Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет __Комп'ютерної інженерії та управління_____
(повна назва)

Кафедра Автоматизації проєктування обчислювальної техніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти ____другий (магістерський)_____
(рівень вищої освіти)

__Технології створення віртуальної бібліотеки кіберуніверситету на основі _____телеграм боту_____
(тема)

Виконав: здобувач 2 року навчання,
групи ___СКСм-23-1

_____Ладоня В.С._____
(прізвище, ініціали)

Спеціальність 123- Комп'ютерна інженерія____
(код і повна назва спеціальності)

Тип програми _____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма _____Спеціалізовані_____
_____комп'ютерні системи_____
(повна назва освітньої програми)

Керівник _____доц. Шкіль О.С._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ _Чумаченко С.В._____
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерної інженерії та управління _____

Кафедра_____Автоматизації проєктування обчислювальної техніки__

Рівень вищої освіти____другий (магістерський)_____

Спеціальність_____123_ Комп'ютерна інженерія _____
(шифр і назва)

Тип програми _____Освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма _____Спеціалізовані комп'ютерні системи_____
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві_____Ладоні Віталію Сергійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи ___Технології створення віртуальної бібліотеки кіберуніверситету на основі телеграм боту_____
_____
затверджена наказом по університету від _08_ ____11_____2024 р. № __1189 Ст___
2. Термін подання студентом роботи до екзаменаційної комісії _15_ _січня_____ 2025 р.
3. Вихідні дані до роботи _____
_Гіперконвенгентні системи_____
_Телеграм бот_____
_Мова програмування Python_____
_____
_____
_____
4. Перелік питань, що потрібно опрацювати в роботі _____
_1 Аналіз актуальності використання телеграм ботів як частини кіберуніверситету_____
_2 Дослідження архітектур гіперконвергентних систем_____
_3 Дослідження процесу розробки програмного забезпечення з використанням API_____
_4 Розробка телеграм боту за технічним завданням на основі потреб університету_____
_5 Дослідження ефективності кросплатформеного програмного застосунку_____
_____
_____

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 18

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

7. Дата видачі завдання 02.09.2024

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання	02.09.2024 – 06.09.2024	
2	Аналіз предметної області	06.09.2024 – 30.09.2024	
3	Аналіз джерел з проблемної галузі	30.09.2024 – 07.10.2024	
4	Розробка логічних та структурних схем програмного продукту	07.10.2024 – 11.10.2024	
5	Підбір навчальної вибірки для проведення дослідження	11.10.2024 – 18.10.2024	
6	Написання програмного коду чат – боту та віртуального хоста з базою даних	18.10.2024 – 23.11.2024	
7	Тестування та налагодження проекту	23.11.2024 – 30.11.2024	
8	Оформлення пояснювальної записки	30.11.2024 – 13.12.2024	
9	Оформлення графічного матеріалу	13.12.2024 – 23.12.2024	
10	Перевірка виконаного проекту керівником	23.12.2024 – 31.12.2024	

Здобувач _____
(підпис)

Керівник роботи _____ доц. Шкіль О.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 59 сторінок, 17 рисунків, 6 лістингів, 8 джерел за переліком посилань.

ВІРТУАЛЬНА МАШИНА, ЧАТ-БОТ, ТЕЛЕГРАМ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ГІПЕРКОНВЕРГЕНТНА СИСТЕМА, API, БАЗА ДАНИХ, КІБЕРФІЗИЧНА СИСТЕМА.

В кваліфікаційній роботі досліджено та розроблено телеграм чат-бот, як частину системи кіберуніверситету для автоматизації регулярних процесів обміну навчальною документацією. Основною задачею є створення інтерфейсу для взаємодії студента за допомогою месенджеру Телеграм з базою даних бібліотеки кіберуніверситету, яка розгорнута на віртуальній машині.

Проведено аналітику актуальності даної технології, її переваг над іншими варіантами розробки. Також було переглянуто та порівняно інші технології кросплатформеної розробки з даним варіантом.

Досліджено архітектуру гіперконвенгентних систем, їх переваги та недоліки в розробці або використанні. Проведено аналіз схожих систем на прикладі вже сформованих, бесіди з користувачами або розробниками таких систем.

Розроблено логічні алгоритми та структурні схеми програми для реалізації поставленої задачі. Створено віртуальну машину та веб-сервер для підключення логіки програми до API Телеграму. Розгорнуто та створено базу даних, з використанням всіх можливих норм та правил нормалізації. Створено програму, яка дозволяє обмін літературою в кібеуніверситеті за допомогою чат-бота, з можливістю модерації вхідного контенту уповноваженою на це людиною.

ABSTRACT

The explanatory note contains 59 pages, 17 figures, 6 listings, 7 sources in the bibliography.

VIRTUAL MACHINE, CHATBOT, TELEGRAM, SOFTWARE, HYPERCONVERGED SYSTEM, ARI, DATABASE, CYBER-PHYSICAL SYSTEM.

In the qualification work, a telegram chatbot was researched and developed as part of a cyber university system to automate the regular processes of exchanging educational documentation. The main task is to create an interface for student interaction using the Telegram messenger with the cyber university library database, which is deployed on a virtual machine.

An analysis of the relevance of this technology and its advantages over other development options was conducted. Other cross-platform development technologies were also reviewed and compared with this option.

The architecture of hyper-converged systems, their advantages and disadvantages in development or use are investigated. The analysis of similar systems on the example of already formed systems, interviews with users or developers of such systems is carried out.

Logical algorithms and structural diagrams of the program for the realization of the task were developed. A virtual machine and a web server were created to connect the program logic to the Telegram API. A database was deployed and created using all possible norms and normalization rules. A program was created that allows the exchange of literature in a cyber university using a chatbot, with the ability to moderate incoming content by an authorized person.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЧБ – чат бот

ТГ – телеграм

ВМ – віртуальна машина

ГКС – гіперконвенгентна система

API - Application Programming Interface

БД – база даних

СУБД – система управління базами даних

ГВ – гіпервізор

ВСТУП

В час різноманітності пристроїв, якими користуються звичайні люди, перед розробниками програмного забезпечення, застосунків, сайтів та інших інформаційних технологій постає задача забезпечити доступ до свого продукту для кожного. Веб-розробники роблять адаптації під різні браузери та пристрої, розробники мобільних застосунків пишуть їх під різні моделі операційних систем тощо. Здебільшого розробник сам обирає під який пристрій, або браузер продукт буде адаптовано, але від цього росте час розробки продукту, час його тестування, та звісно збільшуються витрати на нього.

Для деяких проектів можна використовувати вже готові продукти які надають API (Application Programming Interface). За допомогою API можна створити свій проект на базі вже існуючої програми. Це вирішить проблему адаптації, так як програмний продукт вже адаптований під більшість пристроїв і систем.

Однією з таких програм є месенджер Телеграм, який надає можливість створення на своїй базі телеграм ботів. Цей програмний продукт можна використовувати на персональних комп’ютерах у вигляді застосунку або у вигляді веб версії, на телефонах різних моделей. Телеграм боти є універсальним рішенням для багатьох задач. Для прикладу: боти для спостереженням за ринком крипто активів, боти-адміністратори груп, боти запису до лікаря, боти для пошуку роботи тощо.

В цьому питанні Телеграм чудово пропрацював можливість взаємодії між користувачем і ботом і надав неймовірно розширену палітру інструментів. Розробник може отримати дані від користувача, обробити їх так як потрібно, і відправити результат користувачу, використовуючи бота як засіб для виведення інформації.

1 ВІРТУАЛЬНА БІБЛІОТЕКА – ЯК ОДНА З ЧАСТИН РОЗУМНОГО КІБЕРУНІВЕРСИТЕТУ

1.1 Кіберсоцільна система – розумний кібер-університет

Розумний кіберуніверситет – це система, яка формує культуру соціально-технологічних відносин, об'єднуючи персонал і сучасну інфраструктуру в єдину мережу. Її завдання полягає у проведенні передових наукових досліджень і підготовці затребуваних фахівців із науковими ступенями через ефективний моніторинг та хмарне управління цифровими науково-освітніми процесами. Основною метою є залучення інвестицій і забезпечення високого рівня життя працівників.

Розумна кіберфізична система (Smart Cyber Physical System, SCPS) являє собою мережу взаємопов’язаних віртуальних і реальних компонентів, інтегрованих у цифровий простір. Ця система забезпечує точний фізичний моніторинг, ефективне хмарне управління і здатність до самонавчання, дозволяючи досягати поставлених цілей у реальному часі.

Поняття "розумний" (Smart) описує процеси або явища, які базуються на мережевій взаємодії між адресованими компонентами в часі та просторі, а також із навколишнім середовищем. Це досягається завдяки використанню технологій самонавчання для ефективного досягнення цілей.

Кіберпростір визначається як інтегрована система цифрових процесів і явищ, які взаємодіють між собою за допомогою телекомунікаційної інфраструктури. Цей простір підтримує функції моніторингу, обчислення, зберігання, транзакцій і управління, спрямовані на досягнення конкретних завдань.

Основною перевагою кіберуніверситету є пришвидшення комунікації між різними групами користувачів, human-free контроль над процесами

документообігу, управління кадрами, контролем над освітнім процесом, що дозволяє уникнути конфліктних ситуацій, пов’язаних з корупцією.

Майбутнє людства пов'язане з ідеєю створення human-free хмарного кібер-управління управління соціальними інститутами, спрямованого на реалізацію відкритого та об'єктивного регулювання оцифрованими процесами, де замість корумпованого керівника виступає неупереджена кібер-система. Факт, вимірювання, оцінка, дія - формат циклу кібер-системи управління, пов'язаної з процесами моніторингу, вимірювання та управління, що ґрунтується на постулаті: «Немає вимірювання - немає управління». Синтез матриці компетенцій для рейтингування - цифрового оцінювання процесів або явищ на основі параметрів метрики, складеної експертами або системою аналізу великих даних в Інтернеті. Використання матриці компетенцій для транспарентного кіберрозподілу моральних і матеріальних стимулів між учасниками метричного оцінювання відповідно до рейтингів процесів чи явищ.

Серед основних груп користувачів кібер-університету можна виділити 4 групи людей.

1. Експерти вченої ради, які генерують рішення щодо метричного перетворення всіх структурних компонентів університету: відносини, кадри, інфраструктура, управління, напрямок руху на основі досвіду провідних університетів світу.

2. Виконавці ухвалених рішень: ректор, проректори і декани - чиновники, які створюють комфортний творчий клімат в університеті шляхом надання сервісів, що звільняють учених і професорів від відволікаючої і часозатратної бюрократичної діяльності. Команди експертів і менеджерів-виконавців не повинні перетинатися за аналогією з функціями парламенту та уряду. Тому ректор не повинен бути головою вченої ради. Так само як і членами вченої ради мають бути дійсні науковці - експерти світового рівня, які створюють статутом і положеннями нові конструктивні відносини в університеті.

3. Обслуговуючий персонал у невиробничих відділах забезпечує сервіси, необхідні для творчої життєдіяльності науково-педагогічних кадрів і студентів. Чисельність цього персоналу в університеті не повинна бути більшою за кількість науково-педагогічних працівників.

4. Науково-педагогічні кадри - цінність і надбання університету, які продукують наукову продукцію та освітні сервіси для студентів, що становить предмет експорту на ринок. Увесь неосновний персонал університету, включно з менеджерами вищого рівня, покликаний забезпечувати комфортні моральні та матеріальні умови для творчої праці науковців і викладачів.

Рисунок 1.1 – Хмарні сервіси розумного кіберуніверситету

1.2 Віртуальна бібліотека в системі кіберуніверситету

Віртуальна бібліотека є ключовим елементом цифрової екосистеми кіберуніверситету, яка сприяє розвитку сучасного освітнього процесу, забезпечуючи легкий доступ до навчальних матеріалів, автоматизацію управління ресурсами та інтеграцію інноваційних технологій. Її головною метою є створення централізованої платформи, яка не лише зберігає та

організовує освітні матеріали, а й надає користувачам зручні інструменти для взаємодії та обміну інформацією.

Віртуальна бібліотека дозволяє:

- централізовано зберігати дані, усі ресурси – від підручників до статей – організовані за тематичними категоріями, що спрощує їх пошук і використання;

- автоматизувати процеси, впровадження алгоритмів для модерації завантаженого контенту знижує ризики розповсюдження неякісних або заборонених матеріалів;

- забезпечити багатоплатформенний доступ, інтеграція через Telegram-бот гарантує користувачам доступ до матеріалів з будь-якого пристрою та операційної системи, що робить систему максимально універсальною;

- підтримувати безперервний освітній процес, навіть у разі дистанційного навчання студенти та викладачі можуть швидко отримувати необхідні матеріали, зберігаючи ефективність навчального процесу.

Одним із головних викликів для створення віртуальної бібліотеки є організація її бази даних. Вона повинна бути не лише добре структурованою, але й масштабованою, що дозволить зберігати великі обсяги даних та підтримувати їхню актуальність. Крім того, впровадження автоматизованих механізмів модерації забезпечить дотримання академічних стандартів і виключить можливість розповсюдження матеріалів, які не відповідають освітнім цілям.

Ще однією перевагою віртуальної бібліотеки є можливість інтеграції з іншими системами кіберуніверситету. Це дозволяє створити єдиний інформаційний простір, у якому всі компоненти – від управління навчальним процесом до моніторингу результатів студентів – працюють злагоджено, підвищуючи загальну ефективність функціонування закладу.

Таким чином, віртуальна бібліотека стає не просто цифровим сховищем, а важливим інструментом для розвитку інтелектуального потенціалу кіберуніверситету.

1.3 Переваги використання Telegram ботів

Telegram-боти – це потужний інструмент автоматизації, який стає дедалі популярнішим у бізнес-середовищі завдяки своїй гнучкості та багатофункціональності. Вони здатні виконувати широкий спектр завдань, які зазвичай вимагають значних витрат часу та ресурсів. Це може бути навчання персоналу, надання клієнтам швидкої довідкової інформації, організація опитувань, здійснення продажів, а також інтеграція з іншими системами для обробки даних. Завдяки цьому Telegram-боти активно використовуються в різних галузях – від малого бізнесу до великих корпорацій.

Telegram-боти дозволяють автоматизувати багато рутинних завдань. Наприклад, за допомогою ботів можна організувати автоматичний запис клієнтів на прийом, відправку нагадувань або повідомлень про статус замовлень. Це особливо корисно для підприємств, які працюють у сфері послуг, таких як салони краси, медичні центри чи логістичні компанії. Такі функції значно знижують навантаження на персонал і мінімізують можливі людські помилки.

Одна з головних переваг Telegram-ботів – це швидкість та доступність. Завдяки вбудованому інтерфейсу боти здатні оперативно відповідати на запити користувачів. Це дозволяє значно покращити клієнтський досвід, адже більшість клієнтів очікують миттєвої відповіді на свої запитання. Наприклад, бот може надати інформацію про товари, розповісти про акції, відповісти на поширені запитання або запропонувати індивідуальну консультацію.

Ще однією важливою перевагою Telegram-ботів є можливість персоналізації взаємодії з клієнтами. Бот може збирати інформацію про вподобання користувачів, аналізувати їхню поведінку та пропонувати індивідуальні рішення. Наприклад, бот може рекомендувати товари чи послуги на основі попередніх замовлень клієнта. Крім того, Telegram-боти можуть інтегруватися з CRM-системами, що дозволяє більш ефективно управляти взаємовідносинами з клієнтами.

Платформа Telegram надає унікальні можливості, які вирізняють її серед інших месенджерів. Наприклад:

– кастомні клавіатури: створення інтерактивних кнопок для спрощення навігації користувачів;

– вбудовані запити: можливість отримувати інформацію від бота без необхідності відкривати з ним окремий діалог;

– система платежів: зручний функціонал для здійснення оплат безпосередньо у чаті.

Ці функції дозволяють використовувати Telegram-ботів у таких сферах, як електронна комерція, логістика, туристичні послуги, освітні проєкти та багато інших.

Telegram-боти не тільки допомагають автоматизувати процеси, але й дозволяють значно скоротити витрати на обслуговування клієнтів. Наприклад, замість того, щоб наймати додатковий персонал для обробки замовлень або відповідей на запитання, компанія може впровадити бота, який виконуватиме ці завдання. Це особливо актуально для малого бізнесу, де кожна оптимізація витрат має велике значення.

Ще однією суттєвою перевагою Telegram-ботів є їхня масштабованість. Бот однаково ефективно працює як із десятком, так і з тисячами користувачів, що дозволяє бізнесу легко адаптуватися до зростання клієнтської бази. Крім того, функціонал бота можна поступово розширювати, додаючи нові функції та інтеграції. Наприклад, бот може почати з простих

функцій, як-от відповіді на запити, а з часом отримати можливості для роботи з базами даних, аналітики або інтеграції зі сторонніми сервісами.

А тепер достатньо подивитися на сухі цифри:

– на 92% збільшився обсяг ринку чат-ботів за останні кілька років. Чат-боти є найшвидшим каналом комунікації у брендів;

– 87,2% користувачів-клієнтів повідомляють про нейтральний або позитивний досвід роботи з чат-ботами;

– 58% B2B-компаній використовують чат-ботів, B2C-бізнеси – у 42% випадків.

– до 90% відгуків може згенерувати взаємодія чат бота в Телеграм з найбільш зацікавленими клієнтами. У середньому ж чат-боти генерують 35-40% відповідей;

– за словами 68% користувачів-клієнтів, їм подобаються чат-боти, тому що вони дають швидкі відповіді.

1.4 Постановка задачі

Станом на 2024 рік університет не має єдиної системи для цифрового обміну літературою. Це є доволі критичною проблемою, так як щоб отримати книгу в електронному варіанті, студент має знати в якого викладача ця книга є, які взагалі книги є на даний момент, а для комунікації персонал університету має використовувати пошту.

Метою цього проекту є створення та впровадження до розумного кіберуніверситету (Smart Cyber University) віртуальної бібліотеки, що забезпечує підвищення якості освітніх послуг і наукових досягнень завдяки цифровому моніторингу, активному хмарному управлінню процесами та ліквідації паперових носіїв.

Об'єктом дослідження є науково-освітні процеси та організаційні структури вищих навчальних закладів, спрямовані на підготовку

кваліфікованих кадрів (бакалаврів, магістрів, докторів) та створення конкурентоздатної науково-технічної продукції.

Предметом дослідження є віртуальна бібліотека, та кросплатформенна програма для доступу в неї, яка знаходиться в кіберфізичній системі університету.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- здійснити аналіз існуючих систем віртуальних бібліотек у вищих навчальних закладах для визначення їхніх сильних і слабких сторін;
- дослідити сучасні методи інтеграції баз даних з кросплатформенними інтерфейсами, зокрема Telegram-ботами;
- спроектувати структуру бази даних, яка забезпечить ефективне зберігання та швидкий доступ до навчальних матеріалів;
- розробити програмний модуль для управління даними бібліотеки, що включає функції додавання, модерації та пошуку;
- створити Telegram-бот як основний інтерфейс користувача, який дозволяє взаємодіяти з бібліотекою через текстові команди та вбудовані кнопки;
- протестувати систему в реальних умовах, зокрема в рамках одного факультету, з метою отримання зворотного зв’язку та оптимізації функціоналу.

Додатково необхідно впровадити систему звітності, яка дозволить відстежувати статистику використання бібліотеки, аналізувати популярність матеріалів та виявляти потребу в нових ресурсах.

Усі ці задачі в сукупності сприятимуть формуванню сучасної та ефективної віртуальної бібліотеки, яка інтегрується в систему кіберуніверситету та відповідає сучасним стандартам цифрового навчання.

2 ТЕХНОЛОГІЇ СТВОРЕННЯ TELEGRAM БОТУ З ВИКОРИСТАННЯМ СУБД ТА ГІПЕРКОНВЕНГЕНТНИХ СИСТЕМ

2.1 Мова програмування Python та необхідні бібліотеки

2.1.1 Мова програмування Python та середовище розробки PyCharm

PyCharm - це середовище програмування для мови Python, або IDE. Середовищами називають програми, в яких можна писати, запускати та налагоджувати код, встановлювати нові розширення та додаткові модулі. Це потужний багатофункціональний інструмент для розробників.

Її розробила компанія JetBrains, яка також зробила відому IDE для мови Java під назвою IntelliJ IDEA. PyCharm користується понад 50% розробників на Python: вона зручна, зрозуміла та багатофункціональна. PyCharm існує для кількох операційних систем: Windows, Linux і macOS. Вона підтримує різні версії Python: і 2.x, і 3.x. Її широкі можливості роблять розробку на Python швидшою та ефективнішою.

Мови програмування влаштовані так, що писати код на них можна де завгодно, навіть у «Блокноті». Головне - зберегти файл у потрібному розширенні, а потім запустити за допомогою встановленого інтерпретатора або компілятора. Але більшість розробників все одно користуються IDE. У середовищах, зокрема PyCharm, є підсвічування синтаксису: різні ключові слова і конструкції виділяються кольорами, тож потрібне місце в коді простіше знайти.

В IDE можна не тільки написати код, а й запустити його і відразу подивитися на результат роботи програми. Там є інструменти для налагодження, контролю версій, наприклад, за допомогою Git; можна в кілька кліків встановлювати сторонні бібліотеки та фреймворки.

Професійна версія середовища дає можливість синхронізуватися з іншими розробниками та робити командну роботу ефективнішою.

Для Python існує кілька популярних середовищ. PyCharm обирає більшість: його легко налаштовувати, у ньому зручний і функціональний інтерфейс, а можливостей багато. Він підходить для широкого спектра завдань: від автоматичного тестування до машинного навчання. Інші середовища для мови зазвичай або менш зручні, або більш вузькі в застосуванні.

IDE - це не просто редактор для коду. У нього набагато більше можливостей, що полегшують життя розробнику. PyCharm можна уявити як комбайн, де є більшість функцій, важливих для програміста. Ось лише деякі речі, які в ньому можна робити.

Проект мовою програмування - це не просто створення файлу. Коли розробник створює проєкт у PyCharm, середовище виділяє під нього окрему папку, де зберігає все, що пов'язано з цим проєктом. Так потрібні файли і компоненти знаходяться під рукою. Структуру проєкту PyCharm показує в лівій частині інтерфейсу і дає змогу в будь-який момент перемкнутися на файл, що цікавить, всередині нього. Писати код на Python. Усередині проекту можна створити файл у потрібному розширенні та писати в ньому код. Синтаксис підсвічується автоматично, причому параметри підсвічування можна налаштувати. До того ж PyCharm дає можливість одразу перевіряти правильність написання і виділяти помилкові моменти. Середовище допомагає писати чистіший код.

IDE підключені до інтерпретатора або компілятора потрібної мови. Python - інтерпретована мова, і PyCharm може запустити її інтерпретатор. Тому код можна виконати прямо всередині середовища, для цього не знадобиться відкривати консоль або будь-який сторонній додаток. В інтерфейсі IDE є кнопка запуску: достатньо натиснути на неї, і код запуститься. Результат виконання програма покаже одразу - виведе в спеціальну консоль всередині середовища або відкриє нове вікно.

У середовищі є інструменти для налагодження коду. Наприклад, можна налаштувати режим налагодження так, щоб показувати значення різних змінних у будь-який момент часу. Або зупинити виконання на конкретному рядку і дивитися, чи нормально код працює в цьому моменті, - це допомагає знайти місце, в якому відбувається помилка. Є й покрокове виконання: програма виконує один рядок коду і зупиняється, щоб розробник міг перевірити, чи правильно вона працює на цій ділянці.

Автоматичне тестування - одна з поширених сфер застосування Python. І робити це в PyCharm зручно. У середовищі є інструменти для автоматичної генерації коду, і до нього легко підключити модулі для тестування.

Для самого середовища теж є модулі, що розширюють його функціональність. Їх можна встановити зсередини IDE. Приклади таких модулів - перевірка читабельності коду, підказки за допомогою штучного інтелекту, розставлення дужок, яких бракує, і багато іншого. Деякі плагіни змінюють інтерфейс, якщо розробнику не подобається стандартний. Інші розширюють можливості самого середовища. Після встановлення доповненням можна користуватися як частиною IDE.

PyCharm призначена для Python, але в ній є підтримка й інших мов. Наприклад, Python часто використовується у веб-розробці, тому IDE також підтримує JavaScript для браузера і SQL для баз даних. Крім JavaScript, підтримуються засновані на ньому TypeScript і CoffeeScript, популярні JS-фреймворки, а також мови HTML і CSS для верстки. У середовищі можна користуватися шаблонізаторами - спеціальними інструментами, які допомагають створювати шаблони для веб-сторінок. Мови шаблонізаторів PyCharm теж розуміє.

2.1.2 Бібліотека PyTelebot

PyTelebot — це бібліотека для Python, яка спрощує створення ботів для Telegram. Вона побудована на основі Telegram Bot API і надає зручний

інтерфейс для роботи з ботами, що дозволяє розробникам легко реалізовувати різні функції.

Основні особливості PyTelebot:

– простота використання: Бібліотека має зрозумілу структуру, що дозволяє швидко почати розробку. Ви можете швидко налаштувати бота, створивши прості обробники команд;

– обробка повідомлень: Ви можете легко обробляти текстові повідомлення, команди, а також різні типи медіа (фото, відео, документи тощо);

– робота з API: PyTelebot підтримує всі основні функції Telegram API, такі як створення клавіатур, опитувань, сесій, а також робота з відправкою повідомлень;

– асинхронність: Бібліотека підтримує асинхронне програмування, що дозволяє обробляти багато запитів одночасно, не блокуючи виконання програми;

– розширення функціоналу: Ви можете інтегрувати сторонні API, підключати бази даних, а також реалізовувати будь-які бізнес-логіки у вашого бота.

Ось простий приклад створення бота, який відповідає на команду /start.

Лістинг 2.1 – Приклад простого коду для використання чат-боту

```
import telebot
API_TOKEN = 'YOUR_API_TOKEN'
bot = telebot.TeleBot(API_TOKEN)
@bot.message_handler(commands=['start'])
def send_welcome(message):
    bot.reply_to(message, "Привіт! Я ваш бот.")
@bot.message_handler(func=lambda message: True)
def echo_all(message):
    bot.reply_to(message, message.text)
bot.polling()
```

З переваг pyTelebot можна виділити його відкритий код який дозволяє вносити туди зміни, тому ця бібліотека постійно змінюється та покращується. Велика спільнота, що спрощує вирішення деяких проблем та розширює інформаційну базу з якої можна почерпнути знань. Дуже великим плюсом є доволі проста документація, зрозуміла навіть початківцю. Та найголовніша перевага

Переваги:

– простота у використанні: Telebot надає простий та інтуїтивно зрозумілий спосіб створення ботів, що спрощує початок роботи;

– великі приклади коду: Бібліотека супроводжується безліччю прикладів коду і добре структурованою документацією;

– модульність і розширюваність: Telebot дає змогу створювати модульні боти і легко розширювати їхню функціональність;

– швидкий старт: Створення базового бота з Telebot може зайняти всього кілька хвилин.

Недоліки:

– обмежений функціонал: Telebot може виявитися неспроможним для проєктів, що вимагають складної функціональності;

– менш активна спільнота: Порівняно з деякими іншими бібліотеками, спільнота навколо Telebot може бути менш активною.

2.2 СУБД Maria DB 10.8

MariaDB — це сучасна реляційна система управління базами даних, яка виникла як форк MySQL у відповідь на зміну політики розробки після придбання MySQL корпорацією Oracle. Основною метою MariaDB було зберегти відкритий характер MySQL і надати користувачам можливість отримувати доступ до стабільного, продуктивного та постійно вдосконалюваного інструменту для роботи з даними.

MariaDB залишається сумісною з MySQL, що полегшує міграцію з

однієї системи на іншу. Ця сумісність охоплює більшість SQL-синтаксису, API, драйверів і функціональних можливостей, що дозволяє використовувати MariaDB як пряму заміну MySQL без необхідності серйозних змін у додатках або архітектурі систем.

MariaDB пропонує потужні засоби для управління даними. Вона має високу продуктивність завдяки вдосконаленому механізму оптимізації запитів і підтримці сучасних процесорів. Це дозволяє швидко обробляти великі обсяги даних, навіть у середовищах із високими навантаженнями. Завдяки різноманітним сховищам даних, включно з InnoDB, MyRocks, Aria та іншими, користувачі можуть вибирати найоптимальніший варіант для своїх потреб.

Система вирізняється високим рівнем безпеки, що досягається завдяки підтримці аутентифікації на основі SSL, інтеграції методів шифрування та автоматизованим засобам резервного копіювання. Також забезпечується можливість налаштування доступу до бази даних через гнучкі політики ролей і прав.

Важливим аспектом MariaDB є її гнучка конфігурація. Адміністратори можуть точно налаштовувати параметри роботи сервера, створюючи середовище, яке найкраще відповідає потребам програми. Підтримка широкого спектра мов програмування, таких як PHP, Python, Java, C++, Node.js та інших, розширює можливості інтеграції MariaDB у різноманітні проєкти.

MariaDB активно розвивається й пропонує інноваційні функції, які роблять її більш гнучкою та ефективною. Наприклад, підтримка віртуальних колонок дозволяє створювати динамічні обчислення без необхідності зберігати додаткові дані, а інструменти для роботи з JSON полегшують обробку даних у нестандартних форматах.

У MariaDB реалізовано вдосконалені механізми реплікації. Вони включають асинхронну реплікацію, яка забезпечує високу продуктивність, синхронну реплікацію для збереження цілісності даних і кластерні рішення,

такі як Galera Cluster, які забезпечують стійкість до збоїв і масштабованість.

Ця СУБД ідеально підходить для роботи в хмарних середовищах, оскільки має інтеграцію з популярними платформами, такими як AWS, Google Cloud і Microsoft Azure. Вона також підтримує контейнери, наприклад Docker, що полегшує її розгортання в різних середовищах.

Сучасний оптимізатор запитів MariaDB дозволяє виконувати навіть найскладніші SQL-запити швидше, ніж це можливо в інших СУБД. Крім того, вбудовані інструменти моніторингу продуктивності дають змогу розробникам і адміністраторам ідентифікувати й усувати "вузькі місця" в роботі системи.

Активне співтовариство MariaDB є ще одним її значним плюсом. Постійна підтримка та розвиток бази даних забезпечуються як незалежними розробниками, так і компаніями, які пропонують комерційні рішення. Це дозволяє MariaDB залишатися актуальною навіть в умовах швидкої зміни технологій.

Таким чином, MariaDB — це не просто альтернатива MySQL, а потужний, гнучкий і сучасний інструмент, здатний задовольнити потреби як невеликих проектів, так і великих корпоративних систем. Її відкритий характер, висока продуктивність, розширені функції безпеки та активна підтримка роблять її одним із найкращих виборів у сфері управління реляційними базами даних.

2.3 Утиліта OpenServerPanel 5.4.3

Open Server - це портативний локальний WAMP/WNMP сервер, що має багатофункціональну керуючу програму і великий вибір компонентів, що підключаються. Представлений пакет програм не є черговою аматорською збіркою, зібраною «на коліні», це перший повноцінний професійний інструмент, створений спеціально для веб-розробників з урахуванням їхніх рекомендацій і побажань.

У разі відсутності на комп'ютері потрібних системних компонентів Open Server встановить їх сам, досить вибрати в меню [Інструменти - Перший запуск] якщо сервер запускається на комп'ютері вперше.

Цікаві переваги використання цього локального серверу:

– детальний перегляд логів усіх компонентів у реальному часі;

– вибір HTTP, СУБД і PHP модулів у будь-якому поєднанні;

– підтримка SSL і кириличних доменів з коробки;

– підтримка аліасів або інакше доменних покажчиків, а також зручна форма їх налаштування (привіт любителям мультисайтингу в Drupal!);

– створення локального піддомену без втрати видимості основного домену в мережі інтернет;

– доступ до доменів (в один клік) і швидкий доступ до шаблонів конфігурації модулів;

– багатомовний інтерфейс (Російська, Українська, Білоруська, Англійська);

Програма постійно вдосконалюється, всі адекватні прохання з боку користувачів Open Server детально вивчаються і більшість з них реалізується!

Для того, щоб розпочати роботу із сервером, для початку, необхідно його встановити. Скачувати потрібно з офіційного сайту, відповідно, ввівши код з картинки, та, обравши дистрибутив.

Версія Ultimate є повною. До її складу входять Apache, PHP, MySQL, Nginx та програмне середовище з корисним софтом. Premium – включає тільки серверну частину. Basic – базова версія, в якій немає пакету додаткових програм, модулів PhpPgAdmin, Git та інших.

Після скачування потрібно відкрити файл установки та розпакувати його на диск C, проте, надійнішим зберігання Ваших сайтів буде на диску D. Потім, перейшовши на цей диск, обираєте папку OpenServer та, в залежності від розрядності системи Windows (64 чи 86-розрядна), запускаєте програму (від імені адміністратора).

Якщо все виконано вірно, то після встановлення в області повідомлень (іншими словами – трей) Вашого комп'ютера з'явиться червоний прапорець панелі управління OpenServer. Далі натискаєте лівою кнопкою миші по прапорцю та обираєте «запустити».

Після цього прапорець змінить свій колір на зелений.

Особливостями OpenServer є:

- можливість роботи із флеш-накопичувача;
- швидкість запуску платформи;
- домен створюється завдяки створенню простої папки;
- захист сервера від зовнішнього втручання;
- швидкі доступи до доменів та модулів;
- можливість роботи з кириличними доменами.

Для того, щоб створити новий сайт, необхідно перейти до папки під назвою «domains», натискаєте ПКП + створити папку та задаєте потрібну Вам назву. Після цього в даній папці створюєте новий текстовий документ, наприклад, з назвою – index.php– для перевірки роботи сервера та вводите назву заголовку(h1) за допомогою спеціального коду. Після цього перезапускаєте сервер та перевіряєте наявність нового сайту, перейшовши в меню на «Мої сайти».

OpenServer має хороший інтерфейс, широкі можливості та сферу застосування. Він є одним із найкращих та найбільш використовуваних серверів. Однією із особливостей являється саме простота використання – зручне меню, легкість у створенні хостів, портативність та безліч інших критеріїв, що надають істотні переваги. Основним призначенням, як згадувалось попередньо, є розробка сайтів та їх тестування у локальній мережі.

2.4 Конвергентні та гіперконвергентні системи

2.4.1 Загальні положення про конвергентні та гіперконвергентні

системи

Обсяг, швидкість і динамічність інформаційних потоків сьогодні може залишити непідготовлене підприємство позаду конкурентів. Це стосується не тільки компаній і стартапів, які активно впроваджують нові технології, а й будь-якого бізнесу - від великих корпорацій до середніх і дрібних підприємств.

Основою сучасного бізнесу стала IT-інфраструктура. Якщо вона працює ефективно, то бізнес може адекватно відповідати на запити споживачів, гнучко реагувати на зміни ринку і стабільно розвиватися.

Компанії, що працюють без конвергентних і гіперконвергентних середовищ, змушені самостійно створювати, інтегрувати й обслуговувати IT-інфраструктуру, що забирає ресурси і часто заважає займатися безпосередньо бізнесом.

У середині 2000-х років стало зрозуміло, що самостійно купувати обладнання для дата-центрів, збирати його, налаштовувати, запускати, оптимізувати та обслуговувати - це неефективно, ненадійно і складно. Тоді й почали з'являтися модульні інтегровані системи. Цей момент став початком трансформації. Замість того щоб створювати окремі розрізнені рішення, компанії захотіли купувати готові.

У 2011 році компанія Hewlett-Packard представила кілька апаратно-програмних систем, націлених на оптимізацію роботи з дата-центрами та розвиток хмарної інфраструктури. Компанія була впевнена, що нові конвергентні середовища забезпечать ефективний розвиток корпоративного IT. Так і вийшло.

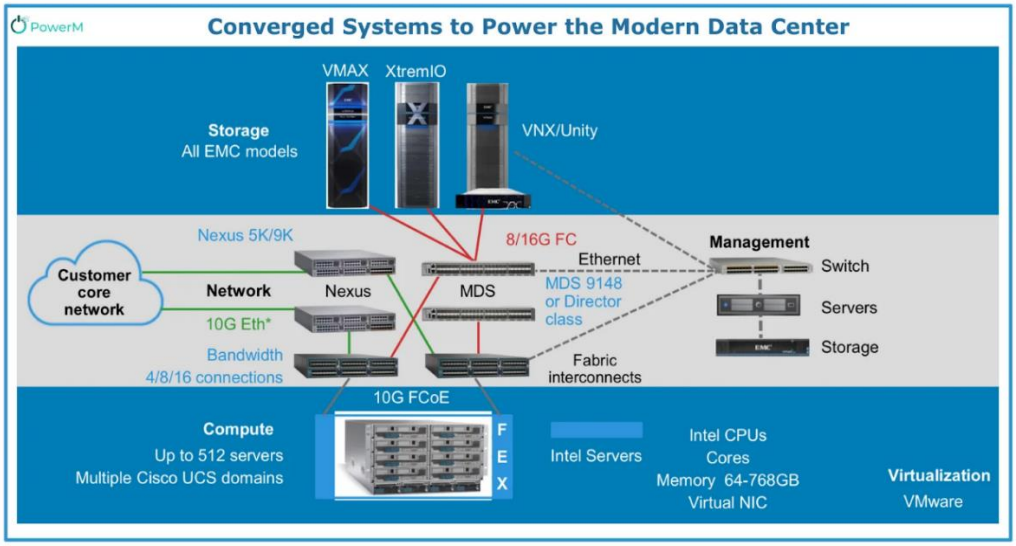

Рисунок 2.1 – Загальний вигляд конвергентної системи

Конвергентна інфраструктура/середовище (Converged Infrastructure → CI) дала змогу створювати інформаційні системи, де різні компоненти групувалися і управлялися в одному комплексі. Типове конвергентне середовище містить у собі сервер, пристрій зберігання даних, мережеве обладнання та програми для управління й автоматизації всього цього добра.

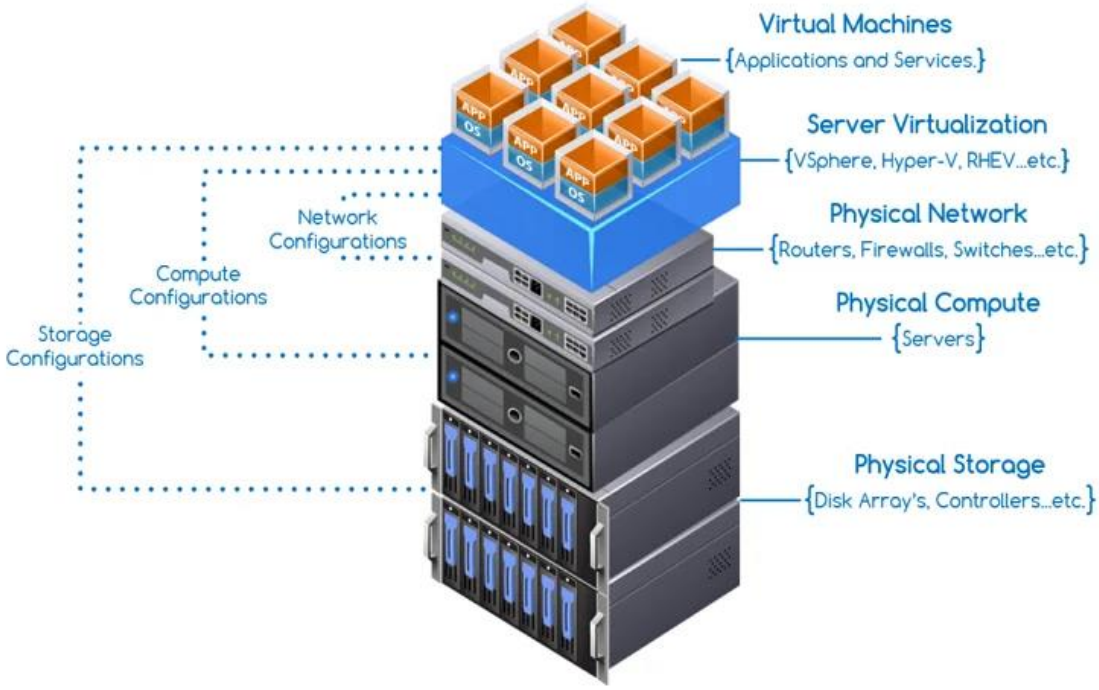

Рисунок 2.2 – Загальний вигляд гіперконвенгентної системи

Через рік концепція конвергентності отримала розвиток. Інженери, аналітики та журналісти заговорили про гіперконвергентні інфраструктури (Hyper Converged Infrastructure → HCI). Такі рішення об'єднують обчислювальні потужності, сховища і мережі за допомогою програмних засобів, а управління ними відбувається через загальну консоль адміністрування.

Конвергентні середовища об'єднали ресурси, дані та управління ними в одній системі, що налаштовується. Гіперконвергентні - дозволили компаніям ще більше абстрагуватися від апаратної інфраструктури і показали, що може дати віртуалізація бізнесу. Останні стали логічною еволюцією перших. Сьогодні ці види існують і розвиваються паралельно, тому що мають особливості та підходять для різних завдань.

Конвергентна інфраструктура підвищує ефективність IT-управління і прискорює розгортання проєкту. Якщо компанія переходить на сервісну модель або тільки виходить на ринок, то CI допоможе:

– централізовано керувати різними функціями і пристроями;

– передавати і динамічно розподіляти різнотипний трафік в одному потоці;

– підготувати і виділити обчислювальні ресурси за години, а не тижні;

– легко переходити на приватні або гібридні хмарні сервіси;

– гнучко реагувати на зміни ринку або бізнес-пріоритетів;

– ефективно масштабувати мережі зберігання даних.

Фахівці виділяють і кілька недоліків конвергентних середовищ. Це висока вартість, порівняно з традиційними рішеннями; обмеження в реалізації, що стосуються, наприклад, конфіденційності даних; висока ціна людської помилки через взаємовплив різних підсистем і ймовірна несумісність компонентів у разі часткової конвергенції. Але загалом CI дає змогу раціональніше використовувати інвестиції в довгостроковій

перспективі.

Гіперконвергентне середовище буде корисним за невеликого штату співробітників і управління складними системами з безліччю завдань. Таким проєктам HCI забезпечить:

– простоту запуску та адміністрування з єдиного інтерфейсу;

– легке розгортання за рахунок стандартних серверних компонентів;

– продуктивність для роботи із середовищами розробки і тестування, хмарними сервісами і Big Data;

– горизонтальне масштабування без лімітів продуктивності контролерів SAN або інтерфейсів передачі даних;

– відмовостійкість завдяки зберіганню копій на різних вузлах і організації територіально розподілених кластерів;

– низьку вартість обслуговування всієї IT-складової бізнесу.

Варто згадати неможливість гранулярного оновлення гіперконвергентної інфраструктури. Якщо обсяг пам'яті закінчуватиметься, а обчислювальна частина працюватиме із запасом, то компанії все одно доведеться нарощувати загальну обчислювальну потужність. Це справедливо для готових рішень, а не для створення власної HCI-платформи.

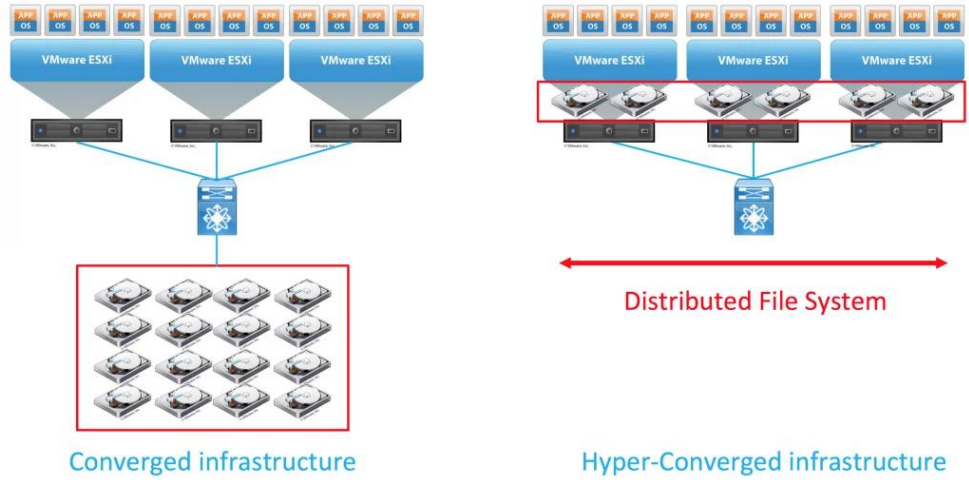

Рисунок 2.3 – Різниця між конвергентною та гіперконвергентною системою

2.4.2 Гіпервізор Hyper-V

Одразу треба зазначити що таке гіпервізор. Гіпервізор або Монітор віртуальних машин – комп'ютерна програма або обладнання процесора, що забезпечує одночасне і паралельне виконання декількох віртуальних машин, на кожній з яких виконується власна операційна система, на одному фізичному комп'ютері (який зветься хост-машина або хост-комп'ютер, англ. host computer). Гіпервізор забезпечує взаємну ізоляцію операційних систем, що виконуються на віртуальних машинах, шляхом розділення фізичних та логічних пристроїв між декількома віртуальними машинами.

Гіпервізор сам по собі є мінімальною операційною системою, що складається з мікро- або нано- ядра та інструментів керування віртуальними машинами. Він створює віртуальні машини, на яких можуть бути запущені окремі операційні системи, віртуалізуючи або емулюючи апаратне забезпечення (в тому числі процесор, оперативну пам'ять та пристрої вводу/виводу), і керує цими віртуальними машинами.

Гіпервізор забезпечує взаємну ізоляцію віртуальних машин шляхом надання пристроїв (у тому числі процесора та оперативної пам'яті) в один з способів:

– розділення часу, коли деякий пристрій надається окремій віртуальній машині на деякий час (звичайно декілька мілісекунд);

– закріплення, коли деякий пристрій надається окремій віртуальній машині в одноосібне володіння;

– емуляції, коли деякий пристрій створюється програмно для кожної окремої віртуальної машини.

Гіпервізор дозволяє незалежне «включення», «перезавантаження», «вимкнення» кожної з віртуальних машин з тією чи іншою ОС. При цьому операційна система, що працює у віртуальній машині під управлінням гіпервізора, може, але не зобов'язана «знати», що вона виконується у віртуальній машині, а не на реальному апаратному забезпеченні.

А тепер повернемося до Hyper-V. Hyper-V, кодова назва Viridian (також відомий як Windows Server Virtualization) – це вбудований гіпервізор, який здатен створювати віртуальні машини в системах під керуванням ОС Windows. Hyper-V прийшов на заміну Windows Virtual PC починаючи з Windows 8. Серверний комп'ютер на якому запущено Hyper-V може бути налаштований як декілька віртуальних серверів, на кожному з яких буде функціонувати своя операційна система і різні додатки.

Hyper-V вперше з'явився в Windows Server 2008 в 64-бітній версії. Автономний сервер Microsoft Hyper-V Server є безкоштовним, однак постачається без GUI, керування відбувається з командного рядка.

Hyper-V реалізує ізоляцію віртуальних машин з точки зору розділу. Розділ є логічною одиницею ізоляції, що підтримується гіпервізором, в якому виконується кожна гостьова операційна система. Гіпервізор обов'язково має принаймні один батьківський розділ, який запускає підтримувану версію Windows Server (2008 та пізнішої версії) або Windows 8 (та пізніші). Програмне забезпечення віртуалізації працює у батьківському розділі та має прямий доступ до апаратних пристроїв. Батьківський розділ створює дочірні розділи, на яких розміщуються гостьові ОС. Батьківський розділ створює дочірні розділи за допомогою Hypercall API, який є інтерфейсом прикладного програмування, що надається Hyper-V. Hypercall API надає методи за допомогою яких батьківські розділи обмінюються даними з дочірніми.

Дочірній розділ не має доступу до фізичного процесора, а також не обробляє його реальні переривання. Замість цього він має віртуальне представлення процесора і виконується в Гостьовій віртуальній адресі (англ. Guest Virtual Address), який, в залежності від конфігурації гіпервізора, може бути не обов'язково неперервним віртуальним адресним простором. Гіпервізор обробляє переривання процесора і перенаправляє їх до відповідного розділу за допомогою логічного Контролера синтетичних переривань (англ. Synthetic Interrupt Controller або SynIC). Апаратне забезпечення Hyper-V може прискорити трансляцію адрес гостьових

віртуальних адресних просторів за допомогою Second Level Address Translation (SLAT), наданого процесором, який називається EPT в Intel та RVI (раніше NPT) в AMD. З приходом Windows Server 2016 для використання ролі Hyper-V підтримка процесором SLAT є обов'язковою (раніше це була лише рекомендація).

Дочірні розділи не мають прямого доступу до апаратних ресурсів, вони мають віртуальний вигляд ресурсів з точки зору віртуальних пристроїв. Будь-який запит на віртуальні пристрої переадресовується через VMBus, який керує запитами, на пристрої батьківського розділу. VMBus є логічним каналом, який забезпечує взаємодію між розділами. Відповідь також перенаправляється через VMBus. Якщо пристрої у батьківському розділі також є віртуальними пристроями, вони будуть перенаправлені далі, поки не досягнуть батьківського розділу, де буде отримано доступ до фізичних пристроїв. Батьківські розділи запускають Virtualization Service Provider (VSP), який підключається до VMBus і обробляє запити доступу до пристроїв з дочірніх розділів. Віртуальні пристрої дочірнього розділу запускають Virtualization Service Client (VSC), який перенаправляє запит до VSP у батьківському розділі через VMBus. Весь цей процес є прозорим для гостьової ОС.

Віртуальні пристрої також підтримують технологію Windows Server Virtualization, що називається прогресивне введення-виведення (англ. Enlightened I/O), для накопичувачів, мережевих та графічних підсистем, зокрема. Enlightened I/O — це спеціалізована реалізація протоколів зв'язку високого рівня для віртуалізації, подібна SCSI, що дозволяє обходити будь-який шар емуляції пристроїв і працювати з VMBus безпосередньо. Це робить комунікацію більш ефективною, але вимагає, щоб гостьова ОС підтримувала Enlightened I/O.

3 ПРОЦЕС РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

3.1 Планування проекту та розділення його на логічні блоки

Перед розробкою будь-якого продукту потрібно одразу створювати план розробки, що спрощує процес роботи, дозволяє вивільняти час на інші заняття та допомагає відстежувати етап, на якому проект знаходиться. Якщо брати проект Telegram чат боту з використання БД, та гіперконвергентної системи, то його можна розділити на наступні етапи:

а) реєстрація чат-боту в месенджері Телеграм;

б) розгортання веб-серверу та створення бази даних для роботи з масивами даних, які вводяться користувачем;

в) написання коду для функціоналу чат боту;

г) створення віртуальної машини та перенесення боту в неї;

д) тестування проекту.

Обов’язково треба зазначити, що створення віртуальної машини та перенесення боту на неї це повторення пункту 2 в зазначеному вище списку. Але чому саме така послідовність? Через те, що бот тестується і розробляється на персональному комп’ютері студента. Саме через це, віртуальній машині не може бути надано більше ресурсів системи, аніж вона має сама. Для прикладу персональний комп’ютер студента має 16 ГБ оперативної пам’яті, шести ядерний процесор від AMD. Віртуальній машині можна надати 8 ГБ, та два ядра для обробки процесів, але це вповільнить процес розробки. Тому було прийнято рішення про розробку та тестування на основній машині, а вже потім готовий продукт буде перенесено до гіпервізора.

Перед розробкою програмного коду потрібно спланувати наступні дії. Для цього потрібно розібрати увесь цикл оброблення та внесення даних, щоб

визначитися з тим, які інструменти потрібні для розробки проекту далі. Якщо узагальнювати увесь проект, то схема зображена на рисунку 3.1 ідеально описує його.

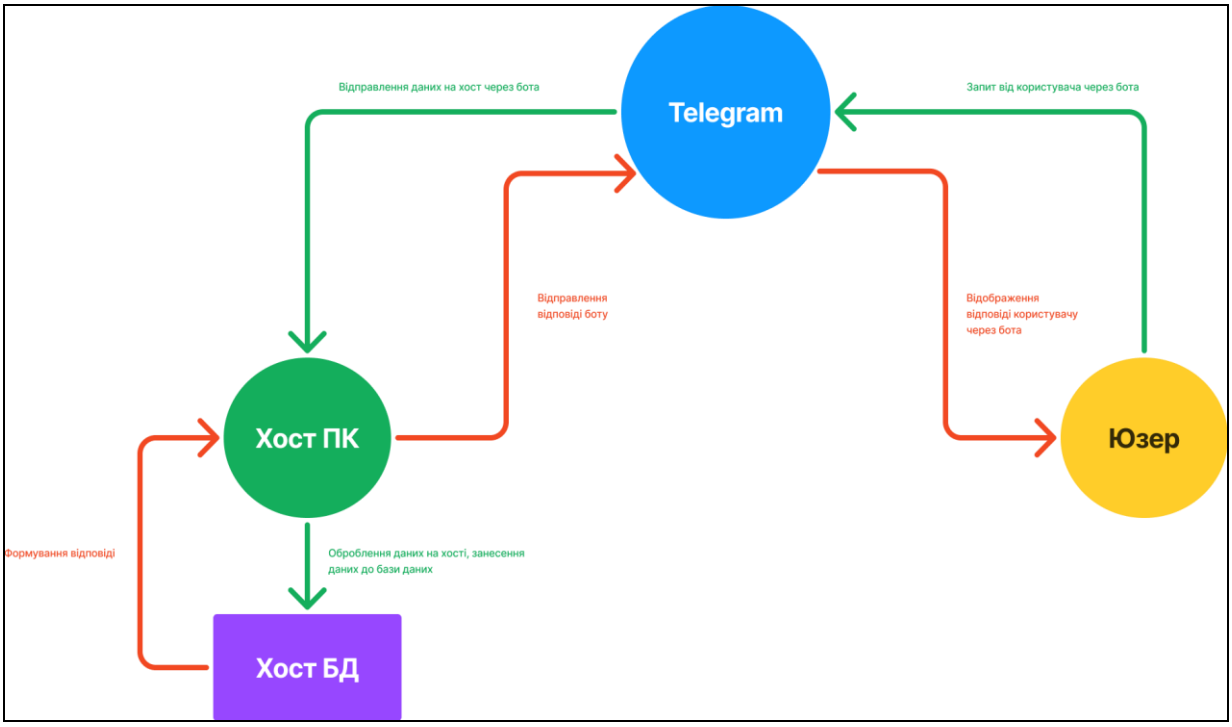

Рисунок 3.1 – Структурна схема проекту

Після створення структурної схеми процесу обробки даних проекту, потрібно спланувати як буде працювати сам чат-бот. В цілому в коді будь-який новий варіант введення або виведення даних виглядає як рендер нового вікна. В самому Python це реалізовано умовними операторами if…else, тобто потрібно розуміти, коли користувачу потрібно рендерити нове вікно з можливими командами. Якщо розбирати проект то ця схема буде виглядати як схема, зображена на рисунку 3.2

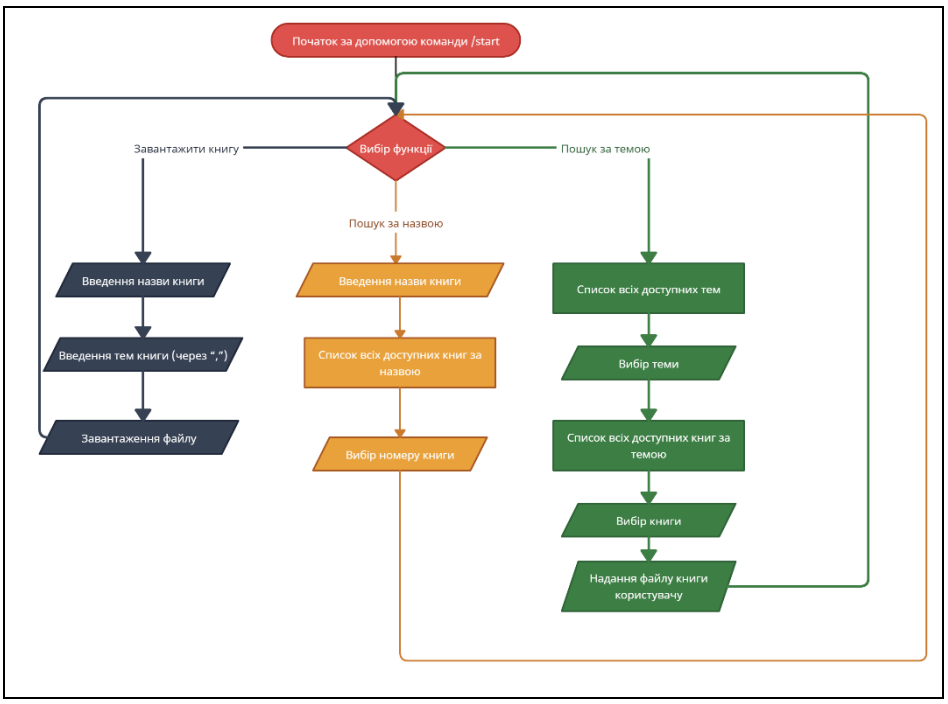

Рисунок 3.2 – Схема роботи бота, та рендеру нових вікон команд

В схемі, яка зображена на рисунку 3.2 не вказано того, що книга має пройти процес модерації після того, як користувач її додав, так як це є дією адміністратора (або модератора), а вона зображує фунціонал на стороні користувача. Процес модерації є обов'язковим, задля уникнення проблем з забороненим контентом в матеріалах бібліотеки. Також цей процес ліквідує можливість спаму інформацією звичайних користувачів, але не ліквідує таку можливість для модератору.

Після цих дій в розробника є розуміння того, як програмний продукт буде виглядати, та що потрібно робити далі.

3.2 Реєстрація бота в месенджері Telegram

Телеграм надає можливість використовувати їх інтерфейс для створення та кастомізації їх ботів. Для цього потрібно скористатися ботом телеграм @BotFather.

Щоб створити бота потрібно скористатися командою /newbot в чаті з BotFather, що зазначено на рисунку 3.3.

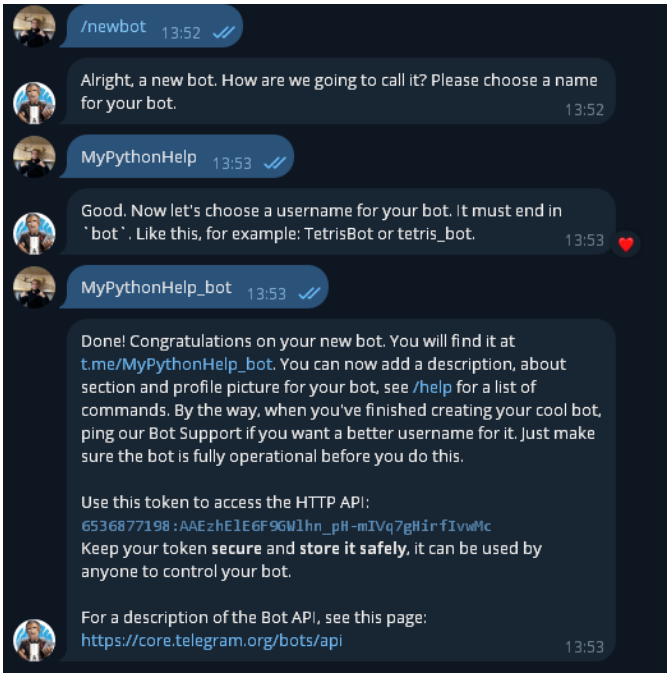

Рисунок 3.3 – Створення боту телеграм

Після того як бота створено, BotFather надає токен бота, за яким розробник матиме доступ до нього.

```
bot = telebot.TeleBot('7815794346:AAHE6j6w7Lxfl8ya_2l33LCDhq1tDLuZhww')

# Словарь для хранения состояния пользователя
user_states = {}
user_data = {}

@bot.message_handler(commands=['start'])
def start(message):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton("👋 Вітання")
    markup.add(btn1)
    bot.send_message(message.from_user.id, "👋 Доброго дня! Я бібліотекар! Чим можу допомогти?", reply_markup=markup)
```

Рисунок 3.4 – Використання токену для роботи з ботом

Для того щоб почати роботу з ботом, створюється об’єкт telebot, якому передається токен за допомогою одноіменного методу. Надалі робота з

конкретним ботом, виконується завдяки локальному об’єкту, як зазначено на рисунку 3.4.

Після реєстрації бота, можна використовувати його в особистих повідомленнях (це його звичайне назначення), або в налаштуваннях об’єкту Телебот можна використати токен певної переписки. Для прикладу є певне підприємство, на якому розкладена багатогілкова мережа. Основним маршрутизатором для серверів є MikroTik світч. Він дозволяє прописати хуки, завдяки яким бот сповіщає в групі системних адміністраторів, про несправності на лінії, та тип несправності (проблеми з електроенергією, або проста втрата сигналу).

Далі можна кастомізувати бота так, як хочеться розробнику. Для цього потрібно скористуватися іншими командами BotFather. Але в цілому косметичні зміни слід проводити вже після того як бот функціонує так, як було заплановано в технічному завданні.

Рисунок 3.5 – Кастомізація бота

3.3 Розгортання серверу Apache

У проекті, де передбачається обробка і зберігання даних користувачів, одним із найкращих рішень для їх структуризації та зберігання є використання реляційної бази даних. Для цього найбільш поширеним варіантом є застосування MySQL, що дозволяє зручно зберігати, організовувати і отримувати дані за допомогою SQL-запитів. Хоча MySQL можна встановити окремо через стандартні інструменти, для спрощення процесу розгортання та інтеграції веб-сервера з базою даних можна використовувати готову збірку, таку як OpenServerPanel.

OpenServerPanel є безкоштовною утилітою, яка дозволяє розгорнути локальний веб-сервер на вашому комп’ютері з мінімальними зусиллями. Це ідеальне рішення для розробників, які працюють із API (наприклад, API Телеграму) або тестують локальні веб-застосунки, такі як сайти чи сервери. Однією з ключових переваг OpenServerPanel є його зручний інтерфейс, який дозволяє швидко налаштувати веб-сервер Apache, встановити базу даних, а також налаштувати всі необхідні компоненти для розробки та тестування програмного забезпечення.

Для взаємодії з базою даних OpenServerPanel передбачає встановлені інтерфейси PHPMyAdmin та SQLAdminer. Ці інтерфейси значно полегшують адміністрування баз даних, оскільки дозволяють легко виконувати SQL-запити, керувати таблицями, редагувати дані та налаштовувати структуру бази даних через зручний веб-інтерфейс. Це особливо корисно для адміністраторів, розробників або модераторів, адже вони можуть зекономити час на виконанні рутинних операцій з базою даних.

Перш за все треба скачати цю утиліту з офіціального сайту ospanel.io. На превеликий жаль, команда розробників цієї утиліти не надає доступу до її попередніх версій, а актуальна версія на наш час це – 6.0.0. Вона є тестовою, дуже незручною, було прибрано більшість інтерфейсу, та перенесено його до платної версії. В безкоштовній версії залишилася командне вікно, та метод

налаштування «руками». Але, так як у студента є досвід роботи з цією утилітою, то для проекту буде використана версія 5.4.3, яка лежала в одному з його проектів. Вона все ще надає можливість адміністрування локального серверу за допомогою зручного та зрозумілого інтерфейсу, та надає в користування безліч додаткових утиліт (phpMyAdmin, SQLAdminer, SublimeText тощо).

Наступним кроком буде встановлення утиліти на комп'ютер та її запуск. В нововстановленій версії не підключені бази даних, тому їх треба запустити «руками» та перезапустити веб-сервер.

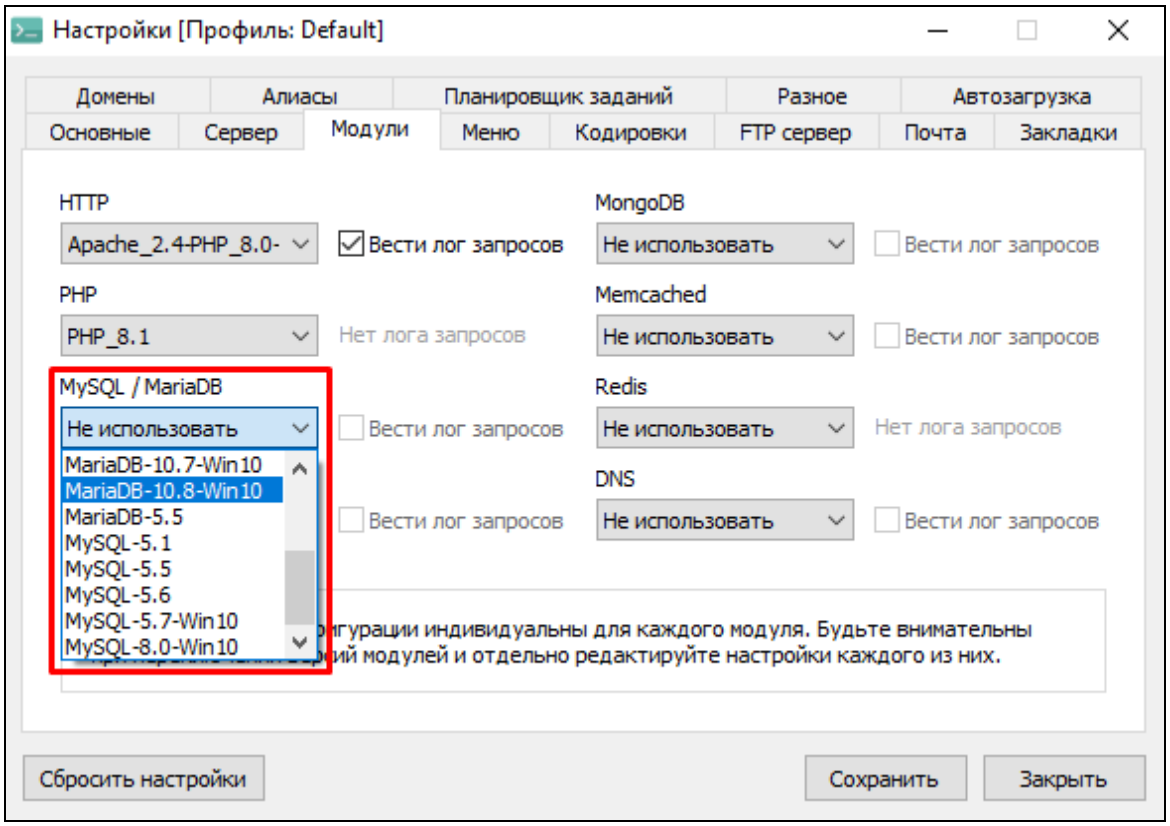

Рисунок 3.6 – Налаштування веб-серверу

Після того як все було налаштовано, можна через іконку прапора на панелі задач запустити веб-сервер. Відтепер можна користуватися базою даних. Для роботи буде створено базу даних library

3.4 Створення бази даних MariaDB

Обов’язковою частиною проекту є база даних, на движку MariaDB 10.8. Саме завдяки базі даних влаштовується ієрархія файлів книг, та пов’язаних з ними тем. Базу даних можна розділити на дві частини:

- дані про книги, які ще не були промодеровані;
- дані про книги, які вже були підтверджені модератором.

За логікою проекту, якщо книга не була підтверджена уповноваженою людино, тоді користувачі не мають можливості знайти її в загальному доступі. Зроблено це задля протидії можливих «жартівників» які захочуть завантажити книги з непристойною або шокуючою інформацією до бібліотеки.

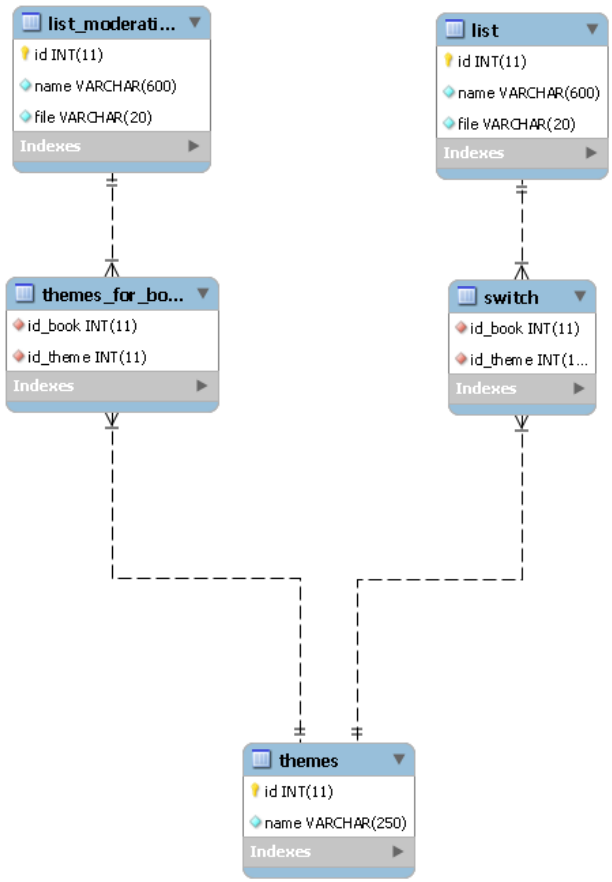

Рисунок 3.7 – Структура бази даних

Якщо звернути увагу на рисунок 3.7, то можна побачити що є дві

«гілки» бази даних. Одна для внесення даних які повинні пройти модерацію, а інша для даних, які можна показати іншому. Таблиці themes_for_books та switch створені задля дотримання правил нормалізації та розподілення тем для певних книг. Інші таблиці є «довідниками» для зберігання масивів інформації.

3.5 Створення функціоналу чат боту

3.5.1 Першочергові налаштування

Як було зазначено в розділі 3.1, логічно програму можна розділити на три частини. Тобто на:

- занесення в базу даних нової книги користувачем;
- пошук користувачем книги за назвою;
- пошук користувачем книги за темою.

Саме ці блоки програмування будуть створені почергово. Але першою задачею буде налаштування потрібних бібліотек. Для роботи проекту потребується дві бібліотеки, які потрібно завантажити. Так як середовище розробки PyCharm надає віртуальне оточення (папка .venv у кожному проекті), то можна скористатися вбудованою системою управління пакетами pip.

В терміналі необхідно виконати наступні команди:

```
#для встановлення pytelebot
pip install pyTelegramBotAPI
#для встановлення mysql-connector-python
pip install mysql-connector-python
```

В папці проекту також потрібно створити файл db_config.py який буде використовуватися в проекті для встановлення зв’язку з базою даних

проекту.

Лістинг 3.1 – Конфігураційний файл для використання БД

```
 db_config={
    'user':                                                  'root',
    'password':                                                 '',
    'host':                                                '127.0.0.1',
    'database':                                              'library',
    'raise_on_warnings':                                          True
}
```

Після цього можна взятися за розробку основного коду.

3.5.2 Завантаження своєї книги

Завантаження книги користувачем є першою функцією яку було реалізовано. Тому для початку потрібно ініціалізувати стартові команди боту. Для цього прописується команда на старт боту.

Лістинг 3.2 – Підключення БД та обробка стартової команди

```
 if not os.path.exists('books'):
    os.makedirs('books')
try:
    connection = mysql.connector.connect(**db_config)
    print("Підключення успішне!")
except mysql.connector.Error as err:
    print(f"Помилка: {err}")
bot =
telebot.TeleBot('7815794346:AAHE6j6w7Lxfl8ya_2l33LCDhq1tDLuZhww'
)
 @bot.message_handler(commands=['start'])
def start(message):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton("□ Вітання")
    markup.add(btn1)
    bot.send_message(message.from_user.id, "□ Доброго дня! Я
бібліотекар! Чим можу допомогти?", reply_markup=markup)
```

В цьому старті йде перевірка на папку books, де будуть збережені книги. Якщо такої папки немає – то вона створюється в директорії проекту. Далі йде спроба підключення до бази даних та закріплення токену бота для

зв’язку. Наступним кроком є обробка команди /start.

Після всього цього починається розробка вищезазначеного функціоналу. Якщо описувати в цілому код то кардинально нового, або цікавого нічого немає. Використовування об’єктів telebot та types, які надаються бібліотекою telebot, умовних операторів if/else які виконують функції створення «гілок» вибору користувача. Але окремо можна розглядати функції, які виконуються в тому чи іншому випадку. Для прикладу, в завантаженні файлу користувачем використовується функція handle_document(message) яка виконується в обробнику подій коли контент який відправляє користувач це документ.

Лістинг 3.3 – Обробка завантаження нової книги

```
     @bot.message_handler(content_types=['document'])
def handle_document(message):
    user_id = message.from_user.id

    if user_id in user_states and user_states[user_id] ==
'waiting_for_file':
        bot.send_message(user_id, 'Я завантажую Ваш файл,
очікуйте...')
        file_id = message.document.file_id
        file_info = bot.get_file(file_id)

        random_filename =
''.join(random.choices(string.ascii_letters + string.digits,
k=10))
        file_extension =
os.path.splitext(message.document.file_name)[1]
        new_file_name = f"{random_filename}{file_extension}"

        downloaded_file = bot.download_file(file_info.file_path)
        with open(os.path.join('books', new_file_name), 'wb') as
new_file:
            new_file.write(downloaded_file)

        book_title = user_data[user_id]['book_title']
        cursor = connection.cursor()
        try:
            # Вставка книги в таблицу `list_moderation`
            cursor.execute("INSERT INTO list_moderation (name,
file) VALUES (%s, %s)", (book_title, new_file_name))
            connection.commit()
            book_id = cursor.lastrowid
```

```
            # Запись тем для книги
            for theme in user_data[user_id]['book_themes']:
                theme = theme.strip()
                cursor.execute("SELECT id FROM themes WHERE name
= %s", (theme,))
                theme_id = cursor.fetchone()

                if theme_id:
                    theme_id = theme_id[0]
                else:
                    cursor.execute("INSERT INTO themes (name)
VALUES (%s)", (theme,))
                    connection.commit()
                    theme_id = cursor.lastrowid

                cursor.execute("INSERT INTO themes_for_book
(id_book, id_theme) VALUES (%s, %s)", (book_id, theme_id))
                connection.commit()

            # Уведомление модератора
            send_moderator_notification(book_id, book_title,
new_file_name)

            bot.send_message(user_id, 'Дякуємо за Ваш внесок в
бібліотеку. Книга стане доступною для інших після її перевірки
модератором.')
            user_states[user_id] = 'main_menu'
            go_to_main_menu(user_id)
        except mysql.connector.Error as err:
            bot.send_message(user_id, f'Сталася помилка при
збереженні книги: {err}')
            go_to_main_menu(user_id)
```

В ній програма отримує файл, його назву, яка була введена користувачем, формат файлу. Після цього програма генерує рандомну назву з 10 символів для файлу, завантажує файл користувача, записує дані до новозгенерованого файлу та намагається вставити дані про книгу в базу даних. Якщо помилок не виникло при спробі завантаження книги, тоді бот повідомить модератора про те, що в списку книг на модерацію додано нову книгу, а користувачу буде показано повідомлення про те що книгу завантажено, і тепер вона очікує підтвердження модератором боту. Якщо виникає якась помилка, тоді користувачу буде виведено повідомлення з текстом помилки.

3.5.3 Пошук книг за назвою

В основному обробнику подій @bot.message_handler(content_types=['text']) задається «гілка» розвитку пошуку за назвою для користувача. Цей обробник генерує кнопки доступних команд, а також забезпечує роботу хуків, які дозволяють боту виконувати відповідні задачі. Зокрема, в цьому сценарії підключено функції пошуку книг у базі даних та обробки вибору користувачем знайденої книги.

Пошук книг здійснюється за допомогою SQL-запиту, що використовує конструкцію LIKE для порівняння назви книги із запитом користувача. Результати пошуку виводяться у вигляді нумерованого списку, з якого користувач може вибрати потрібну книгу.

Коли користувач вводить номер книги, функція перевіряє його коректність, виконує відповідний SQL-запит для отримання деталей книги та надсилає файл книги у відповідь. У разі некоректного введення номера або відсутності книги з таким номером бот виводить відповідне повідомлення з проханням повторити спробу.

Також передбачено функцію для автоматичного повернення користувача до головного меню після отримання книги. Це забезпечує зручність взаємодії з ботом та зменшує кількість кроків, необхідних для повторного використання його функцій.

Лістінг 3.4 – Функції пошуку книг

```
 def search_books(query, user_id):
    cursor = connection.cursor()
    search_query = f"%{query}%"
    cursor.execute("SELECT id, name FROM list WHERE name LIKE 
%s", (search_query,))
    results = cursor.fetchall()

    if results:
        books_list = "\n".join([f"{book[0]}. {book[1]}" for book
```

```
in results])
        bot.send_message(user_id, f"Знайдено
книги:\n{books_list}\n\nВведіть номер книги, щоб вибрати її.")
    else:
        bot.send_message(user_id, "На жаль, книги з таким
запитом не знайдено.")

def handle_book_selection(message, user_id):
    try:
        book_id = int(message.text.strip())
        cursor = connection.cursor()
        cursor.execute("SELECT name, file FROM list WHERE id =
%s", (book_id,))
        book = cursor.fetchone()

        if book:
            book_name, file_name = book
            with open(os.path.join('books', file_name), 'rb') as
book_file:
                bot.send_document(user_id, book_file,
caption=f" {book_name}")
            go_to_main_menu(user_id)
        else:
            bot.send_message(user_id, "Книга з таким номером не
знайдена. Спробуйте ще раз.")
    except ValueError:
        bot.send_message(user_id, "Введіть дійсний номер
книги.")
```

3.5.4 Пошук книги за темою

Як і в минулих кроках в основному тілі програми треба створити інтерфейс для пошуку книги за темою. На думку студента основні три функції які описані вище будуть повністю задовольняти першочергові потреби користувачів. Цікавою частиною цього функціоналу є функція описана в лістингу 3.5. В запиті використовується конструкція SQL JOIN…ON, щоб вивести назву теми, так як назви тем зберігаються в окремій таблиці. Зроблено це задля збереження основних правил нормалізації бази даних, а саме для недопущення повторних записів.

Лістинг 3.5 – Функція пошуку книги за темою

```
     def show_books_by_theme(theme_name, user_id):
    cursor = connection.cursor()
    cursor.execute("""
        SELECT b.id, b.name
        FROM list b
        JOIN switch t_fb ON b.id = t_fb.id_book
        JOIN themes t ON t_fb.id_theme = t.id
        WHERE t.name = %s
    """, (theme_name,))
    results = cursor.fetchall()

    if results:
        books_list = "\n".join([f"{book[0]}. {book[1]}" for book
in results])
        bot.send_message(user_id, f"Книги за темою
'{theme_name}':\n{books_list}\n\nВведіть номер книги, щоб
вибрати її.")
        update_state(user_id, 'books_by_theme')
    else:
        bot.send_message(user_id, f"На жаль, книг за темою
'{theme_name}' не знайдено.")
```

3.5.5 Модерація введених даних

Останньою описаною функцією є модерація введених користувачами даних. Основною ціллю цієї дії є недопущення завантаження до бази даних кіберуніверситету нецензурного, недоречного, або будь-якого іншого контенту, який може бути шокуючим для користувача. Після того як користувач завантажує книгу, вона відправляється модератору, в якого є можливість відхилити дану книгу, або підтвердити. Цей процес можна побачити в лістингу 3.3. Саме для цього створено таблицю list_moderation та таблицю list. В першу потрапляють книги які були завантажені користувачем, а в другу – книги які були відфільтровані модератором та потраплять в основну таблицю, яка вже буде доступна всім користувачам кіберуніверситету.

Лістинг 3.6 – Модерування нових книг

```
     def send_moderator_notification(book_id, book_title,
file_name):
```

```
    markup = types.InlineKeyboardMarkup()
    btn_confirm = types.InlineKeyboardButton('✅Підтвердити',
callback_data=f'confirm_{book_id}')
    btn_reject = types.InlineKeyboardButton('❌Відхилити',
callback_data=f'reject_{book_id}')
    markup.add(btn_confirm, btn_reject)
    with open(os.path.join('books', file_name), 'rb') as
book_file:
        bot.send_document(MODERATOR_ID, book_file, caption=f"ID:
{book_id}\nКнига: {book_title}", reply_markup=markup)


@bot.callback_query_handler(func=lambda call: True)
def handle_moderator_actions(call):
    cursor = connection.cursor()
    action, book_id = call.data.split('_')
    try:
        cursor.execute("SELECT name, file FROM list_moderation
WHERE id = %s", (book_id,))
        book = cursor.fetchone()
        if not book:
            bot.answer_callback_query(call.id, "Книга не
знайдена.")
            return
        if action == 'confirm':
            cursor.execute("INSERT INTO list (name, file) VALUES
(%s, %s)", (book[0], book[1]))
            connection.commit()
            confirmed_book_id = cursor.lastrowid
            # Перенесення зв'язків тем в таблицю `switch`
            cursor.execute("SELECT id_theme FROM themes_for_book
WHERE id_book = %s", (book_id,))
            themes = cursor.fetchall()
            for theme in themes:
                cursor.execute("INSERT INTO switch (id_book,
id_theme) VALUES (%s, %s)", (confirmed_book_id, theme[0]))
                connection.commit()
            # Видалення книги з таблиці `list_moderation`
            cursor.execute("DELETE FROM list_moderation WHERE id
= %s", (book_id,))
            connection.commit()
            bot.answer_callback_query(call.id, "Книга
підтверджена і додана до бібліотеки.")


        elif action == 'reject':
            # Видаляємо файл книги
            os.remove(os.path.join('books', book[1]))
            # Отримуємо всі темы, пов'язані з книгою
            cursor.execute("SELECT id_theme FROM themes_for_book
WHERE id_book = %s", (book_id,))
            themes = cursor.fetchall()
            # Видаляємо зв'язки книги з темами
```

```
            cursor.execute("DELETE FROM themes_for_book WHERE
id_book = %s", (book_id,))
            connection.commit()
            # Перевірка на потрібність видалення теми
            for theme in themes:
                theme_id = theme[0]
                cursor.execute("SELECT COUNT(*) FROM
themes_for_book WHERE id_theme = %s", (theme_id,))
                theme_count = cursor.fetchone()[0]
                if theme_count == 0:
                    # Якщо тема не пов'язана з книгами -
видаляємо
                    cursor.execute("DELETE FROM themes WHERE id
= %s", (theme_id,))
                    connection.commit()
            # Видаляємо книгу з таблиці `list_moderation`
            cursor.execute("DELETE FROM list_moderation WHERE id
= %s", (book_id,))
            connection.commit()
            bot.answer_callback_query(call.id, "Книга відхилена
і видалена.")
    except mysql.connector.Error as err:
        bot.answer_callback_query(call.id, f"Помилка: {err}")
```

3.6 Створення віртуальної машини для розгортання програмного продукту

Останнім кроком в розробці даного проекту є створення віртуальної машини для розгортання програмного продукту на ній. Переваги цього способу були перелічені в пункті 2.4, тому розповідати тут про нього не є доречним. Спочатку потрібно включити систему віртуалізації Hyper-V. На робочій машині студента всі потрібні налаштування виконані. Але часто буває проблема з деякими моделями процесорів AMD Ryzen в тому, що на них немає підтримки віртуалізації. Якщо така проблема виникла, то на цій робочій машині неможливо розгорнути систему віртуалізації.

Повертаючись до робочої машини студента, потрібно включити Hyper-V одним з двох методів. Можна виконати команду DISM/Online/Enable-Feature/All /FeatureName:Microsoft-Hyper-V в PowerShell (від адміністратора), або прослідувати за наступною інструкцією:

а) перейти в панель управління;

б) перейти в розділ «Програми», а потім в «Програми і компоненти»;

в) обрати «Включити або відключити компоненти Windows»;

г) обрати опцію «Hyper-V» та натиснути «ОК»;

д) перезавантажити ПК.

Після виконання цих дій в меню «Пуск» в розробника в папці «Засоби адміністрування Windows» з’явиться значок Hyper-V.

Наступним етапом є створення віртуального комутатора. Він знадобиться під час створення віртуальної машини.

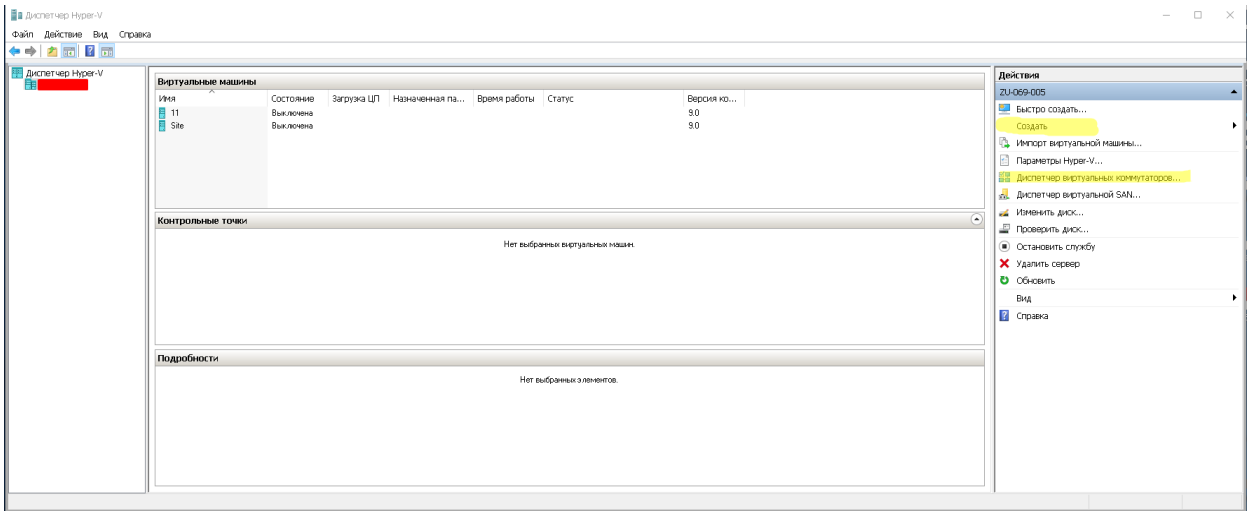

Рисунок 3.8 – Інтерфейс програми Hyper-V

Через диспетчер віртуальних комутаторів потрібно створити приватну мережу. Після цього через кнопку «Створити» створюється віртуальний жорсткий диск, який буде використано в віртуальній машині.

Після всіх приготувань можна створювати віртуальну машину. Процес створення інтуїтивно зрозумілий, і потрібно лише підключати раніше створені елементи. Відтепер розробник може вільно використовувати всі можливості ВМ.

Наступний крок – це завантаження актуальної версії Windows. Можна не встановлювати серверну версію, так як в проекті використовується

OpenServerPanel, що дозволяє просто розвернути локальний домен задля роботи в інтернеті. Обов’язковою умовою є формат, в якому буде файл встановлення системи. Для гіпервізора Hyper-V це файли формату .iso.

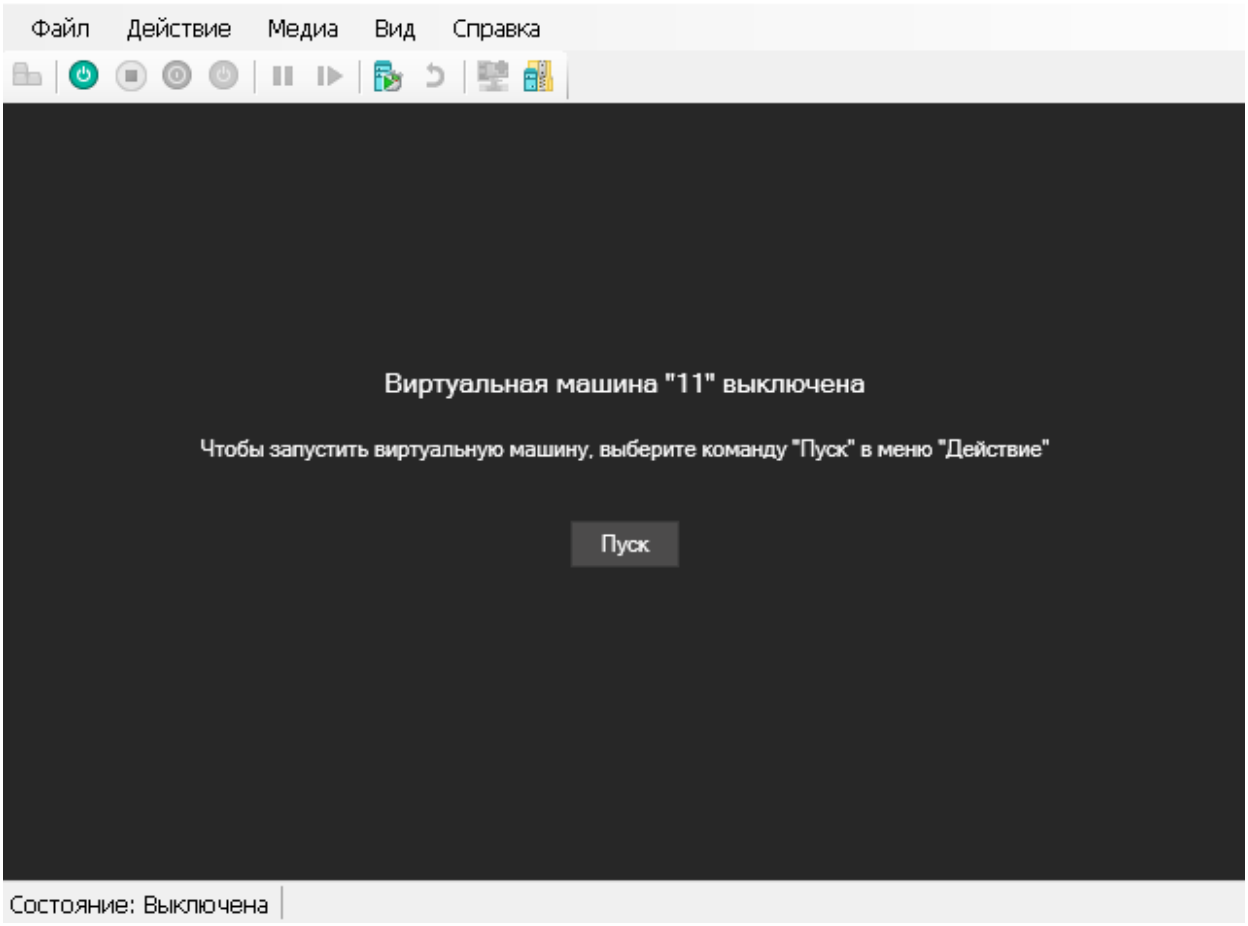

Рисунок 3.9 – Використання активної віртуальної машини

Через Медіа – DVD-Дисковод – Вставити диск можна обрати ISO файл з встановленням системи Windows на комп’ютері. Після цього потрібно встановити Windows. Процес встановлення відомий кожному, тому його не буде описано в цьому проекті. Обов’язково потрібно активувати систему задля завчасної ліквідації можливих проблем.

Коли система готова до використання потрібно встановити PyCharm, OSPanel, налаштувати все під проект і система буде готова для використання. Якщо розробнику потрібно щось занести на віртуальну машину, то можна

використовувати вбудований FTP сервер в OSPanel (на випадок якщо віртуальна машина працює), або вимкнути віртуальну машину, «розвернути» віртуальний диск подвійним кліком на файлі віртуального диску, скопіювати потрібні файли на нього, після чого правою кнопкою миші на диску в меню «Мій комп’ютер» обрати варіант «Вилучити».

4 ТЕСТУВАННЯ ПРОЕКТУ

Першою функцією, яка буде протестована є завантаження книги користувачем. На думку студента, цьому етапу потрібно приділити більше уваги, тому що одним з найголовніших правил безпеки є «найуразливіше місце програми – це момент введення користувачем даних».

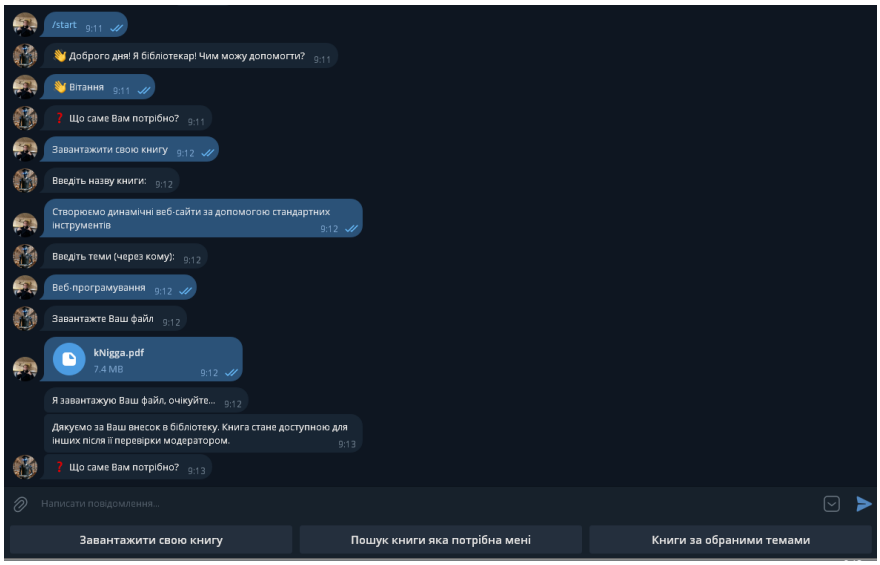

Рисунок 4.1 – Тестування завантаження книги

Після того як користувач відправляє книгу до бібліотеки, модератор бачить повідомлення в своєму чаті від бота та має можливість переглянути матеріал, назву і або відхилити надіслану студентом книгу, або її підтвердити. Після натискання на кнопку буде показано повідомлення з дією, яку обрав модератор. Модератори прописуються вручну в коді програми. Це зроблено через те, що продукт знаходиться в стадії раннього релізу, тому надалі можна використовувати базу даних для збільшення кількості модераторів, або використання спільного чату модераторів.

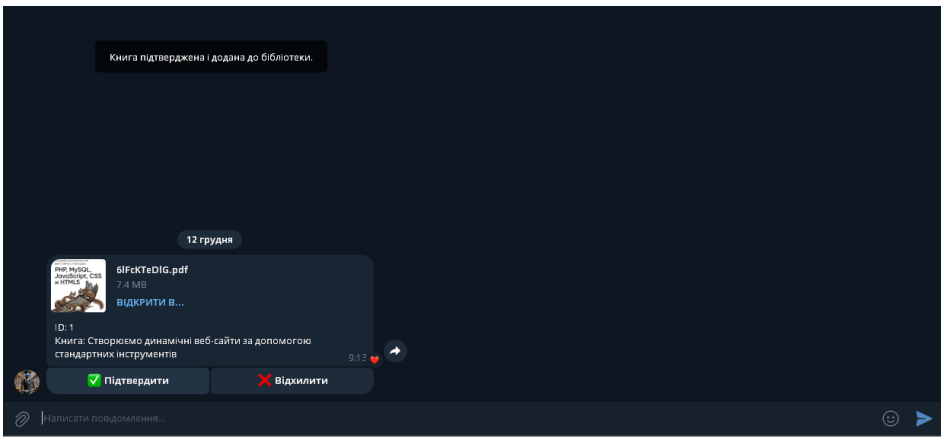

Рисунок 4.2 – Підтвердження книги модератором

Наступна перевірка – пошук книги за назвою. Користувач відправляє запит до боту з частиною назви книги, а бот виконує звичайний SELECT з використанням інструкції LIKE. Тому в запиті будуть показуватися всі книги де фігурує слово, або частина слова з запиту. Якщо книг з співпадіннями не буде, то бот повідомить користувача про це. Якщо результат був позитивним, то бот надішле список знайдених результатів, та запропонує обрати користувачу номер книги в цьому списку. Після вибору номера – бот надсилає файл з бази даних.

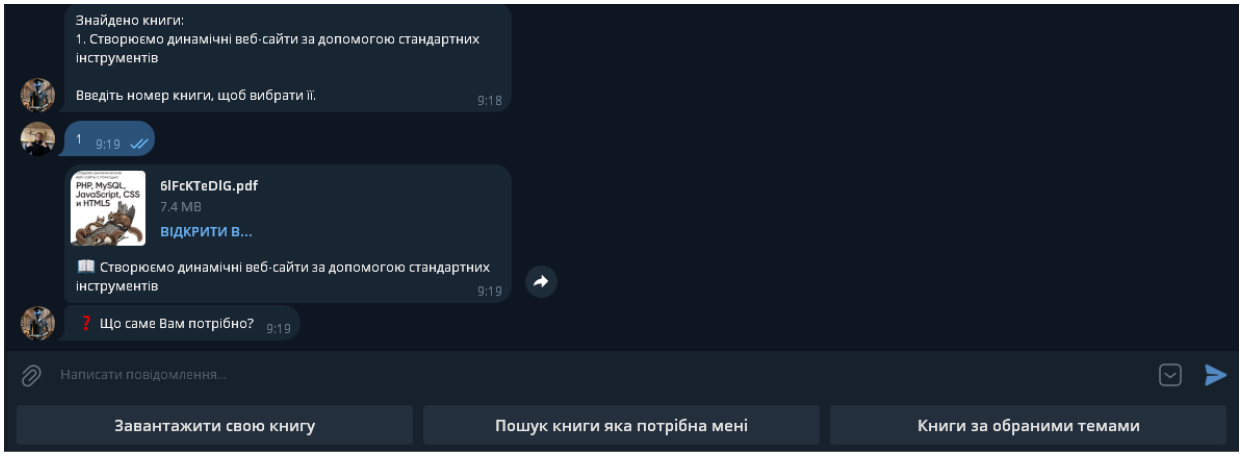

Рисунок 4.3 – Перевірка пошуку за назвою книги

Наступним і останнім кроком перевірки є перевірка пошуку за темами. Часто буває таке що студент не знає яку книгу він хоче отримати, але знає, за якою темою ця книга повинна бути. Бот надсилає користувачу список доступних тем, за якими вже є матеріал. Після цього користувач обирає тему, бачить список книг з номерами, і так як і в пошуку за назвою – обирає номер потрібної книги, після чого бот відправляє файл з бази даних.

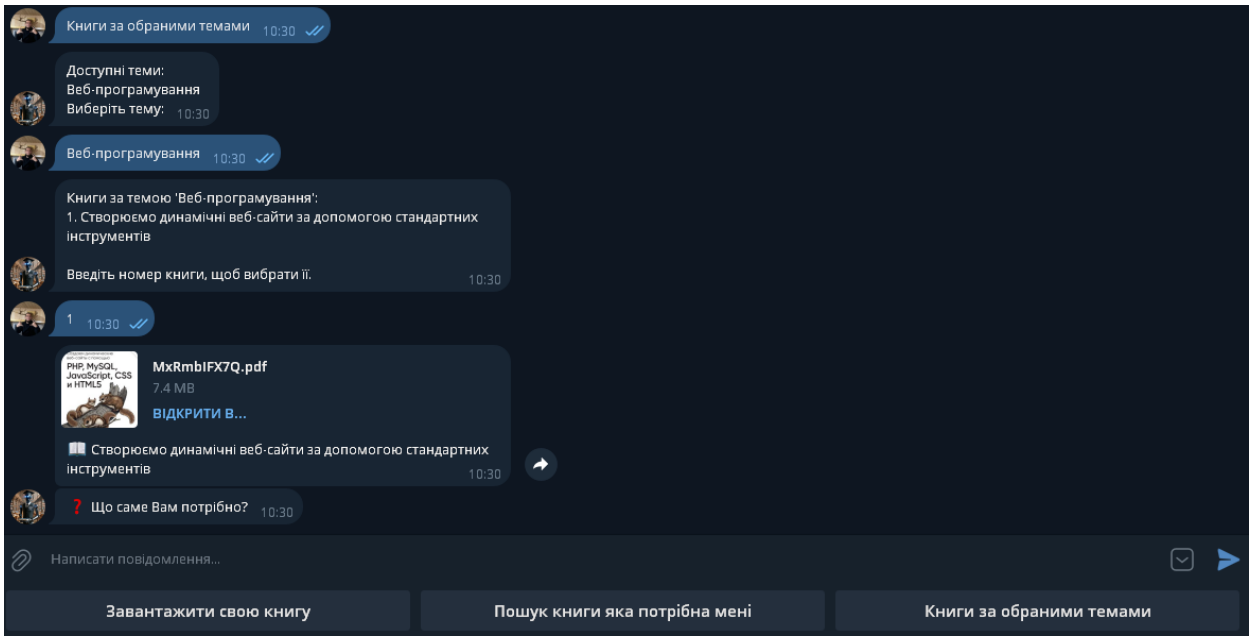

Рисунок 4.4 – Перевірка пошуку книги за темою

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було досліджено та розроблено технології використання Telegram чат-ботів як частини кіберсоціальної системи розумний університет, а також виявлено їх переваги та недоліки. Результатом дослідження студентом було виявлено, що Telegram-боти є надзвичайно корисним інструментом для автоматизації різноманітних процесів в закладах освіти, забезпечуючи ефективність, економії ресурсів та оптимізацію обміном інформації.

Дослідження показало, що однією з головних переваг Telegram-ботів є їхня кросплатформенність що дозволяє використовувати ботів на різних пристроях без необхідності додаткової адаптації продукту, що значно знижує витрати на розробку та тестування. Швидкість розробки також є важливим аспектом, оскільки створення чат-ботів займає значно менше часу порівняно з традиційними веб-застосунками чи мобільними додатками. Це забезпечує швидке впровадження нових функцій і модифікацій, що є важливим для бізнесу в умовах швидких змін ринкового середовища.

Telegram-боти сприяють автоматизації багатьох процесів. Наприклад, вони можуть обробляти запити студентів, надавати інформацію та виконувати інші завдання без участі модератора. Це дозволяє скоротити потребу в людських ресурсах, що важливо для навчальних закладів, які прагнуть оптимізувати свої кадрові витрати. Крім того, Telegram має інтуїтивно зрозумілий інтерфейс, завдяки чому користувачі легко адаптуються до роботи з ботами, що підвищує ефективність взаємодії з клієнтами.

Однак дослідження також виявило певні обмеження. Зокрема, через залежність від API Telegram функціонал ботів є обмеженим, що ускладнює створення кастомізованих інтерфейсів або специфічних функцій. Це може бути проблемою для закладів освіти із «новітнім» баченням інтерфейсу або

необхідністю інтеграції складних процесів. Ще однією проблемою є складність реалізації безпеки проекту від SQL ін’єкцій, та занадто «замудруватою» системою перевірки введених даних, та виведенням помилок у разі якщо вони некоректні.

Загалом, Telegram-боти демонструють значний потенціал для автоматизації навчального процесу. Вони дозволяють закладам вищої освіти знижувати витрати, підвищувати ефективність взаємодії зі студентами та швидко впроваджувати нові рішення. Незважаючи на певні обмеження, Telegram-боти є перспективним і економічно вигідним інструментом, який може стати важливою частиною будь-якого університету.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бенгфорт, Б. Прикладной анализ текстовых данных на Python. Машинное обучение и создание приложений [Текст] / Б. Бенгфорт. – Санкт-Петербург : Вид-во Питер, 2019. – 320 с.

2. McKinney, W. Python for Data Analysis, 3rd Edition [Текст] / W. McKinney. – Reilly Media, Inc., 2022. – ISBN 9781098104030. – 564 с.

3. Любанович, Б. Простий Python. Сучасний стиль програмування [Текст] / Б. Любанович. – Санкт-Петербург : Вид-во Пітер, 2015. – 256 с.

4. Sweigart, A. Automate the boring stuff with Python. Practical Programming for Total Beginners [Текст] / A. Sweigart. – San Francisco : No Starch Press, 2019. – 504 с.

5. Luciano, R. Fluent Python, 2nd Edition [Текст] / R. Luciano. – Reilly Media, Inc., 2021. – ISBN 9781492056355. – 750 с.

6. Силен, Д. Основы Data Science и Big Data. Python и наука о данных [Текст] / Д. Силен. – Санкт-Петербург : Вид-во Питер, 2017. – 384 с.

7. Сергієнко, О. Основи роботи з віртуальними машинами на Python [Текст] / О. Сергієнко. – Харків : ХНУ, 2021. – 240 с.

8. Хаханов В. І., Литвинова Є. І., Чумаченко С. В., Міщенко О. С. Кіберсоціальна система – розумний кіберуніверситет // Радіоелектронні і комп’ютерні системи. – 2016. – №5 (79). – С. 187–194.