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: Серверний застосунок для обміну повідомленнями із віддаленими пристроями

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

_____ Поворознник Р.В.
(підпис) (прізвище та ініціали)

Керівник _____ Литвиненко Я.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 - 2026

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)
«__» _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

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

Студенту Поворознику Роману Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Серверний застосунок для обміну повідомленнями із віддаленими пристроями

Керівник роботи Литвиненко Ярослав Володимирович, д.т.н., проф.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «14» 05 2026 року № 4/9-238

2. Термін подання студентом завершеної роботи 10.06.2026 р.

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
1. Огляд предметної області.
2. Проектування та реалізація застосунку
3. Додаткові сервіси. Тестування розробки
4. Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титулка. 2.Актуальність, мета, задачі дослідження. 3. Взаємодія сервера із клієнтом. 4. Засоби і технології розробки. 5. Структура взаємодії сервера із пристроями та архітектура серверного застосунку. 6. Файлові структури застосунку. 7, 8. Скріншоти інтерфейсу користувача. 9. Метод обробки пакетів даних. 10. Хмарний сервіс. 11. Telegram сервіс. 12. Тестування. 13. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	11.05.26	*Виконано*
2.	Підбір джерел про застосунки для обміну		*Виконано*
	повідомленнями із віддаленими пристроями	12.05 – 14.05.26	*Виконано*
3.	Опрацювання літературних джерел	15.05 – 18.05.26	*Виконано*
4.	Виконання дослідження щодо розробки програмного		
	забезпечення	19.05 – 22.05.26	*Виконано*
6.	Оформлення розділу «Огляд предметної області»	24.05 – 25.05.26	*Виконано*
7.	Оформлення розділу «Проектування та реалізація		
	застосунку»	26.05 – 27.05.26	*Виконано*
8.	Оформлення розділу «Додаткові сервіси. Тестування		
	розробки»	28.05 – 30.05.26	*Виконано*
9.	Виконання завдання до підрозділу «Безпека		
	життєдіяльності, основи хорони праці»	31.05 – 03.06.26	*Виконано*
10.	Оформлення кваліфікаційної роботи	20.05 – 03.06.26	*Виконано*
11.	Нормоконтроль	01.06 – 05.06.26	*Виконано*
12.	Перевірка на плагіат	04.06 – 09.06.26	*Виконано*
13.	Попередній захист кваліфікаційної роботи	07.06 – 11.06.26	*Виконано*
14.	Захист кваліфікаційної роботи	18.06.26	

Студент _____ Поворозник Р.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Литвиненко Я.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Серверний застосунок для обміну повідомленнями із віддаленими пристроями // Поворозник Роман Віталійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра комп'ютерних наук, група СНз–41 // Тернопіль, 2026 // С. – 51, рис. – 36, табл. – 1, слайдів – 13, бібліогр. – 16.

Ключові слова: віддалений пристрій, канал, обмін повідомленнями, хмарний сервіс, шифрування даних, telegram-бот

Кваліфікаційна робота присвячена проектуванню та розробці застосунку, який призначений для обміну повідомленнями із віддаленими пристроями (з інтеграцією із хмарним середовищем та Telegram-ботом).

У першому розділі роботи виконано огляд предметної галузі запланованого дослідження. Наведено поняття та особливості застосування сокет- з'єднань з метою забезпечення надійної взаємодії серверного застосунку із віддаленими пристроями Описані технології та інструменти, котрі використовуватимуться для розробки.

У другому розділі докладно описано процес проектування та програмного втілення застосунку. Наведена та описана гнучка програмна архітектура розробки на базі концепції MVC, представлено користувацький інтерфейс. Реалізовано і проведено тестування протоколу передачі пакетів від сервера до віддаленого пристрою, котрий забезпечує безпечну і надійну передачу даних у процесі взаємодії в мережі при допомозі сокет- з'єднання.

У третьому розділі описано інтеграцію хмарного сервісу для забезпечення доступності та надання зручних інструментів для управління, а також впровадження Telegram-боту для своєчасного оповіщення користувачів. Проведено якісне тестування розробки.

ANNOTATION

Server Application for Message Exchange with Remote Devices // Povoroznyk Roman // Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science // Ternopil, 2026 // P. - 51, Fig. - 36, Table -1, Slide - 13, References - 16.

Keywords: remote device, channel, messaging, cloud service, data encryption, telegram bot

Thesis deals with the design and development of an application that is intended for exchanging messages with remote devices (with integration with the cloud environment and Telegram bot).

The first section of the work reviews the subject area of the planned research. The need for conducting such research is described. The concepts and features of using socket connections are presented to ensure reliable interaction of the server application with remote devices. The technologies and tools that will be used for development are described.

The second section describes in detail the process of designing and software implementation of the application. A flexible software development architecture based on the MVC concept is presented and described, and the user interface is presented. The packet transfer protocol from the server to the remote device is implemented and tested, which ensures safe and reliable data transfer in the process of interaction in the network using a socket connection.

The third section describes the integration of a cloud service to ensure accessibility and provide convenient management tools, as well as the implementation of a Telegram bot for timely notification of users. Qualitative testing of the development was carried out.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) – інтерфейс прикладного програмування, набір правил та інструментів, що дозволяє різним програмним застосункам «спілкуватися» між собою, обмінюючись даними та функціональністю.

REST (англ. Representational State Transfer) – архітектурний стиль взаємодії компонентів розподілених програм у мережі, що використовує протокол HTTP для обміну даними.

REST API – архітектурний стиль, який використовує HTTP протокол для обміну даними, що дозволяє легко інтегрувати різні сервіси та програми, забезпечуючи гнучкість та широкі можливості при розробці/

ПЗ – програмне забезпечення.

ЗМІСТ

ВСТУП

Актуальність теми. Дистанційне управління процесами та механізмами – це основа розвитку економіки в епоху цифровізації та глобалізації сучасного світу. Організація ефективного віддаленого управління та централізованої обробки даних є стандартною проблемою в галузі створення систем автоматизації.

Сучасні технології цифрового зв'язку дозволяють розробляти виконавчі електронні пристрої для керування процесами на відстані за умови безперервного доступу до Інтернету.

На сьогоднішній день месенджери набирають все більшу популярність, оскільки є одним із найбільш зручних та практичних способів обміну інформацією. Програмні засоби для обміну повідомленнями повинні використовувати, перш за все, спеціалізовані хмарні інструменти для синхронізації даних. Варто зауважити, що вимагається наявність шифрування даних, котрі передаються по каналах зв’язку, з метою захисту персональних даних.

Мета роботи – розробка серверної програми з графічним інтерфейсом для обміну повідомленнями з віддаленими пристроями, хмарним сервісом та ботом для месенджера Telegram.

Для досягнення мети виділено ряд завдань:

– вивчення принципу пристрою серверної програми для обміну повідомленнями;

– проектування архітектури застосунку;

– розробка графічного інтерфейсу;

– реалізація установки з'єднання програми з віддаленими пристроями;

– інтеграція хмарного сервісу для синхронізації даних віддалених пристроїв;

– розробка Telegram бота для впровадження користувацьких оповіщень.

РОЗДІЛ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Необхідність проведення досліджень

У сучасному цифровому світі, де об’єми даних постійно зростають експоненційно, надзвичайно важливим є створення серверних застосунків для обміну даними з віддаленими пристроями незаперечна. Такі програми дозволяють централізовано управляти потоками інформації, забезпечуючи їх збирання, обробку та розподіл у реальному часі. А застосування останніх тенденцій у розробці додатково підвищує їх значущість та ефективність.

Інтеграція з хмарним сервісом дозволить розширити можливості зберігання та аналітики даних, шляхом забезпечення доступу до властиво потужних обчислювальних ресурсів, а також глобальної доступності. А використання Telegram бота для надсилання повідомлень та звітів у Telegram канал надасть користувачам можливість отримувати актуальну інформацію швидко та у зручній формі, що особливо важливо для оперативного реагування на події та прийняття рішень. Поєднання даних технологій перетворює застосунок на потужний інструмент для автоматизації та моніторингу.

У зв'язку з цим реалізація такого ПЗ є актуальним завданням, націленим на покращення процесів збору, аналізу та розповсюдження даних, що в кінцевому підсумку сприяє підвищенню ефективності та оптимізації бізнес-операцій у різних галузях.

1.2 Сокет з'єднання

Для забезпечення надійної взаємодії серверної програми з віддаленими пристроями через протокол IP використання сокетів є оптимальним рішенням.

Сокет – це програмний інтерфейс, що є кінцевою точкою для відправлення та отримання пакетів даних у рамках мережного з'єднання (рисунок 1.1). Сокети можуть бути реалізовані для роботи в транспортних протоколах, таких як TCP (Transmission Control Protocol) для встановлення надійного з'єднання з гарантією

доставки повідомлень або UDP (User Datagram Protocol) для більш швидкої, але без гарантійної передачі [1]. В рамках реалізації серверної програми буде використовуватись TCP режим.

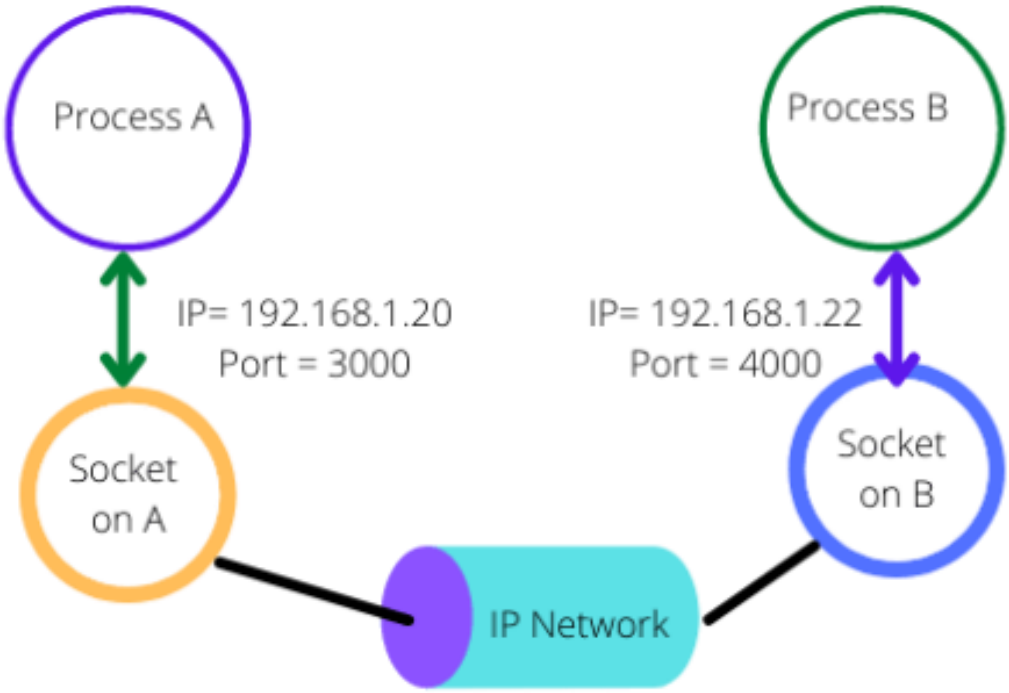

Рисунок 1.1 – Принцип роботи сокетів

Процес створення сокету для сервера відрізняється від клієнта тим, що серверний сокет повинен бути пов'язаний із заздалегідь заданим портом для прослуховування запитів і готовий до прийняття вхідних з'єднань, тоді як клієнтський сокет ініціює з'єднання з сервером, використовуючи вже відому адресу та порт сервера. Порт клієнта у разі вибирається операційною системою з діапазону доступних портів автоматично, дозволяючи спростити процес встановлення з'єднання для клієнтських програм. Процес обміну даними між сервером та клієнтом із використанням сокетів представлений на рисунку 1.2.

Рисунок 1.2 – Взаємодія сервера із клієнтом

Така структура обумовлюється тим, що сервер може обслуговувати безліч вхідних з'єднань, тоді як клієнт ініціює лише одиничне з'єднання до конкретного сокет сервера.

Сокети відіграють ключову роль у мережевому програмуванні, забезпечуючи стандартизований механізм для надійного обміну даними між віддаленими пристроями, що є основою сучасних розподілених систем та мережевих комунікацій.

1.3 Засоби розробки

При розробці багатофункціонального ПЗ важливим аспектом є вибір такого набору технологій, при яких буде досягнуто оптимальна продуктивність і ефективність кінцевого застосунку.

Для забезпечення кросплатформної сумісності був вибраний Python як основна мова програмування. Завдяки своєму потужному вбудованому функціоналу та широкій підтримці різних операційних систем, Python виявився ідеальним вибором, який сприяє прискоренню розробки та спрощенню підтримки ПЗ., забезпечення на різноманітних платформах [2].

У процесі розробки ПЗ було застосовано такі ключові технології:

– середовище розробки Visual Studio Code;

– мова програмування Python;

– система керування версіями Git для ефективної зміни коду та підтримки множинних паралельних гілок розробки;

– стандартна бібліотека Tkinter для реалізації графічного інтерфейсу серверної програми;

– бібліотека ttkboostrap з метою розширення можливостей Tkinter, надаючи динамічне використання сучасних тем та стилів;

– бібліотека matplotlib для візуалізації даних у вигляді графіків у рамках графічного інтерфейсу;

– стандартна бібліотека socket для реалізації низькорівневого мережевої взаємодії;

– стандартна бібліотека threading для організації одночасної роботи графічного інтерфейсу та фонових процесів, таких як мережеві запити або обробка даних, що дозволяє підвищити продуктивність програми за рахунок паралельного виконання завдань;

– бібліотека python-telegram-bot використовується для створення робота в месенджері Telegram, надаючи зручний інтерфейс для взаємодії з Telegram Bot API;

– бібліотека requests для взаємодії сервера з віддаленим хмарним

сервісом за допомогою REST API, що забезпечує ефективну та безпечну передачу даних.

Завдяки перерахованим вище технологіям вдалося створити потужне та гнучке ПЗ, яке не тільки відповідає поточним вимогам та тенденціям, але й легко масштабується для майбутніх завдань та інтеграцій.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

2.1 Архітектура застосунку

Перед початком розробки ПЗ потрібно встановити ключові взаємозв'язки серверної програми між віддаленими пристроями та допоміжними сервісами.

Необхідно організувати структуру таким чином, щоб віддалені пристрої могли надійно підключатися до сервера та обмінюватись даними.

Це потребує налаштування ефективних комунікаційних каналів, які можуть підтримувати постійне з'єднання зі швидким реагуванням на запити. Додаткові сервіси, такі як хмарне сховище та Telegram бот для автоматичного відправлення повідомлень у Telegram канал, повинні інтегруватись із серверним застосунком так, щоб забезпечувати синхронізовану роботу всіх компонентів системи.

У результаті це дозволить створити потужну і гнучку платформу, здатну адаптуватися до вимог, що змінюються, і масштабуватися відповідно до зростання навантаження всієї системи в цілому. На рисунку 12.1 представлено загальну структуру ПЗ.

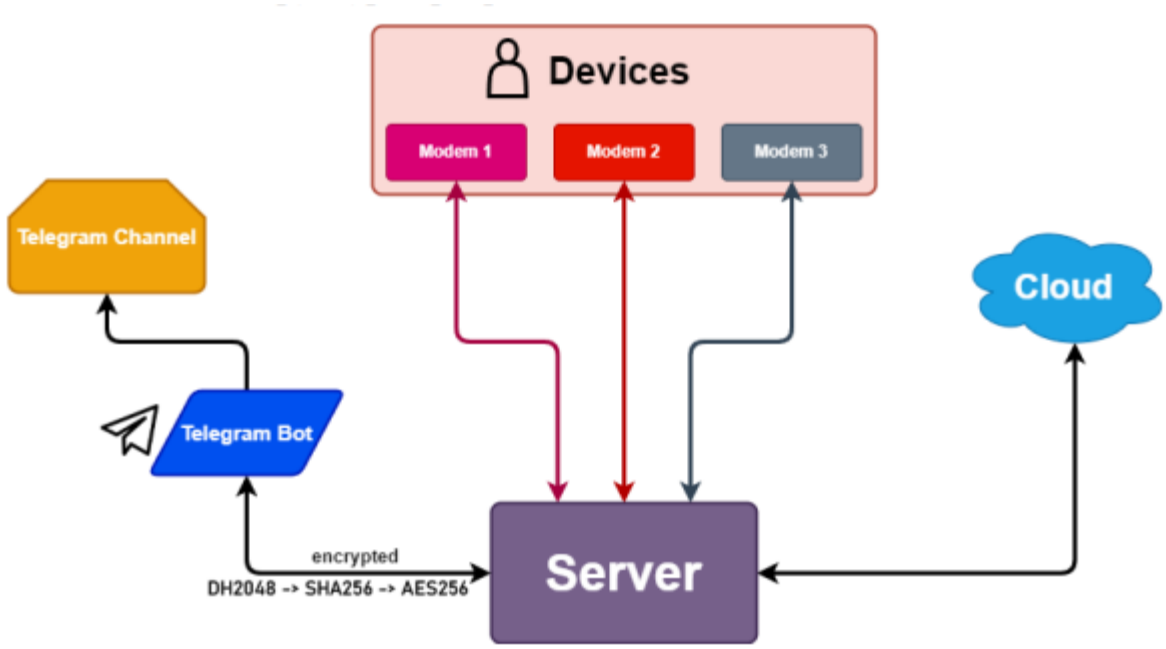

Рисунок 2.1 – Структура взаємодії сервера із пристроями

При проектуванні та організації архітектури серверного застосунку застосовувалася концепція моделі MVC (Model-View-Controller) [3] для поділу на окремі компоненти логіки, що управляє, і даних. Подібна схема дозволяє створювати гнучкі та масштабовані застосунки з високою продуктивністю. Приклад схеми представлений на рисунку 2.2.

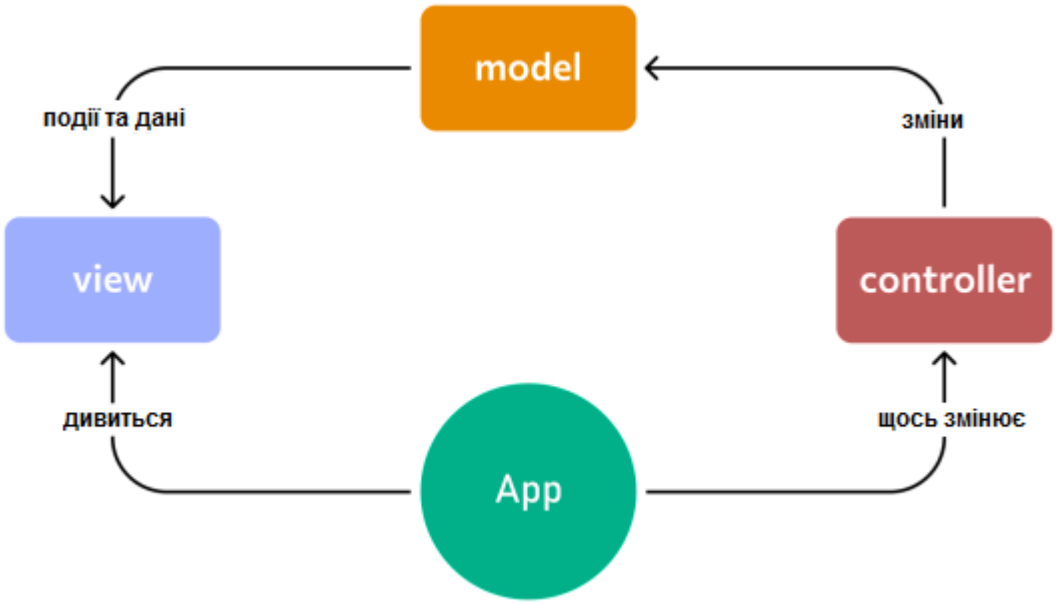

Рисунок 2.2 – Архітектура MVC

Компонент представлення (View) відповідає відповідно за подання даних користувачеві. Він включає все, що пов'язане з візуальним відображенням і тому не обробляє дані безпосередньо, а лише відображає їх відповідно до вказівок, котрі отримані від компонента контролера.

Компонент модель (Model) є бізнес-логікою застосунку та дані. Він відповідає за отримання, обробку та зберігання інформації, яка потім може бути представлена через компонент подання або змінена за командами контролера.

Компонент контролер (Controller) є сполучною ланкою між компонентами моделі та представлення. Він обробляє команди, що входять від користувача, передає їх моделі для виконання необхідних операцій і оновлює представлення, щоб відобразити будь-які зміни в даних.

Розглянемо підсумкову архітектуру серверної програми, представлену на рисунку 2.3.

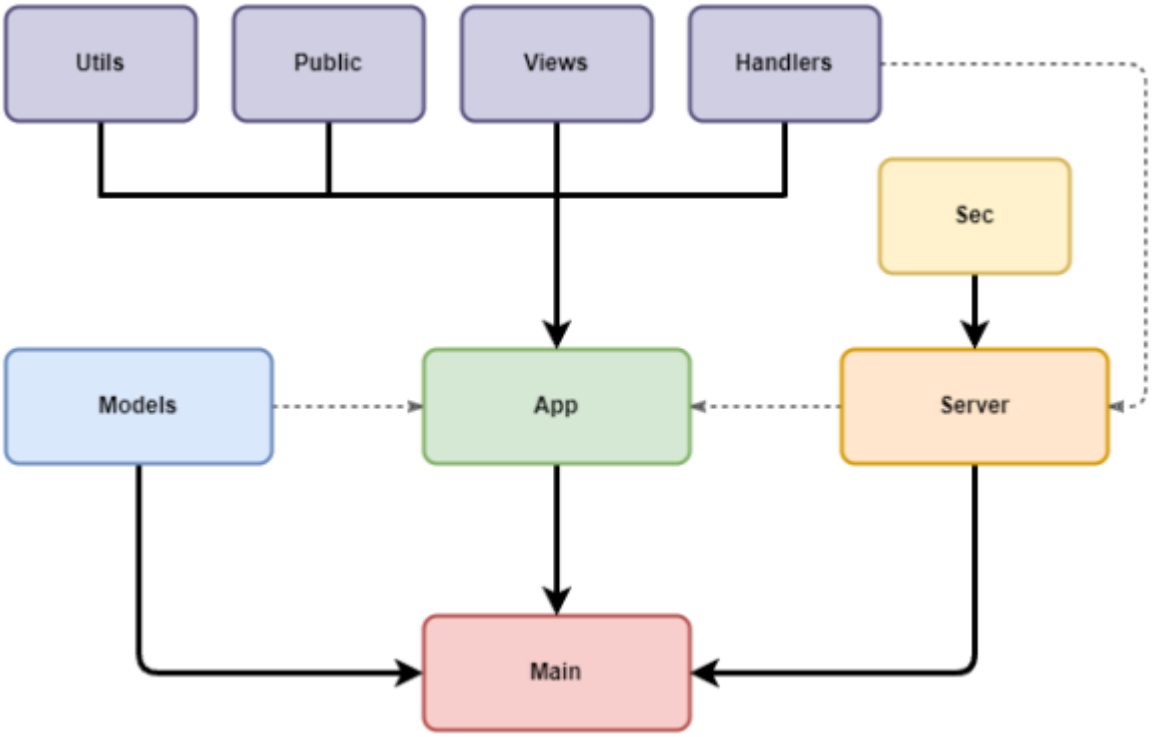

Рисунок 2.3 – Архітектура серверного застосунку

За запуск програми відповідальний безпосередньо модуль Main, всередині якого відбувається складання модуля App, для успішної роботи якого потрібно також заздалегідь проініціалізувати такі модулі як Server та Models.

Модуль Utils (рисунок 2.4) є сукупністю додаткових інструментів, що застосовуються в різних сегментах програми, які є невід'ємними для його коректного функціонування. Даний модуль включає функціонал для конвертації інформації в байтовий код і назад, що необхідно для її передачі через мережеві канали, а також парсер для аналізу даних, що надходять з метою вилучення корисного навантаження.

Модуль Public включає файли стилів та інших ресурсів, необхідні для коректного відображення елементів інтерфейсу. Наприклад, файли іконок програми, які використовуються для покращення візуального сприйняття інтерфейсу користувача та забезпечення інтуїтивно зрозумілої навігації за програмою.

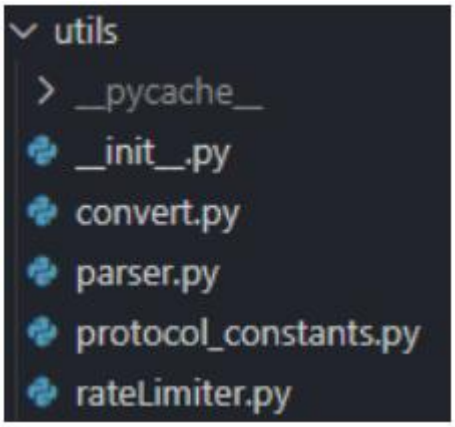

Рисунок 2.4 – Файлова структура Utils

Модуль Handlers (рисунок 2.5) містить у собі функціонал по перетворенню даних у формат, узгодженого з віддаленими пристроями протоколу для подальшого відправлення по мережі. Також модуль застосовується для обробки з'єднання із хмарним сервісом. Крім того, він відповідальний за функцію журналювання системної інформації та відображення даних у графічному інтерфейсі програми для подальшого аналізу.

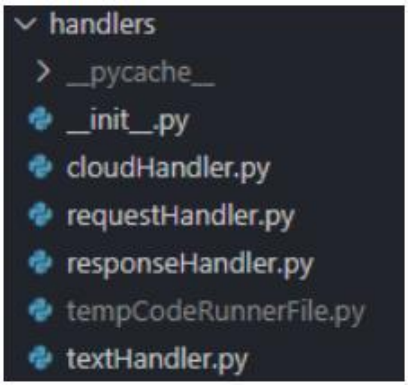

Рисунок 2.5 – Файлова структура Handlers

Модуль Views (малюнок 2.6) забезпечує відтворення всього графічного інтерфейсу серверної програми. Для зручності кожен вид був поділений на окремі сегменти, такі як вкладки керування, конфігурації пристроїв, статистика та інші. Це полегшує процес розробки та тестування окремих елементів інтерфейсу.

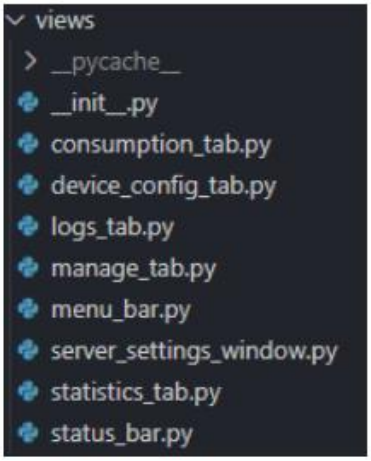

Рисунок 2.6 – Файлова структура Views

Компонент Server відповідає за з'єднання та передачу повідомлень як з віддаленими пристроями, так і хмарним сервісом та Telegram ботом [4]. Модуль Sec, використовуючи симетричні та асиметричні методи, забезпечує комплексне шифрування даних між ботом та сервером. Також Server використовує частину функціоналу компонента Handlers для обробки запитів віддалених пристроїв.

Файлова структура підсумкового проекту представлена на рисунку 2.7

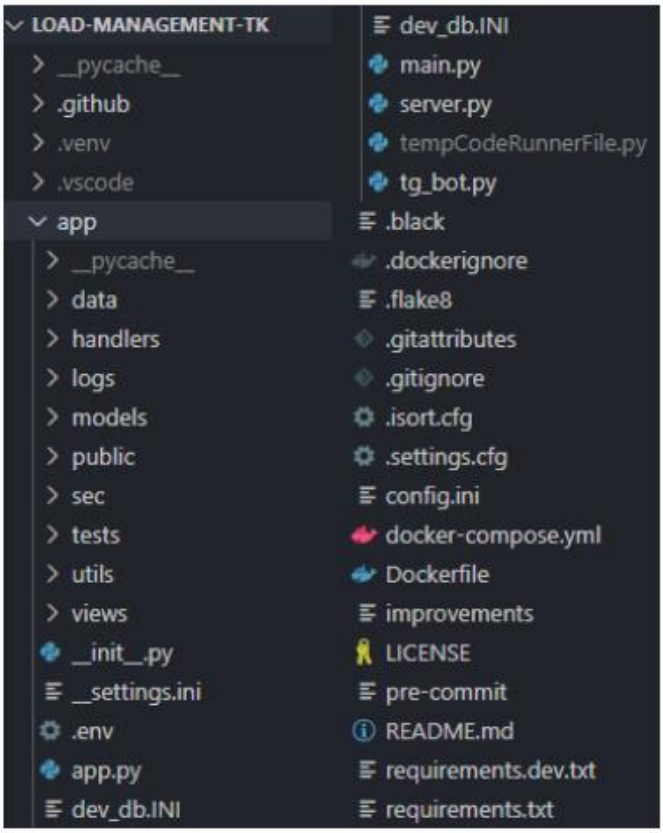

Рисунок 2.7 – Файлова структура проекту

Таким чином вдалося розробити гнучку архітектуру серверного застосунку, здатну до подальшої масштабованості відповідно до зростаючих вимог, забезпечуючи при цьому стабільність і високу продуктивність системи в довгостроковій перспективі.

2.2 Інтерфейс користувача

Розроблений користувальницький інтерфейс має широкий набір функціональних можливостей, спрямованих на зручну та ефективну взаємодію користувача із застосунком, включаючи інтуїтивно зрозумілу навігацію, персоналізовані налаштування та адаптивний дизайн, який гарантує комфортну роботу на різних пристроях та платформах.

При старті програми відкривається вікно з множиною вкладок, усередині яких міститься різний функціонал. Кожна вкладка буде детально розглянута далі. Приклад коду інтеграції вкладки у головне вікно програми наведено на рисунку 2.8.

```
self.tab_control = ttk.Notebook(self)
consumption_tab = ConsumptionTab(
self.tab_control, self.configs.device, self.server)
self.tab_control.add(consumption_tab, text="Consumption")
self.tab_control.pack(expand=1, fill="both")
```

Рисунок 2.8 – Лістинг коду інтеграції вкладки

Також з будь-якої вкладки завжди доступне контекстне меню, розташоване у верхній частині програми. Також користувачі можуть використовувати команди цього меню, використовуючи комбінації гарячих клавіш. Наприклад, для запуску сервера можна натиснути Ctrl+R на пристроях з OCWindows та Linux чи Cmd+R на пристроях Mac (рисунок 2.9).

```
        file_menu.add_checkbutton(
            label="Enable Server",
            command=self.master.toggle_server,
            accelerator="Cmd+R" if platform.system() == "Darwin"
else "Ctrl+R",
        )
```

Рисунок 2.9 – Лістинг коду запуску сервера

Нижня частина інтерфейсу відображає актуальний статус сервера і показує поточну дату та час, синхронізовані з системним годинником комп'ютера, на якому було здійснено запуск програми. Це забезпечує користувачам постійний доступ до важливої інформації про стан сервера та дозволяє відстежувати час без необхідності залишати програму.

Вікно з відкритою вкладкою споживання під час запуску програми представлено на рисунку 2.10.

Вкладка перегляду споживання дозволяє здійснювати моніторинг параметрів у реальному режимі часу для кожного пристрою. Дані виводяться як і числових значеннях, і у вигляді графіка. Графік може бути додатково налаштований для масштабування, щоб користувачі могли збільшувати або зменшувати відображені дані для більш детального вивчення. Також передбачено функцію збереження графіка як окремого файлу, що дозволяє експортувати дані для подальшого аналізу чи подання у звітах. Формати файлів для збереження можуть бути різними, забезпечуючи гнучкість та сумісність з різними програмами для перегляду та редагування зображень [5].

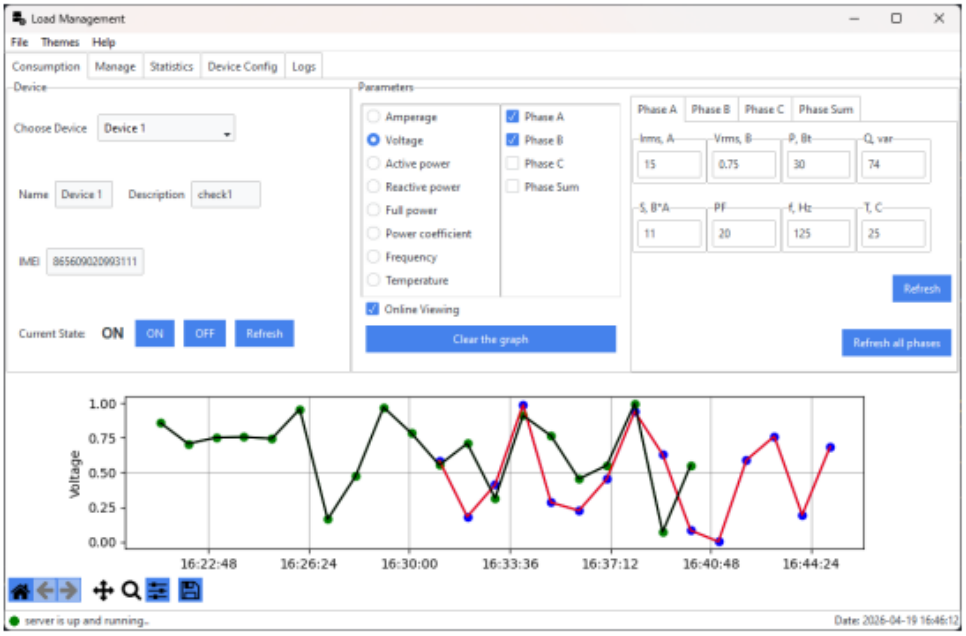

Рисунок 2.10 – Вкладка споживання

У вкладці керування пристроями (рисунок 2.11) користувачі можуть переглядати всі зареєстровані пристрої. Для кожного пристрою передбачені кнопки для увімкнення та вимкнення, а також для оновлення поточного стану. Якщо пристрій активний, відображається зелена індикація, а також активується таймер, що показує час з моменту запуску пристрою. У випадку, коли пристрій неактивний, з'являється червона індикація, що сигналізує про відключення.

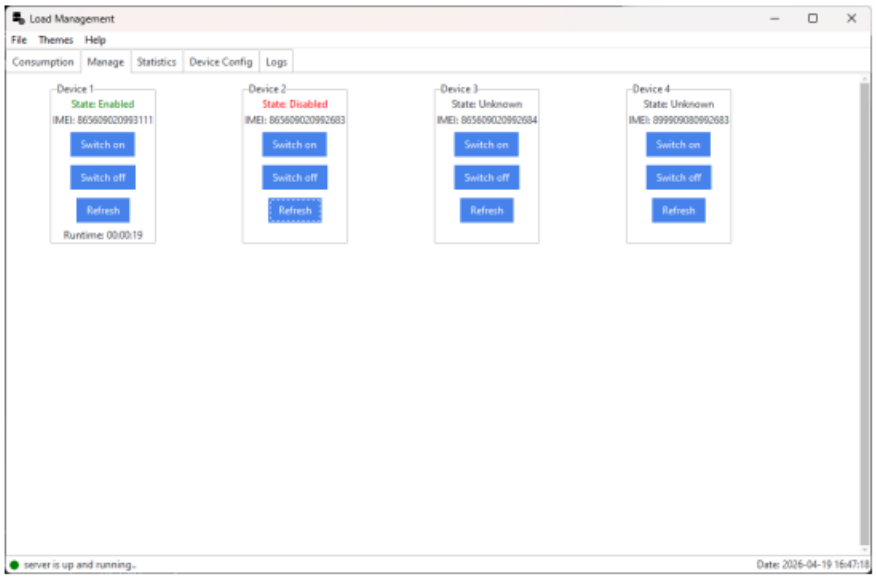

Рисунок 2.11 – Вкладка керування пристроями

Вкладка статистики є інструментом для відображення позамережевої інформації про обраний пристрій, візуалізуючи дані у вигляді графіків. Користувач може вибрати період перегляду даних, які були попередньо завантажені та збережені на локальний диск комп'ютера, де працює програма. Це дозволяє проводити аналіз пристроїв без необхідності підключення до мережі, забезпечуючи доступ до історичних даних для глибокого аналізу. Вкладка представлена на рисунку 2.12.

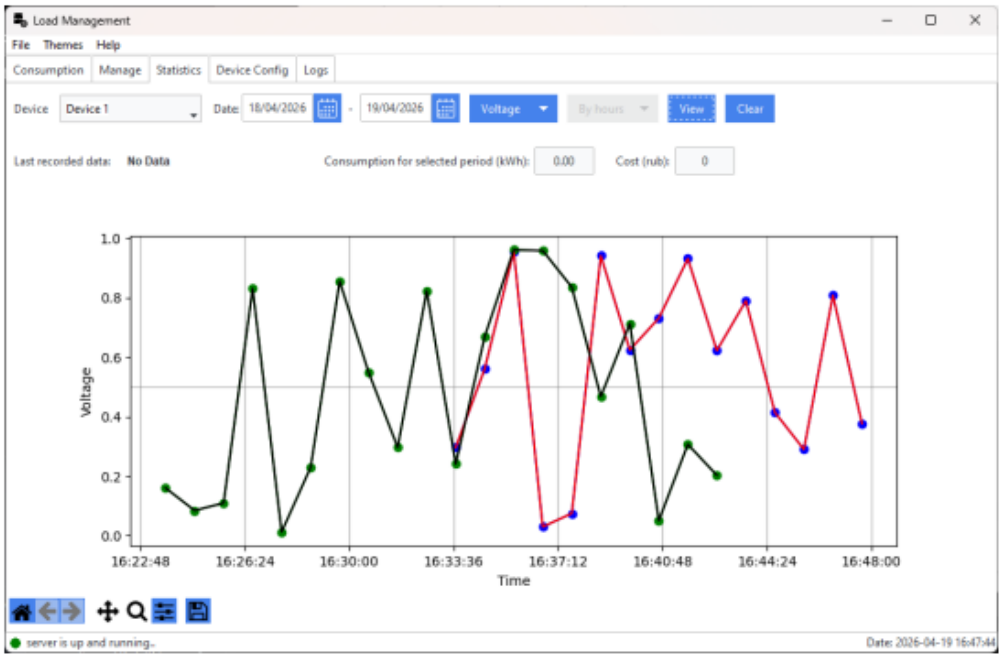

Рисунок 2.12 – Вкладка статистики

У вкладці конфігурацій пристроїв (рисунок 2.13) є багатофункціональна панель інструментів. Наприклад, серед доступних опцій є кнопка виявлення затримки з'єднання між пристроєм і сервером, що дозволяє оцінити якість і швидкість мережевої взаємодії.

Для визначення затримки обчислюється час від передачі мережного пакета сервером до відповіді від пристрою назад (рисунок 2.14)

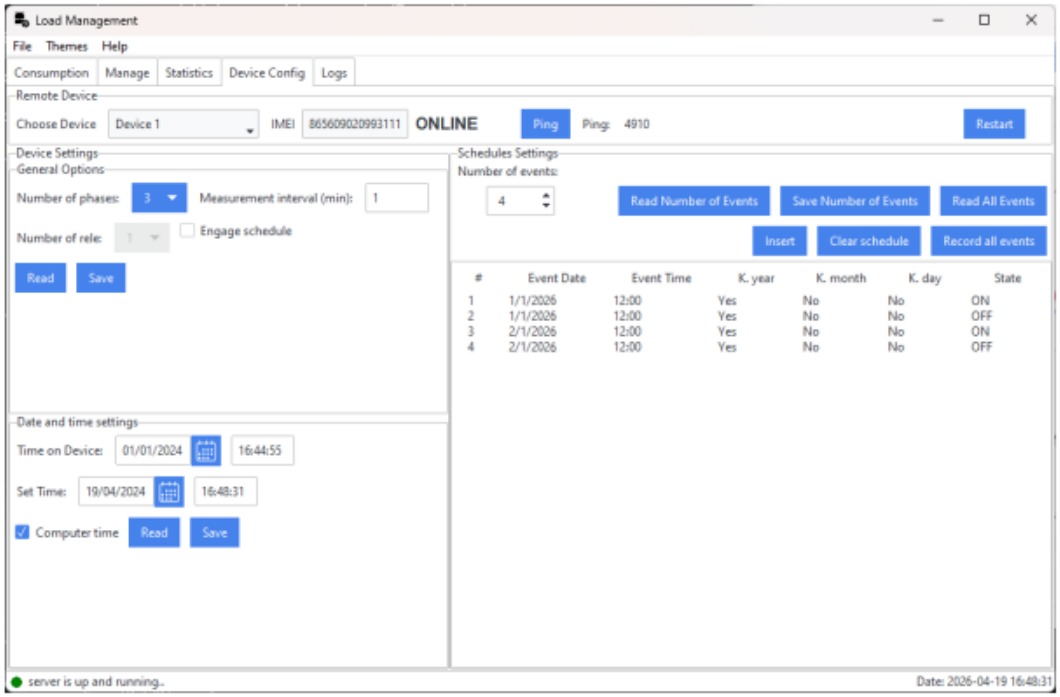

Рисунок 2.13 – Вкладка конфігурації пристроїв

```
def ping_callback(self, *data):
    end_time = time.time()
    val_time = end_time - self.start_time
    val_time = self.get_digits_ping(val_time)
    self.label_ping_value["text"] = val_time
```

Рисунок 2.14 – Лістинг обчислення часу передачі

Також передбачено кнопку для перезавантаження пристрою. Сервер створює спеціальний пакет даних і відправляє його через сокети безпосередньо обраному через графічний інтерфейс пристрою (рисунок 2.15)

```
    def restart(self):
        imei =
int(self.config_manager.get_value_by_name(self.selected_device,
"IMEI"))
        package = ResponseHandler.build_restart(imei)
        client = self.server.clients.get_by_imei(imei)
        if client:
            client.socket.send(package)
```

Рисунок 2.15 – Лістинг коду перезавантаження пристрою

Вкладка журналювання (рисунок 2.16) призначена для своєчасного

реагування на раптові інциденти та всебічного аналізу всіх вхідних та вихідних пакетів даних. Вона забезпечує безперервний моніторинг системних подій, що дає змогу оперативно детектувати і усувати ймовірні проблеми, а також аналізувати поведінку системи для оптимізації її роботи.

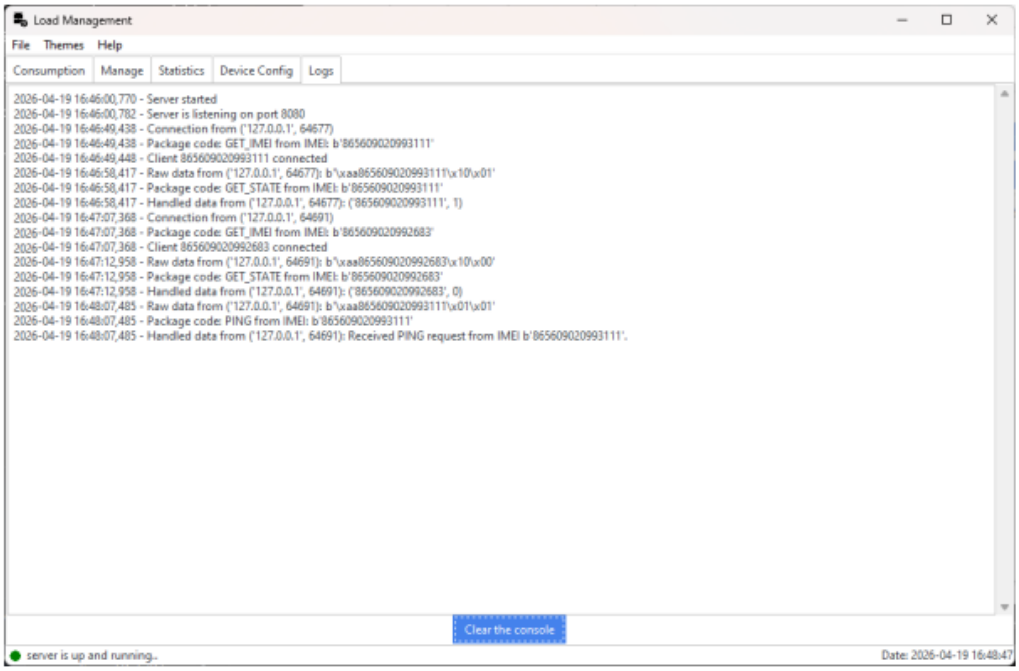

Рисунок 2.16 – Вкладка із системною інформацією

Для доступу до налаштувань сервера необхідно використовувати контекстне меню. При цьому параметри будуть відображатися в новому модальному вікні, яке не закриває основне вікно програми, але робить його неактивним доти, доки вікно налаштувань не буде закрито. Для редагування доступні поля, такі як порт сервера, автоматичне включення сервера на прослуховування підключень при запуску програми, а також інші параметри. Вікно з налаштуваннями представлене на рисунку 2.17.

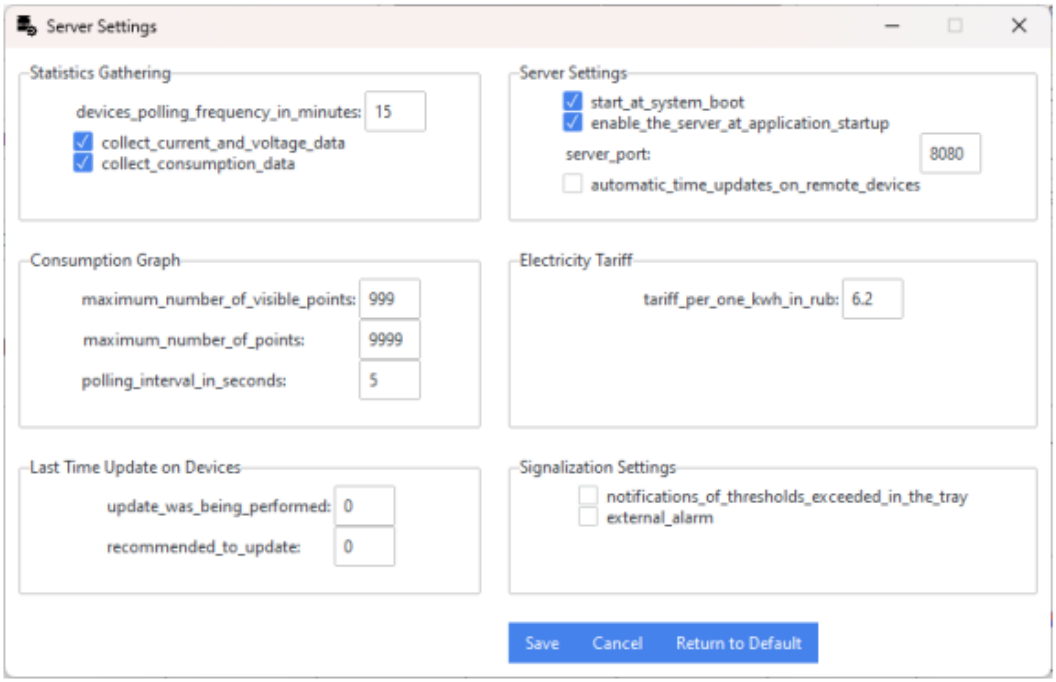

Рисунок 2.17 – Вікно конфігурації сервера

Була додатково реалізована функція зміни візуального оформлення програми (рисунок 2.18), яка активується через опцію перемикання у контекстному меню, розташованому у верхній частині інтерфейсу. Це дозволяє користувачам зробити вибір між світлою чи темною темами згідно із їх уподобаннями або умовами освітлення, забезпечуючи комфортну роботу із застосунком у будь-який час доби.

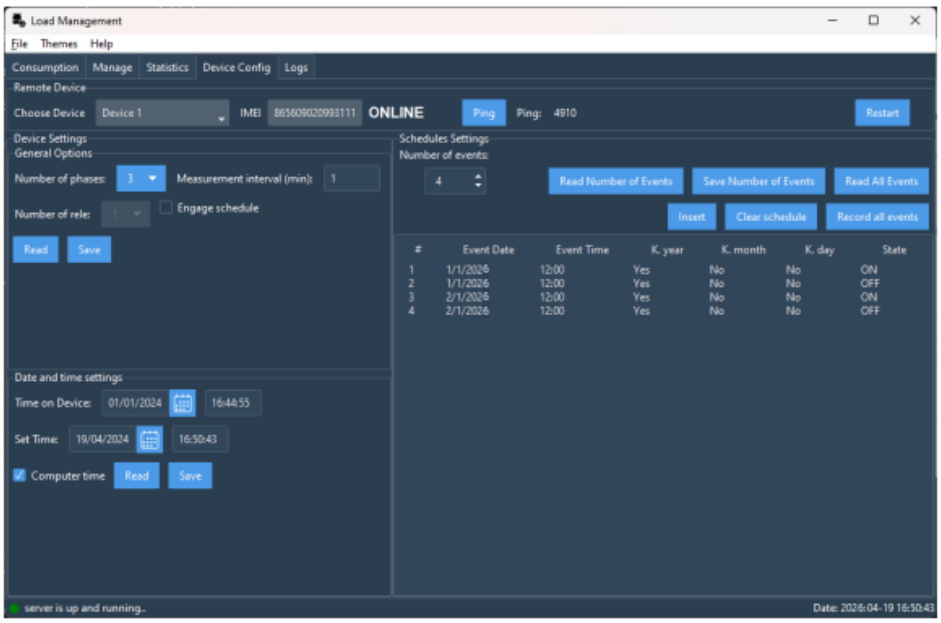

Рисунок 2.18 – Вікно з іншим стилем інтерфейсу

Код методу зміни теми програми наведено на рис. 2.19.

```
def toggle_theme(self):
    if self.checkbox_bar is False:
        self.checkbox_bar = True
        self.master.style.theme_use("superhero")
    else:
        self.checkbox_bar = False
        self.master.style.theme_use("litera")
```

Рисунок 2.19 – Лістинг коду методу зміни теми

2.3 Протокол передачі даних

2.3.1 Опис протоколу

Для скоординованої роботи сервера з віддаленими пристроями необхідно було організувати єдиний метод обміну повідомленнями через Інтернет. У зв'язку з цим, у серверній частині програми був реалізований протокол обміну даними, завдання якого було готувати дані у узгодженому форматі перед відправленням, а також коректно зчитувати інформацію зі сторони віддалених пристроїв. За рахунок цього вдалося встановити однозначний зв'язок між сервером та пристроями. Структура кінцевого пакета представлена у таблиці 2.1.

Таблиця 2.1 – Структура пакета даних

	START_BYTE	IMEI	OPCODE	PAYLOAD
Довжина в байтах	1	15	1	0 - 25
Дані	0xAA	Ідентифікатор пристрою	Номер коду	Корисне навантаження

Встановлення стартового байту як константного значення потрібне для однозначного визначення пакету даних серед інших потоків інформації. Це дозволяє серверу коректно інтерпретувати початок і кінець даних, котрі

передаються, що особливо важливо в умовах передачі даних в режимі реального часу. Саме тому константне значення стартового байту забезпечує надійне розмежування пакетів і допомагає уникнути помилок при декодуванні.

Встановлення унікального значення ідентифікатора віддаленого пристрою дає змогу визначити для когось або від кого надійшов той чи інший пакет даних. Ідентифікатор також застосовується для автентифікації пристрою та перевірки його справжності.

Задання номера коду необхідна для встановлення певних функцій та методів програми для подальшої обробки пакета. Кожен номер є унікальним для відповідності з операційним кодом (далі – опкод), який має на увазі певні дії для виконання застосунком. Усі можливі коди операцій наведено на рисунку 2.20.

1) PING,
2) GET_TIME,
3) SET_TIME,
4) GET_SH_COUNT,
5) SET_SH_COUNT,
6) GET_SH_ITEM,
7) SET_SH_ITEM,
8) DELETE_ALL_SH_ITEMS,
9) SH_ENA,
10) IS_SH_ENA,
11) SEND_SMS,
12) GET_IMEI,
13) GET_POLLTIME,
14) SET_POLLTIME,
15) SET_PHASENUM,
16) GET_STATE,
17) SET_STATE,
18) GET_MES_ITEM,
19) GET_ALLDATA,
20) GET_IRMS,
21) GET_VRMS,
22) GET_ACTIVE_POWER,
23) GET_REACTIVE_POWER,
24) GET_APPARENT_POWER,
25) GET_POWER_FACTOR,
26) GET_FREQUENCY,
27) GET_TEMPERATURE,
28) GET_ENERGY,
29) RESTART,
30) GET_RSSI,
31) SEND_RSSI,
32) UPDATE_TIME.

Рисунок 2.20 – Можливі коди операцій

До пакета як корисне навантаження прикріплюються дані, котрі безпосередньо відповідають вказаному в пакеті номеру коду команди. У випадках, коли корисне навантаження не потрібне для відправлення, ця частина не прикріплюється до кінцевого пакету, наприклад, при запиті даних від віддалених пристроїв. Це дозволяє оптимізувати трафік та прискорити обробку запитів.

2.3.2 Реалізація протоколу

Для реалізації цього протоколу було розроблено два обробники даних, кожен з яких відповідальний за певне завдання:

– клас RequestHandler;

– клас ResponseHandler.

Методи обробки показані на рисунку 2.21.

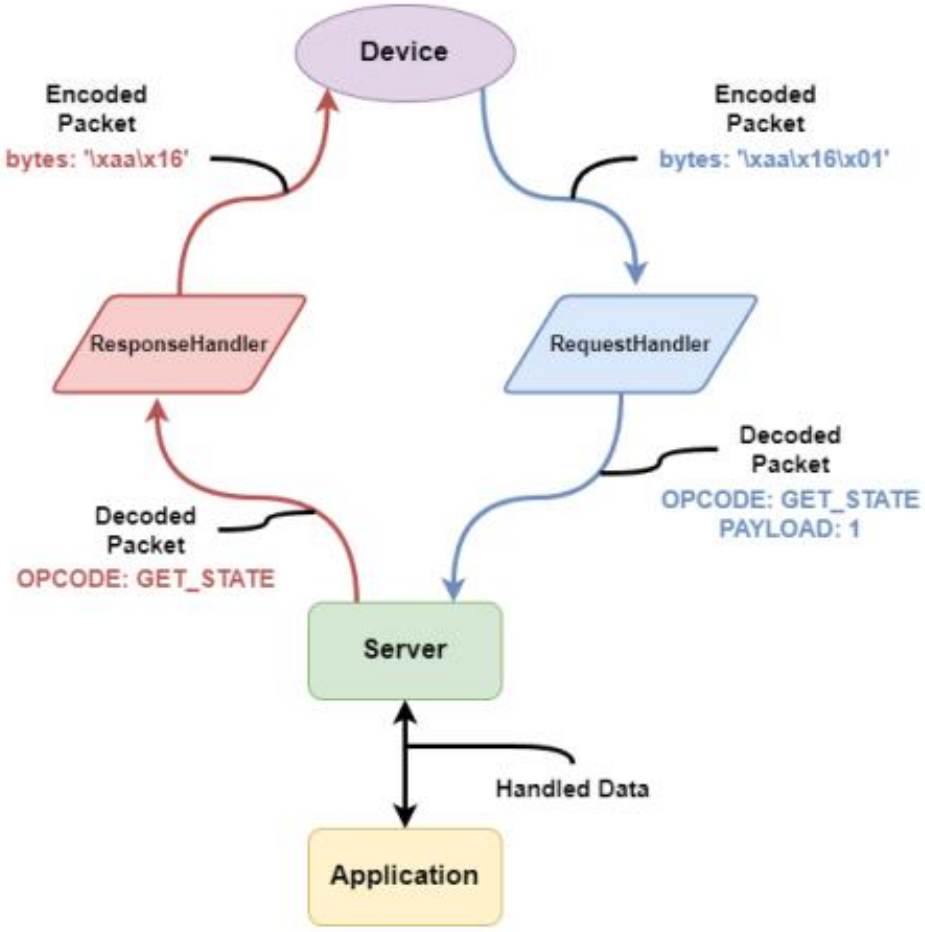

Рисунок 2.21 – Метод обробки пакетів даних

Розберемо ці класи окремо.

Завдання обробника RequestHandler полягає в декодуванні пакета, що надходить з боку віддалених пристроїв для подальшої роботи серверної програми.

Код методів RequestHandler з первинної обробки пакета даних представлений на рисунку 2.22.

```
    def handle_request(self, data: bytes):
        logging.basicConfig(level=logging.INFO)
        try:
            start_byte, imei, package_code_number, payload =
Parser.parse_request(data)
            package_code = Parser.get_code(package_code_number)
            logging.info(f"Package code: {package_code.name}
from IMEI: {imei}")
            handler = self.routes.get(package_code,
self.handle_unknown)
            return handler(imei, payload)
        except Exception as e:
            logging.error(f"Error handling the request: {e}")
            return None

    def handle_unknown(self, imei, payload):
        return f"Unknown package code from IMEI {imei}."
```

Рисунок 2.22 – Лістинг методів RequestHandler

При отриманні даних за допомогою сокетів сервер передусім відправляє пакет обробнику RequestHandler. Під його відповідальність отриманий пакет або обробляється застосунком далі, або у разі некоректного формату відкидається.

ResponseHandler здійснює збірку даних, згідно з протоколом, в єдиний формат пакету даних і передає їх серверу для подальшого відправлення пристрою. Всі необхідні дані для складання пакета, обробник приймає прямо від програми.

Як приклад, код методу ResponseHandler для створення пакету з опкодом «SET_STATE» представлений на рисунку 2.23.

```
    @staticmethod
    def build_set_state(imei: int, payload: int) -> bytes:
        imei = Convert.pack_imei(imei)
        if len(str(payload)) !=
PAYLOAD_LENGTHS.get(pcodes.SET_STATE):
            raise ValueError("Payload length is not valid")
        payload = Convert.pack_to_bytes(payload)
        package_code =
Convert.pack_to_bytes(pcodes.SET_STATE.value)
        complete_package = START_BYTE + imei + package_code +
payload
        return complete_package
```

Рисунок 2.23 – Лістинг методу ResponseHandler

Усередині методів також використовуються допоміжні класи, такі як:

– Convert, призначений для перетворення різних типів даних у байтовий формат;

– Parser, призначений для зворотного перетворення байтів у рядковий або цілий тип даних дозволяючи відновити вихідну інформацію.

Реалізація даних класів досягається за допомогою статичних методів, які можна викликати без створення екземпляра класу. Також подібний метод поділу відповідальності дозволяє підвищити модульність програми, спрощуючи її подальше розширення та підтримку.

Таким чином, обробники даних відіграють ключову роль у забезпеченні ефективної та безпечної роботи протоколу, виключаючи помилки у формуванні запитів до віддалених пристроїв та запобігаючи обробці даних, сформованих згідно протоколу передачі.

РОЗДІЛ 3. ДОДАТКОВІ СЕРВІСИ. ТЕСТУВАННЯ РОЗРОБКИ

Для підвищення функціональності програми були інтегровані додаткові послуги. Вони забезпечують розширені можливості програми, включаючи збереження станів віддалених пристроїв у хмарному сховищі та дистанційне керування цими пристроями через хмару. Також реалізована функція надсилання повідомлень до Telegram каналу через Telegram- бота, що дозволяє користувачам отримувати у доступній формі повідомлення про стан їх пристроїв.

3.1 Хмарний сервіс

Як хмарний сервіс було обрано сторонню платформу ThingSpeak.

ThingSpeak – це хмарна платформа аналітики Інтернету речей, яка дозволяє агрегувати, візуалізувати та аналізувати потоки даних у реальному часі.

Ключовим завданням серверної програми була відправка та читання даних із боку хмарного сервісу та подальша синхронізація з віддаленими пристроями. При обміні повідомлень із сервісом застосовувався підхід REST API [6].

Принцип роботи цього стилю представлений на рисунку 3.1.

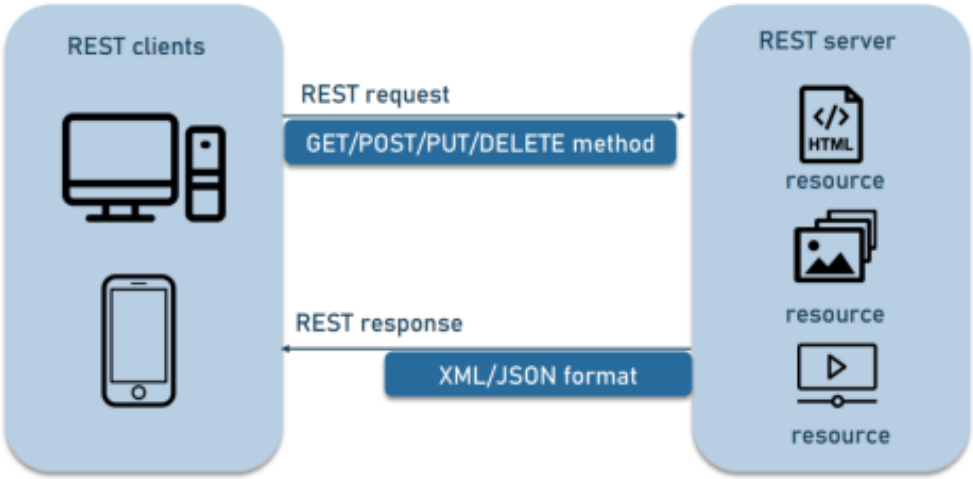

Рисунок 3.1 – Принцип роботи REST API

Перед початком роботи з хмарним сервісом необхідно налаштувати окремі

канали для подальшого відстеження. Кожен канал спеціалізований під конкретний пристрій і призначений для збору даних з нього (рисунок 3.2).

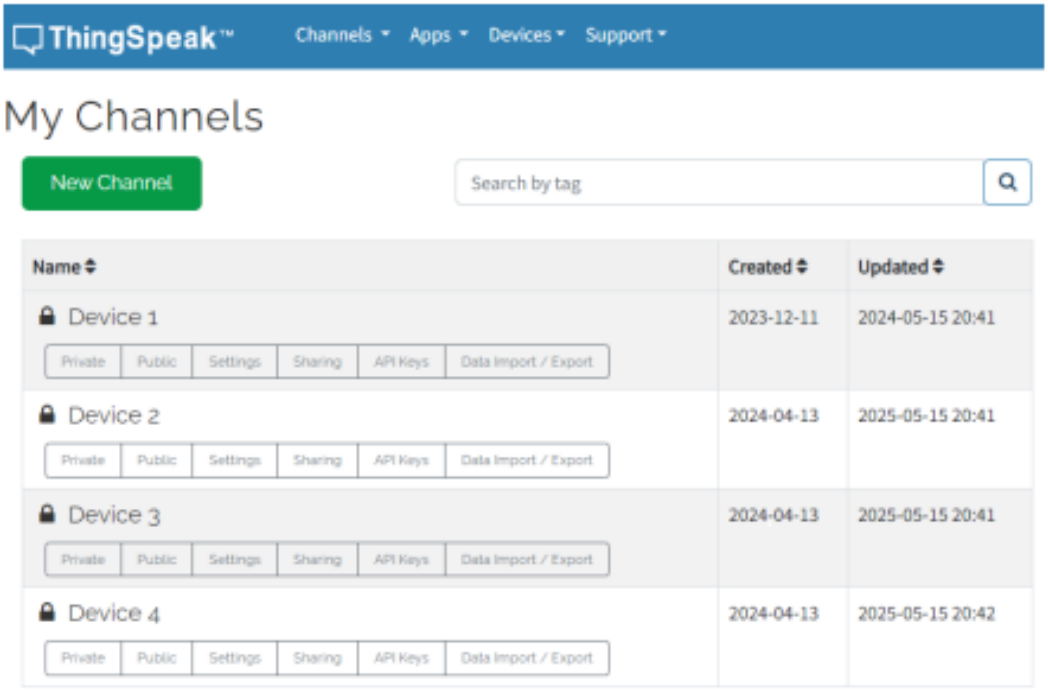

Рисунок 3.2 – Список каналів

У межах кожного каналу налаштовуються параметри полів, які будуть використовуватись для зберігання даних, необхідних для подальшого аналізу та обробки інформації пристрою (рисунок 3.3).

Після успішної реєстрації пристроїв у хмарній інфраструктурі, необхідно зберегти API-ключ кожного каналу всередині програми. Це дозволить здійснювати безпечне читання та модифікацію даних через REST API. API-ключі забезпечують аутентифікацію та авторизацію запитів, гарантуючи, що лише дозволені операції можуть бути виконані з даними пристрою.

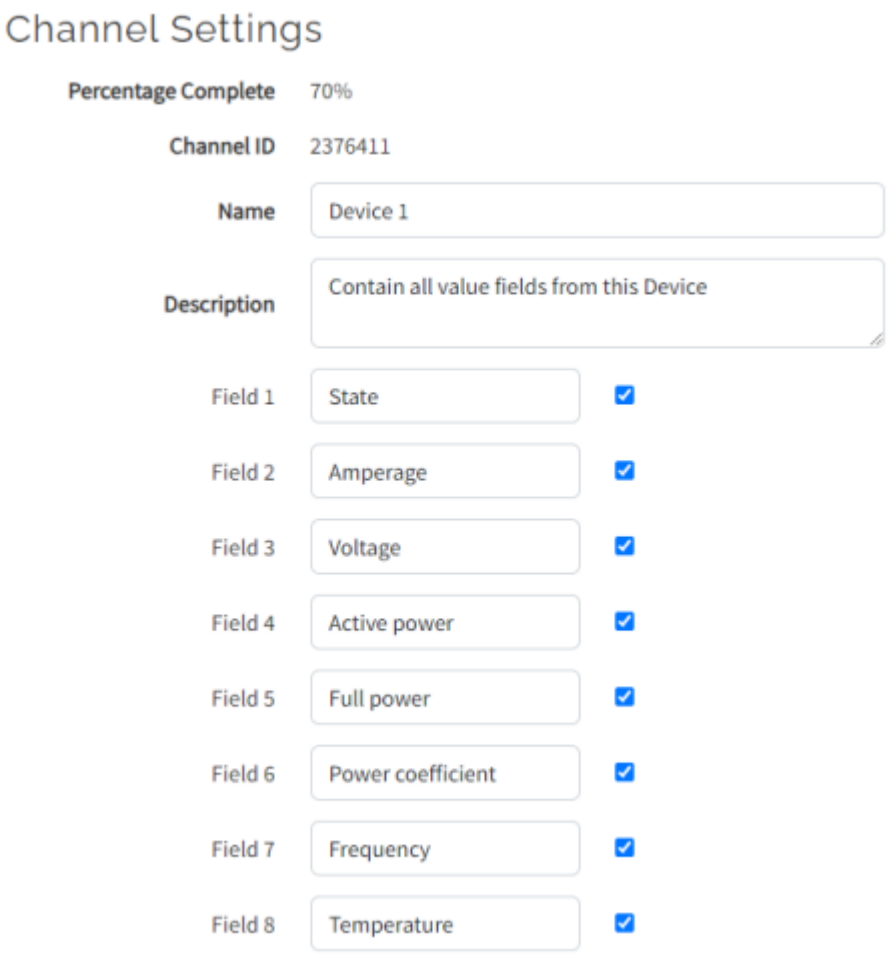

Рисунок 3.3 – Конфігурація полів кожного каналу

Сервер здійснює HTTP GET запит до хмарного сервісу, вказуючи в URL-адресі параметри для отримання необхідних даних, включаючи API-ключ. У відповідь на запит сервер отримує JSON-об'єкт, що містить у собі всі дані, котрі запитуються, що дозволяє ефективно інтегрувати їх в робочий процес програми. Так само здійснюється і завантаження даних пристрою в хмару, додатково вказуючи в URL-адресі необхідні поля пристрою для оновлення, включаючи корисне навантаження.

Код методу класу CloudService для прочитання з хмари останніх змін вказаного поля наведено на рисунку 3.4.

```
    @staticmethod
    def read_last_data_from_field(
        channel_id: str, api_key: str, field_number: Fields
    ) -> tuple[int, datetime, int]:
        """Read the last data from a specific field in a channel

        Args:
            channel_id (str): The channel ID
            api_key (str): The API key
            field_number (Fields): The field number
        Returns:
            The data, created_at and entry_id"""

        if not isinstance(field_number, Fields):
            raise TypeError("Field number should be an instance
of the Fields class")

        field_number = field_number.value
        url =
f"https://api.thingspeak.com/channels/{channel_id}/fields/{field
_number}/last.json?api_key={api_key}"
        res = requests.get(url).json()
        created_at = datetime.strptime(res["created_at"], "%Y-
%m-%dT%H:%M:%SZ")
        entry_id = int(res["entry_id"])
        data = int(res[f"field{field_number}"])
        return data, created_at, entry_id
```

Рисунок 3.4 – Лістинг методу класу CloudService

У разі виявлення змін даних у процесі обміну повідомленнями між сервером та віддаленими пристроями, сервер автоматично синхронізує ці зміни з хмарним сервісом. Крім корисного навантаження, хмара зберігає час зміни поля. Це забезпечує збереження та актуалізацію інформації, гарантуючи цілісність даних та можливість їх відновлення у разі потреби.

Також застосунок із зазначеним у налаштуваннях інтервалом часу періодично надсилає запити до хмарного сервісу. У випадку, якщо у хмарі присутні свіжіші дані, то сервер відправляє мережевий пакет віддаленому пристрою, який потрібно відповідно оновити.

3.2 Telegram сервіс

Важливим завданням автоматизованих систем є оперативне інформування

користувачів про внесені зміни. Для ефективного вирішення цього завдання був використаний Telegram канал для оповіщення про різні зміни станів пристроїв. Повідомлення у канал відправляються через Telegram- бот, з яким сервер підтримує з'єднання по сокетах.

Для забезпечення захищеного з'єднання між сервером та ботом застосовувався комбінований метод шифрування даних. Комбінована система шифрування поєднує переваги симетричних та асиметричних криптосистем:

– симетричне шифрування забезпечує високу швидкість обробки даних та використовує короткі ключі, проте вимагає безпечної передачі ключа;

– асиметричне шифрування дозволяє безпечно обмінюватися ключами у відкритому доступі, але процес шифрування та дешифрування займає більше часу через довгі ключі.

У комбінованій системі симетричний ключ використовується для шифрування самих даних, тоді як асиметричні ключі застосовуються для шифрування симетричного ключа, що забезпечує баланс між безпекою та продуктивністю [7]. Це дозволяє безпечно передавати симетричний ключ та ефективно шифрувати дані.

Приклад схеми шифрування наведений на рисунку 3.5.

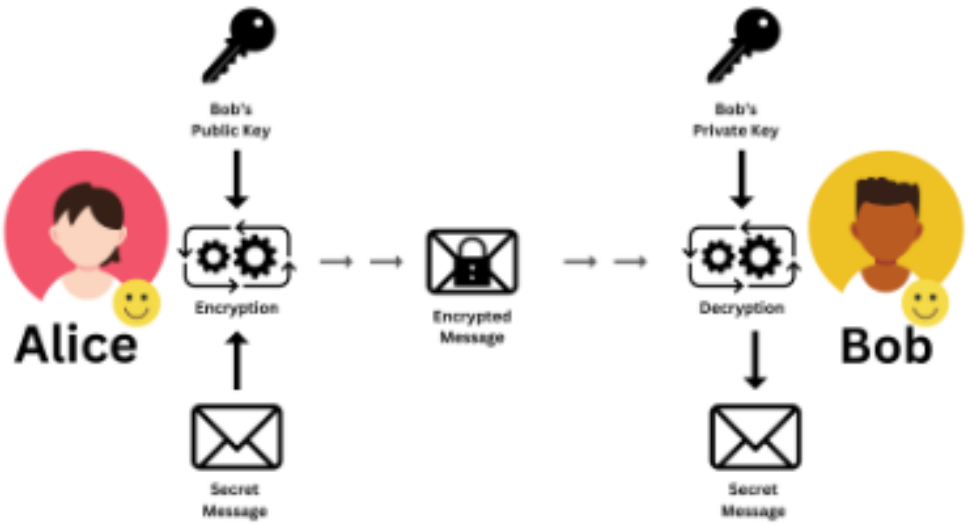

Рисунок 3.5 – Схема комбінованого шифрування

Для шифрування сеансового ключа використовувався асиметричний протокол Діффі-Хелмана (рисунок 3.6), що дозволяє безпечно узгодити загальний секретний ключ навіть за прослуховування каналу зв'язку [8]. У процесі встановлення з'єднання між сервером і ботом за допомогою цього протоколу генерується сеансовий ключ, що має довжину 2048 біта. Дана розмірність ключа дотримується сучасних стандартів безпеки та підвищує стійкість до криптоаналізу.

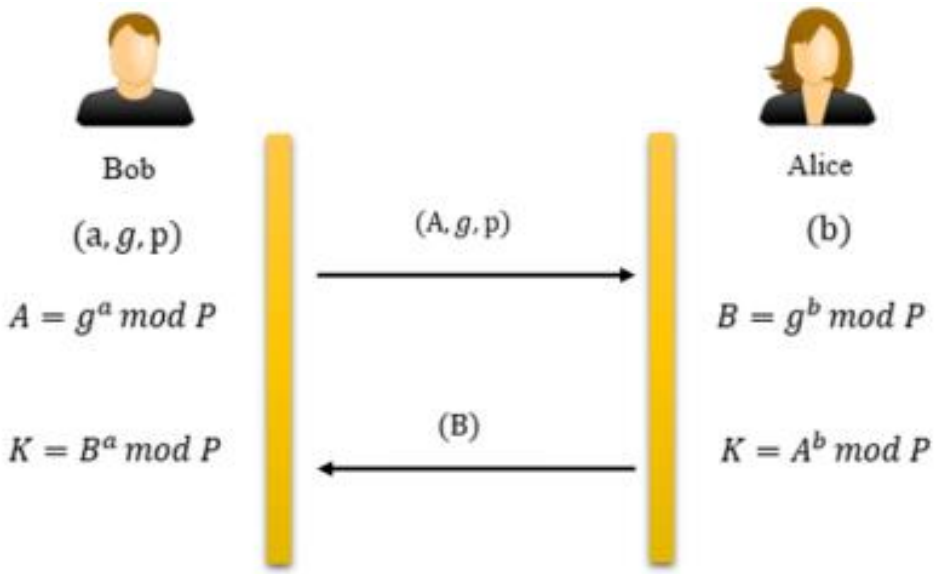

Рисунок 3.6 – Схема протоколу Діффі-Хеллмана

Далі згенерований 2048-бітний ключ обробляється за допомогою хеш-функції SHA-256. Ця дія виконується для отримання унікального хеш-значення, яке потім використовується як ключ для симетричного шифрування. Застосування хеш-функції до ключа підвищує безпеку, забезпечуючи, що навіть якщо вихідний ключ буде скомпрометовано, без знання хеша його не можна буде використовувати для розшифровування даних, забезпечуючи високий рівень захисту інформації.

Як метод симетричного шифрування було обрано протокол Advanced Encryption Standard (далі – AES) з ключем шифрування завдовжки 256 біт, що обумовлено його високим ступенем надійності і широким застосуванням у промисловості безпеки. Симетричний алгоритм блокового шифрування AES є

стандартом, рекомендованим урядовими організаціями та експертами в області кібербезпеки, і використовується для захисту конфіденційної інформації [9]. Схема алгоритму представлена на рисунку 3.7.

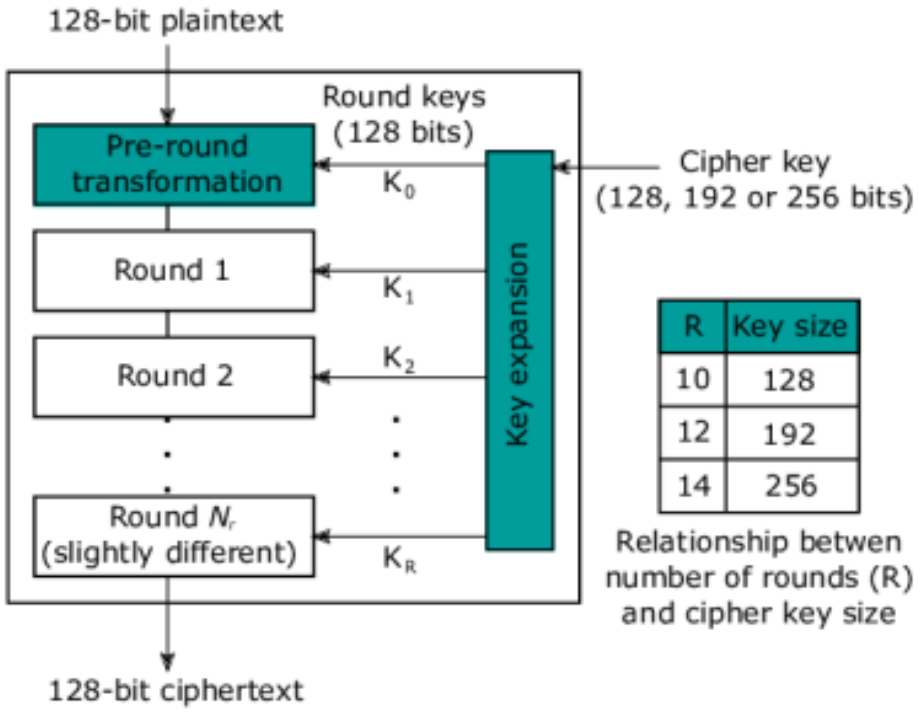

Рисунок 3.7 – Схема AES шифрування

Після завершення налаштування захищеного з'єднання сервер знаходиться в готовності для надсилання даних Telegram боту, які необхідні для сповіщення користувачів про події або зміни станів пристроїв. Сервер формує повідомлення у форматі JSON-об'єктів, що забезпечує зручність та високу читаність даних при розробці. Використання формату JSON дозволяє структурувати інформацію та полегшує її інтеграцію з різними програмними інтерфейсами застосунку. Сервер відправляє запит боту на відправлення повідомлення у канал лише деяку частину операційних кодів. Тим самим регулюючи, які саме дії між сервером і віддаленими пристроями варто додатково висвітлювати перед користувачем.

Наприклад, у разі обробки сервером мережного пакета з операційним кодом «SET_STATE», дане повідомлення з вкладеним корисним навантаженням також надішлеться як повідомлення на Telegram канал вже в опрацьованій формі. Дане повідомлення дасть зрозуміти користувачеві, що вказаний у

повідомленні пристрій, можливо, незаплановано змінив свій внутрішній стан і вимагає додаткової уваги для подальшого аналізу ситуації. На рисунку 3.8 представлені сформовані повідомлення у Telegram каналі.

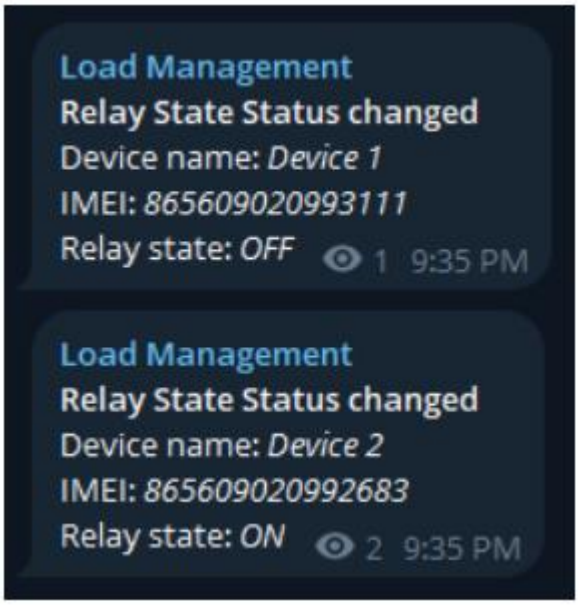

Рисунок 3.8 – Повідомлення в Telegram каналі

Код методу класу TelegramService з підготовки повідомлення для відправки Telegram боту показано на рисунку 3.9.

```
        message = f"<b>Relay State Status changed</b>\nDevice
name: <i>{device_name}</i>\nIMEI: <i>{imei}</i>\nRelay state:
<i>{payload_state}</i>"

        json_message = {
            "delay": False,
            "message": message,
        }
        json_message = json.dumps(json_message)
        return json_message
```

Рисунок 3.9 – Лістинг методу класу TelegramService

Таким чином, була створена захищена система оповіщень, котра використовує Telegram канал для своєчасного інформування користувачів. Завдяки інтеграції з Telegram ботом, сервер може автоматично відправляти

повідомлення про події або зміни, що підвищує оперативність реагування та зручність взаємодії із системою. Це рішення не лише покращує комунікацію з користувачами, а й сприяє підвищенню загальної безпеки за рахунок використання зашифрованого каналу передачі даних.

3.3 Тестування

На заключній стадії розробки важливо провести ретельну перевірку, щоб переконатися в коректній роботі всього функціоналу програми. Саме тому тестування є невід'ємною складовою під час розробки ПЗ, дозволяючи ще ранній стадії виявити непередбачені помилки у роботі програми [10]

Для імітації реальних умов експлуатації було розроблена емульована версія віддаленого пристрою. Ця програма здатна під’єднюватися до сервера застосунку та здійснювати обмін даними відповідно до встановленого протоколу передачі. Це дозволяє не тільки перевірити інтеграцію та взаємодію між компонентами системи, а й гарантувати, що програма буде надійно працювати в реальних сценаріях використання.

Взаємодія з емульованим пристроєм здійснювалася з допомогою консольного інтерфейсу. Емульований пристрій підключається до сервера із зазначеним портом і автоматично відсилає первинний пакет даних із вибраним унікальним ідентифікатором, який слугує для реєстрації в системі серверної програми. Після успішної реєстрації, емульований пристрій знаходиться в режимі очікування подальших команд від сервера.

Код функції зі створення пакету даних згідно з протоколом наведено на рисунку 3.10.

```
    def create_package(imei: int, package_code: int, payload:
str) -> bytes:
        START_BYTE: bytes = b"\xAA"
        imei = Convert.encode_imei(imei)
        if package_code > 32 or package_code < 0:
            raise ValueError("Package code is not valid")

        package_code = Convert.pack_to_bytes(package_code)
        if len(payload) == 15:
            payload = Convert.encode_imei(payload)
        elif len(payload) >= 3 and len(payload) <= 14:
            payload = struct.pack("!I", int(payload))
        else:
            payload = Convert.pack_to_bytes(payload)
        complete_package: bytes = START_BYTE + imei +
package_code + payload
        return complete_package
```

Рисунок 3.10 – Лістинг функції зі створення пакету даних

Після успішної реєстрації пристрою сервер також автоматично надсилає пакет з операційним кодом «GET_STATE» без корисного навантаження. Цей пакет означає, що сервер хоче дізнатися про поточний стан пристрою. У відповідь сервер чекає пакет з тим самим кодом операції, але із прикріпленим станом пристрою у вигляді корисного навантаження. Сервер приймає лише два можливі стани:

– ввімкнений стан кодується цифрою 1,
– вимкнений стан кодується цифрою 0.

Щоб переконатися, що емульований пристрій отримав очікуваний пакет при запиті від сервера, програма в консольному відображенні автоматично виводить мережевий пакет як у байтовому вигляді, так і розділеному і розкодованому на ключові частини протоколу передачі. Таким чином, це полегшує та прискорює процес тестування.

При отриманні емульованим пристроєм мережного пакета з будь-яким операційним кодом, за допомогою пристрою введення в консольному інтерфейсі необхідно вказати необхідний зворотний унікальний номер відправлення операційного коду і корисне навантаження, за необхідності. Значення вводяться у десятковій системі числення. Далі пакет збирається автоматично, виводячи в

консольному інтерфейсі підсумковий його вид у байтовому поданні, і відправляється серверу по сокет з'єднанню.

Розглянувши рисунок 3.11, можна переконатися, що всі стартові процеси ініціалізації між сервером та емульованим пристроєм пройшли успішно.

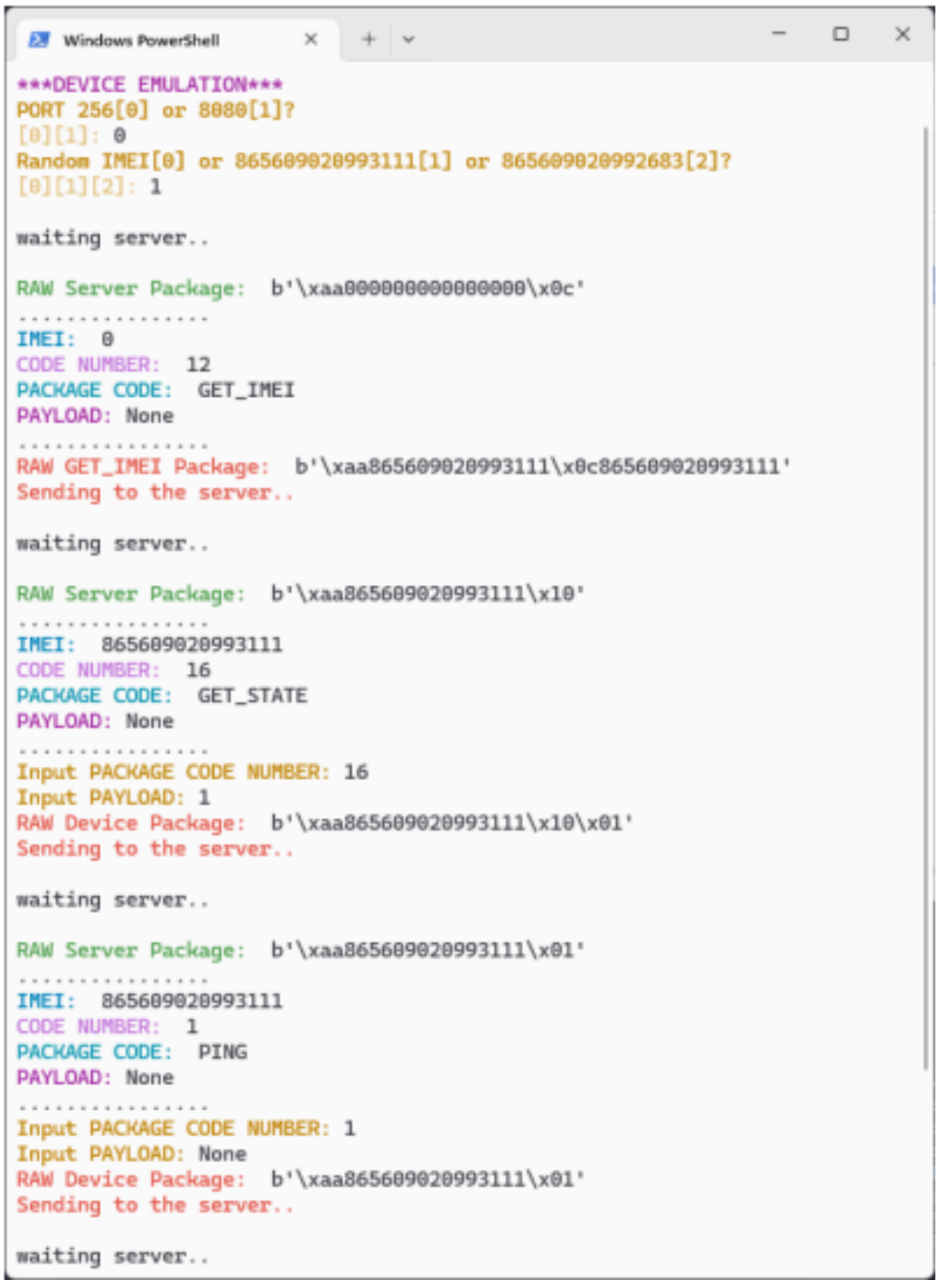

Рисунок 3.11 – Консольний інтерфейс емульованого пристрою

При запиті сервером поточного стану пристрою у відповідь було надіслано

ідентичний операційний код з корисним навантаженням у вигляді увімкненого стану. У процесі отримання даного мережного пакета, серверний застосунок у вкладці «Управління пристроями» переклав стан пристрою, із зазначеним у мережевому пакеті унікальним ідентифікатором, у включений режим і запустив таймер, що відраховує час з моменту включеного стану.

Додатково вручну відправимо з боку серверної програми одну з доступних команд з операційним кодом PING за допомогою кнопки в графічному інтерфейсі і проаналізуємо отриманий результат.

Емульований пристрій успішно прийняв цей пакет і у відповідь відправив ідентичний пакет з необхідним кодом операції за допомогою пристрою введення. Серверний застосунок, зі свого боку, при прийнятті очікуваної відповіді з операційним кодом PING від пристрою, обчислив час між відправкою та отриманням пакета та відобразив в графічному інтерфейсі отримане обчислення у вигляді затримки між сервером та пристроєм у мілісекундах.

Для того, щоб переконатися у коректній роботі діючих процесів між сервером та емульованим пристроєм, також варто проаналізувати дані у вкладці «Журналювання» (рисунок 3.12) всіх основних дій усередині серверної програми.

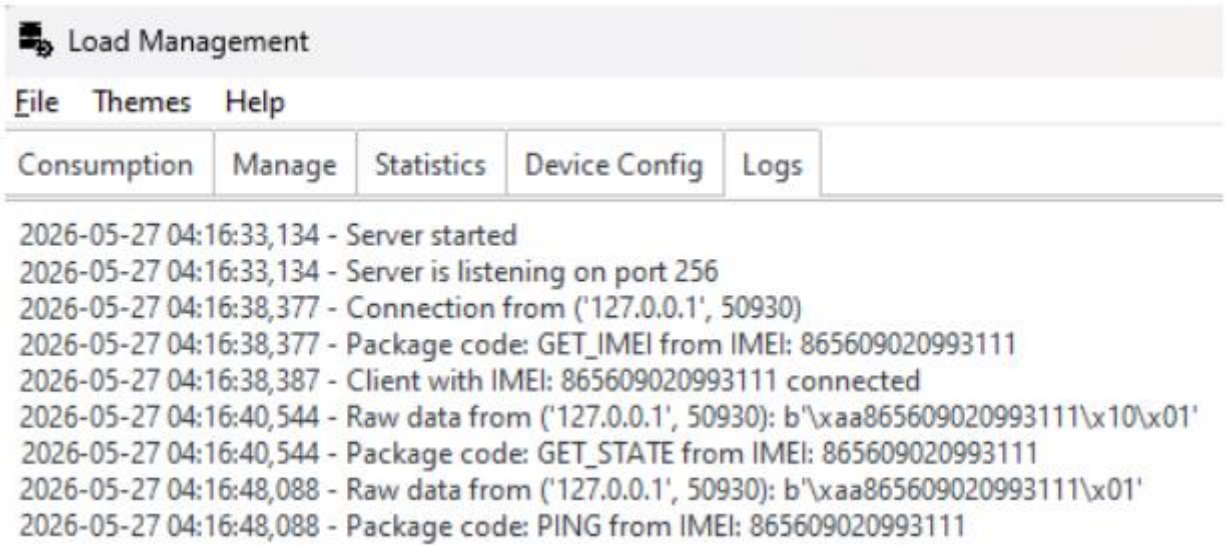

Рисунок 3.12 – Вкладка «Журналювання» мережевих процесів програми

У графічному інтерфейсі програми також для зручності дослідження мережевого трафіку, пакет представлений як байтовий код та розкодовані частини протоколу. Зробивши аналіз отриманих даних, можна зробити висновок про успішну роботу сервера та коректну обробку мережевих пакетів.

Після проведення тестування розробленого застосунку можна дійти висновку, що всі реалізовані функції виконуються належним чином. Коректно працює як графічна оболонка, так і мережева частина програми.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Навчання працюючих і інструктажі з охорони праці

Однією із складових ефективної роботи з профілактики виробничого травматизму є належна підготовка, навчання та підвищення кваліфікації працівників з питань охорони праці. Загальний порядок проведення навчання з питань охорони праці встановлений Законом України «Про охорону праці» (ст. 18. «Навчання з питань охорони праці»).

Виконання вимог Закону України «Про охорону праці» в частині проведення навчання та перевірки знань з питань охорони праці здійснюється відповідно до Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженого наказом Держкомітету України з нагляду за охороною праці 26 січня 2005 р. № 15 (далі — Типове положення).

Нагляд за дотриманням вимог Типового положення здійснюють органи державного нагляду за охороною праці, а координацію та методичний супровід — Головний навчально-методичний центр та навчальні підрозділи експертно-технічних центрів Держгірпромнагляду.

Вивчення предмета «Охорона праці» при підготовці, перепідготовці та підвищенні кваліфікації працівників, які залучаються до виконання робіт з підвищеною небезпекою, на підприємстві регламентується п. 2.3 Типового положення. На підприємствах згідно з п. 1.1 Додатку 3 Типового положення навчання та перевірку знань з питань охорони праці повинні проходити керівники, заступники керівників, головні спеціалісти, керівники основних виробничих та технічних служб, безпосередньо пов’язані з організацією безпечного ведення робіт. Крім цього, згідно з п. 5 Додатку 3, навчання та перевірку знань з питань охорони праці мають проходити керівники, спеціалісти служб охорони праці, члени комісій з перевірки знань з питань охорони праці, особи, відповідальні за технічний стан і безпечну експлуатацію об’єктів підвищеної небезпеки підприємств.

Типове положення встановлює порядок та місце проведення навчання та перевірки знань з питань охорони праці посадових осіб (п. 5.2 та п. 5.3). Посадові особи, перелік яких наведено в п. 5.2, проходять навчання у Головному навчально-методичному центрі Держнаглядохоронпраці. Перевірка знань цієї категорії посадових осіб проводиться комісією, створеною наказом Держгірпромнагляду.

Організацію навчання та перевірки знань з питань охорони праці працівників на підприємстві здійснюють працівники служби кадрів або інші спеціалісти, яким роботодавець доручив організацію цієї роботи. Навчання та перевірка знань з питань охорони праці працівників (виконавців і посадових осіб), які не залучаються до виконання робіт підвищеної небезпеки, проводиться не рідше ніж один раз на три роки. Посадові особи та працівники, які виконують роботи підвищеної небезпеки, проходять спеціальне навчання та перевірку знань відповідних нормативно-правових актів з охорони праці не рідше одного разу на рік.

Посадові особи малих підприємств (п. 5.4), які не мають можливості створити власні комісії з перевірки знань з питань охорони праці та провести навчання з питань охорони праці, проходять навчання та перевірку знань в навчальних закладах, які мають відповідний дозвіл.

Спеціальне навчання з питань охорони праці може проводитись безпосередньо на підприємстві або навчальним закладом, який має відповідний дозвіл. При проведенні такого навчання на підприємстві навчальні плани та програми розробляються з урахуванням конкретних видів робіт, виробничих умов і функціональних обов’язків працівників і затверджуються наказом керівника підприємства.

Періодичність інструктажів, навчання та перевірки знань з питань охорони праці залежить від видів виконуваних робіт та встановлюється Типовим положенням. Перевірка знань з питань охорони праці після проведення спеціального навчання проводиться комісією підприємства.

Якщо на підприємстві неможливо створити комісію з перевірки знань з питань охорони праці (п. 4.4 Типового положення), перевірка знань проводиться

комісією спорідненого підприємства або Теруправління Держгірпромнагляду.

Всі працівники та посадові особи підприємства, включаючи посадових осіб, відповідальних за виконання робіт підвищеної небезпеки (крім зазначених в п. 5.2 та п. 5.3 Типового положення), проходять навчання та перевірку знань з питань охорони праці на підприємстві. Типове положення не зобов'язує, але й не забороняє проводити навчання всіх виконавців та посадових осіб (особливо тих, що виконують роботи підвищеної небезпеки) в навчальних закладах. У нашій країні є багато підприємств, де таке навчання проводиться, і це має позитивні наслідки. Ті витрати, які при цьому несуть підприємства, перекриваються створенням більш безпечних умов праці і в результаті збереженням життя та здоров'я працівників.

Також в навчальних закладах проходять навчання та перевірку знань із загальних питань охорони праці всі посадові особи та фахівці, які проводять інструктажі або навчання підлеглих працівників з питань охорони праці, виконують роботи з проектування об'єктів, а також інші працівники, незалежно від того, передбачено таке навчання Типовим положенням чи ні.

4.2 Санітарно-гігієнічні вимоги до умов праці

На робочих місцях працівників, які відповідальні за експлуатацію сервісу управління механізмом авторських прав на мультимедійні файли, необхідно забезпечити дотримання вимог, затверджених Наказом Мінсоцполітики від 14.02.2018 за № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями».

Приміщення для роботи з ЕОМ мають бути обладнані системами опалення, кондиціонування повітря, або припливно-витяжною вентиляцією. У приміщеннях на робочих місцях мають забезпечуватись оптимальні значення параметрів мікроклімату: температури, відносної вологості та рухливості повітря відповідно до норм та правил, а також ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування», затверджених наказом Мінрегіону від 25.01.2013 р. № 24. Відповідно до санітарних норм мікроклімату виробничих приміщень ДСН

3.3.6.042-99 в офісних приміщеннях, обладнаних ЕОМ, температура повітря повинна становити 22-25°С, відносна вологість повітря – 40-60 %, швидкість руху повітря – не більше 0,1 м/с [15].

Приміщення, призначені для роботи з ЕОМ, повинні мати природне освітлення. У виробничих приміщеннях, обладнаних ЕОМ, необхідно створити належне освітлення. При експлуатації сервісу управління механізмом авторських прав на мультимедійні файли, важливим, з точки зору охорони праці, є забезпечення достатньої величини природного та штучного освітлення, які визначені у НПАОП 0.00-7.15-18 [16]. Природне світло повинно бути бічним, зорієнтованим, як правило, на північ чи північний схід, і забезпечувати коефіцієнт природної освітленості не нижче 1,5%. При виробничій потребі дозволяється експлуатувати ЕОМ у приміщеннях без природного освітлення за узгодженням з органами Держпромгірнагляду та органами й установами санітарно-епідеміологічної служби. Вікна приміщень повинні мати регулювальні пристрої для відчинення, а також жалюзі, штори тощо. Штучне освітлення приміщення з робочими місцями, обладнаними відеотерміналами ЕОМ загального та персонального користування, має бути всеосяжним і рівномірним. У випадку, коли переважають роботи з документами, допускається комбіноване освітлення (додатково до загального освітлення встановлюється світильники місцевого освітлення). Світильники розміщуються збоку від робочих місць (переважно ліворуч), або локально над робочим місцем (при розташуванні відеотерміналів ЕОМ за периметром приміщення). Як джерело світла при штучному освітленні застосовуються, як правило, люмінесцентні лампи. У світильниках місцевого освітлення допускається застосування ламп розжарювання. Рівень освітленості на робочому місці має становити 300-500 лк. При використанні комбінованого освітлення не допускається відблисків на поверхні екрана та збільшення освітлення екрана вище 300 лк.

Орієнтація вікон повинна бути на північ або північний схід, вікна повинні мати жалюзі, які можна регулювати, або штори; не дозволяється розміщувати кабінети обчислювальної техніки у підвальних приміщеннях будинків; кабінети, обладнані комп'ютерною технікою, в навчальних закладах повинні

розміщуватись в окремих приміщеннях з природним освітленням та організованим обміном повітря; стіни, стеля і підлога та обладнання кабінетів комп'ютерної техніки повинні мати покриття із матеріалів з матовою фактурою з коефіцієнтом відбиття: стін — 40- 50 %, стелі — 70 - 80 %, підлоги — 20-30 %, предметів обладнання — 40-60 % (робочого столу — 40-50 %, корпуса дисплею та клавіатури — 30-50 %, стелажів — 40-60 %); поверхня підлоги повинна мати антистатичне покриття та бути зручною для вологого прибирання; забороняється використовувати для оздоблення інтер'єру приміщень комп'ютерних кабінетів полімерні матеріали (дерев'яно-стружкові плити, шпалери, що придатні для миття, плівкові та рулонні синтетичні матеріали, шаровий паперовий пластик та ін.), що виділяють у повітря шкідливі хімічні речовини, які перевищують гранично допустимі концентрації; вміст шкідливих хімічних речовин в повітрі дошкільних та учбових приміщень з комп'ютерною технікою не повинен перевищувати середньодобові концентрації [16].

Організація робочого місця фахівця із експлуатації сервісу повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги».

Відстань від екрана до ока фахівців, які працюють за комп’ютером визначається згідно з вимогами ДСанПіН 3.3.2.007-98.

Рівень шуму не повинний перевищувати: на місцях, де працюють програмісти та оператори ЕОМ, 55 дБА, у лабораторіях, де складаються алгоритми та ведеться робота з документацією – 60 дБА, у машинному залі – 65 дБА, у приміщеннях, де розміщені гучні агрегати обчислювальних машин – 75 дБА.

ВИСНОВКИ

В результаті виконаної роботи було розроблено та протестовано застосунок для обміну даними, котрий є сервером із графічним інтерфейсом та з реалізацією аутентифікації, зберігання і передачі даних з віддаленими пристроями, хмарним сервісом та Telegram-ботом. Сформовано гнучку архітектуру на основі концепції MVC, що дозволяє делегувати завдання між окремими компонентами, спрощуючи при цьому розробку та подальшу підтримку продукту.

Було реалізовано та протестовано протокол передачі пакетів між сервером та віддаленим пристроєм, який забезпечує надійну та безпечну передачу даних під час мережевої взаємодії за допомогою сокет з'єднання. Здійснено аналіз актуальності та обґрунтування вибору реалізованих рішень.

Інтеграція хмарного сервісу була одним із ключових аспектів при розробці серверного застосунку, дозволивши розширити функціональні можливості ПЗ. Розміщення даних віддалених пристроїв додатково у хмарі забезпечує високу доступність та надає зручні інструменти для управління та журналювання стану системи ззовні.

Популярність та значимість месенджерів продовжує зростати і зростатиме далі, оскільки спілкування та обмін інформацією є невід'ємною та важливою частиною життя людини. Тому впровадження Telegram каналу як додаткового методу щодо своєчасного оповіщення користувачів про інформацію, що надійшла з боку віддалених пристроїв, є актуальним рішенням.

У рамках подальшого розвитку ПЗ планується як розширення його функціональних можливостей, так й оптимізація існуючого функціоналу. Варто було б створити реляційну базу даних, яка дозволить комплексно зберігати як системну інформацію, так і дані про віддалені пристрої.

Це забезпечить більш ефективне управління даними та підвищить продуктивність роботи з інформаційними потоками.

ПЕРЕЛІК ДЖЕРЕЛ

1. Socket Programming in Python (Guide) [Електронний ресурс] – Режим доступу: https://realpython.com/python-sockets/ (Дата звертання: 25.04.2026).
2. Benefits of Python Programming Language [Електронний ресурс] – Режим доступу: https://www.developer.com/languages/python/python-benefits/ (Дата звертання: 30.04.2026).
3. MVC: Model, View, Controller [Електронний ресурс] – Режим доступу: https://www.codecademy.com/article/mvc (Дата звертання: 02.05.2026).
4. Python-telegram-bot. Introduction [Електронний ресурс] – Режим доступу: https://docs.python-telegram-bot.org/en/v21.1.1/index.html (Дата звертання: 03.05.2026).
5. Using Matplotlib. Interactive figures [Електронний ресурс] – Режим доступу: https://matplotlib.org/stable/users/explain/figure/interactive.html (Дата звертання: 10.05.2026).
6. What is a REST API? [Електронний ресурс] – Режим доступу: https://www.redhat.com/en/topics/api/what-is-a-rest-api (Дата звертання: 11.05.2026).
7. Hybrid Encryption [Електронний ресурс] – Режим доступу: https://www.techopedia.com/definition/1779/hybrid-encryption (Дата звертання: 12.05.2026).
8. The Diffie-Hellman Key Exchange [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/the-diffie-hellman-key-exchange (Дата звертання: 13.05.2026).
9. Everything You Need to Know About AES-256 Encryption [Електронний ресурс] – Режим доступу: https://www.kiteworks.com/risk-complianceglossary/aes-256-encryption/ (Дата звертання: 17.05.2026).
10. Reasons why software testing is important [Електронний ресурс] – Режим доступу: https://www.nearshore-it.eu/articles/why-is-software-testing-important/ (Дата звертання: 19.05.2026).

11. Буров Є.В., Митник М.М. Комп’ютерні мережі. Підручник. Том другий. Львів: «Магнолія 2006», 2024. 333 с.

12. Методичні вказівки до виконання кваліфікаційної роботи ор Бакалавр для студентів спеціальності 122 – Комп’ютерні науки, всіх форм навчання / укладачі: Готович В.А., Дуда О.М. Никитюк В.В. – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2024. – 43 с.

13. Vyacheslav Nykytyuk, Vasyl Dozorskyy, Nataliia Kunanets, Volodymyr Pasichnyk, Oleksandr Matsiuk, Ihor Bodnarchuk: Electrical Probe-Signal Processing and Criterion for the Determination of Time Parameters of the Teeth Filling Material Polymerization Process in Dentistry. 4th IDDM 2021: Valencia, Spain. P. 54-63

14. Zagorodna, N., Skorenkyy, Y., Kunanets, N., Baran, I., Stadnyk, M. Augmented Reality Enhanced Learning Tools Development for Cybersecurity Major. CEUR Workshop Proceedings., 2022, 3309, pp. 25–32. https://ceur-ws.org/Vol-3309/short1.pdf.

15. Заікіна Д., Глива В. Основи охорони праці та безпека життєдіяльності. 2019. URL: https://doi.org/10.31435/rsglobal/001 (дата звертання: 29.05.2026).

16. Безпека в надзвичайних ситуаціях. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання / укл.: Стручок В. С. Тернопіль: ФОП Паляниця В. А., 2022. 156 с.

ДОДАТКИ

Додаток А

Програмний код основного класу застосунку

```
    class Application(master):
        def __init__(self, server: Server, configs: Configs):
            if os.name == "posix":  # macOS and linux
                super().__init__()
            elif os.name == "nt":
                super().__init__(themename="litera")

            if not isinstance(server, Server):
                raise TypeError("Server should be an instance of
the Server class")
            if not isinstance(configs, Configs):
                raise TypeError("Configs should be an instance
of the Configs class")

            width = 1024
            heigth = 600
            self.title("Load Management")
            self.geometry(f"{width}x{heigth}")
            self.minsize(width=width, height=500)
            self.resizable(width=True, height=True)
            self.protocol("WM_DELETE_WINDOW", self.on_closing)
            self.active_tab = None
            self.configs = configs
            self.server = server
            self.server_thread = None
            self.key_binding()
            self.status_bar = StatusBar(self)
            self.status_bar.pack(side=tk.BOTTOM, fill=tk.X)
            self.menu_bar = MenuBar(self)
            self.config(menu=self.menu_bar)
            self.create_tabs()
            self.autorun_server()
```