МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ДІАЛОГОВОГО ІНТЕРФЕЙСУ КОРИСТУВАЧІВ ДЛЯ АВТОМАТИЗОВАНОГО ВИКОРИСТАННЯ ГРАФІЧНИХ БІБЛІОТЕК МОВИ PYTHON У ПЛАТФОРМІ TELEGRAM»**

Виконала: студентка 2 курсу, групи 8.1228
спеціальності 122 комп'ютерні науки
освітньої програми комп'ютерні науки
(шифр і назва спеціальності)
А.В. Коргун
(ініціали та прізвище)

Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н. Решевська К.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри програмної інженерії, доцент, к.т.н., Мухін В.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 комп'ютерні науки

Освітня програма комп'ютерні науки

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук, к.ф.-м.н., доцент

_____ Борю С.Ю.

(підпис)

« 02 » вересня 2019 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Коргун Анастасії Вікторівні

(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Розробка діалогового інтерфейсу користувачів для автоматизованого використання графічних бібліотек мови Python у платформі Telegram

керівник роботи (проекту) Решевська Катерина Сергіївна, к.т.н, доцент

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 29 » травня 2019 року № 811-с

2. Строк подання студентом роботи 27.12.2019

3. Вихідні дані до роботи 1) Постановка задачі.
2) Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1) Постановка задачі.
2) Аналіз графічних бібліотек обробки зображень.
3) Модель та алгоритми графічних бібліотек Python.
4) Програмна реалізація проекту.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень).
Презентація

6. Консультанти розділів роботи

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

7. Дата видачі завдання 02.09.2019

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	02.09.2019	
2.	Збір вихідних даних.	21.09.2019	
3.	Обробка методичних та теоретичних	01.10.2019	
	джерел.		
4.	Розробка першого і другого розділу.	20.10.2019	
5.	Розробка третього розділу.	23.10.2019	
6.	Оформлення та нормоконтроль	27.12.2019	
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	14.01.2020	

Студент _____ А.В. Коргун
(підпис) (ініціали та прізвище)

Керівник роботи _____ К.С. Решевська
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка діалогового інтерфейсу користувачів для автоматизованого використання графічних бібліотек мови Python у платформі Telegram»: 48 с., 27 рис., 13 джерел, 2 додатки.

БОТ, ГРАФІЧНІ БІБЛІОТЕКИ, ЗОБРАЖЕННЯ, ІНТЕРНЕТ, МАТРИЦІ, СОЦІАЛЬНА МЕРЕЖА, HTTP, TELEGRAM, TELEGRAMBOTAPI, TORNADO WEB SERVER, PYTHON.

Об’єкт дослідження – дослідження графічних бібліотек мови програмування Python.

Предмет дослідження – автоматизована обробка графічних файлів з використанням графічних бібліотек Python.

Мета роботи: розробити автоматизовану систему обробки зображення за допомогою графічних бібліотек обробки зображення за допомогою соціальної мережі Telegram розміщеного в боті у всесвітній мережі Інтернет.

Метод дослідження – моделювання програмних засобів, конструювання програмних засобів та їх тестування, аналізу вимог до застосування веб-додатку в Telegram.

На сьогоднішній день системи обробки зображення стали невід’ємним атрибутом навчальних закладів, та в повсякденному житті. Вони дозволяють студентам отримати зображення більш чіткої якості, або більш розмитим, в залежності від потреб. Але такі системи потребують доопрацювання, так як якість системи залежить від швидкості обробки, та якості обробленого зображення.

У роботі розроблено онлайн-сервіс автоматизованої системи обробки зображень у системі Telegram за допомогою графічних бібліотек мови програмування Python. Для досягнення мети було проведено аналіз предметної області, обрані графічні бібліотеки і платформу реалізації. Реалізовано веб-додаток з використанням pyTelegramBotApi та мови

програмування Python, оформлено відповідну документацію. В результаті роботи розроблено діалоговий інтерфейс користувачів для автоматизованого використання графічних бібліотек задля обробки зображення.

SUMMARY

Master's qualifying paper «A Dialog User Interface Development of the Automated Use of Python Graphical Libraries in the Telegram Platform»: 48 pages, 29 figures, 13 references, 2 supplements.

BOT, GRAPHIC LIBRARIES, IMAGES, INTERNET, MATRIX, SOCIAL NETWORK, HTTP, TELEGRAM, TELEGRAMBOTAPI, TORNADO WEB SERVER, PYTHON.

The object of study is to study Python programming libraries.

The subject of the study is automated processing of image files using Python graphics libraries.

The aim of the study is to develop the automated image processing system using image-processing libraries in the Telegram social network hosted on the bot on the World Wide Web.

The methods of research are modeling software, designing and testing software, analyzing requirements for using a web application in Telegram.

Nowadays, image-processing systems have become an integral attribute of educational institutions and in everyday life. They allow students to obtain images of better quality, or more blurry, depending on the needs. However, such systems require refinement, as the quality of the system depends on the speed of processing and, correspondently, the quality of the processed image.

In this work, we have developed an online service for an automated image processing system in Telegram, using Python programming libraries. To achieve the goal, a domain analysis was carried out; graphic libraries and an implementation platform were selected, a Web application with pyTelegramBotApi and Python programming language was implemented, all the required documentation was written. As a result, a dialogical user interface was successfully developed for automated use of image libraries for image processing.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Скорочення

МФ	Медіанний фільтр
JSON	JavaScript Object Notation
HTTP	HyperText Transfer Protocol
API	Application Programming Interface
OPENCV	Open Source Computer Vision Library
PIL	Python Imaging Library

Терміни

Telegram – месенджер, програмне забезпечення для смартфонів, повідомленнями та різноманітними файлами, зокрема графічними файлами та відеофайлами, а також безкоштовно телефонувати іншим користувачам програми.

Бот – спеціальна програма, що виконує автоматично і/або за заданим розкладом які-небудь дії через ті ж інтерфейси, що й звичайний користувач. Під час обговорення комп'ютерних програм термін уживається в основному в застосуванні до Інтернету.

Бот в telegram – це користувач з ім'ям, описом і аватаркою, ім'я якого закінчується на «bot». Для доступу достатньо додати його в адресну книгу і натиснути кнопку «Start».

Telegrambotapi – це HTTP-інтерфейс, створений для розробників для створення ботів в Telegram.

Python – це інтерпретована об'єктно-орієнтована мова програмування високого рівня. Вона є однією з тих рідкісних мов програмування, які є одночасно і простими, і потужними для проектів різного напрямк

Tornado Web Server – веб-сервер та фреймворк, написаний на Python, який легко розширюється та не блокується під час запитів.

Фільтр – поняття в електроніці, що обробляє цифровий сигнал з метою відокремлення та/або придушення певних частотних складових цього сигналу.

Матриця – математичний об’єкт записаний у вигляді прямокутної таблиці чисел, який допускає операції (додавання, віднімання, множення та множення на скаляр).

Графічні бібліотеки – це бібліотеки програм, призначені для допомоги у відтворенні комп'ютерної графіки монітору.

ВСТУП

Одним із найпопулярніших засобів роботи з зображенням є його обробка за допомогою спеціальних фільтрів, найчастіше з використанням програмних засобів смартфонів або інших мобільних пристроїв. Саме це робить обране направлення актуальним та затребуваним напрямком у сучасному світі IT-технологій.

Особливо дана система матиме попит як і серед студентів, так і серед звичайних користувачів платформи Telegram, так як вона корисна для них як засіб обробки конспектів, та обробки, якщо зображення не передає в повній мірі потрібну інформацію.

Предметом дослідження виступає автоматизована обробка зображення, графічні бібліотеки та діалоговий інтерфейс.

Об'єкт дослідження: дослідження графічних бібліотек мови програмування Python.

Мета дослідження: розробка програмного забезпечення з обробки зображення з використанням графічних бібліотек мови програмування Python з мінімальними витратами часу за допомогою кросплатформенного месенджера Telegram розміщеного в боті в всесвітній мережі Інтернет.

Структура роботи – дипломний проект складається зі вступу, трьох розділів (підрозділи), висновків, літератури, додатки.

Перший розділ містить загальні відомості про зображення, графічні бібліотеки, обробку, алгоритми обробки, переваги та недоліки.

Розглянуті також основні поняття та визначення методів розробки. Перший розділ закінчується технічним завданням.

Другий розділ містить опис вимог та структуру TelegramBotApi для автоматизації системи обробки зображення, методи створення діалогового інтерфейсу, скриншоти макетів бота.

Третій розділ містить опис особливостей та структури реалізації системи діалогового інтерфейсу для автоматизованого використання графічних бібліотек обробки зображення.

1 АНАЛІЗ ГРАФІЧНИХ БІБЛІОТЕК ОБРОБКИ ЗОБРАЖЕНЬ

1.1 Поняття зображення

Цифрове зображення – це модель реального або синтезованого (створеного штучно) зображення, що зберігається в пам’яті ЕОМ у вигляді сукупності цифрових кодів. Режим, при якому користувач у реальному часі може впливати на весь процес формування зображення, тобто зміни в зображення можна вносити безпосередньо в процесі його відтворення (в режимі діалогу), називається інтерактивним [5].

Обробкою цифрового зображення також можна назвати і редагування зображення (лат. Redactus – приведений в порядок) – це зміна оригіналу зображення класичними або цифровими методами. Часто під редагуванням або обробкою розуміють і ретушування (ретуш від фр. Retoucher – підмальовувати, підправляти). Метою редагування/обробки є корекція дефектів, підготовка до публікації, рішення творчих завдань.

Крім статичних двомірних зображень, можна обробляти також і послідовність зображень.

Існують такі види обробки цифрового зображення:

- ретуш;
- реставрація;
- художня обробка і стилізація;
- спеціальна обробка.

Головною задачею редагування зображення – це усунення його дефектів.

1.2 Мова програмування Python

Python - мова програмування високого рівня, легка в засвоєнні, об'єктно-орієнтована, модульна і легко читається. Python широко використовується в освітній галузі, для наукових обчислень, великих даних та машинного навчання, в Інтернеті та комп'ютерній графіці, графічному інтерфейсі, іграх та інших сферах.

Оскільки «екосистема» Python величезна, існує безліч бібліотек, які спрощують програмування на цій мові. Такі бібліотеки полегшують виконання певних завдань без необхідності писати зайвий код.

Для полегшення роботи з графічними бібліотеками використовують сервіс Uploadcare[2]. Сервіс модифікації зображень в Uploadcare – це сервер, до якого приходить HTTP-запит з ідентифікатором зображення і якимись операціями, які потрібно виконати клієнту. Сервер повинен виконати операції і як можна швидше дати відповідь. Як клієнт найчастіше виступає браузер.

Весь сервіс можна описати як обгортку навколо графічної бібліотеки. Саме від якості, продуктивності і зручності використання графічної бібліотеки залежить якість всього проекту.

Коли ми говоримо про роботу з зображеннями, перше, що потрібно взяти до уваги – це продуктивність, тому що для ефективної роботи програмного застосунку важливо знати наскільки дана бібліотека задовольнить роботу.

Продуктивність – це ефективність використання ресурсів під час створення різних різниї програмних засобів і надання послуг. Не можна сказати, що одна бібліотека працює швидше за іншу. У кожній бібліотеці є набір функцій, і кожна функція працює з різною швидкістю[6].

Відповідно, можна сказати про те, що продуктивність однієї функції вище або нижче в конкретній бібліотеці. Або при наявності програми, якому потрібен якийсь набір функціональності, зробити бенчмарк саме під цю

функціональність, і говорити, що для застосування деяка бібліотека працює швидше (повільніше).

1.3 Бібліотека OpenCV

OpenCV - найпопулярніша бібліотека комп'ютерної графіки. Бібліотека включає більше 1000 функцій і алгоритмів. Вона розробляється с 1998 року, спочатку в компанії Інтел, тепер в Itseez за активної участі спільноти. Про високу популярність бібліотеки свідчить кількість завантажень, їх більш 6000000 завантажень (без урахування svn / git трафіку)[3].

Бібліотека розповсюджується за ліцензією BSD, що означає, що її можна вільно і безкоштовно використовувати як у відкритих проектах з відкритим кодом, так і в закритих, комерційних проектах. Бібліотеку не обов'язково копіювати цілком в свій проект, можна використовувати деякі фрагменти коду. Єдина вимога ліцензії – наявність в супроводжуючих матеріалах копії ліцензії OpenCV.

Бібліотека OpenCV реалізує, як правило, тільки базові операції, які використовуються в комп'ютерній графіці. Таким чином, її можна розглядати як в цілому низькорівневую бібліотеку комп'ютерної графіки[7]. Для вирішення серйозних завдань необхідно на основі наданих бібліотекою «цеглинок» створювати свої складні додатки.

Нижче представлена загальна схема обробки зображення, призначеного для вирішення завдань комп'ютерної графіки (див. рис. 2.1).

Відбувається спочатку захват зображення, яке передається на обробку при цьому виділяються особливості. Після сегментування уже йде вимірювання швидкості роботи алгоритму і прийняття рішення.

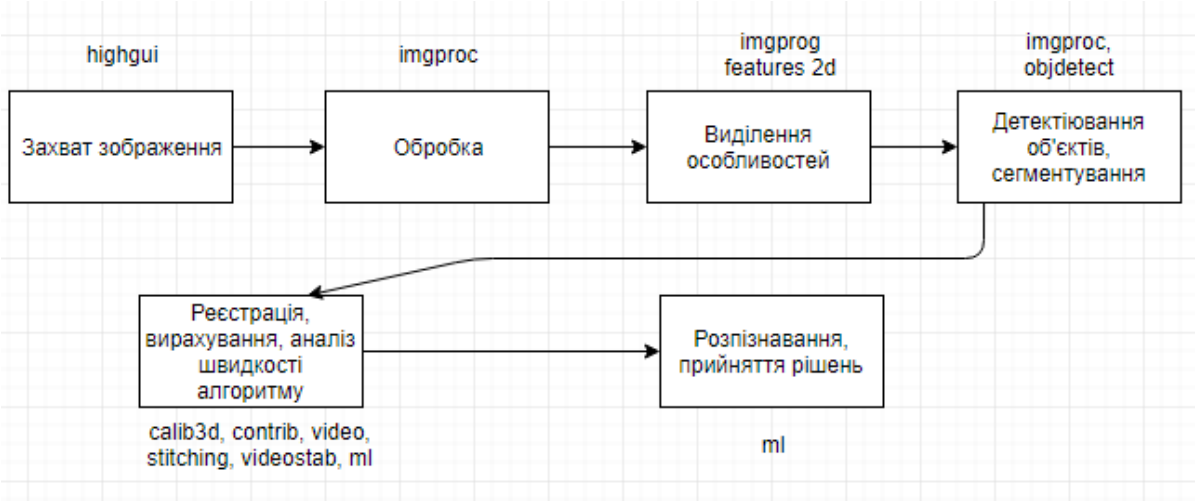

Рисунок 2.1 – Схема обробки зображення

1.4 Бібліотека Pillow

Pillow це бібліотека PIL (Python Image Library) заснована на коді PIL, а потім перетворена в поліпшену, сучасну версію. Надає підтримку при відкритті, управлінні і збереженні багатьох форматів зображення.

Підтримувані типи файлів цієї бібліотеки: BMP, EPS, GIF, IM, JPEG, MSP, PCX PNG, PPM, TIFF, WebP, ICO, PSD, PDF. Деякі типи файлів можливі тільки для читання, в той час як інші доступні тільки для написання.

Основні функції знаходяться в модулі Image. Можна створювати екземпляри цього класу декількома способами. Шляхом завантаження зображень з файлів, обробки інших зображень, або створення зображень з нуля. Необхідно імпортувати модулі Pillow, які потрібно використовувати (див. рис. 2.2).

Необхідно використовувати метод open для ідентифікації файлу на комп'ютері, а потім завантажити ідентифікований файл за допомогою myfile.load()[7].

Як тільки зображення буде завантажено, з ним можна працювати. Часто використовується блок try except при роботі з файлами (див. рис. 2.3).

```
from PIL import Image

myimage = Image.open(filename)

myimage.load()
```

Рисунок 2.2 – Завантаження модуля

```
from PIL import Image, ImageFilter
try:
    original = Image.open("Lenna.png")
except FileNotFoundError:
    print("Файл не найден")
```

Рисунок 2.3 – Завантаження зображення

Коли зчитуються файли з диска за допомогою функції open (), не потрібно знати формат файлу. Бібліотека автоматично визначає формат, заснований на отриманому файлі. Тепер, коли є об'єкт Image, можна використовувати доступні атрибути для перевірки файлу (див. рис 2.4).

Модуль Pillow надає наступний набір визначених фільтрів для поліпшення зображення: BLUR, CONTOUR, DETAIL, EDGE_ENHANCE, EDGE_ENHANCE_MORE, EMBOSS, FIND_EDGES, SMOOTH, SMOOTH_MORE SHARPEN.

```
print("The size of the Image is: ")
print(original.format, original.size, original.mode)
```

Рисунок 2.4 – Атрибути перевірки файлу

1.5 Бібліотека VIPS

Ще одна бібліотека, на яку варто звернути увагу – VIPS. Основна ідея VIPS в тому, що для роботи з зображенням не потрібно завантажувати все зображення в пам'ять. Бібліотека може завантажувати деякі невеликі фрагменти, обробляти їх і зберігати. Таким чином, для обробки гігапіксельних зображень не потрібно витрачати гігабайти пам'яті.

Це досить стара бібліотека – 1993 року, але вона обігнала свій час. Про неї довгий час було мало чутно, але останнім часом для VIPS стали використовувати в різних мовах, в тому числі: Go, Node.js, Ruby, Python.

У даному розділі розглянуто поняття цифрового зображення. За підсумком розділу зроблено висновок, що існуючі редактори обробки зображень мають свої недоліки, та незручні в використанні[5].

Онлайн програми не потребують встановлення програмного додатку на носій, проте використовують багато трафіку Інтернету та повільно передають результат обробки. Якщо говорити про офлайн редактори, то вони займають багато пам’яті на жорсткому пристрої та більшість інструментів обробки зображень мають платну основу[8].

Користувачі звикли редагувати свої зображення за допомогою спеціальних редакторів, з використанням смартфону або іншого мобільного пристрою. Виникає проблема обробки зображення з дефектами з використанням матричних фільтрів.

Сьогодні, дана система матиме попит так як вона корисна для студентів як засіб обробки конспектів, коли деяка інформація відображається нечітко, та якщо зображення не передає в повній мірі потрібну інформацію.

На даний час існує багато додатків, які реалізовані використовуючи недостатню математичну базу, тобто не розглядаються складні математичні задачі, і тому потенціал комп’ютерної графіки реалізується не повністю.

Отже, для поставленої мети сформульовані задачі розробки веб-сервісу для «швидкої» онлайн обробки зображень за допомогою матричних фільтрів

в кросплатформеному месенджері Telegram з використанням мови програмування Python, а саме:

- засвоїти принцип роботи кросплатформенного месенджера Telegram для розробки автоматизованої системи;
- зібрати і проаналізувати методичний та теоретичний матеріали про алгоритми обробки зображення та їх використання у бібліотеках Python;
- дослідити особливості створення автоматизованого програмного забезпечення;
- вивчити принципи створення діалогового інтерфейсу та системи бот в соціальній мережі та месенджері для обробки зображення;
- розглянути способи створення діалогового інтерфейсу бота в меседжері Telegram;
- розробити web-ресурс обробки зображення з використанням графічних бібліотек;
- дослідити швидкість роботи графічних бібліотек;
- розробити діалоговий інтерфейс обробки зображення.

2 МОДЕЛЬ ТА АЛГОРИТМИ ГРАФІЧНИХ БІБЛІОТЕК PYTHON

Процес фільтрації фотографій полягає в усуненні отриманих природним шляхом дефектів зображення у вигляді тріщин, плям, заломів, відшарування емульсії, подряпин, втрати кольоровості, слідів від друкарських предметів, видалення фактури старого паперу, зернистості.

Як правило, реставрацію фото виробляють по заданому алгоритму, тобто в кілька кроків. В даній роботі будуть розглядатися алгоритми обробки зображення графічних бібліотек.

2.1 Методи створення діалогового інтерфейсу

Діалоговий інтерфейс можна створити двома способами: за допомогою програмування та з використанням спеціального бота, який за допомогою деяких команд створює графічний інтерфейс[10].

Якщо створювати бота за допомогою програмування, необхідно підключити бібліотеку TelegramBotApi, зареєструвати його за допомогою згенерованого Token, і додати в метод message.text функціоналу (див. рис. 2.1).

```
if message.text == "Привет": bot.send_message(message.from_user.id, "Привет, чем я могу тебе
помочь?") elif message.text == "/help": bot.send_message(message.from_user.id,
"Напиши привет") else: bot.send_message(message.from_user.id, "Я тебя не понимаю.
Напиши /help.")
```

Рисунок 2.1 – Привітання бота

Створюючи діалоговий інтерфейс за допомогою @ManyBot необхідно задати команду /commands і створити команду /hello, відповіддю на яку буде привітання бота. Даний бота має обмежені можливості, рекламу, тому в даній роботі використовуватись не буде (див. рис. 2.2).

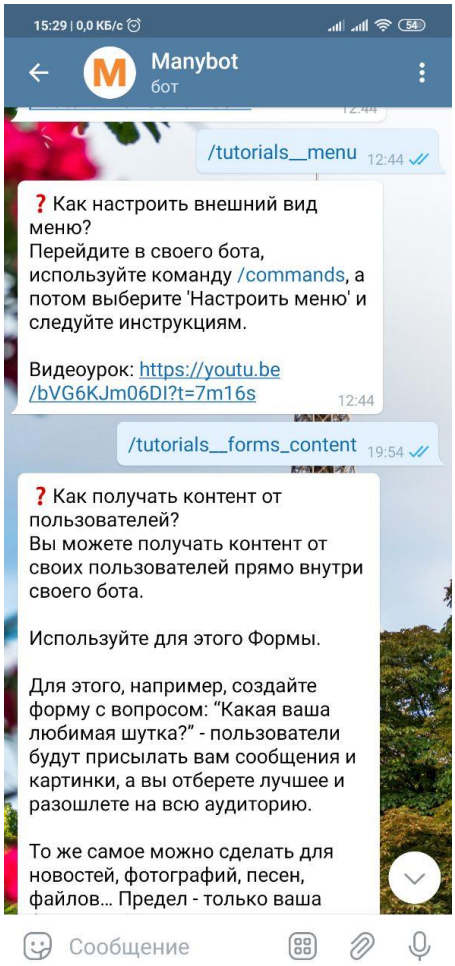

Рисунок 2.2 – Діаграма варіантів використання

2.2 Опис функціональної та динамічної моделі системи

2.2.1 Діаграма варіантів використання

Діаграма варіантів використання - це вихідне концептуальне подання або концептуальна модель системи в процесі її проектування і розробки.

Дана діаграма показує всі можливі дії користувача з програмою (див. рис. 2.3).

Основні цілі, які може досягти користувач**:**

- завантаження зображення;
- вибір необхідного фільтра;
- отримати оброблене зображення;
- використання інструкції.

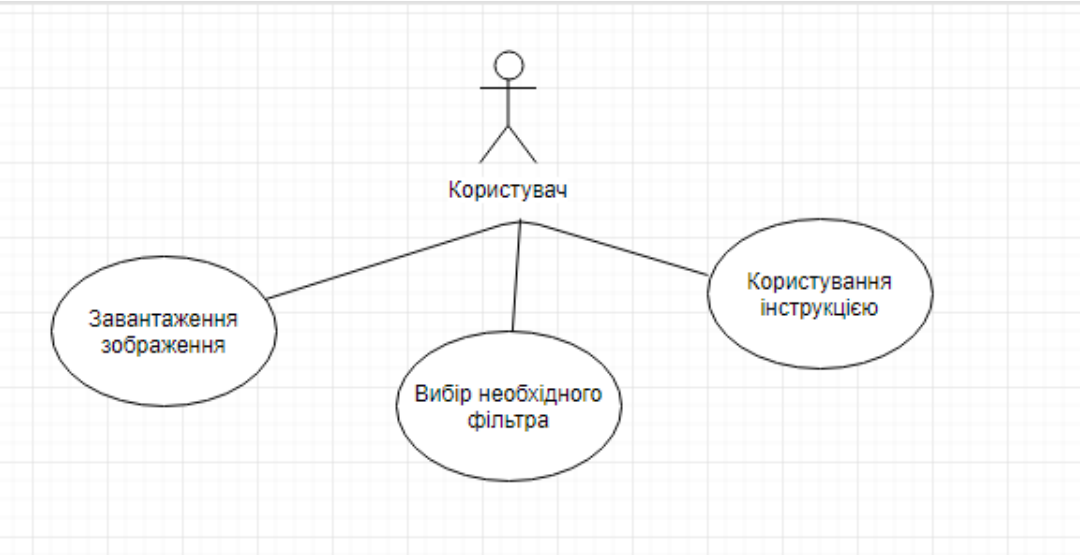

Рисунок 2.3 – Діаграма варіантів використання

Наступна зображена діаграма показує можливості адміністратора бота (див. рис. 2.4).

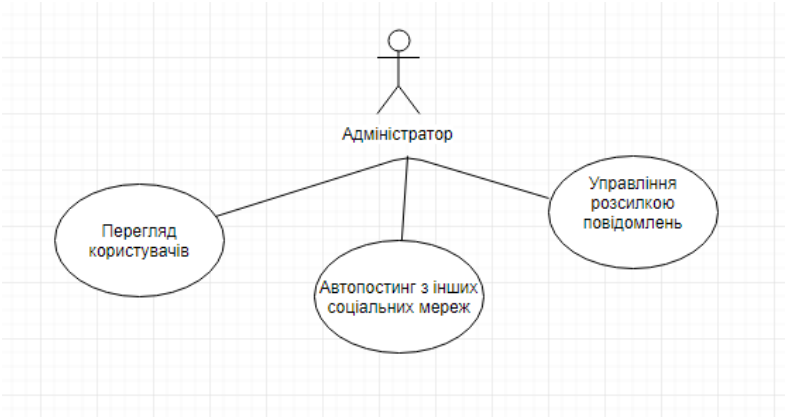

Рисунок 2.4 – Діаграма можливостей адміністратора

Адміністратор має можливість управляти розсилкою повідомлень, редагувати список користувачів, управляти автопостингом з інших соціальних мереж.

2.2.2 UML-діаграма послідовностей

UML – графічна мова для специфікації, візуалізації, проектування та документування систем.

Зображена діаграма показує послідовність роботи обробки зображення (див. рис. 2.5).

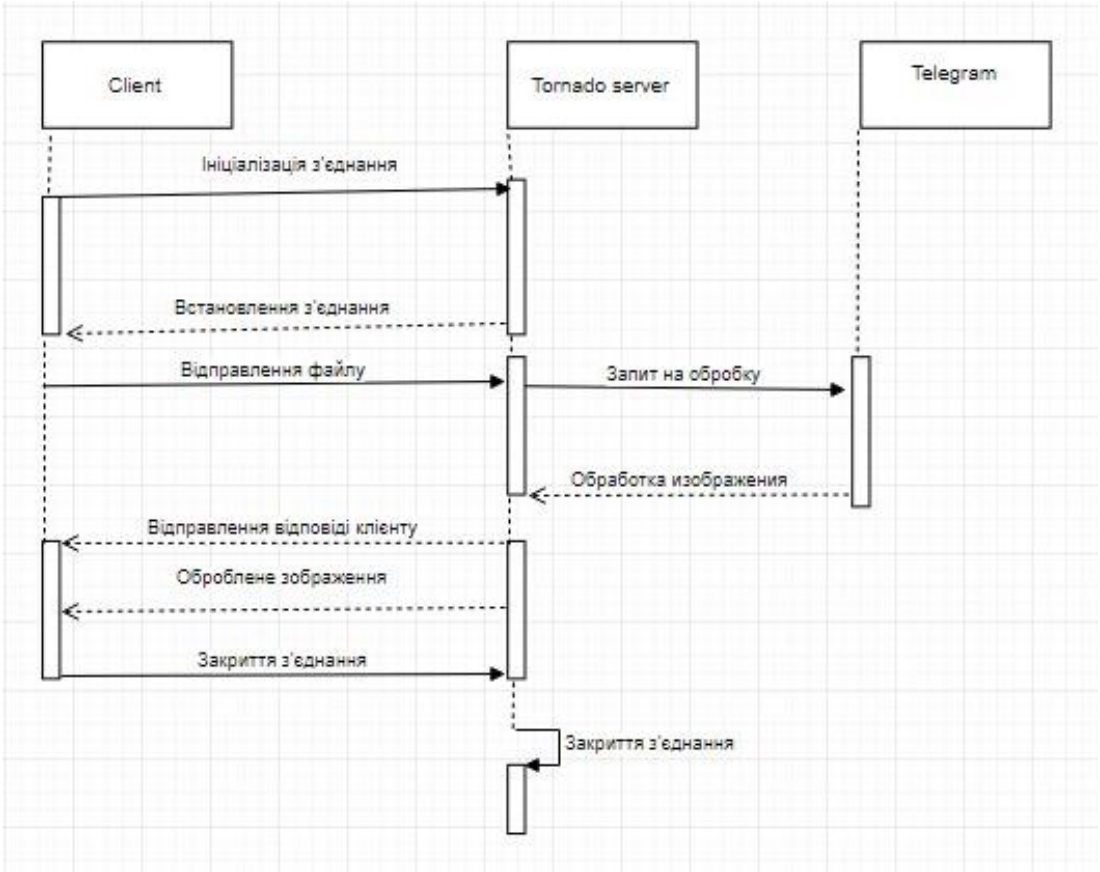

Рисунок 2.5 – Діаграма послідовності дій

Клієнт встановлює з’єднання з сервером, отримує підтвердження про з’єднання, після встановлення з’єднання відправляє файл на обробку. Після

відправлення клієнтом файлу на обробку, Tornado Server, відправляє Telegram запит на обробку зображення. Зображення обробляється та пересилається на Tornado Server [10]. Tornado Server відповідає клієнту на запит та повертає оброблене зображення клієнту. Після обробки зображення, ініціалізується завершення з'єднання з сервером.

2.4 Інструкція використання бота Photo Editor

Для того щоб відфільтрувати вибране зображення необхідно знайти в платформі Telegram за допомогою пошукової кнопки даний бот за назвою «Photo Editor». Підписатися на нього і ознайомитися з короткою інформацією про нього. Далі необхідно вибрати в випадаючому списку «Головне меню», «Operation». Тоді з'явиться доступ до фільтрів обробки зображення. Вибирати необхідний фільтр, а саме: «Black and White», «Blur», «Sharpness», «Negative», «Contour». Далі потрібно завантажити зображення, і очікувати відповіді. Якщо користувачеві щось незрозуміло в інтерфейсі програми, можна звернутися до інструкції, яка вбудована в даний бот. Її можна знайти за допомогою команди «/helpbot». Бот Photo Editor має рівень використання звичайного користувача (див. рис 2.6) та адміністратора (див. рис 2.7).

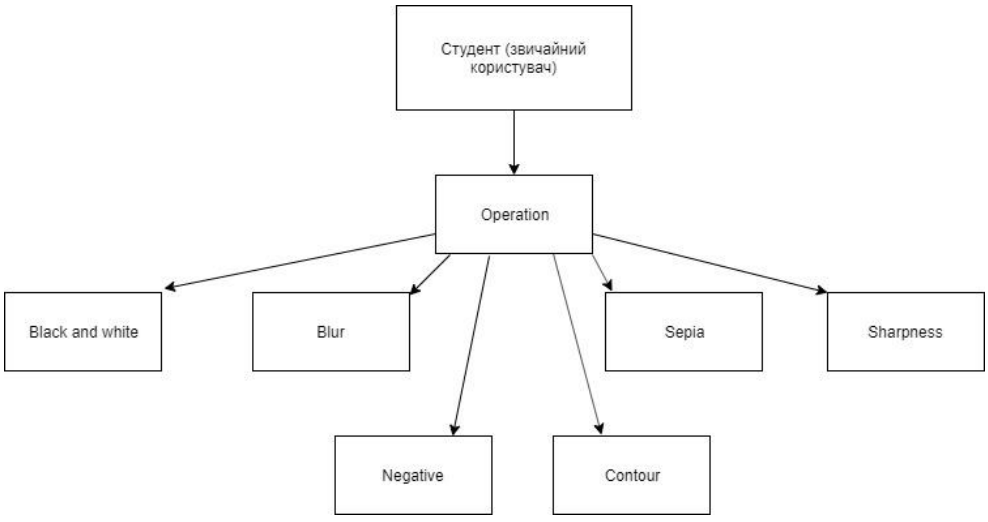

Рисунок 2.6 – Діаграма використання бота звичайним користувачем

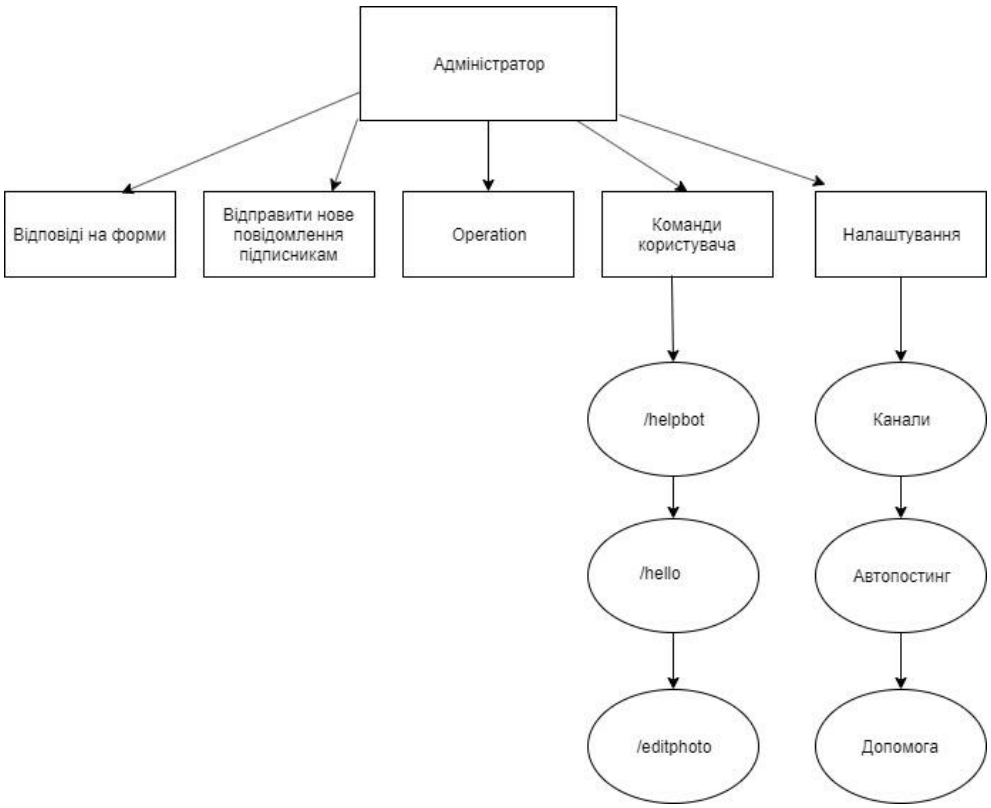

Рисунок 2.7 – Діаграма використання бота адміністратором

За підсумком розділу зроблено висновок, що крім розглянутих видів фільтрів існує безліч різних класифікацій, але немає єдиної системи класифікації. Тому слід провести власну класифікацію і надалі спиратись на неї при реалізації алгоритму.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЕКТУ

3.1 Проектування інформаційної системи

3.1.1 Найменування і область застосування

Даний діалоговий додаток обробки зображення призначений для фільтрації графічних файлів за допомогою фільтрів онлайн.

3.1.2 Підстава для розробки

Підставою для розробки є завдання до кваліфікаційної роботи.

Найменування теми розробки – «Розробка діалогового інтерфейсу користувачів для автоматизованого використання графічних бібліотек мови Python у платформі Telegram».

3.1.3 Призначення розробки

Даний проект призначений для вирішення наступних завдань:

- фільтрації зображень за допомогою фільтра збільшення різкості онлайн;
- фільтрації зображень за допомогою Гаусового фільтра розмиття онлайн;
- фільтрації зображень за допомогою фільтра негатив онлайн;
- фільтрації зображень за допомогою фільтра контур онлайн;
- фільтрації зображень за допомогою чорно-білого фільтра онлайн.

3.1.4 Особливості реалізації

Розробка власного додатку для діалогового інтерфейсу обробки зображень не потребує великих грошових витрат на розробку, програмування, дизайн та розкрутку бота, але на виході даний проект приносить невеликий прибуток так як цей ресурс безкоштовний та не має вбудованої системи платежу.

3.2 Засоби реалізації

Під час розробки бота для діалогового інтерфейсу обробки зображень за допомогою графічних бібліотек були використані такі засоби реалізації:

- Мова програмування Python3;
- Прикладний програмний інтерфейс для взаємодії з Telegram, TelegramBotApi 3.5.

3.3 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація:

а) об’єм оперативного пристрою, що запам’ятовує, 512МБ і більше;

б) об’єм вільного місця внутрішньої пам’яті на диску не менше 1ГБ;

в) Доступ до мережі Інтернет.

Конфігурація, що рекомендується:

а) версія Android 6.0 і вище;

б) об’єм вільного місця, внутрішньої пам’яті, що запам’ятовує, 1ГБ і більше;

в) об’єм вільного місця на жорсткому диску 2ГБ і більше.

3.4 Вимоги до програмної сумісності

Веб-сайт повинен правильно працювати в будь-якому веб-браузері (Internet Exploer, Opera, Chrome, Mozilla Firefox, Safari тощо), або в додатку Telegram завантаженому з PlayМаркет або App Store.

3.5 Структура інтерфейсу додатку

Інтерфейс інформаційної системи для автоматизації обробки зображення матричними фільтрами являє собою веб-додаток, який складається з наступних компонентів:

- стандартна кнопка «СТАРТ» – дозволяє підписатися на даний бот;
- «Головна сторінка» – містить 1 кнопку «Operations», яка містить вкладки фільтрів «Black and White», «Blur», «Sharpness», «Negative», «Contour»;
- фільтр «Black and White»–перетворює зображення в чорно-білий вигляд;
- фільтр «Blur» – оброблює зображення, Гаусовий фільтр розмиття;
- фільтр «Sharpness» –фільтр збільшення різкості зображення;
- фільтр «Negative» – оброблює зображення, фільтр негативу;
- фільтр «Contour» – фільтр, який залишає в зображенні тільки контур;
- вкладка «Help_users»–містить інструкцію користування ботом;
- команда «/start»–сповіщує про початок роботи з ботом;
- команда «/hello»–привітання бота, імітація людського спілкування.

Для того щоб почати роботу з ботом необхідно натиснути «СТАРТ». Дана кнопка дозволяє почати роботу з ботом та автоматично відображається короткий опис використання даного інструменту.

Головна сторінка додатку виглядає наступним чином: (див. рис. 3.1).

Він є головним інтерфейсом взаємодії між користувачем та ботом. З нього за допомогою команди, можна перейти до будь-якої іншої сторінки. Достатньо ввести назву фільтру, що бажає використати користувач й система перейде до використання обраного фільтру з подальшими діями.

Рисунок 3.1 – Макет «Головної сторінки» веб-додатку

При завантаженні зображення і натисканні на кнопку «Black and White» відкривається макет який наведено нижче (див. рис. 3.2).

Користувач вводить в текстове поле назву фільтру «Black and White», після цього завантажує зображення, що бажає змінити. Після завершення

обробки зображення, оброблене зображення виводиться на екран та додаток стає доступним для інших операцій.

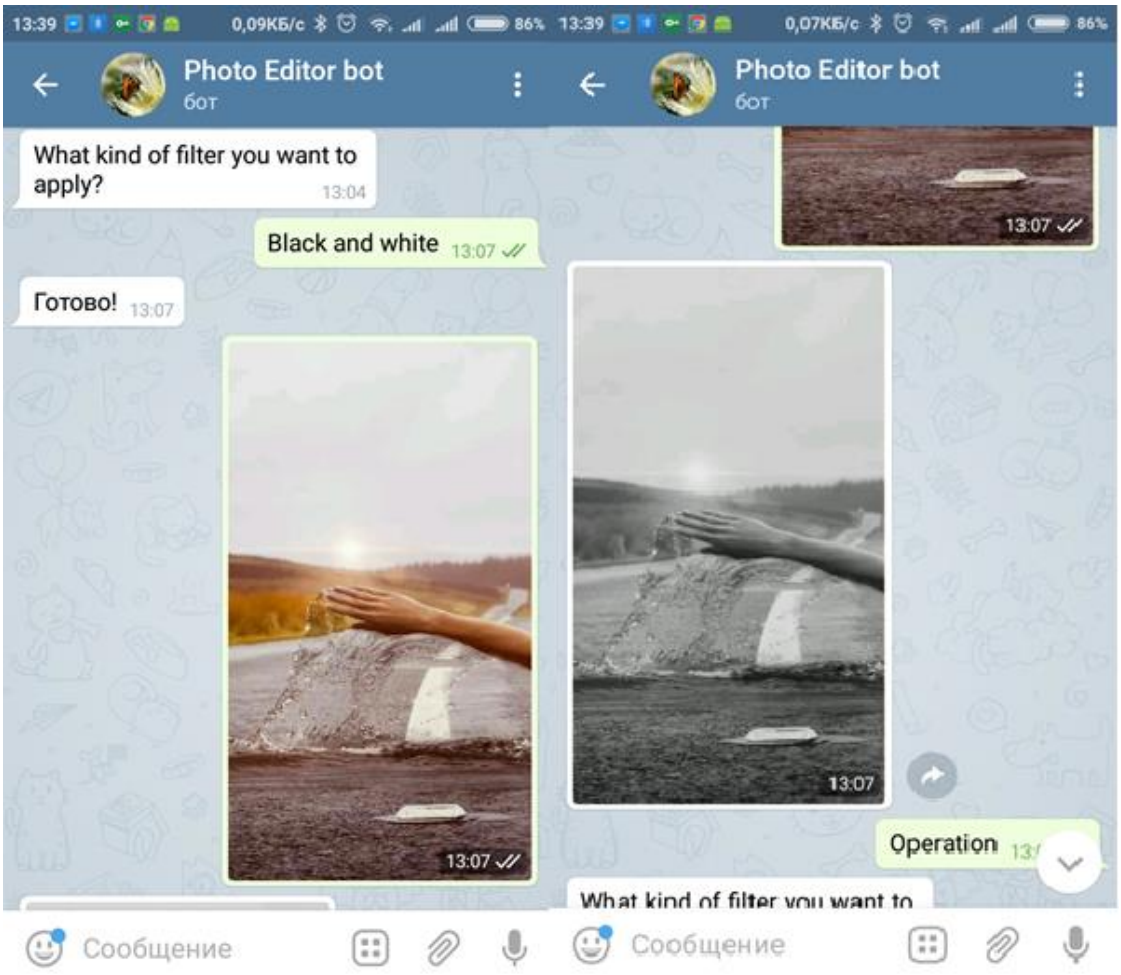

Рисунок 3.2 – Макет вкладки «Black and White»

При завантаженні зображення і натисканні на кнопку «Blur» відкривається макет який наведено нижче (див. рис. 3.3).

Користувач вводить в текстове поле назву фільтру «Blur», після цього завантажує зображення, що бажає змінити. Після завершення обробки зображення, оброблене зображення виводиться на екран та додаток стає доступним для інших операцій.

При завантаженні зображення і натисканні на кнопку «Sharpness» відкривається макет який наведено нижче (див. рис. 3.4).

Користувач вводить в текстове поле назву фільтру «Sharpness», після цього завантажує зображення, що бажає змінити. Після завершення обробки

зображення, оброблене зображення виводиться на екран та додаток стає доступним для інших операцій.

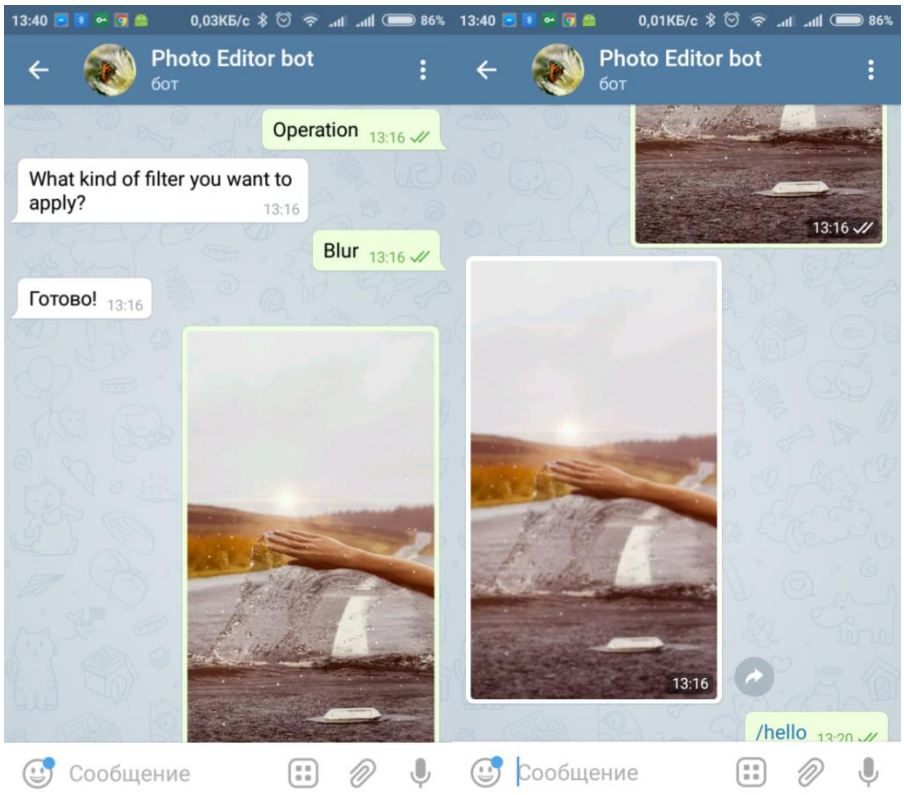

Рисунок 3.3 – Макет вкладки «Blur»

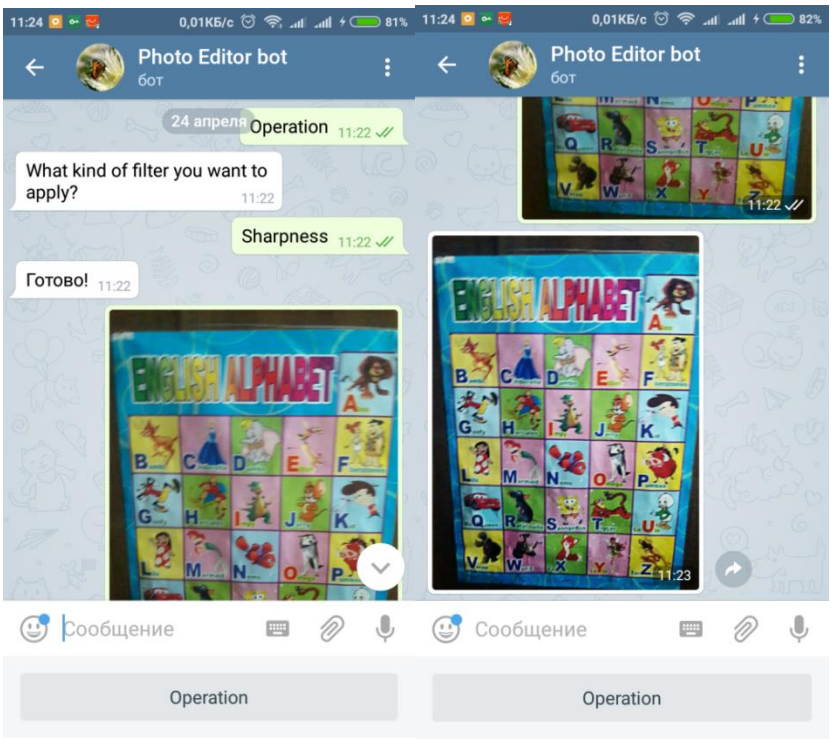

Рисунок 3.4 – Макет вкладки «Sharpness»

При завантаженні зображення і натисканні на кнопку «Contour» відкривається макет який наведено нижче (див. рис. 3.5).

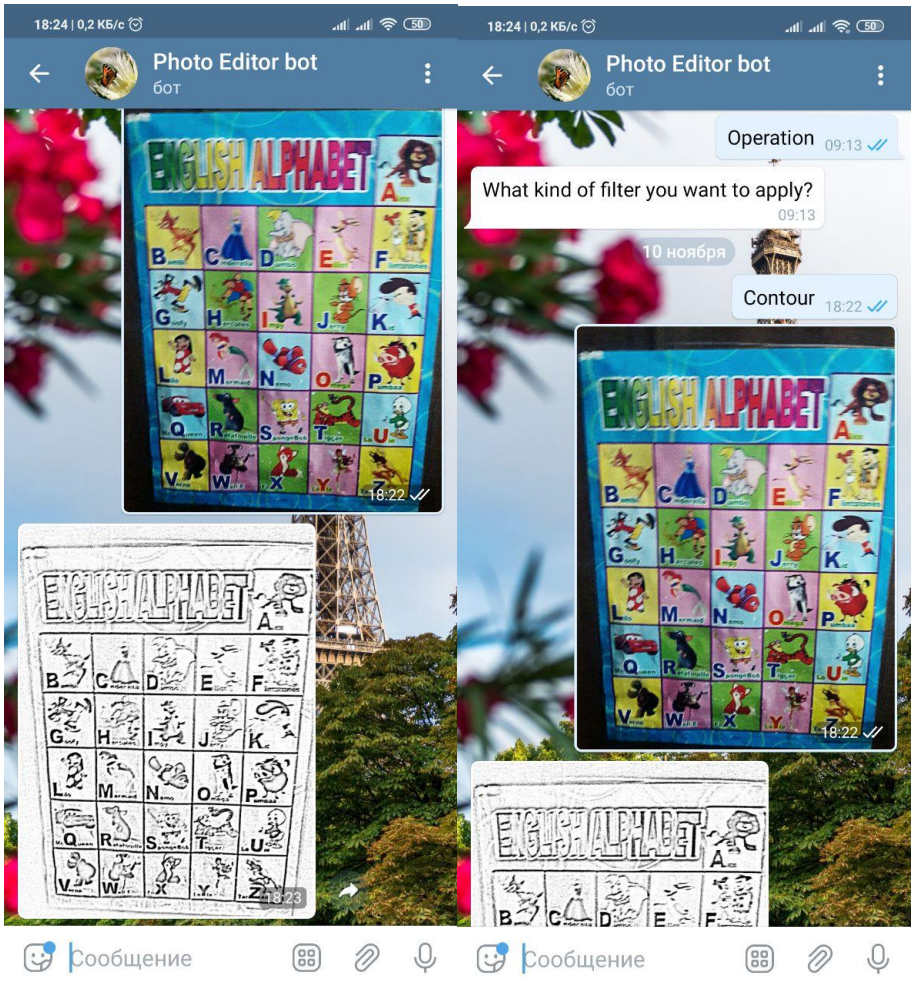

Рисунок 3.5 – Макет вкладки «Contour»

При завантаженні зображення і натисканні на кнопку «Negative» відкривається макет який наведено нижче (див. рис. 3.6)

Якщо у користувача виникають проблеми у використанні бота, натиснувши кнопку «Help_users» можна прочитати інструкцію користування (див. рис. 3.7).

Для імітації людського спілкування була розроблена команда привітання «/hello», на яку бот відповідає рандомно словом, або стікером привітання (див. рис. 3.8).

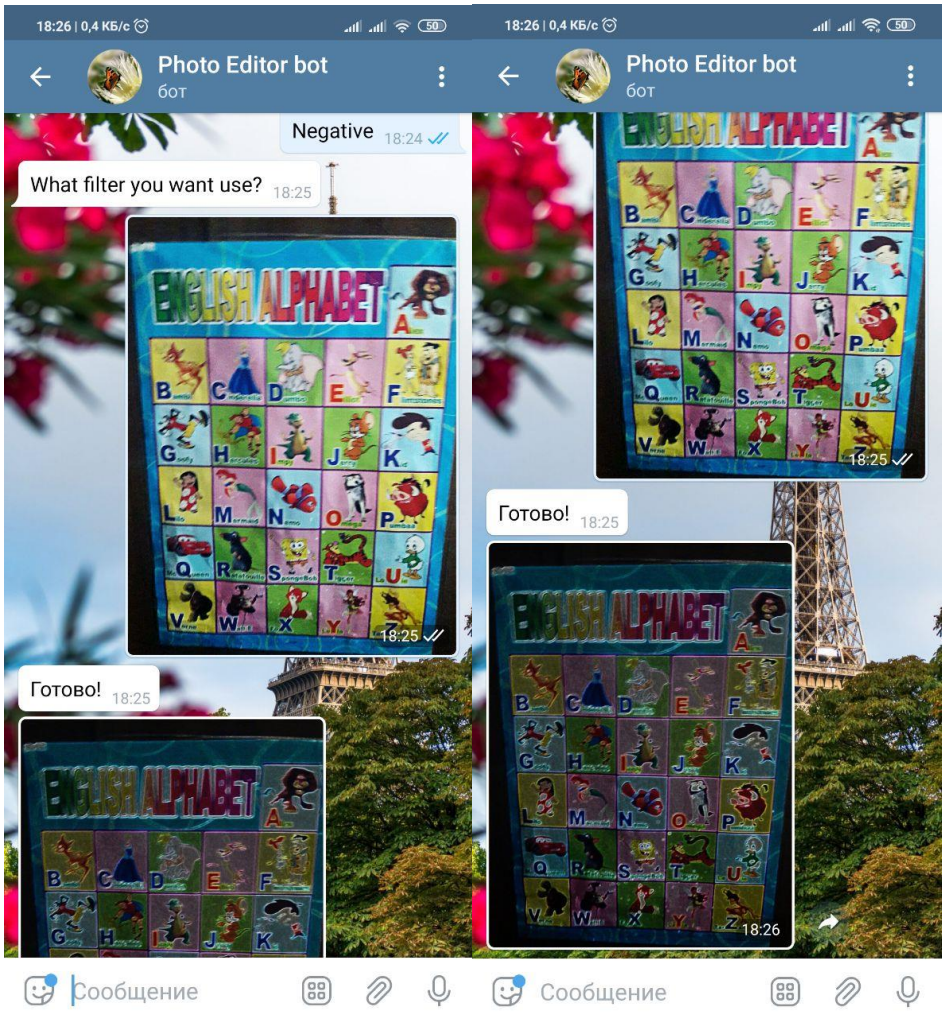

Рисунок 3.6 – Макет вкладки «Contour»

Адміністратор може виконувати розсилку повідомлень для всіх користувачів додатку (див. рис. 3.9).

Розсилку повідомлень можна використовувати в рекламних цілях та для інформування користувачів про оновлення даного боту, розсилки корисних повідомлень.

Рисунок 3.7 – Макет «Help_users» додатку

Рисунок 3.8 – Макет команди «/hello» додатку

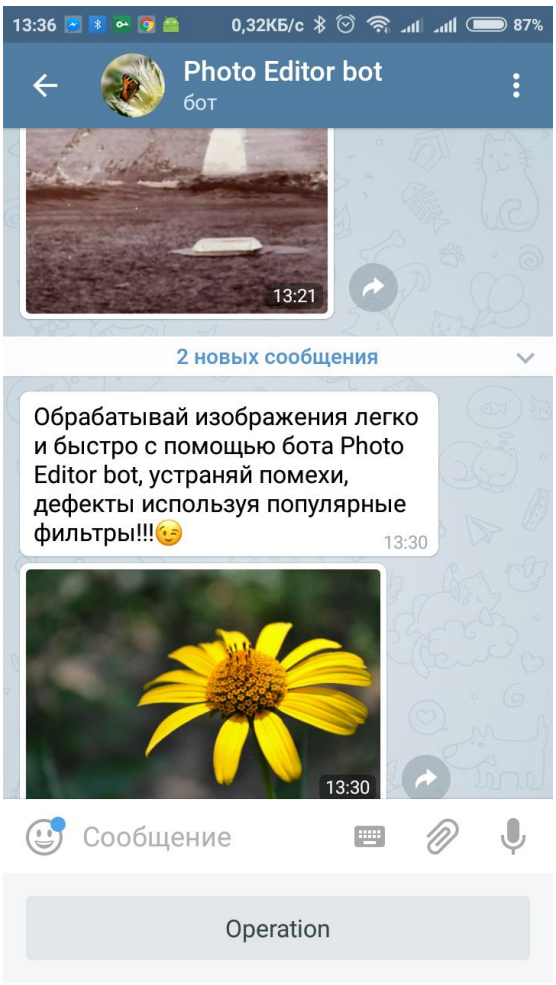

Рисунок 3.9 – Розсилка повідомлень в додатку

3.7 Бібліотека TelegramBotApi

Взаємодія ботів з людьми основана на HTTP-запитах. Для створення API ботів, потрібно використовувати бібліотеку pyTelegramBotAPI, яка бере на себе всі нюанси відправлення та отримання запитів, дозволяючи зосередитися безпосередньо на логіці.

Дана бібліотека дозволяє:

- програмувати власного бота;
- ставити йому різні команди взаємодії з користувачами;
- отримувати зручний доступ.

Фактично без API програмісти ботів повинні були б кожного разу писати свій власний штучний інтелект. З його використанням все стає

набагато простіше. У API є вже готові функції виведення і введення тексту, відповідей на зазначені питання. Тобто, фактично розробнику залишається тільки вписати свій текст [12].

Перед тим як програмно створювати бот, необхідно звернутися до головного бота Telegram–BotFather.

BotFather – це головний бот, створений, для керування іншими ботами. Його використовують, щоб зареєструвати нового бота або налаштовувати вже існуючих (див. рис. 3.10).

Відправляти боту команди виду /command можна кількома способами:

– натиснути на кнопку в тексті повідомлення;

– ввести «/» у полі введення повідомлення, щоб переглянути список доступних команд і вибрати команду зі списку;

– відправити повідомлення з командою.

Ім'я бота буде відображено в контактній інформації про бота. Його завжди можна змінити, якщо надіслати BotFather команду /setname

@username – коротке ім'я латиницею, яке використовується для швидкого пошуку в Telegram і в посиланнях на профіль виду t.me/username.

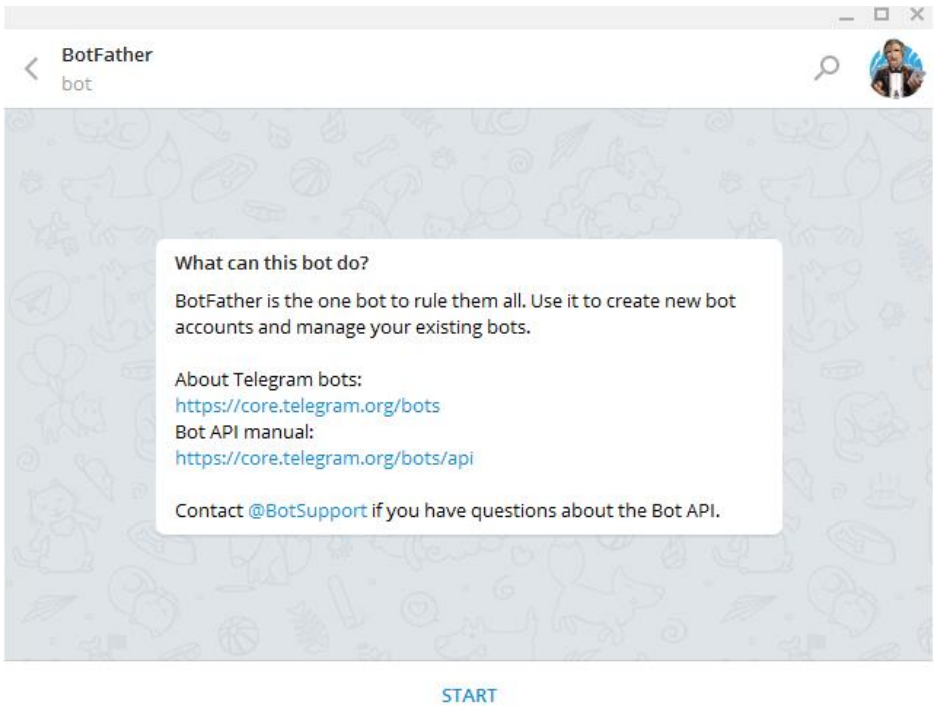

Рисунок 3.10 – Головний бот BotFather

Ім'я користувача може складатися з букв латинського алфавіту, підкреслень і цифр, довжиною від 5 до 32 символів. Ім'я користувача обов'язково має закінчуватися на 'bot', наприклад: @username_bot або @UsernameBot [11].

3.8 Взаємодія з користувачем

3.8.1 Спосіб взаємодії через getUpdates

Цей метод використовується для отримання оновлень через long polling (wiki). Відповідь повертається у вигляді масиву об'єктів Update. У даного способу оновлень є маса недоліків. Розробник повинен сам запитувати (запрашивать) з сервера список повідомлень користувачів, що не дуже зручно. Для цього необхідно запитувати кожні n секунд з сервера Telegram дані. Це ресурсомісткі (ресурсозатратно) і не раціонально [3].

3.8.2 setWebHook

Даний метод допомагає отримати повідомлення з допомогою використання WebHook.

Ідея полягає в тому, що сервер сам буде надсилати повідомлення користувача. Кожен раз при отриманні оновлення на цю адресу буде відправлено HTTPS POST з сериализованным в JSON об'єктом Update. В даному методі використовується об’єкт Message, який отримується з Update[13].

Тому для взаємодії з користувачем будемо використовувати метод Webhook (див. рис. 3.11).

Налаштування синхронізації бота Telegram з допомогою Webhook (додаток А).

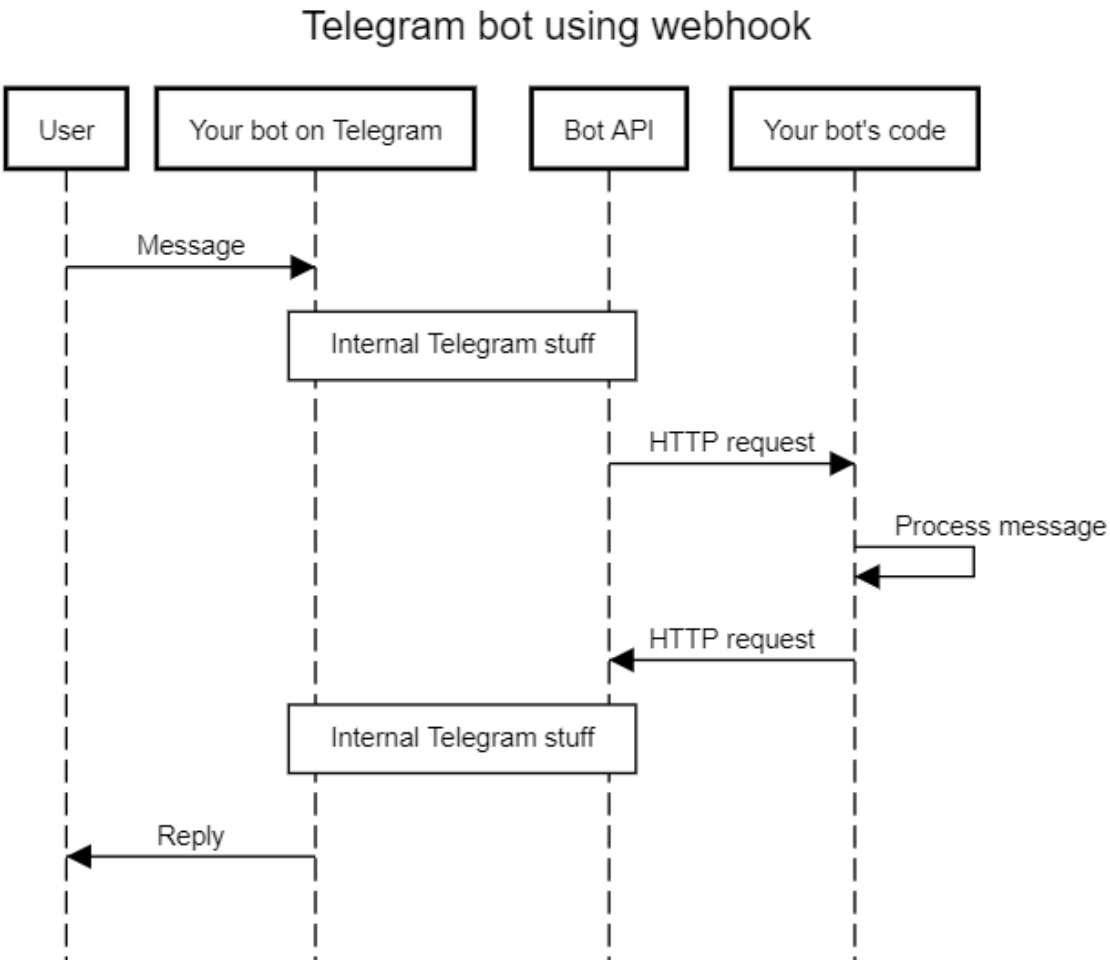

Рисунок 3.11 – Використання Webhook

3.9 Відправлення даних

Для того, щоб впевнитись, що дані правильно відправляться на сервер Telegram, приходять на Tornado Server та проходять там процедуру валідації виконуються такі дії (див. рис. 3.12):

```
class WSHandler(tornado.websocket.WebSocketHandler):
        def __init__(self, application, request, **kwargs):
                super(WSHandler, self).__init__(application, request, **kwargs)
```

Рисунок. 3.12 – Валідація

При створенні нового підключення з користувачем підключиться postgres.

```
self.conn, self.cur = core.connect_postgres()
        self.get_bot(self.conn, self.cur, request.remote_ip)
        def get_bot(self, conn, cur, ip):
                while True:
                        bot = core.get_bot(conn, cur)
                        if bot:
                                self.bot_token = bot[1]
                                self.customer_asked = bot[4]
                                core.make_bot_busy(self.conn, self.cur, self.bot_token)
                                core.add_remote_ip(self.conn, self.cur, self.bot_token, ip)
                        break
        def check_origin(self, origin):
            return True
```

Рисунок 3.13 – Використання ip-адреси бота

Після цього, необхідно, додати до бота ip-адресу (див. рис. 3.13), що забезпечить його відображення в мережі Інтернет та дасть змогу звертатись до нього.

Функція check_origin, забезпечує можливість підключення з різних пристроїв й дає їм змогу працювати з додатком у паралельному режимі.

Функція PeriodicCallback (див. рис. 3.14) перевіряє сервер Telegram на наявність нових повідомлень від менеджера:

```
def PeriodicCallBack:
        ans_telegram = core.get_updates(self.bot_token, self.conn, self.cur)
        if ans_telegram:
           self.write_message(ans_telegram).
```

Рисунок 3.14 – Функція PeriodicCallBack

Якщо прийшло повідомлення від менеджера, то воно буде надіслано в браузер клієнту.

3.10 Порівняльний аналіз роботи графічних бібліотек

Продуктивність однієї функції вище або нижче в конкретній бібліотеці. Якщо є додаток, якому потрібен якийсь набір функціональності, можна зробити бенчмарк саме під цю функціональність, і говорити, що для застосування деяка бібліотека працює швидше (повільніше).

Чому взагалі продуктивність – це проблема при роботі з графікою? Тому що складність будь-яких операцій залежить від декількох параметрів, і частіше за все складність зростає лінійно з кожним з них. А якщо цих факторів, наприклад, три, і складність лінійно залежить від кожного, то виходить складність в кубі.

Pillow – це єдина з усіх розглянутих бібліотек, яка не використовує розпаралелювання завдань. Саме через це її продуктивність може бути найвищою[9].

З точки зору продуктивності розглядаються два параметра. Перший параметр це реальний час виконання однієї операції. Є операція (або послідовність операцій), і за деякий час (wall clock) виконається ця послідовність. Цей параметр важливий на десктопі, де є користувач, який віддав команду і чекає результат (див. рис. 3.15).

Пропускна здатність всієї системи (потоку операцій) являється другим параметром. Коли є набір операцій, який триває постійно, або багато незалежних операцій, і важлива швидкість обробки цих операцій на пристрої. Ця метрика важливіша на сервері, де клієнтів багато, і потрібно обробити всі зображення[4]. Час обслуговування одного клієнта, трохи менше, ніж загальна пропускна здатність (див. рис. 3.16).

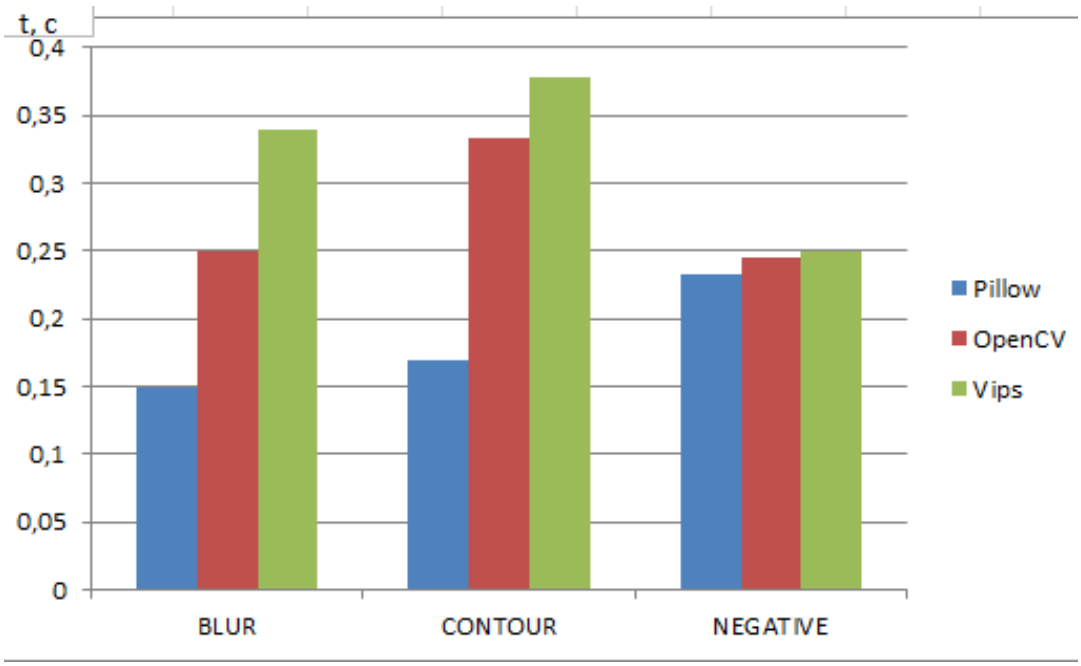

Рисунок 3.15 – Час виконання операцій

З першої діаграми видно, що час виконання операцій (фільтрування) найменший в графічної бібліотеки Pillow, а найвищий у Vips.

Друга діаграма показує пропускну здатність системи, і вона найвища також у Pillow, а найменша в OpenCV.

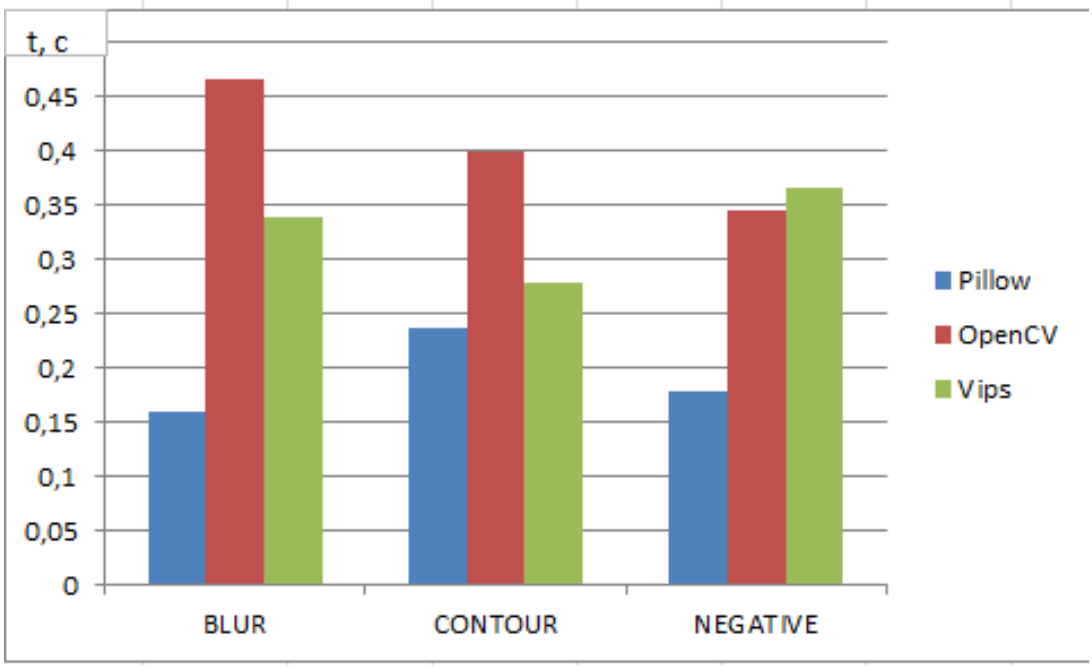

Рисунок 3.16 – Пропускна здатність системи

В третьому розділі, визначені функціональні вимоги, вимоги до складу і параметрів технічних засобів, вимоги до програмної сумісності, вимоги до надійності. Визначено структуру інформаційної системи для автоматизації обробки зображень.

Розглянуто покрокову реалізацію бота, що надає безкоштовні фільтри. Реалізовано можливість застосування розмиття, наведення різкості зображенню, фільтрування в фільтрі негативу, контуру та конвертація зображення у чорно-біле. Проведено порівняльний аналіз графічних бібліотек.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було розглянуто створення інструменту для обробки зображення в онлайн режимі за допомогою кросплатформенного месенджера. Було виконано проектування, розробку та реалізацію веб-додатку.

Також розглянуто поняття зображення, принцип побудови фільтрів та алгоритмів. В практичній частині описана структура програмного забезпечення, технічні вимоги до програми, розглянуто наскільки зручний даний інструмент, вивчився зміст необхідних бібліотек, можливий розвиток та шляхи вдосконалення.

Дослідження, що були проведені під час виконання дипломного проекту, дозволили зробити низку висновків та узагальнень теоретичного і практичного характеру стосовно того, які технології і програмні засоби повинні бути розроблені для даної системи.

Відповідно до поставленої мети під час роботи над дипломним проектом виконані наступні завдання:

Відповідно до поставленої мети під час роботи над дипломним проектом виконані наступні завдання:

а) проведено аналіз інформаційних ресурсів графічних бібліотек;

б) виконано проект та реалізацію веб-додатку інтерфейсу обробки зображення за допомогою фільтрів;

в) проведено тестування розробленого веб-додатку та порівняльний аналіз графічних бібліолтек;

г) розроблено програмну документацію.

Підсумовуючи матеріали, викладені в дипломній роботі, можна сказати, що всі поставлені завдання до дипломного проекту реалізовано, отже, мету дослідження досягнуто. Даний програмний додаток можна використовувати кожній людині самостійно за наявності мережі Інтернет.

ПЕРЕЛІК ПОСИЛАНЬ

1. Быстрое размытие по Гауссу. habr. URL : https://habr.com/post/151157/ (дата звернення : 03.11.2019).

2. Играемся с изображениями в Python. habr. URL : https://habr.com/post/163663/ (дата звернення : 29.10.2019).

3. Как написать Telegram бота : практическое руководство. Khashtamov. URL : https://khashtamov.com/ru/create-telegram-bot-in-python/ (дата звернення : 18.10.2019).

4. Кувшинов Т. Н., Скоцкая Н. С. Инжинерная и компьютерная графика: учеб. пос. Москва : КноРус, 2017. 234 с.

5. Маценко В. Г. Комп'ютерна графіка : навч. посіб. Чернівці : Рута, 2009. 343 с.

6. Медианная фильтрация. Электротехника. URL : http://electrono.ru/4-6-3-mediannaya-fil-traciya-osncifr (дата звернення : 17.09.2019).

7. Моржин А., Ходарев А., Князь В. Обработка и анализ цифровых изображений с примерами на LabVIEW IMAQ : учеб. пос. Москва : ДМК Пресс, 2008. 464 с.

8. Редакторы векторной и растровой графики. SoftHome. URL : https://www.softhome.ru/article/redaktory-vektornoy-i-rastrovoy-grafiki (дата звернення : 25.05.2019).

9. Хуанг Т. С. Быстрые алгоритмы в цифровой обработке изображений : учеб. пос. Москва : Радио и связь, 1984. 224 с.

10. Lutz M. Python Pocket Reference. England : O'Reilly Media, 2017. 266 с.

11. Telegram Bot API. Telegram Bots. URL : https://core.telegram.org/bots/api (дата звернення : 09.05.2019).

12. Welcome to Python Telegram Bot’s documentation!. Python Telegram Bot. URL : https://python-telegram-bot.readthedocs.io/en/stable/ (дата звернення : 28.09.2019).

13. Yao M. Conversational Interfaces: Principles of Successful Bots, Chatbots & Messaging Apps. England : Topbots, 2017. 210 с.

ДОДАТОК А

Код основних функцій бота

```
class WebhookHandler(BaseHTTPServer.BaseHTTPRequestHandler, object):
    server_version = 'WebhookHandler/1.0'
    def __init__(self, request, client_address, server):
        self.logger = logging.getLogger(__name__)
        super(WebhookHandler, self).__init__(request, client_address, server)
    def do_HEAD(self):
        self.send_response(200)
        self.end_headers()
    def do_GET(self):
        self.send_response(200)
        self.end_headers()
    def do_POST(self):
        self.logger.debug('Webhook triggered')
        try:
            self._validate_post()
            clen = self._get_content_len()
        except _InvalidPost as e:
            self.send_error(e.http_code)
            self.end_headers()
        else:
            buf = self.rfile.read(clen)
            json_string = bytes_to_native_str(buf)
            self.send_response(200)
            self.end_headers()
```

```
                self.logger.debug('Webhook received data: ' + json_string)
                update = Update.de_json(json.loads(json_string), self.server.bot)
                self.logger.debug('Received Update with ID %d on Webhook' %
update.update_id)
                self.server.update_queue.put(update)
            def _validate_post(self):
                if not (self.path == self.server.webhook_path and 'content-type' in
self.headers and
                        self.headers['content-type'] == 'application/json'):
                    raise _InvalidPost(403)
            def _get_content_len(self):
                clen = self.headers.get('content-length')
                if clen is None:
                    raise _InvalidPost(411)
                try:
                    clen = int(clen)
                except ValueError:
                    raise _InvalidPost(403)
                if clen < 0:
                    raise _InvalidPost(403)
                return clen
            def log_message(self, format, *args):
        self.logger.debug("%s - - %s" % (self.address_string(), format % args))
```

ДОДАТОК Б

Код створення діалогового інтерфейсу

```
import telebot
import constants
bot = telebot.TeleBot(constants.token)
@bot.message_handler(commands=['start'])
def handle_start(message):
    user_markup = telebot.types.ReplyKeyboardMarkup(True, True)
    user_markup.row('filter')
    user_markup.row('result')
    bot.send_message(message.from_user.id, 'Filters',
reply_markup=user_markup)
@bot.message_handler(content_types=['text'])
def hangle_text(message):
    if message.text == "help":
        bot.send_message(message.chat.id, "loading")
    else:
        bot.send_message(message.chat.id, "eror:(")
@bot.message_handler(content_types=['photo'])
def handle_docs_photo(message):
    try:
        file_info = bot.get_file(message.photo[len(message.photo) - 1].file_id)
        downloaded_file = bot.download_file(file_info.file_path)
        src = '/mnt/files/tmp/' + file_info.file_path;
        with open(src, 'wb') as new_file:
            new_file.write(downloaded_file)
        bot.reply_to(message, "loading..)
```