3. Визначення того, чи знаходиться розглянуте відношення в НФБК. Якщо так, проектування закінчується, якщо ні – відношення розбивається на два відношення;
4. Повторення кроків 2 та 3 для кожного нового відношення, отриманого в результаті декомпозиції.

При виконанні декомпозиції зберігається множина вихідних функціональних залежностей між атрибутами і виконується зворотність. Зворотність означає можливість відновлення вихідної схеми. Функціональні залежності відображають зв’язки між атрибутами, які властиві реальному об’єкту.

Згідно з алгоритмом декомпозиції, попереднє універсальне відношення виглядає так: R(<Ідентифікатор вхідного зображення>, розмір зображення, розширення зображення, вміст зображення, <ідентифікатор вихідного зображення>, час аналізу, перелік домінантних кольорів, вміст проаналізованого зображення).

Після декомпозиції попереднього універсального відношення отримаємо відношення R1 і R2:

- R1 (<Ідентифікатор вхідного зображення>, розмір зображення, розширення зображення, вміст зображення);
- R2 (<Ідентифікатор вихідного зображення>, час аналізу, перелік домінантних кольорів, вміст проаналізованого зображення).

Відношення R1 і R2 знаходиться в НФБК, декомпозиція завершена.

Функціональні залежності між атрибутами універсального відношення зображені на рисунку 3.4.

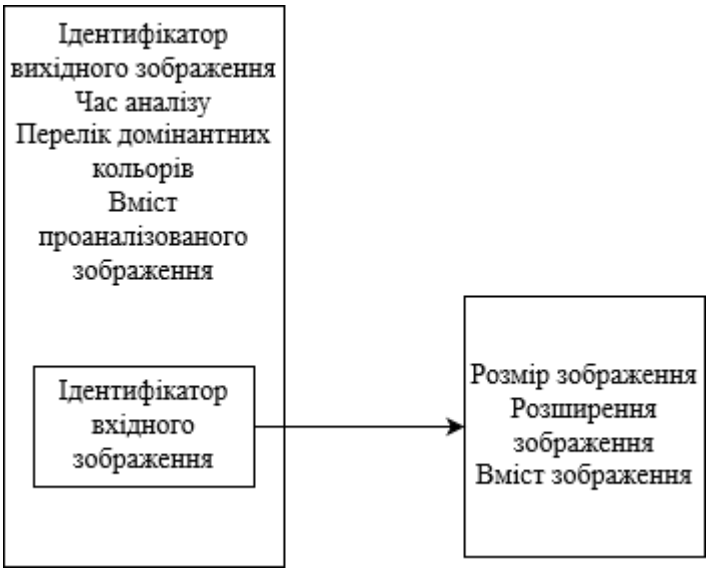

Рисунок 3.4 – Діаграма функціональної залежності

Відношення, що були отримані в результаті декомпозиції відповідають НФБК та мають мінімальний набір функціональних залежностей.

Наступним та кінцевим кроком є оцінка спроектованих НФБК відношень.

Після застосування методу декомпозиції отримано такі кінцеві відношення:

- R1 (<Ідентифікатор вхідного зображення>, розмір зображення, розширення зображення, вміст зображення);
- R2 (<Ідентифікатор вихідного зображення>, час аналізу, перелік домінантних кольорів, вміст проаналізованого зображення).

Після отримання кінцевих відношень, можна зробити такі висновки:

- кожна функціональна залежність не повторюється більше одного разу;
- набір функціональних залежностей є мінімальним.

Серед усіх відношень немає жодного, атрибути якого належали до атрибутів іншої функціональної залежності. Окрім того, неможливо якимось чином об’єднати будь-які два відношення так, щоб вийшло третє, уже існуюче відношення. Можна зробити висновок – жодне з відношень не є надлишковим, що свідчить про правильність проведення проектування.

3.5 Розробка структури системи з графічним інтерфейсом

При створенні програмної системи було використано архітектурний шаблон програмування MVVM (Model-View-ViewModel) [18].

Model-View-ViewModel – це шаблон проектування, що застосовується під час проектування архітектури застосунків (додатків). Публічно вперше був представлений Джоном Госсманом (John Gossman) у 2005 році як модифікація шаблону Presentation Model. MVVM орієнтований на такі сучасні платформи розробки, як Windows Presentation Foundation та Silverlight від компанії Microsoft.

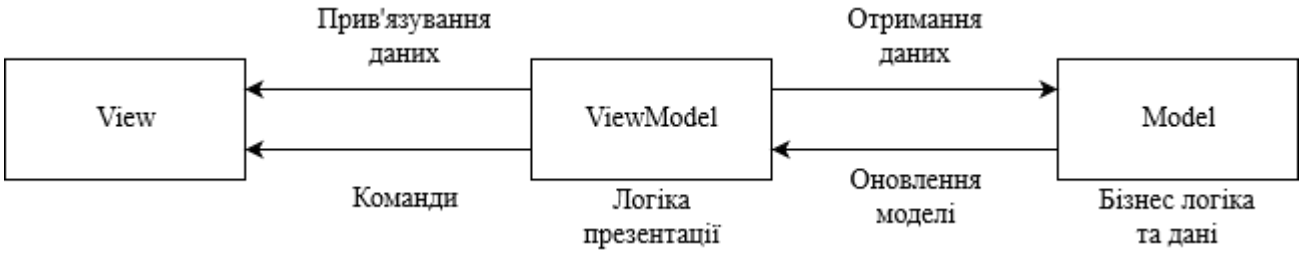

Рисунок 3.5 – Взаємодія між компонентами шаблону

MVVM полегшує відокремлення розробки графічного інтерфейсу від розробки бізнес логіки (бек-енд логіки), відомої як модель (можна також сказати, що це відокремлення представлення від моделі). Модель представлення є частиною, яка відповідає за перетворення даних для їх подальшої підтримки і використання. З цієї точки зору, модель представлення більше схожа на модель, ніж на представлення і оброблює більшість, якщо не всю, логіку відображення даних. Модель представлення може також реалізовувати патерн медіатор, організовуючи доступ до бек-енд логіки навколо множини правил використання, які підтримуються представленням.

Шаблон MVVM ділиться на три частини (див. рисунок 3.5):

- Модель (Model), як і в класичному шаблоні MVC, Модель являє собою фундаментальні дані, що необхідні для роботи застосунку.
- Вид/(Вигляд) (View) як і в класичному шаблоні MVC, Вигляд — це графічний інтерфейс, тобто вікно, кнопки тощо.

- Модель вигляду (ViewModel, що означає «Model of View») з одного боку є абстракцією Вигляду, а з іншого надає обгортку даних з Моделі, які мають зв’язуватись. Тобто вона містить Модель, яка перетворена до Вигляду, а також містить у собі команди, якими може скористатися Вигляд для впливу на Модель.

Фактично ViewModel призначена для того, щоб:

- Здійснювати зв'язок між моделлю та вікном.
- Відслідковувати зміни в даних, що зроблені користувачем.
- Відпрацьовувати логіку роботи View (механізм команд).

MVVM зручно використовувати замість класичного MVC та йому подібних у тих випадках, коли на платформі, де ведеться розробка, присутнє «зв'язування даних».

В MVC/MVP зміни у користувацькому інтерфейсі не впливають безпосередньо на модель, а йдуть через Контролер/Presenter. У таких технологіях, як WPF та Silverlight, присутня концепція «зв’язування даних», що дозволяє зв'язувати дані із візуальними елементами в обидві сторони.

Архітектура MVVM вирішує цю проблему чітким поділом відповідальності:

- Розробка користувацького інтерфейсу здійснюється дизайнером інтерфейсів за допомогою технології, більш-менш природної чіт такої роботи (XAML)
- Логіка користувацького інтерфейсу реалізується розробником як компонент ViewModel
- Функціональні зв’язки між користувацьким інтерфейсом та ViewModel реалізуються через прив’язки (bindings), які, по суті, є правилами типу «якщо кнопка A була натиснута, повинен бути викликаний метод onButtonAClick() з ViewModel». Прив’язки можуть бути написані в коді або визначені декларативним шляхом.

На основі проведеного аналізу, було вирішено реалізувати саме MVVM як архітектурний шаблон розроблювальної програмної системи.

3.6 Розробка структури інтерфейсу користувача

Робота у програмній системі зосереджена в 4 вікнах.

Перше вікно є головним, саме воно з’являється при запуску системи та саме в ньому зосереджено основний функціонал з колористичної обробки зображення.

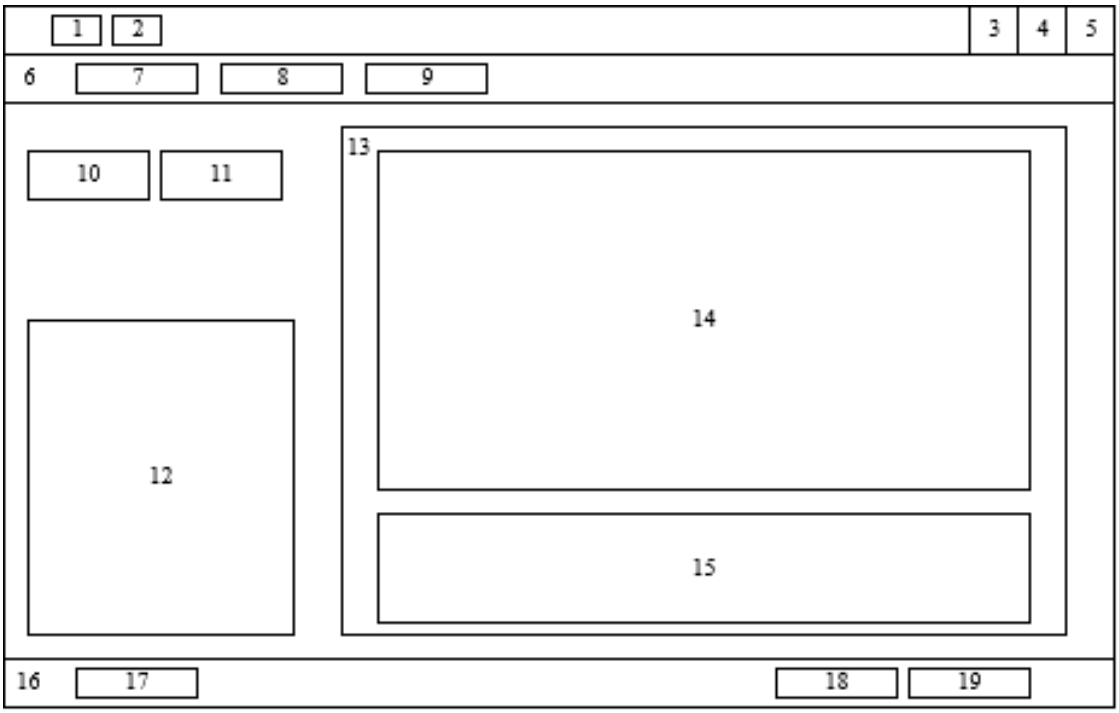

Рисунок 3.6 – Структурна схема головного вікна системи

Структурна схема головного вікна системи зображено на рисунку 3.6.

Відповідно, на вказаній схемі розміщенні наступні елементи:

1. Системний значок.
2. Назва програмного додатку.
3. Кнопка «Згорнути».
4. Кнопка «Розгорнути».
5. Кнопка «Закрити».
6. Панель меню.
7. Елемент меню «База даних…».
8. Елемент меню «Експорт в XML…».

9. Елемент меню «Опції».

10. Кнопка «Запустити обробку усіх зображень».

11. Кнопка «Обрати зображення…»

12. Список завантажених зображень. Графічна схема складається з елементів (рисунок 3.7):

12.1. Область з елементами списку.

12.2. Текстове поле з назвою зображення.

12.3. Кнопка «Запустити обробку».

12.4. Кнопка «Зберегти результат в базу даних».

12.5. Кнопка «Зупинити обробку».

13. Область візуалізації кінцевого результату.

14. Область з кінцевим зображенням.

15. Область з результатами обробки.

16. Поле статусу

17. Текст статусу системи.

18. Кількість здійснених ітерацій обробки зображення.

19. Час виконання обробки зображення

На рисунку 3.7 зображена структурна схема списку завантажених зображень. Цей елемент відповідальний за представлення завантажених зображень та застосовується для керування процесом обробки окремо взятого зображення.

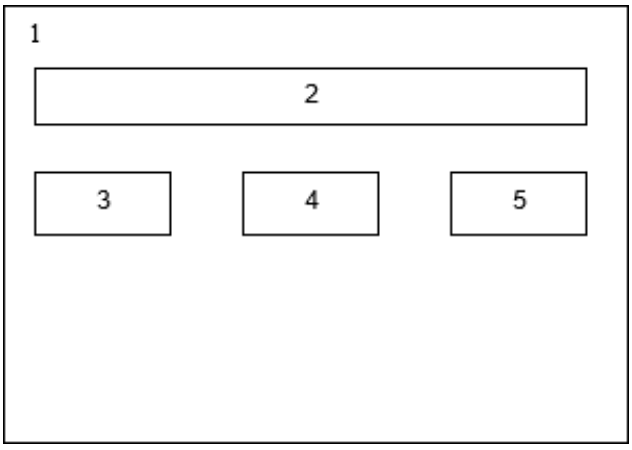

Рисунок 3.7 – Структурна схема списку завантажених зображень

1. Робоча область списку завантажених зображень.
2. Назва файлу зображення.
3. Кнопка «Запустити обробку».
4. Кнопка «Зберегти результат в базу даних».
5. Кнопка «Зупинити обробку».

На рисунку 3.8 зображено структурну схему вікна завантаження зображень. Це вікно відповідальне за завантаження зображень для здійснення процесу їх обробки. Відповідно, користувач може переглянути доступну інформацію про вказане зображення та прийняти рішення про потребу його обробки.

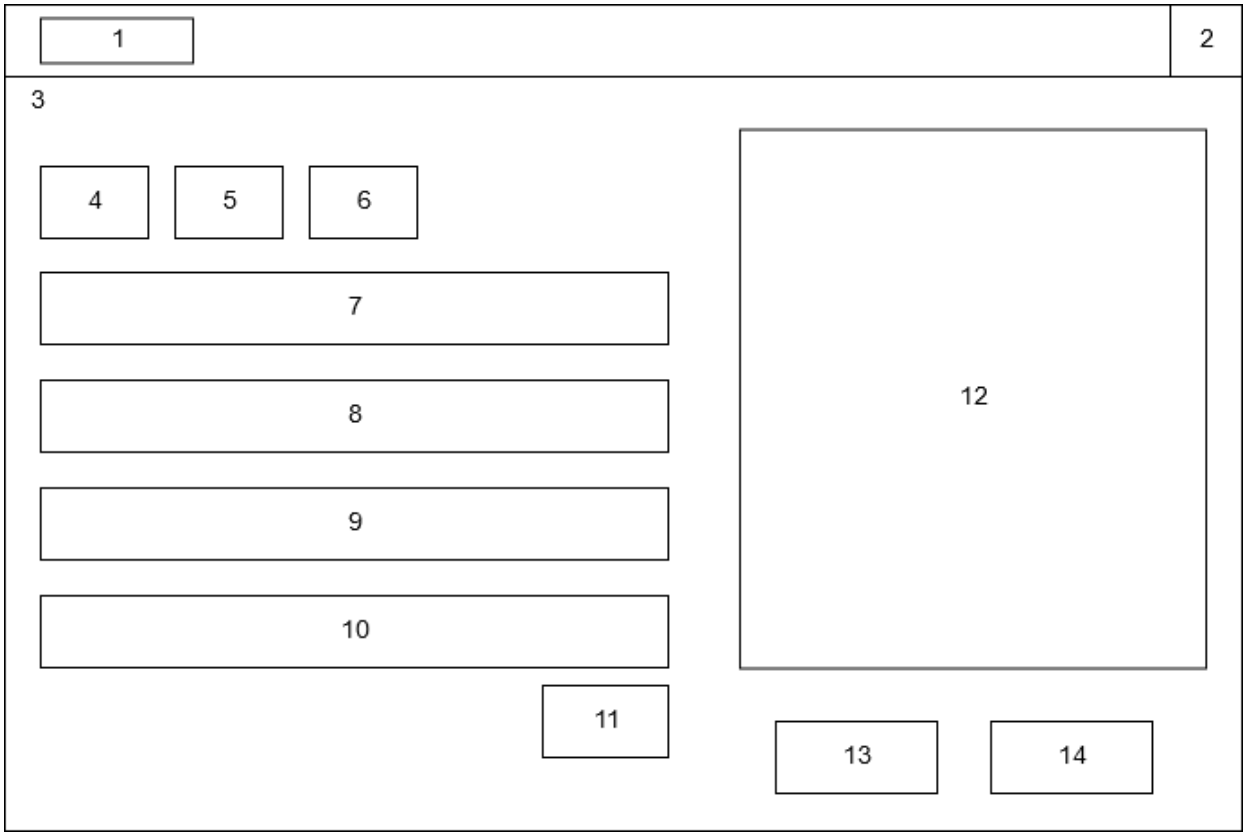

Рисунок 3.8 – Структурна схема вікна завантаження зображень

1. Назва вікна.
2. Кнопка «Закрити вікно».
3. Робоча область вікна завантаження зображень.
4. Кнопка «Попереднє зображення».

5. Порядковий номер поточного зображення.
6. Кнопка «Наступне зображення».
7. Текстове поле з шляхом до зображення.
8. Текстове поле з ім’ям зображення.
9. Текстове поле з роздільною здатністю зображення.
10. Текстове поле з форматом зображення.
11. Кнопка «Видалити поточне зображення».
12. Область попереднього перегляду зображення.
13. Кнопка «Вибрати зображення».
14. Кнопка «Підтвердити завантаження усіх зображень».

На рисунку 3.9 зображено структурну схему вікна взаємодії з базою даних. Це вікно дозволяє переглядати інформації про зображення, що там зберігається, та завантажити його для майбутньої обробки.

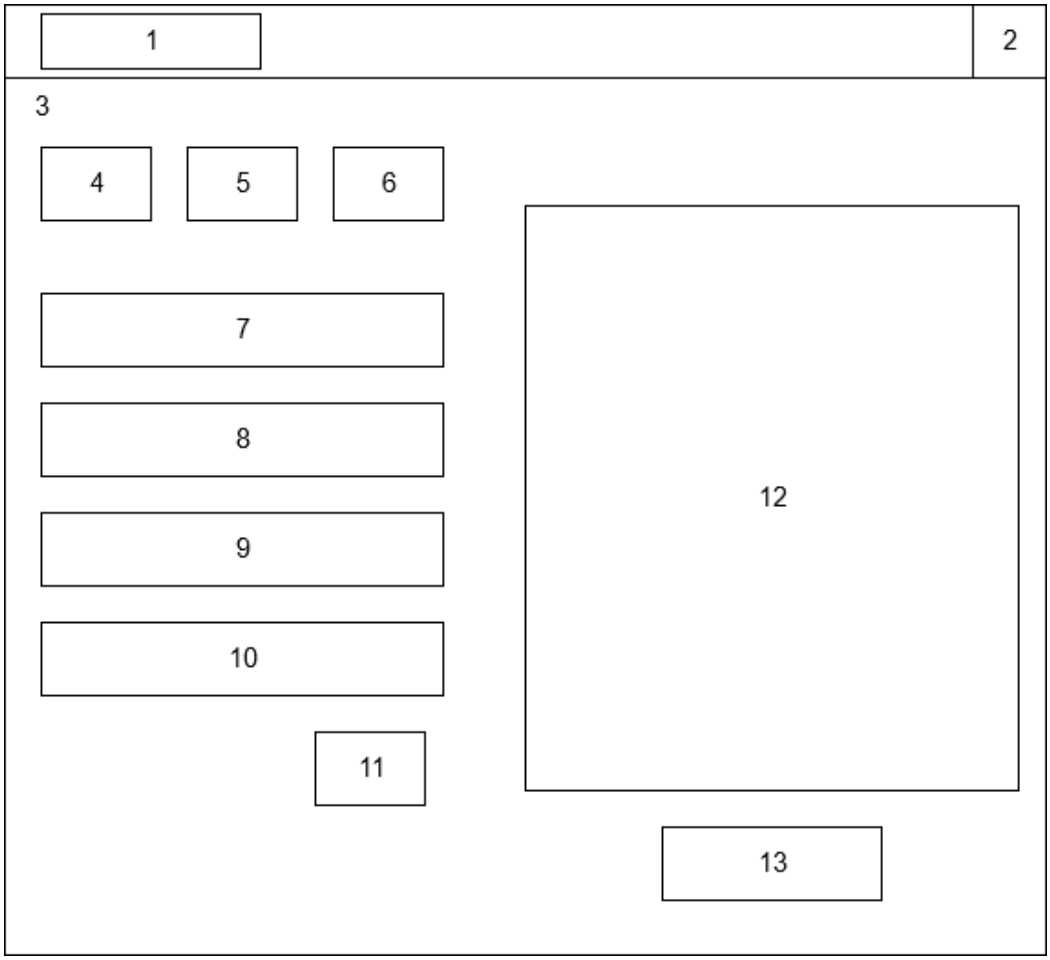

Рисунок 3.9 – Структурна схема вікна взаємодії з базою даних

1. Назва вікна.
2. Кнопка «Закрити вікно».
3. Робоча область вікна взаємодії з базою даних.
4. Кнопка «Попереднє зображення».
5. Порядковий номер поточного зображення.
6. Кнопка «Наступне зображення».
7. Текстове поле з шляхом до зображення.
8. Текстове поле з ім’ям зображення.
9. Текстове поле з роздільною здатністю зображення.
10. Текстове поле з форматом зображення.
11. Кнопка «Вибрати зображення».
12. Область попереднього перегляду зображення.
13. Кнопка «Завантажити зображення».

На рисунку 3.10 зображено структурну схему віна налаштувань параметрів системи. Це вікно дозволяє налаштувати саму системи та процес обробки вхідних зображень.

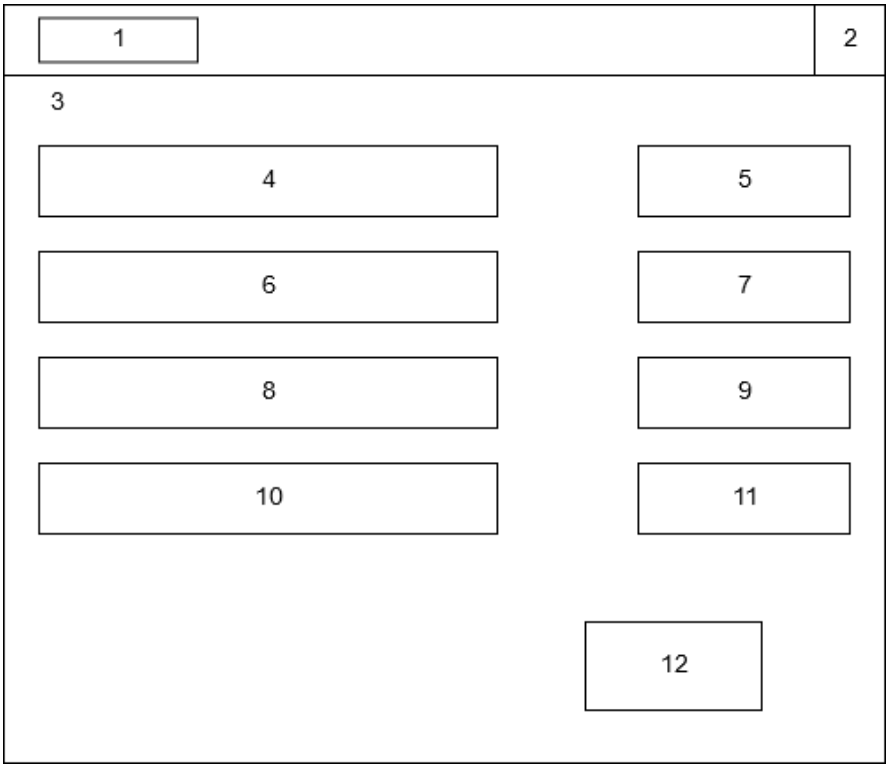

Рисунок 3.10 – Структурна схема вікна налаштувань параметрів системи

1. Назва вікна.
2. Кнопка «Закрити вікно».
3. Робоча область вікна налаштувань параметрів системи.
4. Текстове поле «Кількість шуканих домінантних кольорів».
5. Налаштування параметру «Кількість шуканих домінантних кольорів».
6. Текстове поле« Зменшення розмірів зображення».
7. Налаштування параметру «Зменшення розмірів зображення».
8. Текстове поле «Режим чутливості до яскравих кольорів».
9. Налаштування параметру «Режим чутливості до яскравих кольорів».
10. Текстове поле «Мова системи».
11. Налаштування параметру «Мова системи».
12. Кнопка «Застосувати налаштування».

Отже, було розроблено основні структурні схеми вікон та елементів програмної системи, за допомогою яких, буде здійснюватися налаштування, обробка та візуалізація результатів виконання процесу аналізу вхідних зображень.

3.7 Розробка програмних модулів системи

Розроблена програмна система є виконуваним файлом отриманим в результаті побудови проекту засобами .NET фреймворку. Такий файл також називається збіркою, так як він містить керований код – двійковий код, виконуваний .NET фреймворком та компільований JIT компілятором. Вказана збірка складається з маніфесту, метаданих, інструкцій у форматі MSIL (Microsoft Intermediate Language) та ресурсів.

Так як мова програмування C# є об’єктно орієнтованою мовою та використовує такі елементи як класи та структури, побудуємо діаграму класів для окремих розроблених програмних модулів засобами UML (Unified Modeling Language). Діаграма класів головного вікна системи містить 4 основних елемента – 3 класів та 1 інтерфейс (рисунок 3.11).

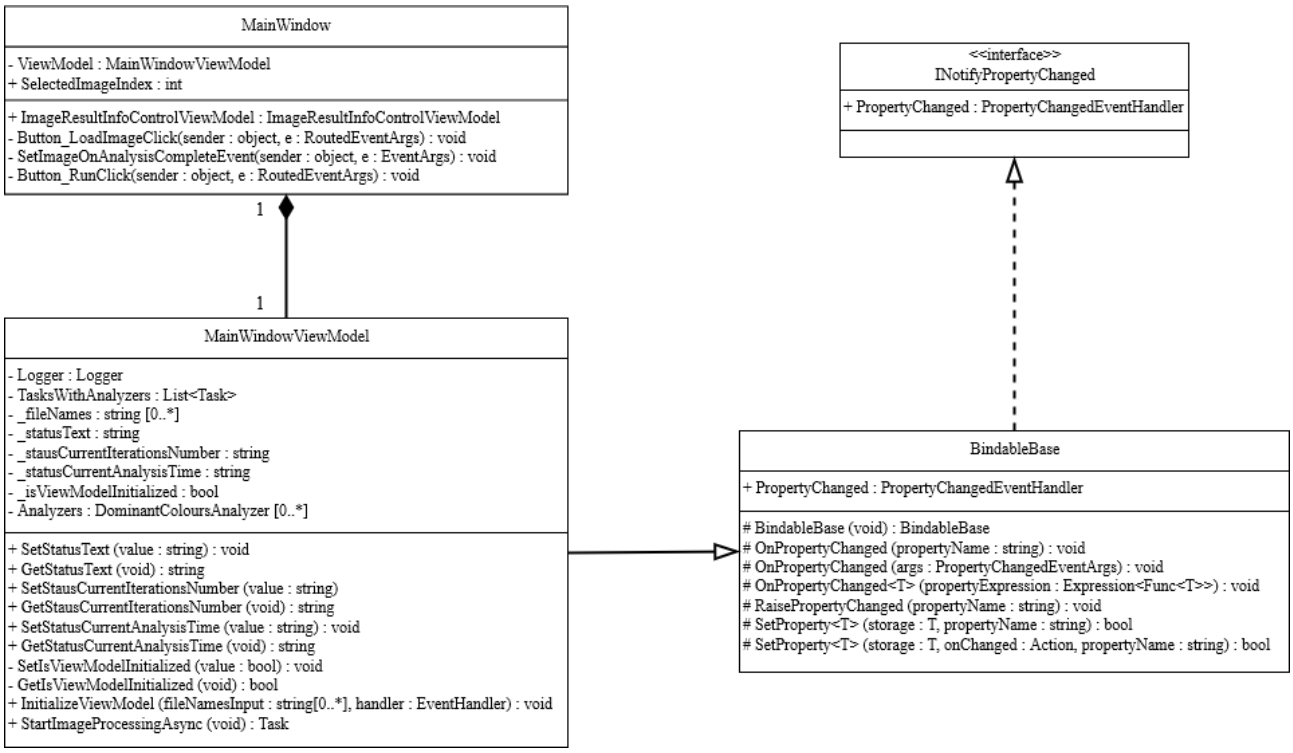

Рисунок 3.11 – Діаграма класів модуля головного вікна системи

Відповідно до обраного архітектурного шаблону, головне вікно має в собі представлення (View) у вигляді класу MainWindow та модель вигляду (ViewModel) – MainWindowViewModel.

Клас представлення містить модель вигляду необхідну для представлення даних, оброблювачі подій та окремі властивості вихідного вікна. Саме цей клас відповідальний за відображення користувацького інтерфейсу.

Модель вигляду – спеціалізований клас, що відповідає за надання даних відповідному елементу представлення (View). Відповідно, має такі властивості як: масив з шляхом до обраних файлів, екземпляри класу аналізу зображень, список виконуваних процесів аналізу та інші допоміжні властивості як логувальник, прапорець ініціалізації тощо. Вказаний клас, також відповідає за розпаралелення процесу обробки зображень, їхній початок та припинення. Це реалізується виконанням відповідних методів класу та використанням вказаних властивостей.

Окрім класів, що відповідають обраному архітектурному шаблону, присутні також клас «BindableBase» та інтерфейс «INotifyPropertyChanged».

Вказаний клас є допоміжним класом, який є відповідальним за реалізацію механізму прив’язки, однієї з фундаментальних особливостей WPF платформи. Використовуючи його, елементи, що знаходяться у представленні, «знають», коли потрібно оновити джерело їхньої інформації і відповідно значення, що відображається користувачу. Цей механізм реалізується за допомогою наявних властивостей та методів, а також, через реалізацію інтерфейсу «INotifyPropertyChanged».

Інтерфейсу «INotifyPropertyChanged» – інтерфейс, що відповідальний за встановлення зв’язку між елементом представлення та елементом моделі вигляду. Його використовують класи, для реалізації вказаного механізму.

Відповідно, вказані класи та інтерфейси перебувають в наступних зв’язках:

- клас «MainWindow» та клас «MainWindowViewModel» мають зв’язок типу композиція, так як головне вікно може мати лише одну модель представлення і вказана модель може належати лише одному вікну і її тривалість «життя» залежить від вказаного вікна;
- клас «MainWindow» та клас «BindableBase» мають зв’язок типу наслідування, де головне вікно наслідує клас «BindableBase»;
- клас «BindableBase» та інтерфейс «INotifyPropertyChanged» мають зв’язок типу реалізація.

Окрім діаграми класів головного вікна, розглянемо діаграму класів модуля аналізатора зображень (рисунок 3.12).

Вказаний модуль відповідає за здійснення колористичної обробки вхідних зображень та формування результату на основі отриманих даних. Окрім цього, цей програмний модуль відповідальний за здійснення візуалізації кластеризації вихідного зображення для подальшого використання в програмній системі та за здійснення фільтрації кольорів вхідних кластерів для підвищення точності отриманих результатів.

Відповідно, діаграма класів модуля аналізатора зображень містить 5 класів та 1 інтерфейс.

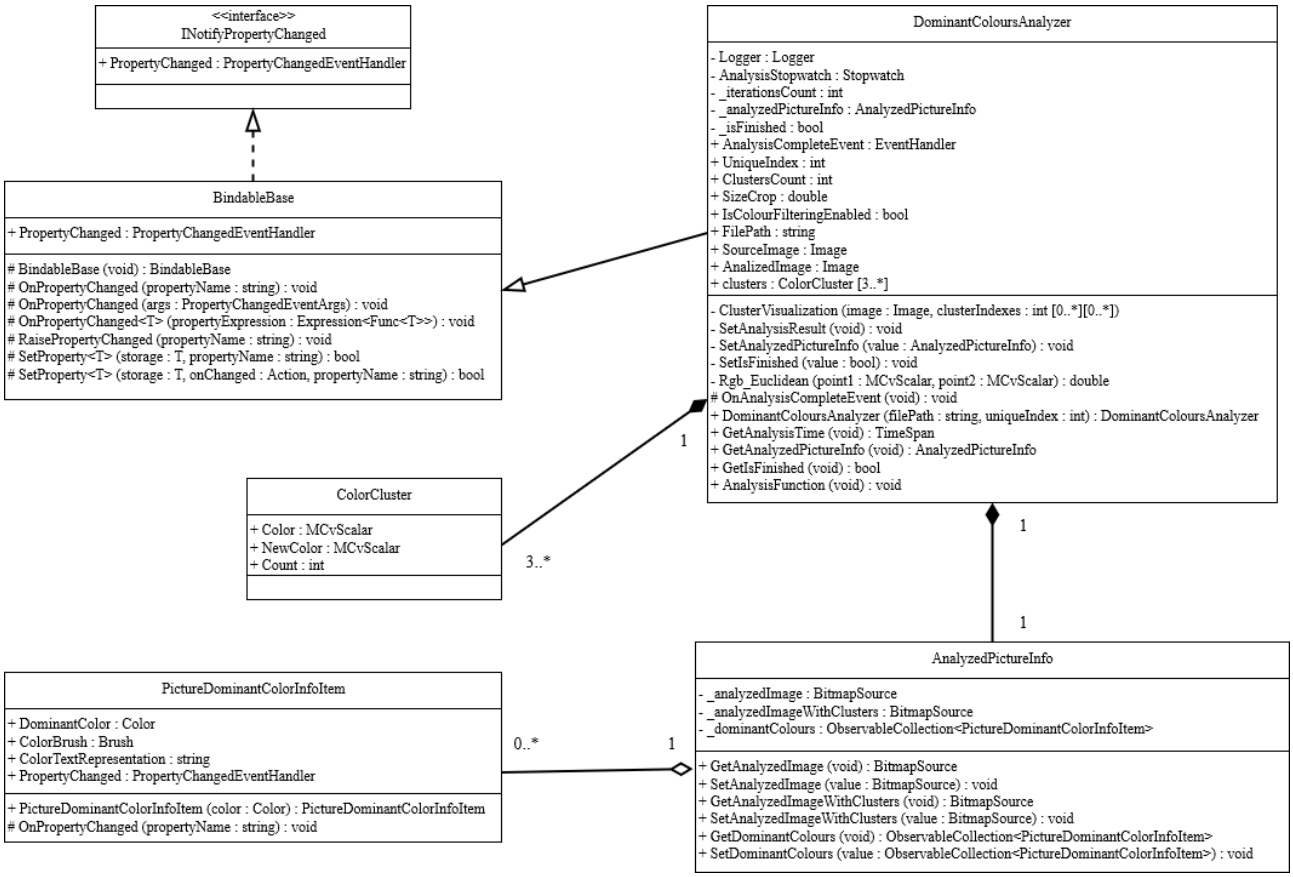

Рисунок 3.12 – Діаграма класів модуля аналізатора зображень

Клас «DominantColoursAnalyzer» відповідальний за здійснення колористичної обробки зображень. Для цього, в якості властивостей він має такі параметри як: час обробки, кількість ітерацій обробки, кількість кластерів, оброблене зображення та інші. Окрім цього, вказаний клас містить метод для формування кінцевого результату обробки, метод візуалізації кластеризації, метод здійснення обробки зображення та методи доступу до властивостей класу.

Клас «ColorCluster» є класом для зберігання інформації про окремий кластер, що задіяний при обробці зображення.

Клас «AnalyzedPictureInfo» є класом для зберігання отриманих зображень – початкового, що був завантажений системою, та кінцевого з позначеними кластерами на ньому. Також цей клас містить інформації про домінантні кольори, які були отримані під час обробки.

Клас «PictureDominantColorInfoItem» є класом, який містить детальну інформації про отримані домінантні кольори.

Клас «BindableBase» та інтерфейс «INotifyPropertyChanged» виконують аналогічну задачу, що й у випадку з головним вікном – забезпечують реалізацію механізму прив’язки WPF платформи.

Відповідно, вказані класи та інтерфейси перебувають в наступних зв’язках:

- клас «DominantColoursAnalyzer» та клас «ColorCluster» мають зв’язок типу композиція, де однин окремо взятий екземпляр класу «DominantColoursAnalyzer» містить не менше трьох екземплярів класу «ColorCluster», тривалість «життя» котрих, залежить від вказаного екземпляру;
- клас «DominantColoursAnalyzer» та клас «AnalyzedPictureInfo» мають зв’язок типу композиція, де одному обробнику зображень належить лише один набір кінцевих результатів й де вказаний набір належить лише одному екземпляру обробника та тривалість «життя» якого, залежить від останнього;
- клас «DominantColoursAnalyzer» та клас «BindableBase» мають зв’язок типу наслідування, де клас обробника зображень наслідує клас «BindableBase»;
- клас «BindableBase» та інтерфейс «INotifyPropertyChanged» мають зв’язок типу реалізація;
- клас «AnalyzedPictureInfo» та клас «PictureDominantColorInfoItem» мають зв’язок типу агрегація, де один екземпляр кінцевого набору результатів обробки зображення може містити декілька екземплярів класу, що характеризують отримані домінантні кольори, а може не містити жодного.

Окрім діаграми класів окремих програмних модулів розробленої системи, використовуючи засоби уніфікованої мови моделювання (UML), побудуємо діаграму компонентів програмної системи (рисунок 3.13).

Діаграма компонентів – це діаграма UML, що відображає зв’язки та залежності між розробленими компонентами системи. Вказана залежність та зв’язок передбачають, що один з компонентів надає необхідні послуги іншому компоненту.

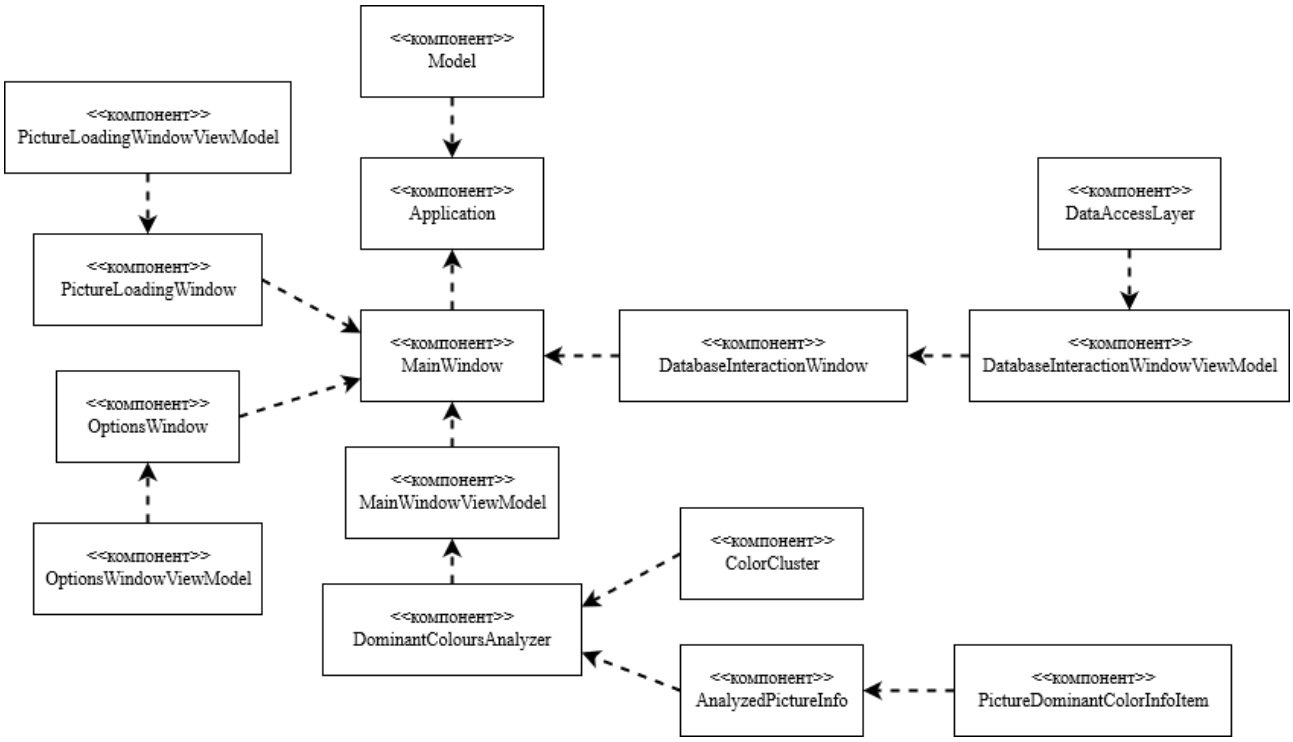

Рисунок 3.13 – Діаграма компонентів розробленої системи

Розроблена діаграма (рисунок 3.13) представляє програмні модулі системи та відображає зв’язки між ними і всього нараховує 15 компонентів.

Кожен компонент на діаграмі – розроблений програмний модуль.

Вказана діаграма складається з програмного модуля «Application», який представляє систему, до якої прив’язана загальна модель та програмний модуль головного вікна. В свою чергу, до головного вікна прив’язані усі інші вікна системи, через які здійснюється взаємодія з такими розробленими компонентами як: база даних, оброблювач зображень та моделі представлення.

Отже, було побудовано діаграми класів для програмного модуля головного вікна системи та для модуля аналізатора зображень. Відображено зв’язки між класами, кратність (потужність зв’язків), їхні властивості та методи. Окрім цього, було побудовано діаграму компонентів розробленої системи, що відображає зв’язки та залежності між окремими компонентами останньої. Побудова діаграм була виконана використовуючи уніфіковану мову моделювання (UML).

3.8 Розробка програмного модуля візуалізації кластеризації зображення та паралельної обробки набору вхідних зображень

Для початку, розглянемо модуль «DominantColoursAnalyzer» (рисунок 3.14). Цей модуль містить в собі необхідні параметри для здійснення оброки зображення, логувальник та методи, що відповідають за процес обробки та візуалізації кластерів на зображені.

```
5 references
public class DominantColoursAnalyzer : BindableBase
{
    2 references
    private Logger Logger { get; } = LogManager.GetCurrentClassLogger();

    1 reference
    public DominantColoursAnalyzer(string filePath, int uniqueIndex)...

    public event EventHandler AnalysisCompleteEvent;
    2 references
    public int UniqueIndex { get; set; }

    #region Analysis setup properties

    0 references
    public int ClustersCount { get; set; }
    0 references
    public double SizeCrop { get; set; }
    0 references
    public bool IsColourFilteringEnabled { get; set; }
    2 references
    public string FilePath { get; private set; }

    #endregion

    5 references
    private Stopwatch AnalysisStopwatch { get; set; }
    3 references
    public int IterationsCount { get; private set; }
    4 references
    public TimeSpan AnalysisTime...
    private AnalyzedPictureInfo _analyzedPictureInfo;
    2 references
    public AnalyzedPictureInfo AnalyzedPictureInfo...
    private bool _isFinished;
    2 references
    public bool IsFinished...
    13 references
    public Image<Bgr, Byte> SourceImage { get; set; }
    4 references
    public Image<Bgr, Byte> AnalizedImage { get; set; }
    public ColorCluster[] clusters;
    public const int ClusterNumber = 10;

    2 references
    private static double Rgb_Euclidean(MCvScalar point1, MCvScalar point2)...
    1 reference
    public void AnalysisFunction()...
    1 reference
    private void ClusterVisualization(Image<Bgr, Byte> image, int[,] clusterIndexes)...
    1 reference
    private void SetAnalysisResult()...
    1 reference
    protected virtual void OnAnalysisCompleteEvent()...
}
```

Рисунок 3.14 – Програмний модуль «DominantColoursAnalyzer»

Для виконання обробки зображення використовується функція «AnalysisFunction». Ця функція містить в собі реалізацію розробленого методу колористичної обробки зображень. Для того, щоб отримати змогу візуалізувати належність окремих елементів дослідження (пікселів зображення) до певного

кластера, перед виконання обробки, створюється двовимірний масив для зберігання цілих чисел, кожне таке число – це індекс кластера обробника. Слід зауважити, що вказаний масив має розміри зображення, що підлягає обробці, а кожен збережений індекс – відповідає координатам відповідного пікселя на зображенні, тим самим зв'язуючи окремий піксель зображення з відповідним кластером. В свою чергу, масив заповнюється відповідними індексами під час виконання обробки зображення (рисунок 3.15).

```
            for (k = 0; k < ClusterNumber; k++)
            {
                double euclid = Rgb_Euclidean(new MCvScalar(B, G, R),
                    new MCvScalar(this.clusters[k].Color.V0, this.clusters[k].Color.V1, this.clusters[k].Color.V2));

                if (euclid < minRgbEuclidean)
                {
                    minRgbEuclidean = euclid;
                    clusterIndex = k;
                }
            }
            // set cluster index
            clusterIndexes[y, x] = clusterIndex;

            this.clusters[clusterIndex].Count++;
            this.clusters[clusterIndex].NewColor = new MCvScalar(this.clusters[clusterIndex].NewColor.V0 + B,
                 this.clusters[clusterIndex].NewColor.V1 + G,
                 this.clusters[clusterIndex].NewColor.V2 + R);
        }
    }
```

Рисунок 3.15 – Отримання індексі для візуалізації кластерів

Після закінчення обробки зображення, на виході утворюється двовимірний масив, кожен елемент, якого відповідає індексу відповідного кластера.

Наступним кроком є перенесення отриманих індексів кластерів на вихідне зображення.

Для цього, в розробленому класі присутній метод «ClusterVisualization».

Саме цей метод відповідальний за візуалізації отриманих кластерів на вихідному зображені. В якості параметрів він приймає копію оригінального вхідного зображення та отриманий масив з індексами кластерів. Наступним кроком, використовуючи доступ до вмісту зображення через властивість «Data», здійснюється встановлення значення колірних каналів окремо взятих пікселів (рисунок 3.16).

```
1 reference
private void ClusterVisualization(Image<Bgr, Byte> image, int[,] clusterIndexes)
{
    for (int y = 0; y < image.Height; y++)
    {
        for (int x = 0; x < image.Width; x++)
        {
            int clusterIndex = clusterIndexes[y, x];

            image.Data[y, x, 2] = (byte)clusters[clusterIndex].Color.V2; //Write to the Red Spectrum
            image.Data[y, x, 1] = (byte)clusters[clusterIndex].Color.V1; //Write to the Green Spectrum
            image.Data[y, x, 0] = (byte)clusters[clusterIndex].Color.V0; //Write to the Blue Spectrum
        }
    }
}
```

Рисунок 3.16 – Метод візуалізації кластерів зображення

Виконавши візуалізацію кластерів, формується набір вихідних даних обробки – результат, що вміщує в собі інформації про домінантні кольори на зображенні, оригінальне зображення та зображення отримане під час візуалізації кластерів (рисунок 3.17).

```
private void SetAnalysisResult()
{
    var dominantColoursCollection = new ObservableCollection<PictureDominantColorInfoItem>();

    this.clusters = this.clusters.OrderByDescending((colourCluster) => colourCluster.Count).ToArray();

    for (int i = 0; i < this.clusters.Length; i++)
    {
        //Debug.WriteLine(this.clusters[i].Count);

        Color color = Color.FromRgb(
            (byte)this.clusters[i].Color.V2,
            (byte)this.clusters[i].Color.V1,
            (byte)this.clusters[i].Color.V0);

        dominantColoursCollection.Add(new PictureDominantColorInfoItem(color));
    }

    this.AnalyzedPictureInfo = new AnalyzedPictureInfo()
    {
        AnalyzedImage = Utility.ToBitmapSource(this.SourceImage.ToBitmap()),
        AnalyzedImageWithClusters = Utility.ToBitmapSource(this.AnalizedImage.ToBitmap()),
        DominantColours = dominantColoursCollection
    };
}
```

Рисунок 3.17 – Формування результату обробки зображення

Отриманий набір вихідних даних надалі буде застосовуватись для відображення результатів обробки зображення, де, власне, і буде використана візуалізація отриманих кластерів.

Наступним, розглянемо модуль «MainWindowViewModel», який є відповідальним за паралельну обробку зображень.

Цей клас містить набір полів задіяних при встановленні «прив'язки» до елементів вигляду (View) та поля задіяні безпосередньо для запуску асинхроного процесу обробки зображень.

```
2 references
public class MainWindowViewModel : BindableBase
{
    1 reference
    private Logger Logger { get; } = LogManager.GetCurrentClassLogger();

    private List<Task> TasksWithAnalyzers;
    private string[] _fileNames;
    3 references
    public string[] FileNames...
    private string _statusText;
    0 references
    public string StatusText...
    private string _stausCurrentIterationsNumber;
    0 references
    public string StausCurrentIterationsNumber...
    private string _statusCurrentAnalysisTime;
    0 references
    public string StatusCurrentAnalysisTime...
    1 reference
    public bool IsViewModelInitialized { get; private set; }
    8 references
    private DominantColoursAnalyzer[] Analyzers { get; set; }

    1 reference
    public void InitializeViewModel(string[] fileNamesInput, EventHandler handler)...
    1 reference
    public async Task StartImageProcessingAsync()...

}
```

Рисунок 3.18 – Програмний модуль «MainWindowViewModel»

Так поле «Analyzers» представляє масив екземплярів обробників зображень задіяних в роботі системи. Кожен елемент цього масиву представляє окремий обробник з унікальним для нього зображенням над яким, власне, і виконується обробка.

Для того, щоб розпаралелити процес обробки зображень, вказаний клас містить поле TasksWithAnalyzers. Це поле є списком, що містить екземпляри класу «Task», який є частиною мови програмування C#, та який застосовується

для асинхронного програмування. Кожен екземпляр наведеного класу, відповідальний за виконання обробки відповідного зображення. Цей процес здійснюється шляхом виклику методу початку обробки зображення екземпляром обробника всередині зазначеного класу (рисунок 3.19).

```
public async Task StartImageProcessingAsync()
{
    try
    {
        for (int i = 0; i < this.Analyzers.Length; i++)
        {
            int index = i;
            this.TasksWithAnalyzers.Add(Task.Run(() => this.Analyzers[index].AnalysisFunction()));
        }

        // Raise exception if any
        await Task.WhenAny(this.TasksWithAnalyzers);
    }
    catch (Exception ex)
    {
        this.Logger.Error(ex, "Error occurred in tasks with analyzers", null);

        throw;
    }
}
```

Рисунок 3.19 – Метод асинхронної обробки зображень

Для того, щоб процес обробки почався для правильного обробника, створюється локальний індекс, котрий потім захоплюється асинхронним потоком і застосовується для виклику відповідного обробника.

Усі створені завдання (екземпляри класу «Task») додаються в список «TasksWithAnalyzers» для зручнішого оперування та обробки потенційних помилок. При настанні моменту, коли обробник закінчить аналіз зображення, буде викликано метод «WhenAny» в комбінації з ключовим словом «await».

Це потрібно для того, щоб виявити можливі помилки, які виникли в інших потоках при обробці зображення і записати відповідну інформацію в файл для логування.

Після того як процес обробки завершиться, буде викликано відповідну подію (Event), що повідомить систему про те, що необхідно оновити візуальну частину отриманими даними.

Вказаний процес здійснюється шляхом виконання методу, який був підписаний на вказану подію (рисунок 3.20).

```
2 references
private void SetImageOnAnalysisCompleteEvent(object sender, EventArgs e)
{
    if (!(sender is DominantColoursAnalyzer analyzer))
    {
        return;
    }

    analyzer.AnalysisCompleteEvent -= SetImageOnAnalysisCompleteEvent;

    if (this.SelectedImageIndex != analyzer.UniqueIndex)
    {
        return;
    }

    this.Dispatcher.Invoke(() => this.ImageResultInfoControlViewModel.ImageResultInfo = analyzer.AnalyzedPictureInfo);
}
```

Рисунок 3.20 – Метод відображення отриманого зображення

Отже, в цьому підрозділі, було розроблено окремі програмні модулі системи відповідальні за паралельну обробку вхідних зображень та візуалізацію кластеризації отриманих результатів, описано основні кроки виконання вказаних процесів та їхні особливості.

3.9 Висновки

Отже, у третьому розділі було проведено варіативний аналіз та обґрунтування вибору програмних засобів, було здійснено вибір середовища програмування, розроблено загальну структуру системи. Також було розроблено універсальнф відношення, ER-модель предметної області, виконано нормалізацію відношень шляхом декомпозиції, побудовано діаграму функціональної залежності. У результаті цього, було отримано структуру бази даних розроблювальної системи. Окрім цього, було визначено структуру графічного інтерфейсу, розроблені структурні схеми вікон та їхніх елементів, побудовано діаграми класів та компонентів, розроблено програмні модулі що забезпечують асинхронне виконання обробки зображень та візуалізують кластеризацію вихідного зображення.

4 ТЕСТУВАННЯ СИСТЕМИ

4.1 Методики тестування

Тестування програмного забезпечення – це процес технічного дослідження [19], призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Тестова діяльність, що пов'язана з аналізом результатів розробки програмного забезпечення, називається статичним тестуванням. Воно передбачає перевірку програмних кодів, контроль та перевірку програми без запуску на комп’ютері. Тестова діяльність, що передбачає експлуатацію програмного продукту, називається динамічним тестуванням.

Існують такі методики тестування, як тестування за стратегією «білого ящика» та тестування за стратегією «чорного ящика».

Тестування за стратегією «білого ящика» [20], дозволяє досліджувати внутрішню структуру програми. В цьому випадку тестувальник отримує тестові дані шляхом аналізу логіки програми. Об’єктом тестування тут є не зовнішня, а внутрішня поведінка програми. Стратегія «білого ящика» включає такі методи тестування: покриття рішень, покриття умов, покриття рішень та умов.

Тестування за стратегією «чорного ящика» дозволяє перевірити роботу кожної функції на своїй області визначення [21]. При використанні цієї стратегії програма розглядається як чорний ящик. Таке тестування має на меті з’ясування обставин, в яких поведінка програми не відповідає її специфікації. Розглядаються системні характеристики програмного додатку, ігнорується їхня внутрішня логічна структура.

Тестування за стратегією «чорного ящика» забезпечує пошук некоректних функцій, помилок інтерфейсу, в доступі до зовнішніх баз даних, помилок ініціалізації та завершення.

Відповідно до вказаних переваг, для тестування програмного додатку було обрано метод «чорного ящика».

4.2 Тестування програмного модуля

Тестування програмного модуля проводиться за допомогою методики «чорної скриньки». Вона базується на використанні шаблонів тестування (тест-кейсів) [22]. Це означає, що буде створено декілька тест-кейсів для перевірки правильності роботи основних функцій розробленої програмної системи.

Тест-кейс №1. Завантаження декількох зображень.

Кроки:

1. Запустити додаток.
2. В головному вікні системи натиснути кнопку «Завантажити зображення».
3. У відкритому вікні натиснути кнопку «Вибрати зображення».
4. З’явиться діалогове вікно вибору зображень де потрібно обрати набір зображень.
5. Після вибору зображень, натиснути кнопку «Обрати» в діалоговому вікні.
6. Переглянути результат у вікні, що з’явиться.

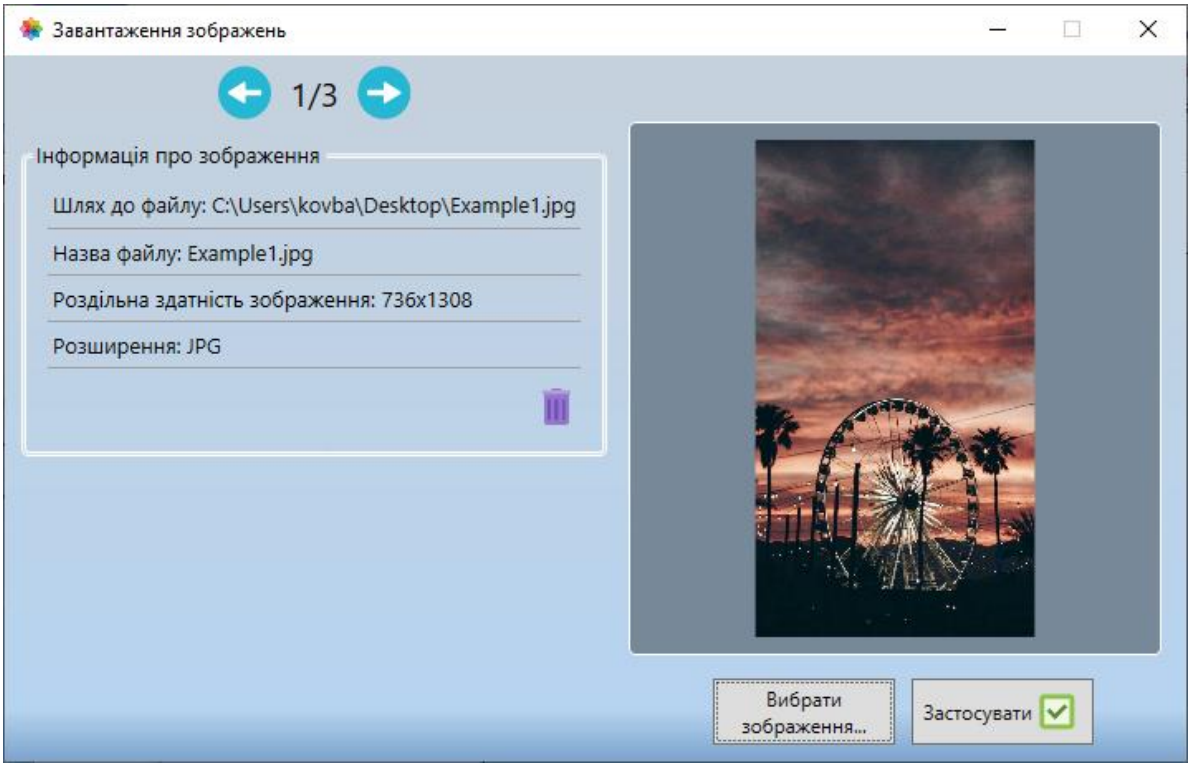

Рисунок 4.1 – Очікуваний результат завантаження зображення

Отриманий результат відповідає очікуваному (рисунок 4.1).

Тест кейс №2. Збереження зображення в базу даних

1. Запустити додаток.
2. Повторити кроки 2-5 з тест кейсу №1.
3. Натиснути кнопку «Застосувати».
4. В головному вікні, натиснути на зображення дискети на відповідному зображенні в списку завантажених зображень
5. Натиснути в головному меню на пункт «База даних».
6. В новому вікні обрати останнє зображення.
7. Переглянути отримані результати.

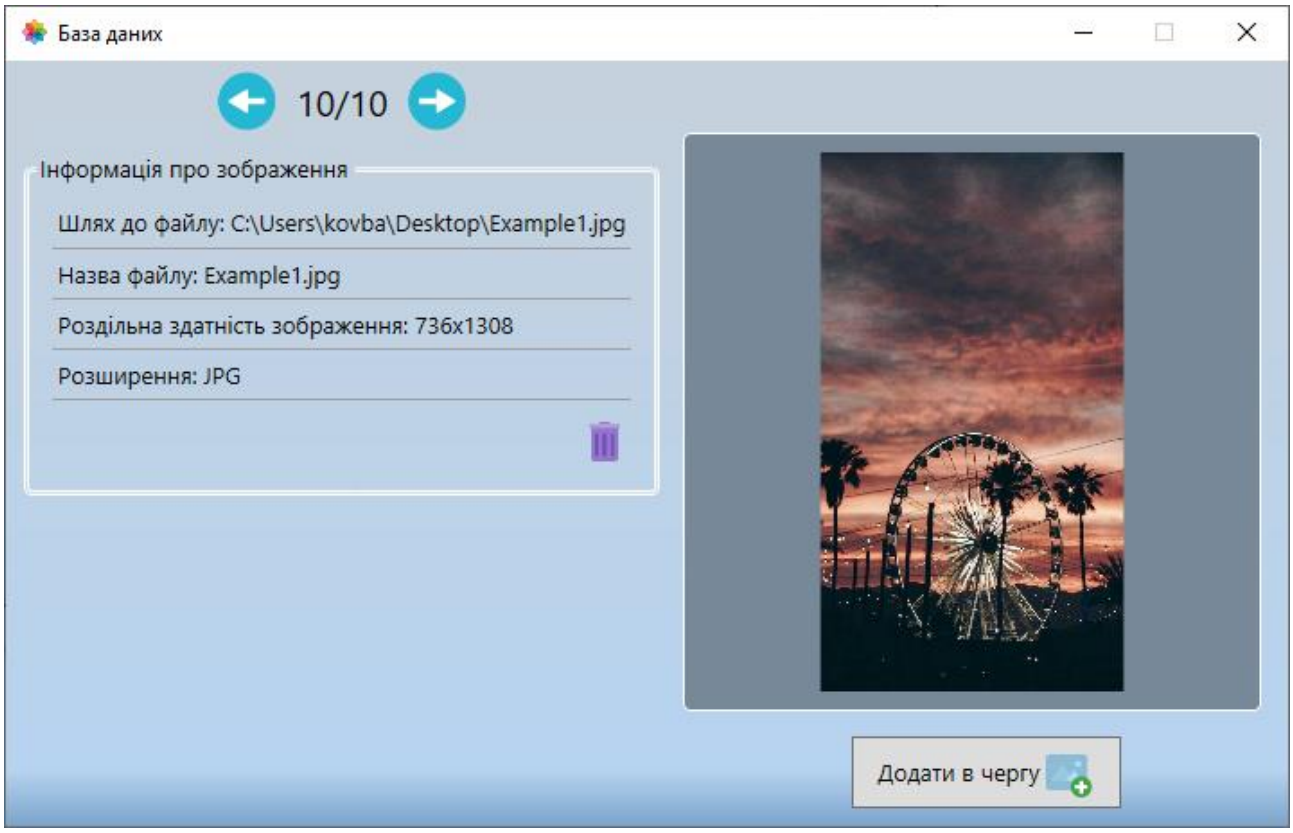

Рисунок 4.2 – Збереження зображення в базу даних

Отриманий результат відповідає очікуваному (рисунок 4.2).

Тест кейс №3. Колористична обробка зображення.

1. Повторити усі кроки з тест кейсу №1.
2. Натиснути кнопку «Застосувати».
3. В головному вікні натиснути кнопку «Запустити все» та очікувати на завершення обробки.
4. Переглянути отриманий результат.

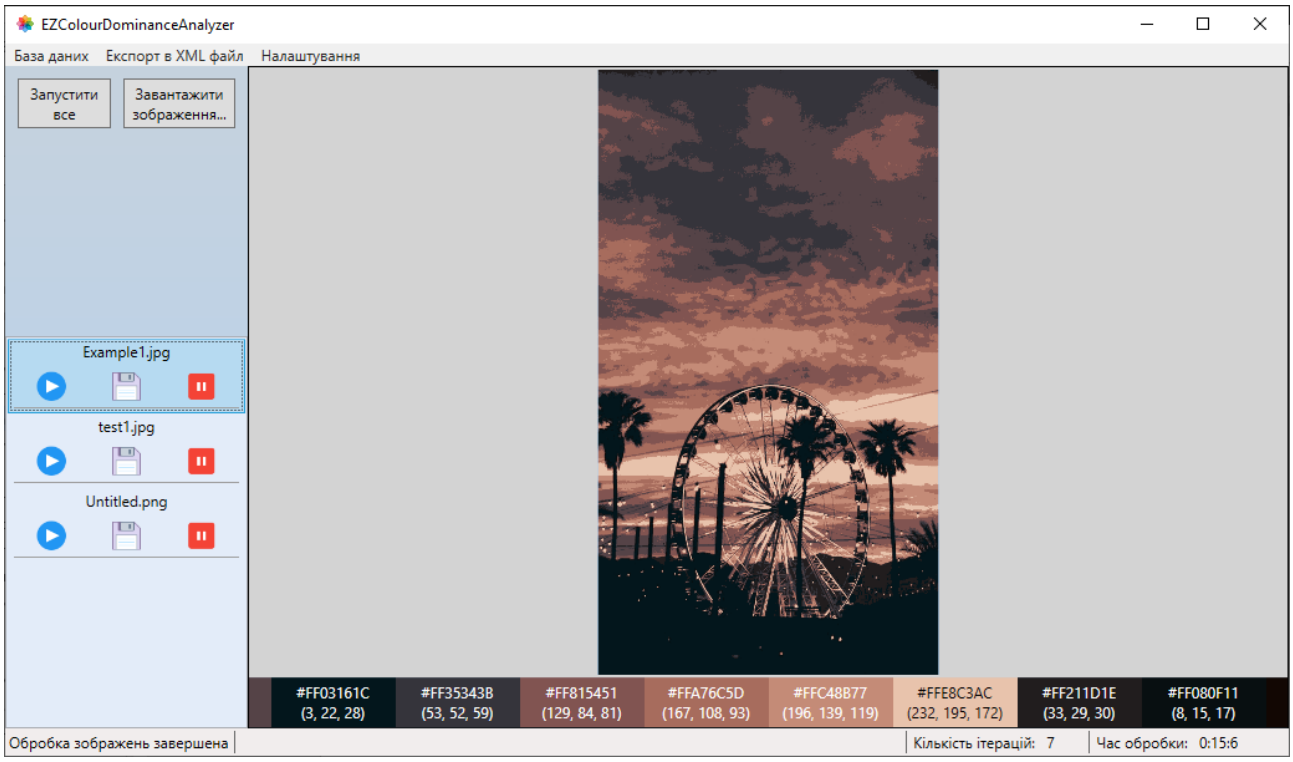

Рисунок 4.3 – Результат роботи обробки зображення

Отриманий результат відповідає очікуваному (рисунок 4.3).

Відповідно усі тести-кейси були завершені успішно.

4.3 Інструкція користувача програмного модуля

Для того, щоб мати змогу використовувати розроблену систему, необхідно, щоб активний користувач мав відповідний доступ до необхідних зображень, для їхнього успішного завантаження в систему.

Для того, щоб завантажити вказані зображення необхідно відкрити відповідне вікно «Завантаження зображень» та використовуючи діалогове вікно обрати одне або декілька зображень (рисунок 4.4).

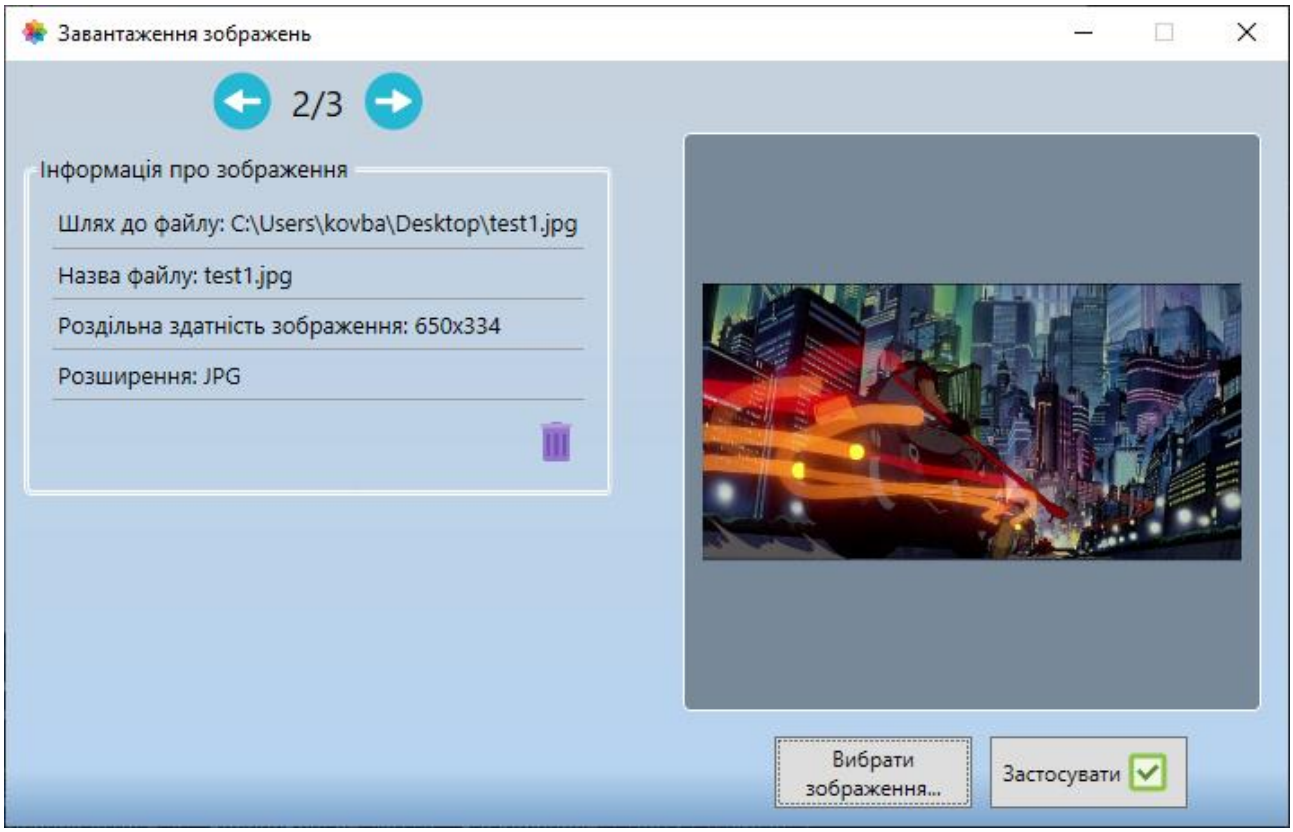

Рисунок 4.4 – Завантаження набору зображень

Однією з переваг розробленої системи є саме можливість одночасної обробки набору вхідних зображень. Після того, як було завантажено вибірку, окремі зображення можна видалити натиснувши на кнопку зі сміттєвим баком. Вказана кнопка лише видалить поточне зображення з завантаженої вибірки, усі інші зображення залишаться в системі де над ними можна буде виконати процес колористичного аналізу.

Для того, щоб перемикатись між завантаженими зображеннями, потрібно натискати на кнопки зі стрілками, що знаходяться вище контейнера з інформацією про зображення. Використовуючи вказані кнопки, можна рухатись в обидва боки вибірки з відображенням поточного індексу.

Після того, як усі дії стосовно завантажених зображень було виконано, необхідно натиснути кнопку «Застосувати», що закриє поточне вікно та повернеться до головного вікна з вже готовою до обробки вибіркою зображень.

Після виконаних дій, головне вікно буде містити одне з завантажених зображень в своїй робочій області (рисунок 4.5).

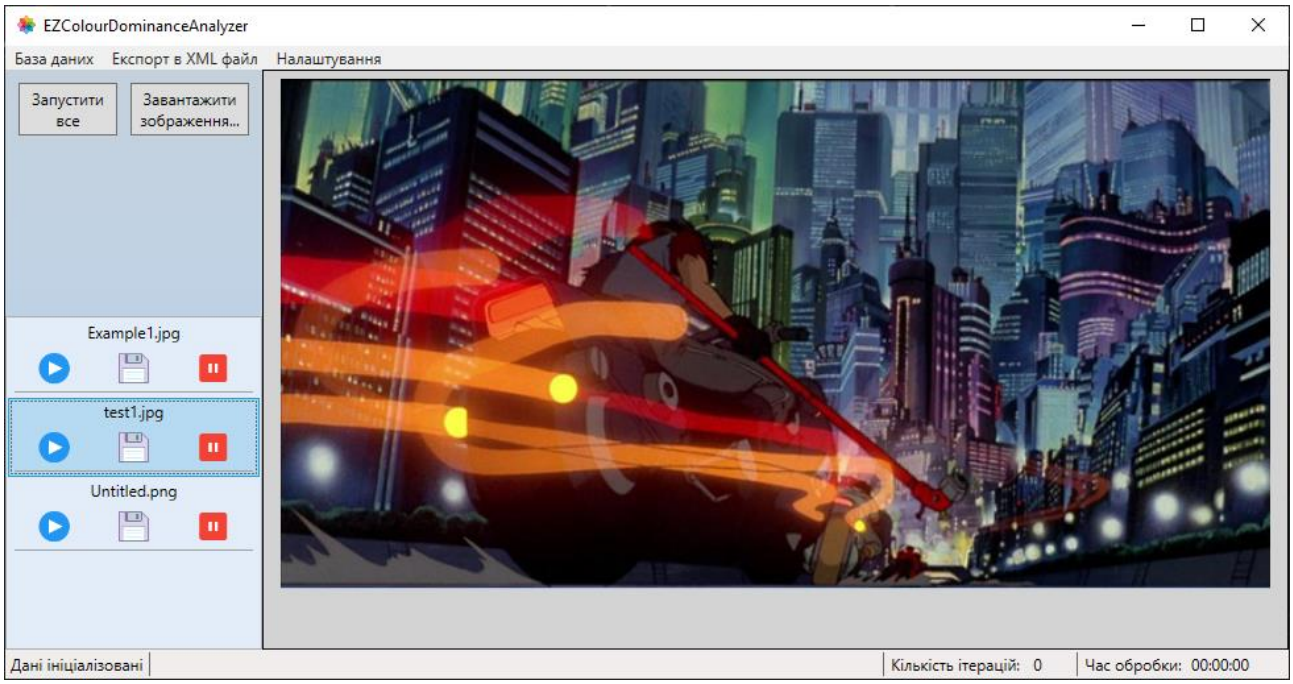

Рисунок 4.5 – Головне вікно з обраним зображенням

Про успішне завантаження вибірки, свідчить відповідне повідомлення в стрічці статусу, що знаходиться внизу головного вікна. В цій самій стрічці також знаходяться такі статистичні дані як кількість виконаних ітерацій алгоритму колористичної обробки зображення та час виконання обробки. Вказані статистичні параметри мають зв’язок лише з обраним зображенням в відповідному списку. Слід зауважити, що перемикаючись між завантаженими зображеннями, статистична інформація буде оновлюватись разом з зображенням, що знаходиться в основній робочій області відповідно до обраного зображення в списку.

Також, в зазначеному списку знаходяться кнопки, які здатні впливати на процес обробки зображення. Кожен елемент списку має три пов’язаних лише до нього кнопки, які здатні самостійно починати обробку або її призупиняти незалежно від будь-якого іншого зображення в списку (на відмінну від кнопки «Запустити все», котра запускає обробки усіх зображень зі списку одночасно). При цьому, існує можливість призупинити процес обробки зображення в будь-який момент шляхом натиснення на відповідну кнопку паузи. Окрім вказаних кнопок, що керують процесом обробки, присутня також кнопка, яка дозволяє

зберегти завантажене зображення в базу даних для його майбутнього опрацювання (рисунок 4.6).

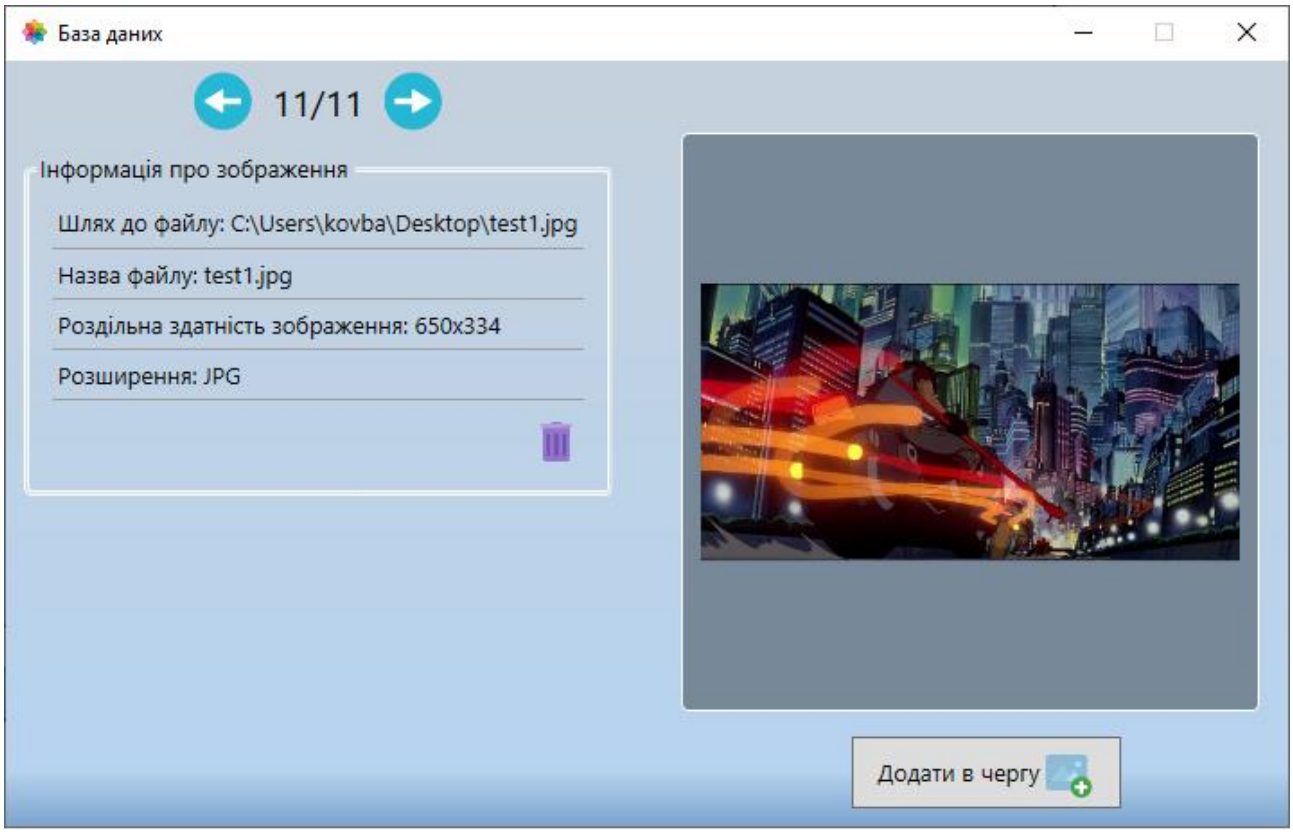

Рисунок 4.6 – Зберігання зображення в базу даних

Після того як вказане зображення було збережене в базі даних, воно отримує власний порядковий індекс, за допомогою якого потім здійснюється вибірка зображень. Кожне таке зображення можна додати в чергу на обробку натиснувши кнопку «Додати в чергу». Це дозволяє звантажувати зображення без потреби у відкритті окремих діалогових вікон.

Окрім цього, програмна система дозволяє налаштувати певні параметри колористичної обробки зображень. Для цього використовується пункт меню «Налаштування» (рисунок 4.7). Натиснувши на вказаний пункт, відкриється вікно з параметрами які можна налаштувати. Для збереження змін потрібно обов’язково натиснути на кнопку «Застосувати» інакше внесені зміни не будуть збережені.

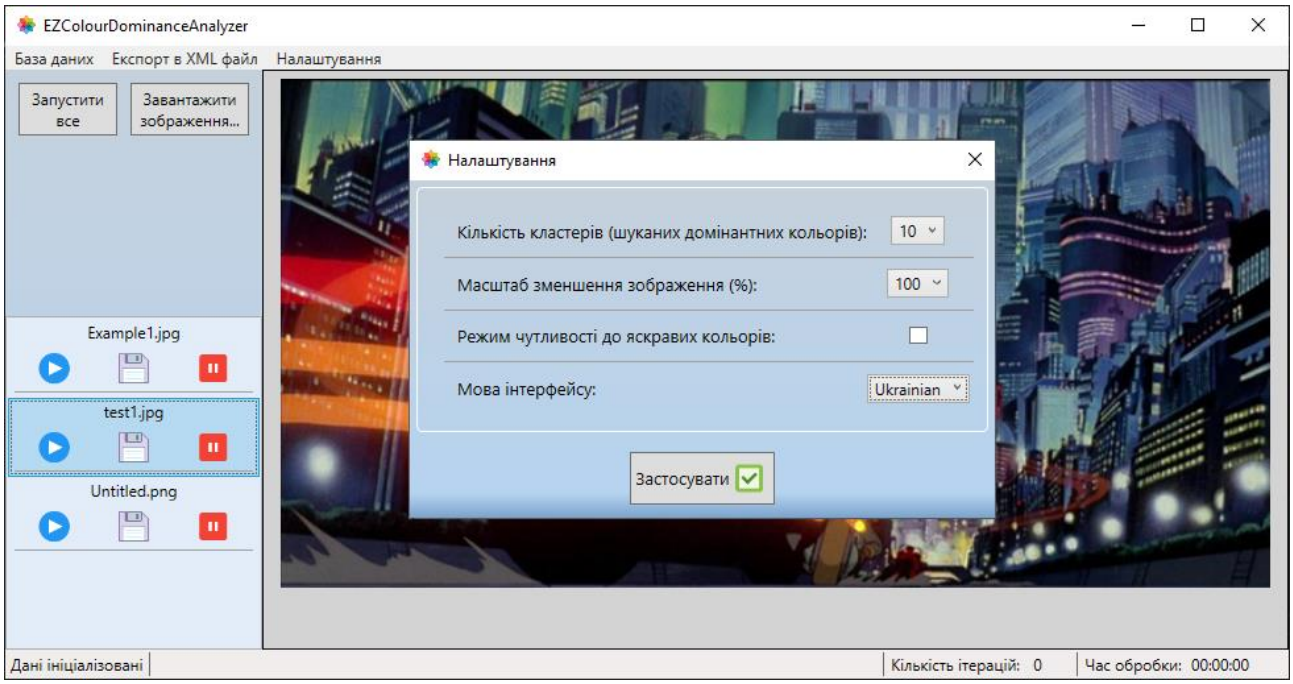

Рисунок 4.7 – Вікно з налаштуваннями системи

Використовуючи вказане вікно, можна, наприклад, змінити кількість шуканих домінантних кольорів на зображені, але є певні обмеження – не можна обрати значення менше 3 або більше 12. Також можна змінити інші параметри обробки як масштаб зменшення чи чутливість до яскравих кольорів. Окрім цього, розроблена система є багатомовною, і підтримує українську та англійську. Відповідно можна змінити мови користувацького інтерфейсу відповідно до потреби користувача.

4.4 Висновки

Отже, було розглянуто методики тестування програмного забезпечення та обрано метод «чорного ящика», так як він відповідає поставленим вимогам для тестування програмних модулів розробленої системи. Крім цього, було розроблено інструкцію користувача та проведено тестування основних функцій програмних модулів системи, що відповідають за завантаження зображень, збереження в базу даних та за виконання колористичної обробки. Кожне тестування показало, що функціонал працює коректно та відповідає загальноприйнятим вимогам у галузі якості програмного забезпечення.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів – Коваленко О. О. та Савчук Т. О.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.1.

Таблиця 5.1

Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали	
	Коваленко О. О.	Савчук Т. О.
	Бали, виставлені експертами:	
1	4	4
2	4	3
3	3	4
4	4	3
5	3	4
6	4	4
7	3	3
8	4	4
9	4	4
10	4	3
11	3	4
12	3	4
Сума балів	$СБ_1 = 44$	$СБ_2 = 44$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 44$	

Отже, з отриманих даних таблиці 5.1 видно, що нова розробка має високий рівень комерційного потенціалу.

5.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько-технологічної роботи

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою 5.1.

$$З_о = \frac{М}{Т_р} \cdot t, \qquad (5.1)$$

де

М – місячний посадовий оклад конкретного розробника;

$Т_р$ – кількість робочих днів у місяці, $Т_р$ = 21 день;

t – число днів роботи розробника, t = 50 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 5.2.

Таблиця 5.2

Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	5500	261,90	5	1309,5
Інженер-програміст	4000	190,47	50	9523,5
Всього:				10833

Розрахуємо додаткову заробітну плату:

$З_{дод}$= 0,1·10833=1083,3 (грн.)

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати (формула 5.2).

$$Н_{зп}= (З_о + З_р)\cdot\frac{\beta}{100} = (10833+1083,3)\cdot\frac{36,3}{100} = 4325,5 \text{ (грн.)} \quad (5.2)$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою 5.3.

$$А=\frac{Ц\cdot Н_а}{100}\cdot\frac{Т}{12}, \quad (5.3)$$

де

Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 5.3

Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн
Персональний комп’ютер	9000	25	3	562,5
Всього:				562,5

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою 5.4.

$$K = \sum_{1}^{n} \text{Н}_\text{і} \cdot \text{Ц}_\text{і} \cdot \text{К}_\text{і}, \quad (5.4)$$

де

n – кількість комплектуючих;

$\text{Н}_\text{і}$ - кількість комплектуючих і-го виду;

$\text{Ц}_\text{і}$ – покупна ціна комплектуючих і-го виду, грн;

$\text{К}_\text{і}$ – коефіцієнт транспортних витрат (приймемо $\text{К}_\text{і}$ = 1,1).

Таблиця 5.4

Витрати на комплектуючі, що були використані для розробки ПЗ

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	180	1	180
Пачка паперу	уп.	120	1	120
Ручка	шт.	10	1	10
Всього з урахуванням транспортних витрат				341

Витрати на силову електроенергію розраховуються за формулою 5.5.

$$\text{В}_\text{е}=\text{В}\cdot\text{П}\cdot\text{Ф}\cdot\text{К}_\text{п}, \quad (5.5)$$

де

В – вартість 1кВт-години електроенергії (В=1,7 грн/кВт);

П – установлена потужність комп’ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп’ютера (Ф=190 год.);

$\text{К}_\text{п}$ – коефіцієнт використання потужності ($\text{К}_\text{п}< 1$, $\text{К}_\text{п} = 0{,}7$).

$$\text{В}_\text{е}= 1{,}7 \cdot 0{,}6 \cdot 190 \cdot 0{,}7 = 135{,}66 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які виконували дану роботу (формула 5.6).

$$В_{ін} = (1..3) \cdot (З_о + З_р) \quad (5.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (10833 + 1083,3) = 11916,3 \text{ (грн.)}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 10833+1083,3+4325,5+562,5+341+135,66+11916,3= 29197,26 (грн.)

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою 5.7.

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (5.7)$$

де

α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{29197,26}{1} = 29197,26$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою 5.8.

$$ЗВ = \frac{В_{заг}}{\beta} \quad (5.8)$$

де

β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{29197,26}{0,9} = 32441,4 \text{ (грн.)}$$

Відповідно до здійснених обрахунків, загальні витрати склали 32441,4 грн.

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою 5.9.

$$\Delta П_i = \sum_{1}^{n} (\Delta П_я \cdot N + П_я \Delta N)_i \quad (5.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 20 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 20 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 200 користувачів, протягом другого року – на 150 користувачів, протягом третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 700 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 200 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 20 \cdot 700 + (200 + 20) \cdot 200 = 58000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 20 \cdot 700 + (200 + 20) \cdot (200 + 150) = 91000 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 20 \cdot 700 + (200 + 20) \cdot (200 + 150 + 100) = 113000 \text{ грн.}$$

Таким чином, було обраховано збільшення чистого прибутку від впровадження результатів протягом перших трьох років.

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - PV),$$

де ПП –вартість чистих прибутків;

PV–теперішня вартість інвестицій.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд графіку (рисунок 5.1).

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_{1}^{m} \frac{\Delta П_i}{(1 + \tau)^t}$$

де

$\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

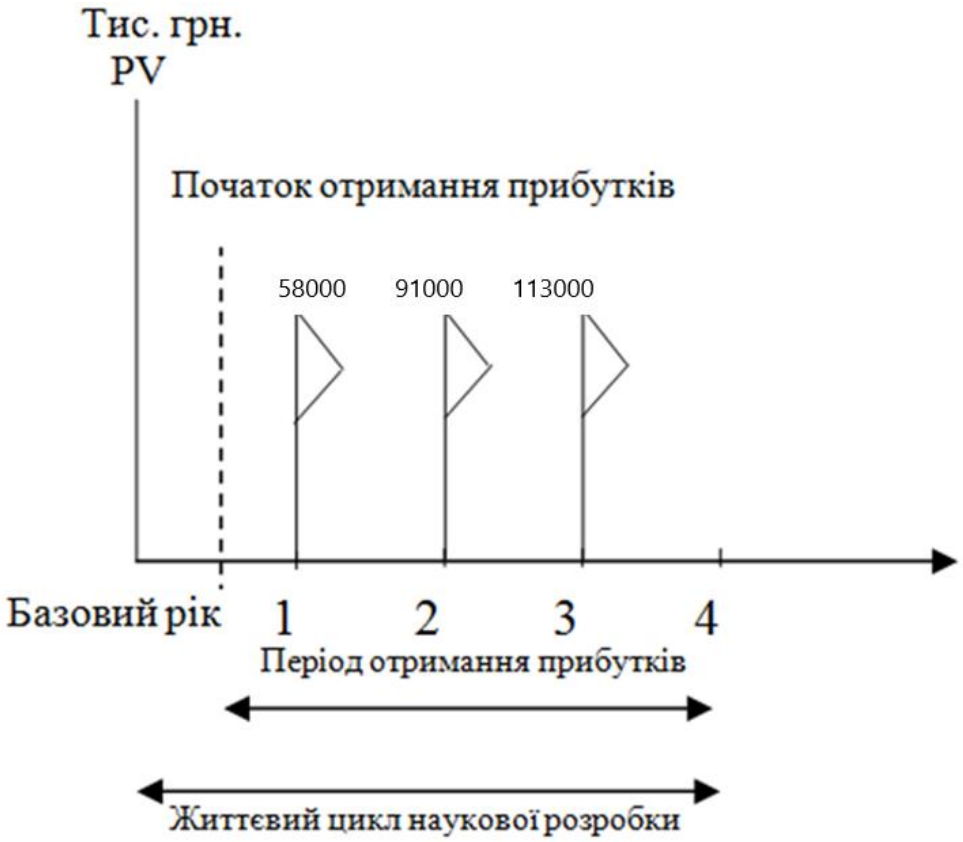

Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{32441,4}{(1+0,1)^0} + \frac{58000}{(1+0,1)^2} + \frac{91000}{(1+0,1)^3} + \frac{113000}{(1+0,1)^4} = 225925,44 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 225925,44 - 32441,4 = 193484,04 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$ за формулою:

$$E_в = \sqrt[T]{1 + \frac{E_{абс}}{PV}} - 1$$

де

$E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$E_в = \sqrt[3]{1 + \frac{193484,04}{32441,4}} - 1 = 0,90 \text{ або } 90\ \%$$

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де

d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в$ = 90% > $\tau_{мін}$ = 0,3 = 30%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в} = \frac{1}{0{,}9} = 1{,}11 \text{ року}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

5.5 Висновки

Отже, було здійснено оцінку комерційного потенціалу розробки за допомогою проведення технічного аудиту з залученням незалежних експертів. Згідно аудиту, нова розробка має високий рівень комерційного потенціалу.

Також було спрогнозовано витрати на виконання науково-дослідної роботи та конструкторсько-технологічної роботи шляхом підрахунку витрат на заробітну платню, вартості використаних комплектуючих та інших видатків, які разом склали 32441,4 гривень.

Окрім цього, булло здійснено прогнозування комерційних ефектів від реалізації результатів розробки, згідно якого в результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться, а кількість користувачів збільшиться.

Окрім цього, був здійснений розрахунок ефективності вкладених інвестицій та періоду їх окупності, згідно якого вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним, а термін окупності вкладених у реалізацію наукового проекту інвестицій складає усього 1.11 року.

ВИСНОВКИ

В ході виконання магістерської кваліфікаційної роботи, було здійснено аналіз переваг та недоліків існуючих методів визначення домінантної палітри кольорів, що підтвердило актуальність розробки власної модифікації на основі вже існуючих методів, виконано порівняльний аналіз аналогів з визначенням переваг та недоліків існуючих програмних реалізацій, що довело необхідність в розробці власної системи, котра б вирішувала недоліки вже існуючих та реалізувала б розроблену модифікацію методу колористичної обробки зображень.

Модифіковано метод кластеризації к-середніх для визначення домінантної палітри кольорів шляхом додання до алгоритму нових кроків, що передбачають зменшення розмірів вхідного зображення та фільтрацію початкових значень кластері, що дало змогу підвищити продуктивність та точність колористичної обробки зображень.

Було проведено аналіз інтегрованих середовищ розробки, варіантний аналіз та обґрунтування вибору програмних засобів для визначення домінантної палітри кольорів, в результаті чого, було обрано середовище розробки Microsoft Visual Studio та такі технології як Windows Presentation Foundation, Microsoft SQL Server, Entity Framework й EmguCV, оскільки вони найбільше відповідають вимогам, що поставленні завданнями магістерської кваліфікаційної роботи.

Розроблено загальну структуру системи, структуру бази даних (в якій було побудовано ER-діаграму, діаграму функціональної залежності та спроектовані нормалізовані відношення), структуру додатку з графічним інтерфейсом на основі використання архітектурного шаблону MVVM, структуру інтерфейсу користувача, діаграми класів й компонентів, окремий програмний модуль для візуалізації кластеризації вихідного зображення та паралельної обробки вхідних даних.

Проведено тестування роботи розробленої програмної системи, яке підтвердило коректність її роботи та відповідність загальноприйнятим вимогам у галузі якості програмного забезпечення.

Виконано оцінювання комерційного потенціалу, спрогнозовано витрати на виконання науково-дослідної роботи, спрогнозовано комерційні ефекти від реалізації результатів розробки що в сукупності доводить окупність цієї наукової розробки та доцільність її фінансування.

Враховуючи все вищезазначене, можна зробити висновок, що задачі магістерської кваліфікаційної роботи були виконанні в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Домінуючий колір. URL: https://uk.wikipedia.org/wiki/Домінуючий_колір (дата звернення: 6.10.2019).

2. Why color matters. URL: https://www.colorcom.com/research/why-color-matters (Last accessed: 5.11.2019).

3. Ковбасюк О.В., Коваленко О.О. Особливості розробки методів і засобів колористичного оброблення зображень на *Міжнарод. наук.-практ. конференції «Інформаційні технології і автоматизація»*, м. Одеса, с. 129-131, 2019.

4. Ковбасюк О.В., Коваленко О.О. Методи та інструментарій колористичного оброблення зображень. Розділ монографії «Інформаційні технології та автоматизація. м. Одеса. ОНАХТ, 2019. URL: https://card-file.onaft.edu.ua/handle/123456789/10179 (дата звернення: 2.12.2019).

5. K-means clustering. URL: https://en.wikipedia.org/wiki/K-means_clustering (Last accessed: 5.11.2019).

6. Кластерний аналіз. URL: https://uk.wikipedia.org/wiki/Кластерний_аналіз (дата звернення: 6.10.2019).

7. IMGonline. URL: https://www.imgonline.com.ua/ (Last accessed: 5.11.2019).

8. TinEye. URL: https://tineye.com/ (Last accessed: 5.11.2019).

9. Rafael C. Gonzales, Richard E. Wood. *Digital Image Processing*, 4th edition, London, 2017, p.: 136-157.

10. Nathans, Jeremy; Thomas, Darcy; Hogness, David S. *Molecular Genetics of Human Color Vision: The Genes Encoding Blue, Green, and Red Pigments*, 1986, Stanford, p.: 193–202.

11. Bruno, Michael H. *Pocket Pal: A Graphic Arts Production Handbook*; 18th edition, 2000, Memphis, p.: 210-229.

12. Windows Presentation Foundatio. URL: https://uk.wikipedia.org/wiki/Windows_Presentation_Foundation (дата звернення: 6.10.2019).

13. Windows Forms. URL: https://uk.wikipedia.org/wiki/Windows_Forms (дата звернення: 6.10.2019).

14. Microsoft SQL Server. URL: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server (дата звернення: 6.10.2019).

15. MySQL. URL: https://uk.wikipedia.org/wiki/MySQL (дата звернення: 6.10.2019).

16. Бази даних. Мови запитів, управління транзакціями, розподілена обробка даних URL: http://posibnyky.vntu.edu.ua/db/index.htm (дата звернення: 6.10.2019).

17. Дейт К. *Введение в системы баз данных.* изд. 7. М.:Вильямс, 2001. – с. 1023-1025.

18. Model-View-ViewModel. URL: https://uk.wikipedia.org/wiki/Model-View-ViewModel (дата звернення: 6.10.2019).

19. Тестування програмного забезпечення. URL: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення (дата звернення: 6.10.2019).

20. White-box testing. URL: https://en.wikipedia.org/wiki/White-box_testing (Last accessed: 5.11.2019).

21. Black-box testing. URL: https://en.wikipedia.org/wiki/Black-box_testing (Last accessed: 5.11.2019).

22. Test case. URL: https://en.wikipedia.org/wiki/Test_case (Last accessed: 7.11.2019).

Додаток А. Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"____ "_____2019 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Розробка методів і засобів колористичного оброблення зображень та системи на їх основі» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____к. т. н., доц. О.О. Коваленко

"____"_____2019 р.

Виконав:

_____студент гр.1ПІ-18м О.В. Ковбасюк

"____"_____2019 р.

Вінниця – 2019 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів і засобів колористичного оброблення зображень та системи на їх основі».

Галузь застосування – обробка зображень, вебдизайн.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення продуктивності та точності пошуку домінантних кольорів при аналізі зображень за рахунок розробки нових та модифікації існуючих методів і засобів колористичного аналізу.

Призначення роботи – розробка методів і засобів колористичної обробки зображень для визначення домінантної палітри кольорів.

4 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Ковбасюк О.В., Коваленко О.О. Особливості розробки методів та засобів колористичного оброблення зображень на *Міжнарод. наук.-практ. конференції «Інформаційні технології і автоматизація»,* м. Одеса, с. 129-131, 2019.

2. Ковбасюк О.В., Коваленко О.О. Методи та інструментарій колористичного оброблення зображень. Розділ монографії «Інформаційні технології та автоматизація. м. Одеса. ОНАХТ, 2019. URL: https://card-file.onaft.edu.ua/handle/123456789/10179.

3. Rafael C. Gonzales, Richard E. Wood. *Digital Image Processing*, 4th edition, London, 2017, p.: 136-157.

4. Nathans, Jeremy; Thomas, Darcy; Hogness, David S. *Molecular Genetics of Human Color Vision: The Genes Encoding Blue, Green, and Red Pigments*, 1986, Stanford, p.: 193–202.

5. Bruno, Michael H. *Pocket Pal: A Graphic Arts Production Handbook*; 18th edition, 2000, Memphis, p.: 210-229.

5. Технічні вимоги

Вихідні дані до роботи: зображення в просторі Truecolor, розмір екрану – 1280x1024, вихідний метод для модифікації – кластеризація методом к-середніх.

6. Конструктивні вимоги.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

- пояснювальна записка до МКР;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	2	3
1	Аналіз предметної області	04.09.2019 – 14.09.2019

1	2	3
2	Розробка модифікації методу кластеризації к-середніх для підвищення ефективності визначення домінуючої палітри кольорів зображення	15.09.2019 – 20.10.2019
3	Розробка програмних засобів реалізації розробленого методу визначення домінуючої палітри кольорів зображення	21.10.2019 – 15.11.2019
4	Тестування системи	16.11.2019 – 21.11.2019
5	Економічна частина	22.11.2019 – 01.12.2019

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б. Лістинг вихідного коду

DominantColoursAnalyzer.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Media.Imaging;
using Emgu.CV;
using Emgu.CV.Structure;
using DominantColoursSearch.CustomClasses;
using Prism.Mvvm;
using System.Collections.ObjectModel;
using System.Windows.Media;
using System.Windows;
using System.Diagnostics;
using NLog;
using Emgu.CV.CvEnum;

namespace DominantColoursSearch.DominantColoursAnalysis
{
   public class DominantColoursAnalyzer : BindableBase
   {
      private Logger Logger { get; } = LogManager.GetCurrentClassLogger();

      public DominantColoursAnalyzer(string filePath, string fileName, int uniqueIndex)
      {
```

```
            this.FilePath = filePath;
            this.FileName = fileName;
            this.UniqueIndex = uniqueIndex;
            this.AnalysisStopwatch = new Stopwatch();

            this.clusters = Utility.GetClusters();
        }

        public event EventHandler AnalysisCompleteEvent;
        public int UniqueIndex { get; set; }

        #region Analysis setup properties

        public int ClustersCount { get; set; }
        public double SizeCrop { get; set; }
        public bool IsColourFilteringEnabled { get; set; }
        public string FilePath { get; private set; }
        public string FileName { get; private set; }

        #endregion

    private Stopwatch AnalysisStopwatch { get; set; }

        public int IterationsCount { get; private set; }

        public TimeSpan AnalysisTime
        {
            get => this.AnalysisStopwatch == null
                ? new TimeSpan(0, 0, 0)
                : this.AnalysisStopwatch.Elapsed;
```

```
}

private AnalyzedPictureInfo _analyzedPictureInfo;
public AnalyzedPictureInfo AnalyzedPictureInfo
{
    get => this._analyzedPictureInfo;
    private set
    {
        if (Object.ReferenceEquals(this._analyzedPictureInfo, value))
        {
            return;
        }

        this._analyzedPictureInfo = value;

        RaisePropertyChanged();
    }
}

private bool _isFinished;
public bool IsFinished
{
    get => this._isFinished;
    private set
    {
        if (this._isFinished.Equals(value))
        {
            return;
        }
```

```
            bool oldValue = this._isFinished;

            this._isFinished = value;

            if (!oldValue && this._isFinished)
            {
                OnAnalysisCompleteEvent();
            }
        }
    }

    public Image<Bgr, Byte> SourceImage { get; set; }

    public Image<Bgr, Byte> AnalizedImage { get; set; }

    public ColorCluster[] clusters;

    public const int ClusterNumber = 10;

    private static double Rgb_Euclidean(MCvScalar point1, MCvScalar
point2)
    {
        return Math.Sqrt((point1.V0 - point2.V0) * (point1.V0 - point2.V0) +
            (point1.V1 - point2.V1) * (point1.V1 - point2.V1) +
            (point1.V2 - point2.V2) * (point1.V2 - point2.V2));
    }

    public void AnalysisFunction()
    {
        this.IsFinished = false;
```

```
            // it's recomennded to call Dispose() method because Image class contains IplImg structure
            this.SourceImage = new Image<Bgr, Byte>(this.FilePath);

            this.SourceImage = this.SourceImage.Resize((int)(this.SourceImage.Width / sizeCrop),
                (int)(this.SourceImage.Height / sizeCrop),
                Inter.Linear);

        int[,] clusterIndexes = new int[this.SourceImage.Height, this.SourceImage.Width];

            int x;
            int y;
            int k;

            double minRgbEuclidean = 0;
            double oldRgbEuclidean = 0;

            this.IterationsCount = 0;
            this.AnalysisStopwatch.Restart();

            while (true)
            {
                this.IterationsCount++;

                for (k = 0; k < ClusterNumber; k++)
                {
                    this.clusters[k].Count = 0;
```

```
            this.clusters[k].Color = this.clusters[k].NewColor;
            this.clusters[k].NewColor = new MCvScalar(0, 0, 0);
        }

        for (y = 0; y < this.SourceImage.Height; y++)
        {
            for (x = 0; x < this.SourceImage.Width; x++)
            {
                // get pixel' RGB components
                int B = (int)this.SourceImage[y, x].Blue;
                int G = (int)this.SourceImage[y, x].Green;
                int R = (int)this.SourceImage[y, x].Red;

                minRgbEuclidean = Double.MaxValue;
                int clusterIndex = -1;

                for (k = 0; k < ClusterNumber; k++)
                {
                    double euclid = Rgb_Euclidean(new MCvScalar(B, G, R),
                        new MCvScalar(this.clusters[k].Color.V0,
this.clusters[k].Color.V1, this.clusters[k].Color.V2));

                    if (euclid < minRgbEuclidean)
                    {
                        minRgbEuclidean = euclid;
                        clusterIndex = k;
                    }
                }
                // set cluster index
                clusterIndexes[y, x] = clusterIndex;
```

```
                this.clusters[clusterIndex].Count++;
                this.clusters[clusterIndex].NewColor = new
MCvScalar(this.clusters[clusterIndex].NewColor.V0 + B,
                    this.clusters[clusterIndex].NewColor.V1 + G,
                    this.clusters[clusterIndex].NewColor.V2 + R);
            }
        }

        minRgbEuclidean = 0;
        for (k = 0; k < ClusterNumber; k++)
        {
            if (this.clusters[k].Count == 0)
            {
                continue;
            }

            // new cluster
            this.clusters[k].NewColor = new
MCvScalar(this.clusters[k].NewColor.V0 / this.clusters[k].Count,
                    this.clusters[k].NewColor.V1 / this.clusters[k].Count,
                    this.clusters[k].NewColor.V2 / this.clusters[k].Count);

    double ecli = Rgb_Euclidean(new MCvScalar(this.clusters[k].NewColor.V0,
this.clusters[k].NewColor.V1, this.clusters[k].NewColor.V2),
                    new MCvScalar(this.clusters[k].Color.V0,
this.clusters[k].Color.V1, this.clusters[k].Color.V2));

            if (ecli > minRgbEuclidean)
            {
```

```
                    minRgbEuclidean = ecli;
                }
            }

            if (Math.Abs(minRgbEuclidean - oldRgbEuclidean) < 1) // basically when they're equal
            {
                break;
            }

            oldRgbEuclidean = minRgbEuclidean;
        }

        this.AnalysisStopwatch.Stop();

        this.AnalizedImage = this.SourceImage.Clone();

        ClusterVisualization(this.AnalizedImage, clusterIndexes);

    Application.Current.Dispatcher.Invoke((SetAnalysisResult));

    / Logging
        this.Logger.Info(String.Format("Analysis took {0}:{1}:{2} and {3} iterations. Total seconds: {4} Image size {5}x{6}",
            this.AnalysisTime.Minutes,
            this.AnalysisTime.Seconds,
            this.AnalysisTime.Milliseconds / 10,
            this.IterationsCount,
            this.AnalysisTime.TotalSeconds,
            this.SourceImage.Width,
```

```
                this.SourceImage.Height
                ));

            StringBuilder logString = new StringBuilder();
            logString.AppendLine($"Dominant colours (sizeCrop: {sizeCrop}):");
            for (int i = 0; i < this.clusters.Length; i++)
            {
                var dominantColour = this.clusters[i];

                logString.AppendLine($"[{i}]
({(int)dominantColour.NewColor.V2}, {(int)dominantColour.NewColor.V1},
{(int)dominantColour.NewColor.V0})");
            }

            this.Logger.Info(logString);
            // End of logging

            this.SourceImage.Dispose();
            this.AnalizedImage.Dispose();

            this.IsFinished = true;

        }

        private void ClusterVisualization(Image<Bgr, Byte> image, int[,]
clusterIndexes)
        {
            for (int y = 0; y < image.Height; y++)
            {
                for (int x = 0; x < image.Width; x++)
```

```
                {
                    int clusterIndex = clusterIndexes[y, x];

                    image.Data[y, x, 2] = (byte)clusters[clusterIndex].Color.V2; //Write to the Red Spectrum
                    image.Data[y, x, 1] = (byte)clusters[clusterIndex].Color.V1; //Write to the Green Spectrum
                    image.Data[y, x, 0] = (byte)clusters[clusterIndex].Color.V0; //Write to the Blue Spectrum
                }
            }
        }

        private void SetAnalysisResult()
        {
            var dominantColoursCollection = new ObservableCollection<PictureDominantColorInfoItem>();

            this.clusters = this.clusters.OrderByDescending((colourCluster) => colourCluster.Count).ToArray();

            for (int i = 0; i < this.clusters.Length; i++)
            {
                //Debug.WriteLine(this.clusters[i].Count);

                Color color = Color.FromRgb(
                    (byte)this.clusters[i].Color.V2,
                    (byte)this.clusters[i].Color.V1,
                    (byte)this.clusters[i].Color.V0);
```

```
                dominantColoursCollection.Add(new
PictureDominantColorInfoItem(color));
            }

            this.AnalyzedPictureInfo = new AnalyzedPictureInfo()
            {
                AnalyzedImage                                    =
Utility.ToBitmapSource(this.SourceImage.ToBitmap()),
                AnalyzedImageWithClusters                        =
Utility.ToBitmapSource(this.AnalizedImage.ToBitmap()),
                DominantColours = dominantColoursCollection
            };
        }

        protected virtual void OnAnalysisCompleteEvent()
        {
            this.AnalysisCompleteEvent?.Invoke(this, EventArgs.Empty);
        }

    }
}
```

ImageResultInfoControl.xaml

```
<UserControl
x:Class="DominantColoursSearch.Controls.ImageResultInfoControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:local="clr-namespace:DominantColoursSearch.Controls"
        xmlns:customClasses="clr-
namespace:DominantColoursSearch.CustomClasses"
        xmlns:viewModel="clr-
namespace:DominantColoursSearch.Controls.ViewModels"
        mc:Ignorable="d"
        d:DesignHeight="450" d:DesignWidth="800">

    <UserControl.DataContext>
      <viewModel:ImageResultInfoControlViewModel />
    </UserControl.DataContext>

    <Border BorderThickness="1" BorderBrush="#FF000000" > <!--
CornerRadius="8"-->
      <Grid>
        <Grid.RowDefinitions>
          <RowDefinition Height="*" />
          <RowDefinition Height="auto" />
        </Grid.RowDefinitions>

        <Image x:Name="imageImageContainer" Source="{Binding
ImageResultInfo.AnalyzedImage}" Margin="0 6"/>

        <ItemsControl x:Name="itemsControlPictureColorInfoContainer"
Grid.Row="1" ItemsSource="{Binding ImageResultInfo.DominantColours}">
          <ItemsControl.ItemsPanel>
            <ItemsPanelTemplate>
```

```
                    <StackPanel Orientation="Horizontal"
HorizontalAlignment="Center"/>
                </ItemsPanelTemplate>
            </ItemsControl.ItemsPanel>

            <ItemsControl.ItemTemplate>
                <DataTemplate DataType="{x:Type
customClasses:PictureDominantColorInfoItem}">
                    <TextBox Width="100" Height="40" Text="{Binding
ColorTextRepresentation, Mode=OneWay}" Background="{Binding ColorBrush,
Mode=OneWay}"
                        TextAlignment="Center"
VerticalContentAlignment="Center" IsReadOnly="True"
IsReadOnlyCaretVisible="False"
                        TextWrapping="Wrap" BorderThickness="0"/>
                </DataTemplate>
            </ItemsControl.ItemTemplate>
        </ItemsControl>
    </Grid>
  </Border>
</UserControl>
```

MainWindow.xaml

```
<Window x:Class="DominantColoursSearch.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
```

```
        xmlns:local="clr-namespace:DominantColoursSearch"
        xmlns:controls="clr-namespace:DominantColoursSearch.Controls"
        xmlns:viewModel="clr-
namespace:DominantColoursSearch.DominantColoursAnalysis"
        mc:Ignorable="d"
        Title="EZColourDominanceAnalyzer" Height="450" Width="800"
        Icon="/Images/colors-48.png"
        d:DataContext="{d:DesignInstance local:MainWindowViewModel}">
    <Grid>
        <Grid.Background>
            <LinearGradientBrush                                    EndPoint="0.5,1"
MappingMode="RelativeToBoundingBox" StartPoint="0.5,0">
                <GradientStop Color="#FFC5D2DE" Offset="0.072"/>
                <GradientStop Color="#FF7CA4CD" Offset="1"/>
                <GradientStop Color="#FFB6D3F0" Offset="0.928"/>
            </LinearGradientBrush>
        </Grid.Background>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="auto" />
            <ColumnDefinition Width="auto"/>
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="*" />
            <RowDefinition Height="2*" />
            <RowDefinition Height="auto" />
        </Grid.RowDefinitions>
```

```
<Menu Grid.Row="0" Grid.ColumnSpan="3" IsMainMenu="True" >
  <MenuItem Header="База даних" Click="DatabaseMenuItem_Click"/>
  <MenuItem Header="Експорт в XML файл" Click="XmlExportMenuItem_Click"/>
  <MenuItem Header="Налаштування" Click="OptionsMenuItem_Click"/>
</Menu>

<StackPanel Grid.Row="1" Grid.Column="0" Orientation="Horizontal" HorizontalAlignment="Center" Grid.ColumnSpan="2" Margin="5">
  <Button Width="75" Height="40" Click="Button_RunClick" Margin="5">
    <TextBlock Text="Запустити все" TextWrapping="Wrap" TextAlignment="Center"/>
  </Button>
  <Button Width="90" Height="40" Click="Button_LoadImageClick" Margin="5">
    <TextBlock Text="Завантажити зображення..." TextWrapping="Wrap" TextAlignment="Center"/>
  </Button>
</StackPanel>

<ListBox x:Name="listBoxLoadedImages" Grid.Column="0" Grid.Row="3" SelectionMode="Single"
    ItemsSource="{Binding Analyzers}" SelectedIndex="{Binding SelectedAnalyzerUniqeIndex, Mode=TwoWay}"
    HorizontalContentAlignment="Stretch" SelectionChanged="listBoxLoadedImages_SelectionChanged"
Grid.ColumnSpan="2" Background="#FFE3ECF9">
  <ListBox.ItemTemplate>
```

```
<DataTemplate DataType="{x:Type viewModel:DominantColoursAnalyzer}">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
        </Grid.RowDefinitions>

        <TextBox Grid.Row="0" HorizontalAlignment="Stretch" VerticalAlignment="Stretch" Text="{Binding FileName, Mode=OneWay}"
                 TextAlignment="Center"
VerticalContentAlignment="Center" IsReadOnly="True" IsReadOnlyCaretVisible="False"
                 TextWrapping="NoWrap" BorderThickness="0" Margin="0 0 0 5" Background="Transparent"/>

        <Grid Grid.Row="1">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="*"/>
            </Grid.ColumnDefinitions>

            <Button Background="Transparent" Width="30" Height="30"
                    BorderThickness="0" VerticalAlignment="Center" HorizontalAlignment="Center">

                <!--<Button.Background>
```

```
                        <ImageBrush ImageSource="/Images/play-button-
48.png" />
                    </Button.Background>-->

                    <Image Source="/Images/play-button-48.png" />

                </Button>
                <Button Grid.Column="1" VerticalAlignment="Center"
HorizontalAlignment="Center"
                        Background="Transparent" BorderThickness="0"
Width="27" Height="27">

                    <Image Source="/Images/save-button-40.png" />

                </Button>
                <Button Grid.Column="2" Width="30" Height="30"
Background="Transparent" BorderThickness="0"
                        VerticalAlignment="Center"
HorizontalAlignment="Center">

                    <Image Source="/Images/stop-button-48.png" />

                </Button>

            </Grid>

            <Separator Grid.Row="2"/>

        </Grid>
    </DataTemplate>
```

```
            </ListBox.ItemTemplate>
        </ListBox>

        <controls:ImageResultInfoControl
x:Name="imageResultInfoControlContainer" Grid.Row="1" Grid.Column="2"
                                    Grid.RowSpan="3"/>

        <StatusBar Grid.Row="4" Grid.ColumnSpan="3">
            <StatusBar.ItemsPanel>
                <ItemsPanelTemplate>
                    <Grid>
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="Auto" />
                            <!--Separator-->
                            <ColumnDefinition Width="Auto" />
                            <ColumnDefinition Width="*" />
                            <!--Separator-->
                            <ColumnDefinition Width="Auto" />
                            <ColumnDefinition Width="Auto" />
                            <ColumnDefinition Width="35" />
                            <!--Separator-->
                            <ColumnDefinition Width="Auto" />
                            <ColumnDefinition Width="Auto" />
                            <ColumnDefinition Width="75" />
                        </Grid.ColumnDefinitions>
                    </Grid>
                </ItemsPanelTemplate>
            </StatusBar.ItemsPanel>

            <StatusBarItem>
```

```
                <TextBlock Text="Дані ініціалізовані" />
            </StatusBarItem>

            <Separator Grid.Column="1"/>

            <Separator Grid.Column="3"/>

            <StatusBarItem Grid.Column="4">
                <TextBlock Text="Кількість ітерацій: " />
            </StatusBarItem>

            <StatusBarItem Grid.Column="5">
                <TextBlock  Text="{Binding  SelectedAnalyzerIterationCountText}"
/>
            </StatusBarItem>

            <Separator Grid.Column="6"/>

            <StatusBarItem Grid.Column="7">
                <TextBlock Text="Час обробки: " />
            </StatusBarItem>

            <StatusBarItem Grid.Column="8">
                <TextBlock Text="{Binding SelectedAnalyzerTimeText}" />
            </StatusBarItem>

        </StatusBar>
    </Grid>
</Window>
```

Додаток В. Ілюстративний матеріал

ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДО ЗАХИСТУ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Завідувач кафедри ПЗ, д. т. н., професор _____ О. Н. Романюк

Науковий керівник, к. т. н., доцент кафедри ПЗ_____О. О. Коваленко

Рецензент, к. т. н., професор кафедри КН _____ Т. О. Савчук

Нормоконтроль, к. т. н., доцент кафедри ПЗ _____О. О. Коваленко

Виконавець, студент групи 1ПІ-18м _____ О. В. Ковбасюк

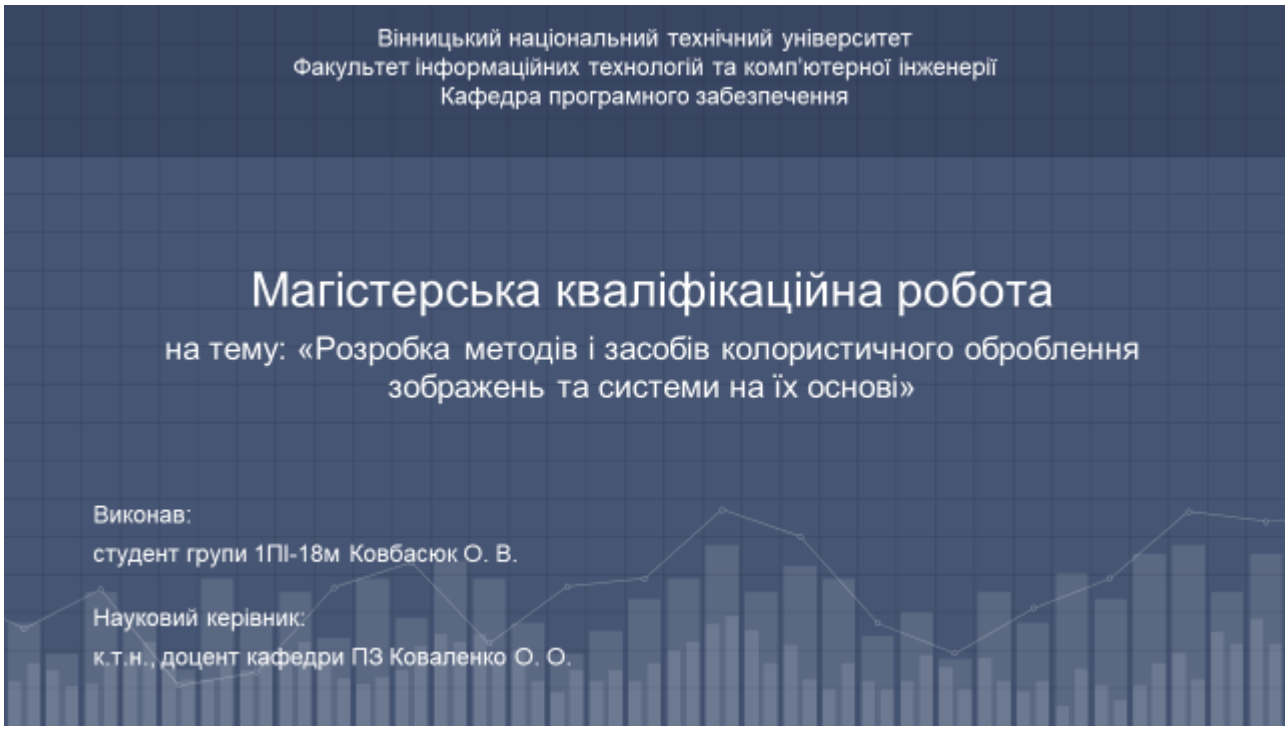

Рисунок В.1 – Початковий слайд презентації

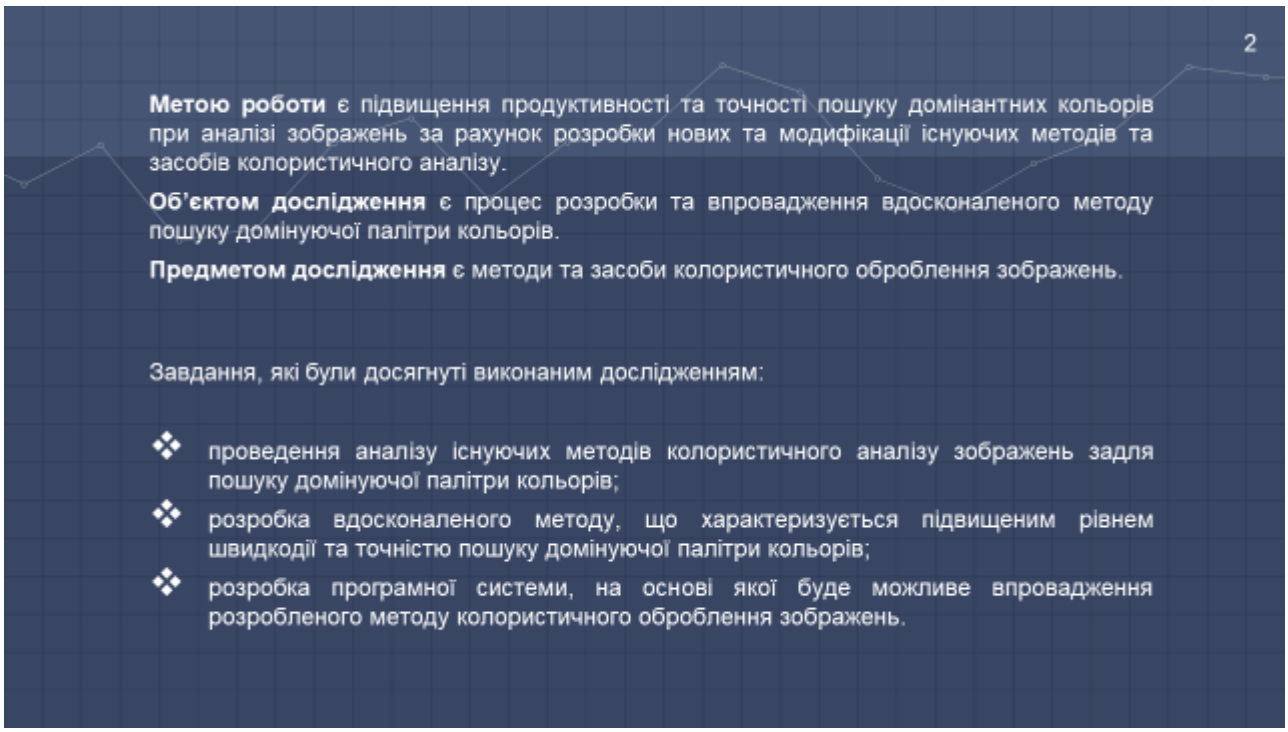

Рисунок В.2 – Мета, об’єкт, предмет та завдання дослідження

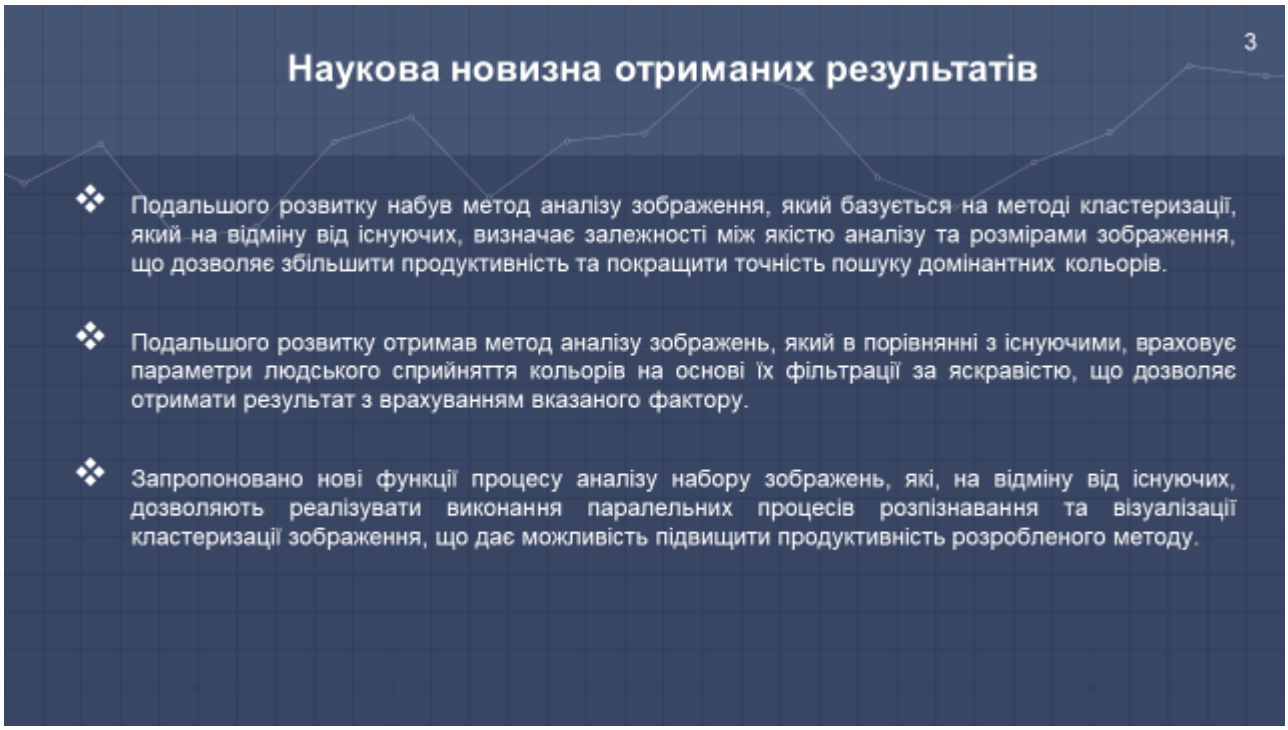

Рисунок В.3 – Наукова новизна отриманих результатів

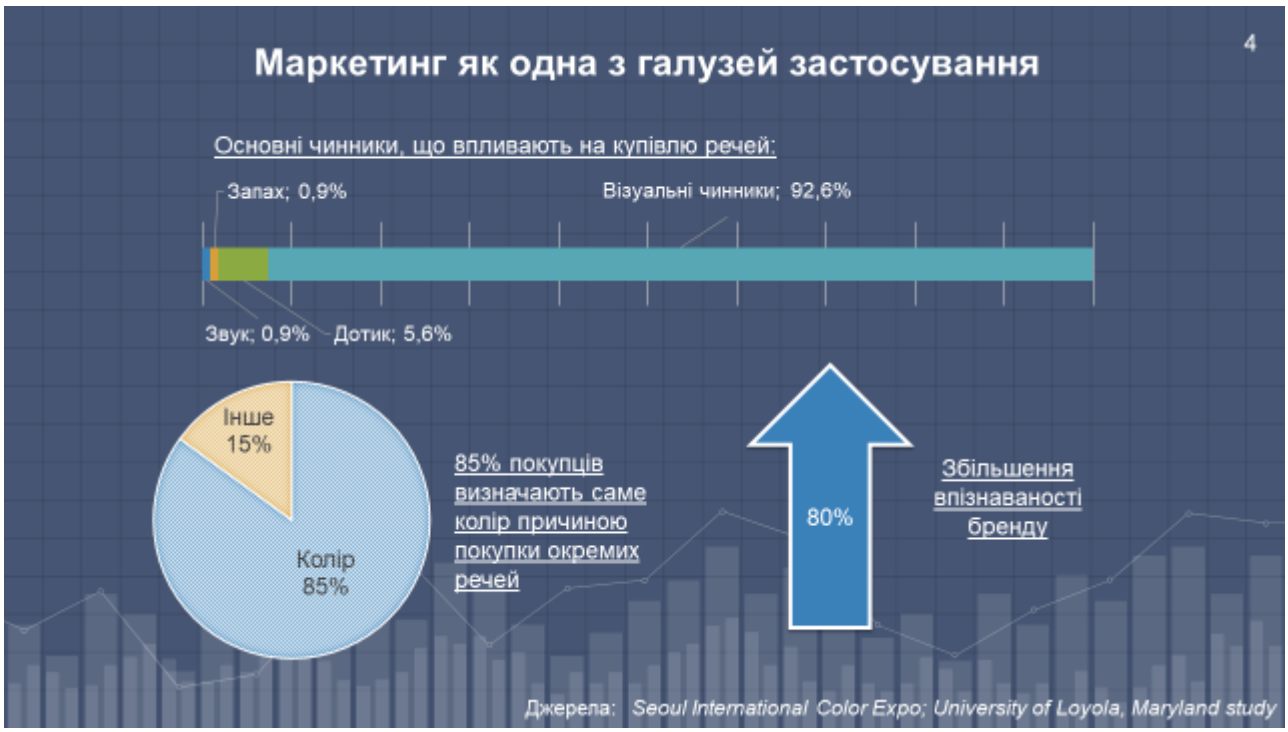

Рисунок В.4 – Маркетинг як одна з галузей застосування

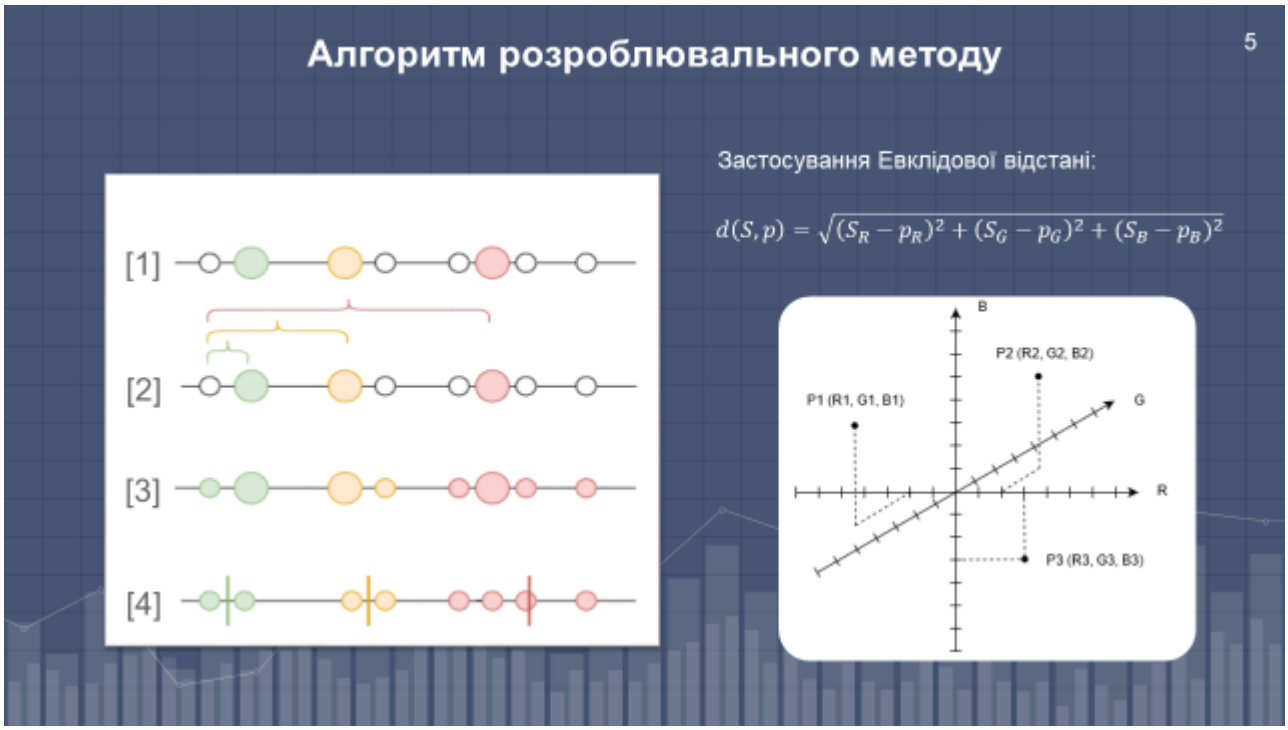

Рисунок В.5 – Алгоритм розроблювального методу

Відсоток від початкових розмірів, %	Розмір	Швидкість, секунд	К-ть ітерацій	Відхилення
100%	736×1308	18,9984839	7	0
67%	490×872	8,4859505	7	3,31
50%	368×654	5,9037977	9	2,56
40%	294×523	2,1571322	5	4,77
33,3%	245×436	1,4734666	5	4,32
28,6%	210×373	1,0905479	5	4,23
25%	184×327	2,897903	16	8,84
22,22%	163×290	2,0792058	16	8,61
20%	147×261	0,8768856	8	11,1
18,18%	133×237	0,4657755	5	15,56

Рисунок В.6 – Залежність розмірів та продуктивності обробки

Рисунок В.7 – Врахування людського сприйняття

Рисунок В.8 – Загальна структура системи

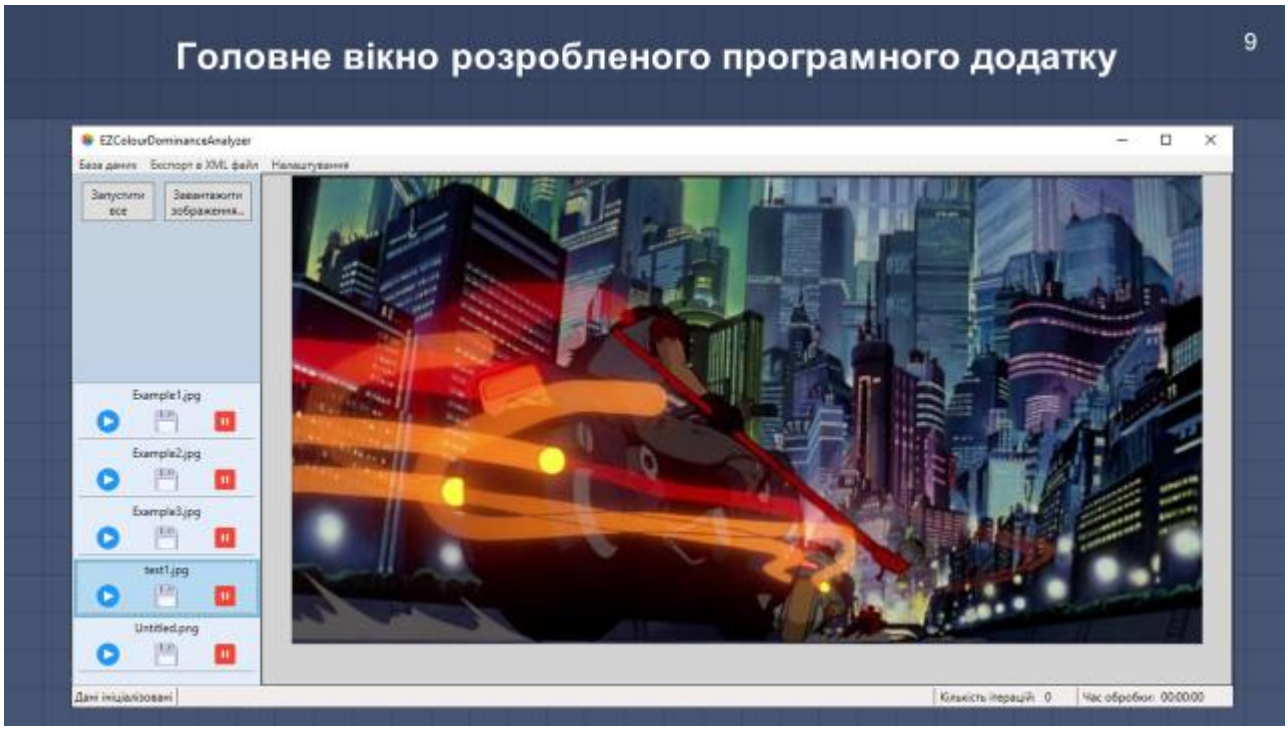

Рисунок В.9 – Головне вікно розробленого програмного додатку

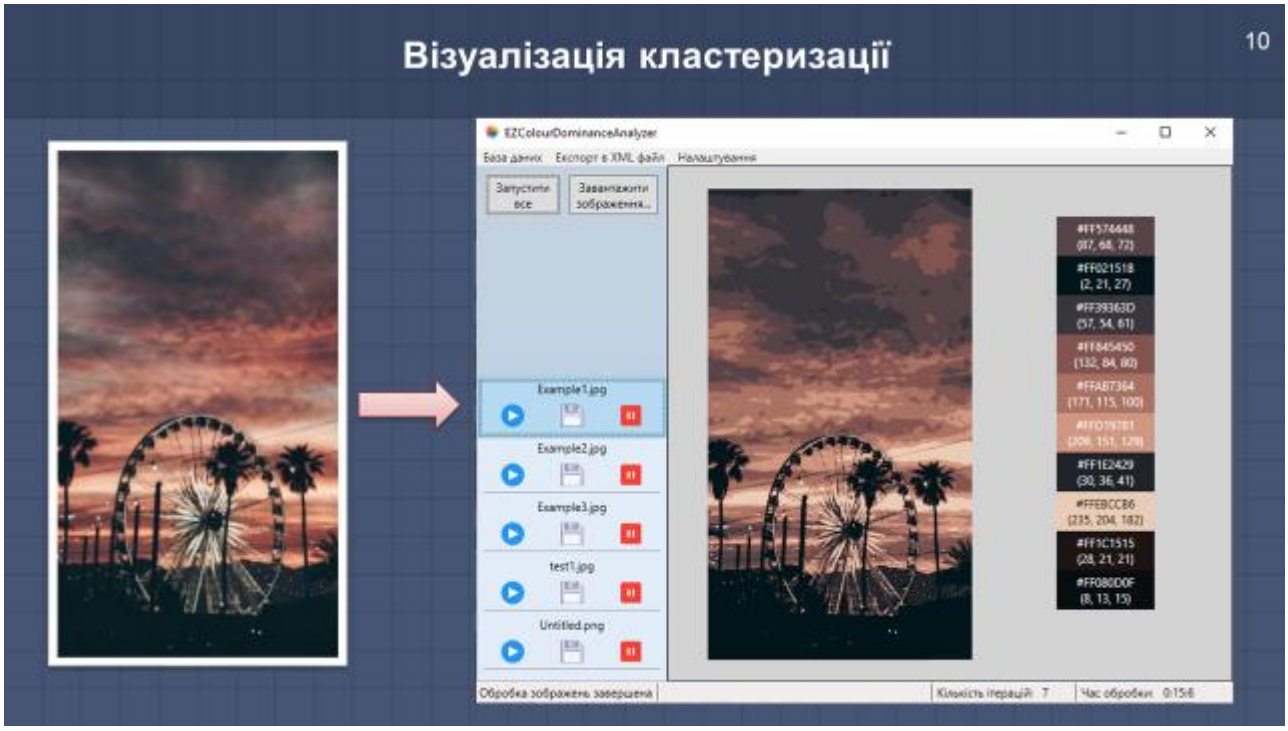

Рисунок В.10 – Візуалізація кластеризації

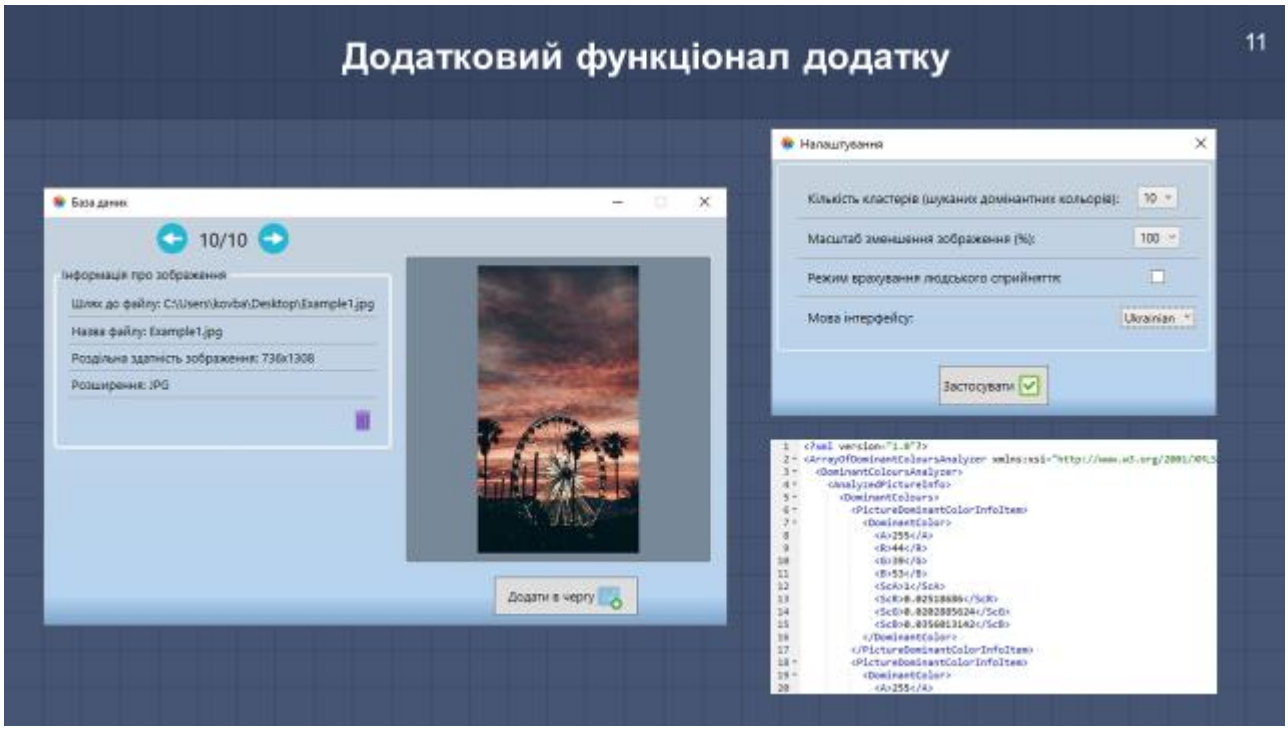

Рисунок В.11 – Додатковий функціонал додатку

Рисунок В.12 – Економічне обґрунтування

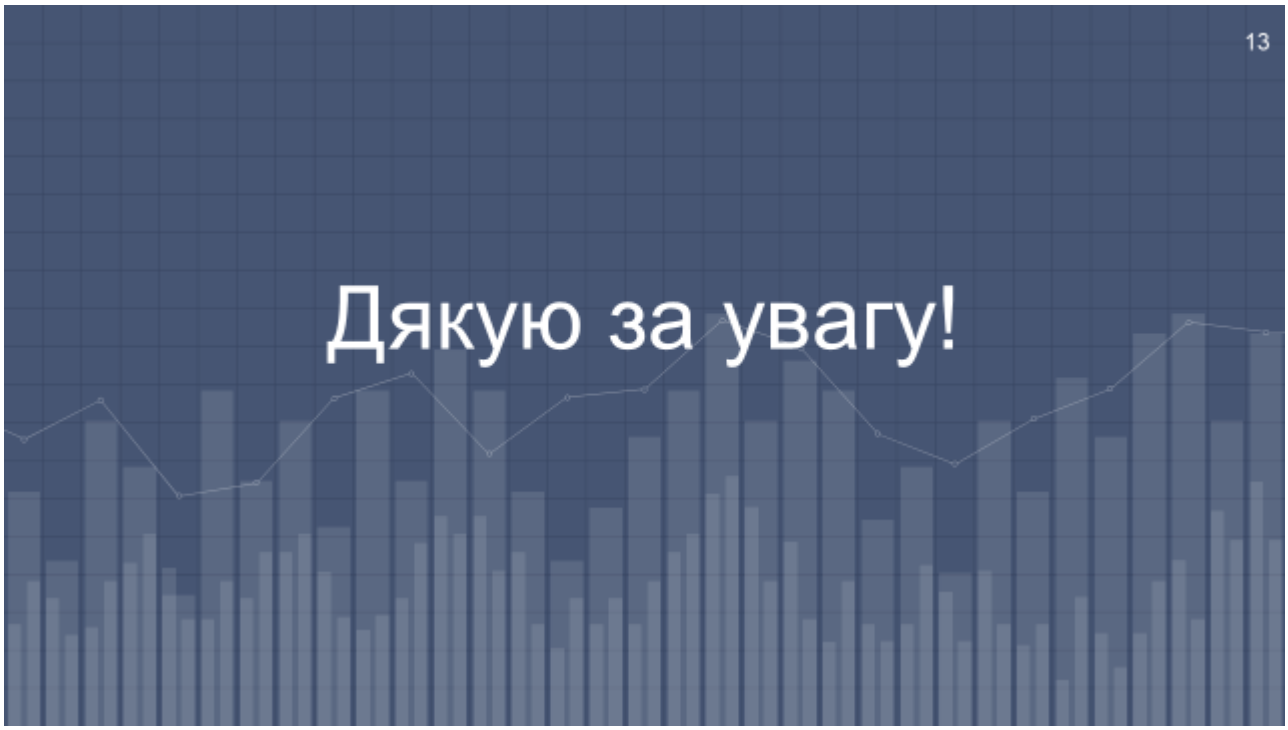

Рисунок В.13 – Кінець презентації