Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка інформаційної системи для адміністрування ТНТУ ім. І. Пулюя за допомогою телеграм-бота

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Віннічeнко О.А.
(підпис) (прізвище та ініціали)

Керівник _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Вінніченку Олександру Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка інформаційної системи для адміністрування ТНТУ ім. І. Пулюя за допомогою телеграм-бота

Керівник роботи Шимчук Григорій Валерійович, Старший викладач кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 29 » квітня 2024 року № 4/7-470

2. Термін подання студентом завершеної роботи 24 червня 2024р.

3. Вихідні дані до роботи Літературні та інтернет джерела щодо розроблення телеграм-ботів для взаємодії з користувачами

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Постановка задачі кваліфікаційної роботи 1.1 Аналіз проблематики існуючого рішення. 1.2 Формування вимог до телеграм-бота 1.3 Визначення акторів і їх ролей у телеграм-боті 1.4 Варіанти використання функціональності інтерфейсу. 1.5 Аналіз існуючого інструментарію. 1.6 Обґрунтування вибору технологій. 1.7 Висновок до першого розділу. 2. Проєктування програмного інтерфейсу. 2.1 Проєктування БД. 2.2 Проєктування взаємодії телеграм-бота з бд. 2.3 Проєктування процесу реєстрації користувачів та адміністратора. 2.4 Проєктування архітектури. 2.5 Огляд потенційних загроз. 2.6 Проєктування механізмів масштабування. 2.7 Висновок до другого розділу. 3. Тестування та експлуатація програмного інтерфейсу. 3.1 Реалізація схеми БД. 3.2 Валідування вхідних даних. 3.3 Тестування готового рішення. 3.4 Ієрархія телеграм-бота. 3.5 Висновок до третього розділу. 4.Безпека життєдіяльності, основи охорони праці. 4.1 Укриття людей в захисних спорудах та евакуаційні заходи в період загрози. 4.2 Аналіз впливу чинників на зорову працездатність людини. 4.3 Висновки до четвертого розділу. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульна сторінка. 2 Мета, цінність. 3 Актори телеграм-бота. 4. Діаграма варіантів використання для акторів "Нерареєстрований користувач" та "Зареєстро ваний користувач". 5 Діаграма варіантів використання для актора "Адміністатор". 6 Використані технології. 7 Схема сутностей бази даних. 8 Схема процесу обробки даних користувача та їх взаємодія з базою даних. 9 Процес реєстрації користувача та адміністратора. 10 Проєктування архітектури. 11 Реалізована схема бд. 12 Реалізований телеграм-бот. 13 Процес реєстрації. 14 Процес Отримання користувальницької інформації. 15 Процес оплати за інтернет-послуги, та передавання квитанції оплати. 16 Ієрархія проєкту телеграм-бота. 17 Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Сенчишин В. С., кандидат технічних наук, доцент кафедри МТ		

7. Дата видачі завдання 29.01.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	30.01.2024	*Виконано*
2.	Підбір джерел про створення телеграм-ботів	31.01.2024-03.02.2024	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	04.02.2024-06.02.2024	*Виконано*
4.	Виконання дослідження щодо існуючих інструментів	07.02.2024-11.02.2024	*Виконано*
	та технологій для створення телеграм-ботів		
5.	Оформлення розділу “Постановка задачі кваліфікацій-	03.03.2024-15.03.2024	*Виконано*
	ної роботи”		
6.	Оформлення розділу “Проєктування програмного ін-	01.04.2024-12.04.2024	*Виконано*
	терфейсу телеграм-боту tntu_internet”		
7.	Оформлення розділу “Тестування та експлуатація про-	13.04.2024-22.04.2024	*Виконано*
	грамного інтерфейсу телеграм-бота tntu_internet”		
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»		
9.	Виконання завдання до підрозділу «Основи хорони		
	праці»		
10.	Оформлення кваліфікаційної роботи		
11.	Нормоконтроль		
12.	Перевірка на плагіат		
13.	Попередній захист кваліфікаційної роботи		
14.	Захист кваліфікаційної роботи		

Студент _____ Вінніченко О.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка інформаційної системи для адміністрування ТНТУ ім. І. Пулюя за допомогою телеграм-бота // Кваліфікаційна робота освітнього рівня “Бакалавр” // Вінніченко Олександр Анатолійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СН-41 // Тернопіль, 2024 // С. 64, рис. – 16 , табл. – 4, презентація –17 , додат. – 8 , бібліогр. – 45.

Ключові слова: телеграм-бот, автоматизація, інтернет-з'єднання, управління мережею, кібербезпека, база даних, користувачі, транзакції.

Кваліфікаційна робота присвячена дослідженню розробки та впровадження телеграм-бота для автоматизованого управління інтернет-з'єднанням у гуртожитках ТНТУ ім. І. Пулюя.

В першому розділі кваліфікаційної роботи описано актуальність теми, мету та задачі дослідження. Висвітлено основні проблеми, пов'язані з управлінням інтернет-з'єднанням у навчальних закладах. Проаналізовано існуючі рішення та їх недоліки.

В другому розділі кваліфікаційної роботи розглянуто процес проєктування бази даних для телеграм-бота tntu_internet. Досліджено взаємодію телеграм-бота з базою даних та впровадження механізмів безпеки.

В третьому розділі кваліфікаційної роботи описано процес тестування та впровадження телеграм-бота. Проаналізовано результати тестування, включаючи взаємодію користувачів з ботом та ефективність його роботи.

Об’єкт дослідження: процеси автоматизованого управління інтернет-з'єднанням у гуртожитках університету.

Предмет дослідження: розробка та впровадження телеграм-бота для автоматизації управління мережею, включаючи базу даних, взаємодію з користувачами та забезпечення безпеки.

ANNOTATION

Development of the an Information System for the Administration of I.Puluy TNTU Utilizing a Telegram Bot // Qualification work of the educational level “Bachelor” // Vinnichenko Oleksandr Anatoliyovych // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2024 // P. 64, presentation. -17, tabl. - 4, chair. - 18, annexes. – 8, references - 45.

Keywords: telegram bot, automation, internet connection, network management, cybersecurity, database, users, transactions.

The qualification work is dedicated to the research of the development and implementation of a telegram bot for automated management of internet connections in the dormitories of TNTU named after I. Puluj.

In the first section of the qualification work, the relevance of the topic, the purpose, and the objectives of the research are described. The main problems related to the management of internet connections in educational institutions are highlighted. Existing solutions and their shortcomings are analyzed.

In the second section of the qualification work, the process of designing a database for the telegram bot tntu_internet is considered. The interaction of the telegram bot with the database and the implementation of security mechanisms.

In the third section of the qualification work, the process of testing and implementing the telegram bot is described. The results of the testing are analyzed, including the interaction of users with the bot and its operational efficiency.

Object of the research: processes of automated management of internet connections in the university dormitories.

Subject of the research: development and implementation of a telegram bot for automating network management, including the database, user interaction, and ensuring security.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) – інтерфейс прикладного програмування; набір інструментів та протоколів для створення програмного забезпечення.

ORM (Object-Relational Mapping) – об'єктно-реляційне відображення; технологія програмування, яка використовується для перетворення даних між несумісними типами систем зберігання даних.

SQL (Structured Query Language) – структурована мова запитів; мова програмування, призначена для управління та маніпуляції реляційними базами даних.

PostgreSQL – об'єктно-реляційна система управління базами даних (СУБД) з відкритим вихідним кодом.

Telegram API – набір інструментів та протоколів, які дозволяють розробникам інтегрувати свої додатки з платформою Telegram.

Docker – платформа для розробки, розгортання та запуску додатків у контейнерах.

Kubernetes (K8s) – система оркестрації контейнерів для автоматизації розгортання, масштабування та управління контейнеризованими додатками.

PgAdmin – графічний інтерфейс для управління базами даних PostgreSQL.

Telegram-bot – автоматизована програма, яка взаємодіє з користувачами через платформу Telegram.

SQLAlchemy – ORM бібліотека для Python, яка дозволяє працювати з реляційними базами даних.

OAuth – протокол відкритої авторизації, який дозволяє стороннім додаткам доступ до ресурсів користувача без передачі паролів.

ЗМІСТ

ВСТУП

Актуальність теми. У результаті безупинного та стрімкого зростання у сфері інформаційних технологій швидкісне та стабільне під’єднання до Інтернету є невіддільною частиною навчального процесу та щоденного життя студентів і працівників освітньої установи. Труднощі в управлінні та обслуговуванні мережевих ресурсів можуть стати причиною неабияких незручностей та спричинити зниження результативності. Отже розроблення систематичного вирішення без використання людських ресурсів має змогу покращити та стабілізувати управління Інтернет-з’єднання в освітніх установах, що є нагальним напрямком для проведення наукових досліджень у цій сфері.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня “Бакалавр” є збільшення результативності послуг з керуванням інтернет-з'єднання у гуртожитках ТНТУ ім. І. Пулюя використовуючи розробку та введення телеграм-бота tntu_internet. Для здійснення поставленої мети слід здійснити ряд завдань, зокрема:

- Провести аналіз даного дослідження у сфері машинального керування мережевими та інтернет-ресурсами в освітніх установах.
- Створити та розробити базу даних з метою збереження та обробки важливих відомостей про інтернет-з'єднання, користувачів та крайні транзакції.
- Розробити та вдосконалити архітектуру телеграм-бота, враховуючи зв’язки з базою даних та навколишніми сервісами.
- Залучити механізми надійності з метою захисту та збереження особистої інформації користувачів та уникнення можливих загроз.
- Втілити та провести перевірку головного функціоналу телеграм-бота, забезпечуючи комфорт та результативне його застосування.

Практичне значення одержаних результатів. Здобуті утилітарні навички будуть спрямовані на розробку якісного та корисного інструменту для створення машинального керування та покращення Інтернет-з'єднання у гуртожитках даної освітньої установи. Закладений та вдосконалений телеграм-

бот tntu_internet неабияк покращить та спростить процес реєстрації та керування користувачами, даючи змогу гарантувати ефективну підтримку у випадку виявлення технічних несправностей, а також збільшити рейтинг вдоволення серед працівників та студентської молоді. Супроводження згаданого вирішення неодмінно сприятиме збільшенню якості надання послуг, унайкращення витрат на обслуговування та збільшенню результативності користування ресурсів освітньої установи.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.1 Аналіз проблематики існуючого рішення

На сьогодні, студентська молодь та працівники Тернопільського національного технічного університету імені Івана Пулюя (ТНТУ ім. І. Пулюя) зіткнулися з питанням браку машинальних вирішень для своєчасного та безперебійного розв’язання проблем, з’єднаних з інтернет-з'єднанням та іншими технічними труднощами. У випадку появи проблем, користувачам слід самостійно шукати інформацію, звертатися по допомогу до адміністрації через електронну пошту або месенджери, що може спричинити довгі затримки у відповідях, особливо при значній кількості звернень [1].

Дана ситуація обтяжується тим, що наявні канали зв'язку не завжди можуть швидко опрацьовувати та систематизувати значну кількість запитів, що може спричинити затримки у забезпеченні допомоги та вирішенні несправностей. Окрім того, брак єдиної платформи для комунікації та керування запитами ускладнює процес спостереження та відстеження статусу вирішення проблем.

З огляду на обставини, метою даної кваліфікаційної роботи є розроблення автоматизованого рішення, яке здатне надавати швидкі та ефективні відповіді на запити користувачів. Згідно з рішенням, впроваджено розробку телеграм-бота, який надає змогу систематизувати проблеми, надавати швидкі та дієві відповіді та покращувати процес взаємодії з адміністрацією університету. Це забезпечить ефективне управління запитами, знизить час реакції на проблеми та збільшить загальний досвід взаємодії користувачів з технічною підтримкою вищого навчального закладу.

Телеграм-бот буде обладнаний зручним інтерфейсом з метою надсилання запитів та отримання відповідей, а також з можливістю перенаправлення детальних та складних питань до відповідних технічних спеціалістів. Це не лише покращить якість обслуговування, але й облегшить ресурси адміністрації для

роботи зі складнішими запитами, забезпечуючи ефективне використання часу та зусиль.

1.2 Формування вимог до телеграм-бота tntu_internet

Для високопродуктивної роботи телеграм-бота, що створюється для покращення адміністрування інтернет-з'єднання у ТНТУ ім. І. Пулюя, вкрай важливо враховувати низку ключових аспектів, що впливають на його корисність та зручність використання.

Насамперед, телеграм-бот повинен бути інтерактивним та здатним підтримувати корисний діалог з користувачами. Це сприяє не тільки швидкості відповідей, але й їх вагомість та релевантність до запитів користувачів. Ключовим аспектом є також забезпечення надійної авторизації та реєстрації користувачів, що дозволить персоналізувати взаємодію та збільшити безпеку особистих даних.

Слід докладно розглянути обробку запитів користувачів. Телеграм-бот повинен класифікувати запити за їх типами, надавати корисні та коректні відповіді або інструкції для розв'язання проблем, а також перенаправляти складніші запити та проблеми до відповідних спеціалістів.

Інтеграція телеграм-бота з базою даних є ключовим аспектом, що дозволить зберігати важливу інформацію щодо користувачів, їхні запити та статуси вирішення питань (див. додаток А). Це вплине на ефективний моніторинг та аналіз зв'язку з телеграм-ботом [2].

З метою забезпечення стабільної та корисної роботи телеграм-бота, слід передбачити механізми логування подій та спостереження його стану. Це дозволить першочергово та своєчасно виявляти та усувати можливі проблеми, а також оптимізувати роботу телеграм-бота.

З огляду на можливе зростання кількості користувачів, телеграм-бот має бути розроблений беручи до уваги можливості масштабування. Це сприятиме ефективній роботі телеграм-бота навіть при збільшенні навантаження.

Зрештою, критичним аспектом є надання конфіденційних даних користувачів та їх запитів. Це передбачає використання сучасних та надійних методів шифрування та аутентифікації, що допоможе уникнути витоку особистої інформації та забезпечити конфіденційність взаємодії з телеграм-ботом [3].

1.3 Визначення акторів і їх ролей у телеграм-боті tntu_internet

Для проєктованого телеграм-бота було визначено визнач наступних акторів наведених на рисунку 1.1:

- незареєстрований користувач;
- зареєстрований користувач;
- адміністратор.

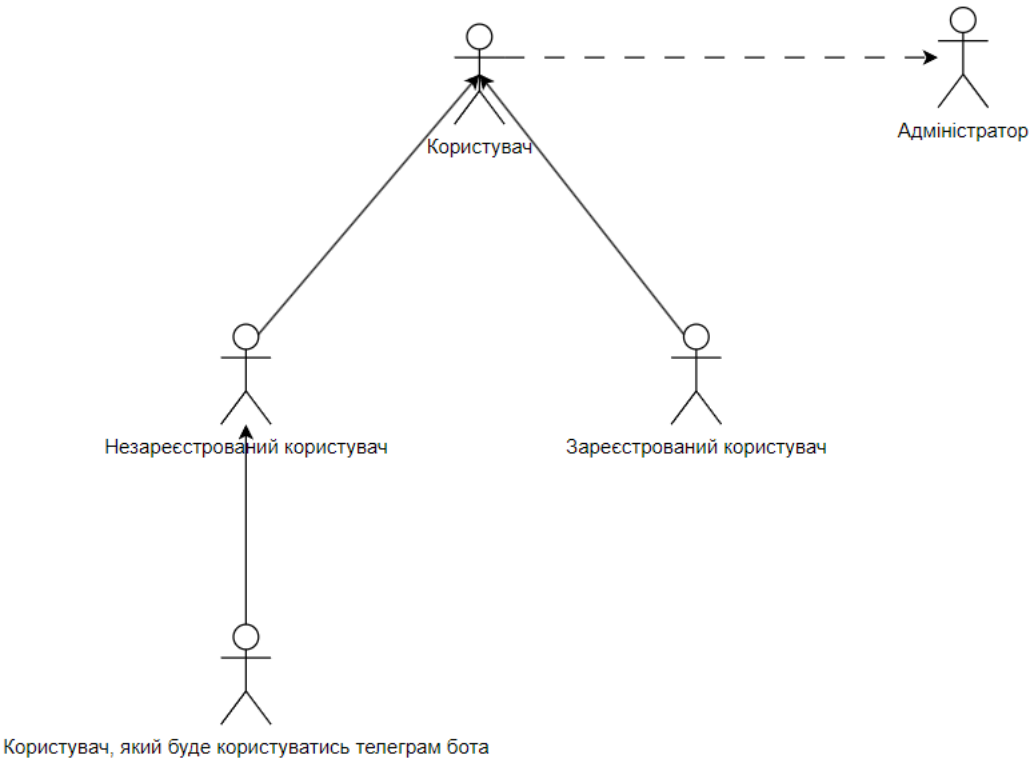

Рисунок 1.1 – Актори телеграм-бота tntu_internet

Було проведено аналіз акторів, представлених на рисунку 1.1, та їх ролі в середовищі телеграм-бота для покращеного адміністрування інтернет з'єднання ТНТУ ім. І. Пулюя.

Незареєстрований користувач: Особа, яка взаємодіє з телеграм-ботом, але не пройшла процес реєстрації, що наведена в таблиці 1.1. Ця група користувачів має закритий доступ до функцій телеграм-бота, який надає лише можливість реєстрації. Обмеження доступу незареєстрованих користувачів не тільки сприяє безпеці усієї системи, але й дозволяє підтримувати чітку структуру взаємодій.

Зареєстрований користувач: Студенти та співробітники вищого навчального закладу, які успішно пройшли реєстрацію в телеграм-боті, отримують і насолоджуються повним спектром можливостей системи. Від отримання коректної інформації до вирішення специфічних проблем – телеграм-бот стає важливим та корисним помічником у їхньому навчальному та робочому житті.

Адміністратор: Важливі фігури в управлінні телеграм-ботом, адміністратори відіграють ключову роль у забезпеченні значимості інформації, відповідності вмісту запитів користувачів та безперебійності роботи системи. Вони також відповідальні за додавання нових адміністраторів, що дозволяє розширити команду для ефективного управління зі зростаючою кількістю запитів.

Таблиця 1.1 – Реєстр варіантів використання для актора “Незареєстрований користувач”

Актор	**Варіант використання**	**Опис**
Незареєстрований користувач	Перехід до реєстрації	Надає акторові можливість перейти до форми реєстрації в телеграм-боті.

Способи використання для актора “Зареєстрований користувач” сформовано на основі способів використання для актора “Незареєстрований користувач” з додаванням нових, які подані в таблиці 1.2.

Таблиця 1.2 – Реєстр варіантів використання для актора “Зареєстрований користувач”

Актор	Варіант використання	Опис
Зареєстрований користувач	Використання телеграм-бота для отримання інформації	Надає акторові можливість взаємодіяти з телеграм-ботом для отримання необхідної інформації та вирішення проблем.
	Надсилання запитів	Дозволяє акторові надсилати запити телеграм-боту та отримувати оперативні відповіді.
	Отримання повідомлень	Дозволяє акторові отримувати повідомлення та оновлення від телеграм-бота.

В таблиці 1.3 подано перелік варіантів використання сформований для актора “Адміністратор”.

Таблиця 1.3 – Реєстр варіантів використання для актора “Адміністратор”

Актор	Варіант використання	Опис
1	**2**	**3**
Адміністратор	Управління контентом телеграм-бота	Надає акторові можливість додавати, оновлювати та видаляти інформаційний контент у телеграм-боті.
	Моніторинг та обробка запитів	Дозволяє акторові моніторити активність у телеграм-боті та ефективно обробляти запити користувачів.

Продовження таблиці 1.3

1	2	3
	Додавання нових адміністраторів	Дозволяє акторові розширювати команду управління, додаючи нових адміністраторів до системи.

Ці актори формують основу для створення телеграм-бота, забезпечуючи, щоб кожний зв’язок був спрямований на задовільність конкретних вимог. У наступних розділах буде докладніше розглянуто функціональність, яка буде доступна кожному з акторів.

1.4 Варіанти використання функціональності інтерфейсу телеграм-боту tntu_internet

На підставі зібраної інформації про вимоги та ролі користувачів, описаних у підрозділі “1.3 Визначення акторів і їх ролей телеграм-боту tntu_internet”, було створено діаграму способів використання для двох ключових акторів: незареєстрованого та зареєстрованого користувача.

На рисунку 1.2 показана детальна діаграма варіантів використання, яка визначає потенційні дії та шляхи взаємодії для “Незареєстрованого користувача” та “Зареєстрованого користувача” у середовищі телеграм-бота. Ця діаграма слугує візуальним представленням функціональності, доступної користувачам на різних рівнях взаємодії з телеграм-ботом, від початку реєстрації до виконання конкретних завдань, таких як редагування особистих даних чи перевірка статусу інтернет-з'єднання.

Діаграма відділяє можливості, які доступні незареєстрованим користувачам, від тих, що доступні лише після успішного проходження процедури реєстрації [4]. Прикладом може слугувати те, що будь-який користувач може звернутися до “Головного меню”, але лише після введення

своїх особистих даних і валідації може перейти до цікавих опцій, як-от “Редагування інформації” чи “Надсилання запиту”.

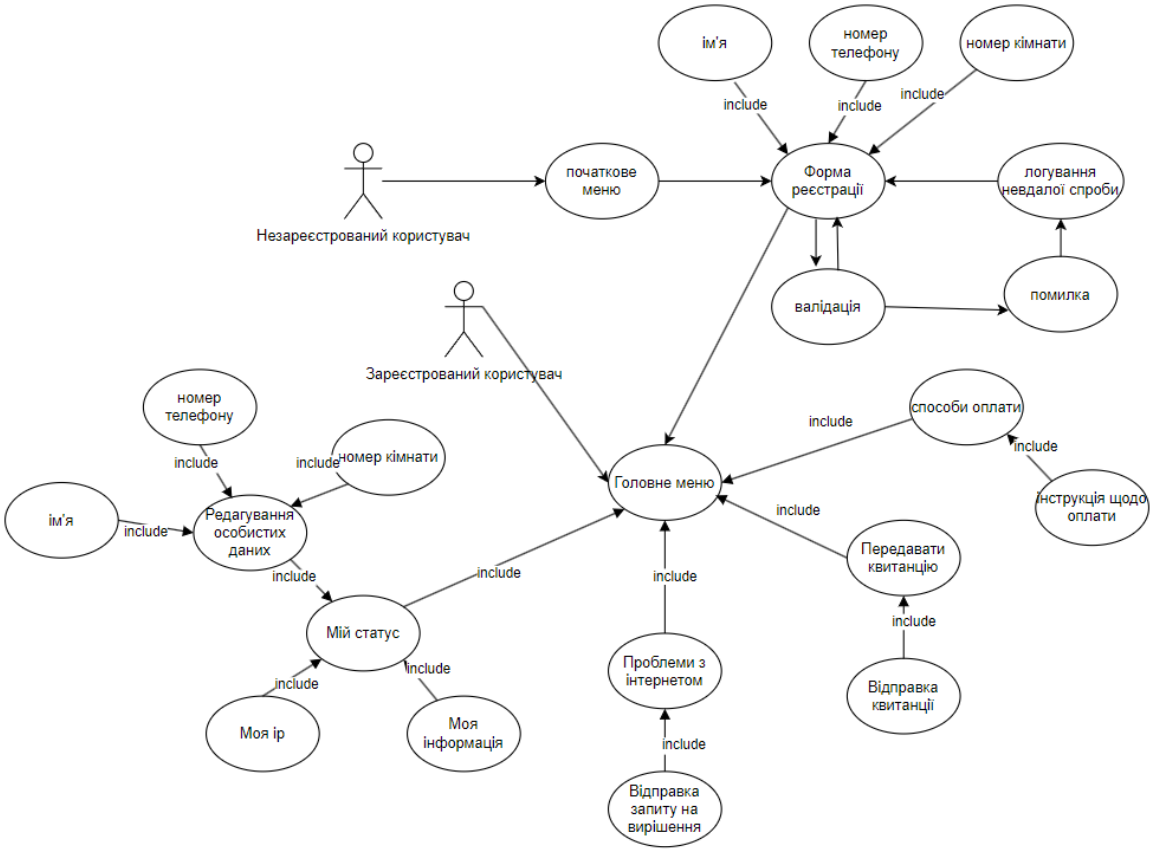

Рисунок 1.2 – Діаграма варіантів використання для акторів “Незареєстрований користувач” та “Зареєстрований користувач”

Для “Зареєстрованого користувача” діаграма докладно показує структурований підхід до взаємодії з телеграм-ботом, даючи змогу користувачу не лише виконувати основні операції, а й користуватися більш розширеними функціями, такі як, отримання допомоги з труднощами інтернет-з’єднання чи перевірка власної IP-адреси.

Дана діаграма виділяє модульність та пристосованість телеграм-бота до різних потреб користувачів і є основним інструментом для того, щоб зрозуміти, як користувачі взаємодіють з різними частинами системи, від простих запитів та питань до складних операцій [5].

Діаграма варіантів використання для актора “Адміністратор” описує функціонал, який стане у нагоді адміністратору для ефективного управління телеграм-ботом, що зображений на рисунку 1.3.

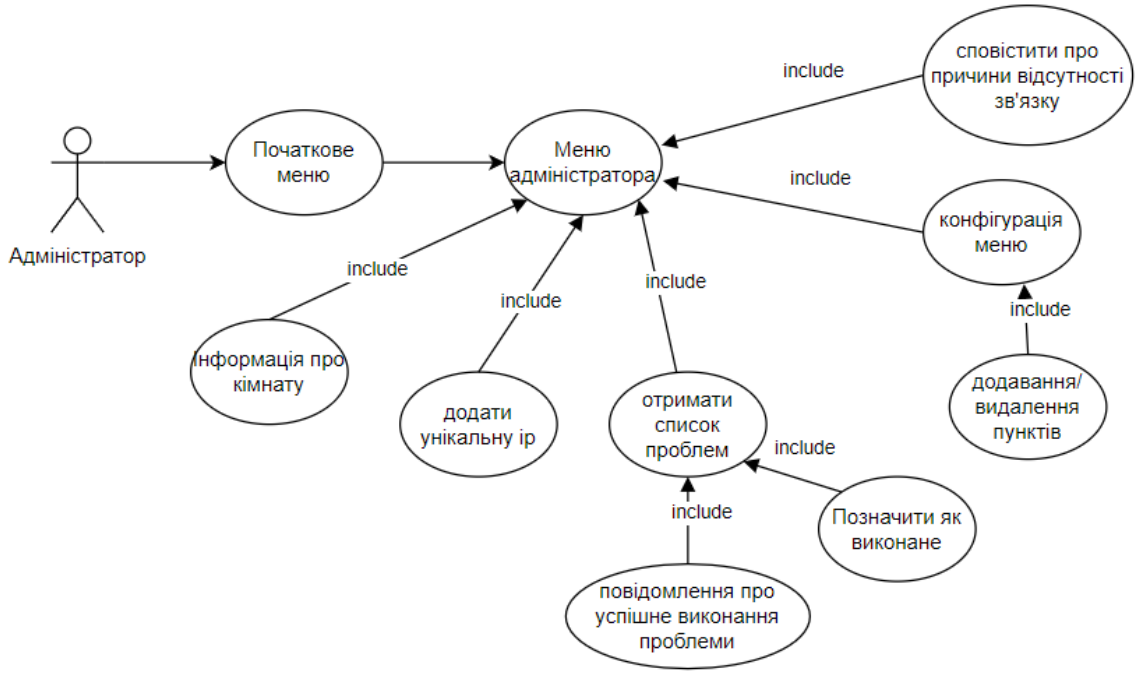

Рисунок 1.3 – Діаграма варіантів використання для актора “Адміністратор”

Увімкнені опції мають змогу забезпечити повний спектр можливостей для адміністратора: від оновлення нещодавньої інформації, управління списком проблем до додавання нових функцій у телеграм-бот [5].

1.5 Аналіз існуючого інструментарію для створення телеграм-боту tntu_internet

Під час вибору інструментарію для розроблення телеграм-бота важливо оцінити різні бібліотеки. Кожна бібліотека пропонує винятковий набір можливостей, який діє на підвищення розвитку проєкту, його постійність та легкість утримання у майбутньому [6].

При виборі мов програмування було сконцентровано увагу на Python та JavaScript, як два найбільш відповідних для роботи з Telegram API. Було розглянуто наступні бібліотеки:

- Python-telegram-bot для Python – відома своїм зрілим API і великою кількістю функціональностей.
- Aiogram для Python – вибір для асинхронної роботи, підтримує швидке виконання завдань.
- Node-telegram-bot-api для Node.js – популярна у JavaScript розробників, пропонує легку інтеграцію з іншими веб-сервісами.
- Telegraf для Node.js – сучасна і гнучка, з підтримкою мідлварів і нових версій JavaScript.
- Telepot для Python – хоча і не така популярна як інші, проте є хорошим вибором для простих ботів.

Для об'єктивного аналізу цих бібліотек було утворено таблицю порівняння, яка відображає їх важливі характеристики та спроможності, що зображено в таблиці 1.4.

Таблиця 1.4 – Порівняння бібліотек в розробці телеграм-бота tntu_internet

Бібліо-тека	**Спрощеність розробки**	**Швид-кість відгуку**	**Підтрим-ка спільноти**	**Масшта-бування**	**Асинхро-нність**
python-telegram-bot	Висока	Висока	Велика	Так	Ні
aiogram	Висока	Висока	Велика	Так	Так
node-telegram-bot-api	Висока	Висока	Велика	Так	Ні
telegraf	Висока	Висока	Велика	Так	Так
telepot	Середня	Середня	Середня	Так	Ні

На підставі цього порівняння було проаналізовано, що aiogram та telegraf пропонують неабиякі переваги в асинхронній обробці запитів, що є ключовим

для телеграм-ботів із великою кількістю користувачів. Водночас python-telegram-bot та node-telegram-bot-api залишаються надійними варіантами для широкого спектру проектів завдяки сильній підтримці спільноти і гнучкості в розробці [7].

1.6 Обґрунтування вибору технологій для розробки телеграм-бота tntu_internet

У даному розділі акцентовано увагу на обґрунтуванні вибору технологій для розробки телеграм-бота tntu_internet для ТНТУ ім. І. Пулюя.

Для виконання завдання кваліфікаційної роботи було обрано наступні засоби:

- Python 3.12.3;
- Aiogram 3.4.1;
- SQLAlchemy 2.0.29;
- Docker 23.0.3;
- Pydantic 2.7.0.

Подано їх короткі описи.

Python – це мова програмування Python була вибрана за її читабельність, широкий спектр застосування та велику підтримку спільноти. Python ідеально підходить для швидкої розробки та прототипування, і є однією з провідних мов у галузі розробки ботів.

Aiogram – це асинхронна бібліотека для створення ботів на Python, яка була обрана завдяки її високій продуктивності та гнучкості [8]. Aiogram надає надійний інтерфейс для взаємодії з Telegram API, що спрощує розробку швидких і відповідальних телеграм-ботів.

SQLAlchemy – це ORM інструмент, SQLAlchemy дозволяє дієво управляти базами даних без прямого використання SQL-запитів, що покращує роботу з даними та збільшує продуктивність розробника.

Docker – використання контейнерізації через Docker забезпечує легкість розгортання та портативність телеграм-бота. Це дозволяє розробити ізольоване середовище для бота, яке легко можна масштабувати та переміщати між серверами [9].

Pydantic – використовується для валідації даних та управління налаштуваннями бота через моделі даних. Pydantic забезпечує сильну типізацію та автоматичну валідацію, що збільшує безпеку та надійність введення особистих даних.

Комбінація цих технологій має в собі міцну основу для розроблення телеграм-бота, який буде не тільки дієвим та зручним у використанні для кінцевих користувачів, але й відповідатиме вимогам вищого навчального закладу щодо підтримки, безпеки та масштабування [10].

1.7 Висновок до першого розділу

Завершуючи перший розділ кваліфікаційної роботи, можна підсумувати, що відсутність автоматизованих рішень для надійного управління запитами та проблемами інтернет-з'єднань у ТНТУ ім. І. Пулюя суттєво обмежує дієвість та оперативність технічної підтримки. Супроводження телеграм-бота, як це було докладно розглянуто у цьому розділі, спрямовано на розв’язання цих проблем шляхом покращення процесу обробки запитів та оптимізації взаємодії між студентами, співробітниками та адміністрацією.

Діаграми варіантів використання, представлені у розділі, показують широкий спектр потенційних можливостей телеграм-бота, починаючи від початку реєстрації користувачів і закінчуючи цікавими функціями для різних ролей.

На підставі зібраних даних, аналітична робота, виконана у цьому розділі, закладає фундамент для наступного проєктування та розроблення телеграм-бота. Ключовим критерієм для вибору технологій послужили їх важливість, можливість швидкої розробки, масштабування та висока продуктивність. Вибір

Python та Aiogram забезпечить гнучкість та ефективність роботи телеграм-бота, водночас як Docker забезпечить легкість в розгортанні та масштабуванні проекту. У цілому, обрані технології і підходи до розробки телеграм-бота відповідають вимогам ТНТУ ім. І. Пулюя і можуть забезпечити високий рейтинг задоволеності серед користувачів.

Розробка телеграм-бота tntu_internet є не лише технічним втіленням вимог вищого навчального закладу до комунікації та технічної підтримки, а й відображає тенденції цифровізації в освітньому процесі, забезпечуючи взаємодію з інформаційними системами в зручний та дієвий спосіб.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ПРОГРАМНОГО ІНТЕРФЕЙСУ ТЕЛЕГРАМ-БОТУ TNTU_INTERNET

2.1 Проєктування БД для телеграм-боту tntu_internet

Створення та вдосконалення ефективної бази даних для телеграм-бота tntu_internet ТНТУ ім. І. Пулюя вимагала створення добре структурованої та масштабованої схеми. В основу схеми було складено ряд ключових таблиць, які пов'язані між собою логічним зв’язком (див. додаток Б). Для надання найвищого рівня безпеки даних були введені механізми шифрування на рівні поля, а також неподільність даних забезпечувалась через налаштування констрейнтів та використання транзакційних протоколів [11].

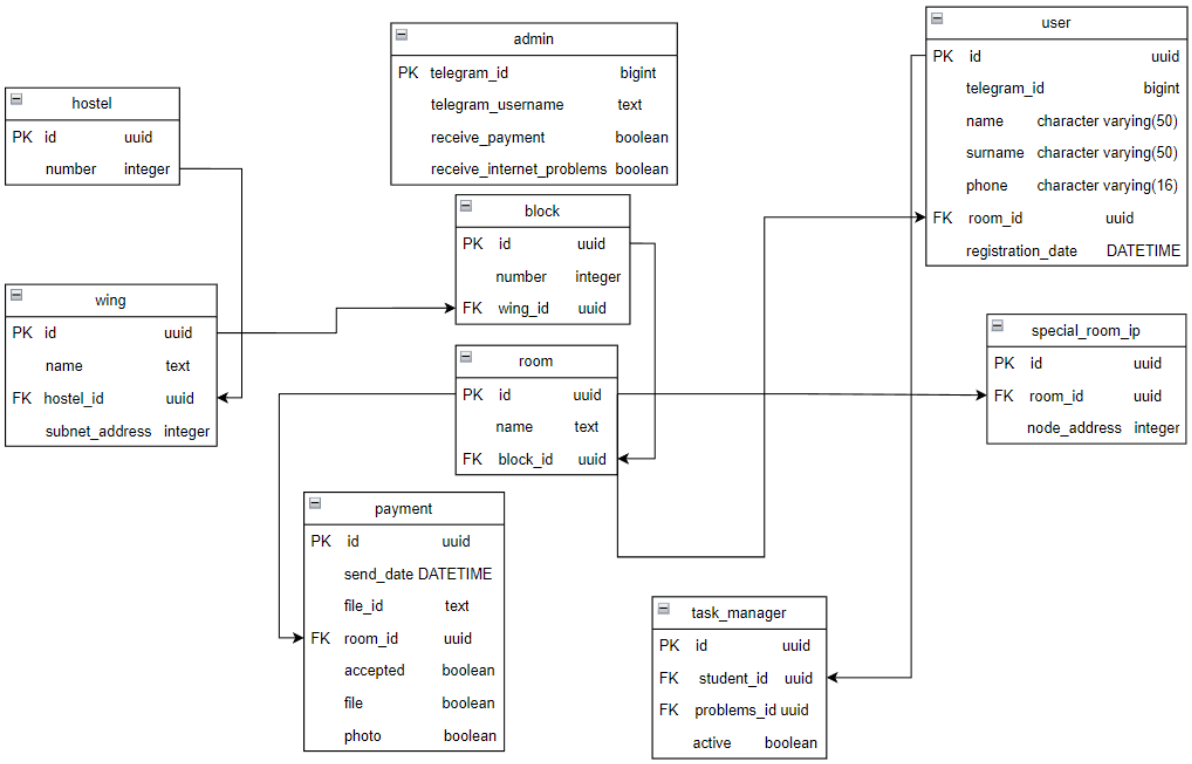

Рисунок 2.1 – Схема сутностей бази даних телеграм-бота tntu_internet

У базі даних, що зображена на рисунку 2.1 слід залучити наступні сутності:

- hostel – представляють гуртожитки в системі;

- wing – визначає окремі крила в межах гуртожитків;
- payments – ця таблиця фіксує усі транзакції та платежі, які були здійснені користувачами;
- admin – відповідає за збереження даних адміністраторів телеграм-бота, включно з їх повноваженнями;
- block – поділ крил на менші блоки, які можуть містити кілька кімнат;
- room – окремі кімнати, де проживають студенти;
- task_manager – сутність для контролю завдань, що стосується повідомлених проблем;
- special_room_ip – ведеться облік спеціальних IP-адрес, які можуть бути присвоєні певним кімнатам або зонам для конкретних потреб;
- person – сутність, яка містить профілі користувачів чат-бота.

Сутність “hostel” займається збереженням відомостей про гуртожитки, їх ідентифікаторах та номерам, створюючи основний рівень локаційної інформації. Кожне крило в гуртожитку представлене в “wing”, де окрім назв вказуються підмережі, асоційовані з кожним крилом, що має значення для мережевих налаштувань.

Таблиці “block” та “room” далі деталізують структуру проживання, дозволяючи вказувати конкретні блоки та кімнати, відповідно, надаючи прив'язку студентів до їхніх житлових просторів.

У “person” ведеться запис основних даних користувачів, що містить інформацію з профілів у Telegram і конфіденційні контактні дані. Ця сутність необхідна для персоналізації взаємодії та збільшення комунікації.

Сутність “payment” призначена для обробки фінансових транзакцій, де кожний платіж відстежується через його ідентифікатор, дату надсилання, статус прийняття, а також через позначення наявності супровідних файлів чи фотографій, що є важливим для валідації транзакцій.

Сутність “task_manager” відповідає за успішну реєстрацію та відслідковування завдань, пов'язаних з технічними запитами, зазначаючи їх активний статус та прив'язку до конкретних студентів та проблем.

Сутність “internet_problems” використовується для каталогізації інтернет-проблем, що допомагає в управлінні запитами на технічну підтримку.

Сутність “special_room_ip” забезпечує інформацію про спеціальні IP-адреси, призначені для конкретних кімнат, що є важливим для налаштувань мережі.

Адміністративна сутність “admin” містить детальну інформацію про адміністраторів телеграм-бота, включно з їхніми правами на отримання повідомлень про платежі та інтернет-проблеми.

Усі сутності спроектовані так, щоб якнайкраще використовувати ресурси і гарантувати високу продуктивність та безпеку.

Окрім того, система була покращена для швидкої обробки великих обсягів даних за допомогою індексації та вибору ефективних типів даних. Це гарантує швидкий доступ до ключової інформації, важливий для забезпечення безперебійної роботи бота, особливо під час пікових навантажень.

Також основною частиною проєкту стала інтеграція з зовнішніми системами, такими як платіжні шлюзи та системи ідентифікації, що вимагало реалізації безпечних API взаємодій. Ці інтерфейси надають боту взаємодіяти з іншими службами без ризику витоку конфіденційних даних [12].

Разом ця структура бази даних гарантує надійну платформу для зберігання та обробки інформації, що є основою для всіх функцій телеграм-бота та його взаємодії з користувачами. Також, регулярне впровадження оновлень безпеки та аудити системи допомагають підтримувати цю структуру в нагальному та захищеному стані.

2.2 Проєктування взаємодії телеграм-бота tntu_internet з БД

У даному розділі досліджено, як телеграм-бот під’єднується до бази даних PostgreSQL через ORM інтерфейс, що надає SQLAlchemy [13]. SQLAlchemy дозволяє працювати з базою даних через Python об'єкти і конструкції натомість

звичайних SQL запитів, що облегшує процес взаємодії і робить код придатним до читання та безпечним.

Для покращення взаємодії між телеграм-ботом та базою даних, можна використати шаблон проектування Repository. Цей підхід дозволяє абстрагувати логіку доступу до даних від бізнес-логіки, що адаптує розробку та тестування. Репозиторії відповідають за взаємодію з базою даних, втілюючи методи для зберігання, вилучення, оновлення та видалення даних. Це не тільки зменшує кількість повторюваного коду, але й покращує його читабельність та підтримку.

Ключову роль відіграє унайкращення запитів до бази даних, що виконуються через пакетну обробку та кешування результатів. Для зменшення навантаження на сервер і зниження часу відгуку, використовуються кеші, які тимчасово зберігають часто запитувані дані [14]. Це дозволяє мінімізувати кількість звернень до бази даних і покращує загальну ефективність системи.

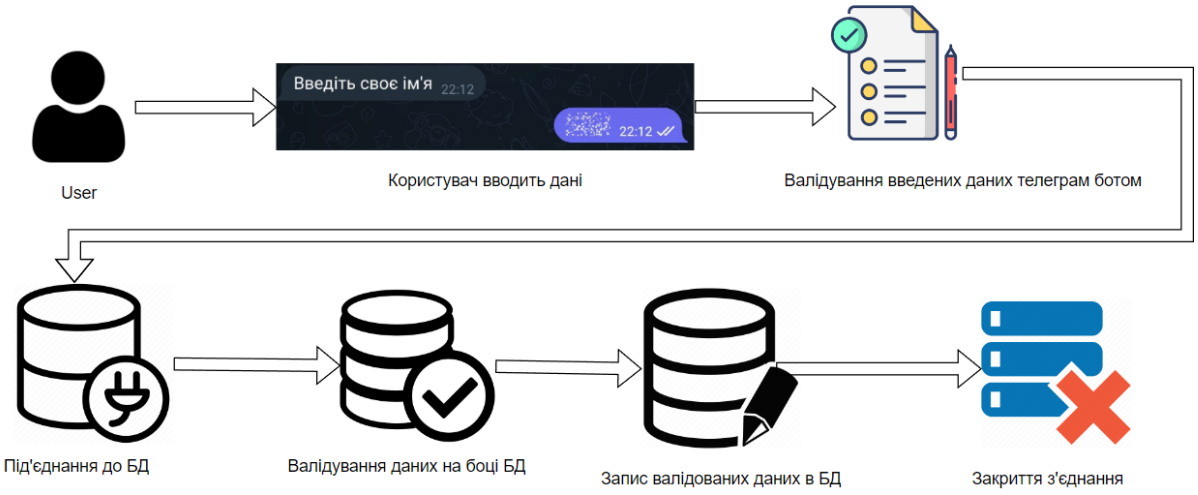

Рисунок 2.2 – Схема процесу обробки даних користувача телеграм-ботом tntu_internet та їх взаємодії з базою даних

Діаграма відображає черговість взаємодії користувача з телеграм-ботом і базою даних, що зображена на рисунку 2.2. Починаючи з лівої сторони, користувач вводить дані через інтерфейс телеграм-бота. Ця інформація передається до сервера, де вони проходять валідацію. Після успішного

підтвердження їхньої коректності, дані заносяться до бази даних. У випадку вдалої операції, користувач отримує підтвердження, а при знаходженні помилки – повідомлення про невдачу. На додаток, система розглядає кожен етап обробки даних, гарантуючи можливість спостереження та відстеження проблем у реальному часі.

Ця взаємодія акцентує важливість вірного проєктування взаємодії між телеграм-ботом та базою даних, у зв’язку з тим, що вона має вплив на якість користувацького досвіду і надійність системи в цілому.

Розробка безпечного API для взаємодії з базою даних також охоплюють використання сучасних механізмів аутентифікації та авторизації, зокрема OAuth та JWT (JSON Web Tokens) [15]. Ці технології гарантують, що лише авторизовані користувачі можуть виконувати операції, які впливають на дані, тим самим збільшуючи безпеку доступу до інформації.

Через використання цих технологій і методів, телеграм-бот tntu_internet може дієво та безпечно взаємодіяти з базою даних, гарантуючи стабільність та надійність в обробці користувацьких даних.

2.3 Проєктування процесу реєстрації користувачів та адміністратора телеграм-бота tntu_internet

Здійснення процесу реєстрації також включно з заходами для запобігання потенційним спробам зловживань або атакам. Окрім того, всі введені дані проходять сувору валідацію, що гарантує їх придатність заданим форматам і виключає можливість SQL ін'єкцій чи інших форм впливу на систему через вхідні дані [16].

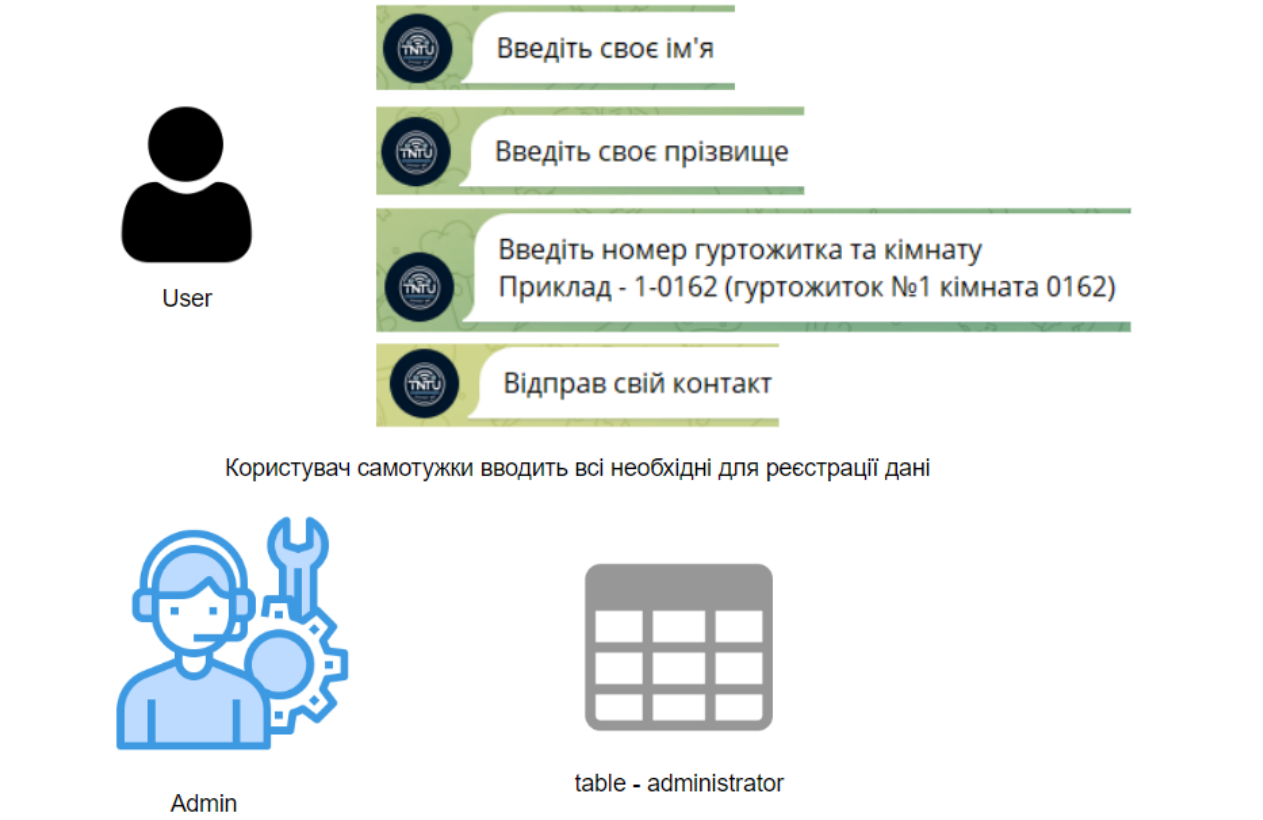

Рисунок 2.3 – Процес реєстрації користувача та адміністратора телеграм-бота tntu_internet

Цей інтерфейс, що зображений на рисунку 2.3, є важливим для розуміння взаємодії між користувачами та системою, і виділяє легкість і безпеку, яку надає реєстрації у створеному телеграм-боті tntu_internet. Виразне зображення кроків реєстрації показує, як користувачі можуть надійно вводити свої дані, які потім виконуються з високим рівнем шифрування [17].

Такий підхід не тільки полегшує процес реєстрації та гарантує захист персональних даних, але й гарантує ефективне управління правами користувачів, надаючи високоякісну організацію та контроль за діяльністю у межах телеграм-бота tntu_internet. Покращені механізми аутентифікації, такі як двоетапна аутентифікація та автоматичне виявлення підозрілих дій, додатково збільшують безпеку і стабільність використання бота.

Завдяки інтеграції зовнішніх API для перевірки автентичності даних, система гарантує високий рівень придатності запитам безпеки, включно з перевірку мобільних номерів в реальному часі. Це значно ускладнює можливість реєстрації за допомогою фіктивних або крадених даних, що є важливим кроком у профілактиці шахрайства і забезпеченні довіри серед користувачів.

Супроводження двоетапної аутентифікації для користувачів, що вимагають вдосконаленого рівня безпеки, також є частиною стратегії захисту. Це включає в себе відправку одноразового паролю (OTP) через SMS або електронну пошту, що значно збільшує рівень безпеки процесу реєстрації і доступу до чутливих функцій бота.

Фінальним етапом процесу реєстрації є засвідчення електронного номеру телефону, що додатково забезпечує перевірку автентичності користувача. Це збільшує впевненість в тому, що зареєстровані облікові записи застосовуються їх справжніми власниками та допомагає запобігти створення фіктивних або дубльованих профілів [18].

Додатково, система реєстрації включає автоматизовані перевірки безпеки на кожному кроці введення даних. Це дозволяє гарантувати, що всі дані, введені користувачем, проходять через ретельний аналіз на предмет придатності встановленим нормам безпеки, включаючи перевірку складності паролів та аналіз загроз на основі крайніх даних про віруси та шкідливе програмне забезпечення.

Інтеграція з сервісами підтвердження, такими як reCAPTCHA, допомагає ліквідувати автоматизовані боти та спам-атаки, підвищуючи загальну безпеку процесу реєстрації. Ці заходи не тільки зменшують ризик беззаконного доступу, але й покращують якість взаємодії користувачів з системою, гарантуючи, що взаємодія з телеграм-ботом є надійною та корисною.

Втім, активне використання шифрування трафіку між клієнтом і сервером через SSL/TLS гарантує, що всі передані дані залишаються секретними та захищеними від злочинців, які можуть намагатися перехопити вкрай важливу інформацію [19].

Комбінація цих технологій і методів надає більш комплексний підхід до безпеки, який є ґрунтовним та довгостроковим в роботі телеграм-бота tntu_internet, захищаючи його від можливих загроз та злочинних атак.

2.4 Проєктування архітектури телеграм-бота tntu_internet

Вдосконалення цієї взаємодії вимагає чіткого розподілу відповідностей між компонентами системи, що дає змогу кожному елементу існувати самостійно та ефективно. Застосування мікросервісної архітектури у телеграм-боті tntu_internet сприяє легкому масштабуванню та гарантує високу відкритість та надійність сервісу. Кожен мікросервіс несе відповідальність за певну практичність, як-от обробка запитів користувачів, управління базою даних або логіка бізнес-процесів [20].

Для узгодження між сервісами та надання неперервної взаємодії з користувачами, використовується система порядкування чергами повідомлень, яка забезпечує, що всі запити обробляються в порядку їх отримання, а відповіді надсилаються без затримок. Це вкрай важливо для функцій, які вимагають швидкої реакції, наприклад, для замовлення послуг або вирішення запитів на підтримку.

Впровадження контейнеризації з використанням Docker дозволяє стандартизувати середовища виконання для різних частин системи, зменшуючи ризики, пов'язані з “труднощами: працює у мене”. Контейнери з легкістю можна розгортати та масштабувати на будь-якій платформі, що підтримує Docker, що робить тестування нових версій бота більш гнучким і контрольованим [21].

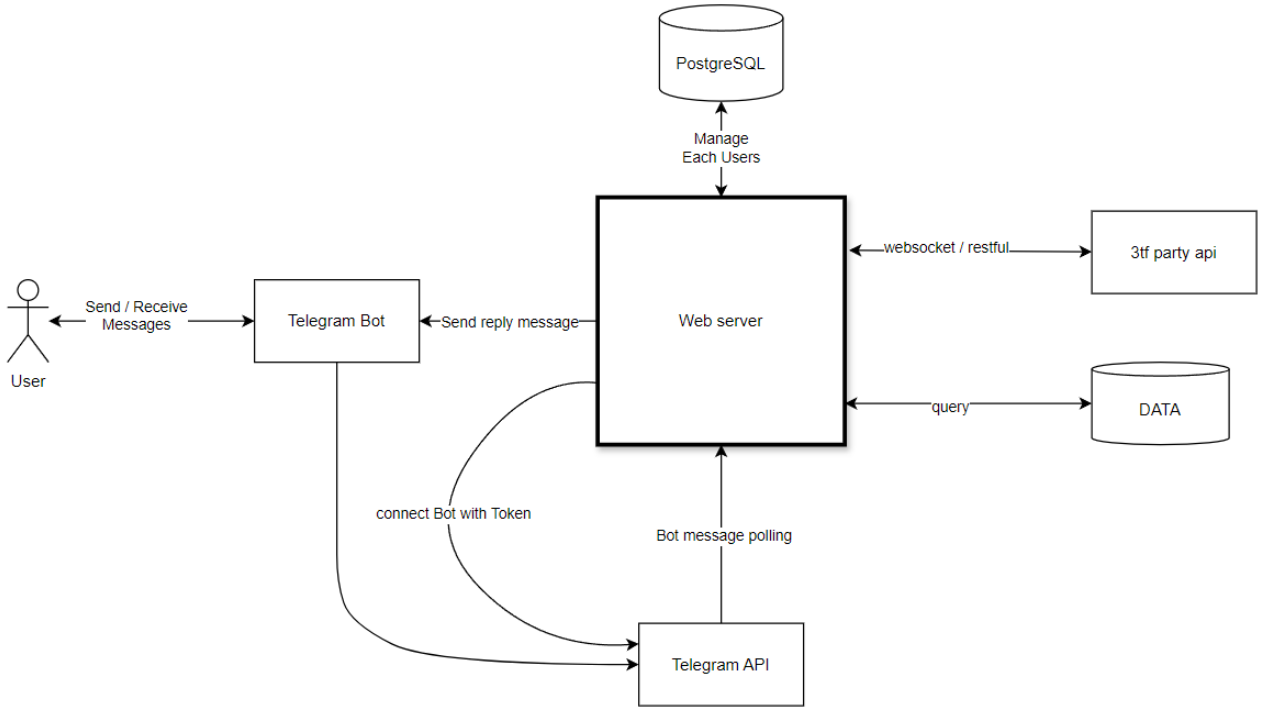

Рисунок 2.4 – Проєктування архітектури телеграм-бота tntu_internet

Крім того, використання балансувальних навантажень і резервних копій даних гарантує високу доступність і витривалість системи до збоїв, подано на рисунку 2.4. Це вкрай важливо для підтримки безупинної роботи телеграм-бота, особливо в умовах великого навантаження або при спробах кібератак [22].

Через вірно сплановану архітектуру телеграм-бота tntu_internet, він здатний ефективно та продуктивно обслуговувати велику кількість користувачів, гарантуючи стійку та безпечну роботу в умовах наявного цифрового середовища.

Користувачі взаємодіють з телеграм-ботом tntu_internet, надсилаючи текстові повідомлення, які бот приймає і виконує за допомогою Telegram API. Усі обмінні повідомлення між користувачами та ботом погоджуються через зазначений API, що гарантує безпечне та комфортне спілкування. Це дозволяє боту відповідати у режимі реального часу, схвалюючи високий рівень інтерактивності та покращенні обслуговування користувачів [23].

Головною частиною системи є веб-сервер, який слугує містком між Telegram API та базою даних. Він обробляє бізнес-логіку бота, керує запитами користувачів, впроваджує дії на основі отриманих даних та зберігає результати

в базі даних PostgreSQL. Ця база даних слугує для збереження вкрай важливої інформації, яка включає профілі користувачів, історію транзакцій, зареєстровані проблеми тощо. Завдяки цьому, всі дані завжди знаходяться у швидкому доступі та можуть бути використані для проведення аналізу та прийняття рішень у режимі реального часу.

Окрім цього, архітектура поєднує в собі зв'язок з третіми сторонами через API, дозволяючи веб-серверу взаємодіяти з зовнішніми сервісами для отримання допоміжних даних або виконання потенційних операцій. Прикладом може слугувати, інтеграція з платіжними системами, що дозволяє виконувати транзакції, а з системами ототожнення – верифікувати особисті дані користувачів [24].

Також архітектура окреслює використання механізму опитування повідомлень для систематичного отримання оновлень від Telegram API, що надає вчасну відповідь бота на вимоги користувачів. Це включає в себе обробку як текстових повідомлень, так і мультимедійних файлів, гарантуючи багатофункціональність даного бота.

Описана архітектура надає надійну та безпечну основу для функціонування чат-бота, покращуючи взаємодію між усіма складниками системи, з можливістю масштабування та пристосування до змінюваних потреб користувачів. Вона також зважає на можливість інтеграції нових функцій та сервісів, що дозволяє постійно збільшувати якість обслуговування та відповідати на виклики новітніх інформаційних технологій. Завдяки цьому, телеграм-бот tntu_internet залишається гнучким і надійним інструментом для гарантування ефективного і безпечного взаємодії з користувачами.

2.5 Огляд потенційних загроз

Існують реальні можливі ризики, які мають вплив на роботу телеграм-ботів, і досить важливо розуміти та визнавати їх з метою створення результативних заходів безпеки [25].

Телеграм-бот tntu_internet налічує розширені функції безпеки, що гарантують надійний захист від безмежного спектру кібератак. Ці загрози містять шкідливі програми, які можуть заразити сервери і привласнити чи зруйнувати вкрай важливі дані, а також атаки "відмова у обслуговуванні", що скероване на перевантаження системи необроблюваною кількістю запитів. Ін'єкція SQL-коду є ще однією надзвичайно серйозною загрозою, у зв’язку з тим, що вона може слугувати причиною незаконного доступу до бази даних. Фішинг та інші види соціальної інженерії також представляють ризик, оскільки вони можуть обманом змусити користувачів надати свої конфіденційні дані, такі як номер телефону та ідентифікатор облікового запису телеграм. Розголошення конфіденційної інформації через невідповідне збереження або передачу є ще однією можливою проблемою.

Заходи безпеки, упорядковані у телеграм-боті, містять застосування наявних криптографічних технологій для шифрування даних, а також створення цілісних систем ідентифікації та валідації, які гарантують додатковий рівень безпеки. Системи спостереження і відповіді на питання допомагають своєчасно визначати та реагувати на будь-яку сумнівну діяльність, зменшуючи ризики від можливих загроз і надаючи стабільне та безпечне середовище для користувачів.

Перегляд та усвідомлення можливих загроз є ключовими для гарантування безпеки та надійності телеграм-бота tntu_internet. Превентивний підхід до уподібнення, запобігання та реагування на загрози надає стійку та надійну роботу бота в умовах безупинно змінюваних кібератак [26].

З метою надання найвищого рівня захисту, телеграм-бот tntu_internet вводить стратегії безпеки, які містять перевірку надійності на всіх етапах життєвого циклу створення та використання бота. Особливу увагу відводять систематичним оновленням безпеки, щоб врахувати крайні відомі уразливості та контратаки.

Надання безпеки також визначає супроводження методів штучного інтелекту та автоматизованого навчання з метою аналізу звичок користувачів і

визначення безпрецедентних або підозрілих патернів. Це дозволяє визначати і зупиняти спроби злочинного втручання ще до того, як вони завдадуть шкоду.

Всебічні аудити безпеки відбуваються систематично, з метою заохочення зовнішніх спеціалістів з кібербезпеки, що гарантує самостійний огляд системи і виявлення можливих незначних місць. Також, важливим складником захисту є створення та введення політик безпеки, які упорядковують взаємодію користувачів і адміністрації з системою, гарантуючи дотримання найкращих практик і стандартів.

Отже, ретельний огляд можливих загроз і введення задовільних заходів безпеки є важливим з метою гарантування довгострокової постійності та захисту телеграм-бота tntu_internet, надаючи користувачам впевненість у надійності та секретності їхніх даних.

2.6 Проєктування механізмів масштабування для телеграм-бота tntu_internet

Для надання постійної та безперебійної роботи та своєчасної обробки надзвичайно великої кількості запитів користувачів, слід вірно спроєктувати механізми масштабування телеграм-бота tntu_internet. Масштабування дозволяє системі якісно та результативно обробляти зростаючу кількість користувачів та запитів без зменшення продуктивності [27].

Цей підхід окреслює підвищення ресурсів одного сервера, на якому працює телеграм-бот tntu_internet. Це може бути набуто шляхом додавання оперативної пам'яті, процесорних ядр або швидших накопичувачів. Переваги вертикального масштабування містять у собі простоту реалізації та керування. Однак, існують фізичні обмеження щодо того, наскільки можна збільшити інтенсивність одного сервера.

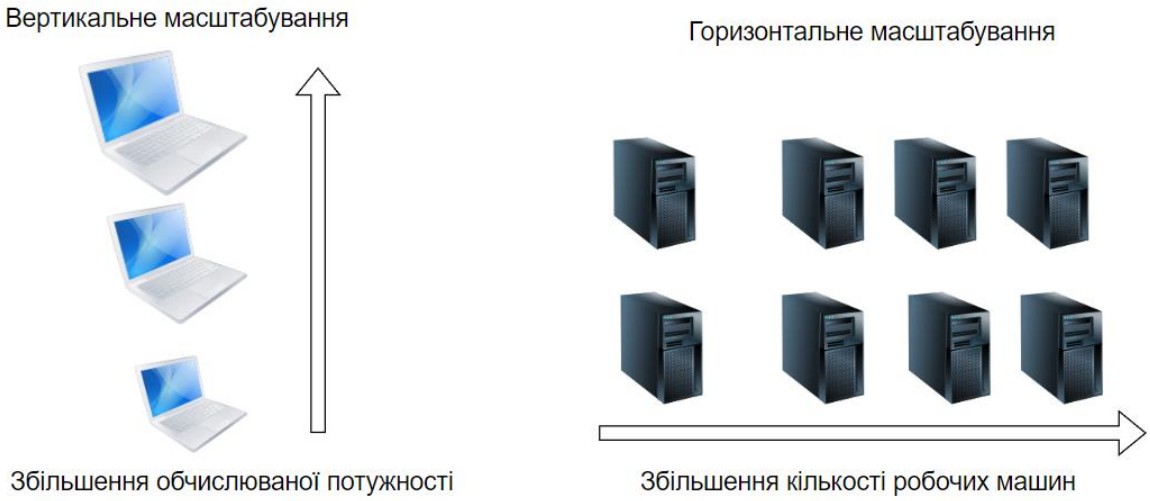

Рисунок 2.5 – Способи масштабування обчислюваних потужностей

Горизонтальне масштабування, що зображено на рисунку 2.5:

Цей підхід передбачає додавання нових серверів, на яких будуть працювати копії телеграм-бота tntu_internet. Застосовуючи балансувальник навантаження, можна рівномірно ділити запити між кількома серверами. Це дозволяє системі опрацьовувати значно більшу кількість запитів і гарантує високу відкритість, оскільки у разі виходу з ладу одного сервера, інші можуть продовжувати обробку запитів [28].

Контейнеризація за допомогою Docker дозволяє відлучати додатки та їх залежності у контейнери, які можуть працювати самостійно від хостової системи [29]. Це гарантує:

Портативність: Контейнери можуть бути розгорнуті з використанням будь-якої платформи, що підтримує Docker.

Швидкість розгортання: Контейнери запускаються швидше, ніж віртуальні машини.

Масштабованість: Легко розгортати нові контейнери для опрацювання допоміжних навантажень.

Kubernetes (K8s) є потужним та ефективним інструментом для оркестрації контейнерів. Він дозволяє автоматизувати розгортання, масштабування та керування контейнеризованими додатками [30]. Головні можливості Kubernetes:

Автоматичне масштабування: Автоматичне додавання або видалення контейнерів залежно від навантаження.

Управління конфігураціями: Комфортне керування конфігураціями та секретами.

Балансування навантаження: Розподіл запитів між контейнерами з метою рівномірного навантаження.

Відновлення після збоїв: Автоматичний повторний запуск пошкоджених контейнерів та переобладнання їх на здорові вузли.

Впровадження масштабованої архітектури телеграм-бота tntu_internet може спричинити наступні економічні переваги:

Зниження витрат на обладнання: Використання контейнерів та оркестрації дозволяє результативно використовувати вільні ресурси, що може знизити потребу в покупці нового обладнання.

Гнучкість розгортання: Легкість розгортання нових дублікатів бота та керування ними дозволяє швидко пристосовуватися до змін у навантаженні, що гарантує безперебійний сервіс для користувачів.

Підвищення задоволеності користувачів: Надання стійкої та безупинної роботи бота спричиняє підвищення рівня задоволеності користувачів, що може збільшити їх кількість та, відповідно, можливий дохід.

Зниження витрат на обслуговування: Автоматизація процесів розгортання та керування контейнерами зменшує потребу у використанні людських ресурсів, що зменшує витрати на обслуговування.

Масштабування доходів: Із підвищенням кількості користувачів та постійною роботою бота можливе збільшення прибутків від реклами, платних функцій або донатів.

Таким чином, правильно спроєктована система масштабування не тільки гарантує технічну стійкість та результативність, але й призводить до

економічного зростання та підвищення прибутковості телеграм-бота tntu_internet.

2.7 Висновок до другого розділу

У другому розділі було визначено вкрай важливі аспекти проєктування програмного інтерфейсу телеграм-бота для ТНТУ ім. І. Пулюя. Було створено структуровану та масштабовану базу даних, яка є основою для збереження та обробки інформації, потрібної для всіх операцій телеграм-бота. Згадані сутності та зв'язки між ними гарантують гнучкість та продуктивність в обробці даних, що є критичним для безпеки та результативності системи.

Проводячи аналіз взаємодій телеграм-бота з базою даних, було введено ORM-підхід через SQLAlchemy, який надає зручно управляти запитами та знизити кількість потенційних помилок в SQL-операціях. Інтеграція веб-сервера з Telegram API і базою даних була ретельно продумана для гарантування безшовної взаємодії між користувачами та системою.

Перебіг реєстрації користувачів та адміністраторів був спроєктований з міркувань зручності та безпеки, надання простоти введення даних та їхню надійну валідацію. Представлення цього процесу яскраво показує всі етапи взаємодії та ролі, які відіграють користувачі та адміністратори у життєвому циклі телеграм-бота.

У підсумку було проведено докладний огляд архітектури телеграм-бота, який зважає на необхідність ефективного зв'язку між користувачами та системою, здатності до своєчасної реакції на запити та невагомості в керуванні та масштабуванні. Архітектура, заснована на використанні Docker контейнерів, гарантує високу ступінь ізоляції компонентів та легкість у простяганні та розвитку.

Додатково було проаналізовано можливі загрози, які з’являються під час експлуатації телеграм-бота, та розроблено комплекс заходів для їх противаги. Це включає супроводження сучасних криптографічних технологій, системи

спостереження та реагування на непорозуміння, а також політики безпеки, які надають надійний захист даних користувачів. Дослідження механізмів масштабування, таких як вертикальне і горизонтальне масштабування, контейнеризація за допомогою Docker та оркестрація контейнерів за допомогою Kubernetes, акцентує важливість готовності системи до обробки зростаючої кількості запитів та гарантування високої продуктивності та доступності.

Таким чином, другий розділ надає універсальний аналіз та проєктні висновки, спрямовані на розроблення надійного, корисного та масштабованого телеграм-бота tntu_internet, здатного якісно обслуговувати користувачів і відповідати на виклики сучасного цифрового середовища.

РОЗДІЛ 3. ТЕСТУВАННЯ ТА ЕКСПЛУАТАЦІЯ ПРОГРАМНОГО ІНТЕРФЕЙСУ ТЕЛЕГРАМ-БОТА TNTU_INTERNET

3.1 Реалізація схеми БД для телеграм-бота tntu_internet

Після проєктування в підрозділі 2.1, база даних для телеграм-бота була успішно реалізована, що дозволило перейти до етапу практичної реалізації.

За допомогою бази даних PostgreSQL, було створено структуру для бази даних телеграм-бота, котра забезпечує надійність та високу результативність при роботі з великими об'ємами інформації(див. додаток В).

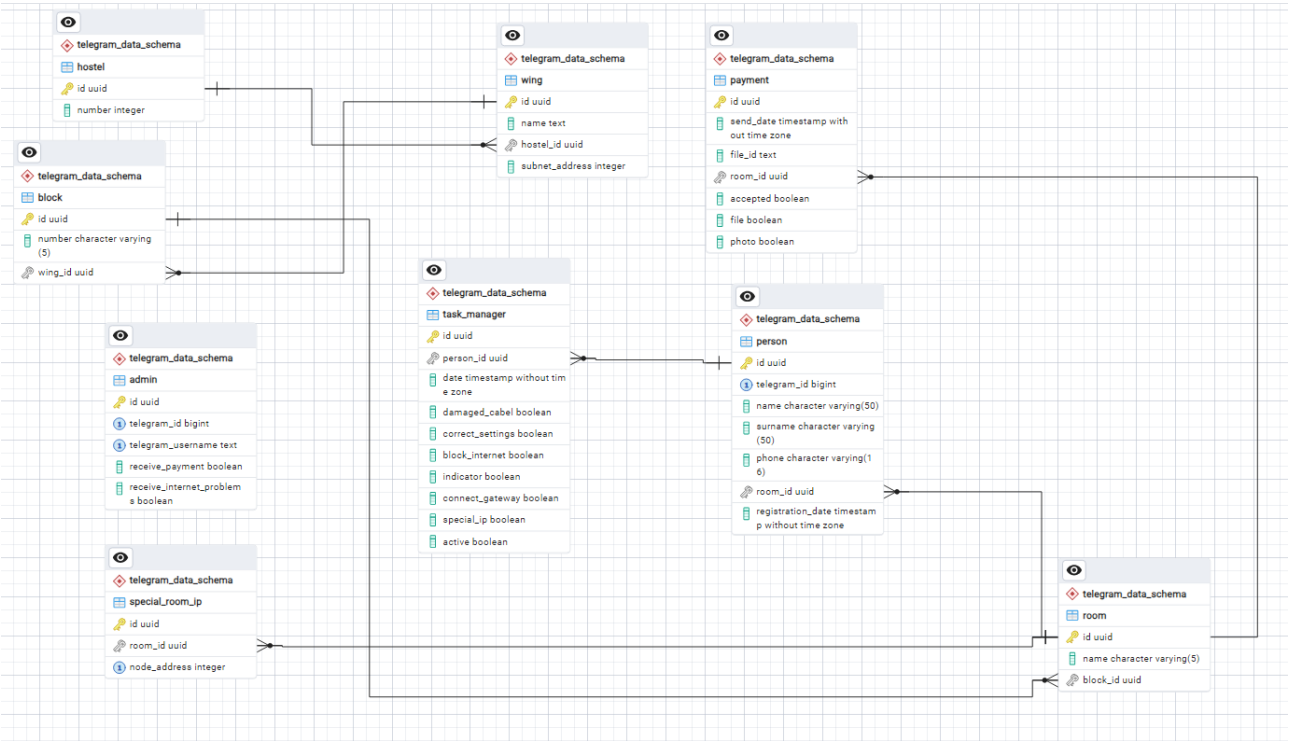

Рисунок 3.1 – Реалізована схема бд для телеграм-бота tntu_internet

Завдання реалізації структурованої та функціональної бази даних для підтримки телеграм-бота в Telegram було виконано з використанням інструменту PgAdmin, подано на рисунку 3.1. Цей графічний інтерфейс для управління базами даних PostgreSQL надав можливість детально спланувати,

створити та налагодити взаємозв'язки між таблицями, забезпечивши інтуїтивно зрозумілу та ефективну роботу з даними.

Створення бази даних через PgAdmin дозволило наочно представити ієрархію відносин між різними сегментами даних, включаючи інформацію про гуртожитки, крила, блоки та індивідуальні кімнати. Такий підхід у проєктуванні полегшив налагодження логіки збереження інформації про користувачів, а також опрацювання фінансових транзакцій.

Ключову роль в адмініструванні системи відіграють таблиці admin та task_manager, які містять відомості про адміністраторів телеграм-бота та поточні завдання відповідно. Інші таблиці, як-от special_room_ip, відіграють важливу роль у керуванні мережевими налаштуваннями, забезпечуючи специфічні вимоги для підключення до інтернету в окремих кімнатах.

В процесі реалізації в PgAdmin було враховано всі необхідні заходи для забезпечення цілісності, безпеки та оптимізації запитів до бази даних. Створені первинні ключі та зовнішні ключі забезпечують структурування та зв'язність даних, що є фундаментом для забезпечення стабільної роботи телеграм-бота.

В результаті використання PgAdmin база даних була створена з особливою увагою до деталей, які гарантують ефективне використання і легкість обслуговування, відкриваючи широкі можливості для масштабування та розвитку системи.

3.2 Валідування вхідних даних телеграм-бота tntu_internet

Є найважливіша частина розробки, а саме валідування даних, які вводять користувачі, під час реєстрації, передавання платіжок, тобто всього функціоналу телеграм-бота. Це не тільки забезпечує коректність та безпеку інформації, яка обробляється та зберігається в базі даних, а й задає коректне функціонування всієї системи(див. додаток Д). Нижче наведено приклади функцій валідації, що використовуються для перевірки різних типів введених даних, які користувачі можуть надсилати через телеграм-бота [31].

Функція “validate_name” перевіряє, чи ім'я користувача відповідає встановленому формату. Ця функція надає можливість записувати імена, виключно які починаються з великої літери, та можуть містити дефіси, апострофи та різні знаки, обмежуючи довжину до 50 символів, ця функція представлена нижче.

```
def validate_name(name: str) -> True | False:
    return bool(re.match(r"\b([A-ZÀ-ŷA-ЩЬЮЯҐЄІЇ][-,a-z. 'A-
щьюяґєіїÀ-ŷ]{0,49} *)+", name))
```

Функція “validate_phone_number” гарантує обов’язковий контроль для перевірки номерів телефону. Вона ревізує формат українських мобільних номерів, дозволяючи наявність міжнародного коду України (+38) та роблячи зігальну перевірку довжини номера, ця функція представлена нижче.

```
def validate_phone_number(phone_number):
    return bool(re.match(r"^\+?3?8?(0\d{9})$", phone_number))
```

Функція “validate_room_name” інспектує номери кімнат для подальшого використання в функціоналі телеграм-бота. Назва може складатися з числа (до чотирьох цифр) та необов'язкової літери в кінці, що відповідає більшості схем нумерації кімнат, ця функція представлена нижче.

```
def validate_room_name(room_name):
    return bool(re.match(r"^\d{1,4}[a-яA-Яa-z]?$", room_name))
```

Цей декоратор є особливо практичним у контексті телеграм-бота, де динамічність введених даних може призвести до можливих критичних помилок або небезпечних зловживань. Користування “validate_params” дозволяє централізовано керувати перевіркою на вірність формату в межах всього проєкту, забезпечуючи консистентність та надійність перевірок на рівні всіх наявних функцій, що вимагають введення інформації від користувачів.

Декоратор “validate_params” може бути легкий у додаванні у будь-яку функцію, що сприймає параметри, зокрема, обробники команд телеграм-бота, які взаємодіють з користувачами через текстові повідомлення, зображений в лістингу 3.1. Його експлуатація знижує ризик помилок, зберігаючи високий рівень безпеки при обробці запитів, а також полегшує розробку та підтримку коду, оскільки IT-фахівцям не потрібно вручну додавати перевірки на кожному кроці проєкту.

Лістинг 3.1 – Validate_params

```
def validate_params(params_to_check, validation_functions):
    def decorator(func):
        @wraps(func)
        def wrapper(*args, **kwargs):
            for param, validation_function in zip(params_to_check,
validation_functions):
                if param in kwargs and not
validation_function(kwargs[param]):
                    return False
            return func(*args, **kwargs)
        return wrapper
    return decorator
```

Під час створення бази даних були введені обмеження на поля таблиць для забезпечення введення даних у певному форматі [32].

Функція “validate_email” – ця функція перевіряє, чи введена електронна адреса відповідає стандартному формату електронних адрес. Важливим є відсікання можливих помилок у введенні, а також використання електронної адреси в процесах надсилання повідомлень чи сповіщень, ця функція представлена нижче.

```
def validate_email(email: str) -> bool:
       return bool(re.match(r"[^@]+@[^@]+\.[^@]+", email))
```

Ця функція є незамінною для забезпечення правильної обробки та використання дат у різноманітних функціях телеграм-бота, особливо коли це стосується планування подій, термінів платежів або логування активності

користувачів. Коректне визначення та валідація формату дат дозволяє запобігти помилкам, які можуть виникати через неправильне введення даних користувачами.

Ця проста, але ефективна валідація гарантує, що дата вводиться у строго визначеному форматі, знижуючи ризик помилок при подальшому використанні даних, таких як обчислення періодів часу або взаємодія з зовнішніми API, які вимагають специфічних форматів дат, ця функція представлена нижче.

```
def validate_date(date_text: str) -> bool:
    return bool(re.match(r"\d{4}-\d{2}-\d{2}", date_text))
```

Завдяки інтеграції цієї функції, яка зображена в лістингу 3.2 телеграм-бот може ефективно перевіряти важливі мережеві параметри, забезпечуючи, що вони відповідають стандартам і є придатними для подальшого використання. Така перевірка є необхідною для функцій, що залежать від точності мережевих налаштувань, наприклад, при діагностиці мережевих збоїв або конфігурації внутрішніх з'єднань.

Лістинг 3.2 – Validate_ipv4_address

```
def validate_ipv4_address(ip_address: str) -> bool:
    pattern = r"^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$"
    return bool(re.match(pattern, ip_address) and all(0 <=
int(num) < 256 for num in ip_address.rstrip().split('.')))
```

Процес валідації в телеграм-боті може також включати динамічну обробку введених даних. За допомогою підписок на події або real-time валідаційних хуків, можна забезпечити миттєву перевірку даних в момент їх введення. Це знижує імовірність помилок на більш пізніх етапах обробки і підвищує задоволеність користувачів за рахунок миттєвого зворотного зв'язку.

Реалізація таких розширених валідаційних функцій підвищує надійність системи і зменшує витрати часу на усунення помилок та вирішення проблем з мережею, сприяючи стабільній та безпечній роботі телеграм-бота.

3.3 Тестування готового рішення телеграм-бота tntu_internet

Тестування функціональності телеграм-бота tntu_internet, що зображений на рисунку 3.2, є критично важливим для забезпечення його ефективної та безперебійної роботи.

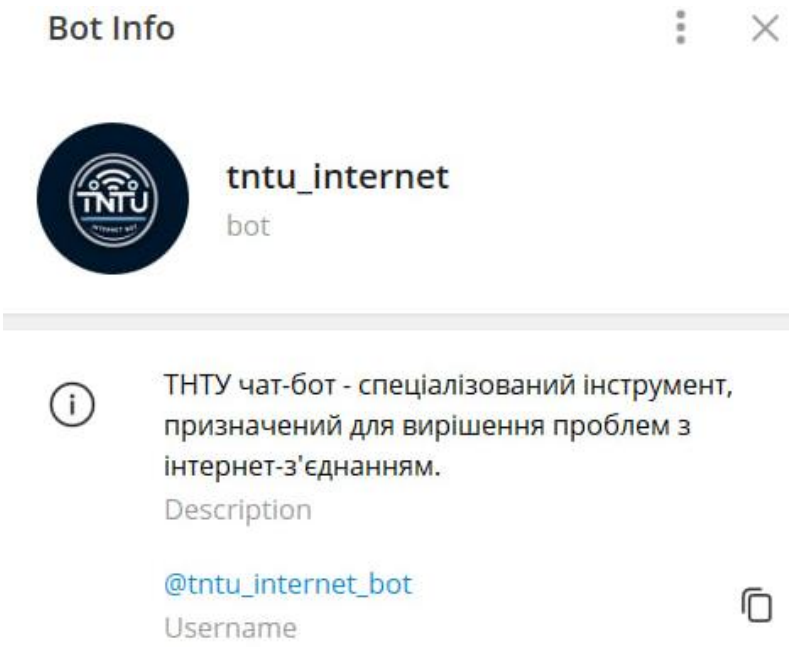

Рисунок 3.2 – Телеграм-бот tntu_internet

Цей етап перевіряє, як бот виконує різноманітні операції, передбачені для кінцевого користувача: від реєстрації до використання основних функцій, таких як отримання інформації, взаємодія з адміністраторами, та подання заявок на вирішення проблем з інтернет-з'єднанням [33].

Перший крок тестування включає процес реєстрації користувачів у боті. Перевіряється, наскільки інтуїтивно зрозумілим є інтерфейс реєстрації, чи ясно користувачеві, які кроки потрібно виконати, та чи правильно система обробляє введені дані. Тестуються також різні сценарії, коли користувач надає неправильні або неповні відомості.

В ході цього етапу тестування, особлива увага приділяється коректності валідації введених даних. Перевіряється, як телеграм-бот реагує на різні введені

телефонних номерів та інших персональних даних, що забезпечує надійність зберігання інформації і мінімізує ризик виникнення помилок або зловживань. Наприклад, система має відхилити некоректні формати телефонних номерів або адреси без потрібного домену, забезпечуючи тим самим додатковий шар захисту для даних користувачів.

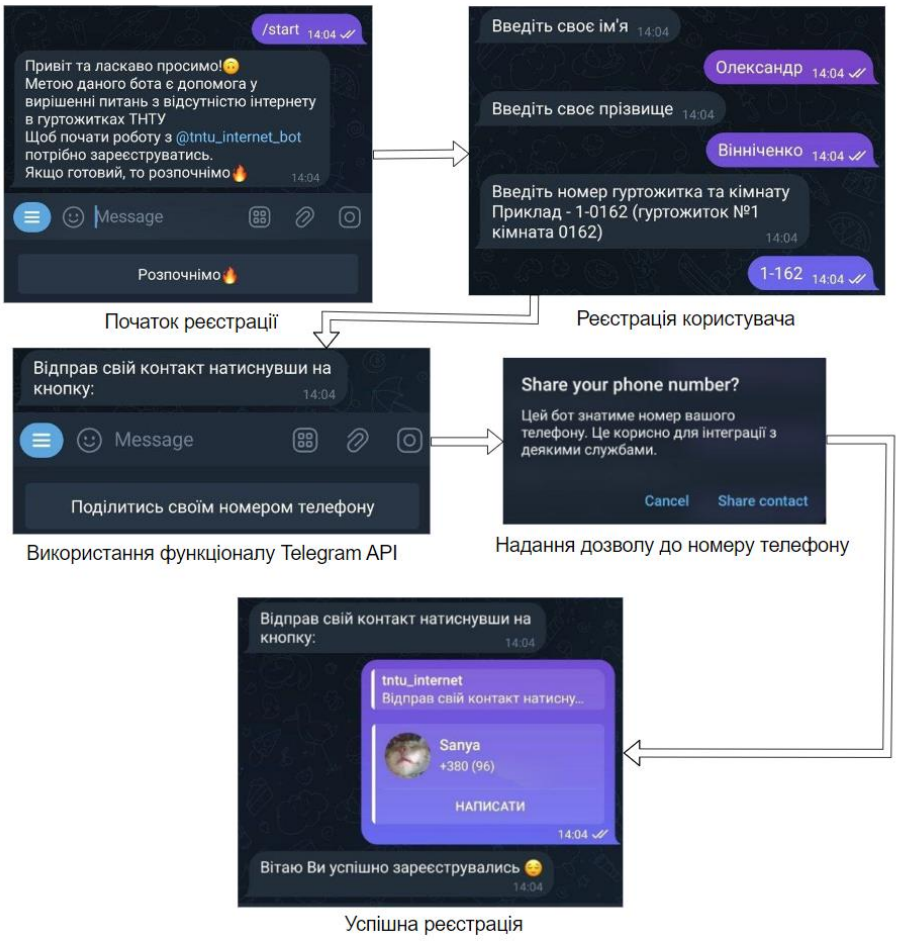

Рисунок 3.3 – Процес реєстрації в телеграм-боті tntu_internet

Далі відбувається оцінка головного функціоналу, доступного після реєстрації, подано на рисунку 3.3. Особлива увага приділяється відповідям бота на запити: чи вони зрозумілі, чи містять всю необхідну інформацію та як швидко бот реагує (див. додаток Ж).

Оцінюється, наскільки ефективно бот надає підтримку та керівництво через свої інтерактивні меню та команди. Також тестується, як система обробляє помилкові або неочікувані введення від користувачів. Перевіряється, чи

коректно телеграм-бот відповідає на невідповідні запити або помилки у введених даних, і як він керує користувачем назад до коректного вводу або інструктує про правильне використання опцій.

Важливою частиною цього етапу є вимірювання швидкості реакції бота на введення користувача. Визначається, чи бот надає миттєві відповіді або забезпечує швидкі рішення на складні запити, що критично важливо для забезпечення задоволеності користувачів і підвищення ефективності їх взаємодії з системою (див. додаток Е).

Тестування також охоплює спостереження за тим, як бот керує великою кількістю одночасних запитів, забезпечуючи стабільність та надійність при високих навантаженнях.

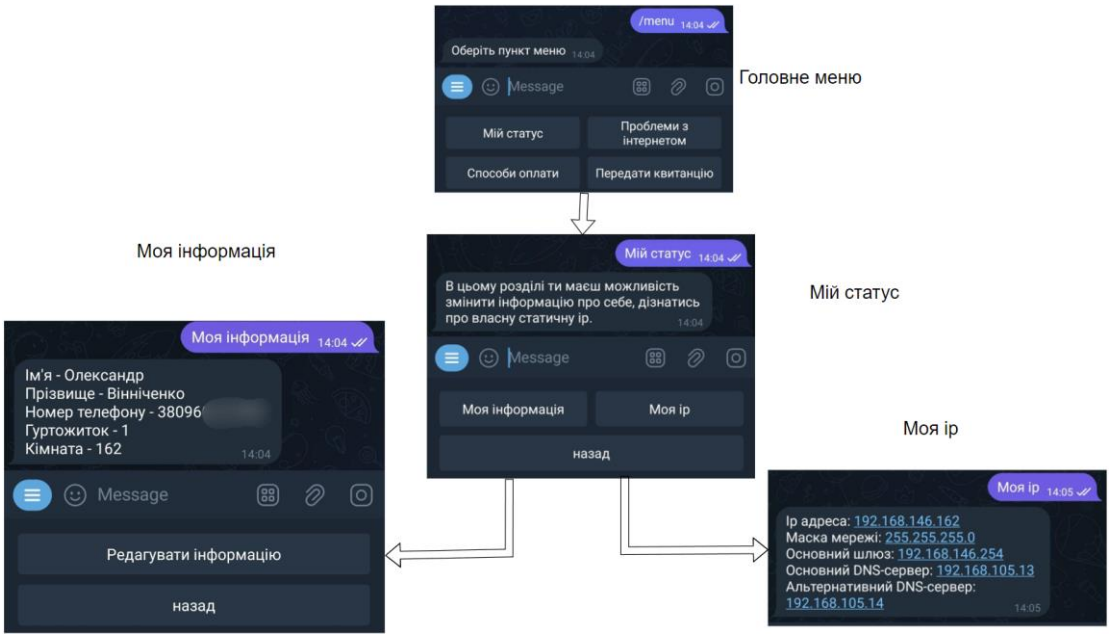

Рисунок 3.4 – Процес отримання користувальницької інформації в телеграм-боті tntu_internet

Користувачі мають доступ до головного меню, звідки можуть вибрати “Моя інформація”, щоб переглянути свої особисті дані, “Мій статус”, для відстеження актуального стану інтернет-з'єднання, а також “Моя IP”, де вони можуть побачити свою мережеву адресу, зображено на рисунку 3.4.

З опції “Моя інформація” користувачі можуть переглянути свої основні дані, такі як ім'я, прізвище, номер телефону, групу та номер кімнати у гуртожитку. Вони також мають можливість редагувати ці дані, використовуючи кнопку “Редагувати інформацію”, що забезпечує простоту та зручність управління особистою інформацією (див. додаток К).

“Мій статус” надає інформацію про поточний стан інтернет-з'єднання користувача, у тому числі статус підключення та наявність мережевих проблем. Це дозволяє користувачам швидко визначати причини можливих перебоїв у роботі мережі.

Варіант “Моя IP” показує користувачам їхній IP-адрес, маску мережі, шлюз за замовчуванням та DNS-сервери, що дозволяє здійснити діагностику мережевих проблем або планувати мережеві налаштування.

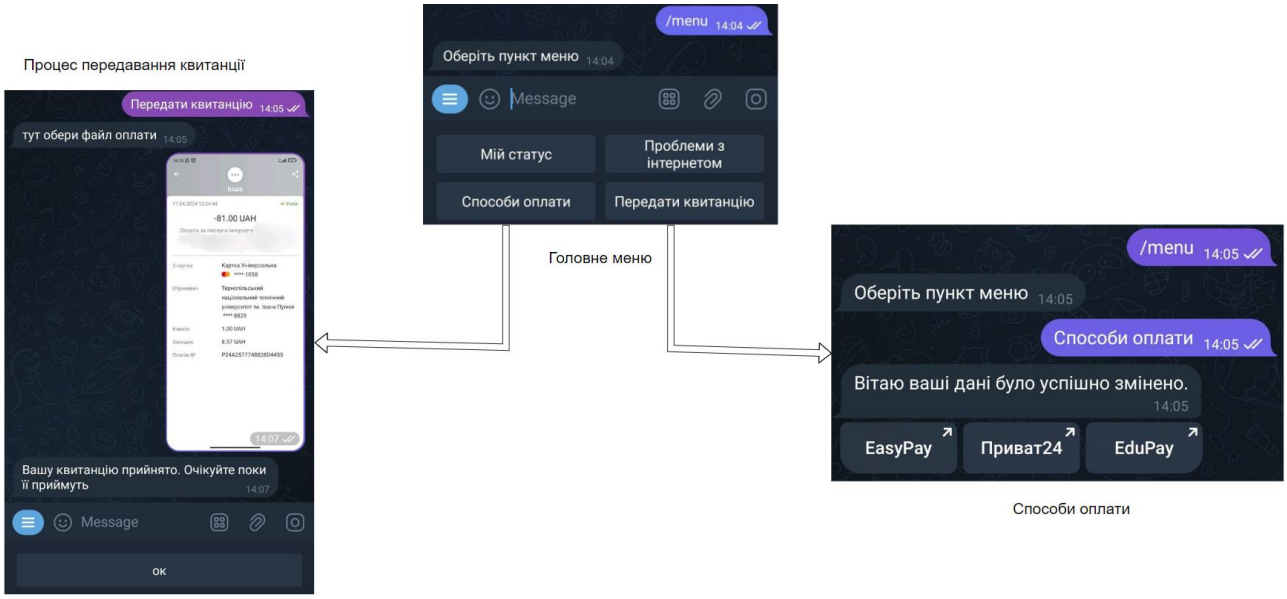

Рисунок 3.5 – Процес оплати за інтерне-послуги, та передавання квитанції оплати в телеграм-боті tntu_internet

Користувач має змогу легко здійснити оплату безпосередньо через інтерфейс бота, що значно спрощує процедуру і збільшує комфорт користування, що зображено на рисунку 3.5.

З головного меню користувач може вибрати опцію “Способи оплати”, що надасть йому доступ до різних методів оплати, таких як EasyPay, Приват24, EduPay. Після вибору відповідного методу користувач може виконати платіж відповідно до вказівок, що з'являються на екрані.

Оплативши послугу, користувач може надіслати квитанцію через бота, вибравши опцію “Передати квитанцію”. Для цього користувачу потрібно завантажити файл квитанції або фотографію, яка підтверджує здійснення платежу. Бот автоматично обробляє надіслану квитанцію та надає повідомлення про успішне отримання та обробку документу.

Ця функція робить телеграм-бота цінним інструментом для користувачів, оскільки вона забезпечує зручний та ефективний спосіб управління оплатами за інтернет. Вона відкриває можливість здійснювати фінансові операції в будь-який час, не відволікаючись від інших справ, та гарантує швидку відповідь на дії користувача, що сприяє підвищенню їх загальної задоволеності використанням телеграм-бота.

Шифрування даних перед передачею та зберіганням забезпечує, що особиста і фінансова інформація залишається конфіденційною та захищеною від несанкціонованого доступу [34].

Функціонал телеграм-бота також включає засоби для верифікації і підтвердження транзакцій. Після кожної оплати користувач отримує деталізоване підтвердження транзакції, яке включає унікальний ідентифікатор операції, дату та час, суму та статус операції, забезпечуючи повну прозорість фінансових операцій.

У разі виявлення помилок чи запитань щодо транзакцій, користувачі можуть звернутися до служби підтримки через самого бота. Цей канал комунікації дозволяє швидко реагувати на запити користувачів і вирішувати будь-які проблеми, пов'язані з оплатою.

Така інтеграція платіжних опцій в телеграм-бот забезпечує не тільки зручність, але й додатковий рівень довіри та безпеки для користувачів, роблячи

цей інструмент незамінним для щоденного використання в умовах сучасного цифрового середовища.

3.4 Ієрархія телеграм-бота tntu_internet

Організація кодової бази ТНТУ телеграм-бота tntu_internet – це продумана система, яка дозволяє ефективно керувати і взаємодіяти з усіма складовими проекту, що подана на рисунку 3.6. Модульність також сприяє легкій інтеграції з зовнішніми API та сервісами, які можуть знадобитися для розширення функціоналу бота, наприклад, для інтеграції платіжних систем або забезпечення додаткових сервісів [35].

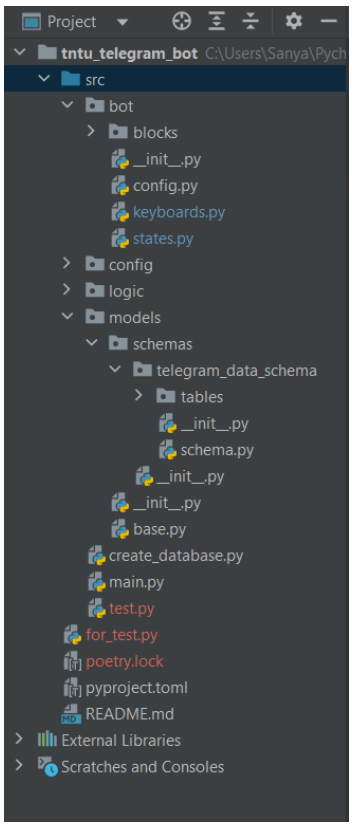

Рисунок 3.6 – Ієрархія проєкту телеграм-бота tntu_internet

Документація коду є ще однією критичною частиною проєкту. Вона включає детальні описи функцій, модулів та їхніх інтерфейсів, що значно полегшує ознайомлення нових розробників з проєктом та сприяє кращому

розумінню загальної архітектури. Якісна документація також важлива для майбутнього масштабування та підтримки проєкту, дозволяючи швидко адаптувати систему до змінюваних вимог та внести необхідні зміни в код.

Завдяки такій організації кодової бази, телеграм-бот tntu_internet стає міцною, добре організованою системою, яка здатна ефективно реагувати на динамічні вимоги сучасного цифрового світу.

В директорії src, серце цього проекту, кожна підпапка відіграє ключову роль у розділенні логіки та функціоналу.

Компоненти інтерфейсу телеграм-бота tntu_internet, які користувач бачить і з якими взаємодіє – клавіатури та команди, розміщуються у папці bot. Це включає файли, такі як keyboards.py, де визначаються кнопки відповідей, та states.py, який управляє станами бесіди, що дозволяє нашому боту зберігати контекст взаємодії з кожним користувачем.

Налаштування, що керують зовнішніми залежностями та змінними середовища, живуть у config, ізолюючи критичні параметри від основної бізнес-логіки, що зосереджена в logic. Таке розташування дозволяє легко змінювати поведінку бота без потреби занурюватися в глибинні шари коду.

Моделі та схеми даних, які формують кістяк нашої бази даних, є у папці models. Вони відображають структури, що бот використовує для інтерпретації та збереження інформації, в той час як schemas містить деталізовані визначення схем баз даних і таблиць, створені за допомогою PgAdmin і оптимізовані для продуктивності та надійності [36].

Файли main.py та create_database.py запускають основний цикл роботи бота, ініціюють базу даних і запроваджують тестування. Вони як рушійні сили проекту спрямовують ключові операції бота від старту до моніторингу робочих процесів.

Файли конфігурації залежностей, poetry.lock та pyproject.toml, забезпечують зв'язність та однорідність розробницького середовища, гарантуючи, що всі розробники працюють з однаковим набором інструментів та бібліотек, що вкрай важливо для ефективної командної роботи.

Все це, разом з детально розробленою документацією в README.md, формує проект, який легко читати, оновлювати та масштабувати, створюючи міцний фундамент для майбутніх розширень і покращень функціональності нашого чат-бота.

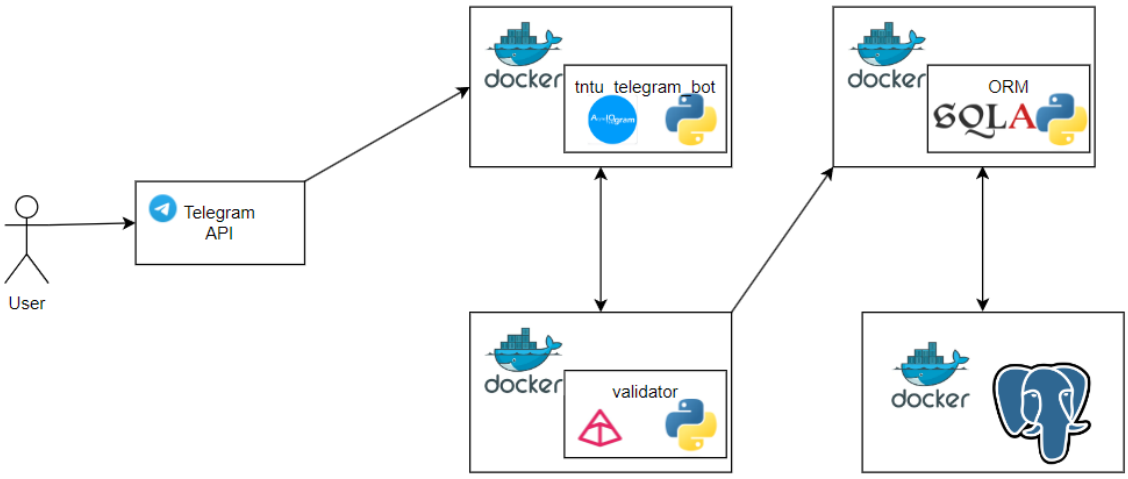

Рисунок 3.7 – Архітектура телеграм-бота tntu_internet з використанням Docker

Діаграма відображає архітектуру системи, що зображена на рисунку 3.7, де кожен компонент телеграм-бота tntu_internet ізольований у власному Docker контейнері, що забезпечує модульність і легкість в розгортанні. Користувач взаємодіє з Telegram API, яке в свою чергу спілкується з основним сервісом бота, реалізованим на Python за допомогою бібліотеки Aiogram [37].

Для обробки даних і валідації використовується окремий сервіс validator, який також запущений у Docker контейнері. ORM SQLAlchemy у своєму контейнері забезпечує зв'язок з базою даних, розміщеною у PostgreSQL контейнері, виконуючи роль інтерфейсу для виконання запитів до бази даних без необхідності написання чистого SQL.

Ця діаграма ілюструє чітке розмежування відповідальності між сервісами, що сприяє більшій ефективності розробки та підтримки, і показує, як використання контейнеризації може поліпшити масштабування і впровадження бота.

3.5 Висновок до третього розділу

У третьому розділі було здійснено тестування та оцінено експлуатацію програмного інтерфейсу телеграм-бота tntu_internet. Поєднання теоретичних засад проєктування бази даних та практичної її реалізації забезпечило створення надійного та ефективного засобу комунікації для студентів та співробітників ТНТУ ім. І. Пулюя.

Ретельно спланована структура БД і налаштовані взаємозв'язки між сутностями виявилися ключовими для забезпечення безперебійної роботи телеграм-бота. Процес валідування вхідних даних, втілений у боті, показав свою ефективність у попередженні помилок та забезпеченні відповідності даних вимогам безпеки.

Етапи тестування готового рішення включали перевірку інтерфейсу бота на інтуїтивність, швидкість відгуку та зручність використання, з акцентом на якість обслуговування користувачів.

Процеси оплати через інтерфейс телеграм-бота та подальша передача квитанцій відбувалися без зайвих ускладнень, що сприяло збільшенню задоволення користувачів сервісом. Передбачена можливість обрати з різноманіття платіжних систем дозволяє здійснювати оплату відповідно до особистих переваг.

Ієрархія проекту, втілена у чіткій структурі папок і файлах, спрощує подальше управління кодом та його масштабування. Використання Docker для розгортання компонентів телеграм-бота гарантує високий рівень модульності та портативності, підвищуючи зручність впровадження та експлуатації рішення.

Таким чином, тестування підтвердило, що телеграм-бот tntu_internet відповідає усім технічним та функціональним вимогам і готовий до впровадження в університетську інфраструктуру, надаючи студентам та співробітникам надійний та зручний інструмент для комунікації та управління їхніми запитами.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Укриття людей в захисних спорудах та евакуаційні заходи в період загрози

Для вирішення питань щодо укриття населення в ЗС ЦЗ органами виконавчої влади всіх рівнів та суб'єктами господарювання створюється фонд таких споруд.

До фонду ЗС ЦЗ належать:

- сховища – герметичні споруди для захисту людей, в яких протягом певного часу створюються умови, що виключають вплив на них небезпечних факторів, які виникають внаслідок НС, воєнних (бойових) дій та терористичних актів;

- протирадіаційні укриття – негерметичні споруди для захисту людей, в яких створюються умови, що виключають вплив на них іонізуючого опромінення у разі радіоактивного забруднення місцевості;

- швидкоспоруджувані ЗС ЦЗ – захисні споруди, що зводяться із спеціальних конструкцій за короткий час для захисту людей від дії засобів ураження в особливий період; для захисту людей від деяких факторів небезпеки, що виникають внаслідок НС у мирний час, та дії засобів ураження в особливий період також використовуються споруди подвійного призначення та найпростіші укриття (пристосовані приміщення);

- споруди подвійного призначення – це наземні або підземні споруди, що можуть бути використані за основним функціональним призначенням і для захисту населення;

- найпростіші укриття – це фортифікаційні споруди, цокольні або підвальні приміщення, що знижують комбіноване ураження людей від небезпечних наслідків НС, а також від дії засобів ураження в особливий період.

Проектування, будівництво, пристосування і розміщення ЗС ЦЗ та об'єктів подвійного призначення здійснюються згідно з нормами, які розробляються

відповідно до Закону України «Про будівельні норми». Будівництво ЗС ЦЗ і їх утримання потребує багато часу і коштів. Тому ведеться накопичення фонду ЗС ЦЗ. Шляхи накопичення:

- будівництво сховищ одночасно з будівництвом нових підприємств, розрахованих на укриття робітників найбільшої працюючої зміни;
- будівництво окремих сховищ та протирадіаційних укриттів;
- використання метрополітену підземного пролягання;
- обладнання сховищ в підземних та інших заглиблених приміщеннях існуючих будівель і споруд;
- пристосування і використання частини приміщень освоєного підземного простору міст для захисту населення;
- використання гірничих виробок і природних порожнин;
- масове будівництво найпростіших сховищ і укриттів в період загрози виникнення надзвичайних ситуацій за скорочені терміни (3-6 діб).

Наявний фонд захисних споруд в повсякденних умовах життєдіяльності використовується для господарських, культурних і побутових потреб у порядку, який забезпечує використання їх за прямим призначенням в установлений короткий термін.

Як спосіб захисту, це передбачає своєчасне евакуювання населення із зон потенційного ураження, катастрофічного затоплення, забруднення або зараження до безпечних районів для тимчасового або постійного проживання.

В умовах неповного забезпечення захисними спорудами в містах та інших населених пунктах, що мають об’єкти підвищеної небезпеки, а також на випадок війни евакуація є основним способом захисту населення і проведення її планується і готується заздалегідь. Залежно від обстановки, яка склалася на час НС, може бути загальна або часткова евакуація. Загальна евакуація проводиться для всіх категорій населення і планується на випадок війни, можливого небезпечного радіоактивного забруднення територій навколо атомних електростанцій, виникнення загрози катастрофічного затоплення місцевості з

чотиригодинним добіганням проривної хвилі, лісових і торф'яних пожежах, інших явищ з тяжким наслідками, що загрожують населеним пунктам.

Під час проведення часткової евакуації завчасно вивозиться не зайняте у сфері виробництва та обслуговування населення: діти, учні навчальних закладів, вихованці дитячих будинків разом з викладачами та вихователями, студенти, пенсіонери та інваліди, які утримуються у будинку для осіб похилого віку разом з обслуговуючим персоналом та членами їх сімей.

Проведення евакуації забезпечується шляхом:

- утворення регіональних, місцевих та об'єктових органів з евакуації;
- планування евакуації;
- визначення безпечних районів, придатних для розміщення
- евакуйованого населення та майна;
- організації оповіщення керівників суб'єктів господарювання і
- населення про початок евакуації;
- організації управління евакуацією;
- життєзабезпечення евакуйованого населення в місцях їх безпечного
- розміщення;
- навчання населення діям під час проведення евакуації.

Залежно від обстановки, яка склалася на час НС, може бути загальна або часткова евакуація. Загальна евакуація проводиться для всіх категорій населення і планується на випадок війни, можливого небезпечного радіоактивного забруднення територій навколо атомних електростанцій, виникнення загрози катастрофічного затоплення місцевості, лісових і торф'яних пожеж, інших явищ з тяжкими наслідками, що загрожують населеним пунктам. Під час проведення часткової евакуації завчасно вивозяться діти, учні навчальних закладів, вихованці дитячих будинків разом з викладачами та вихователями, студенти, пенсіонери та інваліди, які утримуються у будинках для осіб похилого віку разом з обслуговуючим персоналом та членами їх сімей.

Процес евакуації через повітряну тривогу полягає у своєчасному оповіщенні населення про небезпеку та організованому переміщенні до захисних

споруд або безпечних районів. Це включає координацію дій рятувальних служб, забезпечення транспортними засобами, а також контроль за дотриманням правил безпеки під час евакуації. Важливо, щоб населення було підготовлено до швидкого реагування та знало місця розташування найближчих укриттів.

4.2 Аналіз впливу чинників на зорову працездатність людини

В процесі праці і загалом всього життя, людина отримує 90 % інформації за допомогою зору. Тому умови праці та безпека виконання робіт суттєво залежить від зорового сприйняття [46].

Невідповідність світлового середовища функціонального стану працівника призводить до значних порушень здоров’я та травматизму. За даними Всесвітньої організації охорони здоров'я (ВООЗ), у світі 150 млн. хворих зі значним зниженням зорових функцій. За останні 20 років кількість сліпих зросло на 12 млн.

Особливе місце серед захворювань ока посідає атрофія зорового нерву, що значно знижує зорові функції людини працездатного віку. За даними Є. С. Лібмана, ця патологія посідає одне з провідних місць у нозологічній структурі сліпоти та слабкозорості, поступаючись лише глаукомі та дегенеративній короткозорості. Згідно зі статистикою ВООЗ, за останні десятиріччя рівень інвалідності пацієнтів з атрофією зорового нерва підвищився удвічі.

В Україні останніми роками розширилась група соціально значущих захворювань ока, за рахунок катаракти, глаукоми, хвороби зорового нерву та сітківки, які найчастіше є причинами сліпоти. Серед таких захворювань превалюють саме атрофії зорового нерву, що пов'язане з високою напруженістю зорового сприйняття, яке є змістом діяльності операторів (менеджерів). За останні десять років атрофія зорового нерву в Україні зросла з 73,6 до 84,6 на 100 тис. населення і за темпами збільшення посіла одне з перших місць, що свідчить про недосконаліть вибору систем освітлення на робочих місцях. Згідно досліджень медиків, атрофія зорового нерву є наслідком низки патологічних

процесів, що впливають на різні ділянки зорового аналізатора – від гангліозних клітин сітківки до зорової кори головного мозку, подана інформація зображена на рисунку 4.1.

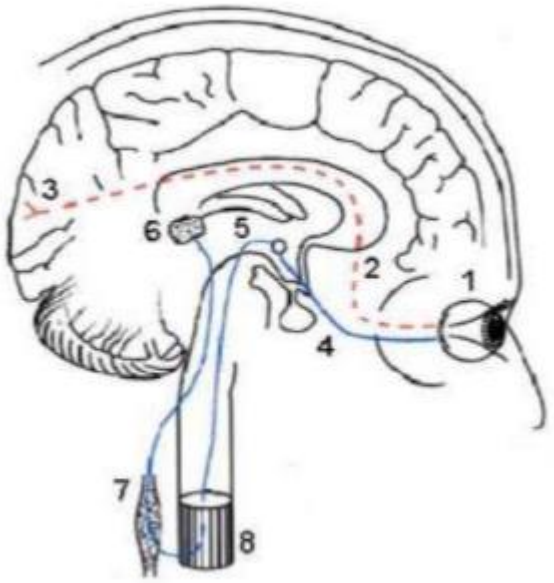

Рисунок 4.1 – Структура мозку яка забезпечує вплив світла на функціонування різноманітних систем організму: 1 – ретина; 2 – оптичний нерв; 3 – зорова кора головного мозку; 4 – ретиногіпоталамусний тракт; 5 – супрахіазматичне ядро; 6 – шишковидна залоза; 7 – спинномозковий ствол; 8 – верхній шийний нервовий вузол; - - - - - - - - зоровий шлях; ———— фотобіологічний шлях

Серед етіологічних факторів переважають захворювання центральної нервової системи (об'ємні процеси головного мозку, запальні захворювання мозку, черепномозкові травми, патологічні процеси в зоровому нерві і сітківці (запалення, дистрофія, порушення кровопостачання, дія токсинів), загальні захворювання (атеросклероз, гіпертонічна хвороба, цукровий діабет), спадкові фактори.

Дисфункція зорового сприйняття за останні роки пов’язана також із погіршенням екологічного середовища. Доведено збільшення захворюваності на атрофію зорового нерву при хімічному забрудненні питної води (солями важких

металів, синтетичними поверхневоактивними речовинами, пестицидами), а також внаслідок впливу електромагнітних полів (високої частоти, електростатичних, постійного магнітного поля, електромагнітного поля промислової частоти, полів радіолокаційних станцій). Умови праці на кожному робочому місці формуються чинниками виробничого середовища та трудового процесу. Згідно проведеного аналізу змісту роботи працівників (операторів та апаратників, що виконують цілу низку технологічних операцій при виробництві будівельних виробів і матеріалів на підприємствах будівельної індустрії та при будівництві з застосуванням цілого ряду машин та устаткування) визначено, що за останні роки у зв'язку зі зміною технологій та автоматизацією технологічних процесів у виробництві різних галузей збільшилась частка операторів-фахівців, професіоналів, технічних службовців, робітників з обслуговування, експлуатації та контролю технологічного устаткування та машин. Для яких особливо важливим є створення безпечного світлового середовища.

4.3 Висновки до четвертого розділу

В розділі "Безпека життєдіяльності, основи охорони праці" розглянуто ключові аспекти захисту населення та забезпечення безпеки працівників в умовах надзвичайних ситуацій. Особливу увагу приділено питанням укриття людей у спеціальних захисних спорудах та проведенню евакуаційних заходів у разі повітряної тривоги. Було детально описано різновиди захисних споруд, їхні функції та процеси організації і підтримки в готовності до використання.

Також, розглянуто важливість планування та підготовки евакуаційних заходів, що включають координацію дій рятувальних служб, оповіщення населення та забезпечення транспортними засобами. Наголошено на необхідності швидкого реагування та знання населенням місць розташування найближчих укриттів для своєчасного захисту від небезпеки.

Запропоновано заходи для поліпшення умов праці та зменшення ризиків, пов'язаних з впливом шкідливих чинників.

ВИСНОВКИ

У кваліфікаційній роботі освітнього рівня “Бакалавр” було досягнуто мети розробки телеграм-бота tntu_internet для покращення адміністрування інтернет-з'єднання у ТНТУ ім. І. Пулюя. На основі проведених досліджень та аналізу було визначено ключові аспекти створення ефективного, зручного та надійного інструменту для взаємодії студентів та співробітників університету з адміністрацією.

В першому розділі кваліфікаційної роботи освітнього рівня “Бакалавр”:

- Подано аналіз існуючого рішення та визначено проблеми, що виникають при адмініструванні інтернет-з'єднання.
- Розглянуто потреби користувачів та визначено вимоги до функціональності телеграм-бота.
- Обґрунтовано вибір технологій для розробки телеграм-бота.
- Сформовано діаграму варіантів використання для акторів телеграм-бота.

В другому розділі кваліфікаційної роботи:

- Розроблено структуру бази даних для телеграм-бота tntu_internet.
- Спроєктовано процес реєстрації користувачів та адміністраторів з урахуванням безпеки.
- Описано архітектуру телеграм-бота, яка забезпечує ефективну взаємодію між користувачами та системою.

У третьому розділі кваліфікаційної роботи:

- Реалізовано схему бази даних для телеграм-бота.
- Протестовано функціональність телеграм-бота, включаючи процеси реєстрації, взаємодії з користувачами та обробки запитів.

У розділі “Безпека життєдіяльності, основи охорони праці” висвітлено укриття людей у захисних спорудах та евакуаційні заходи в період загрози, а також проаналізовано вплив чинників на зорову працездатність людини та заходи для поліпшення умов праці.

ПЕРЕЛІК ДЖЕРЕЛ

1 Послуги інтернету [Електронний ресурс] // ТНТУ. – 2023. – Режим доступу до ресурсу: https://tntu.edu.ua/?p=uk/services/internet.

2 Build a Telegram Bot With Payments and a Database From A-Z [Електронний ресурс] // Medium. – 2022. – Режим доступу до ресурсу: https://medium.com/@tr_18329/build-a-telegram-bot-with-payments-and-a-database-from-a-z-8f54ee1e1ecf.

3 Telegram bot security solutions [Електронний ресурс] // Medium. – 2023. – Режим доступу до ресурсу: https://medium.com/@bioogrami7/telegram-bot-security-solutions-6c5b8105a0e1.

4 User Authorization [Електронний ресурс] // core.telegram. – 2020. – Режим доступу до ресурсу: https://core.telegram.org/api/auth.

5 Adaptive telegram bot [Електронний ресурс] // Umnico. – 2023. – Режим доступу до ресурсу: https://umnico.com/blog/telegram-bots/.

6 Add new functionallity [Електронний ресурс] // Medium. – 2024. – Режим доступу до ресурсу: https://medium.com/@moraneus/building-telegram-bot-with-python-telegram-bot-a-comprehensive-guide7e33f014dc79.

7 Top 10 Python Libraries to Create Your Telegram Bot Easily (GitHub) [Електронний ресурс] // finxter. – 2023. – Режим доступу до ресурсу: https://blog.finxter.com/top-10-python-libraries-to-create-your-telegram-bot-easily-github/.

8 Telegraf VS Node-Telegram-Bot-API [Електронний ресурс] // dev.to. – 2022. – Режим доступу до ресурсу: https://dev.to/maklut/telegraf-vs-node-telegram-bot-api-36fk.

9 Add postgres to docker compose and dokku infrastructure for telegram bot in NestJS [Електронний ресурс] // DEV. – 2022. – Режим доступу до ресурсу: https://dev.to/endykaufman/add-postgres-to-docker-compose-and-dokku-infrastructure-for-telegram-bot-in-nestjs-23ih.

10 Aiogram [Електронний ресурс] // aiogram. – 2024. – Режим доступу до ресурсу: https://aiogram.dev/.

11 Making Telegram Bots with Python [Електронний ресурс] // dev. – 2022. – Режим доступу до ресурсу: https://dev.to/tbhaxor/making-telegram-bots-with-python-hid.

12 Політика конфіденційності Telegram [Електронний ресурс] // Telegram. – 2023. – Режим доступу до ресурсу: https://telegram.org/privacy/ua.

13 Build Your Telegram bot with Python and SQLAlchemy [Електронний ресурс] // Medium. – 2023. – Режим доступу до ресурсу: https://medium.com/@arian.shariat/build-your-telegram-bot-with-python-and-sqlalchemy-b4fec440c456.

14 Creating a Telegram bot with a connected database [Електронний ресурс] // KWORK. – 2024. – Режим доступу до ресурсу: https://kwork.com/chatbots/28470021/creating-a-telegram-bot-with-a-connected-database.

15 Creation of a Telegram Bot along with database [Електронний ресурс] // Freelancer. – 2023. – Режим доступу до ресурсу: https://www.freelancer.com/projects/php/creation-telegram-bot-along-with.

16 Using SELECT Statements [Електронний ресурс] // SQLAlchemy. – 2024. – Режим доступу до ресурсу: https://docs.sqlalchemy.org/en/20/tutorial/data_select.html.

17 Telegram Bot JSON API to Read and Write [Електронний ресурс] // NoCodeAPI. – 2022. – Режим доступу до ресурсу: https://nocodeapi.com/marketplace/telegram-bot/.

18 Python telegram bot registration script [Електронний ресурс] // stackoverflow. – 2023. – Режим доступу до ресурсу: https://stackoverflow.com/questions/75760276/python-telegram-bot-registration-script.

19 Шифрування даних в Telegram [Електронний ресурс] // Medium. – 2023. – Режим доступу до ресурсу: https://medium.com/@BrandScribe.agency/шифрування-даних-в-telegram-b419f985e47

20 REGISTRATION IN TELEGRAM AND VERIFICATION BY CALL FOR DIFFERENT COUNTRIES [Електронний ресурс] // Medium. – 2022. – Режим

доступу до ресурсу: https://medium.com/@jbstudio/registration-in-telegram-and-verification-by-call-for-different-countries-5abf5d834319.

21 Recaptcha [Електронний ресурс] // Google. – 2022. – Режим доступу до ресурсу: https://www.google.com/recaptcha/about/.

22 Two design patterns for Telegram Bots [Електронний ресурс] // Dev. – 2021. – Режим доступу до ресурсу: https://dev.to/madhead/two-design-patterns-for-telegram-bots-59f5.

23 How to build a reliable, scalable, and cost-effective Telegram bot [Електронний ресурс] // Medium. – 2022. – Режим доступу до ресурсу: https://medium.com/wearewaes/how-to-build-a-reliable-scalable-and-cost-effective-telegram-bot-58ae2d6684b1.

24 How to Dockerize a Telegram Bot: A Step-by-Step Guide [Електронний ресурс] // Medium. – 2024. – Режим доступу до ресурсу: https://tjtanjin.medium.com/how-to-dockerize-a-telegram-bot-a-step-by-step-guide-b14bc427f5dc.

25 How to pay via Telegram Pay [Електронний ресурс] // TRANZZO. – 2024. – Режим доступу до ресурсу: https://tranzzo.com/telegram-pay.

26 Telegram-bot-api vulnerabilities [Електронний ресурс] // Snyk security. – 2024. – Режим доступу до ресурсу: https://security.snyk.io/package/npm/telegram-bot-api.

27 10 Common Web Security Vulnerabilities [Електронний ресурс] // Toptal. – 2024. – Режим доступу до ресурсу: https://www.toptal.com/cybersecurity/10-most-common-web-security-vulnerabilities.

28 Horizontal Vs. Vertical Scaling: How Do They Compare? [Електронний ресурс] // CLOUDZERO. – 2023. – Режим доступу до ресурсу: https://www.cloudzero.com/blog/horizontal-vs-vertical-scaling/.

29 Масштабованість хмари: горизонтальне та вертикальне масштабування IT-інфраструктур [Електронний ресурс] // Cases.media. – 2024. – Режим доступу до ресурсу: https://cases.media/article/masshtabovanist-khmari-gorizontalne-ta-vertikalne-masshtabuvannya-it-infrastruktur.

30 Creating a Telegram Bot with Python - Hosted in a Docker Container [Електронний ресурс] // YouTube. – 2023. – Режим доступу до ресурсу: https://www.youtube.com/watch?v=zMAIzj6FcRs&ab_channel=AlessioMancinelli.

31 Create a Telegram Bot, from Development to Deployment [Електронний ресурс] // The programmer Chest. – 2019. – Режим доступу до ресурсу: https://elbauldelprogramador.com/en/create-deploy-telegram-bot/.

32 A Step-by-Step Guide to Data Validation in Python [Електронний ресурс] // Medium. – 2023. – Режим доступу до ресурсу: https://medium.com/@shuklamansi800/a-step-by-step-guide-to-data-validation-in-python-83d87871f04.

33 Database Validation [Електронний ресурс] // Teach computer science. – 2024. – Режим доступу до ресурсу: https://teachcomputerscience.com/database-validation/.

34 How to Test a Telegram Bot [Електронний ресурс] // Medium. – 2020. – Режим доступу до ресурсу: https://medium.com/@duketemon/how-to-test-a-telegram-bot-ba54eb1cadc0.

35 Які технології шифрування використовуються в Telegram? [Електронний ресурс] // Telegramm. – 2024. – Режим доступу до ресурсу: https://telegramm.com.ua/pytannya-ta-vidpovidi/yaki-tekhnologiji-shyfruvannya-vykorystovuyutsya-v-telegram/.

36 Lesson 4 - Quick Review of Project Structure in Telegram Bot Development with Aiogram [Електронний ресурс] // YouTube. – 2024. – Режим доступу до ресурсу: https://www.youtube.com/watch?v=Pj2DH4EKdAA&ab_channel=Programmingforbeginners.Python.JavaScript.

37 How to create a Telegram bot using Docker and host it on Azure [Електронний ресурс] // Blog Danielabg. – 2023. – Режим доступу до ресурсу: https://blog.danielabg.com/how-to-create-a-telegram-bot-using-docker-and-host-it-on-azure.

38 V. Kozlovskyi, Y. Balanyuk, H. Martyniuk, O. Nazarevych, L. Scherbak and G. Shymchuk, «Information Technology for Estimating City Gas Consumption During the Year,» 2022 International Conference on Smart Information Systems and

Technologies (SIST), Nur-Sultan, Kazakhstan, 2022, pp. 1-4, doi: 10.1109/SIST54437.2022.9945786.

39 Approach to gas consumption process forecasting on the basis of a mathematical model in the form of a random cyclic process / Serhii Lupenko, Iaroslav Lytvynenko, Oleg Nazarevych, Grigorii Shymchuk, Volodymyr Hotovych // ICAAEIT 2021, 15-17 December 2021. – Tern. : TNTU, Zhytomyr «Publishing house „Book-Druk“» LLC, 2021. – P. 213–219. – (Mathematical modeling in power engineering and information technologies).

40 Lytvynenko, S. Lupenko, O. Nazarevych, G. Shymchuk and V. Hotovych, «Mathematical model of gas consumption process in the form of cyclic random process,» 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), LVIV, Ukraine, 2021, pp. 232-235, doi: 10.1109/CSIT52700.2021.9648621.

41 Additive mathematical model of gas consumption process / Iaroslav Lytvynenko, Serhii Lupenko, Oleh Nazarevych, Hryhorii Shymchuk, Volodymyr Hotovych // Scientific Journal of TNTU. – Tern. : TNTU, 2021. – Vol 104. – No 4. – P. 87–97.

42 O. Nazarevych, Y. Leshchyshyn, S. Lupenko, V. Hotovych, G. Shymchuk and N. Shabliy, «Method of Gas Consumption Change-point Detection Based on Seasonally Multicomponent Model,» 2020 10th International Conference on Advanced Computer Information Technologies (ACIT), Deggendorf, Germany, 2020, pp. 152-155, doi: 10.1109/ACIT49673.2020.9208924.

43 Y. Leshchyshyn, L. Scherbak, O. Nazarevych, V. Gotovych, P. Tymkiv and G. Shymchuk, «Multicomponent Model of the Heart Rate Variability Change-point,» 2019 IEEE XVth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH), Polyana, Ukraine, 2019, pp. 110-113, doi: 10.1109/MEMSTECH.2019.8817379.

44 Стручок В. С. “Техноекологія та цивільна безпека. Частина «Цивільна безпека» Навчальний посібник [Електронний ресурс] / Володимир Сергійович Стручок // Тернопільський національний технічний університет

імені Івана Пулюя. – 2022. – Режим доступу до ресурсу: https://elartu.tntu.edu.ua/bitstream/lib/39424/1/lib/39424/1/Навчальн%20посібник.%20Техноекологія%20та%20цивільна%20безпека..pdf

45 Рабенко Р. С. Аналіз впливу чинників на зорову працездатність [Електронний ресурс] / Р. С. Рабенко, І. В. Мещерякова // XVII Міжвузівська студентська науко во-практична конференція, ПДАБА, 8 – 9 квітня 2021 р. – 2021. – Режим доступу до ресурсу: https://jm.snau.edu.ua/wp-content/uploads/2021/04/zbirnik-tez-dopov.-stud.konf.bzhd-2021-pdaba-1.pdf.

46 Жидецький В. Ц. Основи охорони праці / В. Ц. Жидецький. – Львів: Афіша, 2005. – 319 с.

ДОДАТКИ

Додаток А

Програмний код для запуску телеграм-бота tntu_internet

```
import asyncio
import logging

import sys

from pydantic import BaseModel
from bot.config import dp, bot


class User(BaseModel):
    telegram_id: int
    name: str
    surname: str
    phone: str
    hostel: int
    room: str


async def main() -> None:
    await dp.start_polling(bot)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, stream=sys.stdout)
    asyncio.run(main())
```

Додаток Б

Програмний код для створення схеми бази даних для телеграм-бота tntu_internet

```
from sqlalchemy import create_engine, text

from config import config
from models.base import Base

engine = create_engine(config.DB_CONN_URL)

with engine.connect() as connection:
    connection.execute(text("CREATE    SCHEMA    IF    NOT    EXISTS
telegram_data_schema"))
    connection.commit()

Base.metadata.create_all(engine)
```

Додаток В

Програмний код для створення схеми бази даних для телеграм-бота tntu_internet

Лістинг В.1 – Код файлу «admin.py»

```
import uuid

from sqlalchemy import Column, BigInteger, Text, Boolean
from sqlalchemy.dialects.postgresql import UUID

from models.base import Base
from ..schema import SCHEMA_NAME


class Admin(Base):
    __tablename__ = 'admin'
    __table_args__ = {"schema": SCHEMA_NAME}
    id = Column(UUID, primary_key=True, default=uuid.uuid4())
    telegram_id = Column(BigInteger, nullable=False, unique=True)
    telegram_username = Column(Text, nullable=False, unique=True)
    receive_payment = Column(Boolean, default=False)
    receive_internet_problems = Column(Boolean, default=False)
```

Лістинг В.2 – Код файлу «block.py»

```
import uuid

from sqlalchemy import Column, String, ForeignKey
from sqlalchemy.dialects.postgresql import UUID
from sqlalchemy.orm import relationship

from models.base import Base
from ..schema import SCHEMA_NAME


class Block(Base):
    __tablename__ = 'block'
    __table_args__ = {"schema": SCHEMA_NAME}
    id = Column(UUID, primary_key=True, default=uuid.uuid4())
    number = Column(String(5), nullable=False)
    wing_id = Column(UUID, ForeignKey(f'{SCHEMA_NAME}.wing.id'),
nullable=False)
    wing = relationship("Wing", back_populates="blocks")
    rooms = relationship("Room", back_populates="block")
```

Лістинг В.3 – Код файлу «hostel.py»

```
import uuid

from sqlalchemy import Column, Integer
from sqlalchemy.dialects.postgresql import UUID
from sqlalchemy.orm import relationship

from models.base import Base
from ..schema import SCHEMA_NAME


class Hostel(Base):
    __tablename__ = 'hostel'
    __table_args__ = {"schema": SCHEMA_NAME}
    id = Column(UUID, primary_key=True, default=uuid.uuid4())
    number = Column(Integer, nullable=False)
    wings = relationship("Wing", back_populates="hostel")
```

Лістинг В.4 – Код файлу «payment.py»

```
import uuid

from sqlalchemy import Column, Text, Boolean, ForeignKey, TIMESTAMP
from sqlalchemy.dialects.postgresql import UUID
from sqlalchemy.orm import relationship
from sqlalchemy.sql import func

from models.base import Base
from ..schema import SCHEMA_NAME


class Payment(Base):
    __tablename__ = 'payment'
    __table_args__ = {"schema": SCHEMA_NAME}
    id = Column(UUID, primary_key=True, default=uuid.uuid4())
    send_date = Column(TIMESTAMP, default=func.now())
    file_id = Column(Text, nullable=False)
    room_id  =  Column(UUID,  ForeignKey(f'{SCHEMA_NAME}.room.id'),
nullable=False)
    accepted = Column(Boolean, default=False)
    file = Column(Boolean, default=False)
    photo = Column(Boolean, default=False)
    room = relationship("Room", back_populates="payment")
```

Лістинг В.5 – Код файлу «person.py»

```
import uuid

from sqlalchemy import Column, BigInteger, Text, ForeignKey,
TIMESTAMP, String
from sqlalchemy.dialects.postgresql import UUID
from sqlalchemy.orm import relationship
from sqlalchemy.sql import func
from models.base import Base
```

```
from ..schema import SCHEMA_NAME


class Person(Base):
    __tablename__ = 'person'
    __table_args__ = {"schema": SCHEMA_NAME}
    id = Column(UUID, primary_key=True, default=uuid.uuid4)
    telegram_id = Column(BigInteger, nullable=False, unique=True)
    name = Column(String(50), nullable=False)
    surname = Column(String(50), nullable=False)
    phone = Column(String(16), nullable=False)
    room_id = Column(UUID, ForeignKey(f'{SCHEMA_NAME}.room.id'),
nullable=False)
    registration_date = Column(TIMESTAMP, default=func.now())
    room = relationship("Room", back_populates="people")
    task_manager_entries = relationship("TaskManager",
back_populates="people")
```

Лістинг В.6 – Код файлу «room.py»

```
import uuid

from sqlalchemy import Column, Text, ForeignKey, String
from sqlalchemy.dialects.postgresql import UUID
from sqlalchemy.orm import relationship

from models.base import Base
from ..schema import SCHEMA_NAME


class Room(Base):
    __tablename__ = 'room'
    __table_args__ = {"schema": SCHEMA_NAME}
    id = Column(UUID, primary_key=True, default=uuid.uuid4())
    name = Column(String(5), nullable=False)
    block_id = Column(UUID, ForeignKey(f'{SCHEMA_NAME}.block.id'),
nullable=False)
    people = relationship("Person", back_populates="room")
    block = relationship("Block", back_populates="rooms")
    payment = relationship("Payment", back_populates="room")
    special_room_ips = relationship("SpecialRoomIp",
back_populates="room")
```

Лістинг В.7 – Код файлу «special_room_ip.py»

```
import uuid

from sqlalchemy import Column, Integer, ForeignKey
from sqlalchemy.dialects.postgresql import UUID
from sqlalchemy.orm import relationship

from models.base import Base
from ..schema import SCHEMA_NAME
```

```
class SpecialRoomIp(Base):
    __tablename__ = 'special_room_ip'
    __table_args__ = {"schema": SCHEMA_NAME}
    id = Column(UUID, primary_key=True, default=uuid.uuid4())
    room_id = Column(UUID, ForeignKey(f'{SCHEMA_NAME}.room.id'),
nullable=False)
    node_address = Column(Integer, nullable=False, unique=True)
    room = relationship("Room", back_populates="special_room_ips")
```

Лістинг В.8 – Код файлу «task_manager.py»

```
import uuid

from sqlalchemy import Column, Boolean, ForeignKey, TIMESTAMP
from sqlalchemy.dialects.postgresql import UUID
from sqlalchemy.sql import func
from sqlalchemy.orm import relationship

from models.base import Base
from ..schema import SCHEMA_NAME


class TaskManager(Base):
    __tablename__ = 'task_manager'
    __table_args__ = {"schema": SCHEMA_NAME}
    id = Column(UUID, primary_key=True, default=uuid.uuid4())
    person_id = Column(UUID,
ForeignKey(f'{SCHEMA_NAME}.person.id'), nullable=False)
    date = Column(TIMESTAMP, default=func.now())
    damaged_cabel = Column(Boolean)
    correct_settings = Column(Boolean)
    block_internet = Column(Boolean)
    indicator = Column(Boolean)
    connect_gateway = Column(Boolean)
    special_ip = Column(Boolean)
    active = Column(Boolean, default=True)
    people = relationship("Person",
back_populates="task_manager_entries")
```

Лістинг В.9 – Код файлу «wing.py»

```
import uuid

from sqlalchemy import Column, Integer, Text, ForeignKey
from sqlalchemy.dialects.postgresql import UUID
from sqlalchemy.orm import relationship

from models.base import Base
from ..schema import SCHEMA_NAME
```

```
class Wing(Base):
    __tablename__ = 'wing'
    __table_args__ = {"schema": SCHEMA_NAME}
    id = Column(UUID, primary_key=True, default=uuid.uuid4())
    name = Column(Text, nullable=False)
    hostel_id = Column(UUID,
ForeignKey(f'{SCHEMA_NAME}.hostel.id'), nullable=False)
    subnet_address = Column(Integer, nullable=False)
    hostel = relationship("Hostel", back_populates="wings")
    blocks = relationship("Block", back_populates="wing")
```

Додаток Д

Програмний код для валідації введених даних у телеграм-боті tntu_internet

```
import re
from functools import wraps


def validate_name(name: str) -> True | False:
    return bool(re.match(r"\b([A-ZÀ-ÿA-ЩЬЮЯҐЄІЇ][-,a-z. 'A-
щьюяґєіїÀ-ÿ]{0,49} *)+", name))


def validate_phone_number(phone_number):
    return bool(re.match(r"^\+?3?8?(0\d{9})$", phone_number))


def validate_room_name(room_name):
    return bool(re.match(r"^\d{1,4}[a-яA-Яa-z]?$", room_name))


def validate_params(params_to_check, validation_functions):
    def decorator(func):
        @wraps(func)
        def wrapper(*args, **kwargs):
            for param, validation_function in zip(params_to_check,
validation_functions):
                if param in kwargs and not
validation_function(kwargs[param]):
                    return False
            return func(*args, **kwargs)

        return wrapper

    return decorator
```

Додаток Е

Програмний код для управління користувачами та адміністраторами у телеграм-боті tntu_internet

```
import uuid
from typing import List, Tuple

from sqlalchemy import func, Integer
from sqlalchemy.orm import Session
from sqlalchemy.orm import joinedload

from logic.validations import validate_params, validate_room_name
from models import Wing, Hostel, Person, Room, Block, TaskManager,
Payment, SpecialRoomIp, Admin


def user_registration(session: Session, telegram_id: int,
first_name: str, last_name: str,
                      phone_number: str, hostel_number: int,
room_name: str) -> None:
    room_id = session.query(
        Room
    ).join(
        Block
    ).join(
        Wing
    ).join(
        Hostel
    ).where(
        Hostel.number == hostel_number
    ).where(
        Room.name == room_name
    ).first().id
    person = Person(telegram_id=telegram_id, name=first_name,
surname=last_name, phone=phone_number,
                    room_id=room_id)
    session.add(person)
    session.commit()


def update_user(session: Session, telegram_id: int,
new_first_name: str = None, new_last_name: str = None,
                new_phone_number: str = None, new_hostel_number:
int = None, new_room_number: str = None) -> None:
    user = session.query(Person).filter(Person.telegram_id ==
telegram_id).first()
    if new_first_name:
        user.name = new_first_name
    if new_last_name:
        user.surname = new_last_name
```

```
    if new_phone_number:
        user.phone = new_phone_number
    if new_room_number:
        if not new_hostel_number:
            new_hostel_number = session.query(
                Hostel
            ).join(
                Wing
            ).join(
                Block
            ).join(
                Room
            ).join(
                Person
            ).where(
                Person.telegram_id == telegram_id
            ).one().number
        user.room_id = session.query(
            Room
        ).join(
            Block
        ).join(
            Wing
        ).join(
            Hostel
        ).where(
            Hostel.number == new_hostel_number
        ).where(
            Room.name == new_room_number
        ).one().id
    session.commit()


def get_user_info(session: Session, telegram_id: int) -> dict:
    user_info = session.query(
        Person, Room, Block, Wing, Hostel
    ).select_from(
        Person
    ).join(
        Room
    ).join(
        Block
    ).join(
        Wing
    ).join(
        Hostel
    ).filter(
        Person.telegram_id == telegram_id
    ).options(

joinedload(Person.room).joinedload(Room.block).joinedload(Block.wi
ng).joinedload(Wing.hostel)
    ).first()
```

```
    if not user_info:
        return {}

    person, room, block, wing, hostel = user_info

    return {
        'telegram_id': person.telegram_id,
        'first_name': person.name,
        'last_name': person.surname,
        'phone_number': person.phone,
        'hostel_number': hostel.number,
        'wing_name': wing.name,
        'block_number': block.number,
        'room_number': room.name
    }


def get_admin_info(session: Session) -> List[Tuple[int]]:
    return session.query(Admin.telegram_id).all()


def hostel_exist(session: Session, hostel_number: int) -> bool:
    if session.query(Hostel).where(Hostel.number ==
hostel_number).all():
        return True


def room_exist(session: Session, hostel_number: int, room_name:
str) -> bool:
    if hostel_exist(session=session, hostel_number=hostel_number):
        if session.query(
                Room
        ).join(
            Block
        ).join(
            Wing
        ).join(
            Hostel
        ).where(
            Hostel.number == hostel_number
        ).where(
            Room.name == room_name
        ).all():
            return True


def get_user_ip(session: Session, telegram_id: int) -> dict |
None:
    try:
        user_ip = session.query(
            func.concat("192.168.", Wing.subnet_address, ".",
```

```
                        func.coalesce(SpecialRoomIp.node_address,
func.cast(Room.name, Integer))),
            func.concat("192.168.", Wing.subnet_address, ".254")
        ).select_from(
            Person
        ).join(
            Room,
            full=True
        ).join(
            Block,
            full=True
        ).join(
            Wing,
            full=True
        ).join(
            SpecialRoomIp,
            full=True
        ).where(
            Person.telegram_id == telegram_id
        ).one()
        ip, subnet_address = user_ip
        return {
            "ip_address": ip,
            "mask_network": "255.255.255.0",
            "default_gateway": subnet_address,
            "preferred_dns_server": "192.168.105.13",
            "alternative_dns_server": "192.168.105.14"
        }
    except Exception:
        return None


def report_internet_problem(session: Session, telegram_id: int,
problem_id: str) -> None:
    if session.query(
            Person
    ).where(
        Person.telegram_id == telegram_id
    ).where(
        TaskManager.problems_id == problem_id
    ).one():
        raise ValueError("Problem already exists")
    person_id = session.query(Person).where(Person.telegram_id ==
telegram_id).one().id
    task_manager = TaskManager(person_id=person_id,
problems_id=problem_id)
    session.add(task_manager)
    session.commit()


def send_payment(session: Session, telegram_id: int, file_id: str,
file: bool = False, photo: bool = False) -> None:
```

```
    payment = Payment(id=uuid.uuid4(), file_id=file_id,
room_id=session.query(
        Room
    ).join(
        Person
    ).where(
        Person.telegram_id == telegram_id
    ).first().id, file=file, photo=photo)
    session.add(payment)
    session.commit()


def accept_payment(session: Session, room_name: str) -> None:
    accepted_payment =
session.query(Payment).join(Room).where(Room.name ==
room_name).all()
    for payment in accepted_payment:
        payment.accepted = True
    session.commit()


def fix_problem(session: Session, person_telegram_id: int,
problem_id: str) -> None:
    task_manager = session.query(TaskManager).join(Person). \
        where(Person.telegram_id == person_telegram_id). \
        where(TaskManager.problems_id == problem_id).first()
    task_manager.active = False
    session.commit()


def get_reports(session: Session) -> list[dict[str, str]]:
    reports_list = []

    report_info = session.query(
        Person, Room, Block, Wing, Hostel
    ).select_from(
        Person
    ).join(
        Room
    ).join(
        Block
    ).join(
        Wing
    ).join(
        Hostel
    ).join(
        TaskManager, TaskManager.person_id == Person.id
    ).all()

    if not report_info:
        return reports_list

    for reports in report_info:
```

```
        person, room, block, wing, hostel, internet_problem =
reports

        report_dict = {
            'telegram_id': person.telegram_id,
            'first_name': person.name,
            'phone_number': person.phone,
            'hostel_number': hostel.number,
            'wing_name': wing.name,
            'block_number': block.number,
            'room_number': room.name,
            'problem': internet_problem.description
        }

        reports_list.append(report_dict)

    return reports_list


def get_payment(session: Session) -> list[dict[str, str]]:
    payment_list = []

    payment_info = session.query(
        Person, Room, Block, Wing, Hostel, Payment
    ).select_from(
        Person
    ).join(
        Room
    ).join(
        Block
    ).join(
        Wing
    ).join(
        Hostel
    ).join(
        Payment, Payment.room_id == Room.id
    ).where(
        Payment.accepted == False
    ).all()

    if not payment_info:
        return payment_list

    for payment in payment_info:
        person, room, block, wing, hostel, payments = payment

        payment_dict = {
            'first_name': person.name,
            'last_name': person.surname,
            'phone_number': person.phone,
            'hostel_number': hostel.number,
            'wing_name': wing.name,
            'block_number': block.number,
```

```
            'room_number': room.name,
            'payment_send_date': payments.send_date,
            'payment_path': payments.path
        }

        payment_list.append(payment_dict)

    return payment_list


def get_payment_by_id(session: Session, payment_id: str) -> bool:
    payment = session.query(
        Payment
    ).where(
        Payment.file_id == payment_id
    ).first()
    if payment:
        return True
    else:
        return False


@validate_params(['room_name'],
                 [validate_room_name])
def change_special_ip(session: Session, room_name: str,
node_address: int) -> None:
    special_ip =
session.query(SpecialRoomIp).join(Room).where(Room.name ==
room_name).first()
    if not special_ip:
        special_ip =
SpecialRoomIp(room_id=session.query(Room).where(Room.name ==
room_name).first().id,
                                       node_address=node_address)
        session.add(special_ip)
        session.commit()
    special_ip.node_address = node_address
    session.commit()


def change_receive_mess(session: Session, telegram_id: int,
                        receive_payment: bool = None,
receive_internet_problems: bool = None) -> None:
    receive = session.query(Admin).where(Admin.telegram_id ==
telegram_id).first()
    if receive_payment:
        receive.receive_payment = receive_payment
    else:
        receive.receive_payment = receive_payment
    if receive_internet_problems:
        receive.receive_internet_problems =
receive_internet_problems
    else:
```

```
        receive.receive_internet_problems =
receive_internet_problems
    session.commit()


def get_admin_receives(session: Session, telegram_id: int) ->
dict:
    admin_info = session.query(
        Admin.receive_payment, Admin.receive_internet_problems
    ).where(
        Admin.telegram_id == telegram_id
    ).first()
    payment, internet_problem = admin_info
    return {
        "payment": payment,
        "internet_problem": internet_problem
    }
```

Додаток Ж

Програмний код для реєстрації користувачів у телеграм-боті "TNTU_Internet"

```
import re
from aiogram import F
from aiogram.fsm.context import FSMContext
from aiogram.types import Message, ReplyKeyboardMarkup,
KeyboardButton, ReplyKeyboardRemove
from bot.config import dp, bot, SessionLocal
from bot import Registration, main_menu
from config import messages
from logic.queries import user_registration, hostel_exist,
room_exist
from main import User


# F.from_user.id.in_({42, 1000, 123123}) for admin
@dp.message(Registration.ready_to_register, F.text ==
"Розпочнімо🔥")
async def write_name(message: Message, state: FSMContext):
    await state.set_data({"telegram_id": message.from_user.id})
    await bot.send_message(message.from_user.id, "Введіть своє
ім'я")
    await state.set_state(Registration.waiting_for_name)


@dp.message(F.text, Registration.ready_to_register)
async def incorrect_start(message: Message):
    await message.answer(
        "Для того щоб почати реєстрацію, нажми кнопку
Розпочнімо🔥",
        reply_markup=ReplyKeyboardMarkup(
            keyboard=[
                [
                    KeyboardButton(text="Розпочнімо🔥"),
                ]
            ],
            resize_keyboard=True, one_time_keyboard=True
        ),
    )


@dp.message(F.text, Registration.waiting_for_name)
async def write_surname(message: Message, state: FSMContext):
    name = message.text.strip()
    if not re.match(r"^[А-ЩЬЮЯҐЄІЇ][а-щьюяґєіїʼ]*$", name):
        if not re.match(r"^[a-zA-Z\s'-]+$", name):
            await message.answer("Ім'я має невірний формат.
Введіть ім'я згідно вимог.")
            return
```

```
    await state.update_data({"name": name})
    await bot.send_message(message.from_user.id, "Введіть своє
прізвище")
    await state.set_state(Registration.waiting_for_surname)


@dp.message(~F.text or F.sticker, Registration.waiting_for_name)
async def error_name(message: Message):
    await message.answer("Це не ім'я, напишіть своє ім'я")
    return


@dp.message(~F.text or F.sticker,
Registration.waiting_for_surname)
async def error_surname(message: Message):
    await message.answer("Це не прізвище, напишіть своє Прізвище")
    return


@dp.message(Registration.waiting_for_surname)
async def write_number_hostel_room(message: Message, state:
FSMContext):
    surname = message.text.strip()
    if not re.match(r"\b([A-ZÀ-ÿA-ЩЬЮЯҐЄІЇ][-,a-z. 'А-щьюяґєіїÀ-
ÿ]{0,49} *)+", surname):
        await message.answer("Прізвище має невірний формат.
Введіть прізвище згідно вимог.")
        return
    await bot.send_message(message.from_user.id, "Введіть номер
гуртожитка та кімнату\n"
                                                 "Приклад - 1-0162
(гуртожиток №1 кімната 0162)")
    await state.update_data({"surname": message.text})
    await
state.set_state(Registration.waiting_for_hostel_room_number)


@dp.message(Registration.waiting_for_hostel_room_number)
async def write_phone_number(message: Message, state: FSMContext):
    hostel_room_number = message.text.strip().replace("а", "a")
    if not re.match(r"\d-\d{1,4}a?$", hostel_room_number):
        await message.answer("Перевірте чи вірно Ви вписали номер
гуртожитка та кімнату")
        return
    with SessionLocal() as session:
        if not hostel_exist(session=session,
hostel_number=hostel_room_number[0]):
            await message.answer(f"Такого гуртожитку під номером
№{hostel_room_number[0]} не існує."
                                 f"Спробуйте ввести заново")
            return
```

```
        if not room_exist(session=session,
hostel_number=hostel_room_number[0],
room_name=hostel_room_number[2:]):
            await message.answer(f"Такої кімнати під номером
{hostel_room_number[2:]}"
                                 f" в гуртожитку
№{hostel_room_number[0]} немає")
            return
    await state.update_data({"hostel": hostel_room_number[0],
"room": hostel_room_number[2:]})
    keyboard = ReplyKeyboardMarkup(
        keyboard=[
            [
                KeyboardButton(text="Поділитись своїм номером
телефону", request_contact=True),
            ]
        ],
        resize_keyboard=True, one_time_keyboard=True
    )
    await state.set_state(Registration.waiting_for_phone)
    await message.answer("Відправ свій контакт натиснувши на
кнопку:", reply_markup=keyboard)


@dp.message(Registration.waiting_for_phone)
async def incorrect_contact(message: Message, state: FSMContext):
    if message.contact is None:
        await message.answer("Нажміть на кнопку, щоб передати
номер телефону.")
        return
    await state.update_data({"phone":
message.contact.phone_number})
    await write_all_vars(message=message, state=state)


@dp.message(Registration.result)
async def write_all_vars(message: Message, state: FSMContext):
    user_info = await state.get_data()

    user = User(telegram_id=user_info["telegram_id"],
                name=user_info["name"],
                surname=user_info["surname"],
                phone=user_info["phone"],
                hostel=user_info["hostel"],
                room=user_info["room"])
    with SessionLocal() as session:
        user_registration(session, telegram_id=user.telegram_id,
first_name=user.name, last_name=user.surname,
                          phone_number=user.phone,
hostel_number=user.hostel, room_name=user.room)
    await message.answer('Вітаю Ви успішно зареєструвались 😊',
reply_markup=ReplyKeyboardRemove())
```

Додаток К

Програмний код для реалізації функцій особистого кабінету адміністратора в телеграм-боті "TNTU_Internet"

```
import re

from aiogram import F
from aiogram.fsm.context import FSMContext
from aiogram.types import Message, ReplyKeyboardMarkup,
KeyboardButton, ReplyKeyboardRemove
from bot.config import dp, SessionLocal
from bot import InternetProblem, main_menu,
inline_keyboard_for_payment, admin_menu, Admin
from bot.keyboards import admin_cabinet, get_menu
from config import messages
from logic.queries import change_receive_mess, hostel_exist,
room_exist


@dp.message(F.text == "Особистий кабінет")
async def personal_account(message: Message, state: FSMContext):
    await state.set_state(Admin.personal_cabinet)
    await admin_cabinet(message)


@dp.message(F.text == "Отримання сповіщень квитанцій/проблем",
Admin.personal_cabinet)
async def get_mess_payment(message: Message, state: FSMContext):
    await state.set_state(Admin.receive_mess)
    await message.answer(
        "Ти хочеш отримувати сповіщення про надіслені квитанції?",
        reply_markup=ReplyKeyboardMarkup(
            keyboard=[
                [
                    KeyboardButton(text="✅"),
                    KeyboardButton(text="❌"),
                ],
            ],
            resize_keyboard=True, one_time_keyboard=True
        ),
    )


@dp.message(F.text.in_({"✅", "❌"}), Admin.receive_mess)
async def personal_account(message: Message, state: FSMContext):
    await state.set_data({"receive_payment": message.text})
    await state.set_state(Admin.end)
    await message.answer(
        "Ти хочеш отримувати сповіщення про наявні проблеми?",
```

```
        reply_markup=ReplyKeyboardMarkup(
            keyboard=[
                [
                    KeyboardButton(text="✅"),
                    KeyboardButton(text="❌"),
                ],
            ],
            resize_keyboard=True, one_time_keyboard=True
        ),
    )


@dp.message(Admin.end)
async def return_to_menus(message: Message, state: FSMContext):
    await state.update_data({"receive_report": message.text})
    data = await state.get_data()
    receive_payment, receive_report = [True if value == '✅' else
False if value == '❌' else None for value in
                                       data.values()]
    with SessionLocal() as session:
        change_receive_mess(session=session,
telegram_id=message.from_user.id,
                            receive_payment=receive_payment,
receive_internet_problems=receive_report)
    await message.answer("Вітаю твої побажання враховані",
reply_markup=ReplyKeyboardRemove())


@dp.message(Admin.personal_cabinet, F.text == "назад")
async def personal_account(message: Message):
    await get_menu(message)
@dp.message(F.text == "Отримати інформацію про кімнату")
async def get_num_room(message: Message, state: FSMContext):
    await state.set_state(Admin.info_room)
    await message.answer(
        "В цьому меню ти можеш дізнатись про кімнату(ip, блок,
крило, оплата, остання проблема(якщо були))"
    )
@dp.message(Admin.info_room)
async def get_number_room(message: Message, state: FSMContext):
    hostel_room_number = message.text.strip().replace("а", "a")
    if not re.match(r"\d-\d{1,4}a?$", hostel_room_number):
        await message.answer("Перевірте чи вірно Ви вписали номер
гуртожитка та кімнату")
        return
    with SessionLocal() as session:
        if not hostel_exist(session=session,
hostel_number=hostel_room_number[0]):
            await message.answer(f"Такого гуртожитку під номером
№{hostel_room_number[0]} не існує."
                                 f"Спробуйте ввести заново")
            return
```

```
        if not room_exist(session=session,
hostel_number=hostel_room_number[0],
room_name=hostel_room_number[2:]):
            await message.answer(f"Такої кімнати під номером
{hostel_room_number[2:]}"
                                 f" в гуртожитку
№{hostel_room_number[0]} немає")
            return

    await state.set_data({"hostel": hostel_room_number[0], "room":
hostel_room_number[2:]})
    await message.answer(str(await state.get_data()))
    await state.set_state(Admin.personal_cabinet)
    await message.answer(message.text)
    await message.answer(
        "вся його інфа",
        reply_markup=ReplyKeyboardMarkup(
            keyboard=[
                [
                    KeyboardButton(text="Редагувати ip"),
                ],
                [
                    KeyboardButton(text="меню"),
                ],
            ],
            resize_keyboard=True, one_time_keyboard=True
        ),
    )


@dp.message(F.text == "Редагувати ip")
async def edit_ip_configuration(message: Message, state:
FSMContext):
    await state.set_state(Admin.edit_ip)
    await message.answer("Введи нову ip для цієї кімнати",
reply_markup=ReplyKeyboardRemove())
@dp.message(Admin.edit_ip)
async def write_new_ip(message: Message, state: FSMContext):
    await state.set_data({"new_ip": message.text})
    await message.answer(str(await state.get_data()))
    await message.answer("Вітаю нову ip записано",
reply_markup=ReplyKeyboardRemove())
@dp.message(F.text == "меню")
async def get_back_menu(message: Message):
    await get_menu(message)
@dp.message(F.photo)
async def get_photo_id(message: Message):
    # data = await message.answer(message.photo[-1].file_id)
    print(message.photo[-1].file_id)
    await message.answer_photo(photo=message.photo[-1].file_id)
```