

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему СТВОРЕННЯ ПРОГРАМНОЇ СИСТЕМИ ОПРАЦЮВАННЯ ДАНИХ
ДЛЯ ПЕРЕГЛЯДУ ТА ЗАВАНТАЖЕННЯ АНІМОВАНИХ ЗОБРАЖЕНЬ
DESIGN OF A DATA PROCESSING SYSTEM FOR
VIEWING AND DOWNLOADING ANIMATED IMAGES

Виконав(ла): студент(ка) 4 курсу, групи КНТ-222
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

ДИДЯК М.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ЗАЙКО Т.А.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ДИДЯКА Миколи Володимировича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Створення програмної системи опрацювання даних для перегляду та завантаження анімованих зображень. Design of a Data Processing System for Viewing and Downloading Animated Images

керівник проєкту (роботи) к.т.н., доцент, ЗАЙКО Тетяна Анатоліївна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 7 ” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 03 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис системи опрацювання даних. 4. Експлуатація, тестування та експериментальне дослідження системи опрацювання даних.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЗАЙКО Т.А., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання “7” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури системи опрацювання даних.	2 тиждень	Розділ 3
5	Розробка системи опрацювання даних.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження системи опрацювання даних.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Микола ДИДЯК
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна ЗАЙКО
(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
85 с., 3 табл., 20 рис., 3 дод., 18 джерел.

ANDROID STUDIO, JAVA, БАЗА ДАНИХ, МОБІЛЬНИЙ
ЗАСТОСУНОК, МОВА ПРОГРАМУВАННЯ, СИСТЕМА КЕРУВАННЯ
БАЗАМИ ДАНИХ.

Об'єкт дослідження – алгоритми та процеси обчислень, пов'язані з переглядом та завантаженням анімованих зображень.

Предмет дослідження – програмні системи опрацювання даних для перегляду та завантаження анімованих зображень.

Мета роботи – розробка програмної системи опрацювання даних для перегляду та завантаження анімованих зображень для скорочення витрат користувача на пошук графічного контенту.

Матеріали, методи та технічні засоби: мова програмування Java, середовище розробки Android Studio.

Результати. Розроблено проєктні рішення для створення системи опрацювання даних для перегляду та завантаження анімованих зображень. Створено систему опрацювання даних для перегляду та завантаження анімованих зображень. Проведено дослідження розробленої системи опрацювання даних для перегляду та завантаження анімованих зображень.

Висновки. Мету роботи досягнуто. Розроблено систему опрацювання даних для перегляду та завантаження анімованих зображень за допомогою мови програмування Java та середовища розробки Android Studio.

Галузь використання – мобільні застосунки.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 85 pages, 3 tables, 20 figures, 3 appendixes, 18 sources.

ANDROID STUDIO, JAVA, DATABASE, MOBILE APPLICATION, PROGRAMMING LANGUAGE, DATABASE MANAGEMENT SYSTEM.

The object of research is algorithms and computational processes related to viewing and downloading animated images.

The subject of the research is data processing systems for viewing and downloading animated images.

The purpose of this work is to develop a data processing system for viewing and downloading animated images to reduce the user's costs of searching for graphic content.

Materials, methods and technical tools: Java programming language, Android Studio development environment.

Results. Project solutions for the creation of data processing system for viewing and downloading animated images has been designed. The data processing system for viewing and downloading animated images has been developed. The study of the developed data processing system for viewing and downloading animated images has been conducted.

Conclusions. The goal of the work has been achieved. The data processing system for viewing and downloading animated images using the Java programming language, the Android Studio development environment has been developed.

Scope of use – mobile applications.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	12
1.1 Системи опрацювання даних для перегляду та завантаження анімованих зображень.....	12
1.2 Аналіз систем опрацювання даних для перегляду та завантаження анімованих зображень	16
1.3 Висновки за розділом 1	25
2 Матеріали і методи	27
2.1 Вибір мови програмування	27
2.2 Вибір середовища розробки для створення системи опрацювання даних для перегляду та завантаження анімованих зображень.....	31
2.3 Вибір засобів зберігання та опрацювання даних.....	35
2.4 Висновки за розділом 2	37
3 Опис системи опрацювання даних	39
3.1 Структура системи опрацювання даних для перегляду та завантаження анімованих зображень	39
3.2 Функціонування системи опрацювання даних для перегляду та завантаження анімованих зображень.....	41
3.3 Опис класів системи опрацювання даних для перегляду та завантаження анімованих зображень	45
3.4 Проектування інтерфейсу системи опрацювання даних для перегляду та завантаження анімованих зображень.....	48
3.5 Висновки за розділом 3	50
4 Експлуатація, тестування та експериментальне дослідження системи опрацювання даних	52
4.1 Призначення й умови застосування системи опрацювання даних.....	52

4.2	Характеристики системи опрацювання даних для перегляду та завантаження анімованих зображень.....	53
4.3	Інструкція по експлуатації програми.....	55
4.3.1	Звернення до програми.....	55
4.3.2	Вхідні й вихідні дані	55
4.3.3	Повідомлення.....	55
4.4	Виконання системи опрацювання даних для перегляду та завантаження анімованих зображень	56
4.5	Тестування системи опрацювання даних для перегляду та завантаження анімованих зображень	59
4.6	Висновки за розділом 4	59
	Висновки	60
	Перелік джерел посилання	63
	Додаток А Технічне завдання	65
	Додаток Б Фрагмент тексту програми	69
	Додаток В Слайди презентації	79

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

АЗ – анімоване зображення;

БД – база даних;

ІС – інформаційна система;

СОД – система опрацювання даних.

ВСТУП

Актуальність системи опрацювання даних, створеної для перегляду та завантаження анімованих зображень, визначається постійним зростанням обсягів мультимедійного контенту й необхідністю його швидкої, стабільної та зручної обробки. Сучасний цифровий простір формує підвищений попит на анімацію як засіб комунікації, розваги та передачі візуальної інформації, тому виникає потреба у технічних рішеннях, здатних забезпечити плавне відтворення динамічних зображень незалежно від складності їх структури чи формату. Поширення інтерактивних платформ, навчальних сервісів, соціальних мереж і маркетингових інструментів посилює роль таких систем, оскільки анімовані матеріали стають одним із ключових елементів сучасної взаємодії між користувачем і цифровим середовищем [1], [2].

Підвищення технічних вимог до швидкості завантаження та якості відтворення спонукає до створення ефективних алгоритмів стиснення, оптимізації та маршрутизації мультимедійних даних. Саме тому системи, орієнтовані на опрацювання анімації, набувають особливої значущості у контексті роботи з великими масивами інформації, коли від продуктивності залежить досвід користувача та можливість масштабування цифрових сервісів. Поява анімованих форматів нового покоління, розвиток технологій штучного інтелекту та ускладнення структури графічних ефектів висувають нові вимоги до платформ, що відповідають за їх обробку [3], [4].

У таких умовах розроблення й використання спеціалізованих рішень для перегляду та завантаження анімованих зображень набуває стратегічного значення, оскільки забезпечує надійні інструменти для керування візуальним контентом, підтримує розвиток цифрових сервісів і сприяє задоволенню зростаючих потреб користувачів у швидкому доступі до анімаційних матеріалів [5], [6].

Проте деякі системи опрацювання даних для перегляду та завантаження анімованих зображень характеризуються обмеженнями

продуктивності, які стають відчутними під час роботи з великими файлами або матеріалами високої якості. У багатьох рішень спостерігається зниження швидкості обробки, що спричинює затримки при відтворенні, неплавність анімації або повільне завантаження. Такі ситуації погіршують взаємодію з контентом і знижують комфорт використання програмного забезпечення в умовах інтенсивної роботи з графічними даними. Ще однією слабкою стороною є обмежена підтримка різних форматів анімованих файлів. Частина програм коректно працює лише з найпоширенішими типами анімації, тоді як матеріали у сучасних або менш популярних форматах можуть відображатися некоректно, потребувати додаткових кодеків або взагалі не відкриватися. Додаткові труднощі виникають і через недосконалість інтерфейсних рішень. У ряді програмної продукції інструменти навігації та керування анімацією реалізовані неінтуїтивно, що ускладнює доступ до необхідних функцій і знижує загальну ефективність роботи. У сукупності ці недоліки вказують на необхідність постійного удосконалення програмних рішень, які забезпечують роботу з анімованими зображеннями, щоб гарантувати стабільність, універсальність та безпечність їх використання у сучасному цифровому середовищі [1]-[7].

Тому актуальною є розробка системи опрацювання даних для перегляду та завантаження анімованих зображень. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки системи опрацювання даних для перегляду та завантаження анімованих зображень для скорочення витрат користувача на пошук графічного контенту.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та систем опрацювання даних для перегляду та завантаження анімованих зображень;
- здійснити проєктування системи опрацювання даних для перегляду та завантаження анімованих зображень;

- створити систему опрацювання даних для перегляду та завантаження анімованих зображень;
- дослідити розроблену систему опрацювання даних для перегляду та завантаження анімованих зображень.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Системи опрацювання даних для перегляду та завантаження анімованих зображень

Система опрацювання даних для перегляду та завантаження анімованих зображень є програмним засобом, призначеним для отримання, обробки, відображення та збереження мультимедійного контенту динамічного типу. Її основне призначення полягає у забезпеченні зручної взаємодії користувача з анімованими зображеннями, що зберігаються локально або надходять із зовнішніх інформаційних ресурсів. Така система поєднує механізми роботи з даними, інструменти візуалізації та засоби керування файловими ресурсами в єдиному програмному середовищі [1].

Функціонування подібних систем базується на автоматизованому опрацюванні структурованої інформації, що містить посилання на мультимедійні файли та супровідні метадані. У результаті забезпечується швидке завантаження анімованих зображень, їх коректне відтворення та збереження з урахуванням можливостей платформи. Важливою рисою таких програмних рішень є оптимізація процесів обміну даними, що дозволяє зменшити час очікування та підвищити загальну продуктивність роботи застосунку [1], [2].

Перевагою систем опрацювання даних для перегляду та завантаження анімованих зображень є орієнтація на зручність використання та доступність функціоналу. Інтерфейс, як правило, проєктується таким чином, щоб забезпечити інтуїтивне керування та швидкий доступ до основних можливостей без необхідності тривалого навчання. Окрім цього, такі системи здатні адаптуватися до різних умов експлуатації, зокрема змін параметрів екрана, швидкості мережевого з'єднання та обсягу доступної пам'яті [1], [2].

Особливістю подібних програмних продуктів є їх універсальність і масштабованість, оскільки вони можуть використовуватися як для повсякденного перегляду розважального контенту, так і для професійних

завдань, пов'язаних із роботою з мультимедійними даними. Поєднання можливостей пошуку, фільтрації, попереднього перегляду та завантаження формують цілісне середовище, у якому забезпечується ефективне управління анімованими зображеннями та підвищується загальна якість взаємодії користувача з цифровим контентом [1], [2].

У межах таких систем реалізуються механізми узгодження форматів даних, що дозволяє коректно відтворювати анімацію незалежно від способу її створення або зберігання. Завдяки цьому забезпечується стабільність відображення та знижується ймовірність помилок, пов'язаних із несумісністю програмних компонентів [2], [3].

Важливим аспектом функціонування подібних систем є обробка великих обсягів мультимедійних даних у режимі реального часу. Для цього застосовуються алгоритми оптимізації використання оперативної пам'яті та ресурсів процесора, що сприяє плавному відтворенню анімації без затримок і спотворень. У результаті досягається баланс між якістю зображення та швидкістю його завантаження, що має принципове значення для користувацького досвіду [2], [3].

Такі системи також розглядаються як елемент інформаційної інфраструктури, у межах якої забезпечується контроль доступу до даних і захист мультимедійного контенту. Реалізація механізмів перевірки цілісності файлів і безпечного збереження даних дозволяє мінімізувати ризики втрати або несанкціонованого використання анімованих зображень. Це особливо актуально в умовах активного обміну цифровими матеріалами та зростання кількості зовнішніх джерел інформації [3], [4].

Окрему увагу приділяють можливостям інтеграції таких систем з іншими програмними рішеннями. Підтримка взаємодії з операційною системою, файловими менеджерами та мережевими сервісами розширює функціональні межі програмного продукту та підвищує його практичну цінність. Завдяки цьому система опрацювання даних перестає бути

ізолюваним інструментом і перетворюється на частину єдиного цифрового середовища користувача [3], [4].

Загалом подібні програмні системи сприяють впорядкуванню роботи з анімованими зображеннями, формуванню структурованого підходу до зберігання та перегляду мультимедійних даних, а також підвищенню ефективності їх використання. Поєднання технічних рішень, спрямованих на продуктивність, надійність і зручність, визначає актуальність таких систем у сучасному інформаційному просторі та обґрунтовує доцільність їх подальшого розвитку [4], [5].

Процес створення системи опрацювання даних для перегляду та завантаження анімованих зображень включає комплекс послідовних етапів, які поєднують аналітичну роботу, проектування програмного забезпечення та безпосередню розробку коду. На початковому етапі визначаються функціональні та нефункціональні вимоги до системи, аналізуються цільова аудиторія та сценарії використання, оцінюється тип і обсяг мультимедійних даних, з якими планується працювати. У ході цього формуються ключові цілі проєкту, такі як забезпечення швидкого завантаження анімованих зображень, стабільне відтворення, підтримка різних форматів та інтеграція з зовнішніми сервісами [4], [5].

Далі здійснюється проектування архітектури системи, де визначаються її структурні компоненти та взаємодія між ними. Передбачаються модулі для керування мережею та API, обробки даних, управління інтерфейсом користувача, контролю помилок та завантаження файлів. На цьому етапі проєктується логіка обробки запитів, обробки JSON-файлів та метаданих, що надходять від сервісів з анімованими зображеннями. Розробляється схема взаємодії між класами та пакетами, визначаються об'єктні моделі даних, формуються механізми для асинхронного завантаження файлів та їх кешування для підвищення продуктивності [4], [5].

Після проектування переходять до безпосередньої реалізації програмного забезпечення. Використовуються обрані мови програмування та

середовища розробки, створюються класи та інтерфейси, що відповідають за отримання даних, відображення зображень, пошук і фільтрацію, а також за взаємодію з файловою системою пристрою. Особлива увага приділяється забезпеченню стабільності при роботі з великою кількістю анімованих файлів, оптимізації споживання ресурсів і коректності відтворення анімації в різних режимах екрану. Паралельно розробляється інтерфейс користувача, який забезпечує інтуїтивну навігацію, швидкий доступ до популярних або трендових анімацій та можливість персоналізації перегляду [5], [6].

Наступним етапом є тестування, що включає перевірку функціональності всіх модулів, роботу з різними форматами файлів, обробку помилок та стрес-тестування під навантаженням. Під час цього відбувається усунення багів, оптимізація алгоритмів обробки даних та удосконалення взаємодії з користувачем. Окремо перевіряються механізми безпечного збереження файлів та інтеграція з зовнішніми сервісами, щоб забезпечити стабільність під час підключення до мережі або при відсутності Інтернет з'єднання [5], [6]. На завершальному етапі здійснюється впровадження системи та її підготовка до використання. Передбачаються інструкції для користувачів, механізми оновлення та підтримки програмного забезпечення, а також моніторинг продуктивності у реальному часі. В результаті формується цілісна система, здатна ефективно організовувати роботу з анімованими зображеннями, забезпечувати зручний перегляд та швидке завантаження файлів, а також адаптуватися до змін у структурі даних і умовах використання, що визначає її практичну цінність та актуальність для кінцевого користувача [5], [6].

Визначено, що система опрацювання даних для перегляду та завантаження анімованих зображень є програмним засобом, призначеним для отримання, обробки, відображення та збереження мультимедійного контенту динамічного типу. Її основне призначення полягає у забезпеченні зручної взаємодії користувача з анімованими зображеннями, що зберігаються локально або надходять із зовнішніх інформаційних ресурсів.

1.2 Аналіз систем опрацювання даних для перегляду та завантаження анімованих зображень

Проаналізуємо системи опрацювання даних для перегляду та завантаження анімованих зображень [7]-[14].

Програмне забезпечення 1Gallery (рис. 1.1) орієнтоване на користувачів, яким важлива лаконічність інтерфейсу та відсутність перевантаження другорядними можливостями [8], [12].

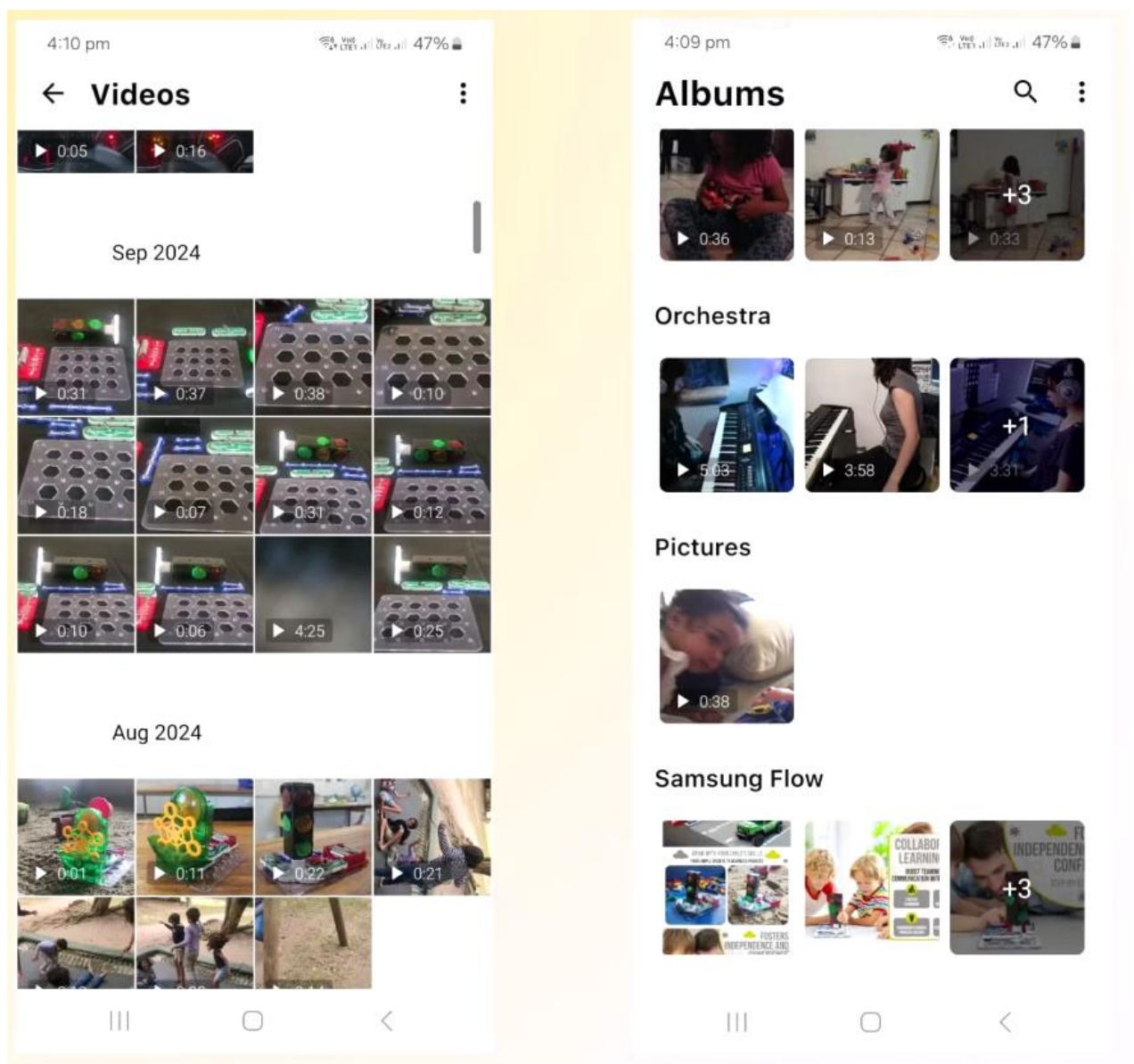


Рисунок 1.1 – Програмна система 1Gallery [12]

У межах мобільного застосунку 1Gallery всі мультимедійні каталоги впорядковуються за назвами, що спрощує навігацію між папками зі збереженими зображеннями й анімованими файлами. Перехід до конкретного альбому супроводжується відображенням вмісту у вигляді зручної сітки, параметри якої можуть змінюватися відповідно до індивідуальних уподобань, забезпечуючи комфортний перегляд як дрібних, так і збільшених елементів. Окрема увага приділяється розмежуванню типів медіаконтенту, оскільки передбачено самостійні розділи для відеоматеріалів та обраних файлів. Для підвищення рівня приватності реалізовано захищене сховище, доступ до якого обмежується автентифікаційними даними користувача. Такий підхід дозволяє ізолювати конфіденційні матеріали від загального переліку та зберігати їх у безпечному середовищі [8], [12].

Функціональні можливості редагування у 1Gallery зведено до мінімально необхідного набору, що відповідає концепції простоти. Передбачено базові операції зміни вигляду зображень, пов'язані з корекцією розмірів, пропорцій та основних візуальних параметрів. При цьому відсутні інструменти для художнього доопрацювання або складної обробки, що дозволяє зосередитися саме на перегляді та впорядкуванні контенту. Обмін файлами реалізовано через інтеграцію з іншими програмами, встановленими на пристрої, завдяки чому передавання анімованих зображень і медіафайлів здійснюється швидко та без додаткових налаштувань [8], [12].

Основними характеристиками програмного забезпечення 1Gallery є такі [8], [12]:

- мінімалістичний інтерфейс: програмне забезпечення 1Gallery орієнтоване на просту та зрозумілу взаємодію без перевантаження екрана зайвими елементами керування, що полегшує навігацію та сприйняття контенту [8], [12];

- впорядкування медіаданих: усі зображення та анімовані файли автоматично групуються за папками з відображенням у логічній послідовності, що спрощує пошук потрібних матеріалів [8], [12];

- адаптивне відображення: передбачено можливість змінювати масштаб елементів у сітці перегляду, що дозволяє налаштовувати вигляд галереї відповідно до індивідуальних потреб користувача [8], [12];
- розділення типів контенту: мультимедійні дані відображаються у спеціалізованих розділах, що забезпечує зручний доступ до анімованих зображень, відео та вибраних файлів [8], [12];
- базове редагування: реалізовано мінімальний набір інструментів для корекції зображень, спрямований на швидке внесення простих змін без використання складних функцій [8], [12];
- захист приватних даних: наявність захищеного альбому дозволяє обмежувати доступ до окремих файлів за допомогою механізмів автентифікації, що підвищує рівень конфіденційності [8], [12];
- інтеграція з іншими застосунками: передавання файлів реалізовано через взаємодію з установленими на пристрої програмами, що спрощує обмін анімованими зображеннями [8], [12].

Серед недоліків програмного забезпечення 1Gallery можна відзначити обмежений набір функцій редагування та відсутність розширених можливостей для глибокої обробки анімованих зображень, що знижує його привабливість для користувачів, які потребують професійних інструментів роботи з медіаконтентом [8], [12].

Програмне забезпечення Movavi Photo Viewer (рис. 1.2) позиціонується як універсальний інструмент для перегляду та впорядкування зображень, у тому числі анімованих, який поєднує стабільну роботу з високою швидкістю відгуку. Його використання дозволяє відмовитися від стандартних засобів, вбудованих у популярні операційні системи, оскільки програма забезпечує ширші можливості керування мультимедійними файлами та зручніший інтерфейс. Завдяки підтримці великої кількості форматів графічних даних, у середовищі Movavi стає можливим відкриття й перегляд зображень різної природи без попередньої конвертації [11], [13].

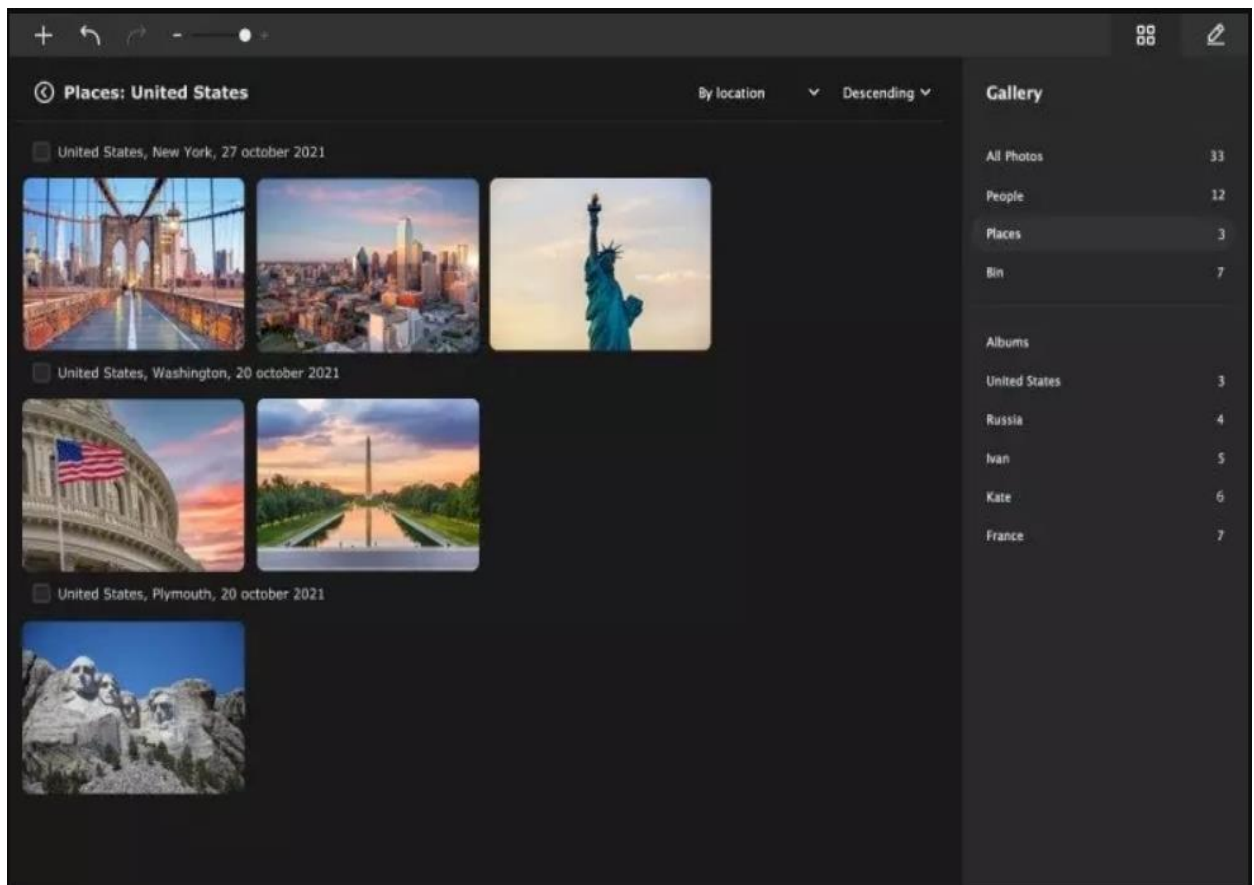


Рисунок 1.2 – Програмна система Movavi Photo Viewer [13]

Особливістю цього програмного продукту є орієнтація на оптимізацію процесів роботи з великою кількістю медіафайлів. Механізми сортування та групування дозволяють систематизувати зображення за змістом або іншими логічними ознаками, що значно спрощує доступ до потрібних матеріалів. Додатковою перевагою виступає можливість швидкого перегляду інформації, пов'язаної з місцем створення зображень, що підвищує інформативність роботи з фотоколекціями [11], [13].

Функціональність Movavi Photo Manager доповнюється інструментами базової обробки, які дають змогу виконувати прості зміни як над окремими файлами, так і над групами зображень одночасно. Такий підхід дозволяє скоротити час на рутинні операції та зробити процес редагування більш ефективним. Разом із цим, під час роботи з великими за обсягом графічними даними може спостерігатися зниження швидкодії, що варто враховувати при використанні програми для обробки ресурсомістких зображень [11], [13].

Програмне забезпечення Movavi Photo Viewer має такі особливості [11], [13]:

- широка підтримка форматів: програмне забезпечення Movavi Photo Viewer забезпечує роботу з великою кількістю графічних типів файлів, що дозволяє переглядати та опрацьовувати зображення без попередньої підготовки або конвертації [11], [13];

- висока швидкодія: система Movavi Photo Viewer орієнтована на швидке відкриття медіафайлів і стабільну роботу під час стандартних операцій перегляду та сортування [11], [13];

- зручна організація даних: реалізовані механізми групування та впорядкування зображень за різними логічними ознаками, що спрощує навігацію у великих колекціях [11], [13];

- використання тегів: підтримується маркування зображень тематичними позначками, що дозволяє швидко знаходити потрібні файли та структурувати робочі матеріали [11], [13];

- пакетна обробка: передбачена можливість одночасного виконання базових операцій над групою зображень, що значно скорочує витрати часу на однотипні дії [11], [13];

- базові інструменти редагування: програмне забезпечення Movavi Photo Viewer надає функції для простого коригування зображень, зокрема зміни орієнтації та розмірів, без використання складних засобів [11], [13];

- інтуїтивна навігація: інтерфейс Movavi Photo Viewer побудовано таким чином, щоб забезпечити швидкий доступ до основних функцій без тривалого навчання користувача [11], [13].

Серед недоліків програмного забезпечення Movavi Photo Viewer можна відзначити можливе зниження продуктивності під час роботи з великими за розміром зображеннями та обмежені можливості глибокого редагування порівняно з професійними графічними редакторами [11], [13].

Програмне забезпечення Aves Gallery (рис. 1.3) позиціонується як універсальний інструмент для перегляду та керування зображеннями, у тому

числі анімованими, який на перший погляд має стриманий і звичний вигляд. Основний екран орієнтований на відображення мультимедійного вмісту у вигляді впорядкованої сітки, де матеріали подаються відповідно до часу їх створення або додавання. Окрім загального перегляду, передбачено швидкий доступ до окремих категорій контенту, що спрощує перемикання між різними типами медіафайлів і каталогами збереження [8], [14].

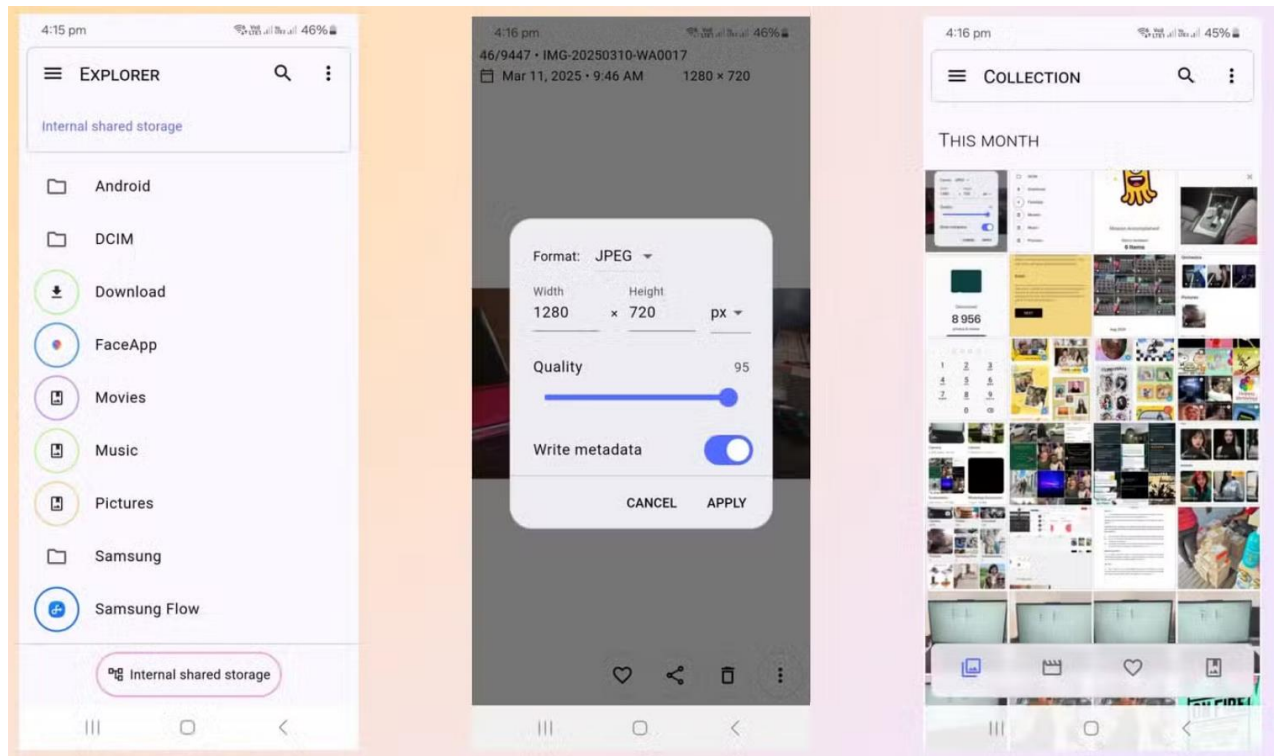


Рисунок 1.3 – Програмна система Aves Gallery [14]

Функціональні можливості Aves Gallery значно виходять за межі стандартних галерейних застосунків. Реалізований механізм навігації дозволяє здійснювати пошук безпосередньо у внутрішній структурі сховища пристрою, надаючи користувачеві більший контроль над розміщенням і доступом до файлів. Розширені інструменти фільтрації дають змогу швидко знаходити необхідні зображення за різними характеристиками, що підвищує ефективність роботи з великими колекціями мультимедійних даних [8], [14].

Aves Gallery вирізняється широкою сумісністю з графічними та відеоформатами, що дозволяє відкривати як поширені типи файлів, так і менш

розповсюджені варіанти без використання сторонніх програм. Додатково передбачено засоби перетворення зображень, які дозволяють змінювати їх параметри та зберігати у зручному форматі. Відсутність власного редактора компенсується можливістю взаємодії з іншими застосунками та функцією передавання зображень і відео на зовнішні екрани для комфортнішого перегляду [8], [14].

Особливу увагу в Aves Gallery приділено роботі з різними типами мультимедійного контенту, включаючи нестандартні формати та спеціалізовані файли. Програма автоматично аналізує медіаколекцію, визначаючи анімовані матеріали, панорамні зображення та інші специфічні об'єкти, що забезпечує їх коректне відображення. Тісна інтеграція з операційною системою дозволяє використовувати застосунок як універсальний засіб перегляду та вибору медіафайлів, забезпечуючи зручну навігацію та швидкий доступ до необхідної інформації [8], [14].

Програмне забезпечення Aves Gallery має такі особливості [8], [14]:

- розширена навігація: програмне забезпечення Aves Gallery забезпечує зручний перехід між альбомами, окремими файлами, тегами та картографічним поданням даних, що спрощує роботу з великими мультимедійними колекціями [8], [14];

- глибокий пошук: реалізовані інструменти фільтрації Aves Gallery дозволяють знаходити анімовані зображення та інші медіафайли за різноманітними параметрами, включаючи вміст, місце створення та додаткові атрибути [8], [14];

- підтримка різних форматів: система Aves Gallery здатна відкривати як поширені, так і спеціалізовані формати зображень і відео, що розширює сферу її застосування [8], [14];

- робота з анімованим контентом: програмне забезпечення Aves Gallery коректно розпізнає та відображає анімовані зображення, панорами та інші нестандартні типи мультимедійних файлів [8], [14];

– інструменти конвертації: передбачено можливість змінювати параметри зображень та зберігати їх у різних форматах без використання сторонніх застосунків [8], [14];

– інтеграція з системою: застосунок Aves Gallery тісно взаємодіє з операційною системою, виконуючи функції універсального переглядача та засобу вибору медіафайлів [8], [14];

– підтримка трансляції: у Aves Gallery реалізовано можливість відображення зображень і відео на зовнішніх екранах для зручнішого перегляду та подальшої обробки [8], [14].

Серед недоліків програмного забезпечення Aves Gallery можна відзначити відсутність вбудованих інструментів повноцінного редагування та відносну складність інтерфейсу для користувачів, які надають перевагу максимально простим галерейним застосункам [8], [14].

Порівняльну характеристику систем опрацювання даних для перегляду та завантаження анімованих зображень наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння систем опрацювання даних для перегляду та завантаження анімованих зображень

Критерій порівняння	1Gallery [12]	Movavi Photo Viewer [13]	Aves Gallery [14]
Простота інтерфейсу	+	+	+—
Підтримка анімованих зображень	+—	+—	+
Можливості пошуку та фільтрації	—	+—	+
Організація медіаколекцій	+—	+	+
Базові інструменти редагування	+—	+—	—
Орієнтація на мобільні пристрої	+	—	+

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато систем опрацювання даних для перегляду та завантаження анімованих зображень. Проте деякі системи опрацювання даних для перегляду та завантаження анімованих зображень характеризуються обмеженнями продуктивності, які стають відчутними під час роботи з великими файлами або матеріалами високої якості. У багатьох рішень спостерігається зниження швидкості обробки, що спричинює затримки при відтворенні, неплавність анімації або повільне завантаження. Такі ситуації погіршують взаємодію з контентом і знижують комфорт використання програмного забезпечення в умовах інтенсивної роботи з графічними даними. Ще однією слабкою стороною є обмежена підтримка різних форматів анімованих файлів. Частина програм коректно працює лише з найпоширенішими типами анімації, тоді як матеріали у сучасних або менш популярних форматах можуть відображатися некоректно, потребувати додаткових кодеків або взагалі не відкриватися. Додаткові труднощі виникають і через недосконалість інтерфейсних рішень. У ряді програмної продукції інструменти навігації та керування анімацією реалізовані неінтуїтивно, що ускладнює доступ до необхідних функцій і знижує загальну ефективність роботи. У сукупності ці недоліки вказують на необхідність постійного удосконалення програмних рішень, які забезпечують роботу з анімованими зображеннями, щоб гарантувати стабільність, універсальність та безпечність їх використання у сучасному цифровому середовищі. Тому актуальною є розробка системи опрацювання даних для перегляду та завантаження анімованих зображень.

При розробці системи опрацювання даних для перегляду та завантаження анімованих зображень необхідно забезпечити такі функціональні вимоги:

- підтримка можливості відображення переліку анімованих зображень;
- підтримка можливості пошуку анімованих зображень за тегами;
- можливість перегляду обраного зображення;

- можливість завантаження анімованого зображення;
- підтримка можливості пошуку анімованого зображення за текстовим запитом.

1.3 Висновки за розділом 1

Визначено, що система опрацювання даних для перегляду та завантаження анімованих зображень є програмним засобом, призначеним для отримання, обробки, відображення та збереження мультимедійного контенту динамічного типу. Її основне призначення полягає у забезпеченні зручної взаємодії користувача з анімованими зображеннями, що зберігаються локально або надходять із зовнішніх інформаційних ресурсів. Така система поєднує механізми роботи з даними, інструменти візуалізації та засоби керування файловими ресурсами в єдиному програмному середовищі.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато систем опрацювання даних для перегляду та завантаження анімованих зображень. Проте деякі системи опрацювання даних для перегляду та завантаження анімованих зображень характеризуються обмеженнями продуктивності, які стають відчутними під час роботи з великими файлами або матеріалами високої якості. У багатьох рішень спостерігається зниження швидкості обробки, що спричинює затримки при відтворенні, неплавність анімації або повільне завантаження. Такі ситуації погіршують взаємодію з контентом і знижують комфорт використання програмного забезпечення в умовах інтенсивної роботи з графічними даними. Ще однією слабкою стороною є обмежена підтримка різних форматів анімованих файлів. Частина програм коректно працює лише з найпоширенішими типами анімації, тоді як матеріали у сучасних або менш популярних форматах можуть відображатися некоректно, потребувати додаткових кодеків або взагалі не відкриватися. Додаткові труднощі виникають і через недосконалість інтерфейсних рішень. У ряді програмної продукції інструменти навігації та керування анімацією

реалізовані неінтуїтивно, що ускладнює доступ до необхідних функцій і знижує загальну ефективність роботи. У сукупності ці недоліки вказують на необхідність постійного удосконалення програмних рішень, які забезпечують роботу з анімованими зображеннями, щоб гарантувати стабільність, універсальність та безпечність їх використання у сучасному цифровому середовищі. Тому актуальною є розробка системи опрацювання даних для перегляду та завантаження анімованих зображень.

Сформульовано функціональні вимоги до системи опрацювання даних для перегляду та завантаження анімованих зображень.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці системи опрацювання даних для перегляду та завантаження анімованих зображень було використано мову програмування Java [15], [16].

Мова програмування Java побудована на ідеї створення універсального середовища, у якому програмний код може працювати передбачувано та однаково на різних платформах, що сприяє формуванню широкої екосистеми застосунків. В основі Java лежить принцип строгої структурованості, що дозволяє підтримувати однозначність програмної логіки та уникати низькорівневих помилок, які часто виникають у мовах, наближених до апаратури [15], [16].

Особливого значення набуває орієнтація на об'єктну модель, яка робить програмні рішення більш масштабованими й придатними до розширення. Цей підхід створює умови для побудови складних систем з чіткою ієрархією, можливістю повторного використання компонентів і зручним керуванням залежностями. Завдяки цьому мова програмування Java активно застосовується у програмних проєктах, що потребують довготривалої підтримки, розрахунку на подальший розвиток та інтеграцію з іншими технологіями [15], [16].

Середовище виконання Java забезпечує надійну ізоляцію коду, що сприяє безпечності під час роботи з даними. Механізми виявлення винятків і контрольоване виконання дають змогу унеможливити неконтрольований доступ або неправильну взаємодію з критичними ресурсами. Це формує довіру до Java у сферах, де потрібний високий рівень надійності, зокрема у фінансових, банківських та телекомунікаційних системах [15], [16].

Еволюція мови не зупиняється, адже постійно додаються нові можливості, що підвищують зручність написання коду, продуктивність виконання та функціональність стандартної бібліотеки. Удосконалення

інструментів компіляції, поява нових синтаксичних конструкцій і поглиблення оптимізацій дозволяють Java залишатися актуальною в умовах швидкої зміни технологічних вимог. Такий поступ забезпечує мові конкурентоспроможність на ринку, де нові технології з'являються щороку [15], [16].

Подальше поширення Java підтримується великим співтовариством розробників, яке впливає на розвиток інструментів, фреймворків і методологій. Цей фактор формує багату інфраструктуру, що охоплює як високорівневі засоби розробки вебзастосунків, так і складні серверні рішення, системи обробки великих даних та інструменти інтеграції. Завдяки відкритості та прозорості Java зберігає статус мови, яка здатна адаптуватися до сучасних викликів і залишатися універсальним інструментом для створення програмних систем різного призначення [15], [16].

У результаті Java постає не просто як мова програмування, а як комплексна технологічна екосистема, що поєднує ефективність, стабільність, сучасність та здатність до постійного вдосконалення, роблячи її придатною для широкого спектра задач — від невеликих додатків до масштабних корпоративних платформ [15], [16].

Використання Java для створення системи опрацювання даних, призначеної для перегляду та завантаження анімованих зображень, є доречним завдяки поєднанню технічних властивостей мови та особливостей її екосистеми. Основна перевага полягає у платформній незалежності, оскільки програма, написана на Java, здатна працювати в однаковому вигляді на різних операційних системах. Це дозволяє створювати універсальний інструмент для опрацювання анімації без необхідності адаптації коду під кожне середовище окремо [15], [16].

Суттєву роль відіграє наявність розвинутого набору бібліотек і фреймворків, які спрощують роботу з графічними форматами, мережевими підключеннями, потоками даних та асинхронною обробкою. Завдяки цьому забезпечується можливість підтримки широкого спектра анімованих файлів, ефективного завантаження матеріалів з різних джерел і стабільного

відтворення анімаційних ефектів. Високий рівень абстракції й багатofункціональні інструменти дають можливість сконцентруватися на логіці програми, а не на низькорівневих технічних складнощах [15], [16].

Важливою причиною вибору Java є її стабільна, контрольована робота з пам'яттю та вбудований механізм керування ресурсами. Під час обробки анімацій часто виникає потреба у завантаженні великих файлів або виконанні кількох процесів паралельно. Автоматичне керування пам'яттю та розвиток багатопоточних можливостей Java дозволяють уникати перевантаження системи, реалізовувати потокову обробку та забезпечувати плавність відтворення навіть у складних умовах [15], [16].

Надійність і безпека також роблять Java придатною для розроблення систем, що працюють із зовнішнім контентом. Механізми перевірки, винятків та обмеження доступу допомагають запобігти виконанню шкідливого коду, некоректному завантаженню або неконтрольованому доступу до файлової системи. Це є особливо важливим у середовищах, де користувач завантажує анімовані матеріали з відкритих джерел [15], [16].

Додатковою перевагою стає можливість створення як настільних застосунків, так і веборієнтованих або серверних рішень. Така гнучкість дозволяє обирати будь-який архітектурний підхід — від локальної системи перегляду до хмарної платформи для обробки анімації. Завдяки цьому Java є придатною для масштабування, модернізації та подальшого розширення функціональності [15], [16].

Усі ці особливості спільно формують підґрунтя для вибору Java як технологічної основи системи, орієнтованої на роботу з анімованими зображеннями, оскільки забезпечується поєднання кросплатформності, ефективності, надійності та гнучкості у розробленні [15], [16].

Обґрунтування вибору мови програмування для розробки системи опрацювання даних для перегляду та завантаження анімованих зображень наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки системи опрацювання даних для перегляду та завантаження анімованих зображень

Критерій порівняння мов програмування	Мова програмування		
	Java	Python	C#
Платформна незалежність	+	+-	+-
Продуктивність під час опрацювання анімації	+	+-	+-
Наявність бібліотек для графіки та мультимедіа	+	+	+-
Зручність для створення ПЗ для перегляду та завантаження анімованих зображень	+	-	-
Придатність для кросплатформних застосунків	+	+	+-

Отже, для реалізації системи опрацювання даних для перегляду та завантаження анімованих зображень обрано мову програмування Java, яка забезпечує платформну незалежність та розвинений набір бібліотек і фреймворків, що спрощують роботу з графічними форматами, мережевими підключеннями, потоками даних та асинхронною обробкою. Крім того, надійність та безпека Java роблять її ефективною для розроблення систем, що працюють із зовнішнім контентом. Іншою перевагою є те, що мова Java є придатною для масштабування, модернізації та подальшого розширення функціональності. Автоматичне керування пам'яттю та розвиток багатопоточних можливостей Java дозволяють реалізовувати потокову обробку та забезпечувати плавність відтворення навіть у складних умовах.

2.2 Вибір середовища розробки для створення системи опрацювання даних для перегляду та завантаження анімованих зображень

В якості середовища розробки для створення системи опрацювання даних для перегляду та завантаження анімованих зображень було обрано середовище Android Studio [17], [18].

Середовище розробки Android Studio являє собою цілісну та глибоко інтегровану платформу, створену для повноцінного циклу розроблення мобільних застосунків. Її особливість полягає в тому, що вона надає не просто набір інструментів, а сформоване програмне середовище, у якому кожен етап проектування, створення, оптимізації та тестування виконується у межах єдиної інфраструктури. Завдяки цьому процес розробки за допомогою цього середовища набуває послідовності й передбачуваності, що є важливим для застосунків, які працюють із графічними даними, мережевими потоками та інтерактивними елементами [17], [18].

Однією з визначальних ознак середовища Android Studio є його здатність працювати із складною структурою проєкту, де кожний компонент чітко організовано у модулі, ресурси та конфігурації. Такий підхід спрощує підтримку програмного коду, дозволяє легко масштабувати функціональність і знижує ймовірність виникнення хаотичної архітектури, яка ускладнює обслуговування системи. Завдяки роботі інтелектуальних механізмів підказок і автоматичного доповнення коду середовище сприяє підвищенню точності та швидкості програмування, оскільки воно адаптується до стилю розробника й пропонує найбільш доречні варіанти [17], [18].

Середовище вирізняється щільною інтеграцією з системами емуляції, що дає можливість активно перевіряти поведінку застосунку в умовах різної продуктивності пристроїв, різної версії операційної системи та різних характеристик апаратного забезпечення. Такий підхід формує реалістичне уявлення про те, як застосунок реагує на обмеження пам'яті, нестабільне

мережеве підключення чи різний розмір екрана, що особливо важливо у випадках роботи з анімованим графічним контентом [17], [18].

Android Studio також надає широкі можливості для роботи з інструментами відлагодження, здатними фіксувати найдрібніші відхилення у роботі застосунку. Середовище постійно аналізує навантаження на процесор, стан мережевої активності, інтенсивність використання пам'яті та інші параметри, що дозволяє розробнику своєчасно виявляти приховані проблеми. Це створює підґрунтя для високої надійності застосунків, що працюють з медіадинамікою, де будь-яке порушення плавності чи затримка стають критичними [17], [18].

Значущою рисою середовища Android Studio є його здатність до постійного оновлення та розширення, оскільки вона тісно пов'язана з екосистемою Google. У результаті розробник отримує доступ до найсучасніших технологій, нових API та фреймворків, що дозволяє підтримувати актуальність застосунку та впроваджувати інноваційні функції без суттєвих технічних обмежень. Такий підхід сприяє швидкій адаптації до нових вимог і стандартів мобільного програмування [17], [18].

У сукупності Android Studio постає як потужний інструмент, що забезпечує не лише розробку, а й повноцінну взаємодію з усіма аспектами мобільного середовища: від графічних ресурсів до хмарних сервісів. Завдяки цьому вона перетворюється на платформу, яка здатна підтримувати створення високопродуктивних, візуально привабливих і технічно досконалих застосунків [17], [18].

Використання середовища розробки Android Studio для створення системи опрацювання даних, орієнтованої на перегляд і завантаження анімованих зображень, є доцільним завдяки поєднанню спеціалізації цього середовища, його інструментальних можливостей та оптимізації під мобільні платформи. Середовище розробки Android Studio створює умови, у яких робота з мультимедійними файлами здійснюється у стабільному, передбачуваному та добре структурованому середовищі, що особливо

важливо під час розроблення застосунків, здатних опрацьовувати значні обсяги анімаційного контенту [17], [18].

Функціональні можливості середовища розробки Android Studio забезпечують глибоку інтеграцію з Android SDK, завдяки чому відкривається доступ до вбудованих механізмів обробки графіки, багатопотокового виконання, кешування та оптимізації ресурсів. Ці інструменти дозволяють працювати з анімованими зображеннями ефективно, оскільки система отримує можливість розподіляти навантаження, прискорювати обробку медіафайлів та зменшувати затримки під час їхнього відтворення. Ефективність досягається також за рахунок оптимізованих бібліотек, які адаптовані саме до мобільного середовища [17], [18].

Важливою перевагою середовища розробки Android Studio є можливість проводити автоматизоване тестування й профілювання, що допомагає відстежувати навантаження на пам'ять, процесор та графічний підсистеми. Така підтримка дозволяє своєчасно виявляти проблеми у продуктивності, усувати можливі затримки відтворення та підвищувати плавність анімації. У результаті створюваний застосунок працює передбачувано навіть у ситуаціях, коли відбувається активне завантаження або обробка мультимедійних даних [17], [18].

Додатковою причиною вибору Android Studio стає наявність потужного інструментарію для роботи з інтерфейсом. Сучасні компоненти, доступ до системи ресурсів, можливість створення адаптивних макетів і використання анімаційних фреймворків дозволяють не лише опрацьовувати зображення, а й забезпечувати привабливе та інтуїтивне їхнє подання. Усе це створює передумови для формування повноцінної системи, здатної поєднувати технічну ефективність із якісним користувацьким досвідом [17], [18].

Істотне значення має й те, що середовище розробки Android Studio підтримує Java – мову, яка добре зарекомендувала себе у контексті роботи з анімованим контентом та інтенсивною обробкою даних. Поєднання мови та середовища утворює цілісну платформу, де всі компоненти системи

розроблення працюють узгоджено, а їхня сумісність гарантує стабільність роботи кінцевого програмного продукту [17], [18].

Таким чином, вибір Android Studio пояснюється тим, що це середовище забезпечує доступ до оптимізованих бібліотек, пропонує зручні механізми керування ресурсами, підтримує ефективну роботу з графікою та медіа, а також створює сприятливі умови для розроблення мобільних застосунків, які повинні опрацьовувати анімовані зображення швидко, надійно та без втрат у якості.

Обґрунтування вибору середовища розробки для створення системи опрацювання даних для перегляду та завантаження анімованих зображень наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення системи опрацювання даних для перегляду та завантаження анімованих зображень

Критерій порівняння середовищ розробки	Середовища розробки		
	Android Studio	IntelliJ IDEA	NetBeans
Оптимізація для мобільної графіки та анімації	+	+—	—
Інтеграція з Android SDK	+	+—	—
Зручність створення інтерфейсу для мобільних застосунків	+	+—	+—
Інструменти профілювання ресурсів	+	+—	—
Можливості емуляції пристроїв	+	—	—

Таким чином, для створення системи опрацювання даних для перегляду та завантаження анімованих зображень обрано середовище розробки Android Studio завдяки поєднанню інструментальних можливостей, оптимізації під мобільні платформи та глибокої інтеграцію з Android SDK, завдяки чому відкривається доступ до вбудованих механізмів обробки графіки, багатопотокового виконання, кешування та оптимізації ресурсів. Ці інструменти дозволяють працювати з анімованими зображеннями ефективно, оскільки система отримує можливість розподіляти навантаження, прискорювати обробку медіафайлів та зменшувати затримки під час їхнього відтворення.

2.3 Вибір засобів зберігання та опрацювання даних

Зберігання та опрацювання даних у системі, призначеній для перегляду та завантаження анімованих зображень, реалізується без використання локальної бази даних, що зумовлює орієнтацію на зовнішні інформаційні ресурси. Уся необхідна інформація надходить у систему з віддаленого сервісу, який виконує роль централізованого сховища мультимедійного контенту та супровідних даних. Передавання відомостей здійснюється у вигляді структурованих текстових повідомлень, що містять як службову інформацію, так і посилання на самі анімовані зображення, доступні для подальшого відтворення або збереження.

Обмін даними між клієнтською частиною застосунку та сервером відбувається через спеціалізований програмний інтерфейс, який дозволяє формувати запити до сервісу з урахуванням заданих параметрів. У відповідь на такий запит система отримує файл у форматі JSON, що слугує універсальним засобом представлення даних у мережевих застосунках. Використання цього формату забезпечує компактність передавання інформації, зручність обробки та незалежність від конкретної платформи, що особливо важливо для мобільних рішень.

Отриманий файл містить ієрархічну структуру, у межах якої зосереджено масив записів із даними про анімовані зображення. У цих записах зберігаються посилання на відповідні GIF-анімації та додаткові атрибути, необхідні для їх коректного відображення. Поряд із цим передається службова інформація, яка використовується для керування процесом отримання даних, зокрема для навігації між сторінками результатів і контролю кількості доступних матеріалів. Окрему роль відіграють дані, що інформують про стан виконання запиту та можливі помилки, які можуть виникати під час взаємодії з сервером.

Опрацювання отриманих даних виконується безпосередньо у застосунку шляхом розбору JSON-структури та вилучення необхідних елементів. Посилання на анімовані зображення використовуються для динамічного завантаження контенту, що дозволяє зменшити обсяг передаваних даних і підвищити швидкість роботи системи. Такий підхід виключає потребу у зберіганні великої кількості файлів локально та сприяє ефективному використанню ресурсів мобільного пристрою.

Нижче наведено узагальнений приклад JSON-структури, адаптованої для роботи з анімованими зображеннями.

```
{
  "data": [
    {
      "id": "abc123",
      "title": "Animated image example",
      "type": "gif",
      "images": {
        "original": {
          "url":
"https://media.example.com/animation_original.gif",
          "width": "480",
          "height": "270",
          "size": "2048000"
        },
        "preview": {
          "url":
"https://media.example.com/animation_preview.gif",
          "width": "240",
          "height": "135",
```

```

        "size": "512000"
    }
}
],
"pagination": {
    "total_count": 5000,
    "count": 1,
    "offset": 0
},
"meta": {
    "status": 200,
    "msg": "OK",
    "response_id": "xyz789"
}
}

```

Подана структура відображає логіку організації даних, за якої вся основна інформація зосереджується у центральному масиві, що містить описи анімованих зображень. Кожен елемент цього масиву пов'язується з унікальним ідентифікатором та супроводжується метаданими, необхідними для коректного відтворення анімації. Усередині вкладених об'єктів зберігаються посилання на файли різної якості та розміру, що дозволяє системі адаптувати відображення під можливості пристрою та умови мережевого з'єднання.

У результаті обрана модель зберігання та опрацювання даних забезпечує гнучкість, масштабованість і простоту підтримки системи, оскільки всі актуальні ресурси надходять із зовнішнього сервісу в уніфікованому форматі. Це дозволяє зосередитися на обробці та відображенні анімованих зображень, не ускладнюючи архітектуру застосунку власними механізмами зберігання даних.

2.4 Висновки за розділом 2

Для реалізації системи опрацювання даних для перегляду та завантаження анімованих зображень обрано мову програмування Java, яка

забезпечує платформну незалежність та розвинений набір бібліотек і фреймворків, що спрощують роботу з графічними форматами, мережевими підключеннями, потоками даних та асинхронною обробкою. Крім того, надійність та безпека Java роблять її ефективною для розроблення систем, що працюють із зовнішнім контентом.

Для створення системи опрацювання даних для перегляду та завантаження анімованих зображень обрано середовище розробки Android Studio, що забезпечує поєднання інструментальних можливостей, оптимізації під мобільні платформи та глибокої інтеграцію з Android SDK, завдяки чому відкривається доступ до вбудованих механізмів обробки графіки, багатопотокового виконання, кешування та оптимізації ресурсів. Ці інструменти дозволяють працювати з анімованими зображеннями ефективно, оскільки система отримує можливість розподіляти навантаження, прискорювати обробку медіафайлів та зменшувати затримки під час їхнього відтворення.

Для зберігання та опрацювання даних використовуються JSON-структури даних, що забезпечує гнучкість, масштабованість і простоту підтримки системи, оскільки всі актуальні ресурси надходять із зовнішнього сервісу в уніфікованому форматі. Це дозволяє зосередитися на обробці та відображенні анімованих зображень, не ускладнюючи архітектуру застосунку власними механізмами зберігання даних.

3 ОПИС СИСТЕМИ ОПРАЦЮВАННЯ ДАНИХ

3.1 Структура системи опрацювання даних для перегляду та завантаження анімованих зображень

Структуру системи опрацювання даних для перегляду та завантаження анімованих зображень наведено на рис. 3.1.

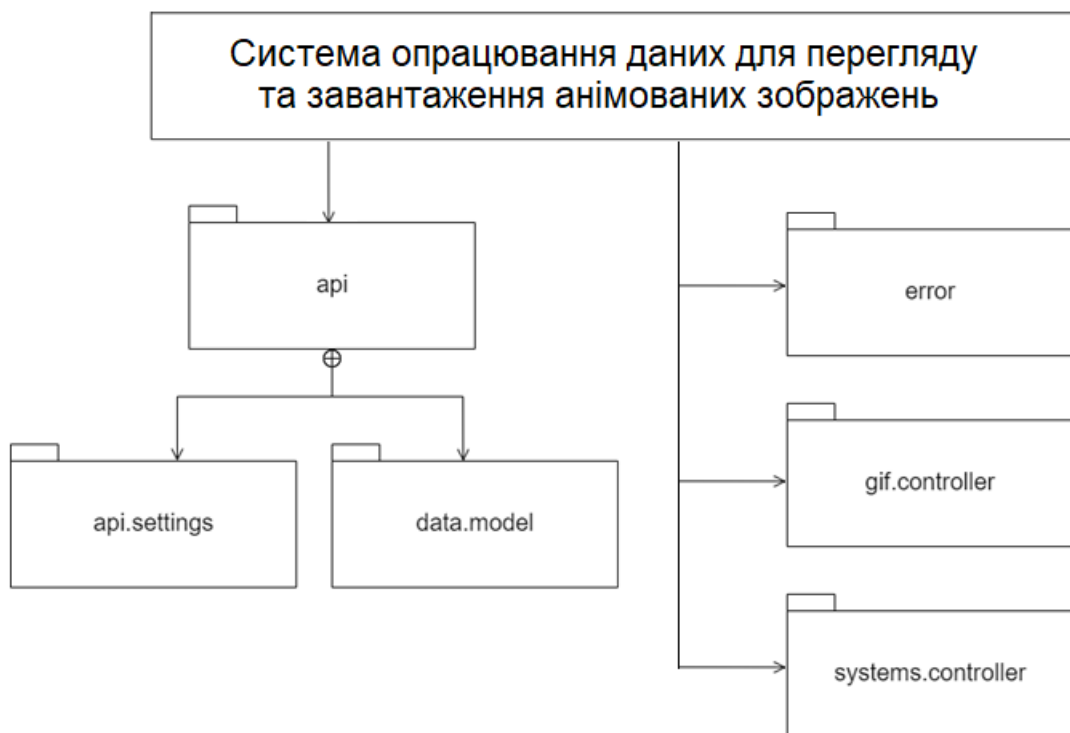


Рисунок 3.1 – Структура системи опрацювання даних для перегляду та завантаження анімованих зображень

Структура системи опрацювання даних для перегляду та завантаження анімованих зображень побудована за модульним принципом, що дозволяє чітко розмежувати відповідальність між окремими частинами програми та забезпечити зручність її підтримки й розширення. В основі архітектури лежить поділ на кілька логічно пов'язаних пакетів, кожен з яких виконує визначену роль у загальному процесі взаємодії користувача із зовнішніми сервісами та мультимедійними ресурсами.

Центральне місце у системі займає пакет `gif.controller`, у межах якого зосереджено механізми, пов'язані з контролем коректності мережевої взаємодії. Саме тут відбувається аналіз стану з'єднання, обробка відповідей від зовнішніх сервісів та виявлення можливих збоїв під час отримання анімованих зображень. Логіка цього пакета забезпечує стабільність роботи системи у випадках нестабільного Інтернет з'єднання або помилок на стороні сервера, що дозволяє своєчасно реагувати на проблеми та запобігати некоректному відображенню контенту.

Доповнює цю функціональність пакет `error`, який використовується для організації взаємодії безпосередньо з файлами формату GIF. У його межах реалізуються механізми контролю правильності роботи з анімованими зображеннями, фіксації помилок, що можуть виникати під час їх завантаження, декодування або збереження. Такий підхід дозволяє ізолювати обробку помилкових ситуацій, пов'язаних із мультимедійними файлами, та підвищити загальну надійність системи.

Окремий рівень архітектури формує пакет `systems.controller`, який відповідає за взаємодію програми з апаратними та системними ресурсами пристрою. У цьому пакеті зосереджена логіка обробки введення з клавіатури, керування відображенням інформації на екрані та забезпечення мережевої активності. Завдяки такому рішенню досягається узгоджена робота між користувацьким інтерфейсом і внутрішніми процесами системи, що особливо важливо під час перегляду анімованих зображень і виконання операцій їх завантаження.

Функціональну основу взаємодії з зовнішніми сервісами утворює пакет `api`, який поділяється на дві логічні складові. Пакет `api.settings` використовується для зберігання та керування параметрами доступу до програмного інтерфейсу, зокрема налаштуваннями запитів і підключення до віддалених ресурсів. Це забезпечує гнучкість у роботі з API та дозволяє змінювати конфігурацію без втручання в основну логіку програми. У свою чергу пакет `data.model` відповідає за опис бізнес-моделей системи, які

відображають структуру отриманих даних та слугують основою для їх подальшого опрацювання й відображення.

У сукупності така структура формує цілісну систему опрацювання даних для перегляду та завантаження анімованих зображень, у якій кожен пакет виконує чітко визначену функцію, не перевантажуючи інші модулі зайвою логікою. Подібний підхід сприяє підвищенню стабільності роботи програми, полегшує тестування та створює передумови для подальшого розвитку системи опрацювання даних для перегляду та завантаження анімованих зображень.

3.2 Функціонування системи опрацювання даних для перегляду та завантаження анімованих зображень

Функціонування системи опрацювання даних для перегляду та завантаження анімованих зображень організовано таким чином, щоб взаємодія з програмним продуктом відбувалася послідовно та без ускладнень, навіть за умови обмеженої кількості доступних можливостей. Після ініціалізації застосунку відбувається підготовка середовища до роботи, що передбачає перевірку необхідних дозволів на доступ до пам'яті пристрою. У разі відсутності таких дозволів система автоматично ініціює запит, оскільки без цього неможливо коректно зберігати анімовані файли у локальному сховищі.

Після успішного запуску користувачу відкривається основний екран, на якому автоматично відображається підбірка найбільш популярних та актуальних анімацій. Такий контент формується за замовчуванням і дозволяє одразу ознайомитися з можливостями програми без необхідності виконувати додаткові дії. Для швидкої навігації між різними категоріями передбачено зручний механізм фільтрації, який дає змогу оперативно змінювати набір відображуваних зображень відповідно до заданих тегів.

Функціонування системи опрацювання даних для перегляду та завантаження анімованих зображень подамо за допомогою схеми, зображеної на рис. 3.2.



Рисунок 3.2 – Схема функціонування системи опрацювання даних для перегляду та завантаження анімованих зображень

Пошук потрібних анімацій реалізовано через введення текстового запиту, що дозволяє знаходити зображення за ключовими словами або фразами. У відповідь на такий запит система формує оновлений набір результатів, які можуть бути одразу переглянуті. Кожне анімоване зображення підтримує режим детального перегляду, у якому воно відображається на весь екран із можливістю зміни орієнтації відповідно до положення пристрою. Це забезпечує зручність перегляду та дозволяє максимально повно оцінити візуальний контент.

За необхідності обране зображення може бути збережене у пам'яті мобільного пристрою. Процес завантаження відбувається у фоновому режимі та не перешкоджає подальшій роботі з програмою. Після завершення всіх дій користувач може безпечно завершити роботу із застосунком, не побоюючись втрати даних або некоректного завершення процесів.

У цілому взаємодія з системою побудована на принципах простоти та передбачуваності. Логіка роботи орієнтована на очікувану поведінку користувача, що дозволяє оптимально організувати інтерфейс, раціонально розмістити елементи керування та забезпечити інтуїтивно зрозуміле використання програмного забезпечення.

Діаграма прецедентів системи опрацювання даних для перегляду та завантаження анімованих зображень (рис. 3.3.) відображає логіку взаємодії між користувачем і функціональними можливостями застосунку, демонструючи послідовність дій та залежності між окремими сценаріями використання. У центрі цієї моделі перебуває користувач, від якого ініціюється основний сценарій роботи із системою. Саме через його дію відбувається запуск застосунку, що є ключовим прецедентом, з яким пов'язані всі інші функціональні можливості.

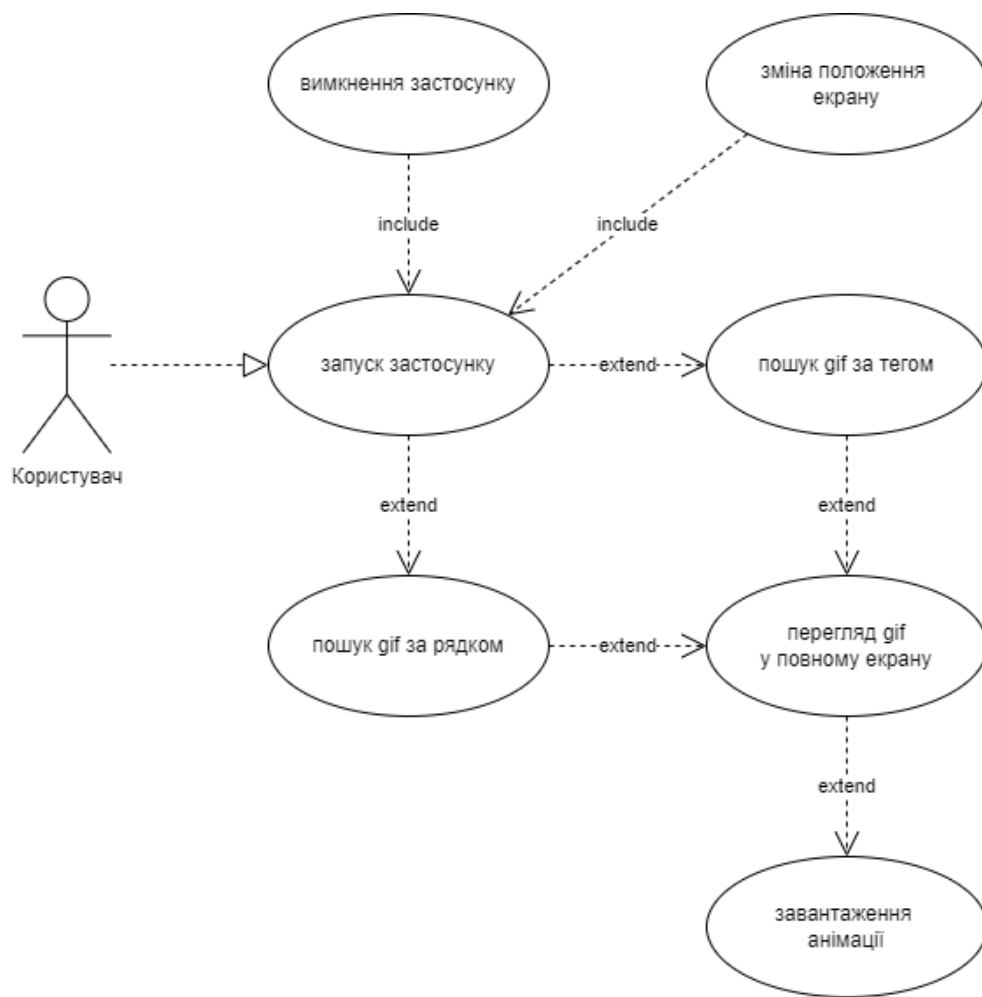


Рисунок 3.3 – Діаграма прецедентів системи опрацювання даних для перегляду та завантаження анімованих зображень

Після активації застосунку система переходить у стан готовності до взаємодії з користувачем, при цьому окремі дії логічно включаються до основного процесу. Зміна орієнтації екрана та завершення роботи застосунку розглядаються як обов'язкові складові сценарію запуску, оскільки вони нерозривно пов'язані з базовим функціонуванням програми. Такі дії не існують автономно, а реалізуються в межах основного прецеденту, забезпечуючи коректне відображення інтерфейсу та контрольований вихід із системи.

Паралельно з основним сценарієм передбачено розширення функціональності, які активуються за ініціативою користувача. Пошук анімованих зображень за тематичними позначками або за довільним

текстовим запитом виступає як додаткові можливості, що не є обов'язковими, але значно розширюють сценарій використання. Такі прецеденти логічно залежать від факту запуску застосунку, оскільки без активного середовища виконання їх реалізація неможлива. Вони доповнюють базовий сценарій, дозволяючи адаптувати роботу системи до конкретних потреб користувача.

Результатом виконання операцій пошуку стає можливість детального перегляду вибраних анімованих зображень. Перехід до режиму повноекранного відображення реалізується як розширення сценаріїв пошуку, оскільки такий перегляд можливий лише після отримання відповідного результату. У цьому режимі система надає змогу зосередитися на візуальному контенті, мінімізуючи вплив допоміжних елементів інтерфейсу та забезпечуючи зручність сприйняття анімації.

Подальше розширення сценарію відбувається у випадку, коли виникає потреба зберегти обране зображення. Завантаження анімації логічно впливає з повноекранного перегляду, оскільки рішення про збереження приймається після детального ознайомлення з контентом. Така залежність підкреслює послідовність взаємодії та відображає реальну поведінку користувача під час роботи із системою.

Загалом діаграма прецедентів формує цілісне уявлення про структуру функціональних зв'язків у системі, демонструючи, як основний сценарій доповнюється допоміжними та розширеними діями. Такий підхід дозволяє чітко окреслити межі відповідальності кожного прецеденту та забезпечує логічну й зрозумілу модель взаємодії користувача з системою опрацювання даних для перегляду та завантаження анімованих зображень.

3.3 Опис класів системи опрацювання даних для перегляду та завантаження анімованих зображень

Класи системи опрацювання даних для перегляду та завантаження анімованих зображень сформовано з урахуванням архітектурних рішень,

спрямованих на спрощення підтримки коду та забезпечення чіткого розмежування відповідальності між окремими компонентами. У процесі проєктування було свідомо обрано поєднання двох патернів проєктування, які доповнюють один одного та відповідають специфіці функціонування застосунку. Такий підхід дозволив вибудувати зрозумілу структуру класів навіть за умови невеликого обсягу функціональності та обмеженої кількості екранів.

Діаграма класів системи опрацювання даних для перегляду та завантаження анімованих зображень наведено на рис. 3.4.

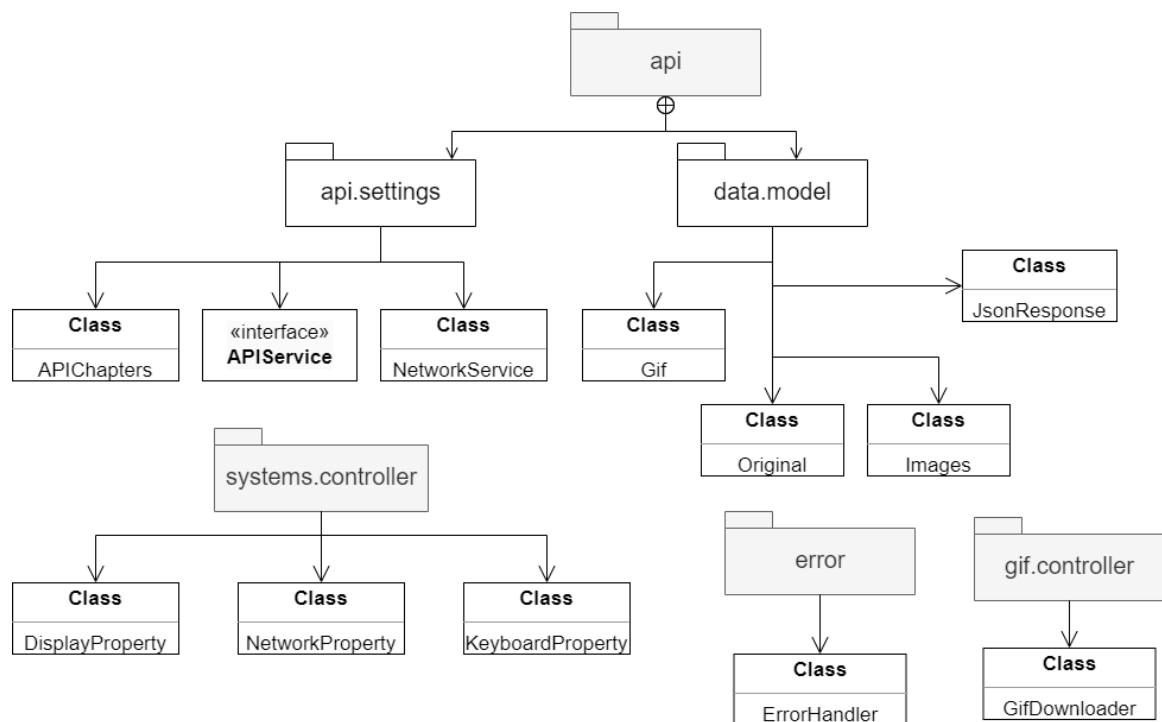


Рисунок 3.4 – Діаграма класів системи опрацювання даних для перегляду та завантаження анімованих зображень

Використання патерна Singleton зумовлене необхідністю централізованого керування мережевою взаємодією. Мережевий компонент системи створюється один раз і надалі використовується повторно протягом усього життєвого циклу застосунку. Це дає змогу уникнути зайвих витрат ресурсів та забезпечити стабільність обміну даними з віддаленим сервісом. Саме з цією метою було реалізовано клас `NetworkService`, який відповідає за

ініціалізацію REST-клієнта та організацію обміну JSON-даними. Його робота тісно пов'язана з інтерфейсом `APIService`, що визначає набір мережових запитів і слугує контрактом для бібліотеки `Retrofit`, яка фактично виконує запити до API.

Для формування параметрів звернення до зовнішнього сервісу використовується окремий клас `APIChapters`, у якому зосереджено статичні значення, необхідні для побудови запитів. Така ізоляція дозволяє зручно керувати налаштуваннями та спрощує зміну параметрів без втручання в логіку інших класів. У поєднанні з цим у системі реалізовано патерн MVP, що забезпечує поділ між рівнем даних, рівнем представлення та проміжним керуючим шаром. Завдяки цьому логіка обробки даних не змішується з логікою відображення, що робить код більш зрозумілим і структурованим.

Окрему групу класів утворюють компоненти, відповідальні за взаємодію з апаратними та системними можливостями пристрою. Клас `DisplayProperty` надає доступ до характеристик екрана, його розмірів і параметрів відображення, що необхідно для коректного показу анімованих зображень. Перевірка стану мережевого з'єднання реалізується за допомогою класу `NetworkProperty`, який дозволяє своєчасно визначати наявність або відсутність доступу до Інтернету. Робота з клавіатурою та керування її поведінкою зосереджені у класі `KeyboardProperty`, що забезпечує коректне введення пошукових запитів.

Для підвищення надійності програмного продукту застосовується клас `ErrorHandler`, який доповнює основну логіку перевітками та обробкою потенційно небезпечних ситуацій. Завдяки цьому система стає більш стійкою до помилок, пов'язаних із мережею, даними або взаємодією з файловою системою. Безпосереднє завантаження анімованих зображень реалізується у класі `GifDownloader`, який використовує можливості системного менеджера завантажень і відповідає за збереження файлів у пам'яті пристрою.

Модель даних системи представлена низкою класів, що відображають структуру інформації, отриманої з сервера. Клас `Gif` описує логічне уявлення

анімованого зображення та містить поля, які відповідають даним окремих елементів JSON-відповіді. Для зберігання посилання на оригінальну анімацію використовується клас `Original`, який інкапсулює доступ до основного ресурсу. Цей клас входить до складу класу `Images`, що фактично представляє розширену структуру зображення з різними варіантами подання. Завершує ланцюг обробки даних клас `JSONResponse`, призначений для перетворення відповіді сервера у зручний для подальшого використання об'єкт із метаданими анімацій.

У сукупності описані класи формують узгоджену та логічну структуру, яка забезпечує коректну роботу системи опрацювання даних для перегляду та завантаження анімованих зображень, поєднуючи мережеву взаємодію, обробку даних, роботу з ресурсами пристрою та відображення контенту в єдиному програмному рішенні.

3.4 Проєктування інтерфейсу системи опрацювання даних для перегляду та завантаження анімованих зображень

Проєктування інтерфейсу взаємодії користувача з системою опрацювання даних для перегляду та завантаження анімованих зображень виконано з урахуванням вимог технічного завдання. При розробленні системи опрацювання даних для перегляду та завантаження анімованих зображень враховані функціональні вимоги до програми:

- підтримка можливості відображення переліку анімованих зображень;
- підтримка можливості пошуку анімованих зображень за тегами;
- можливість перегляду обраного зображення;
- можливість завантаження анімованого зображення;
- підтримка можливості пошуку анімованого зображення за текстовим запитом.

Інтерфейс головного екрану системи опрацювання даних для перегляду та завантаження анімованих зображень наведено на рис. 3.5.



Рисунок 3.5 – Інтерфейс головного екрану системи опрацювання даних для перегляду та завантаження анімованих зображень

Екран перегляду завантаження анімованого зображення на пристрій користувача наведено на рис. 3.6.

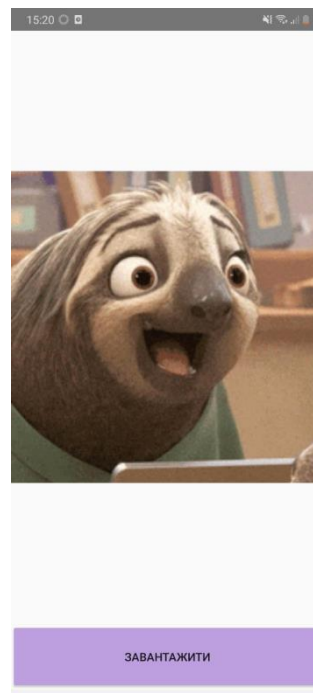


Рисунок 3.6 – Екран перегляду завантаження анімованого зображення на пристрій користувача

Екран пошуку анімованих зображень за текстовим запитом наведено на рис. 3.7.



Рисунок 3.7 – Екран пошуку анімованих зображень за текстовим запитом

3.5 Висновки за розділом 3

Розроблено систему опрацювання даних для перегляду та завантаження анімованих зображень.

Запропоновано структуру системи опрацювання даних для перегляду та завантаження анімованих зображень, що побудована за модульним принципом та дозволяє чітко розмежувати відповідальність між окремими частинами програми й забезпечити зручність її підтримки та розширення. В основі архітектури лежить поділ на кілька логічно пов'язаних пакетів, кожен з

яких виконує визначену роль у загальному процесі взаємодії користувача із зовнішніми сервісами та мультимедійними ресурсами.

Описано функціонування системи опрацювання даних для перегляду та завантаження анімованих зображень. Визначено, що функціонування програми організовано таким чином, щоб взаємодія з програмним продуктом відбувалася послідовно та без ускладнень, навіть за умови обмеженої кількості доступних можливостей. Після ініціалізації застосунку відбувається підготовка середовища до роботи, що передбачає перевірку необхідних дозволів на доступ до пам'яті пристрою. У разі відсутності таких дозволів система автоматично ініціює запит, оскільки без цього неможливо коректно зберігати анімовані файли у локальному сховищі.

Побудовано діаграму прецедентів системи опрацювання даних для перегляду та завантаження анімованих зображень, що відображає логіку взаємодії між користувачем і функціональними можливостями застосунку, демонструючи послідовність дій та залежності між окремими сценаріями використання. У центрі цієї моделі перебуває користувач, від якого ініціюється основний сценарій роботи із системою.

Виконано проєктування інтерфейсу взаємодії користувача з системою опрацювання даних для перегляду та завантаження анімованих зображень.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ СИСТЕМИ ОПРАЦЮВАННЯ ДАНИХ

4.1 Призначення й умови застосування системи опрацювання даних

Система опрацювання даних призначена для підтримки процесу перегляду та завантаження анімованих зображень. Система опрацювання даних написана на мові Java, яка забезпечує платформну незалежність та розвинений набір бібліотек і фреймворків, що спрощують роботу з графічними форматами, мережевими підключеннями, потоками даних та асинхронною обробкою. В якості середовища розробки для створення системи опрацювання даних для перегляду та завантаження анімованих зображень обрано Android Studio завдяки поєднанню інструментальних можливостей, оптимізації під мобільні платформи та глибокої інтеграцію з Android SDK, завдяки чому відкривається доступ до вбудованих механізмів обробки графіки, багатопотокового виконання, кешування та оптимізації ресурсів.

Система опрацювання даних для перегляду та завантаження анімованих зображень забезпечує виконання таких функцій:

- підтримка можливості відображення переліку анімованих зображень;
- підтримка можливості пошуку анімованих зображень за тегами;
- можливість перегляду обраного зображення;
- можливість завантаження анімованого зображення;
- підтримка можливості пошуку анімованого зображення за текстовим запитом.

Для роботи системи опрацювання даних для перегляду та завантаження анімованих зображень потрібні такі технічні та програмні ресурси: мобільний пристрій на базі операційної системи Android, оснащений процесором із тактовою частотою не нижче середнього рівня, що забезпечує стабільне відтворення графічного контенту та коректну роботу мережових запитів. Обсяг оперативної пам'яті має бути достатнім для одночасної обробки

інтерфейсу застосунку, кешування анімованих зображень і фонових операцій, при цьому внутрішнє сховище повинно мати вільний простір для збереження завантажених gif-файлів. Наявність графічного прискорювача є бажаною умовою для плавного відображення анімацій у повноекранному режимі без затримок і втрати якості.

З програмної точки зору необхідною є встановлена та коректно функціонуюча операційна система Android з підтримкою сучасних API, що дозволяють працювати з мережею, файловою системою та мультимедійним контентом. Для обміну даними з віддаленими сервісами потрібне активне підключення до мережі Інтернет через мобільну мережу або бездротове з'єднання Wi-Fi. Також передбачається наявність стандартних системних компонентів Android, зокрема менеджера завантажень, засобів керування дозволами та служб, відповідальних за обробку мережевих запитів.

Окрім цього, для коректної роботи програми необхідно, щоб користувач надав відповідні дозволи на доступ до сховища пристрою та мережевих ресурсів. Сукупність зазначених апаратних і програмних вимог забезпечує стабільне функціонування застосунку, коректний перегляд анімованих зображень, їх швидке завантаження та безпечне збереження в пам'яті мобільного пристрою.

Функціональні характеристики розробленої системи опрацювання даних для перегляду та завантаження анімованих зображень наведено у технічному завданні (додаток А). Фрагмент тексту програмного коду системи опрацювання даних для перегляду та завантаження анімованих зображень наведено у додатку Б.

4.2 Характеристики системи опрацювання даних для перегляду та завантаження анімованих зображень

Система опрацювання даних для перегляду та завантаження анімованих зображень характеризується орієнтацією на швидку взаємодію з

віддаленими інформаційними ресурсами та ефективне відображення мультимедійного контенту на мобільному пристрої. Функціонування програмного продукту побудовано таким чином, щоб забезпечувати оперативне отримання актуальних анімацій, їх коректну обробку та відтворення без суттєвих затримок, навіть за умови змін якості мережевого з'єднання. Внутрішні механізми роботи з даними оптимізовані для обміну інформацією у текстовому форматі, що дозволяє зменшити обсяг переданих даних і прискорити реакцію інтерфейсу.

Важливою характеристикою системи опрацювання даних для перегляду та завантаження анімованих зображень є її модульність і логічна структурованість, завдяки чому основні функції, пов'язані з мережею, обробкою помилок, взаємодією з апаратними компонентами та бізнес-логікою, чітко відокремлені одна від одної. Такий підхід сприяє підвищенню надійності роботи програми, спрощує супровід коду та полегшує подальше розширення функціональних можливостей.

Інтерфейс користувача системи опрацювання даних для перегляду та завантаження анімованих зображень реалізовано з урахуванням принципів інтуїтивної зрозумілості, що забезпечує комфортний перегляд анімованих зображень як у звичайному режимі, так і в повноекранному форматі.

Система опрацювання даних для перегляду та завантаження анімованих зображень також вирізняється адаптивністю до різних параметрів пристрою, оскільки враховуються особливості екрана, орієнтація та доступні ресурси пам'яті. Процеси завантаження анімацій виконуються з урахуванням обмежень мобільної платформи, що дозволяє уникати перевантаження системи та зберігати стабільність роботи. Сукупність цих характеристик формує програмний продукт, який поєднує зручність використання, ефективність обробки даних і практичну придатність для щоденного застосування.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску системи опрацювання даних для перегляду та завантаження анімованих зображень на мобільному пристрої користувача необхідно встановити застосунок та натиснути на його зображення.

4.3.2 Вхідні й вихідні дані

Вхідними даними до системи опрацювання даних для перегляду та завантаження анімованих зображень є інформація, що надходить від користувача у процесі взаємодії з інтерфейсом застосунку, а також дані, отримані з віддаленого мережевого сервісу. До таких даних належать пошукові запити, введені у текстовому вигляді, вибрані тематичні позначки, команди на перегляд або збереження анімацій.

Вихідними даними системи опрацювання даних для перегляду та завантаження анімованих зображень є візуально представлений мультимедійний контент, що відображається на екрані користувача у вигляді анімованих зображень, а також результати виконання запитів на пошук і фільтрацію.

4.3.3 Повідомлення

Користувач системи опрацювання даних для перегляду та завантаження анімованих зображень може отримати такі повідомлення: попередження про відсутність мережевого підключення, повідомлення про неможливість виконання запиту через помилки сервісу чи обмеження доступу, а також службові підказки, що інформують про необхідність надання дозволів або коректність виконуваних дій.

4.4 Виконання системи опрацювання даних для перегляду та завантаження анімованих зображень

Після запуску системи опрацювання даних для перегляду та завантаження анімованих зображень користувачу відображається головний екран (рис. 4.1).



Рисунок 4.1 – Головний екран системи опрацювання даних для перегляду та завантаження анімованих зображень

Користувач може підбирати анімовані зображення за тегами, натискаючи відповідні кнопки (рис. 4.2).



Рисунок 4.2 – Пошук та перегляд анімованих зображень за тегом

Після вибору потрібного зображення воно відображається у повноекранному форматі. Користувач має можливість завантаження (рис. 4.3) обраного зображення на свій пристрій шляхом натискання відповідної кнопки.

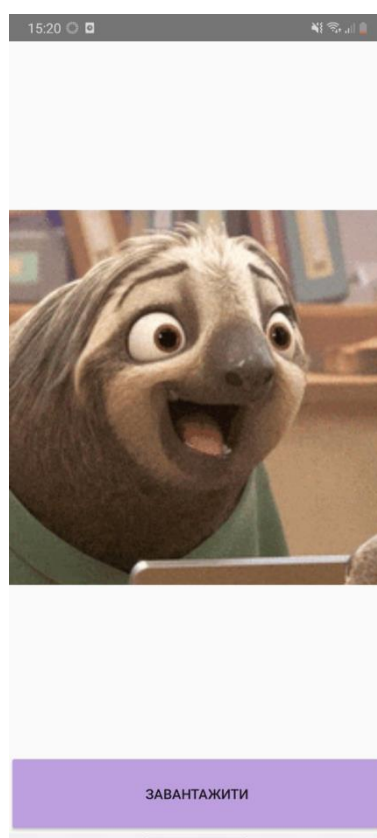


Рисунок 4.3 – Завантаження анімованого зображення на пристрій користувача

Також існує можливість пошуку зображень за текстовим запитом (рис. 4.4) у відповідному текстовому полі.



Рисунок 4.4 – Пошук анімованих зображень за текстовим запитом

При виборі зображення воно відображається для перегляду (рис. 4.5).



Рисунок 4.5 – Перегляд анімованого зображення

4.5 Тестування системи опрацювання даних для перегляду та завантаження анімованих зображень

Проведено тестування системи опрацювання даних для перегляду та завантаження анімованих зображень.

Підтверджено, що системи опрацювання даних для перегляду та завантаження анімованих зображень працює коректно і стабільно, виконуючи всі необхідні функціональні вимоги та успішно справляючись із основною задачею підтримки процесу роботи з зображеннями.

Розроблена система забезпечує організацію та опрацювання даних, пов'язаних із переглядом та завантаженням анімованих зображень.

4.6 Висновки за розділом 4

Описано систему опрацювання даних для перегляду та завантаження анімованих зображень.

Проведено тестування створеної системи опрацювання даних для перегляду та завантаження анімованих зображень. Результати тестування показали, що система забезпечує організацію та опрацювання даних, пов'язаних із переглядом та завантаженням анімованих зображень.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процеси обчислень, пов'язані з переглядом та завантаженням анімованих зображень.

Визначено, що система опрацювання даних для перегляду та завантаження анімованих зображень є програмним засобом, призначеним для отримання, обробки, відображення та збереження мультимедійного контенту динамічного типу. Її основне призначення полягає у забезпеченні зручної взаємодії користувача з анімованими зображеннями, що зберігаються локально або надходять із зовнішніх інформаційних ресурсів.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато систем опрацювання даних для перегляду та завантаження анімованих зображень. Проте деякі системи опрацювання даних для перегляду та завантаження анімованих зображень характеризуються обмеженнями продуктивності, які стають відчутними під час роботи з великими файлами або матеріалами високої якості. У багатьох рішеннях спостерігається зниження швидкості обробки, що спричинює затримки при відтворенні, неплавність анімації або повільне завантаження. Такі ситуації погіршують взаємодію з контентом і знижують комфорт використання програмного забезпечення в умовах інтенсивної роботи з графічними даними. Ще однією слабкою стороною є обмежена підтримка різних форматів анімованих файлів. Частина програм коректно працює лише з найпоширенішими типами анімації, тоді як матеріали у сучасних або менш популярних форматах можуть відображатися некоректно, потребувати додаткових кодеків або взагалі не відкриватися. Додаткові труднощі виникають і через недосконалість інтерфейсних рішень. У ряді програмної продукції інструменти навігації та керування анімацією реалізовані неінтуїтивно, що ускладнює доступ до необхідних функцій і знижує загальну ефективність роботи. У сукупності ці недоліки вказують на необхідність постійного удосконалення програмних рішень, які забезпечують

роботу з анімованими зображеннями, щоб гарантувати стабільність, універсальність та безпечність їх використання у сучасному цифровому середовищі. Тому актуальною є розробка системи опрацювання даних для перегляду та завантаження анімованих зображень.

Сформульовано функціональні вимоги до системи опрацювання даних для перегляду та завантаження анімованих зображень.

Для реалізації системи опрацювання даних для перегляду та завантаження анімованих зображень обрано мову програмування Java, яка забезпечує платформну незалежність та розвинений набір бібліотек і фреймворків, що спрощують роботу з графічними форматами, мережевими підключеннями, потоками даних та асинхронною обробкою. Крім того, надійність та безпека Java роблять її ефективною для розроблення систем, що працюють із зовнішнім контентом.

Для створення системи опрацювання даних для перегляду та завантаження анімованих зображень обрано середовище розробки Android Studio, що забезпечує поєднання інструментальних можливостей, оптимізації під мобільні платформи та глибокої інтеграцію з Android SDK, завдяки чому відкривається доступ до вбудованих механізмів обробки графіки, багатопотокового виконання, кешування та оптимізації ресурсів. Ці інструменти дозволяють працювати з анімованими зображеннями ефективно, оскільки система отримує можливість розподіляти навантаження, прискорювати обробку медіафайлів та зменшувати затримки під час їхнього відтворення.

Для зберігання та опрацювання даних використовуються JSON-структури даних, що забезпечує гнучкість, масштабованість і простоту підтримки системи, оскільки всі актуальні ресурси надходять із зовнішнього сервісу в уніфікованому форматі. Це дозволяє зосередитися на обробці та відображенні анімованих зображень, не ускладнюючи архітектуру застосунку власними механізмами зберігання даних.

Розроблено систему опрацювання даних для перегляду та завантаження анімованих зображень.

Запропоновано структуру системи опрацювання даних для перегляду та завантаження анімованих зображень, що побудована за модульним принципом та дозволяє чітко розмежувати відповідальність між окремими частинами програми й забезпечити зручність її підтримки та розширення. В основі архітектури лежить поділ на кілька логічно пов'язаних пакетів, кожен з яких виконує визначену роль у загальному процесі взаємодії користувача із зовнішніми сервісами та мультимедійними ресурсами.

Описано функціонування системи опрацювання даних для перегляду та завантаження анімованих зображень. Визначено, що функціонування програми організовано таким чином, щоб взаємодія з програмним продуктом відбувалася послідовно та без ускладнень, навіть за умови обмеженої кількості доступних можливостей. Після ініціалізації застосунку відбувається підготовка середовища до роботи, що передбачає перевірку необхідних дозволів на доступ до пам'яті пристрою. У разі відсутності таких дозволів система автоматично ініціює запит, оскільки без цього неможливо коректно зберігати анімовані файли у локальному сховищі.

Побудовано діаграму прецедентів системи опрацювання даних для перегляду та завантаження анімованих зображень, що відображає логіку взаємодії між користувачем і функціональними можливостями застосунку, демонструючи послідовність дій та залежності між окремими сценаріями використання. У центрі цієї моделі перебуває користувач, від якого ініціюється основний сценарій роботи із системою.

Виконано проєктування інтерфейсу взаємодії користувача з системою опрацювання даних для перегляду та завантаження анімованих зображень.

Проведено тестування створеної системи опрацювання даних для перегляду та завантаження анімованих зображень. Результати тестування показали, що система забезпечує організацію та опрацювання даних, пов'язаних із переглядом та завантаженням анімованих зображень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Image viewer. URL: <https://www.computerhope.com/jargon/i/image-viewer.htm> (date of access: 27.04.2026).
2. What is an Image Viewer. URL: <https://www.virtualcuriosities.com/articles/4806/image-viewer> (date of access: 27.04.2026).
3. Image Viewer. Image Resizer, Batch Image Converter and More. URL: <https://www.xnview.com/en/> (date of access: 27.04.2026).
4. Online Image Viewer. URL: <https://image-viewer.com/> (date of access: 27.04.2026).
5. Image Viewer App in Java – View and Navigate Through Images in Java. URL: <https://learnprogramming.in.net/image-viewer-app-in-java-view-and-navigate-through-images-in-java/> (date of access: 27.04.2026).
6. How to Build a JavaScript Image Viewer. URL: <https://www.nutrient.io/blog/how-to-build-a-javascript-image-viewer-with-viewerjs/> (date of access: 27.04.2026).
7. Best Photo Viewers for Android. URL: <https://sourceforge.net/software/photo-viewers/android/> (date of access: 27.04.2026).
8. I use these 6 Android gallery apps instead of Google Photos and Samsung Gallery. URL: <https://www.pocket-lint.com/i-use-these-gallery-apps-to-view-photos-and-videos-on-android/> (date of access: 27.04.2026).
9. Best Photo Viewers for Android. URL: <https://slashdot.org/software/photo-viewers/android/> (date of access: 27.04.2026).
10. Top Photo Downloader Apps for Android. URL: <https://thinglabs.io/photo-downloader-for-android> (date of access: 27.04.2026).
11. Best Photo Viewers for Mac and Windows. URL: <https://fixthephoto.com/best-photo-viewer.html> (date of access: 27.04.2026).
12. 1Gallery. URL: <https://1gallery.app/> (date of access: 27.04.2026).

13. Movavi Photo Viewer. URL: <https://www.movavi.com/support/how-to/photo-viewer.html> (date of access: 27.04.2026).
14. Aves Gallery. URL: <https://apkgk.com/deckers.thibault.aves> (date of access: 27.04.2026).
15. Java Platform Standard Edition Development Kit. URL: <https://www.oracle.com/java/technologies/javase/jdk25-readme-downloads.html> (date of access: 27.04.2026).
16. Java Platform Evolution. URL: <https://dev.java/evolution/> (date of access: 27.04.2026).
17. Android Studio Download Archives. URL: <https://developer.android.com/studio/archive> (date of access: 27.04.2026).
18. Mastering Android Studio: A Comprehensive Guide to the IDE. URL: <https://codezup.com/mastering-android-studio-comprehensive-guide/> (date of access: 27.04.2026).

ДОДАТОК А
Технічне завдання

Вступ

Програмна система опрацювання даних може використовуватися для підтримки процесу перегляду та завантаження анімованих зображень.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Створення програмної системи опрацювання даних для перегляду та завантаження анімованих зображень», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 7 квітня 2026 р.

A.2 Призначення розробки

Система опрацювання даних призначена для перегляду та завантаження анімованих зображень.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Система опрацювання даних для перегляду та завантаження анімованих зображень забезпечує виконання таких функцій:

- підтримка можливості відображення переліку анімованих зображень;
- підтримка можливості пошуку анімованих зображень за тегами;
- можливість перегляду обраного зображення;
- можливість завантаження анімованого зображення;
- підтримка можливості пошуку анімованого зображення за текстовим запитом.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс системи опрацювання даних для перегляду та завантаження анімованих зображень повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

A.3.3 Вимоги до надійності

Система опрацювання даних для перегляду та завантаження анімованих зображень повинна забезпечити надійне функціонування.

A.3.4 Умови експлуатації

Для експлуатації системи опрацювання даних для перегляду та завантаження анімованих зображень необхідна наявність персонального комп'ютера.

A.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи системи опрацювання даних для перегляду та завантаження анімованих зображень потрібні такі технічні та програмні ресурси: мобільний пристрій на базі операційної системи Android, оснащений процесором із тактовою частотою не нижче середнього рівня, що забезпечує стабільне відтворення графічного контенту та коректну роботу мережових запитів. Обсяг оперативної пам'яті має бути достатнім для одночасної обробки інтерфейсу застосунку, кешування анімованих зображень і фонових операцій, при цьому внутрішнє сховище повинно мати вільний простір для збереження завантажених gif-файлів. Наявність графічного прискорювача є бажаною

умовою для плавного відображення анімацій у повноекранному режимі без затримок і втрати якості.

З програмної точки зору необхідною є встановлена та коректно функціонуюча операційна система Android з підтримкою сучасних API, що дозволяють працювати з мережею, файловою системою та мультимедійним контентом. Для обміну даними з віддаленими сервісами потрібне активне підключення до мережі Інтернет через мобільну мережу або бездротове з'єднання Wi-Fi.

A.3.6 Вимоги до маркування і пакування

Система опрацювання даних для перегляду та завантаження анімованих зображень може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва – «Система опрацювання даних для перегляду та завантаження анімованих зображень».

A.4 Вхідні дані до роботи

Вхідними даними до системи опрацювання даних для перегляду та завантаження анімованих зображень є інформація, що надходить від користувача у процесі взаємодії з інтерфейсом застосунку, а також дані, отримані з віддаленого мережевого сервісу. До таких даних належать пошукові запити, введені у текстовому вигляді, вибрані тематичні позначки, команди на перегляд або збереження анімацій.

Вихідними даними системи опрацювання даних для перегляду та завантаження анімованих зображень є візуально представлений мультимедійний контент, що відображається на екрані користувача у вигляді анімованих зображень, а також результати виконання запитів на пошук і фільтрацію.

ДОДАТОК Б
Фрагмент тексту програми

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ctx = this;
    scrll = new ScrollView(ctx);
    lnrlt = new LinearLayout(ctx);
    lnrlt.setOrientation(LinearLayout.VERTICAL);
    scrll.addView(lnrlt, new ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.WRAP_CONTENT
    ));
    setContentView(scrll);
    hndlr = new Handler(Looper.getMainLooper());
    dtprvdr = new DtPrvdr();
    gfdwnldr = new GfDwnldr(ctx);
    ntwrkstts = new NtwrkStts(ctx);
    gflst = new ArrayList<>();
    prcssStrt();
}

private void prcssStrt() {
    if (ntwrkstts.chkCnct()) {
        ldGfs();
    } else {
        shwMsg("Ntwrk nt vlbl");
    }
}

private void ldGfs() {
    hndlr.postDelayed(new Runnable() {
        @Override
        public void run() {
            List<GfMdl> rslts = dtprvdr.gtDt();
            gflst.clear();
            gflst.addAll(rslts);
            rndrGfs();
        }
    }, 800);
}

private void rndrGfs() {
    lnrlt.removeAllViews();
    for (int n = 0; n < gflst.size(); n++) {
        final GfMdl gf = gflst.get(n);
        LinearLayout itm = new LinearLayout(ctx);
        itm.setOrientation(LinearLayout.VERTICAL);
        TextView ttl = new TextView(ctx);
        ttl.setText(gf.gtTtl());
    }
}

```

```

        ImageView mg = new ImageView(ctx);
        mg.setLayoutParams(new ViewGroup.LayoutParams(
            ViewGroup.LayoutParams.MATCH_PARENT,
            600
        ));
        Button btn = new Button(ctx);
        btn.setText("Dwnld");
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                gfdwnldr.dwnldGf(gf.gtUrl());
            }
        });
        itm.addView(ttl);
        itm.addView(mg);
        itm.addView(btn);
        lnrlt.addView(itm);
    }
}

private void shwMsg(String txt) {
    TextView tv = new TextView(ctx);
    tv.setText(txt);
    lnrlt.addView(tv);
}

package gfs.pckg;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

class DtPrvdr {

    public List<GfMdl> gtDt() {
        List<GfMdl> lst = new ArrayList<>();
        for (int k = 0; k < 20; k++) {
            GfMdl mdl = new GfMdl(
                "Gf " + k,
                "https://giphy.com/sample" + k + ".gif"
            );
            lst.add(mdl);
        }
        prcssDt(lst);
        return lst;
    }
}

```

```

private void prcssDt(List<GfMdl> lst) {
    for (int m = 0; m < lst.size(); m++) {
        GfMdl md = lst.get(m);
        md.sttFlg(trFlg());
    }
}

private boolean trFlg() {
    Random rnd = new Random();
    return rnd.nextBoolean();
}
}

```

```
package gfs.pckg;
```

```

class GfMdl {

    private String ttl;
    private String url;
    private boolean flg;

    public GfMdl(String t, String u) {
        ttl = t;
        url = u;
        flg = false;
    }

    public String gtTtl() {
        return ttl;
    }

    public String gtUrl() {
        return url;
    }

    public void sttFlg(boolean f) {
        flg = f;
    }

    public boolean gtFlg() {
        return flg;
    }
}

```

```
package gfs.pckg;
```

```

import android.app.DownloadManager;
import android.content.Context;

```

```

import android.net.Uri;
import android.database.Cursor;

class GfDwnldr {

    private Context ctx;
    private DownloadManager dwnmgr;

    public GfDwnldr(Context c) {
        ctx = c;
        dwnmgr = (DownloadManager)
ctx.getSystemService(Context.DOWNLOAD_SERVICE);
    }

    public void dwnldGf(String lnk) {
        Uri r = Uri.parse(lnk);
        DownloadManager.Request rq = new
DownloadManager.Request(r);

rq.setNotificationVisibility(DownloadManager.Request.VISIBILITY_V
ISIBLE_NOTIFY_COMPLETED);
        long d = dwnmgr.enqueue(rq);
        trckDwnld(d);
    }

    private void trckDwnld(long d) {
        boolean rnnng = true;
        while (rnnng) {
            DownloadManager.Query q = new
DownloadManager.Query();
            q.setFilterById(d);
            Cursor c = dwnmgr.query(q);
            if (c != null && c.moveToFirst()) {
                int stts =
c.getInt(c.getColumnIndex(DownloadManager.COLUMN_STATUS));
                if (stts == DownloadManager.STATUS_SUCCESSFUL) {
                    rnnng = false;
                }
                if (stts == DownloadManager.STATUS_FAILED) {
                    rnnng = false;
                }
            }
            if (c != null) {
                c.close();
            }
        }
    }
}

```

```

package gfs.pckg;

import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;

class NtwrkStts {

    private Context ctx;

    public NtwrkStts(Context c) {
        ctx = c;
    }

    public boolean chkCnct() {
        ConnectivityManager m = (ConnectivityManager)
ctx.getSystemService(Context.CONNECTIVITY_SERVICE);
        if (m != null) {
            NetworkInfo n = m.getActiveNetworkInfo();
            if (n != null && n.isConnected()) {
                return true;
            }
        }
        return false;
    }
}

package gfs.pckg;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class VwActvty extends Activity {

    private TextView txtvw;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        txtvw = new TextView(this);
        txtvw.setText("Gf vw");
        setContentView(txtvw);
        prcssVw();
    }

    private void prcssVw() {

```

```

        for (int p = 0; p < 50; p++) {
            updTxt(p);
        }
    }

    private void updTxt(int v) {
        String s = txtvw.getText().toString();
        s = s + " * ";
        txtvw.setText(s);
    }
}

public void srchGfByTg(String tg) {
    if (tg == null) {
        return;
    }
    List<GfMdl> nwLst = new ArrayList<>();
    for (int i = 0; i < gflst.size(); i++) {
        GfMdl gf = gflst.get(i);
        String ttl = gf.gtTtl();
        if (ttl != null &&
ttl.toLowerCase().contains(tg.toLowerCase())) {
            nwLst.add(gf);
        }
    }
    gflst.clear();
    gflst.addAll(nwLst);
    rndrGfs();
}

public void srchGfByStr(String str) {
    if (str == null || str.length() == 0) {
        ldGfs();
        return;
    }
    List<GfMdl> fltrd = new ArrayList<>();
    for (int j = 0; j < gflst.size(); j++) {
        GfMdl md = gflst.get(j);
        String t = md.gtTtl();
        if (t != null &&
t.toLowerCase().indexOf(str.toLowerCase()) >= 0) {
            fltrd.add(md);
        }
    }
    gflst.clear();
    gflst.addAll(fltrd);
    rndrGfs();
}

```

```

public void rfshDt() {
    if (!ntwrkstts.chkCnct()) {
        shwMsg("Ntwrk lst");
        return;
    }
    hndlr.postDelayed(new Runnable() {
        @Override
        public void run() {
            List<GfMdl> nw = dtprvdr.gtDt();
            gflst.clear();
            gflst.addAll(nw);
            rndrGfs();
        }
    }, 600);
}

public void opnFlScrn(GfMdl gf) {
    if (gf == null) {
        return;
    }
    Intent nt = new Intent(this, VwActvty.class);
    nt.putExtra("lnk", gf.gtUrl());
    startActivity(nt);
}

public void chkPrmsn() {
    String prm =
    android.Manifest.permission.WRITE_EXTERNAL_STORAGE;
    int st = checkSelfPermission(prm);
    if (st !=
    android.content.pm.PackageManager.PERMISSION_GRANTED) {
        requestPermissions(new String[]{prm}, 100);
    }
}

public void hndlOrnttn(int w, int h) {
    if (w > h) {
        shwMsg("Lndscp md");
    } else {
        shwMsg("Prtrt md");
    }
}

public void clrGfs() {
    gflst.clear();
    lnrlt.removeAllViews();
}

public void srtGfs() {

```

```

        for (int i = 0; i < gflst.size(); i++) {
            for (int j = i + 1; j < gflst.size(); j++) {
                String a = gflst.get(i).gtTtl();
                String b = gflst.get(j).gtTtl();
                if (a != null && b != null && a.compareTo(b) > 0) {
                    GfMdl tmp = gflst.get(i);
                    gflst.set(i, gflst.get(j));
                    gflst.set(j, tmp);
                }
            }
        }
        rndrGfs();
    }

    public void stFlgAll(boolean f) {
        for (int i = 0; i < gflst.size(); i++) {
            gflst.get(i).sttFlg(f);
        }
    }

    public void dwnldAll() {
        for (int i = 0; i < gflst.size(); i++) {
            GfMdl gf = gflst.get(i);
            if (gf != null && gf.gtUrl() != null) {
                gfdwnldr.dwnldGf(gf.gtUrl());
            }
        }
    }

    public void vrfyDwnlds() {
        for (int i = 0; i < gflst.size(); i++) {
            boolean f = gflst.get(i).gtFlg();
            if (f) {
                shwMsg("Dwnld rdy");
            }
        }
    }

    public void prcssBgLd() {
        new Thread(new Runnable() {
            @Override
            public void run() {
                List<GfMdl> dt = dtprvdr.gtDt();
                hndlr.post(new Runnable() {
                    @Override
                    public void run() {
                        gflst.clear();
                        gflst.addAll(dt);
                        rndrGfs();
                    }
                });
            }
        });
    }

```

```
        }  
    });  
}  
}).start();  
}  
  
public void rsrvMmry() {  
    List<byte[]> mm = new ArrayList<>();  
    for (int i = 0; i < 10; i++) {  
        mm.add(new byte[1024]);  
    }  
    mm.clear();  
}
```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Створення програмної системи опрацювання даних для перегляду та завантаження анімованих зображень

Виконав
ст. гр. КНТ-222

Микола ДИДЯК

Керівник
к.т.н., доцент

Тетяна ЗАЙКО

Рисунок В.1 – Слайд 1

ОБ’ЄКТ, ПРЕДМЕТ ТА МЕТА РОБОТИ

Об’єктом дослідження є алгоритми та процеси обчислень, пов’язані з переглядом та завантаженням анімованих зображень.

Предмет дослідження – програмні системи опрацювання даних для перегляду та завантаження анімованих зображень.

Мета роботи – розробка програмної системи опрацювання даних для перегляду та завантаження анімованих зображень для скорочення витрат користувача на пошук графічного контенту.

2

Рисунок В.2 – Слайд 2

ЗАВДАННЯ РОБОТИ

- виконати аналіз предметної області та систем опрацювання даних для перегляду та завантаження анімованих зображень;
- здійснити проектування системи опрацювання даних для перегляду та завантаження анімованих зображень;
- створити систему опрацювання даних для перегляду та завантаження анімованих зображень;
- дослідити розроблену систему опрацювання даних для перегляду та завантаження анімованих зображень.

3

Рисунок В.3 – Слайд 3

Дослідження систем опрацювання даних для перегляду та завантаження анімованих зображень

Критерій порівняння	1Gallery	Movavi Photo Viewer	Aves Gallery
Простота інтерфейсу	+	+	+–
Підтримка анімованих зображень	+–	+–	+
Можливості пошуку та фільтрації	–	+–	+
Організація медіаколекцій	+–	+	+
Базові інструменти редагування	+–	+–	–
Орієнтація на мобільні пристрої	+	–	+

4

Рисунок В.4 – Слайд 4

ВИБІР МОВИ ПРОГРАМУВАННЯ

Критерій порівняння мов програмування	Мова програмування		
	Java	Python	C#
Платформна незалежність	+	+-	+-
Продуктивність під час опрацювання анімації	+	+-	+-
Наявність бібліотек для графіки та мультимедіа	+	+	+-
Зручність для створення ПЗ для перегляду та завантаження анімованих зображень	+	-	-
Придатність для кросплатформних застосунків	+	+	+-

5

Рисунок В.5 – Слайд 5

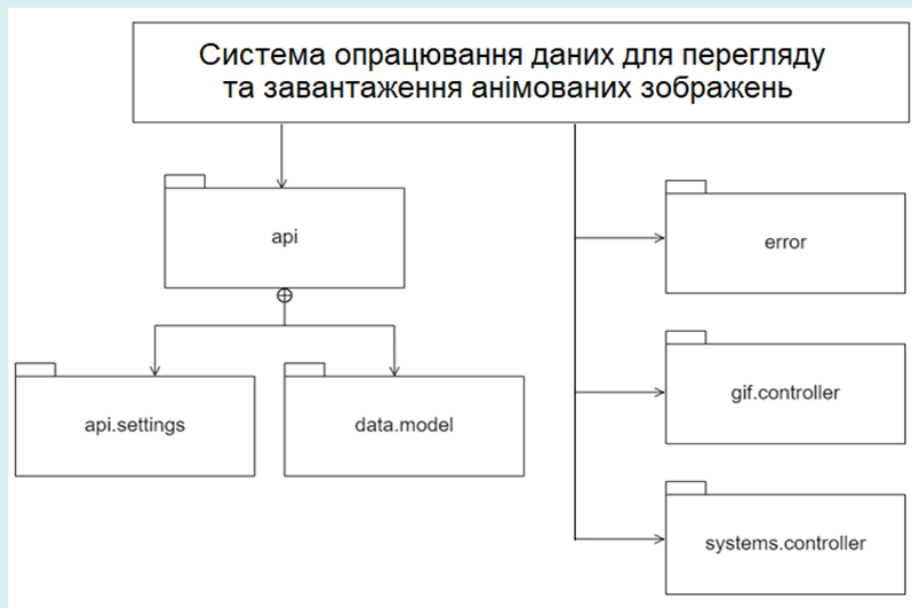
ВИБІР СЕРЕДОВИЩА РОЗРОБКИ

Критерій порівняння середовищ розробки	Середовища розробки		
	Android Studio	IntelliJ IDEA	NetBeans
Оптимізація для мобільної графіки та анімації	+	+-	-
Інтеграція з Android SDK	+	+-	-
Зручність створення інтерфейсу для мобільних застосунків	+	+-	+-
Інструменти профілювання ресурсів	+	+-	-
Можливості емуляції пристроїв	+	-	-

6

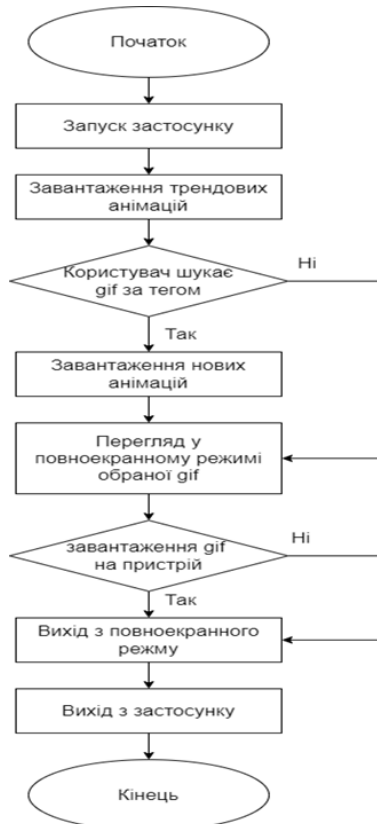
Рисунок В.6 – Слайд 6

СТРУКТУРА СИСТЕМИ ОПРАЦЮВАННЯ ДАНИХ ДЛЯ ПЕРЕГЛЯДУ ТА ЗАВАНТАЖЕННЯ АНІМОВАНИХ ЗОБРАЖЕНЬ



7

Рисунок В.7 – Слайд 7

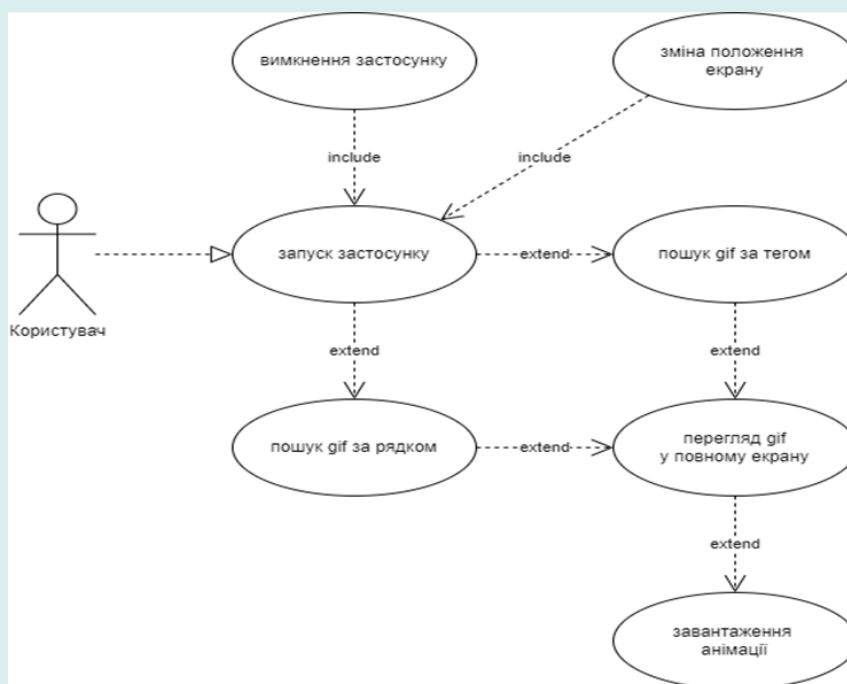


8

ФУНКЦІОНУВАННЯ СИСТЕМИ ОПРАЦЮВАННЯ ДАНИХ ДЛЯ ПЕРЕГЛЯДУ ТА ЗАВАНТАЖЕННЯ АНІМОВАНИХ ЗОБРАЖЕНЬ

Рисунок В.8 – Слайд 8

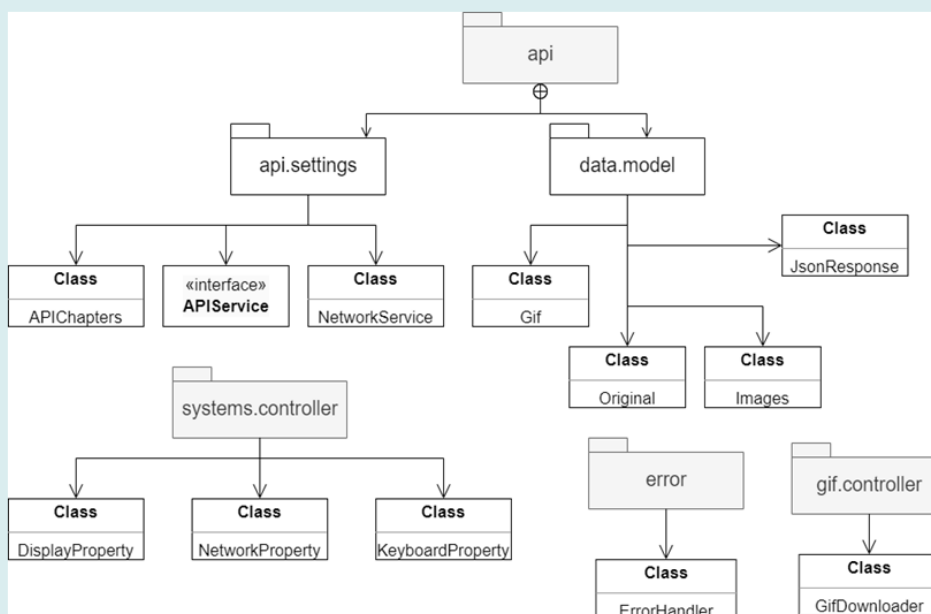
ДІАГРАМА ПРЕЦЕДЕНТІВ



9

Рисунок В.9 – Слайд 9

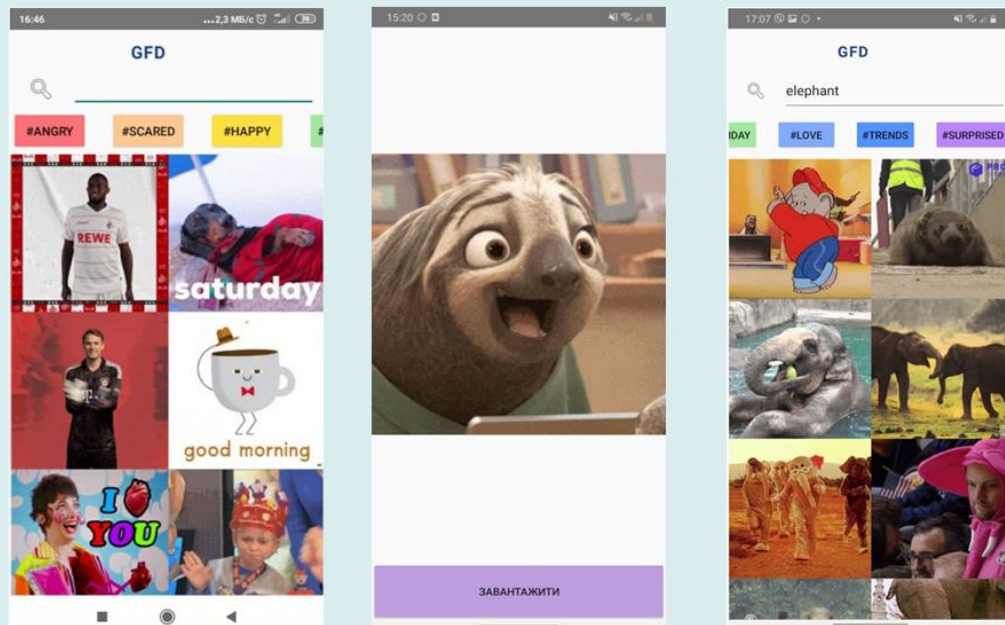
ДІАГРАМА КЛАСІВ



10

Рисунок В.10 – Слайд 10

РОБОТА З ПРОГРАМОЮ



11

Рисунок В.11 – Слайд 11

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процеси обчислень, пов'язані з переглядом та завантаженням анімованих зображень.

Для реалізації системи опрацювання даних для перегляду та завантаження анімованих зображень обрано мову програмування Java та середовище розробки Android Studio. Для зберігання та опрацювання даних використовуються JSON-структури даних, що забезпечує гнучкість, масштабованість і простоту підтримки системи, оскільки всі актуальні ресурси надходять із зовнішнього сервісу в уніфікованому форматі.

Розроблено систему опрацювання даних для перегляду та завантаження анімованих зображень.

Виконано тестування розробленої системи опрацювання даних для перегляду та завантаження анімованих зображень. Всі завдання роботи виконано.

12

Рисунок В.12 – Слайд 12