НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“___” _____2022 р.

ДИПЛОМНИЙ ПРОЕКТ

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

на тему: Телеграм-бот для розпізнавання текстів

Виконав: студент __4__ курсу, групи ___ІВ-83_____
(шифр групи)

_____Герасімов Станіслав Сергійович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____асистент___Ковальчук О.М._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль)_Сімонеко А.В._____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент_____
(підпис)

Київ – 2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки
Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 123 «Комп'ютерна інженерія»
за освітньо-професійною програмою "Комп'ютерні системи та мережі"

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
"___"_____20__ р.

ЗАВДАННЯ
на дипломний проект студенту

Герасімову Станіславу Сергійовичу

1. Тема проекту «Телеграм-бот для розпізнавання текстів»
керівник проекту Ковальчук Олександр Миронович, асистент, затверджені наказом по університету від «__» _____ __ 2022р. №__

2. Термін здачі студентом закінченого роботи _____.

3. Вихідні дані до проекту: технічна документація, теоретичні та статистичні дані.

4. Зміст пояснювальної записки:

Розділ 1. Месенджери як модерний засіб обміну даними

Розділ 2. Вибір технологій та середовища розробки платформи

Розділ 3. Етапи розробки та створення програмного додатку

Розділ 4. Розробка платформи для створення Telegram-бота для забезпечення розпізнавання тексту

5. Консультант роботи, з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Розділ 1	Викладач кафедри ОТ Сімонеко А.В.		
Розділ 2	Викладач кафедри ОТ Сімонеко А.В.		
Розділ 3	Викладач кафедри ОТ Сімонеко А.В.		
Розділ 4	Викладач кафедри ОТ Сімонеко А.В.		

6. Дата видачі завдання 14.12 .2021 року

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту(роботи)	Примітки
1.	Затвердження теми роботи	14.12.2021-19.12.2021	Виконано
2.	Вивчення та аналіз завдання	19.12.2021-19.03.2022	Виконано
3.	Розробка архітектури та загальної структури	19.03.2022-30.03.2022	Виконано
4.	Розробка структур окремих підсистем	30.03.2022-10.04.2022	Виконано
5.	Програмна реалізація	10.04.2022-20.04.2022	Виконано
6.	Оформлення пояснювальної записки	15.04.2022-10.05.2022	Виконано
7.	Захист програмного продукту	25.04.2022	Виконано
8.	Перед захист	28.05.2022	Виконано
9.	Захист	19.06.2022	Виконано

Студент Станіслав Герасімов _____
(підпис)

Керівник Олександр Ковальчук_____
(підпис)

АННОТАЦІЯ

Кваліфікаційна робота на здобуття ступеня бакалавра на тему «Розробка телеграм-боту для розпізнавання текстів» складається з 71 сторінок, 32 ілюстрацій, 3 таблиць та 29 джерел посилань.

Ключові слова: ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, РОЗПІЗНАВАННЯ ЗОБРАЖЕННЯ, МЕСЕНДЖЕРИ, ПЛАТФОРМА ДЛЯ СТВОРЕННЯ TELEGRAM-БОТІВ, СИСТЕМИ ШВИДКОГО ОБМІНУ ДАНИМИ, ЧАТ-БОТ, PYTHON, TELEGRAM, TELEGRAM BOT API, TELEGRAM-БОТИ.

Актуальність кваліфікаційної роботи обумовлена високою популярністю месенджерів та таких засобів автоматизації як платформа для розробки Telegram-бота для забезпечення інформаційної розпізнавання тексту серед користувачів мережі Інтернет. Такі платформи дозволяють спростити щоденні рутинні завдання, такі як отримання інформації важливої інформації щодо обраного предмету, оновлення даних, тощо. Головною перевагою перед класичними додатками є можливість суміщення всіх можливостей на платформі одного месенджера.

У даному випадку було обрано популярний та зручний застосунок (месенджер) Telegram. Крім цього, у 2020 році навчання онлайн набуло піку своєї популярності через глобальну пандемію сумнозвісного вірусу COVID-19, що своїм поширенням змусила перейти не тільки на навчання, а навіть на життя онлайн.

Мета дослідження: описати ключові принципи створення ботів для месенджерів. Розглянути можливості застосування засобів нейронних мереж для покращення досвіду користувача при використанні бота.

Виходячи з поставленої мети, були поставлені наступні завдання:

- аналіз обраної предметної області;
- порівняння наявних аналогів чат-ботів;
- вибір технологій та середовища розробки;
- розробка платформи для розробки Telegram-бота для забезпечення розпізнавання тексту.

Об’єктом дослідження у даній кваліфікаційній роботі є засоби миттєвого обміну повідомленнями.

Предметом кваліфікаційної роботи є Telegram-боти, як засіб взаємодії із користувачем.

Для створення платформи для розробки Telegram-бота для забезпечення розпізнавання тексту було обрано найбільш стабільні засоби та інструменти розробника, а саме: мову програмування Python разом з набором корисних бібліотек, що використовувались при реалізації програмного коду в багатофункціональному середовищі розробки PyCharm, використано бібліотеку Telebot, яка реалізує безпосередньо Telegram Bot API, а також базу даних на основі SQLAlchemy.

Проаналізовано вже існуючі аналоги до розроблюваного програмного продукту та наведено приклади їх використання, основні переваги та недоліки. В результаті, було виявлено необхідність створення графічного інтерфейсу для взаємодії з користувачами, а не обмежуватись стандартним набором команд для керування ботом у Telegram. Крім того, було систематизовано та формалізовано алгоритм вибору інструментів розробки.

Як результат кваліфікаційної роботи, на базі вищеописаних принципів та із використанням зазначених технологій було створено платформу для розробки Telegram-бота для забезпечення розпізнавання тексту. Даний програмний продукт є основним доказом концепції автоматизації навчального процесу за допомогою ботів для месенджерів (зокрема, у даному випадку для Telegram) та ілюструє можливу швидкість та гнучкість розробки.

Зміст

ВСТУП

Інтернет в сучасних умовах – це універсальне середовище для спілкування, розваг, а також навчання. В даний час в світі існує велика кількість засобів, форм та способів спілкування, й чимала частина з них так чи інакше пов'язана з сучасними технічними можливостями, які, зокрема, представлені використанням глобальної комп'ютерної мережі.

Інтернет, окрім джерела для отримання різноманітної та корисної для користувачів інформації, також є основною формою віртуального спілкування. Зв'язок з друзями та родичами, контакти з робочими партнерами, нові знайомства – все це є важливим компонентом повсякденного життя сучасної людини, причому вибір найбільш зручних способів спілкування онлайн у користувача досить великий.

Останні декілька років популярність систем обміну миттєвими повідомленнями (месенджерів) тільки зростає. Спочатку вони розроблювались у якості засобів спілкування між користувачами, а на даний час перетворилися на засоби отримання корисної інформації та навіть у потужні маркетингові інструменти.

Чималу роль у процесі такого перетворення відіграли боти. Зараз існує безліч їх варіацій – від ботів для отримання RSS-розсилок до ботів для вивчення якихось навчальних дисциплін. Звісно, таким корисним інструментом зацікавились не тільки користувачі та власники бізнесу, а також навчальні заклади. І хоча боти для месенджеру Telegram уже декілька років активно використовуються закордоном, в Україні вони не так поширені та тільки набувають популярності серед користувачів, а отже є дуже перспективним напрямком не тільки для розробників, а також й для самих користувачів.

Слід враховувати, що доволі важливим аспектом є досвід користувача при роботі із ботом, який включає те як бот сприймає команди, наскільки добре він сприймає суть сформованого запиту користувача та наскільки доречні й зрозумілі його відповіді при безпосередній взаємодії з користувачем.

Актуальність кваліфікаційної роботи обумовлена високою популярністю месенджерів та таких засобів автоматизації як платформа для розробки Telegram-бота для забезпечення інформаційної підтримки розпізнавання тексту серед користувачів мережі Інтернет. Такі платформи дозволяють спростити щоденні рутинні завдання, такі як отримання інформації важливої інформації щодо обраного предмету, оновлення даних, тощо. Головною перевагою перед класичними додатками є можливість суміщення всіх можливостей на платформі одного месенджера.

У даному випадку було обрано популярний та зручний застосунок (месенджер) Telegram. Крім цього, у 2020 році навчання онлайн набуло піку своєї популярності через глобальну пандемію сумнозвісного вірусу COVID-19, що своїм поширенням змусила перейти не тільки на навчання, а навіть на життя онлайн.

Мета дослідження: описати ключові принципи створення ботів для месенджерів. Розглянути можливості застосування засобів нейронних мереж для покращення досвіду користувача при використанні бота.

Виходячи з поставленої мети, були поставлені наступні **завдання**:

- аналіз обраної предметної області;
- порівняння наявних аналогів чат-ботів;
- вибір технологій та середовища розробки;
- розробка платформи для розробки Telegram-бота для забезпечення розпізнавання тексту.

Об’єктом дослідження у даній кваліфікаційній роботі є засоби миттєвого обміну повідомленнями.

Предметом кваліфікаційної роботи є Telegram-боти, як засіб взаємодії із користувачем.

Наукова новизна одержаних результатів полягає у описанні загальних **принципів** побудови ботів для сучасних месенджерів.

Можливі сфери застосування. Після процесу розробки дану платформу можна буде використовувати для забезпечення інформаційної підтримки та спрощення роботи з даними.

РОЗДІЛ 1

МЕСЕНДЖЕРИ ЯК МОДЕРНИЙ ЗАСІБ ОБМІНУ ДАНИМИ

1.1 Історія створення та становлення месенджерів з використанням новітніх інформаційних технологій

Доступність комп’ютерів та смартфонів у широкому розумінні цих понять стала дійсно важливою подією та дала сильний поштовх у розвитку новітніх засобів зв'язку між декількома користувачами онлайн. Замінили звичні для багатьох телефонні розмови онлайн голосові повідомлення та навіть відеодзвінки. Саме з цього моменту користувачі мережі Інтернет почали використовувати комп'ютери не тільки для роботи, а й для швидкісного спілкування та обміну даними.

Першим з подібних інструментів можна вважати появу електронної пошти [1], яка стала таким собі електронним наслідником звичних для всіх рукописних або друкованих листів. Але ефективність зазначеного методу спілкування та швидкість доставки повідомлень зросла в рази.

Далі популярності набули чати. Тоді їх почали використовувати для організації групового спілкування між декількома користувачами (зазвичай між двома та більше користувачами). Першим з таких інтерактивних чатів можна вважати Talkomatic. Він був розроблений у 1973 р. Д. Брауном та Д. Р. Уоллі в системі PLATO (Programmed Logic for Automated Teaching Operations) [2], яка на той час вважалась першою системою для електронного навчання в світі та була призначена для автоматизації навчального процесу та обміну даними Університету Іллінойсу. Інтерфейс Talkomatic виглядав наступним чином (рис. 1.1).

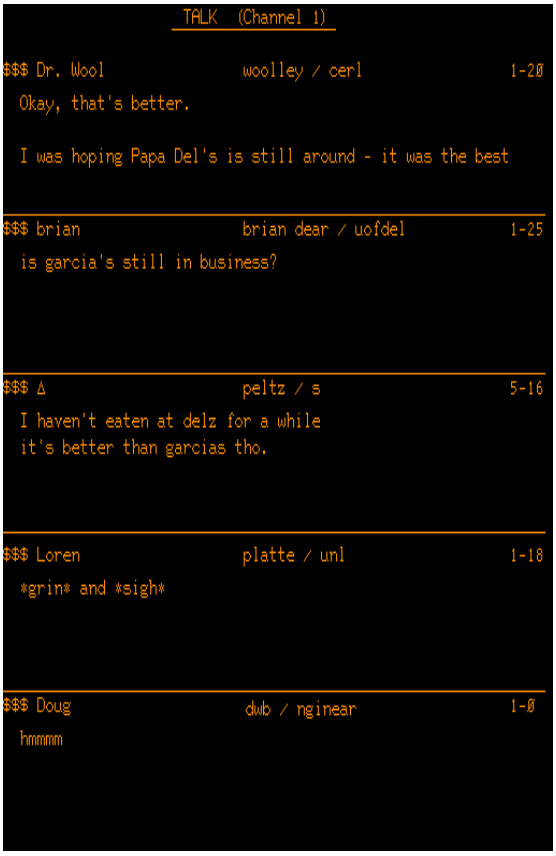


Рисунок 1.1 – Інтерфейс Talkomatic

У даній системі можна було отримати доступ до шести різних каналів, кожен з яких мав змогу вмістити до п'яти користувачів одночасно [3]. Звісно, з того часу чати еволюціонували, й тепер практично кожний сервіс, що надає послуги обміну повідомленнями та даними онлайн, немає жорстких обмежень щодо кількості користувачів та каналів, у яких бере участь користувач, як це було на початку 1970-х років. Також сучасні чати все частіше використовують вузькоспрямовані технології, такі як нейронні мережі та штучний інтелект, та інші.

У будь-якому випадку, усі ці нововведення були направлені на покращення взаємодії безпосередньо з користувачем. Паралельно з розробкою та освоєнням користувачами перших чатів для швидкого обміну текстовими повідомленнями та іншими даними, юзери починали переходити на взаємодію з іншими обивателями мережі Інтернет за допомогою перших так званих месенджерів. Примітивні застосунки для передачі даних побачили світ майже 50 років тому.

Перші системи, що базувались на використанні графічного інтерфейсу користувача для відображення отриманого повідомлення, набули популярності у 1990-х роках. До перших таких месенджерів належали ICQ (рис. 1.2.) та AOL Instant Messenger. У той же час інші компанії розробили власне програмне забезпечення для обміну миттєвими повідомленнями, а саме Ubique, MSN та Yahoo!.

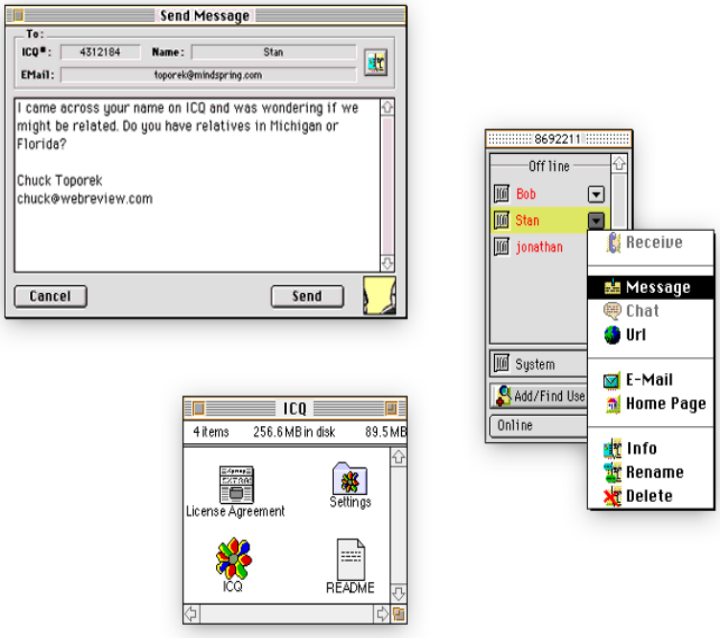


Рисунок 1.2 – Інтерфейс відомого месенджера ICQ

Кожен з таких застосунків, що на даний час здаються примітивними, мав свій власний протокол та клієнт, тому для того, щоб використовувати одразу декілька месенджерів, користувачам необхідно було запускати відразу декілька додатків для коректної роботи. На початку 2000-х років було запущено у розробку ПЗ з відкритим програмним кодом та відкритий протокол Jabber, який успішно пройшов стандартизацію під звичною назвою XMPP (або Extensible Messaging and Presence Protocol). Сервери такого типу можуть слугувати в якості шлюзів для інших протоколів, що не потребує запуску відразу декількох клієнтів для роботи.

Як вже зазначалось, інтерфейс перших месенджерів був доволі примітивним як для нашого часу, а саме програмне забезпечення надавало користувачеві

тільки базовий набір функцій. Також у багатьох із перших таких систем було обмеження на максимально допустиму довжину повідомлення, що доволі схоже з обмеженням на довжину смс-повідомлення. Завдяки цьому парадоксу був неофіційно створений сленг подібних месенджерів, що й використовується і у наш час. Таким чином, користувачі іноді використовують Інтернет-сленг, щоб скоротити слова чи фрази для прискорення написання або зменшення кількості символів. Наприклад, мабуть найбільш відома фраза: «LOL», що те ж саме, що й «laughing out loud». З англійської ця сленгова фраза переводиться як «голосно сміятися» [4].

Скоріш за все, AOL Instant Messenger є справжнім попередником сучасних соціальних мереж та модерних засобів обміну текстовими повідомленнями та даними. У персональному профілі користувачі могли заповнити деякі відомості про себе. Профілі у мережі були доступні для пошуку, щоб інші користувачі могли переглядати інформацію. Як не дивно, але це був один з найбільш інноваційних інструментів того часу [4]. Інтерфейс месенджеру зображено на рис. 1.3.

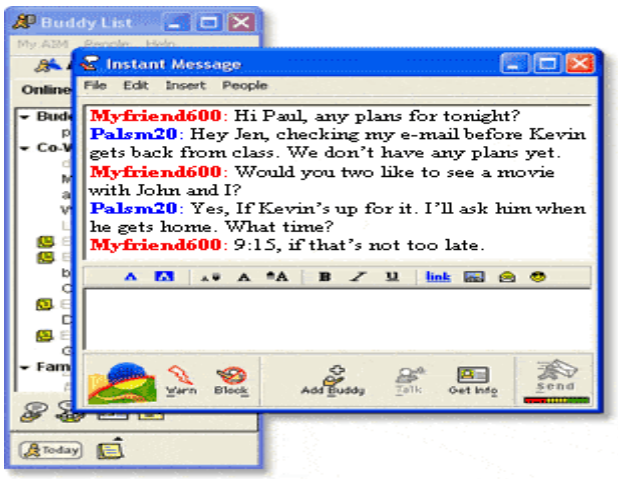


Рисунок 1.3 – Інтерфейс месенджера AOL Instant Messenger

Першою соціальною мережею, більш схожою на сучасні своїм інтерфейсом та функціоналом, можна вважати сайт Friendster. Впродовж перших трьох місяців після свого запуску інноваційна соціальна мережа змогла залучити до себе близько 3 мільйонів користувачів. У той час це означало, що 1 з 126 юзерів

мережі Інтернет використовують сайт Friendster (рис. 1.4) в якості засобу зв'язку з іншими користувачам по всьому світу [5].

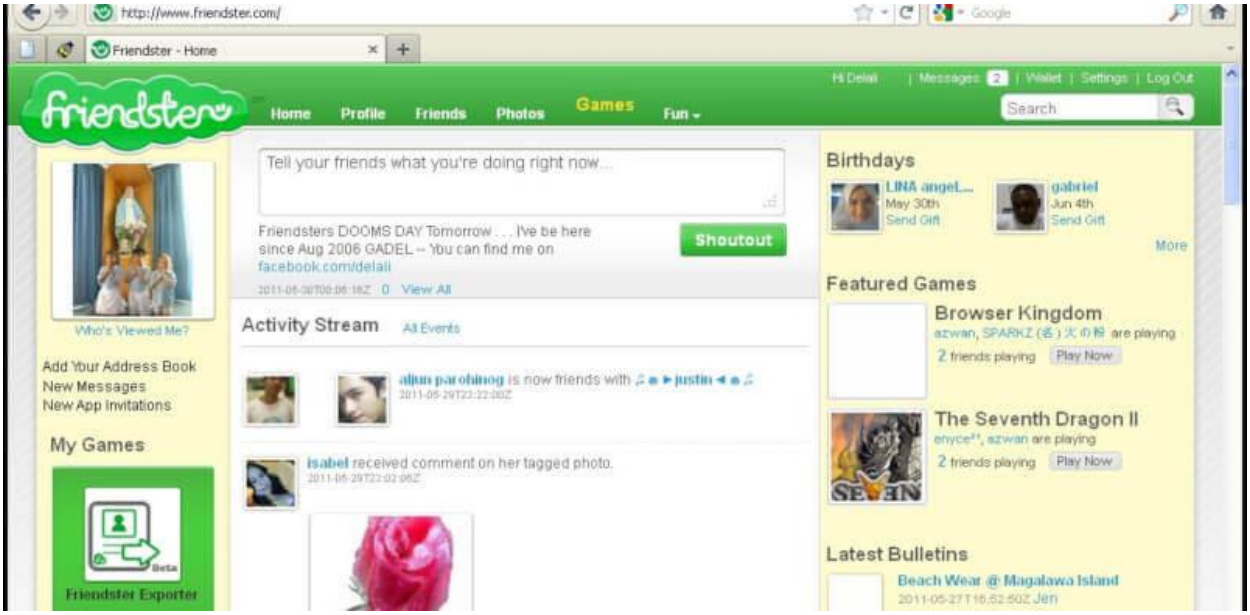


Рисунок 1.4 – Інтерфейс соціальної мережі Friendster

Один з найбільших за аудиторією сервісів зі спілкування Facebook (рис. 1.5) побачив світ у 2004 році. Слід зазначити, що першочерговим наміром роботи соціальної мережі був інноваційний спосіб зв'язати між собою студентів, що навчались у коледжі. Мережа Facebook вперше почала функціонувати у Гарвардському університеті, де й саме навчався головний її ідеолог Марк Цукерберг. Спочатку повноправним членом цієї мережі можна було стати тільки за запрошенням вже активного користувача Facebook. Так звана ексклюзивна особливість для реєстрації в мережі виявилась доволі успішною. Тільки за перший місяць роботи соціальної мережі до лав її учасників приєдналась більша половина студентів альма-матеру розробника [6].

За два роки сайт відкрили для широкого загалу. Вже у 2008 році Facebook обігнав MySpace та Friendster за аудиторією. На кінець 2017 року кількість унікальних користувачів мережі перевищила 2 мільярди юзерів, 10 мільйонів з яких налічувалось на території України.

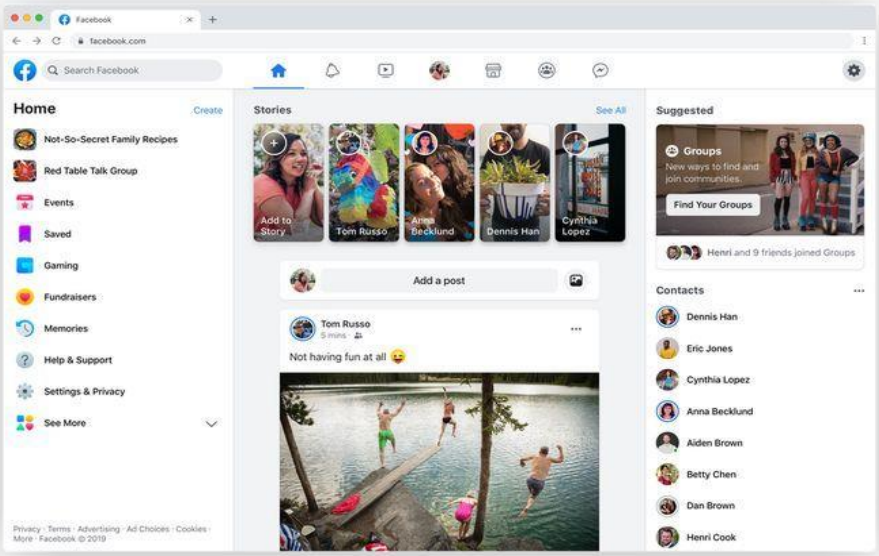


Рисунок 1.5 – Інтерфейс найбільшої соціальної мережі Facebook

1.2 Модерні месенджерів

Свій розвиток Інтернет-сервіси для спілкування почали з чатів, потім було розроблено чимало месенджерів, після цього пік популярності прийшовся на соціальні мережі, але з недавніх пір месенджери знову очолили список найбільш перспективних сервісів. Це підтверджує дослідження експертів у 2018 році: кількість користувачів месенджерів в Україні збільшилася на 12%. Одночасно з цим, середня кількість месенджерів, які використовуються одним користувачем, збільшилася за 2018 рік з 1,5 до 3 застосунків [7].

Месенджери використовують переважно дорослі користувачі мережі Інтернет, тому на частку осіб до 18 років припадає 8% від загальної кількості аудиторії, користувачі у віці від 18 до 25 років становлять 12%, від 25 до 35 років – 35,7%, від 35 до 45 років – 22,1%. При цьому 14,2% користувачів месенджерів знаходяться у віці від 45 до 55 років, а 6,7% – від 55 до 64 років. І лише 1,3% аудиторії месенджерів знаходиться у віці старше 64 років [7].

На графіку (рис. 1.6) зображено відсоток популярності тих чи інших додатків для спілкування та обміну даними на території України, згідно з даними tns-ua.com [8].

Рисунок 1.6 – Рейтинг мобільних додатків

Як бачимо, очевидним лідером є Viber з показником 97,9%, що встановлений майже у кожного користувача смартфоном на території України. Другу позицію після нього займає, якщо враховувати безпосередньо месенджери, Facebook Messenger з 73,6% завантажень. Неподалік від нього розмістились ще два сучасні застосунки для спілкування та обміну даними Instagram та Telegram з 71,3% та 70,6% відповідно.

Причина повторної хвилі популярності месенджерів – зміни в області мобільного Інтернету: високі швидкості, набагато нижчі ціни, ніж раніше, широке поширення смартфонів, тощо. Кріс Мессіна в своїй статті про розвиток електронної комерції зазначив [9], що ще 2016 рік, за його думкою, мав стати роком діалогової або розмовної комерції, вже не кажучи про 2020 рік. Скоріше за все, експерт мав на увазі під цим поняттям чати, месенджери або інші інтерфейси на природній мові (тобто голосом), які забезпечують взаємодію людей з торговими брендами або послугами.

Кінцевим результатом можна вбачати те, що користувачі матимуть змогу взаємодіяти з брендами, компаніями, а також з іншими користувачами за

допомогою месенджерів, що й реалізується у наш час. При цьому, неважливо, з ким будуть спілкуватися люди – з живою людиною, з ботом, або з їх комбінацією.

Слід зазначити, що всі сучасні месенджери мають безліч схожих функцій, таких як відправка повідомлень, медіа файлів, аудіо- та відео спілкування. В даний час існує велика різноманітність месенджерів, серед яких варто відзначити деякі з них, а також їх переваги (на прикладі месенджеру Telegram, на базі якого було розроблено програмний продукт для забезпечення розпізнавання тексту):

- найбільш зручна платформа для розробки та підтримки чат-ботів;
- відкритий вихідний код;
- вбудований редактор фото;
- спеціально розроблений протокол шифрування MTPromo;
- секретні чати та повідомлення, що видаляються в автоматичному режимі.

Telegram – це кросплатформний додаток, розроблений на мові програмування C ++, що дозволяє обмінюватися повідомленнями та файлами більшості форматів [10]. Месенджер використовує спеціально розроблену серверну частину з закритим кодом, яка працює на серверах Німеччини та США. Приклад інтерфейсу наведено на рис. 1.7.

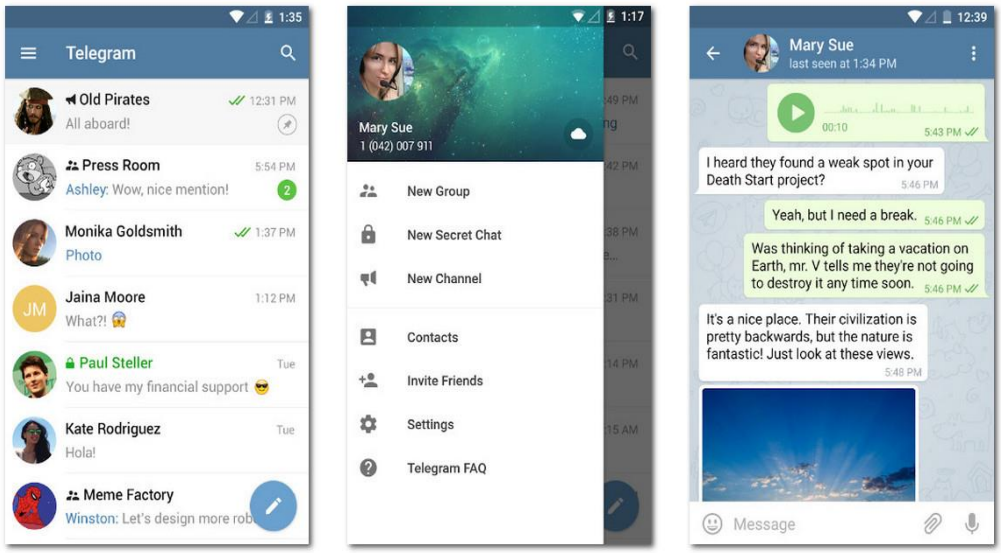


Рисунок 1.7 – Інтерфейс месенджера Telegram

Варто також зазначити, що обрана платформа для інтеграції програмного продукту Telegram пропонує користувачу можливість використання секретних чатів. Також, у порівнянні з найбільш популярними у світі застосунками WhatsApp та Facebook Messenger, Telegram працює швидше та надає можливість передачі файлів будь-яких форматів. Суттєвим недоліком WhatsApp та багатьох інших месенджерів є те, що у свій обліковий запис можна увійти тільки з одного пристрою, тому відповідно до правил використання застосунку, користувача у випадку спроби входу у свій обліковий запис з декількох пристроїв одночасно можуть заблокувати.

Як вже згадувалось раніше, особливостями даного месенджера є секретні чати та спеціально розроблений протокол шифрування MTProto [11], принцип роботи якого зображено на рис. 1.8.

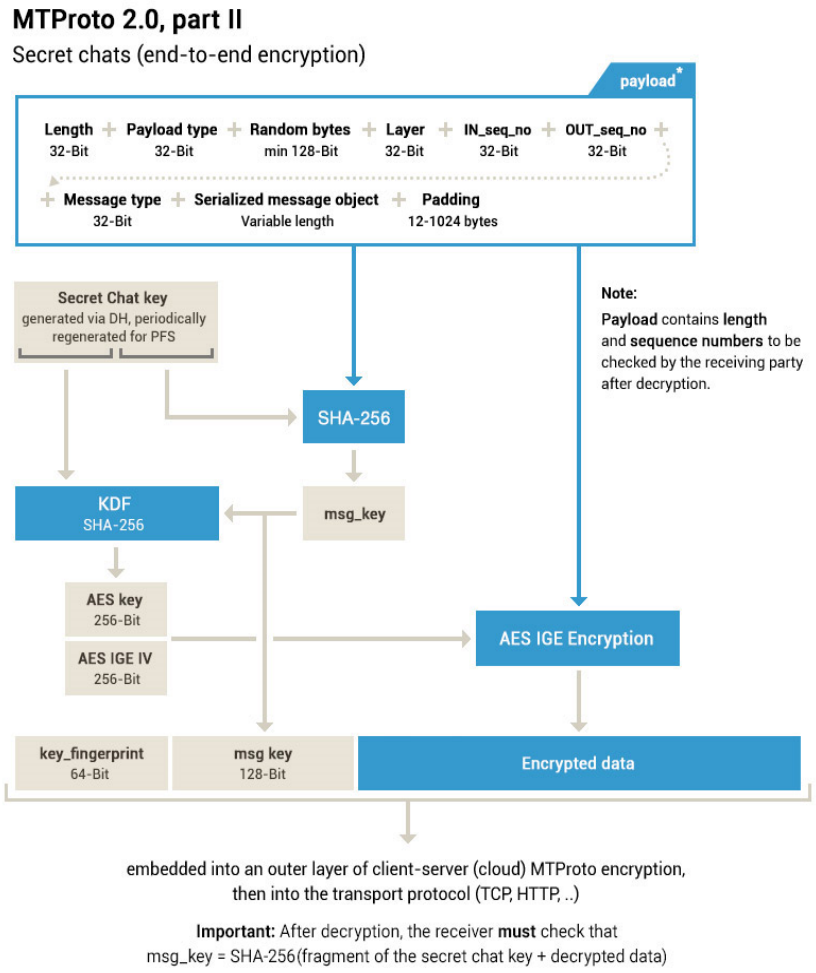


Рисунок 1.8 – Принцип роботи протоколу MTProto

Секретні чати потрібні для людей, які хочуть забезпечити себе максимально безпечним спілкуванням. Крім використання основного протоколу, всі повідомлення шифруються ключами за принципом від пристрою до пристрою. Ніхто не може перехопити або розшифрувати повідомлення, у тому числі безпосередньо працівники Telegram. Повідомлення з таких чатів не можна пересилати й вони не зберігаються на серверах додатку. Але у секретних чатів є два недоліки – це неможливість відкрити чат на пристрої, який не брав участі в його створенні, й ніхто не захищений від скріншота інтерфейсу Telegram з відритим секретним чатом [11].

Крім того, месенджер Telegram доступний для всіх популярних платформах, наприклад, Android OS, iOS, Windows Phone, Linux та інші. Так само в більшості випадків версія додатку для різних операційних систем локалізована щонайменше на англійську, російську, польську та німецьку мови.

Як показали всесвітні дослідження, останні десятки років традиційні телефонні дзвінки стають все менш популярними у порівнянні із засобами миттєвого обміну повідомленнями чи сервісами, що надають своїм користувачам послуги онлайн-зв’язку. Більше того, чимала кількість користувачів мережі Інтернет надає перевагу текстовим чи мультимедійним повідомленням у порівнянні із дзвінками.

Як відображає запропонований автором даної кваліфікаційної роботи огляд модерних засобів миттєвого обміну повідомленнями, подібні застосунки є невід'ємною частиною для взаємодії декількох користувачів або групи користувачів між собою у мережі Інтернет. Тому саме із розквітом індустрії розробки месенджерів та розширення їхнього функціоналу, користувачі надають перевагу дешевому та зручному спілкуванню, тому саме й обирають подібні застосунки. Звісно, існує велика кількість різноманітних месенджерів зі своїми відмінними особливостями, але саме Telegram має найбільші темпи росту кількості активних користувачів у останній час, тому й перспективи росту аудиторії для розроблюваної платформи для розробки Telegram-бота для забезпечення розпізнавання тексту. Таким чином, питання розробки

програмного забезпечення та наукового супроводження на даний час можна вважати актуальним, за думкою автора.

1.3 Чат-боти

Останнім часом в месенджерах набирають популярність такі сервіси як чат-боти. Сам термін «чат-бот» був придуманий Майклом Молдінгом у 1994 році для опису розмовних програм. Так розробник Майкл Молдінг назвав свою електронну помічницю Julia [12]. Julia, звичайно, вміла робити не багато чого на той час – вона аналізувала написаний користувачем текст і зі свого боку відповідала фразами, які більш-менш відносилися до теми розмови. Чат-боти добре знайомі й завсідникам Інтернету кінця 90-х – початку 2000-х років: у IRC та IM вони розповідали анекдоти, влаштовували вікторини та «банили» злісних порушників правил чату.

Вивчивши сучасний стан використання чат-ботів в месенджерах, можна прийти до висновку, що чат-боти є універсальними засобами, здатними до вирішення різноманітних завдань – від спілкування до розваг, від надання медичної консультації до замовлення товарів та послуг за допомогою спеціалізованих прикладних рішень, від розпізнавання емоцій до вирішення складних консалтингових завдань в службах підтримки клієнтоорієнтованих інформаційних систем, а також забезпечення розпізнавання тексту, якщо це необхідно [13].

У будь-якому випадку, в незалежності від платформи, чат-бот — це прикладна програма, яка, отримуючи інформацію від користувача, формує коректні, логічно обґрунтовані відповіді. На сьогоднішній момент існує величезна різноманітність чат-ботів [13]. Деякі з можливих варіантів представлені нижче:

- ігрові чат-боти (квести або рольові ігри);
- рекламні чат-боти;
- новинні чат-боти;

•чат-боти для доставок, магазинів та сервісів (замовлення таксі, їжі, бронювання квитків, тощо);

•чат-боти для консультації та підтримки клієнтів;

•навчальні чат-боти.

Останнім часом все більші можливості відкривають інформаційні технології, пов'язані з чат-ботами. Вони настільки міцно увійшли в життя людей, що застосовуються в усіх сферах діяльності людини, і з кожним днем їх роль все більше зростає. В першу чергу, це пояснюється тим, що основну частину свого часу люди проводять за смартфоном в email-клієнтах та месенджерах [13].

Боти – це програми, які виконують різні завдання для користувача, що знаходиться в месенджері. Бот виглядає як звичайний чат, однак спілкування відбувається не з людиною, а з програмою, яка може приймати певні дані та видавати відповіді за запропонованими варіантами [12].

Зовсім нещодавно чат-боти здобули велику популярність, перевтілившись з розваги в більш серйозну річ, так як вони, в основному, стали використовуватися для вирішення серйозних бізнес-завдань. В епоху інформаційних технологій – це доволі нормальне явище, а тим більше у мережі Інтернет, оскільки суспільство давно перейшло на новий рівень ділового та повсякденного спілкування. По-перше, чат-боти – це «платформи» для вирішення певних рутинних завдань, що допомагають у взаємодії з користувачами. По-друге, чат-бот – це програма, яка підтримує діалог з користувачем, вибираючи відповіді з бази даних: ви формуєте певний запит і відразу отримуєте миттєву відповідь згідно зі сформованого запиту. Крім того, чат-боти виконують безліч корисних функцій у виконанні рутинних операцій, пошуку інформації, об'єднання даних, роботі зі користувачами, тощо [12].

Чат-бот як віртуальний співрозмовник має базу знань, яка представляє собою набори можливих питань користувача та відповідних їм відповідей. Найбільш поширеними варіантами для отримання потрібної відповіді є ключові слова, збіг фрази, збіг контексту або інші вказані параметри. У сучасному суспільстві для збору та відображення інформації можна використовувати чат-ботів. Звичайно, це робиться за допомогою спілкування з людьми, але аж ніяк не

завжди. Наприклад, в рамках якого-небудь заходу чат-боти можуть повідомляти всім учасникам новини та надавати довідкову інформацію [14].

1.4 Огляд існуючих технологій

InfoBot – з цим конструктором можна створити бота для відправки повідомлень, вміє відображати клавіатуру, можна переглядати діалог з користувачем та робити розсилки. Доступ для використання такої платформи пропонується розробниками на платній основі, але при реєстрації надається безкоштовна тріал-версія на десять днів. Мінімальний платний тариф складає 1200 грн. на місяць [15].

Під час тестування даної платформи створювати команди довелося з використанням інтуїції, оскільки відсутні підказки та плейсхолдери. Не весь функціонал даної платформи реалізовано гранично зрозумілим. Також слід зазначити, що зовнішній вигляд та можливості конструктора доволі примітивні (рис 1.9). Доступ до платформи з мобільних пристроїв також не реалізовано.

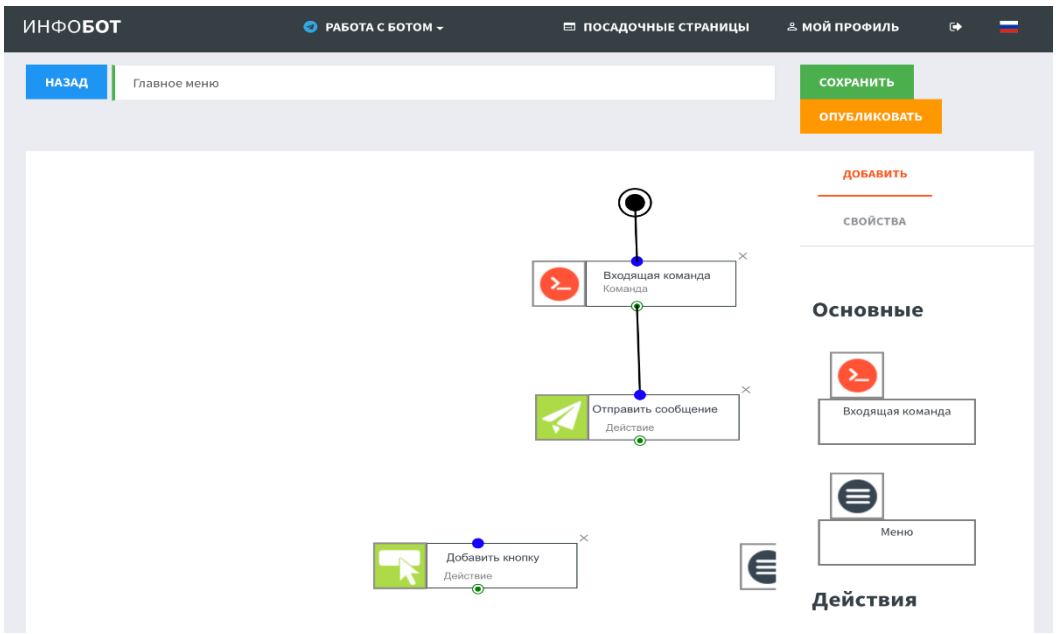


Рисунок 1.9 – Інтерфейс платформи InfoBot

Ebot one – дана платформа для створення та редагування Telegram-ботів дозволяє без навичок програмування сформувати структуру бота та реалізувати

її. Якщо певні навички програмування все ж є – логіку роботи можна зробити набагато складніше, а бота цікавішим й кориснішим для використання. Доступ для використання такої платформи пропонується розробниками на платній основі, але для щойно зареєстрованих користувачів є безкоштовний тариф, що обмежується використанням одного бота та неповним функціоналом самої платформи. Мінімальний платний тариф складає 100 грн. на місяць [16].

Слід також зазначити, що доступ до платформи з мобільних пристроїв не реалізовано. До того ж, платформа (рис. 1.10) має доволі складний функціонал. Останнє оновлення сервісу було ще у 2018 році.

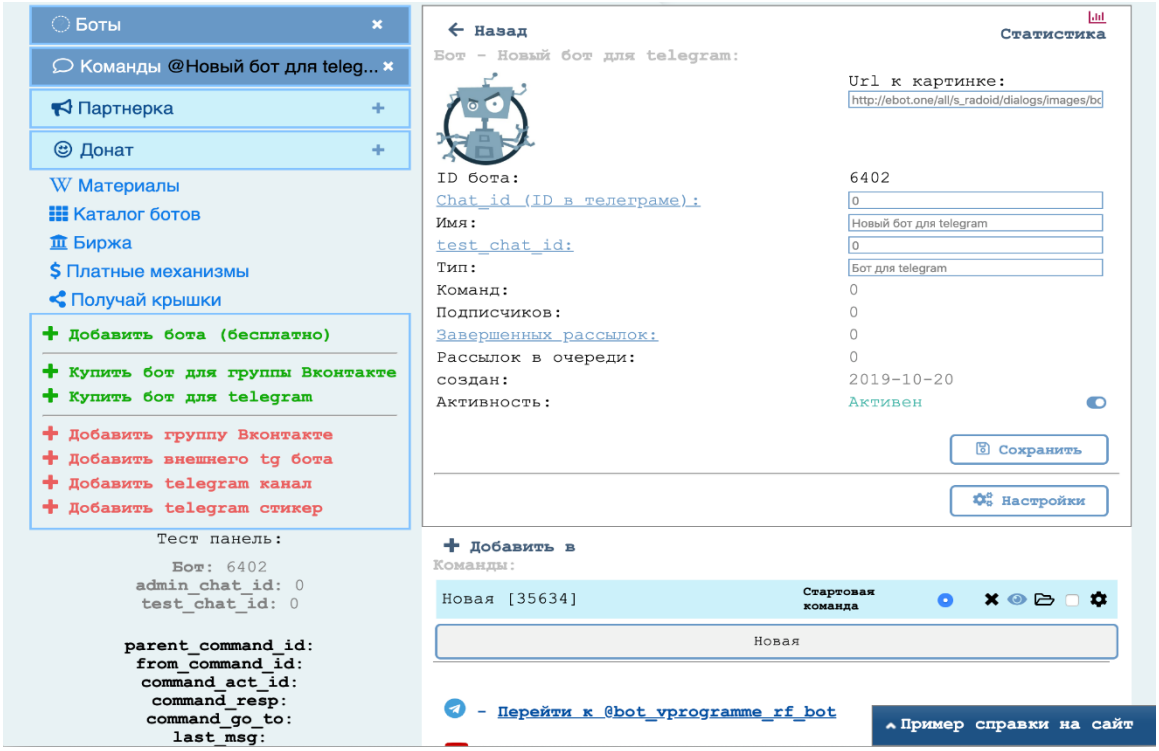


Рисунок 1.10 – Інтерефейс платформи Ebot one

Botkits – простий сервіс для створення простих ботів, які можуть надсилати різного роду інформацію (текст, зображення або документи) з можливістю прикріплювання клавіатури. Для усіх нових користувачів передбачено використання безкоштовного тарифу, на який діє обмеження в 50 вихідних повідомлень на добу та немає рекламного повідомлення сервісу. Мінімальний платний тариф для використання сервісу Botkits (рис. 1.11) складає 250 грн [17].

Щодо наявного функціоналу, це доволі простий сервіс з обмеженими функціональними можливостями для створення примітивних ботів. Більшість користувачів також відзначає, що дана платформа досить довго обробляє запити та завантажує дані, часто з'являються помилки з повідомленням, що сервер на даний час не відповідає, тому в повній мірі протестувати цей сервіс не вдалось.

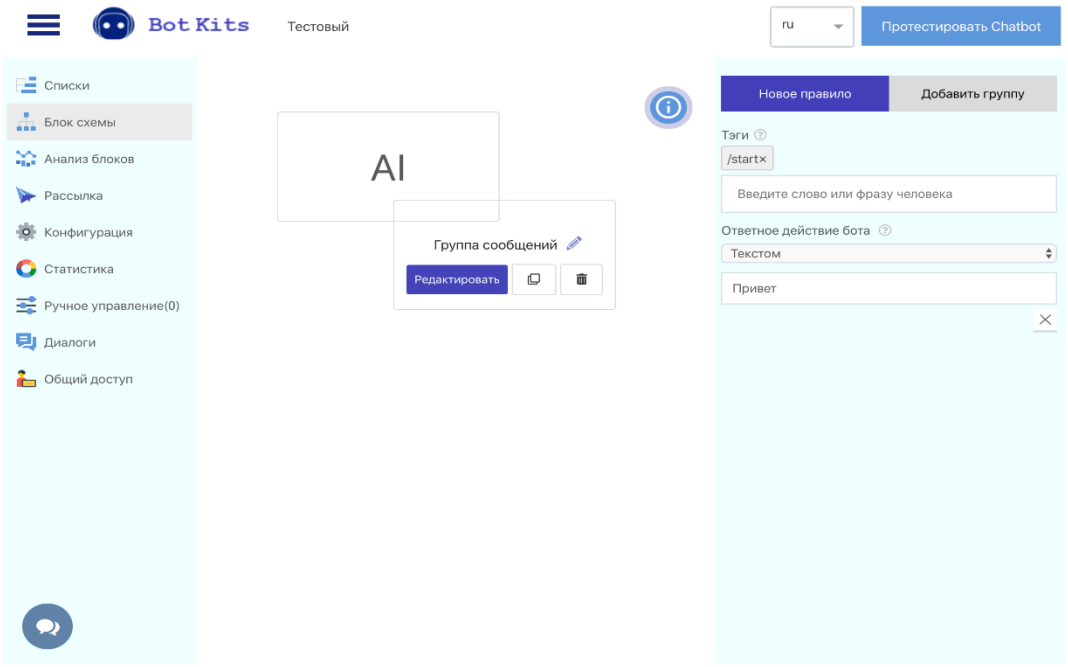


Рисунок 1.11 – Інтерефейс платформи Botkits

Manybot – особистий кабінет сервісу представлений у вигляді бота в Telegram. Конструктор дозволяє створювати меню, підменю, форму зворотного зв'язку, підключати RSS-стрічки та робити розсилки по всім користувачам, що є підписниками. Вартість використання такого функціоналу є безкоштовною, але включає в себе рекламу платформи, де створено бот. Змінити тариф або відключити рекламне повідомлення не можна.

Позитивним моментом є створена мобільна версія, що також представлена у вигляді сервісу, що працює через месенджер Telegram [18]. Дана платформа підходить для невеликих проектів, але створювати меню більше чотирьох рівнів через інтерфейс бота стає складно. Крім цього, бот довго відповідає або інколи не відповідає зовсім на деякі запити користувача. Слід також додати, що сервіс не оновлюється з 2015 року. Інтерфейс бота наведено на рис. 1.12.

Рисунок 1.12 – Інтерефейс платформи Manybot

Після стислого огляду кожного з конкуруючих проектів, автор вважає за потрібне навести порівняльну характеристику аналогів розроблюваного програмного продукту.

1.5 Формування вимог до програмного продукту

У сучасному світі значного розвитку набувають технології швидкісного обміну інформацією та повідомленнями між користувачами за допомогою мережі Інтернет. Ця тенденція актуальна не тільки для особистого користування, а й активно використовується у процесі підтримки навчального процесу по всьому світі.

Доцільність та важливість розроблюваного програмного продукту полягає в наступних аспектах:

1. Зручний, інтуїтивно зрозумілий інтерфейс. легкий у використанні та вивченні графічний інтерфейс є одним із надважливих пунктів для проекту, адже він дозволить використовувати Telegram-бот всім користувачам, незалежно від рівня їх компетенцій.

2. Незалежність від платформи. Telegram-бот може бути запущений на будь-якій платформі з мінімальними характеристиками, незалежно від використовуваної операційної системи та встановленого програмного забезпечення.

3. Необмежена кількість користувачів. Платформа Telegram дозволяє створювати сутності, до яких може приєднатися до 200 тис. незалежних користувачів, що в реальних умовах, зважаючи на розміри конкретного навчального закладу (та тим паче, навчальної дисципліни), є фактично безлімітною кількістю унікальних користувачів [19].

В розроблюваній системі заплановано перелічені нижче функціональні можливості, які будуть делегуватися на два рівні доступу користувачів – Адміністратор та Користувач. Адміністратор – це тип користувача, який має право:

- додавати та видаляти користувачів;
- надавати та обмежувати доступ до матеріалів;
- публікувати, видаляти та редагувати повідомлення у рамках цих матеріалів;
- дивитися, створювати та змінювати паролі для всіх груп користувачів.

За замовчуванням Адміністратор має доступ до повного функціоналу чат-бота (має право видаляти, створювати та редагувати повідомлення та інші дані). Також даний тип користувачів має доступ до панелі користувача, а також може змінювати свій особистий пароль в панелі Адміністратора.

Меню Адміністратора виглядає наступним чином:

- додавання та редагування контенту (додавання, редагування та видалення контенту в категоріях «Програма курсу», «Система оцінювання», «Лекції», «Матеріали по лабораторним, практичним або семінарських заняттям», «Література»);
- додавання адміністраторів курсу (для бота);
- повідомлення підписників (можливість створення повідомлень для підписників чат-бота);

•додавання нових команд користувача (розширення функціоналу бота за допомогою додавання нових команд користувача);

•перегляд користувачів бота.

Користувач – тип користувачів системи, який може використовувати бота з метою перегляду релевантної інформації щодо розпізнавання тексту, до яких має доступ. Доступ до розділів чат-бота налаштовується Адміністратором.

Меню Користувача виглядає наступним чином:

•перегляд програми дисципліни (виводиться меню з переліком дисциплін, що вивчає студент, далі можна вибрати бажану дисципліну та переглянути інформацію про неї);

•перегляд лекцій (виводиться меню з переліком дисциплін та відповідних лекцій до них, що вивчає студент, далі можна вибрати бажану дисципліну та переглянути інформацію про неї);

•ознайомлення з матеріалами по лабораторним, практичним або семінарським заняттям (виводиться меню з переліком дисциплін, що вивчає студент, далі можна вибрати бажану дисципліну та лабораторні, практичні або семінарські заняття та переглянути інформацію про них);

•пошук літератури (виводиться меню з переліком літературних джерел та дисциплін);

•створення повідомлення викладачеві (в даному випадку Користувач має можливість відправити приватне повідомлення викладачеві за допомогою чату);

•отримання допомоги (виводиться меню з переліком найбільш поширених запитань та відповідних відповідей на ці питання).

Для успішного використання розроблюваного рішення також необхідна наявність пристроїв, які будуть відповідати мінімальним вимогам для створюваної платформи для розробки телеграм-ботів для забезпечення розпізнавання тексту. З переліком мінімальних вимог для використання месенджера Telegram, як основи, на базі якої й розроблюється програмний продукт, можна ознайомитись при встановленні застосунку Telegram у офіційних джерелах.

Висновки до розділу 1

В першому розділі розповідається про історію створення месенджерів, про розробку чат-ботів. Розглянуті аналоги та визначені вимоги до програмного забезпечення.

Останнім часом все більші можливості відкривають інформаційні технології, пов'язані з чат-ботами. Вони настільки міцно увійшли в життя людей, що застосовуються в усіх сферах діяльності людини, і з кожним днем їх роль все більше зростає. В першу чергу, це пояснюється тим, що основну частину свого часу люди проводять за смартфоном в email-клієнтах та месенджерах [13].

Боти – це програми, які виконують різні завдання для користувача, що знаходиться в месенджері. Бот виглядає як звичайний чат, однак спілкування відбувається не з людиною, а з програмою, яка може приймати певні дані та видавати відповіді за запропонованими варіантами [12].

РОЗДІЛ 2

ВИБІР ТЕХНОЛОГІЙ ТА СЕРЕДОВИЩА РОЗРОБКИ ПЛАТФОРМИ

2.1 Обрані технології розробки

2.1.1 Мова програмування Python та бібліотеки

Python – високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника та сприйняття коду. Синтаксис ядра Python мінімалістичний. У той же час, стандартна бібліотека включає великий обсяг корисних функцій [20].

Python підтримує структурне, узагальнене, об'єктно-орієнтоване, функціональне та аспектно-орієнтоване програмування. Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень, високорівневі структури даних, тощо. Підтримується розбиття програм на модулі, які, в свою чергу, можуть об'єднуватися в пакети.

Розробка мови Python розпочалась наприкінці 1980-х років співробітником голландського інституту CWI Гвідо ван Россумом [21]. Для розподіленої ОС Amoeba потрібно було створити розширювану скриптову мову, і Гвідо почав писати Python як хобі, запозичивши деякі напрацювання для мови ABC (Гвідо брав участь в розробці цієї мови, орієнтованої на навчання програмуванню). У лютому 1991 року розробник опублікував вихідний текст в групі новин alt.sources. З самого початку Python проектувався як об'єктно-орієнтована мова.

Розвиток мови відбувається згідно чітко регламентованого процесу створення, обговорення, відбору та реалізації документів PEP (англ. Python Enhancement Proposal) – пропозицій щодо розвитку Python.

Слід також зазначити, що назва мови програмування пішла не від назви сімейства плазунів. Автор назвав мову на честь популярного британського комедійного телешоу 1970-х «Літаючий цирк Монті Пайтона» [21]. Втім, все одно назву Python частіше пов'язують саме зі змією, ніж з передачею — піктограми файлів в KDE або в Microsoft Windows та навіть емблема на сайті

python.org (до виходу версії 2.5) зображують зміїні голови. Важлива мета розробників Python – створити мову програмування цікаву для використання. Це відображено в його назві, яка прийшла з Монті Пайтона.

Приклад програми на мові програмування Python наведено на рис. 2.1.

Рисунок 2.1 – Приклад програми на мові Python

OpenCV (Open Source Computer Vision Library) — бібліотека з відкритим кодом функцій і алгоритмів для комп'ютерного зору, обробки зображень і числових алгоритмів загального призначення.

OpenCV створено для розробки загальної інфраструктури для програм комп'ютерного зору та прискорення використання машинного сприйняття в комерційних продуктах. Будучи ліцензованим продуктом BSD, OpenCV полегшує підприємствам використання та зміну коду [22].

Бібліотека містить понад 2500 оптимізованих алгоритмів, включаючи повний набір як класичного, так і сучасного комп'ютерного зору та алгоритмів машинного навчання. Їх можна використовувати для виявлення та розпізнавання облич, ідентифікації об'єктів, класифікації дій людини на відео, відстеження рухів камери та рухомих об'єктів, вилучення 3D-моделей об'єктів, отримання точок 3D хмар із стереокамер, спільного захоплення зображень для отримання зображення високої роздільної здатності. всю сцену, знаходьте схожі зображення з бази даних зображень, видаляйте червоні очі зі спалахів, відстежуйте рухи очей, розпізнавайте пейзажі та встановлюйте маркери для покриття доповненої реальності та багато іншого.

OpenCV має понад 47 000 користувачів спільноти та понад 18 мільйонів завантажень. Бібліотека широко використовується компаніями, науковими групами та державними установами.

Поряд із такими відомими компаніями, як Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota, які використовують бібліотеку, є багато стартапів, таких як Applied Minds, VideoSurf і Zeitera, які широко використовують OpenCV.

OpenCV має інтерфейси C++, Python, Java і MATLAB і підтримує Windows, Linux, Android і MacOS. В основному він орієнтований на програми перегляду в реальному часі та використовує інструкції MMX та SSE, якщо вони доступні.

Зараз активно розробляються повнофункціональні інтерфейси CUDA і OpenCL. Існує понад 500 алгоритмів і приблизно в 10 разів більше функцій, які складають або підтримують ці алгоритми.

OpenCV написаний на C++ і має інтерфейс шаблону, який легко працює з контейнерами STL [22].

2.1.2 Бібліотека os

Модуль os зі стандартної бібліотеки мови програмування Python зазвичай використовується для роботи зі встановленою ОС, а також файловою системою ПК. Він містить масу корисних методів для взаємодії з файлами та папками на жорсткому диску. Програми, що працюють з модулем os, не залежать від типу ОС та легко переносяться на іншу платформу.

Модуль os в Python – це бібліотека функцій для роботи з операційною системою [22]. Методи, включені в неї, дозволяють визначати тип операційної системи, отримувати доступ до змінних оточення, управляти директоріями та файлами:

- перевірка існування об'єкта згідно заданого шляху;
- визначення розміру в байтах;
- видалення;
- перейменування та інше.

При виклику функцій os необхідно враховувати, що деякі з них можуть не підтримуватися поточною операційною системою.

Щоб користуватися методами з os, потрібно підключити бібліотеку. Для цього в Python використовується import os, який необхідно описати в файлі до першого звернення до модуля. Рекомендується використовувати цю інструкцію на початку файлу з вихідним кодом. Приклад реалізації модулю зображено на рис 2.2.

```
import os
print(os.name)

nt
```

Рисунок 2.2 – Приклад отримання інформації про ОС

2.1.3 Модуль logging

Пакет Logging є дуже корисним інструментом в наборі інструментів програміста. Він може допомогти розробнику краще зрозуміти суть програми та виявити сценарії, про які програміст, можливо, навіть не замислювалися при розробці програмного продукту [24].

Логи надають розробникам додатковий набір «очей», які постійно «дивляться» на потік, через який проходить додаток. Вони можуть зберігати інформацію про те, який користувач або IP отримав доступ до додатка. Якщо виникає помилка, то вони можуть надати більше інформації, ніж трасування стека, повідомивши програмісту, в якому стані перебувала програма до того, як вона досягла рядки коду, де сталася помилка.

Записуючи корисні дані з потрібних місць, розробник може не тільки легко налагоджувати помилки, але й використовувати дані для аналізу продуктивності додатка, планування масштабування або перегляду схем використання для планування маркетингу.

Модуль logging в Python – це готовий до використання, потужний модуль, призначений для задоволення потреб як початківців, так й корпоративних

команд. Він використовується більшістю сторонніх бібліотек Python, тому можна інтегрувати логи з повідомленнями з цих бібліотек для створення єдиного журналу логів розроблюваного додатку.

З імпортованим модулем logging (рис. 2.3) можна використовувати інструмент, що називається «logger», для логування повідомлень, які необхідно переглядати. За замовчуванням існує 5 стандартних рівнів severity, що вказують на важливість подій, а саме debug, info, warning, error та critical. У кожного є відповідний метод, який можна використовувати для логування подій на обраному рівні severity.

```
import logging

logging.debug('This is a debug message')
logging.info('This is an info message')
logging.warning('This is a warning message')
logging.error('This is an error message')
logging.critical('This is a critical message')
```

Рисунок 2.3 – Приклад підключення та запиту за допомогою модулю logging

2.1.4 Бібліотека SQLAlchemy

SQLAlchemy – це бібліотека на мові програмування Python для роботи з реляційними СУБД з застосуванням технології ORM. Служить для синхронізації об'єктів Python та записів реляційної бази даних. SQLAlchemy дозволяє описувати структури баз даних та способи взаємодії з ними на мові Python без використання SQL [25].

Використання SQLAlchemy (рис. 2.4) для автоматичної генерації SQL-коду має кілька переваг у порівнянні з ручним написанням SQL:

•Безпека. Параметри запитів виводяться на екран, що робить атаки типу впровадження SQL-коду малоймовірними.

•Продуктивність. Підвищується ймовірність повторного використання запиту до сервера бази даних, що може дозволити йому в деяких випадках застосувати план виконання запиту повторно.

•Переносимість. SQLAlchemy, при належному підході, дозволяє писати код на Python, сумісний з декількома back-end СУБД. Незважаючи на стандартизацію мови SQL, між базами даних є відмінності в його реалізації, абстрагуватися від яких і допомагає SQLAlchemy.

```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///:memory:')
>>> engine.execute("select 'Hello, World!'").scalar()
u'Hello, World!'
```

Рисунок 2.4 – Простий приклад використання бібліотеки SQLAlchemy

2.2 Telegram Bot API

Bot API представляє собою HTTP-інтерфейс для роботи з ботами в Telegram. Кожен бот – це спеціальний аккаунт, створений для оброблення та відправлення повідомлень. Існує два протилежних за логікою способу отримання оновлень від бота:

•long pulling – додаток автоматично опитує сервера Telegram на наявність будь-яких оновлень для бота, де час затримки між запитами складає за замовчуванням 100 мс;

•webhook – сервера Telegram самі сповіщають додаток на сервері, як тільки з'являться будь-які оновлення.

Вхідні оновлення будуть зберігатися на сервері до тих пір, поки їх не оброблять, але не довше 24 годин. Незалежно від способу отримання оновлень, у відповідь відправляється об'єкт Update, серіалізований в JSON [26].

Всі запити до Telegram Bot API повинні здійснюватися через HTTPS в наступному вигляді: https://api.telegram.org/bot<token>/НАЗВА_МЕТОДУ.

Принцип роботи взаємодії чат-бота і користувача зображений на рис 2.5.

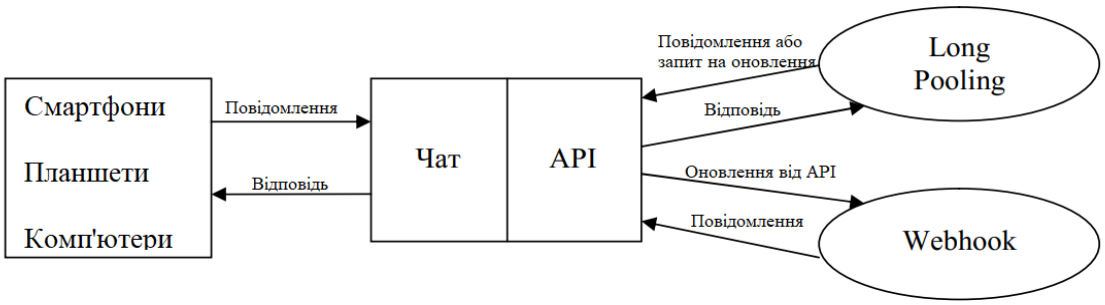


Рисунок 2.5 – Принцип роботи Telegram BOT API

Для того, щоб отримати token, необхідно написати спеціальному боту @BotFather.

Приклади доступних для API методів описані нижче:

•getUpdates – цей метод використовується для отримання оновлень за технологією long polling;

•setWebhook – метод прив'язує до боту url домену, де міститься запущений бот;

•sendMessage – метод відправляє текстове повідомлення до клієнту Telegram;

•sendLocation – метод відправляє повідомлення з координатами до клієнту Telegram;

getFile – метод повертає долучення по його імені та інше.

2.3 Ознайомлення з середовищами для розробки програмного додатку

Для реалізації платформи для розробки Telegram-бота для забезпечення інформаційної підтримки розпізнавання тексту було обрано середовище розробки PyCharm.

PyCharm – це найстабільніша інтелектуальна Python IDE з повним набором засобів для ефективної розробки застосунків на мові програмування Python. Випускається в двох варіантах – безкоштовна версія PyCharm Community Edition, та розширена версія додатку за підпискою, що підтримує більший набір

можливостей PyCharm Professional Edition. PyCharm виконує інспекцію коду на льоту, автодоповнення, в тому числі ґрунтуючись на інформації, що була отримана під час виконання коду, навігацію по коду, забезпечує безліч рефакторингів, тощо [27].

PyCharm також являє собою інтегроване середовище розробки для мови програмування Python. Надає засоби для аналізу коду, графічний відладчик, інструмент для запуску юніт-тестів та підтримує веб-розробку на Django. PyCharm розроблена компанією JetBrains на основі IntelliJ IDEA. Більше 10 років розробки платформи IntelliJ надає PyCharm більше 50 найрізноманітніших плагінів, включаючи підтримку додаткових VCS, інтеграції з різними інструментами та фреймворками, редактором оновлень, таким як емуляція Vim.

Рисунок 2.6 — PyCharm

Ключові можливості середовища розробки:

•потужний та функціональний редактор коду з підсвічуванням синтаксису, авто-форматуванням та авто-відступами для підтримуваних мов;

•проста й потужна навігація в коді;

•допомога при написанні коду, що включає в себе автодоповнення, авто-імпорт, шаблони коду, перевірка на сумісність версії інтерпретатора мови, та багато іншого;

•швидкий перегляд документації для будь-якого елемента прямо у вікні редактора, перегляд зовнішньої документації через браузер, підтримка docstring – генерація, підсвічування, автодоповнення та багато іншого;

•велика кількість інспекцій коду;

•потужний рефакторинг коду, який надає широкі можливості щодо виконання швидких глобальних змін у проекті [27].

2.4 Технічні вимоги

Після завантаження ПК він знаходиться в реальному режимі, де адресація обмежена 1 МБ і немає захисту розробленого коду. Реалізувати в режимі дискової операційної системи (DOS) досить просто. Щоб завантажувати та запускати великі програми та отримувати доступ до більшого обсягу пам'яті, необхідно навчитися переходити до захищеного режиму за допомогою інструкцій (код програми) мовою асемблера.

Однак усі основні виклики вводу-виводу (BIOS) доступні лише в реальному режимі. Якщо потрібно використовувати BIOS для введення-виводу, необхідно мати механізми для перемикання з безпечного режиму на реальний для виконання викликів BIOS.

Отже, програма на Python або якась керуюча програма повинні виконувати ці інструкції.

Якщо при використанні програмного переривання (255) замість переривань BIOS, необхідно написати на складання власний код для обробки всіх операцій введення-виводу.

У разі компіляції, щоб мати можливість прочитати документ специфікації архітектури Intel, доступний на їхньому веб-сайті. Цей документ є корисним

ресурсом для розуміння та реалізації переривань, схем адресації, інструментів та винятків.

У звичайну програму на Python включаємо файли, що становлять бібліотеки для ОС, які будуть включені в файл, що виконується. Якщо ми будемо використовувати будь-які конструктори Python, вони автоматично включатимуть бібліотеки, які не мають сенсу в середовищі без ОС. Отже, середовище компіляції має використовувати пакетні файли для компіляції та зв'язування необхідних модулів. Як програміст Python програм, ми розробили файл інтерфейсу, який містить всі інтерфейси, необхідні для цих програм. Цей файл має API, у тому числі: введення-виведення, завдання, пам'ять, таймер, загальну пам'ять, блокування і т. д. Для створення файлів ми використовуємо компілятор Visual Studio Python (пакетний режим), асемблер MASM 6.11 і компілятор Turbo. Ми написали всі пакетні файли для компіляції та компіляції для завантаження, вихідні коди програм та програм. Ці пакетні файли прості та легкі для розуміння та використання. API є викликом Python, який, у свою чергу, запускає виклик C, який, у свою чергу, запускає виклик складання. Програміст на Python бачить лише API. Якщо інтерфейс можна створити на рівні Python, код буде реалізований лише на рівні Python. Ми уникаємо використання будь-яких прямих дзвінків збірки в Python; однак такий виклик дозволено компілятором.

Характеристики комп'ютера, необхідні для стабільної та швидкої роботи програми, повинні відповідати вимогам, зазначеним у таблиці 2.1.

Таблиця 2.1 - Системні вимоги

Складова комп'ютера	Характеристика
CPU	Intel чи AMD 4-core, 3.0GHz або краще
GPU	NVIDIA GeForce 750ti, 2Gb VRAM або краще
RAM	8Gb або більше
Простір на диску	150Mb
Додаткове ПЗ	Visual Studio 2019
Операційна система	Windows 10, macOS High Sierra, Linux (Ubuntu, CentOS, RedHat)

Розробка програма здійснювалася відповідно до встановлених параметрів.

2.5 Опис архітектури програмного забезпечення

Проведемо проектування програмного забезпечення. Основним та найважливішим етапом проектування програмного забезпечення є вибір архітектури.

Архітектура програмного забезпечення - це процес структурування програмного забезпечення на слабко пов'язані та незалежні частини, що утворюють зв'язки та описують процеси передачі даних між ними.

Архітектура програмного забезпечення побудована таким чином, щоб максимально відповідати вимогам проекту.

Відповідно до сформованих функціональних вимог було прийнято рішення розглянути такі архітектурні шаблони та стилі:

1. Клієнт-серверна архітектура.
2. Front та Back end.
3. Монолітна/мікросервісна архітектура.
4. MVC.

Архітектура клієнт-сервер є домінуючою концепцією створення розподілених мережних додатків, що встановлює правила взаємодії та обміну даними між ними.

Передбачені такі компоненти:

1. Набір серверів, які надають інформацію на запит.
2. Набір клієнтів, які використовують сервери та отриману від них інформацію.
3. Мережа, що забезпечує обмін інформацією між клієнтами та серверами.

Система управління чат-ботом має три сторони:

1. API ботів Telegram.
2. Сервер із базовою логікою чат-бота.
3. Клієнт.

Серверна частина - це сервери Telegram по відношенню до сервера логіки чат-бота, сервер логіки чат-бота по відношенню до веб-інтерфейсу клієнта, а клієнти - це сервери логіки чат-бота по відношенню до серверів Telegram та веб-інтерфейс клієнта по відношенню до сервери логіки чату. бот (рис. 2.1).

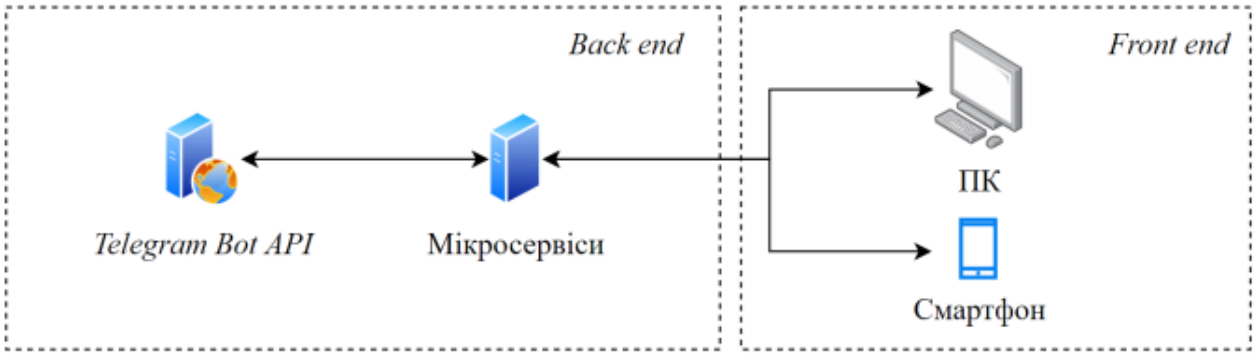


Рисунок 2.7 - Клієнт-серверний погляд на архітектуру системи

Клієнтський веб-інтерфейс відповідає за рівень представлення даних (відображення даних та введення команд), Мікросервіси – за рівень управління даними та прикладний рівень (зберігання даних та логічна обробка).

Монолітна/мікросервісна архітектура. При проектуванні великих систем чи систем із великою кількістю незакріплених компонентів дуже важливо вибрати базову архітектуру. За роки розробки програмного забезпечення сформувалися два основних підходи.

Монолітна архітектура — це програмна архітектура, коли всі компоненти програми перебувають у межах процесу ОС і обмінюються даними через внутрішні інтерфейси.

Використання монолітної архітектури має такі переваги:

1. Спрощений процес розробки. Сучасні інтегровані середовища розробки (IDE) насамперед підтримують розробку монолітних програм.

2. Спрощений процес розгортання.

3. Спрощений процес масштабування. Програму можна масштабувати, запускаючи кілька копій на балансувальнику навантаження.

Однак зі збільшенням складності та розміру програмного забезпечення монолітна архітектура набуває недоліків:

1. Складність у підтримці, розумінні та доопрацюваннях при великій кодовій базі програми.

2. Уповільнення швидкості IDE, процесу завантаження програм.

3. Запобігання частим оновленням програми, оновлення одного компонента заважає роботі інших через необхідність перезапуску всієї програми.

4. Помилка в одному компоненті призводить до відмови всієї системи.

5. Масштабування програми можливе лише в одній площині.

6. Складність зміни технологій та їх конкретних версій.

Мікросервісна архітектура - це програмна архітектура, в якій всі її компоненти логічно розділені для виконання слабо пов'язаних функцій і виконання різних процесів ОС. Мікросервіси однієї програми можуть працювати на різних фізичних машинах, а обмін між ними ґрунтується на стандартизованих протоколах RPI або обміну повідомленнями.

Використання мікросервісної архітектури має такі переваги:

1. Створення невеликих слабопов'язаних компонентів спрощує їх розуміння, розробку, тестування, підтримку, оновлення та розгортання.

2. Прискорити розробку за рахунок збільшення швидкості IDE та скорочення часу завантаження мікросервісів.

3. Підвищення надійності системи. Відмова або помилка в одному компоненті ізольовані від втручання у роботу інших.

4. Можливість заміни або оновлення технологій та фреймворків без критичного впливу на терміни або роботу проекту.

Недоліки мікросервісної архітектури:

1. Необхідно створити механізми обміну даними між мікросервісами.

2. Ускладнює синхронізацію даних.

3. Ускладнюється процедура розгортання мікросервісної програми.

Відповідно до завдання дисертації, вимог та аналізу недоліків монолітної архітектури робиться висновок про те, що використання мікросервісної архітектури дасть наступні переваги:

1. Забезпечити незалежне розроблення кожного компонента системи.

2. Приведе до можливості більш простого масштабування за так званими осями X, Y та Z у порівнянні з монолітною архітектурою (рис. 2.8) [13].

3. Забезпечити використання сучасних технологій та фреймворків, з можливістю легкої заміни за потреби.

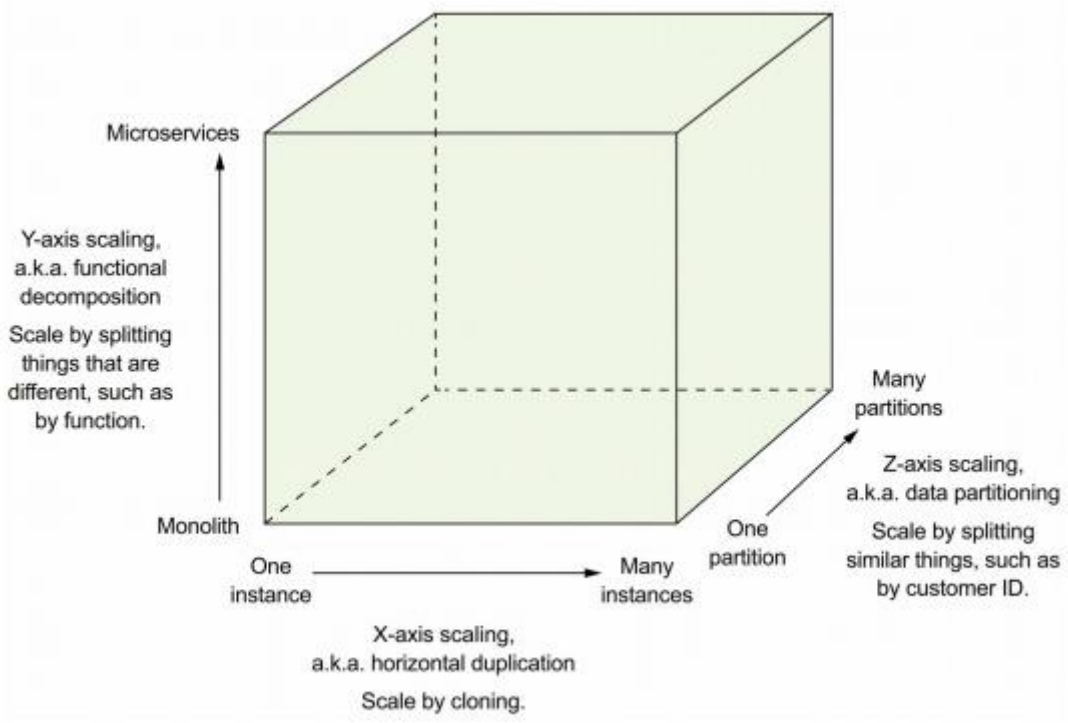


Рисунок 2.8. Масштабування мікросервісних програм

Model-View-Controller (MVC) – шаблон розробки програмного забезпечення, що розділяє класичне відображення, зберігання та обробку даних на окремі взаємопов'язані частини (рис. 2.9).

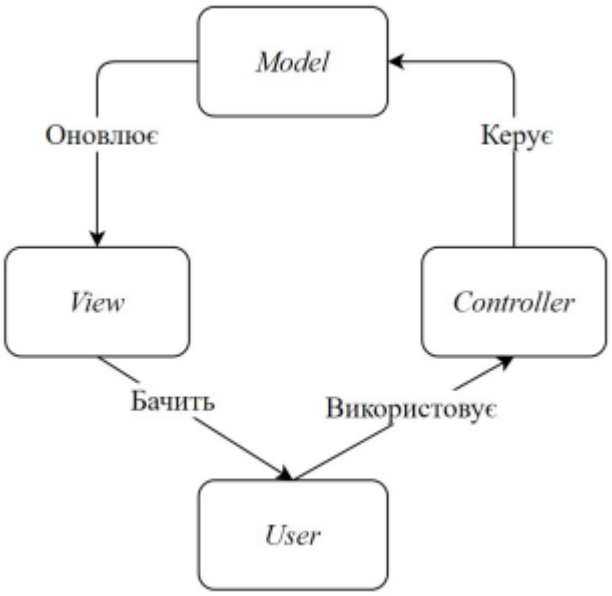


Рисунок 2.9. Діаграма дій MVC

Модель – це та частина, яка відповідає за зберігання даних, та яка оновлюється залежно від дій користувача (елементи керування – контролер).

Вид - це та частина, яка відповідає за відображення даних (статусу моделі) користувачеві. Вигляд оновлюється лише при оновленні моделі.

Контролер - це та частина, яка відповідає за обробку дій користувача і найчастіше виступає в ролі стандартних елементів управління інтерфейсу користувача (кнопки, поле введення, перемикачі і т. д.).

REST API

REST — це стиль архітектури програмного забезпечення, що використовується для створення розподілених веб-сервісів, що масштабуються, що використовують HTTP-запити.

REST (Representational State Transfer) — стиль взаємодії компонентів розподіленої програми у мережі. REST має узгоджений набір обмежень, які враховуються під час розробки розподіленої системи гіпермедіа. У певних випадках (інтернет-магазини, пошукові системи) це призводить до підвищення продуктивності та зміни архітектури, а саме її спрощення.

Компоненти в REST своєю взаємодією нагадують взаємодію клієнта та сервера в Інтернеті.

Виклик віддаленої процедури в Інтернеті може бути простим запитом HTTP (зазвичай «GET» або «POST»; такий запит називається «REST-запитом»), а всі дані, необхідні для передачі, передаються як параметри запиту. Для сервісів, створених з урахуванням REST, використовується так званий термін RESTful.

На відміну від веб-служб на основі SOAP, немає єдиного офіційного стандарту для терміну веб-API RESTful. Тому що REST – це архітектурний стиль, а SOAP – це протокол. REST сам не є стандартом, і, незважаючи на це, у більшості реалізацій RESTful використовуються добре відомі стандарти, такі як HTTP, URL, JSON і XML.

Оскільки розроблена система насамперед є програмним інструментом для перетворення стилів, її можна використовувати для будь-якої архітектури, оскільки вона заснована на алгоритмі та інструментах його обробки, однакових для будь-якої платформи – Web, Desktop, Mobile або Embedded. Отже, програмне забезпечення, що розробляється, має являти собою єдине ядро машинного навчання, що має програмний інтерфейс (API), призначений для забезпечення реалізації наступної логіки:

- Початок з вхідного зображення, обробка, завершення з виходом;
- Перетворене зображення.

Готове програмне забезпечення можна розмістити на будь-якій платформі, додавши додаткове програмне забезпечення – сервер, плагін або інтерфейс програми.

Клієнт-серверна архітектура підходить для організації на Web-платформі.

Для цього необхідно створити середовище у вигляді контейнера Docker з необхідною операційною системою, розмістити перетворювач стилів, сервер, який керуватиме перетворювачем через свій програмний інтерфейс, та веб-сторінку для доступу та управління користувачами.

Звичайно, хостинг повинен підтримувати зовнішні статичні адреси. Також можна об'єднати перетворювач стилю та сервер в одному процесі, але це знизить загальну гнучкість системи. Загальна схема такої архітектури представлена нижче.

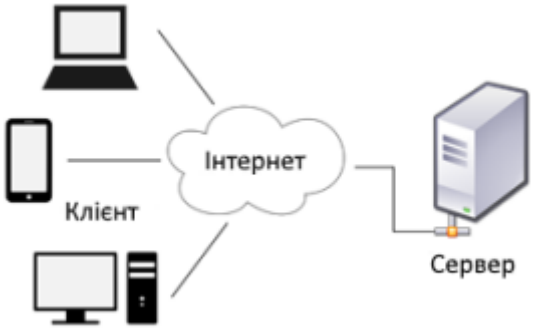


Рисунок 2.10 – Загальна схема клієнт-серверної архітектури

Наступна версія платформи – Desktop та Mobile. З архітектурного погляду вони однакові. Обидва мають високорівневу розробку для конкретної операційної системи. Як і в веб-платформі, перетворювач стилів працюватиме як дочірній процес основної програми, який міститиме графічний інтерфейс користувача та всю необхідну бізнес-логіку. Така архітектура називається міжпроцесною взаємодією.

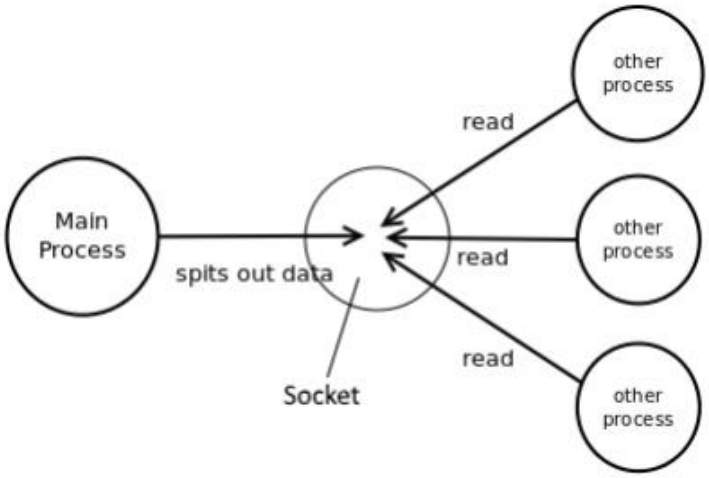


Рисунок 2.11 – Загальна схема міжпроцесної взаємодії

Остання платформа Embedded. Він передбачає розробку системи на найнижчому апаратному рівні. Це також зводиться до взаємодії між процесами, крім того, що ресурси дуже обмежені разом з інструментами розробки, але продуктивність такої системи найвища.

Хорошим рішенням було б використовувати апаратне прискорення для перетворювача стилю, чи то потужний графічний процесор, чи будь-який спеціалізований прискорювач штучного інтелекту.

Виходячи з вимог кроссплатформенного перетворювача, що розробляється, і простоти реалізації готового програмного забезпечення, що демонструє його роботу, для розробки була обрана платформа Desktop. Готове програмне забезпечення матиме власний програмний інтерфейс та виконуватиме дочірній процес програми з графічним інтерфейсом користувача. Windows була обрана як цільова операційна система.

Черга повідомлень

Черга повідомлень (MQ) — це архітектура обміну повідомленнями між різними компонентами програмних систем в асинхронному режимі. Це дозволяє відправити повідомлення в один відправляючий компонент системи в один момент часу, а прийняти та обробити інший приймаючий компонент у зовсім інший. Системи цього типу складаються з виробника (відправника) та споживача (одержувача), які взаємодіють один з одним через провайдера (брокера), який інакше можна визначити як сервер MQ.

Черга - це структура даних з обмеженим доступом до елементів, яку можна описати як "перший прийшов, першим обслужений". Обмеження в тому, що елемент можна додати тільки до кінця черги, а вибрати елемент тільки з початку, при видаленні елемента з черги він знищується.

Особливості використання MQ-архітектури:

а) слабке зв'язування - створюються неявні інтерфейси, які обмінюються даними, дозволяють процесам бути незалежними один від одного та дозволяють підтримувати потрібний формат повідомлень;

б) надмірність - дозволяє уникнути нераціонального використання ресурсів системи чи мережі, своєю чергою, зберігання необробленої інформації;

в) масштабованість - за рахунок розподілу процесів обробки інформації дозволяє збільшити продуктивність MQ-сервера;

г) еластичність і здатність витримувати пікові навантаження - при високому навантаженні крім цього черги повідомлень можуть бути так званим буфером

для накопичення інформації, дозволяючи змінювати швидкість обробки інформації і, таким чином, знижувати загальне навантаження на систему або мережу;

д) стабільність – дозволяє відокремити один процес від іншого та отримувати повідомлення, таким чином, у разі відмови процесора є можливість відновити робочий стан системи, відклавши обробку повідомлень;

е) гарантована доставка – повідомлення у будь-якому разі будуть оброблені та доставлені, незалежно від робочого статусу системи-відправника та системи-одержувача повідомлень. Це досягається за рахунок використання асинхронного зв'язку та можливості зберігати повідомлення на сервері MQ доти, доки воно не буде оброблено;

ж) гарантована та одноразова процедура доставки - значна частина серверів MQ дозволяє доставляти дані в тому порядку, в якому вони були надіслані, при цьому гарантуючи, що після прочитання повідомлення воно буде видалено з черги;

з) буферизація - це гарантія доставки повідомлень з максимальною ефективністю, яка досягається за рахунок незалежності швидкості обробки повідомлень від швидкості їх відправлення, це своє чергу пов'язано з використанням вищеописаної структури черги - своєрідного буфера між системою обробника і системою відправника.

2.5.1 Опис класів

Клас на Python – це будівельний блок, який веде до об'єктно-орієнтованого програмування. Це визначений користувачем тип даних, який містить власні дані-члени і функції-члени, до яких можна отримати доступ і які можна використовувати, створивши екземпляр цього класу. Клас Python виглядає як план об'єкта.

Наприклад: розглянемо клас автомобілів. Може бути багато автомобілів з різними назвами та марками, але всі вони матимуть деякі спільні риси,

наприклад, всі вони будуть мати 4 колеса, обмеження швидкості, пробіг тощо. Отже, машина – це клас та колеса, обмеження швидкості, пробіг. їх властивості.

Клас - це визначений користувачем тип даних, який має дані-члени та функції-члени.

Дані-члени - це змінні даних, а функції-члени - це функції, які використовуються для управління цими змінними, і разом ці дані-члени та функції-члени визначають властивості та поведінку об'єктів у класі.

У прикладі з класом Car елементом даних буде обмеження швидкості, пробіг тощо, а функціями-членами може бути гальмування, збільшення швидкості тощо.

Об'єкт є екземпляром класу. При визначенні класу пам'ять не виділяється, але за його створенні (тобто створенні об'єкта) пам'ять виділяється.

Визначення класу та оголошення об'єкта

Клас визначається в Python за допомогою ключового слова class, за яким слідує ім'я класу. Тіло класу визначається всередині фігурних дужок і закінчується крапкою з комою в кінці.

Оголошення об'єкта: щодо класу визначається лише специфікація об'єкта; пам'ять чи сховище не виділено. Ви повинні створювати об'єкти для використання даних та функцій доступу, визначених у класі.

Синтаксис:

ІмяКласуІмяОб'єкта;

Доступ до членів даних та функцій-членів. Доступ до членів даних та функцій-членів класу можна отримати за допомогою оператора точки ('.') з об'єктом. Наприклад, якщо ім'я об'єкта є obj і ви хочете отримати доступ до функції-члена з ім'ям printName(), вам потрібно ввести obj.printName().

Доступ до членів даних

Доступ до відкритих елементів даних здійснюється так само, але об'єкт не може мати прямого доступу до закритих елементів даних. Доступ до елемента даних залежить лише від керування доступом до цього елемента даних.

Цей контроль доступу забезпечується модифікаторами доступу до Python. Існує три модифікатори доступу: відкритий, закритий та безпечний.

Клас - це шаблон або план, який пов'язує властивості та функції сутності. Ви можете помістити всі сутності або об'єкти зі схожими атрибутами під один дах, відомий як клас. Класи додатково реалізують базові концепції, такі як інкапсуляція, приховування даних та абстракція. Python клас діє як тип даних, який може мати кілька об'єктів або екземплярів типу класу.

2.5.2 Опис методів

Написання безпечного коду, що забезпечує безпеку без уразливостей, є критичним завданням для кібербезпеки. Написання коду без уразливостей вже давно не менш складне, ніж написання коду без помилок. Хоча у програмному забезпеченні існує багато інших потенційних джерел ризику безпеки, розробка коду без відомих класів уразливостей завжди здавалася життєздатною метою. Він покладається на те, що розробники використовують інструменти, методи та процеси для створення програмного забезпечення, яке не має загальновідомих типів дефектів.

Один з найефективніших підходів — вивчення мов та інструментів програмування — навів технології, які, як було показано, протистоять вразливості, переважно шляхом їх запобігання. Безпечні мови пам'яті, які контролюють виділення та звільнення пам'яті, замість того, щоб вимагати цього від програміста, унеможливлюють створення розробниками вразливостей переповнення буфера та деяких інших типів впливу через відсутність перевірок меж масиву, нульового покажчика та використання даних. витік через повторне використання пам'яті. Потокові мови можуть обробляти випадки, коли умови гонки можуть використовуватись для підриву перевірок безпеки у програмі.

У спільноті розробників програмного забезпечення групи та організації з розробки безпечного програмного забезпечення включили інструменти та методи у свій життєвий цикл розробки програмного забезпечення, щоб увімкнути життєвий цикл безпечної розробки. Ранне високонадійне програмне

забезпечення використовувало формальні методи для визначення функцій безпеки системи, а також перегляд коду, щоб використовувати людей для виявлення таких недоліків на рівні кодування.2 Microsoft побудувала свій життєвий цикл розробки безпеки, додавши аналіз першопричин, навчання моделювання, конкретні вимоги до безпечного кодування та тестування безпеки, включаючи тестування на проникнення та нечітке тестування. Практика зазвичай реалізується на основі потреб бізнесу, значно впливає на безпеку і узгоджується з встановленими або методами розробки, що розвиваються.

Потрібні дослідження, які вплинуть на те, що працює і може працювати для безпечного розвитку. На жаль, поточні дослідження, схоже, грають обмежену роль у розробці, пропозиції, оцінці та перевірці інструментів, методів та процесів, що використовуються на практиці для сталого розвитку. Зокрема дослідження рідко безпосередньо пов'язані з інструментами і методами, оскільки вони використовуються в тому контексті, в якому вони використовуються. Нам потрібні додаткові дослідження ефективності та результатів безпечних інструментів, методів та процесів розробки. Про це дослідження можна судити з його впливу те що, як розробка програмного забезпечення працює практично. Властивості дослідження впливають на можливість такого впливу.

Суворість науково-дослідних експериментів вимагає виконання низки технологічних вимог, включаючи формулювання перевіреної гіпотези, контроль експериментальних змінних, щоб переконатися, що експеримент дійсно перевіряє гіпотезу, та аналіз експериментальних даних та результатів для математичного підтвердження гіпотези (або спростування нульової гіпотези). гіпотеза). Хоча ці процеси можуть стати основою важливих фундаментальних досліджень у галузі безпечної розробки, вони часто уникають хаотичних реалій практичного застосування технологій саме тому, що хаотичні реальності ускладнюють планування експериментів.

Негативні результати досліджень, що не підтверджують, що безпечні методи розробки підвищують безпеку, хоч і важливі для галузі досліджень, навряд чи вплинуть на безпечну розробку на практиці. Перший урок розробника безпеки у великій технологічній компанії полягав у тому, що вказівка

розробникам не робити щось майже завжди було неефективним, якщо воно не поєднувалося з альтернативою, яку вони могли використовувати для досягнення мети застарілої практики. Крім того, виявлення експериментальної неефективності інструменту або методу для забезпечення безпеки не доводить, що вони неефективні поза контрольованим експериментом, у більш широкому, хаотичному та різноманітному контексті розробки програмного забезпечення.

Багато розробок програмного забезпечення засновані на існуючому програмному забезпеченні з використанням фреймворків, бібліотек та відкритого вихідного коду. Пропозиція артефакту, який використовується для ідентифікації та перевірки дослідницької ідеї, знижує бар'єри на шляху передачі цієї ідеї на розробку програмного забезпечення. Доступ до відкритого вихідного коду з умовами ліцензії, що багаторазово використовуються, може підвищити його зручність використання. Деякі дослідні стимули змінюються, щоб заохочувати відправлення артефактів у рамках процесу подачі та публікації наукових статей на конференціях з безпеки, таких як USENIX Security та ACSAC.

2.6 Логічна та фізична модель бази даних

Теоретичні основи інформаційних та комунікаційних технологій, засновані на найважливіших поняттях та законах інформатики (інформатика як наука, об'єкт та предмет інформатики; поняття інформації, її властивості та ознаки, до яких відносяться цінність, повнота, актуальність, компактність, достовірність та логічність; різні класифікації інформації , основні інформаційні процеси, види інформаційних ресурсів, види інформаційної діяльності, принципи обчислювальної техніки, алгоритми інформаційного моделювання, використання ІКТ)

Методи ІКТ включають моделювання, системний аналіз, проектування систем, методи передачі, збору, виробництва, накопичення, зберігання, обробки, передачі та захисту інформації.

Інструменти ІКТ поділяються на:

- апаратне забезпечення: персональний комп'ютер та його основні компоненти, локальні та глобальні мережі, сучасна периферія;

- Програмне забезпечення: система, додаток, інструмент.

Рисунок 2.12 Компоненти інформаційно-комунікаційних технологій

Висновки до розділу 2

В другому розділі розповідається про обрані технології та мову програмування для розробки Telegram бота. Описується бібліотеки та модулі які використовуються для розробки. Також описується архітектура програмного забезпечення та побудовані моделі ІКТ та діаграми.

Для реалізації платформи для розробки Telegram-бота для забезпечення інформаційної підтримки розпізнавання тексту було обрано середовище розробки PyCharm.

РОЗДІЛ 3

ЕТАПИ РОЗРОБКИ ТА СТВОРЕННЯ ПРОГРАМНОГО ДОДАТКУ

3.1 Опис розроблювального алгоритму

Алгоритм – це докладний набір інструкцій для виконання операції чи вирішення проблеми. Технічно комп'ютери використовують алгоритми отримання детальних інструкцій з виконання операції. З точки зору ефективності різні алгоритми можуть легко і швидко виконувати операції або

вирішувати задачі.

Для побудови алгоритму спочатку необхідно знати базову структуру побудови чат-бота, його компоненти та їхню взаємодію, можна використовувати загальну схему на рис. 3.1.

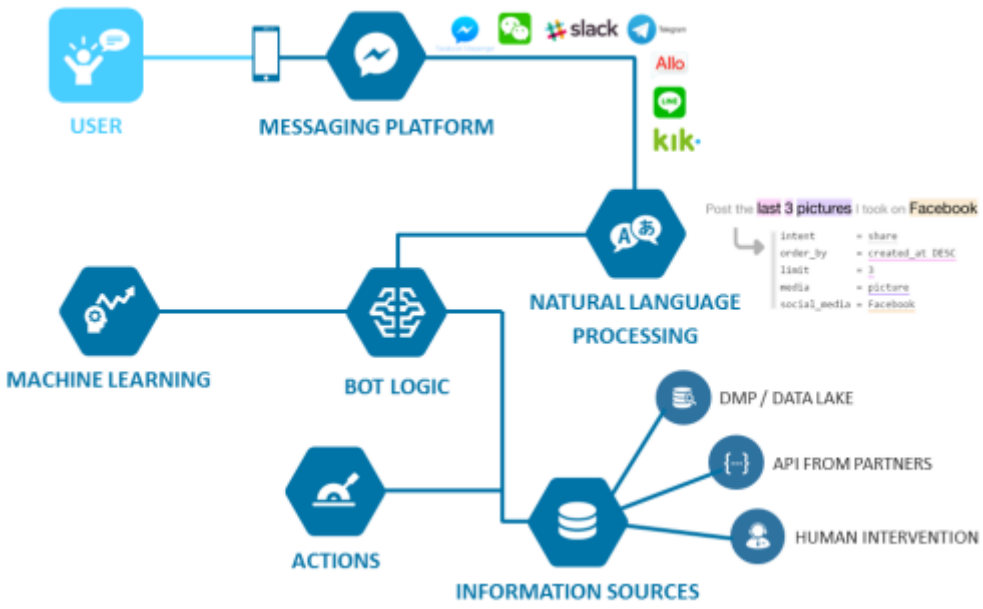


Рисунок 3.1 – Схема логіки чат-бота в месенджерах (Джерело: Hackernoon)

В алгоритмі розробки бота на першому кроці потрібно з'ясувати, яка платформа використовується користувачем, потім вивчити вибрану платформу, перевірити, чи готовий API, або ще потрібно програмувати самому (або

переконатися в наявності готового API обраною мовою програмування) більше ви можете побачити в розділі розробки власного продукту.

Наступним кроком буде розгляд питання — чи використовувати нейромережі: чи готові бібліотеки, чи потрібно будувати свою нейромережу, чи взагалі використовувати звичайні команди. Останнім кроком є розробка основного функціоналу робота. Результатом дослідження є необхідність поліпшення типового алгоритму з таргетом.

3.2 Розробка власного продукту

Відповідно до завдання дипломного проекту та відповідно до розглянутих підходів було прийнято рішення використовувати TeleBot, CherryPy та Requests мовою програмування Python для створення телеграм-бота, який надаватиме необхідну інформацію.

Основні аспекти, які планується реалізувати під час реалізації дипломного проекту:

а) отримання запиту користувача;

б) обробка запиту користувача;

в) у разі потреби уточнити деякі аспекти запиту;

г) відповідно до обробленого запиту надати необхідну інформацію користувачеві;

д) прогнозувати реакцію робота на непередбачені запити;

Опишемо кожен із аспектів докладніше. Ми впевнені, що користувач захоче поставити питання природною мовою, або вибрати з наших варіантів, потім це повідомлення буде відправлено в сервіс, через який ми можемо отримати контекст потрібного нам повідомлення, потім буде надіслана ця інформація. нашій програмі, в якій ми визначимо, що робити з цим повідомленням, негайно надати інформацію або потрібно поставити певні уточнюючі питання, процес уточнення може повторюватися доти, доки ми не отримаємо необхідну інформацію для видачі найбільш повної відповіді користувачеві, після процесу

отримання аналізу користувача та аналізу, ми надсилаємо йому необхідну інформацію.

Якщо користувач ввів непередбачений запит або запит, що не стосується нашої предметної області, є два варіанти розвитку подій. Якщо запит не стосується теми, з'явиться повідомлення з проханням ввести запитання лише для навчання. Якщо користувач ввів запит, пов'язаний з навчанням, але бот з якоїсь причини або його не зрозумів, або ми не змогли надати необхідну інформацію щодо цього запиту, ми відповімо заздалегідь підготовленими відповідями.

3.2.1 Розробка структури даних

MongoDB — це база даних, що орієнтована на документи, з відкритим вихідним кодом, що не вимагає опису схем таблиць. Основні функції та можливості:

а) підтримання відмовостійкості та масштабованості:

б) асинхронна реплікація, набір реплік та розподіл БД по вузлах;

в) JSON-подібна схема зберігання;

г) ефективне зберігання великих об'єктів, адміністративний інтерфейс, серверні функції, Map/Reduce тощо;

е) використання Javascript як мови для створення запитів;

f) запитувати профіль;

ж) широкий спектр атомарних операцій із даними (умовний пошук, складна вставка/оновлення тощо);

h) різні типи даних (включаючи підтримку масивів); підтримувати

індекси (B-дерево);

і) повнотекстовий пошук, у тому числі українською, п.

підтримка морфології;

j) журнал операцій, що змінюють дані у базі даних.

MongoDB пропонує модель даних, орієнтовану на документи, на відміну від реляційних баз даних, тому вона працює набагато швидше, має кращу масштабованість та простіше у використанні. Однак, враховуючи всі недоліки

традиційних баз даних та переваги MongoDB, важливо розуміти, що завдання різні та методи їх вирішення теж різні.

У певній ситуації MongoDB дійсно здатна підвищити продуктивність програми, наприклад ситуація, що потребує збереження складних даних. В іншій ситуації краще використовувати традиційні реляційні бази даних.

MongoDB може бути більшим, ніж просто одна база даних на одному фізичному сервері. Функціональність MongoDB дозволяє розміщувати декілька баз даних на декількох фізичних серверах, і ці бази даних можуть легко обмінюватися даними та підтримувати цілісність.

У процесі створення телеграм-бота було розроблено схему взаємодії користувача з базою даних, що забезпечує відповіді на його повідомлення (рис. 3.2).

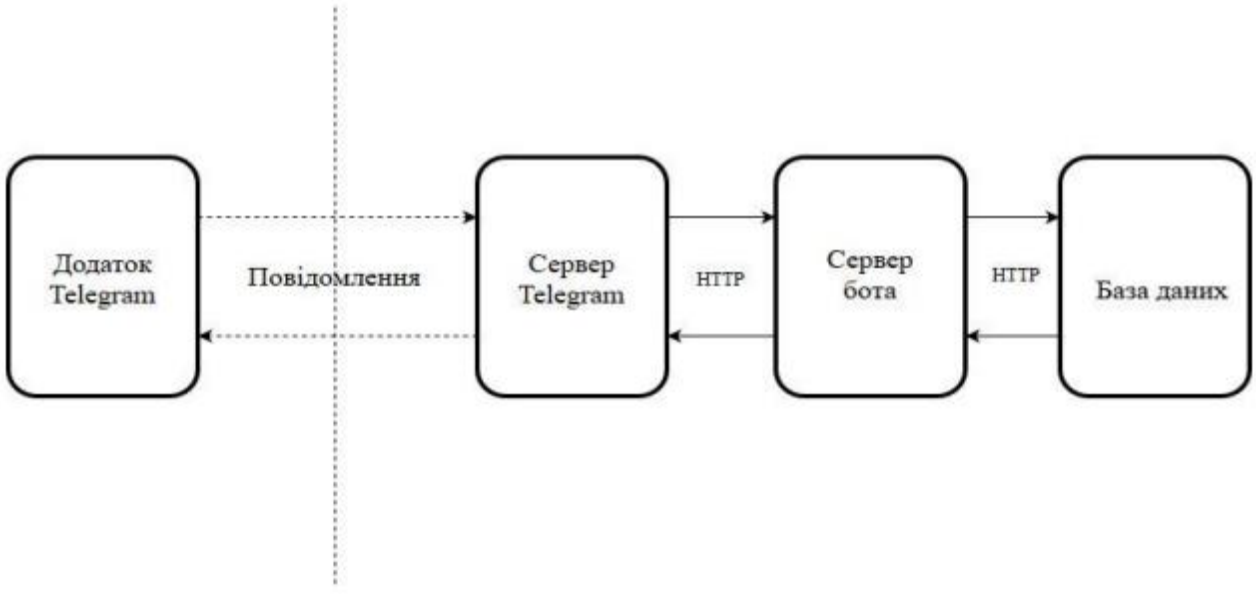


Рисунок 3.2 - Схема взаємодії користувача з базою даних

Ось деякі важливі особливості цієї бази даних:

а) формат даних у MongoDB. Одним із популярних стандартів обміну та зберігання даних є JSON (JavaScript Object Notation). JSON ефективно описує складні дані. Спосіб зберігання даних MongoDB в цьому плані аналогічний JSON, хоча формально JSON не використовується.

MongoDB використовує формат під назвою BSON або скорочення від бінарного JSON. BSON дозволяє працювати з даними швидше: пошук та обробка

виконуються швидше. Хоча слід зазначити, що BSON, на відміну від зберігання даних у форматі JSON, має невеликий недолік: загалом дані у форматі JSON займають менше місця, ніж у форматі BSON, з іншого боку, цей недолік з лишком окупається у швидкість;

б) кросплатформність. MongoDB написаний на C++, тому його легко переносити різні платформи. MongoDB можна розгорнути на платформах Windows, Linux, MacOS, Solaris. Ви також можете завантажити вихідний код та скомпілювати MongoDB самостійно, але рекомендується використовувати бібліотеки з офіційного сайту;

в) збірки. Якщо традиційному світі SQL є таблиці, то світі MongoDB є колекції. І якщо в реляційних базах даних таблиці зберігають однотипні жорстко структуровані об'єкти, колекція може містити безліч об'єктів з різною структурою і різним набором властивостей;

г) реплікація. Система зберігання MongoDB є набір реплік. Цей набір має первинний вузол і може бути набором вторинних вузлів. Усі вторинні вузли залишаються недоторканими та автоматично оновлюються разом із оновленням основного вузла. А якщо основний вузол з якихось причин виходить з ладу, то один із другорядних вузлів стає основним;

д) простота використання. Відсутність жорсткої схеми бази даних і, отже, необхідність як мінімум змінити концепцію зберігання даних для відтворення цієї схеми значно полегшує роботу з базами даних MongoDB та їхнє подальше масштабування. Це також заощаджує час розробників. Їм більше не потрібно думати про повторне створення бази даних та витрачати час на побудову складних запитів;

3.2.2 Написання програмного продукту

Bot API – це HTTP-інтерфейс для роботи з ботами в Telegram. Кожен бот — це спеціально створений обліковий запис для автоматичної обробки та надсилання повідомлень.

У документації Telegram Bot API передбачено два максимально протилежні способи отримання оновлень:

а) періодичні запити;

б) встановлення веб-хуків.

Оновлення, що надійшли, зберігаються до моменту їх обробки сервером, але не більше 24 годин. У відповідь ми отримуємо об'єкт Update, який серіалізується на JSON незалежно від того, як ви отримуєте оновлення.

Перший та найпростіший варіант – періодично опитувати сервери Telegram для отримання нової інформації. З'єднання відкривається на короткий час і всі оновлення надсилаються роботові відразу, все через з'єднання Long Polling. Цей метод простий, але дуже надійний.

Веб-хуки працюють трохи інакше. Якщо в чат надходить повідомлення, це говорить сам Telegram, це робота вебхука. Тепер не потрібно періодично опитувати сервери, тим самим усуваючи причину помилок пошукових систем. Однак за цю функцію потрібно платити необхідністю встановлення повноцінного веб-сервера на пристрій, на якому ви плануєте запускати пошукові роботи.

Аналогічно, для коректної роботи необхідно мати власний сертифікат-SSL (Secure Sockets Layer), тому що веб-хуки Telegram працюють тільки за HTTPS (рис. 3.3).

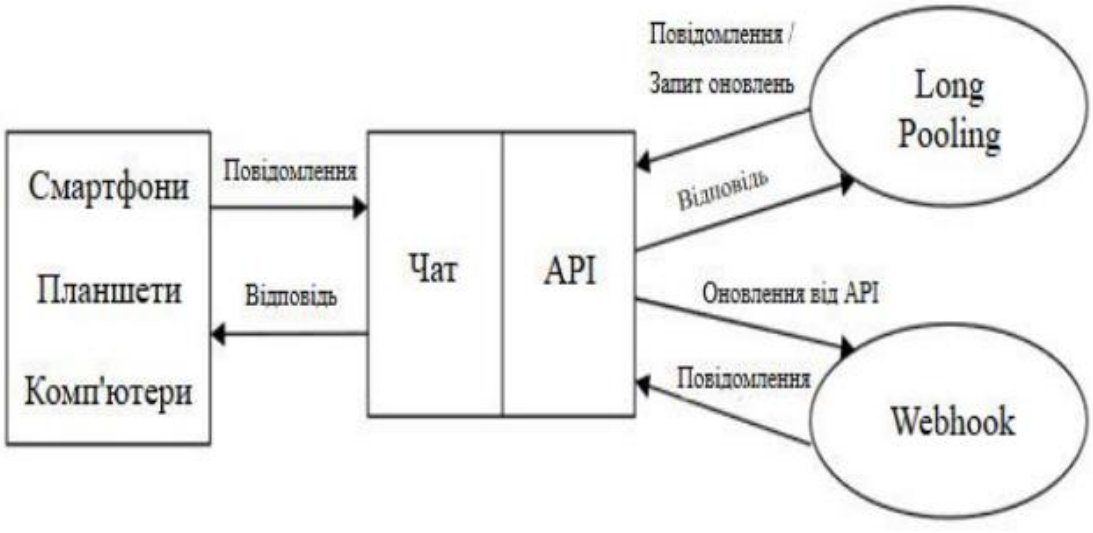


Рисунок 3.3 – Принцип роботи чат-бота на платформі Telegram

Для отримання токена потрібно написати спеціальний бот @BotFather.

Нижче наведено приклади доступних методів для API:

а) getUpdates - це метод, який використовується для отримання довготривалі оновлення технології опитування;

б) setWebhook - цей спосіб прив'язує до робота URL домену, на якому знаходиться запущений в даний момент бот;

в) sendMessage — метод, що надсилає текстове повідомлення в клієнтській частині Telegram;

г) sendLocation - метод, який надсилає повідомлення з координатами розташування пристрою;

д) getFile – метод, який повертає вкладення з їхнього імені.

Запити POST та GET дозволені.

Існує 4 способи передачі параметрів у Bot API:

а) URL-запит;

б) додаток /x-www-form-urlencoded;

в) додаток/json;

г) multipart/form-data (підходить для завантаження файлів).

Для роботи з Telegram Bot API ми вивчили документацію, в якій описані всі методи та параметри, з'ясувалося, що всі відповіді надходять у JSON-форматі.

3.3 Опис функціоналу

Для відображення підходів до розробки чат-бота було запропоновано розробити бота, для розробки якого необхідні певні інструменти та інструменти, в даній роботі я використовуватиму:

- JSON для обміну інформацією (оскільки бот повертає дані саме у такому вигляді);

- Node.js для серверної частини;

- ДОДАТОК API ChatPot.

JSON – це стандартний текстовий формат для представлення структурованих даних на основі синтаксису JavaScript. Зазвичай він використовується для передачі даних у веб-програмах (наприклад, для надсилання деяких даних із сервера клієнту, щоб ви могли відобразити їх на веб-сторінці або навпаки). Ви будете мати справу досить часто, тому в цій статті ми даємо вам все необхідне для роботи з JSON за допомогою JavaScript, включаючи синтаксичний аналіз JSON, щоб ви могли отримати доступ до даних всередині нього і створити JSON.

Node.js - це платформа на основі JavaScript для запуску JavaScript, що дозволяє легко створювати швидкі та масштабовані мережеві програми. Node.js використовує керовану подіями неблокуючу модель введення-виводу, яка робить її простою та ефективною, що ідеально підходить для використання в реальному часі програм реального часу, що працюють на розподілених пристроях.

APP ChatBot – це система розуміння мови, яка дозволяє створювати розумні чати для будь-якого сервісу. Ви можете легко інтегрувати своїх роботів з вашими улюбленими програмами обміну повідомленнями та дозволити їм постійно обслуговувати своїх клієнтів.

Кожен запит API вимагає автентифікації для визначення ліцензії, яка відповідає за надсилання запиту. Аутентифікація забезпечується токен доступу (рис. 3.4).

```
var http = require("https");

var options = {
    "method": "GET",
    "hostname": "api.chatbot.com",
    "port": null,
    "path": "/stories",
    "headers": {
        "authorization": "Bearer ${DEVELOPER_ACCESS_TOKEN}"
    }
};

var req = http.request(options, function (res) {
    var chunks = [];

    res.on("data", function (chunk) {
        chunks.push(chunk);
    });

    res.on("end", function () {
        var body = Buffer.concat(chunks);
        console.log(body.toString());
    });
});

req.end();
```

Рисунок 3.4 – Приклад запиту аутентифікації (код Node js)

Висновки до розділу 3

В третьому розділі описується розроблений алгоритм. Розповідається про розробку власного продукту. Описується функціонал продукту.

Відповідно до завдання дипломного проекту та відповідно до розглянутих підходів було прийнято рішення використовувати TeleBot, CherryPy та Requests мовою програмування Python для створення телеграм-бота, який надаватиме необхідну інформацію.

Основні аспекти, які планується реалізувати під час реалізації дипломного проекту:

а) отримання запиту користувача;

б) обробка запиту користувача;

в) у разі потреби уточнити деякі аспекти запиту;

г) відповідно до обробленого запиту надати необхідну інформацію користувачеві;

д) прогнозувати реакцію робота на непередбачені запити;

РОЗДІЛ 4

РОЗРОБКА ПЛАТФОРМИ ДЛЯ СТВОРЕННЯ TELEGRAM-БОТА ДЛЯ ЗАБЕЗПЕЧЕННЯ РОЗПІЗНАВАННЯ ТЕКСТУ

4.1 Робота боту з використанням Telegram Bot API

Першим кроком розробки платформи є вхід у спеціальному чат-боті, який має назву @kpi_sgera_bot. Перейшовши за посиланням https://t.me/kpi_sgera_bot, перед нами відкриється розроблений нами телеграм бот (рис. 4.1)

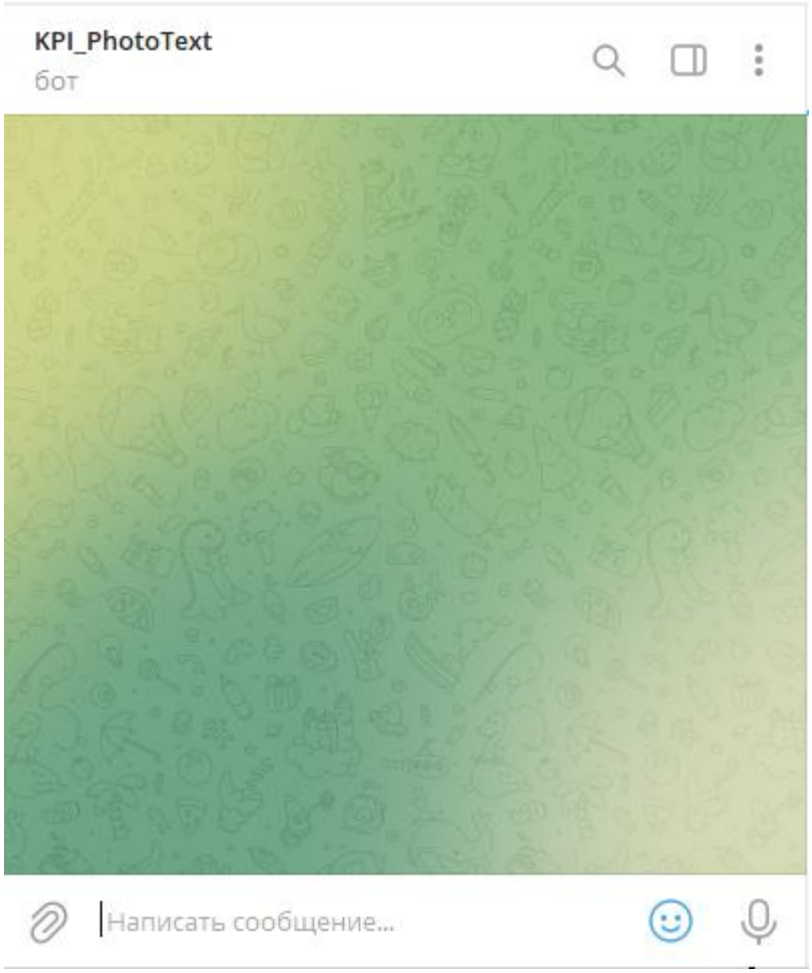


Рисунок 4.1 – Запуск Telegram-бота

Вхід починається з введення користувачем команди /start (рис. 4.2).

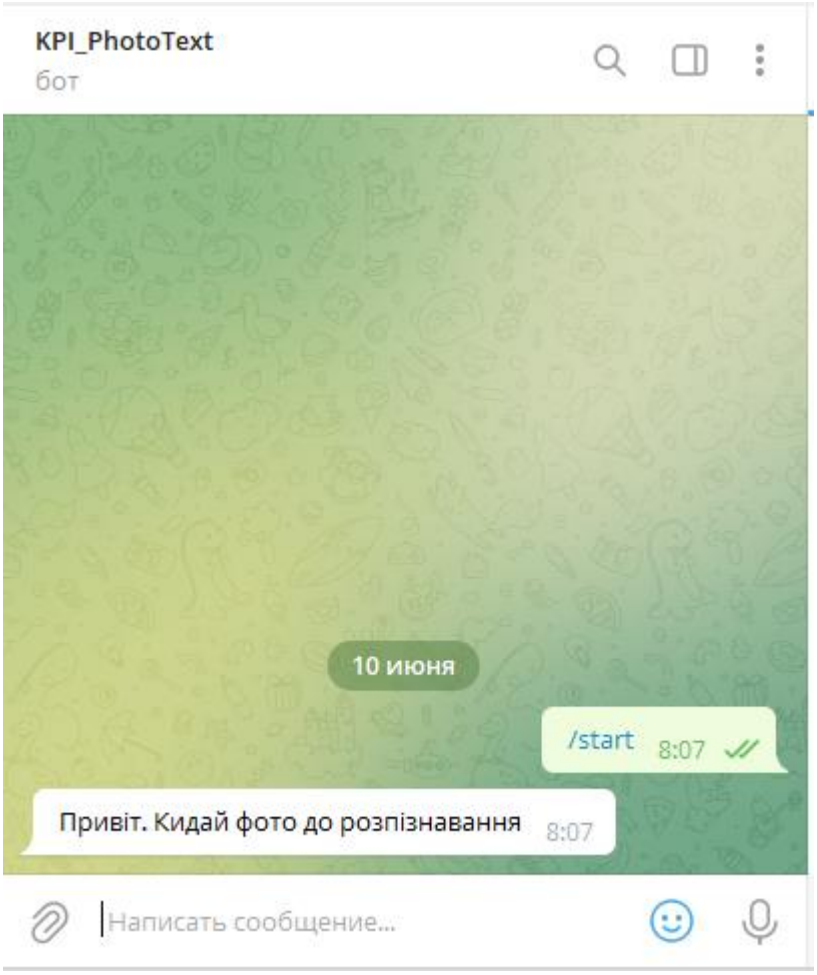


Рисунок 4.2 – запуск бота

Завантаживши фото для розпізнавання тексту, отримаємо наступну картину (рис. 4.3):

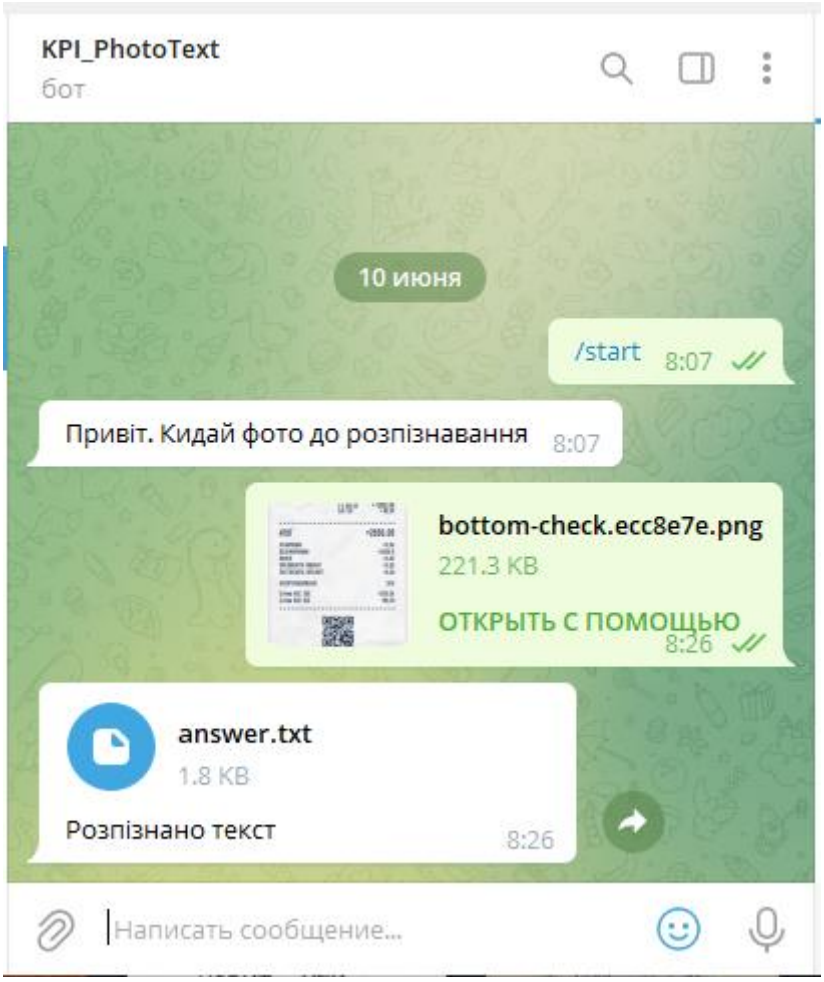


Рисунок 4.3 –розпізнавання тексту

Завантажений файл виглядає наступним чином (рис. 4.4)

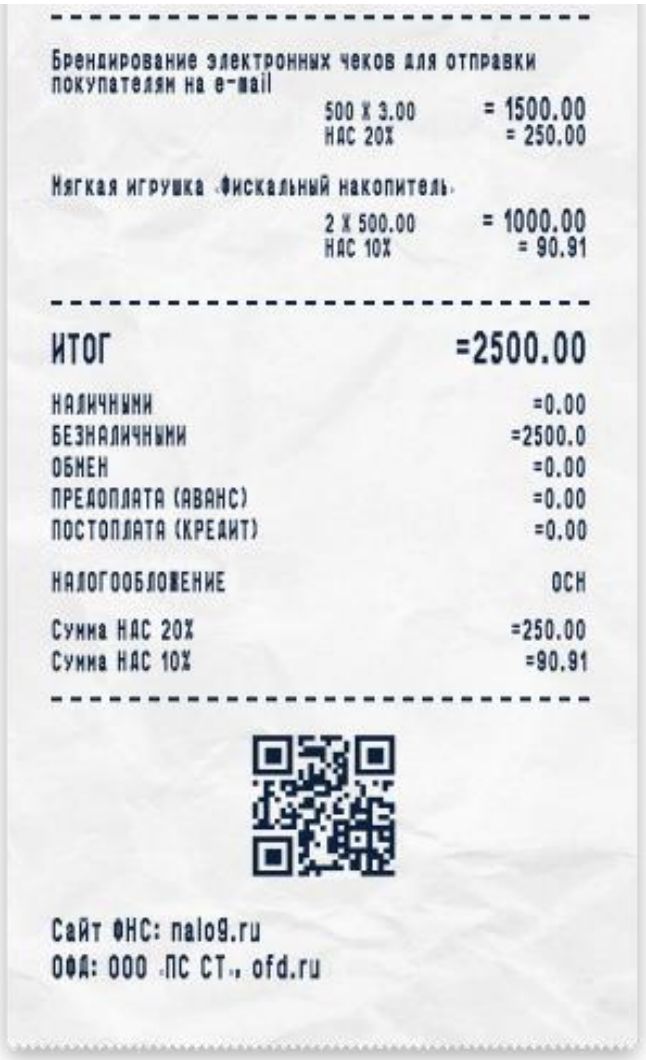


Рисунок 4.4 – завантажений файл

А розпізнаний файл виглядає наступним чином (рис. 4.5)

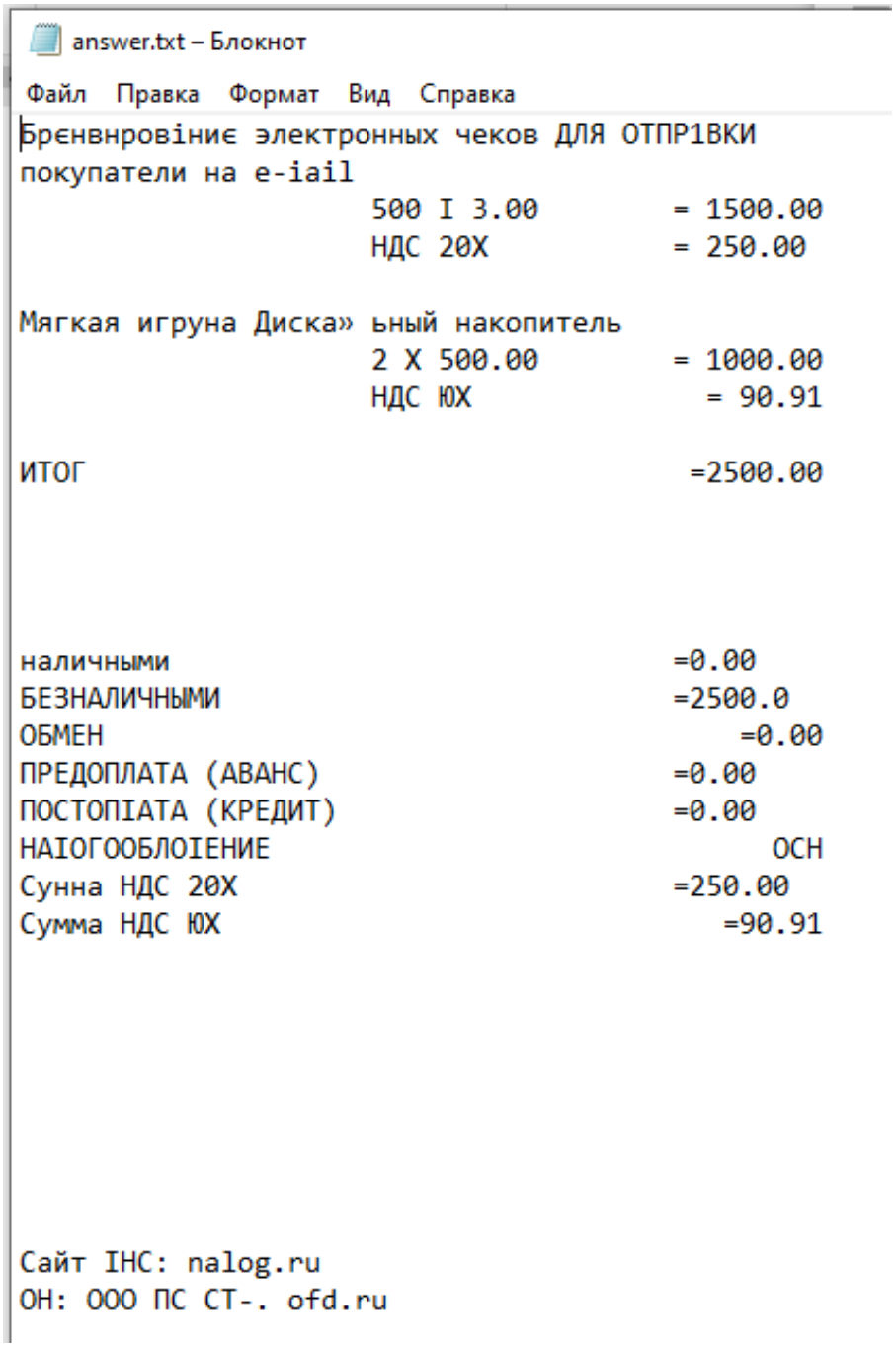


Рисунок 4.5 – розпізнаний файл

Як бачимо, бот впорався з розпізнавання тексту дуже добре.

Для установки додаткових параметрів, таких як іконка Telegram-бота, вітальне повідомлення, опис Telegram-бота, й так само видалення наявних Telegram-ботів, існують такі команди [29], що проілюстровані далі у табл. 4.1.

Таблиця 4.1 – Доступні команди для управління ботом

Назва команди	Опис
/setname	Змінює існуючу назву
/setdescription	Присваює текст, що буде відображатись при першому запуску бота
/setabouttext	Присваює текст в полі “Про чат-бот”
/setcommands	Дозволяє створити список доступних команд
/deletebot	Видаляє обраного чат-бота

Крім команд для зміни основних параметрів Telegram-бота існує ряд команд [29], які дозволяють виводити незмінні параметри (наприклад, токен), а також присвоювати значення, які представлені в таблиці 4.2.

Таблиця 4.2 – Доступні команди для додаткового налаштування Telegram-боту

Назва команди	Опис
/token	Повертає отриманий раніше токен у обраного бота
/revoke	Анулює отриманий токен доступу до бота
/setinline	Вмикає або вимикає можливість викликати бота з інших чатів
/setinlinegeo	Вмикає або вимикає можливість передавати розташування бота з іншого чату
/setinlinefeedback	Дозволяє отримувати інформацію про кількість обраних користувачами команд
/setjoingroup	Визначає чи може бути доданий бот до групових діалогів
/setprivacy	Включає режим конфіденційності. В цьому режимі бот отримує, обробляє і відсилає назад інформацію окремо для кожного користувача в чаті

Після всіх необхідних налаштувань платформи з боку платформи telegram, яка є базисною для розробки Telegram-бота для забезпечення інформаційної підтримки розпізнавання тексту, можна завершити процес створення таких ботів.

4.2 Тестування програми

Тестування телеграм бота проводилося вручну. Після кожного етапу розробки проводилося тестування продуктивності на заздалегідь підготовлених тест-кейсах, які складалися для визначення цілей розробки та використовуваних інструментів.

Робота з Telegram-ботом:

а) швидкість реагування бота на повідомлення про початок роботи;

б) правильна обробка повідомлень різного типу від користувача.

в) правильне відображення діалогу;

г) обробка натискання клавіші відправити повідомлення;

д) швидкість відповіді після того, як користувач написав повідомлення;

е) швидкість роботи бази даних;

є) оцінювання правильності відповідей.

Бот був протестований на мобільному пристрої Apple Iphone X з наступними характеристиками.

а) екран: 5,5", IPS LCD, 1920x1080, мультитач;

б) процесор: Apple A10 Fusion, 4x1;

в) операційна система: iOS 14.1.4;

г) ОЗУ: 3 ГБ;

д) вбудована пам'ять: 128 ГБ;

е) навігація: GPS;

g) Telegram: Telegram v 8.

Нижче наведено графік швидкості відповіді бота в залежності від об’єму завантаженого файлу.

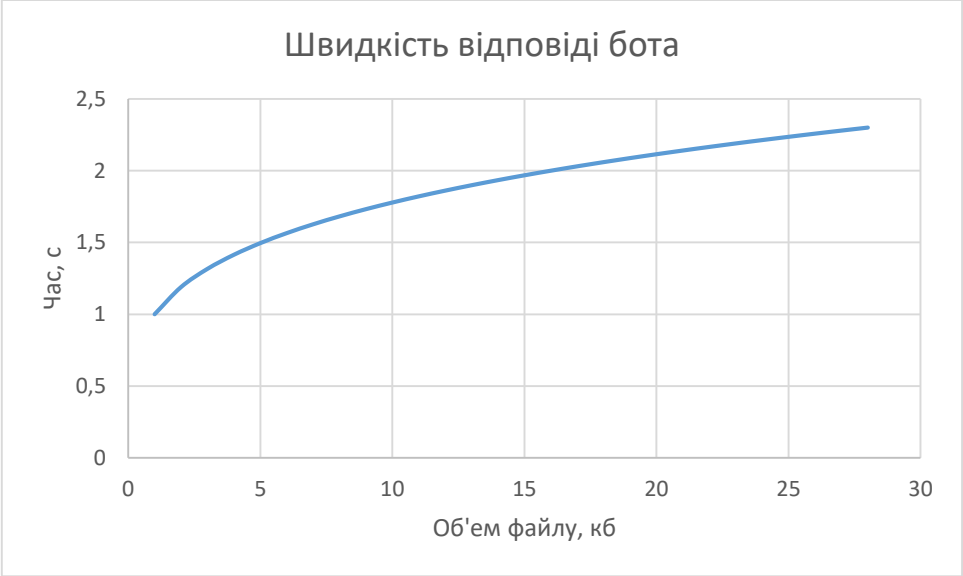


Рисунок 4.6 – графік відповіді бота

Також наведемо приклади розпізнавання ботом різних чеків

Рисунок 4.7 – вид чеку

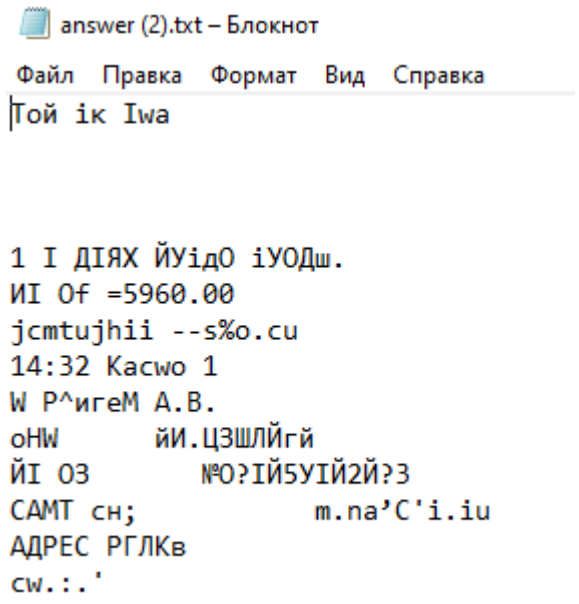
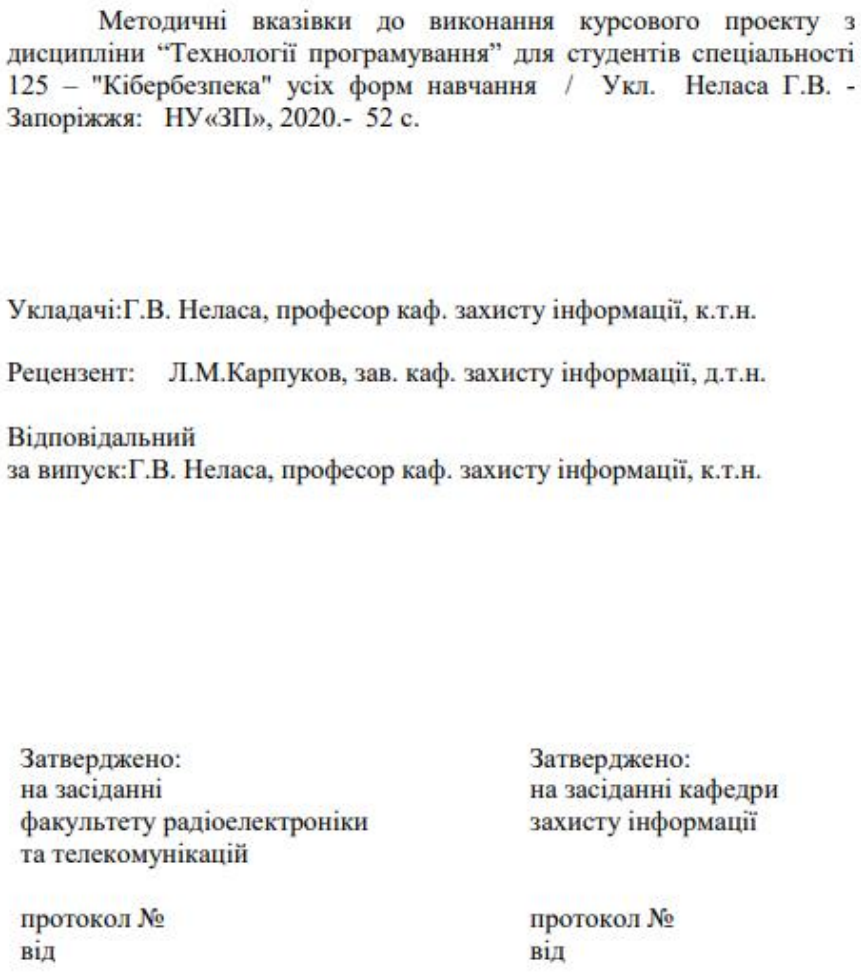


Рисунок 4.8 - розпізнаний текст

Також тестування бота відбувалося і на методичках



Методичні вказівки до виконання курсового проекту з дисципліни "Технології програмування" для студентів спеціальності 125 – "Кібербезпека" усіх форм навчання / Укл. Неласа Г.В. - Запоріжжя: НУ«ЗП», 2020.- 52 с.

Укладачі:Г.В. Неласа, професор каф. захисту інформації, к.т.н.

Рецензент: Л.М.Карпуков, зав. каф. захисту інформації, д.т.н.

Відповідальний
за випуск:Г.В. Неласа, професор каф. захисту інформації, к.т.н.

Затверджено:
на засіданні
факультету радіоелектроніки
та телекомунікацій

протокол №
від

Затверджено:
на засіданні кафедри
захисту інформації

протокол №
від

Рисунок 4.9 – Текст методички

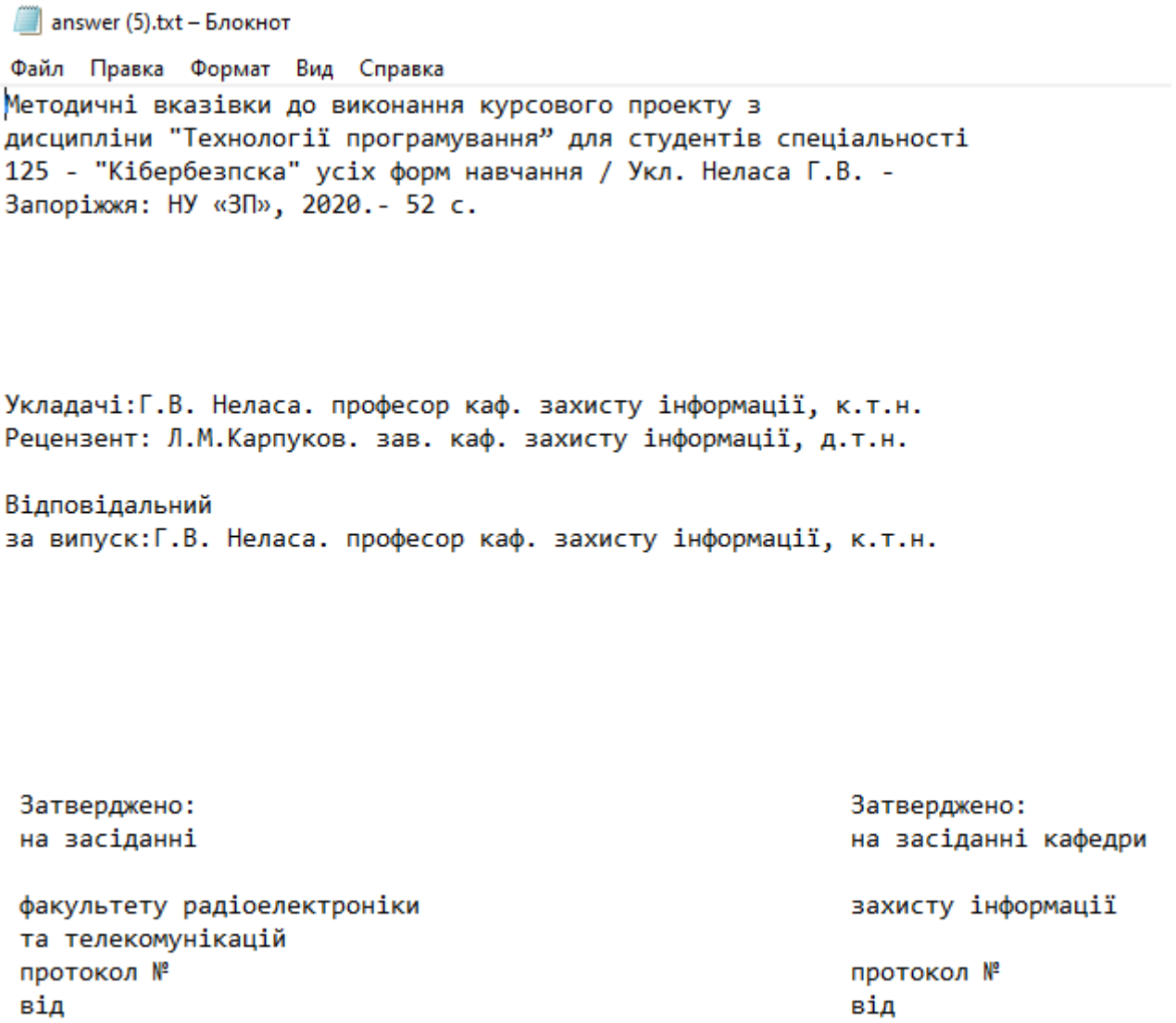


Рисунок 4.10 - Текст розпізнаний

Також бот був протестований і на різних книга (нижче наведено декілька прикладів)

Книга посвящена проблеме выживания человеческой популяции в условиях развивающегося экологического кризиса. Показано, что восстановление баланса между изымаемым у природы объёмом ресурсов и, соответственно, объёмом отходов и способностью природы восстанавливать равновесие является центральной проблемой, от решения которой зависит возможность устойчивого развития и даже существование человечества. Именно такое равновесие обеспечивает биоте возможность путём обратных связей сохранять на Земле условия, необходимые для существования живых существ. Доказывается, что это может быть достигнуто либо уменьшением объёма потребления ресурсов планеты на порядок, либо аналогичным сокращением численности человечества. Все другие обсуждаемые и принимаемые меры по сохранению окружающей среды, такие как отказ от углеводородной энергетики, рециклинг и т.п., проблему в целом решить не могут. Утверждается, что путь радикальной трансформации общества потребления является весьма проблематичным, но не безнадёжным, однако необходима смена парадигмы социально-экономического развития и отказ от стратегии непрерывного роста.

Книга предназначена для всех, кто интересуется судьбой человечества, и может быть использована в качестве учебного пособия в различных курсах по экологии.

Рисунок 4.11 – текст в книзі

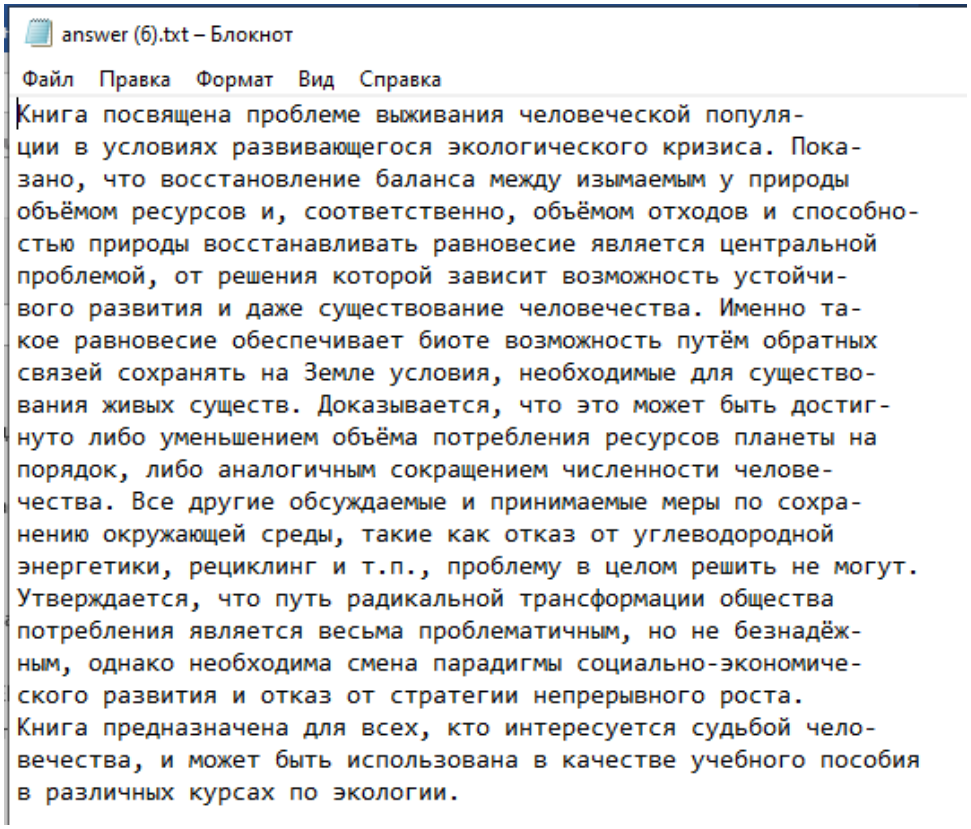


Рисунок 4.12 – Розпізнавання тексту книги

Також були і не дуже вдалі розпізнавання чеків

Berghotel
Grosse Scheidegg
3818 Grindelwald
Familie R.Müller

Rech.Nr. 4572 30.07.2007/13:29:17
Bar Tisch 7/01

2xLatte Macchiato	à	4.50	CHF	9.00
1xGlobi	à	5.00	CHF	5.00
1xSchweinschnitzel	à	22.00	CHF	22.00
1xChässpätzli	à	18.50	CHF	18.50

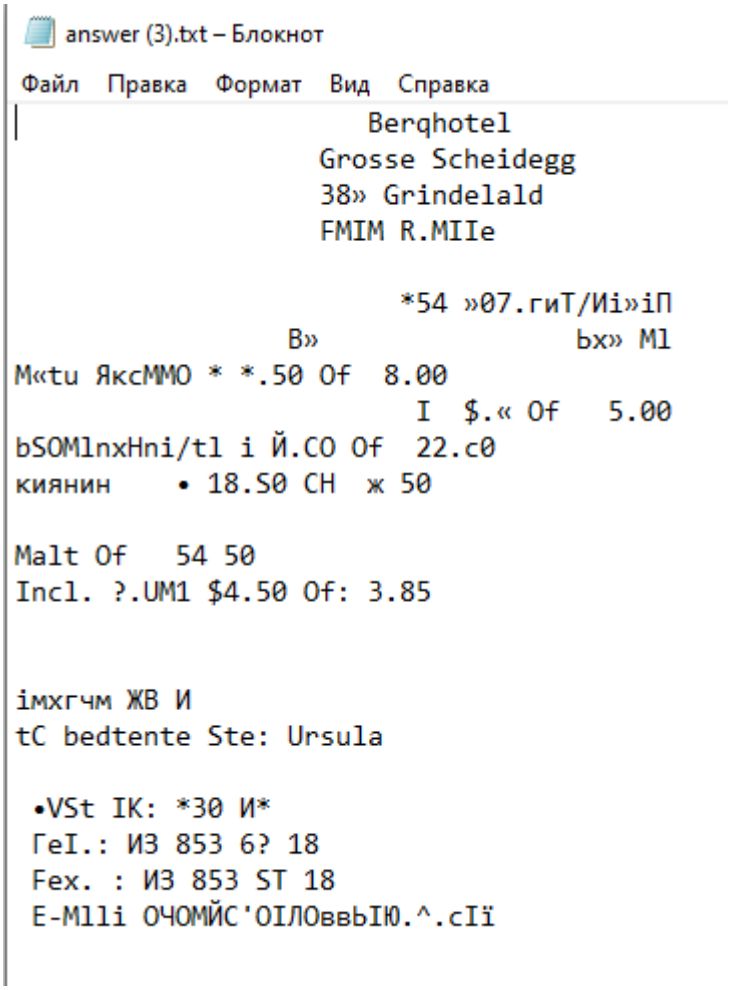
Total : CHF 54.50

Incl. 7.6% MwSt 54.50 CHF: 3.85

Entspricht in Euro 36.33 EUR
Es bediente Sie: Ursula

MwSt Nr.: 430 234
Tel.: 033 853 67 16
Fax.: 033 853 67 19
E-mail: grossescheidegg@bluewin.ch

Рисунок 4.13 – вид чеку



answer (3).txt – Блокнот

Файл Правка Формат Вид Справка

Berghotel
Grosse Scheidegg
38» Grindelald
FMIM R.MIIe

*54 »07.гиТ/Ии»іП
B» bx» Ml
M«tu ЯксММО * *.50 Of 8.00
I $.« Of 5.00
bSOMlnxHni/tl i Й.CO Of 22.c0
киянин • 18.S0 CH ж 50

Malt Of 54 50
Incl. ?.UM1 $4.50 Of: 3.85

імхгчм ЖВ И
tC bedtente Ste: Ursula

•VSt IK: *30 И*
ГеІ.: И3 853 6? 18
Fex. : И3 853 ST 18
E-Mlli ОЧОМЙС'ОІЛОввЬІЮ.^.сІї

Рисунок 4.14- розпізнаний текст

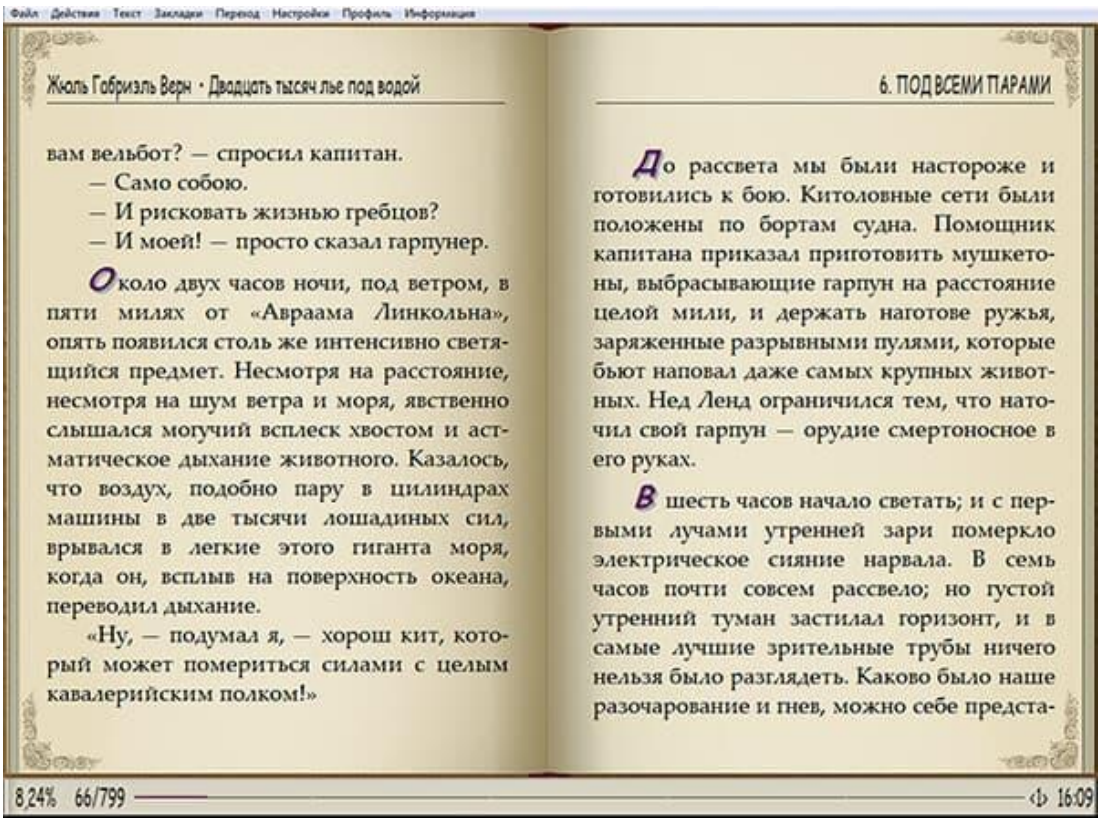


Рисунок 4.15 – вид сторінки книги

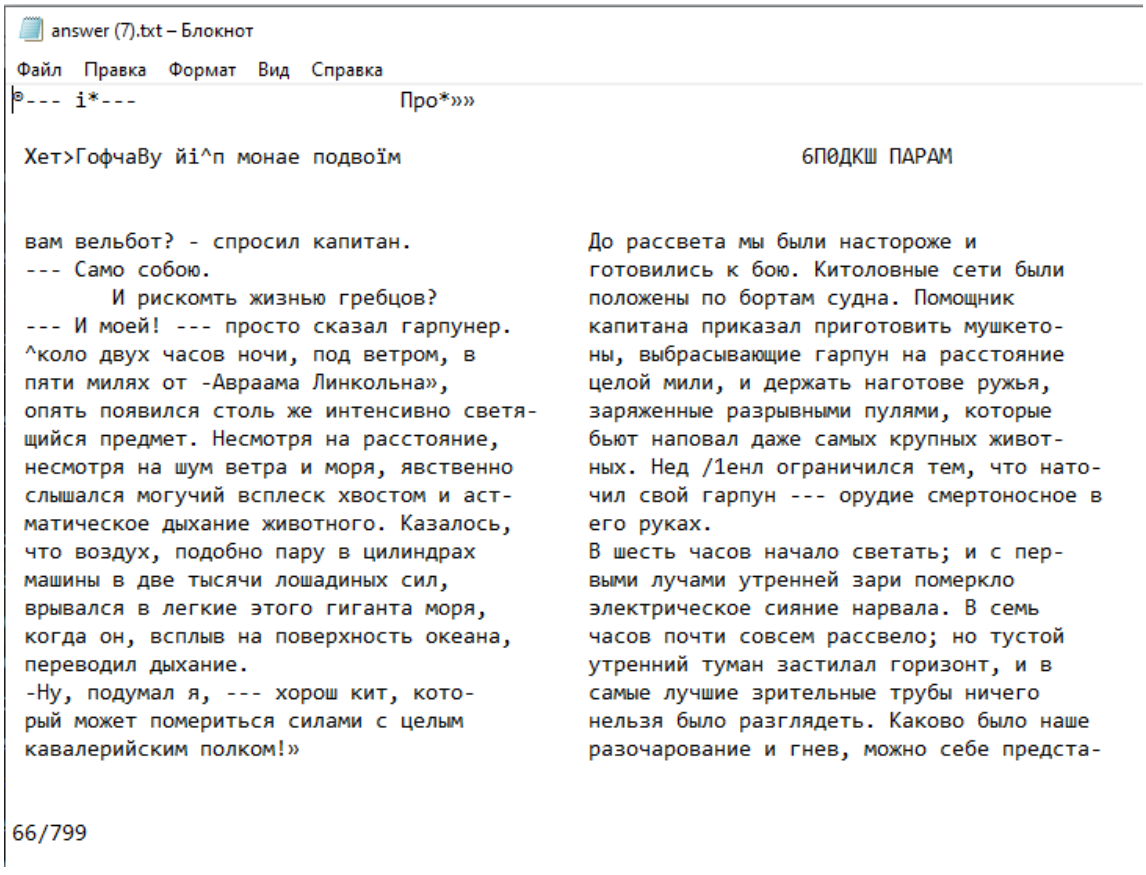


Рисунок 4.16 – розпізнаний текст

Як показало тестування, на більшості видів зображень бот зміг розпізнати текст з точністю більше 95%. В інших випадках, коли розпізнавання вийшло невдало, скоріш а все, це було пов’язане із шрифтом тексту, або нечіткім (нерозбірливим) виглядом тексту. Також на розпізнавання тексту могло вплинути і такий параметр як освітлення. В подальшому потрібно буде поліпшити датасет, та більш детально проаналізувати можливі фактори для покращення розпізнавання тексту.

Висновки до розділу 4

В четвертому розділі розповідається про розробку платформи. Також розповідається про роботу бота з використанням Telegram Bot API та про реалізацію клієнтської частини.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розглянуто історію створення та розвитку месенджерів як модерного засобу обміну інформацією та спілкування. Також детально розглянуто приклади різноманітних ботів, що спрощують виконання рутинних задач та дозволяють взаємодіяти з користувачами в автоматичному режимі. Детально описано принципи розробки подібного програмного забезпечення та висунуто основні задачі, які потрібно врахувати при розробці платформи для розробки Telegram-бота для забезпечення розпізнавання тексту з картинки. Слід зазначити, що акцент зроблено на підборі оптимальних інструментів, що є необхідними для коректної роботи зазначеної платформи.

Для створення платформи для розробки Telegram-бота для забезпечення інформаційної підтримки розпізнавання тексту було обрано найбільш стабільні засоби та інструменти розробника, а саме: мову програмування Python разом з набором корисних бібліотек, що використовувались при реалізації програмного коду в багатофункціональному середовищі розробки PyCharm, використано бібліотеку Telebot, яка реалізує безпосередньо Telegram Bot API, а також базу даних на основі SQLAlchemy.

Проаналізовано вже існуючі аналоги до розроблюваного програмного продукту та наведено приклади їх використання, основні переваги та недоліки. В результаті, було виявлено необхідність створення графічного інтерфейсу для взаємодії з користувачами, а не обмежуватись стандартним набором команд для керування ботом у Telegram. Крім того, було систематизовано та формалізовано алгоритм вибору інструментів розробки.

Як результат кваліфікаційної роботи, на базі вищеописаних принципів та із використанням зазначених технологій було створено платформу для розробки Telegram-бот для забезпечення інформаційної підтримки розпізнавання тексту. Даний програмний продукт є основним доказом концепції автоматизації розпізнавання тексту за допомогою ботів для месенджерів (зокрема, у даному випадку для Telegram) та ілюструє можливу швидкість та гнучкість розробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Комп'ютерна система PLATO [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/PLATO.
2. Talkomatic – Just think of it [Електронний ресурс] – Режим доступу: https://just.thinkofit.com/talkomatic/.
3. AIM messenger changed the way we communicate [Електронний ресурс] – Режим доступу: https://www.vox.com/culture/2017/12/15/16780418/aim-aol-instant-messenger-shutdowncultural-impact.
4. History of Friendster [Електронний ресурс] – Режим доступу: https://socialnetworking.lovetoknow.com/history-friendster.
5. Facebook [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Facebook.
6. Хто та як використовує месенджери в Україні? [Електронний ресурс] – Режим доступу : https://minfin.com.ua/2018/04/14/33167841/.
7. Рейтинг мобільних додатків за березень 2020 [Електронний ресурс] – Режим доступу : https://tns-ua.com/news/reyting-mobilnih-dodatkiv-za-berezen-2020.
8. Messina, C. 2016 will be the year of conversational commerce [Електронний ресурс] / C. Messina // Medium. – 2016. – Режим доступу: https://medium.com/chris-messina/2016-will-be-the-year-of-conversationalcommerce-1586e85e3991.
9. MTPoto Mobile Protocol [Електронний ресурс] – Режим доступу: https://core.telegram.org/mtproto. Дата звернення – травень 2020 р.
10. Козлов А. А. Телеграм-бот як простий та зручний спосіб отримання інформації [Електронний ресурс] / А. А. Козлов, А. В. Батищев // Територія науки. – 2017. – №5. – с. 55-64.
11. Ebot one – редактор ботів для Telegram [Електронний ресурс] – Режим доступу: https://ebot.one/.
12. Manybot – платформа для створення ботів [Електронний ресурс] – Режим доступу: https://manybot.io/.

13. How to make a responsive telegram bot [Електронний ресурс] – Режим доступу: https://www.sohamkamani.com/blog/2016/09/21/making-atelegram-bot/.
14. Васильєв О. Python на прикладах. Практичний курс по програмуванню / Васильєв Олексій., 2016. – С. 432.
15. Введення до модулю os [Електронний ресурс] – Режим доступу : https://docs.python.org/3/library/os.html.
16. Використання модулю subprocess [Електронний ресурс] – Режим доступу: https://docs.python.org/3/library/subprocess.html?highlight=subprocess#module-subprocess.
17. Використання пакету logging [Електронний ресурс] – Режим доступу: https://docs.python.org/3/library/logging.html?highlight=logging#module-logging..
18. The Python SQL Toolkit and Object Relational Mapper [Електронний ресурс] – Режим доступу : https://www.sqlalchemy.org/.
19. Приклади ботів для Telegram [Електронний ресурс] – Режим доступу: https://github.com/TelegramBots/telegram.bot.examples.

ДОДАТКИ

Додаток А (main.py)

```
import logging
import time
import traceback

from aiogram import executor

import config
# noinspection PyUnresolvedReferences
import handlers
from misc import dp, bot



async def on_startup(args):
    try:
        await bot.send_message(config.DEVELOPER_ID, 'Бот увімкнено')
    except Exception:
        pass



if __name__ == "__main__":
    while True:
        try:
            executor.start_polling(dp, on_startup=on_startup)
        except Exception:
            logging.exception('Catch exception in MAIN:\n' +
traceback.format_exc())
            time.sleep(1)
```

Додаток Б(misc.py)

```
from ABBYY import CloudOCR


from misc import bot



async def get_text_from_photo(photo):
    download_file = await bot.download_file_by_id(photo.file_id)
    ocr_engine = CloudOCR(application_id='aa464c13-2dba-4947-a7bc-
10ecaa4f1332', password='RuJYEH/Kx6zD/RKDDazz4mhC')
    file = {'photo.jpg': download_file}
    result = ocr_engine.process_and_download(file, exportFormat='txt',
language="Russian,Ukrainian,English")
    answer = result['txt'].getvalue().decode('utf-8')
    print(answer)
    true_answer = ''
    for a in answer:
        if ord(a) < 10000:
            true_answer += a
    with open('answer.txt', 'w') as f:
        f.write(true_answer)
            return 'answer.txt'
```

Додаток В(photo_utils.py)

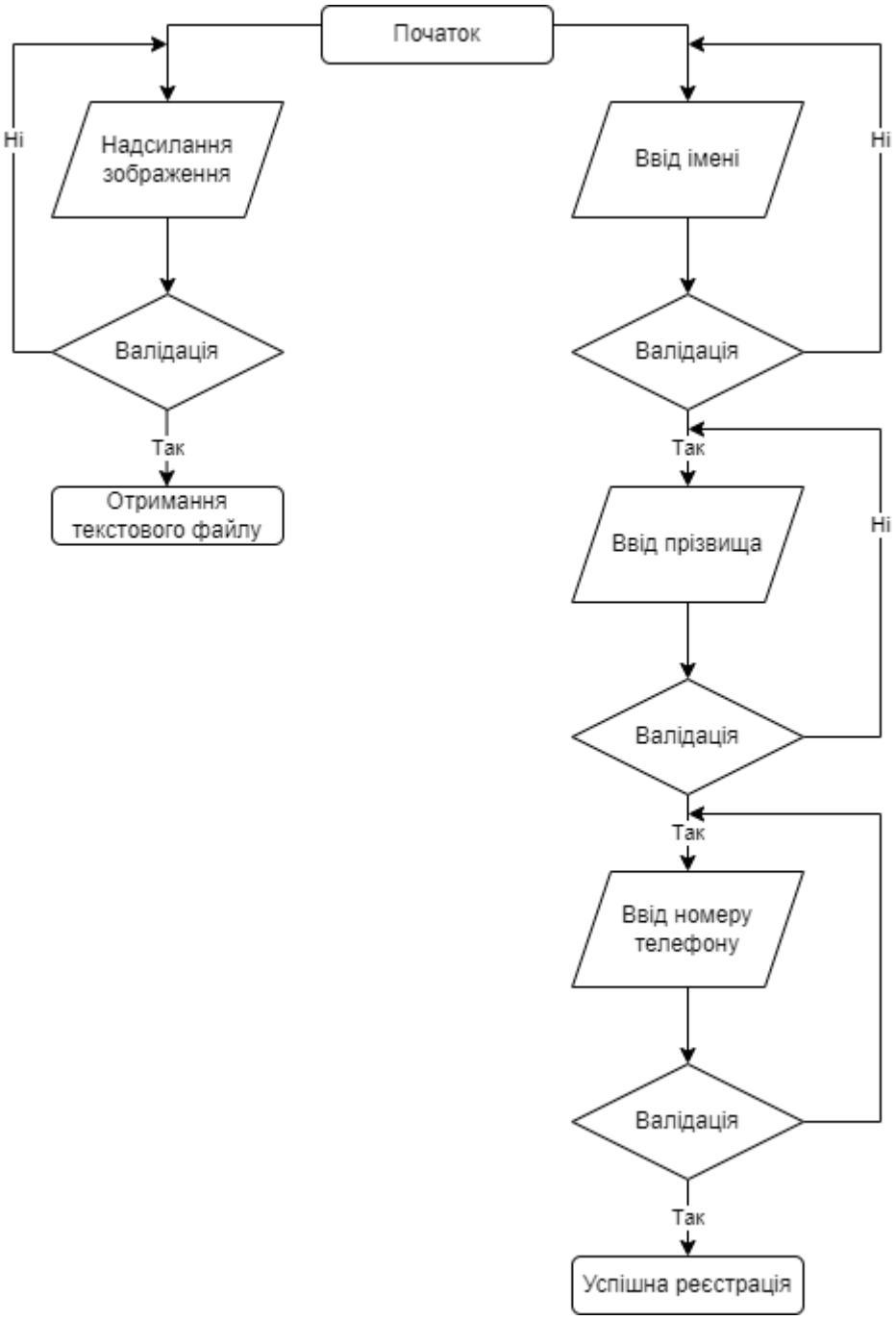
```
import asyncio
import logging
import typing


from aiogram import Dispatcher as _Dispatcher
from aiogram.bot import Bot
from aiogram.dispatcher.filters import FiltersFactory
```

```
from aiogram.dispatcher.middlewares import MiddlewareManager
from aiogram.dispatcher.storage import BaseStorage, DisabledStorage

from .handler import Handler

log = logging.getLogger(__name__)

DEFAULT_RATE_LIMIT = .1



def _ensure_loop(x: "asyncio.AbstractEventLoop"):
    assert isinstance(
        x, asyncio.AbstractEventLoop
    ), f"Loop must be the implementation of {asyncio.AbstractEventLoop!r}, " \
       f"not {type(x)!r}"



class Dispatcher(_Dispatcher):
    def __init__(self, bot, loop=None, storage: typing.Optional[BaseStorage] = None,
                 run_tasks_by_default: bool = False,
                 throttling_rate_limit=DEFAULT_RATE_LIMIT, no_throttle_error=False,
                 filters_factory=None):


        if not isinstance(bot, Bot):
            raise TypeError(f"Argument 'bot' must be an instance of Bot, not
'{type(bot).__name__}'")


        if storage is None:
            storage = DisabledStorage()
        if filters_factory is None:
```

```
            filters_factory = FiltersFactory(self)


        self.bot: Bot = bot
        if loop is not None:
            _ensure_loop(loop)
        self._main_loop = loop
        self.storage = storage
        self.run_tasks_by_default = run_tasks_by_default

        self.throttling_rate_limit = throttling_rate_limit
        self.no_throttle_error = no_throttle_error

        self.filters_factory: FiltersFactory = filters_factory
        self.updates_handler = Handler(self, middleware_key='update')
        self.message_handlers = Handler(self, middleware_key='message')
        self.edited_message_handlers = Handler(self,
middleware_key='edited_message')
        self.channel_post_handlers = Handler(self, middleware_key='channel_post')
        self.edited_channel_post_handlers = Handler(self,
middleware_key='edited_channel_post')
        self.inline_query_handlers = Handler(self, middleware_key='inline_query')
        self.chosen_inline_result_handlers = Handler(self,
middleware_key='chosen_inline_result')
        self.callback_query_handlers = Handler(self, middleware_key='callback_query')
        self.shipping_query_handlers = Handler(self,
middleware_key='shipping_query')
        self.pre_checkout_query_handlers = Handler(self,
middleware_key='pre_checkout_query')
        self.poll_handlers = Handler(self, middleware_key='poll')
        self.poll_answer_handlers = Handler(self, middleware_key='poll_answer')
```

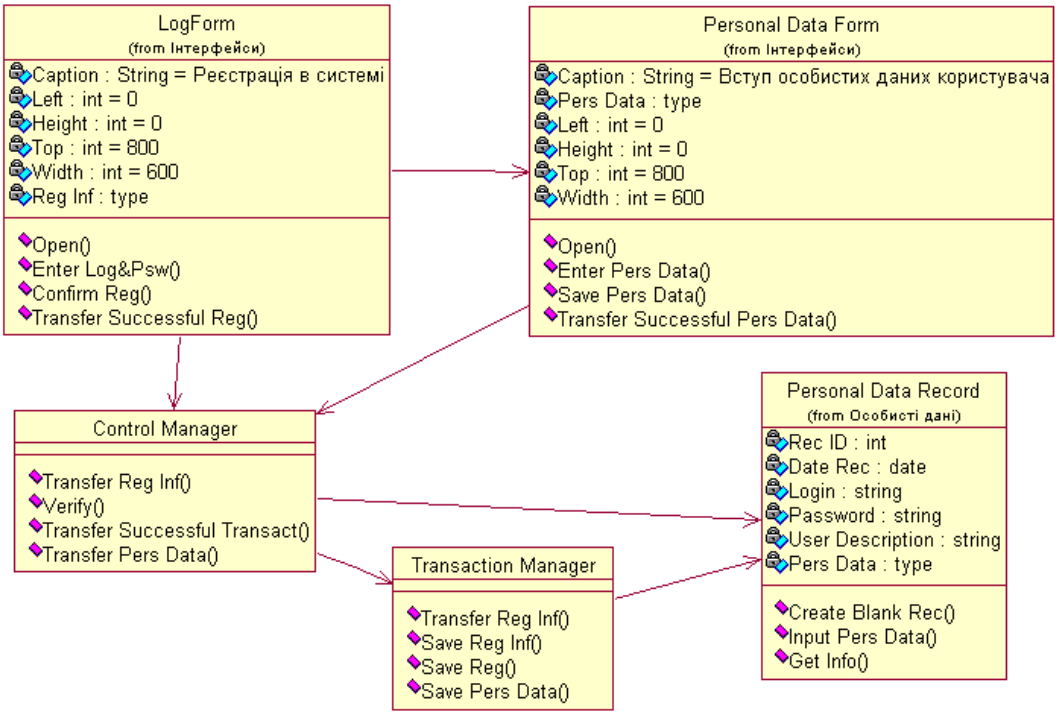
```
        self.my_chat_member_handlers = Handler(self,
middleware_key='my_chat_member')

        self.chat_member_handlers = Handler(self, middleware_key='chat_member')

        self.chat_join_request_handlers = Handler(self,
middleware_key='chat_join_request')

        self.errors_handlers = Handler(self, once=False, middleware_key='error')


        self.middleware = MiddlewareManager(self)


        self.updates_handler.register(self.process_update)


        self._polling = False

        self._closed = True

        self._dispatcher_close_waiter = None


        self._setup_filters()
```

Додаток Г (Принципова схема)



Послідовність дій

Додаток Д (Функціональна схема)



Діаграма класів

Додаток Г (Структурна схема)