Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Кіберфізична система "Розумні ліхтарі". Серверна частина
Назва теми

КвРКІ 210236.21.02.03 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент IV курсу, група КІ2-21-2 _____ Максим КОЗИР
Підпис — Ініціали, прізвище

Керівник _____ Катерина БЕРЕЗЬКА
Підпис, дата — Ініціали, прізвище

Нормоконтролер _____ Тетяна КИСІЛЬ
Підпис, дата — Ініціали, прізвище

До захисту допускаю:
зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Ольга ПАВЛОВА
Підпис — Ініціали, прізвище

«16» червня 2025 р.

Хмельницький 2025

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Тетяна КИСІЛЬ, доцент кафедри КІС		
Антиплагіат	Андрій НІЧЕПОРУК, доцент кафедри КІС		

7. Дата видачі завдання «_10_»_____01_____2025 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2025	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	02.02.2025	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	04.03.2025	виконано
4	Робота над розділом 2 – вибір компонентів для проєктування Серверної частини кіберфізичної системи «Розумні ліхтарі»	15.04.2025	виконано
5	Робота над розділом 3 – проєктування Серверної частини кіберфізичної системи «Розумні ліхтарі»	29.04.2025	виконано
6	Оформлення пояснювальної записки згідно вимог	22.05.2025	виконано
7	Попередній захист ВКР	23.05.2025	виконано
8	Захист ВКР на засіданні ЕК	Червень 2025 року	

Студент _____ Максим КОЗИР
Підпис Ініціали, прізвище

Керівник роботи _____ Катерина БЕРЕЗЬКА
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 210236.21.02.03 ПЗ	Пояснювальна записка	57		
			Графічні матеріали			
2		КвРКІ 210236.21.02.03 Е8	Структура кіберфізичної системи	1		
3		КвРКІ 210236.21.02.03 Е8	Модулі керування системою	1		
4		КвРКІ 210236.21.02.03 Е8	Схема результатів роботи програми	1		

КвРКІ 210236.21.02.03 ВП

Зм	Арк	№ докум	Підпис	Дата
Розробив		Козир М.		
Перевір.		Березька К.		
Н. контр.		Кисіль Т.М.		
Затв.		Павлова О.О		

Відомість проекту

Літера	Аркуш	Аркушів
У	1	1

ХНУ, КІ2-21-2

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Кіберфізична система «Розумні ліхтарі» . Серверна частина».

Автор роботи: Максим КОЗИР.

Керівник роботи: Катерина БЕРЕЗЬКА.

Пояснювальна записка: 57 с., 9 рис., 6 табл., 3 дод., 52 джерел.

Графічна частина: 3 креслення.

РОЗУМНІ ЛІХТАРІ, КІБЕРФІЗИЧНА СИСТЕМА, СЕРВЕРНА ЧАСТИНА, ІНТЕРНЕТ РЕЧЕЙ, АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, БАЗА ДАНИХ, API, РОЗУМНЕ МІСТО.

Метою дипломної роботи є розробка та дослідження серверної частини кіберфізичної системи «Розумні ліхтарі», яка забезпечує прийом, обробку та зберігання даних від освітлювальних пристроїв і датчиків.

Об'єктом дослідження є процеси функціонування та взаємодії компонентів у серверній частині кіберфізичної системи «Розумні ліхтарі»

Предметом дослідження є архітектурні рішення, моделі даних, програмні інтерфейси, алгоритми обробки інформації та керування для серверної частини системи «Розумні ліхтарі».

Під час проведення даного дослідження були використанні методи системного аналізу літературних джерел та існуючих рішень, об'єктно-орієнтоване проєктування програмного забезпечення,принципи побудови розподілених систем та бази даних, а також моделювання процесів функціонування сервеної частини.

Підпис студента

30.05.2025

Дата

ЗМІСТ

Зм.	Арк.	№докум.	Підпис	Дата	КвРКІ.210236.21.02.03 ПЗ		
Виконав		Максим КОЗИР			Кіберфізична система "Розумні ліхтарі". Серверна частина	Літера: у	Аркуш: 2 / Аркушів: 72
Перевір.		Березька К.					
Н.контр.		Тетяна КИСІЛЬ				ХНУ КІ2-21-2	
Затвер.		Ольга ПАВЛОВА					

ВСТУП

Сучасне місто неможливо уявити без вуличного освітлення. Воно не лише створює візуальний комфорт у темний час доби, але й відіграє ключову роль у забезпеченні громадської безпеки та нормального функціонування транспортної інфраструктури. Проте, традиційні системи освітлення, які часто працюють за простими графіками або реагують лише на загальний рівень освітленості, стають дедалі менш ефективними в умовах зростаючих вимог до енергозбереження та оптимізації міського господарства. Вони споживають надлишкову енергію, освітлюючи порожні вулиці, а їх обслуговування та діагностика несправностей потребують значних ресурсів.

Революційні зміни в цій сфері стали можливими завдяки розвитку концепції кіберфізичних систем, оскільки їх застосування дозволяє перетворити звичайні вуличні ліхтарі на інтелектуальні елементи міської мережі - «Розумні ліхтарі». Такі системи поєднують фізичні пристрої (самі ліхтарі, оснащені сенсорами та модулями зв'язку) з потужними обчислювальними та комунікаційними можливостями, що відкриває шлях до гнучкого, адаптивного та ефективного керування освітленням.

Ключовою перевагою «Розумних ліхтарів» є їхня здатність працювати в режимі реального часу: Адаптувати яскравість відповідно до присутності людей чи автомобілів, реагувати на зміну погодних умов, автоматично виявляти та повідомляти про власні несправності. Це не лише призводить до суттєвої економії електроенергії та зниження витрат на обслуговування, але й підвищує якість освітлення та безпеку на вулицях, а також створює платформу для збору цінних даних про міське середовище.

Однак, для реалізації повного потенціалу такої складної розподіленої системи необхідний потужний центр керування та обробки інформації. Саме серверна частина виступає «Мозком» усієї системи «Розумні ліхтарі». Вона відповідає за агрегацію даних від тисяч ліхтарів, їх аналіз, виконання алгоритмів інтелектуального керування, надсилання команд окремим пристроям чи групам, а

також забезпечення інтерфейсів для моніторингу, адміністрування та взаємодії з іншими міськими службами. Від ефективності, надійності та функціональності серверної частини безпосередньо залежить успішність функціонування всієї системи інтелектуального освітлення. Таким чином, розробка та дослідження серверної частини для кіберфізичної систем «Розумні ліхтарі» є актуальним науково - практичним завданням.

Метою дипломної роботи є розробка та дослідження серверної частини кіберфізичної системи «Розумні ліхтарі», яка забезпечує прийом, обробку та зберігання даних від освітлювальних пристроїв і датчиків, а також реалізує інтелектуальні алгоритми керування вуличним освітленням.

Для досягнення поставленої мети в роботі вирішуються ось такі основні завдання: Проаналізувати сучасний стан систем інтелектуального освітлення, роль кіберфізичних систем у їх побудові та сформулювати детальні вимоги до серверної частини системи «Розумні ліхтарі»; обґрунтувати вибір архітектурного підходу та технологічного стеку для реалізації серверної частини «Розумні ліхтарі»; спроєктувати архітектуру серверної частини, включаючи розробку структури бази даних для зберігання інформації системи та програмного інтерфейсу для взаємодії компонентів; розробити та описати ключові алгоритми обробки даних та логіки керування освітленням; провести програмну реалізацію основних модулів серверної частини та розробити методику її тестування для перевірки функціональності та працездатності. Об’єктом дослідження є процеси функціонування та взаємодії компонентів у серверній частині кіберфізичної системи «Розумні ліхтарі».

Предметом дослідження є архітектурні рішення, моделі даних, програмні інтерфейси, алгоритми обробки інформації та керування, що застосовуються для розробки та функціонування серверної частини «Розумні ліхтарі».

Методи дослідження. Для вирішення поставлених завдань у роботі використано методи системного аналізу науково-технічної літератури та існуючих рішень у сфері інтелектуального освітлення та Інтернету речей; методи

об'єктно-орієнтованого проєктування для розробки архітектури програмного забезпечення; принципи побудови розподілених систем та клієнт-серверної взаємодії; методи проєктування баз даних; а також методи програмної реалізації та тестування для перевірки працездатності розроблених рішень.

Практичне значення отриманих результатів полягає в тому, що розроблені в дипломній роботі архітектурні та програмні рішення для серверної частини можуть бути використанні як основа для створення або модернізації систем інтелектуального вуличного освітлення в містах та населених пунктах. Запропоновані підходи дозволяють підвищити енергоефективність, оптимізувати процеси моніторингу та обслуговування освітлювальної інфраструктури. Результати роботи також можуть бути корисними для інженерів та розробників, що займаються проєктуванням та впровадженням систем класу «Розумне місто» та Інтернету речей.

1 КІБЕРФІЗИЧНА СИСТЕМА АДАПТИВНОГО ЗАСТОСУВАННЯ МОНІТОРИНГОВИХ ЕЛЕМЕНТІВ РОЗВІДУВАЛЬНОГО БПЛА ТА ПОСТАНОВКА ЗАДАЧІ ЩОДО ЇЇ УДОСКОНАЛЕННЯ

1.1 Аналіз сучасного стану систем інтелектуального освітлення та роль кіберфізичних систем

Належне функціонування міської системи вуличного освітлення є невід'ємною та фундаментальною складовою безпечного, комфортного та економічно активного міського середовища. Воно виконує не лише утилітарну функцію створення візуального комфорту в темний час доби, але й є ключовим фактором у забезпеченні громадської безпеки, значно знижуючи ризики дорожньо-транспортних пригод, рівень злочинності та вандалізму, а також підтримуючи економічну активність у вечірні та нічні години. Проте, переважна більшість існуючих систем, що були спроєктовані та впроваджені десятиліття тому, і досі домінують у багатьох містах світу, характеризуються низкою суттєвих технологічних, операційних та економічних недоліків. Основна, системна проблема полягає в їхній застарілій, примітивній логіці роботи, яка є абсолютно неадаптивною до реальних потреб. Як правило, вона обмежується централізованим увімкненням та вимкненням цілих районів або магістралей за допомогою простих електромеханічних реле часу або загального фотодатчика, що реагує лише на загальний рівень природної освітленості. Такий підхід абсолютно не враховує реальну, надзвичайно мінливу та гетерогенну динаміку міського життя, що призводить до колосального, часто більш ніж на 50%, перевитрачання електроенергії на освітлення вулиць, парків та територій з низькою або нульовою інтенсивністю руху пішоходів та транспорту в певні години доби. Окрім прямих та значних фінансових втрат для муніципальних бюджетів, це створює значне навантаження на енергосистему та негативно впливає на екологію через

надлишкові викиди CO2, що є побічним продуктом генерації електроенергії [14, 25].

Додатковими системними недоліками, що є не менш значущими, є повна відсутність гнучкості в індивідуальному управлінні окремими освітлювальними приладами та надзвичайна складність процесів оперативного виявлення та усунення несправностей. Діагностика в таких системах зводиться до періодичних, затратних візуальних оглядів, що виконуються ремонтними бригадами, або до реактивного реагування на скарги від мешканців. Такий підхід є неефективним та може призводити до тривалого простою непрацюючих ліхтарів, створюючи так звані «темні плями» на мапі міста, що безпосередньо підвищує рівень небезпеки для громадян та створює соціальну напругу [4, 32].

Відповіддю на ці комплексні виклики стала концепція «Розумного освітлення», яка є одним з ключових та найбільш зрілих напрямків розвитку глобальної ініціативи «Розумне місто». Вона передбачає глибоке впровадження сучасних інформаційно-комунікаційних технологій для створення інтелектуальних, адаптивних, енергоефективних та економічно вигідних систем управління міською освітлювальною інфраструктурою. Фундаментальну роль у реалізації таких систем відіграють кіберфізичні системи [1, 5].

Кіберфізична система − це складна, багатокомпонентна інженерна система, що характеризується глибокою, тісною та синергетичною інтеграцією обчислювальних ресурсів із фізичними процесами та об’єктами реального світу. У кіберфізичної системи обчислювальна система, як правило, є географічно розподіленою по всій фізичній системі, що є її носієм, та тісно пов’язана з її елементами через безперервні цикли зворотного зв'язку, що описуються тріадою «сприйняття - обчислення - дія» [25, 11].

Основними компонентами будь-якої кіберфізичної системи є фізичні елементи, до яких належать сенсори (датчики), що виконують функцію «органів чуття» системи, збираючи різноманітні дані про стан фізичних процесів та навколишнього середовища, та виконавчі механізми, що здійснюють

цілеспрямований фізичний вплив на ці процеси. Ці елементи об’єднані в єдину мережу за допомогою комунікаційних компонентів, що забезпечують надійний, безпечний та своєчасний обмін даними між усіма елементами системи. Кібернетичні компоненти, що включають програмне забезпечення, математичні моделі, складні алгоритми та обчислювальні платформи, відповідають за збір, агрегацію, зберігання, обробку, глибокий аналіз даних, а також за прийняття інтелектуальних рішень та формування керуючих впливів, які передаються на актуатори [16, 22].

У контексті системи «Розумні ліхтарі» такий підхід дозволяє здійснити парадигматичну трансформацію кожного окремого вуличного ліхтаря з пасивного, некерованого споживача електроенергії на активний, інтелектуальний та комунікаційний вузол єдиної міської цифрової мережі.

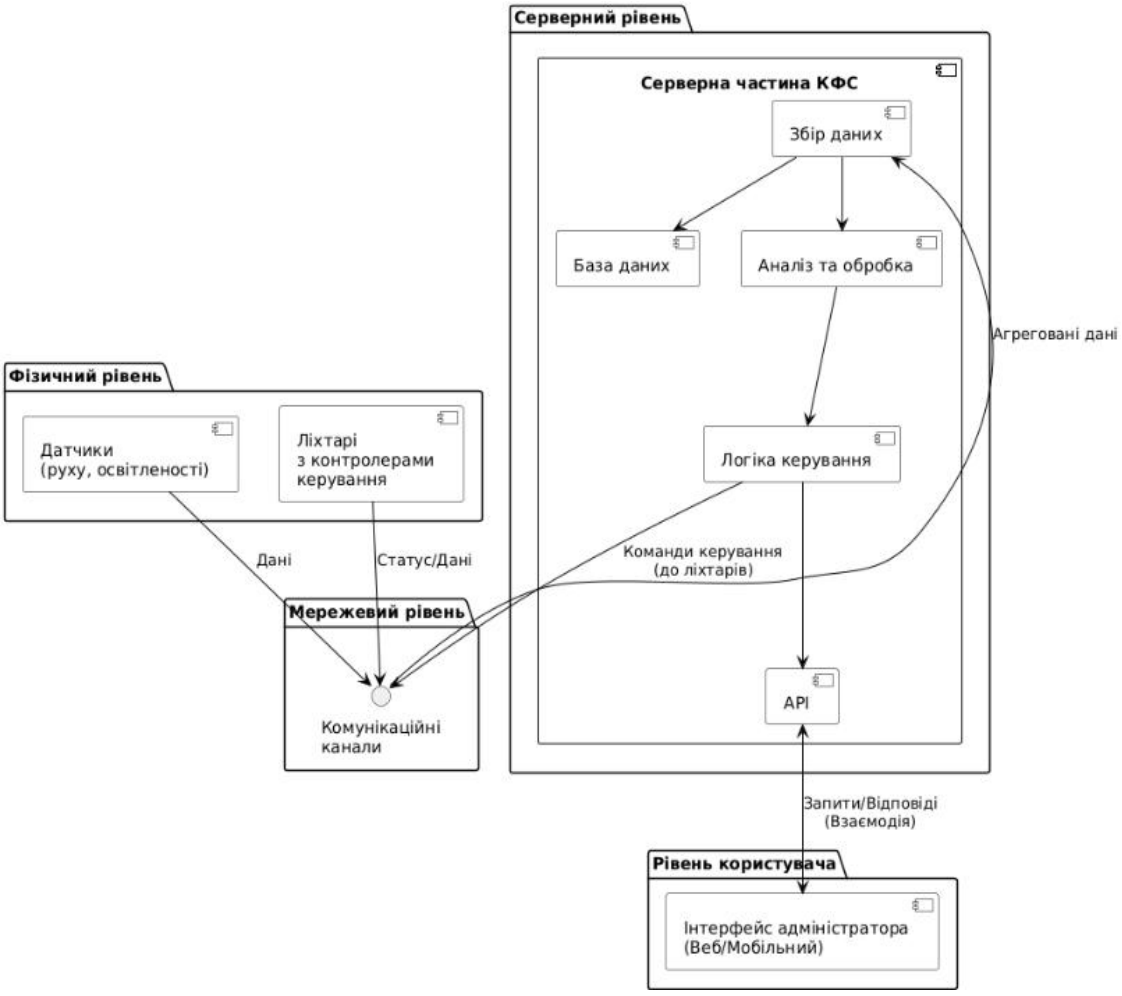


Рисунок 1.1 – Узагальнена структура кіберфізичної системи «Розумні ліхтарі»

Узагальнену структуру такої кіберфізичної системи, що представлена на рисунку 1.1, доцільно розглядати через кілька ієрархічних та взаємопов'язаних рівнів.

На найнижчому, фізичному рівні, знаходяться безпосередні виконавці та джерела даних – це сучасні світлодіодні ліхтарі, оснащені спеціалізованими мікроконтролерами для точного керування яскравістю (диміювання), та цілий набір різноманітних датчиків. Це можуть бути пасивні інфрачервоні датчики руху для детектування пішоходів, мікрохвильові радари для виявлення автомобілів, високоточні датчики освітленості для вимірювання рівня природного світла, а в більш просунутих конфігураціях – навіть сенсори для моніторингу якості повітря, рівня шуму чи температури, що перетворює ліхтар на багатофункціональну метеостанцію [44, 21].

Наступний, мережевий рівень, забезпечує стабільну та надійну двонаправлену комунікацію між фізичним рівнем та центральною серверною частиною. З огляду на географічну розподіленість, складний міський рельєф та вимоги до енергоефективності, тут можуть застосовуватися різноманітні технології бездротового зв'язку, такі як далекобійні мережі з низьким енергоспоживанням LoRaWAN або NB-IoT, стільникові мережі нового покоління 5G або спеціалізовані самоорганізуючі mesh-мережі. Цей рівень відповідає за надійну та своєчасну передачу телеметричних даних та статусів від ліхтарів на сервер, а також за гарантовану доставку команд керування у зворотному напрямку.

Центральним ядром, або «мозком», усієї системи є серверний рівень. Він реалізує найскладніші та найбільш обчислювально-інтенсивні функції: здійснює збір та агрегацію даних від тисяч пристроїв, забезпечує їх довготривале та структуроване зберігання у спеціалізованих базах даних, проводить глибокий аналіз інформації за допомогою складних алгоритмів машинного навчання та статистичного аналізу, реалізує адаптивну логіку керування для формування

оптимальних режимів роботи та надає стандартизовані програмні інтерфейси (API) для взаємодії з іншими частинами системи.

Нарешті, на найвищому, користувацькому рівні, знаходяться інструменти для ефективної взаємодії людини з системою. Це можуть бути повнофункціональні веб-інтерфейси для адміністраторів та операторів, спеціалізовані мобільні додатки для ремонтних бригад, що дозволяють отримувати завдання та діагностичну інформацію в польових умовах, а також аналітичні панелі (дашборди), що дозволяють здійснювати моніторинг, налаштування та керування системою в режимі реального часу, візуалізуючи складні дані у вигляді зрозумілих графіків та карт.

Застосування такої комплексної архітектури надає системам «розумного освітлення» низку кардинальних та незаперечних переваг у порівнянні з традиційними [18]. Перш за все, це можливість реалізації гнучкого, адаптивного керування освітленням в режимі реального часу, що веде до значного, часто до 80%, підвищення енергоефективності та, як наслідок, до суттєвого скорочення бюджетних витрат. Іншою важливою перевагою є централізований моніторинг стану всієї системи та автоматизована діагностика несправностей, що дозволяє перейти від застарілого реактивного до сучасного предиктивного (прогнозного) обслуговування, попереджаючи вихід обладнання з ладу. Крім того, як вже зазначалось, кіберфізична система «Розумні ліхтарі» створюють унікальну інфраструктурну платформу для збору та аналізу цінних даних про міське середовище, які можуть бути використані для оптимізації транспортних потоків, планування міської забудови та підвищення загальної якості життя громадян [13, 25]. Таким чином, глибоке розуміння принципів побудови та функціонування кіберфізичних систем є ключовим для аналізу сучасного стану та перспектив розвитку інтелектуального міського освітлення, а також для коректної та повної постановки задачі на розробку ефективної, надійної та масштабованої серверної частини для таких систем.

Як видно з наведеної на рис. 1.1 схеми, система «Розумні ліхтарі» складається з кількох взаємопов’язаних рівнів та компонентів: Фізичний рівень - цей рівень безпосередньо взаємодіє з навколишнім середовищем та реалізує основну функцію освітлення. Він включає: Датчики (руху, освітленості), призначені для збору інформації про поточні умови, та «Ліхтарі з контролерами керування», що є виконавчими пристроями системи, оснащеними світлодіодними джерелами світла та мікроконтролерами для дистанційного керування; мережевий рівень: Забезпечує комунікацію між фізичним рівнем та серверною частиною. Дані від датчиків («Дані», «Статус/Дані») та статусна інформація від ліхтарів передаються через відповідні комунікаційні канали на сервер, формуючи «Агреговані дані». У зворотному напрямку передаються «Команди керування» від сервера до контролерів ліхтаріакселерометр; серверний рівень: Є центральним ядром системи. Ключовими компонентами та функціями серверного рівня є: «Збір даних», «База даних» для зберігання інформації, «Аналіз та обробка» для застосування алгоритмів, «Логіка керування» для формування оптимальних режимів роботи, та «API» для взаємодії з іншими частинами системи; рівень користувача. Надає інструменти для взаємодії людини з системою, через «Інтерфейс адміністратора (Веб/Мобільний)», що дозволяє здійснювати моніторинг, налаштування та керування системою.

1.2 Формулювання вимог та постановка задачі розробки серверної частини для системи «Розумні ліхтарі»

Проведений у попередньому підрозділі системний аналіз сучасного стану технологій інтелектуального освітлення та фундаментальної ролі кіберфізичних систем у їх побудові дозволяє перейти до наступного логічного етапу - детального та формалізованого визначення вимог до ключового компонента такої системи - її серверної частини. Чітке, повне та несуперечливе формулювання вимог є наріжним каменем та необхідною умовою для

успішного проєктування, розробки, тестування та подальшого впровадження будь-якого ефективного програмного забезпечення. Серверна частина кіберфізичної системи «Розумні ліхтарі» виконує цілу низку критично важливих функцій, які в сукупності забезпечують інтелектуальність, економічну ефективність та операційну стабільність всієї інфраструктури вуличного освітлення [52, 3].

По-перше, однією з фундаментальних задач є забезпечення надійної агрегації даних від величезної кількості (від сотень до десятків тисяч) кінцевих пристроїв. Ці пристрої є географічно розподіленими, часто працюють в умовах нестабільного бездротового зв'язку та можуть використовувати різні протоколи передачі даних. Серверна частина повинна не просто пасивно приймати дані, але й активно керувати цим процесом, забезпечуючи їхню цілісність, валідацію та своєчасність доставки.

По-друге, не менш важливою є функція обробки та глибокого аналізу зібраної інформації. Цей процес виходить далеко за межі простого збереження даних. Він включає фільтрацію шумів, перевірку даних на достовірність, виявлення аномалій та застосування складних математичних та статистичних алгоритмів. Наприклад, сервер має аналізувати часові ряди даних з датчиків руху для побудови патернів міської мобільності та прогнозування потреби в освітленні, виявляти нетипові режими енергоспоживання окремих ліхтарів, що можуть свідчити про їхню деградацію або несправність, та розраховувати точні показники енергоспоживання для об'єктивної оцінки ефективності системи [22, 44].

По-третє, на основі результатів аналізу даних серверна частина повинна приймати обґрунтовані рішення та генерувати відповідні команди керування. Це може бути як відносно просте регулювання яскравості окремих ліхтарів чи їх груп відповідно до заздалегідь заданих статичних сценаріїв (наприклад, зменшення яскравості до 30% на другорядних вулицях після опівночі), так і реалізація значно складніших адаптивних стратегій. Такі стратегії можуть

враховувати не лише час доби, але й погодні умови, дані з датчиків руху в реальному часі та навіть інформацію, отриману з інших міських систем (наприклад, з системи керування дорожнім трафіком для підсвічування маршрутів громадського транспорту, що запізнюється) [4, 20].

По-четверте, сервер має забезпечувати потужні та інтуїтивно зрозумілі засоби моніторингу та візуалізації. Оператори та адміністратори системи повинні мати можливість у реальному часі бачити стан всієї мережі освітлення на інтерактивній геоінформаційній карті, отримувати детальну телеметричну та діагностичну інформацію про кожен окремий ліхтар, переглядати історію подій та генерувати комплексні звіти про роботу системи, виявлені несправності та досягнуту економію енергії. Функція автоматичної генерації та надсилання сповіщень про критичні події (наприклад, масові відключення, системні збої, вихід з ладу обладнання) для обслуговуючого персоналу також є надзвичайно важливою [13, 37].

Виходячи з вищеописаних функцій та завдань, до серверної частини кіберфізичної системи «Розумні ліхтарі» можна висунути деталізований перелік функціональних та нефункціональних вимог. До ключових функціональних вимог належить надійний збір даних від сенсорів у режимі, близькому до реального часу; агрегація та валідація цих даних; застосування аналітичних алгоритмів; прийняття рішень та формування команд керування; реалізація адаптивних стратегій; надання засобів моніторингу та візуалізації; наявність системи сповіщень; надання документованого програмного інтерфейсу для зовнішньої інтеграції; підтримка поширених IoT-протоколів; реалізація надійних механізмів автентифікації та авторизації; а також ефективне зберігання великих обсягів даних, зокрема часових рядів [2, 28].

Не менш важливими є нефункціональні вимоги, які визначають якісні атрибути системи. Перш за все, це масштабованість, тобто здатність системи підтримувати зростаючу кількість підключених пристроїв без суттєвої втрати продуктивності. Надійність та відмовостійкість мають гарантувати стабільне

функціонування системи з мінімальним часом простою, що може досягатися за рахунок резервування ключових компонентів. Продуктивність визначає швидкість обробки повідомлень та реакції на команди. Безпека вимагає впровадження комплексних заходів, включаючи шифрування, захист від атак та аудит доступу. Гнучкість та конфігурованість дозволяють легко адаптувати сценарії освітлення та оновлювати алгоритми. Нарешті, сумісність та інтероперабельність, що досягаються за рахунок підтримки відкритих стандартів, забезпечують можливість інтеграції з обладнанням різних виробників та іншими міськими платформами.

Таким чином, ключова технологічна складність і основна функціональна цінність системи «Розумні ліхтарі» зосереджена саме в її серверній частині. Розробка програмного забезпечення, здатного ефективно впоратися з усіма перерахованими завданнями, є основною науково-технічною проблемою, яку необхідно вирішити для успішного впровадження та експлуатації таких систем. Від якості архітектури, надійності алгоритмів та коректності реалізації серверної платформи безпосередньо залежить, чи зможе система «Розумні ліхтарі» повною мірою реалізувати свій величезний потенціал щодо енергоефективності, надійності та покращення якості життя в місті.

Отже, основне завдання даної дипломної роботи полягає у розробці та дослідженні таких архітектурних та програмних рішень для серверної частини, які б дозволили ефективно керувати масштабними, географічно розподіленими мережами «Розумних» ліхтарів, забезпечуючи при цьому високу надійність, продуктивність, гнучкість та безпеку [31, 36]. Це вимагає системного підходу до вирішення комплексу взаємопов'язаних проблем, включаючи глибокий аналіз архітектурних патернів, розробку та вдосконалення алгоритмічного забезпечення для обробки даних та інтелектуального керування, забезпечення інтеграції та інтероперабельності через гнучкі API, а також реалізацію комплексних заходів безпеки. Успішне вирішення цих завдань дозволить створити потужне програмне ядро, здатне повною мірою розкрити потенціал

технологій інтелектуального освітлення для сучасних міст. Основні функціональні та нефункціональні вимоги до серверної частини кіберфізичної системи «Розумні ліхтарі» узагальнено в таблиця 1.1 та 1.2.

Таблиця 1.1 – функціональні вимоги до серверної частини системи «Розумні ліхтарі»

Вимога	Опис
Збір даних у реальному часі	Збір даних з сенсорів, моніторинг стану та діагностика
Підтримка протоколів	MQTT, CoAP, LwM2M, інші IoT-протоколи
Автентифікація	Авторизація пристроїв і користувачів
База даних	Time-Series DB для історичних і поточних даних
Обробка даних	Фільтрація, валідація, первинна обробка
Команди керування	Формування та надсилання (групове/індивідуальне)
API	REST API для інтеграції з іншими системами
Веб-інтерфейс	Карта, звіти, історія подій
Сповіщення	Генерація сповіщень про критичні події

Таблиця 1.2 – нефункціональні вимоги до серверної частини системи «Розумні ліхтарі»

Вимога	Опис
Масштабованість	Підтримка великої кількості пристроїв
Продуктивність	Швидка обробка запитів
Надійність	Безперервна робота 24/7
Безпека	Захист даних та кіберзагроз
Сумісність	Відкриті стандарти для інтеграції
Гнучкість	Легка зміна сценаріїв та налаштувань

Висока доступність	Резервування, живлення, доступ до сервісу

1.3 Висновки до першого розділу

У першому розділі даної кваліфікаційної роботи було проведено комплексний, всебічний та глибокий аналіз предметної області, що пов'язана з розробкою, функціонуванням та інтеграцією серверної частини для кіберфізичної системи «Розумні ліхтарі». Виконане на цьому етапі дослідження дозволило сформувати міцне, багатогранне та методологічно обґрунтоване підґрунтя для всієї подальшої проєктної, програмної та експериментальної роботи. Цей розділ заклав концептуальний фундамент, визначив ключові проблеми та окреслив чіткі рамки для подальших досліджень та розробки.

В рамках підрозділу 1.1 було здійснено детальний та критичний аналіз сучасного стану систем інтелектуального освітлення. Було виявлено, систематизовано та детально описано ключові технологічні, економічні та експлуатаційні недоліки традиційних підходів до управління міським освітленням. Такі недоліки, як критично низька енергоефективність через неадаптивну логіку роботи, повна відсутність гнучкості та індивідуального керування, а також високі та непрозорі експлуатаційні витрати, пов'язані з реактивним обслуговуванням, однозначно обґрунтовують високу актуальність та економічну доцільність переходу до інтелектуальних рішень. На противагу застарілим системам, було детально розглянуто концепцію кіберфізичних систем як фундаментальну парадигму для побудови систем нового покоління. Було проаналізовано їхню узагальнену структуру, основні компоненти та принципи функціонування. Особливу увагу було приділено специфічній ролі кіберфізичної системи у побудові систем «Розумні ліхтарі», де було детально проаналізовано, як саме фізичні пристрої, такі як сенсори та актуатори, мережева інфраструктура та централізовані кібернетичні компоненти взаємодіють у безперервному циклі

зворотного зв'язку для забезпечення адаптивного, проактивного та ефективного керування освітлювальною інфраструктурою в масштабах цілого міста.

На основі проведеного глибокого аналізу, у підрозділі 1.2 було сформульовано вичерпний та структурований перелік детальних функціональних та нефункціональних вимог до розроблюваної серверної частини. Функціональні вимоги охопили всі ключові аспекти життєвого циклу даних та логіки керування в системі. Вони включають вимоги до надійного збору, валідації, обробки та довготривалого зберігання телеметричної інформації; вимоги до реалізації складних інтелектуальних алгоритмів для адаптивного керування та автоматичної діагностики; вимоги до забезпечення потужних інструментів моніторингу та візуалізації стану системи; а також вимоги до наявності стандартизованих програмних інтерфейсів (API) для забезпечення гнучкої взаємодії з іншими компонентами та системами. Нефункціональні вимоги, в свою чергу, визначили критично важливі якісні атрибути майбутньої системи, такі як її здатність до горизонтального масштабування для підтримки тисяч пристроїв, висока надійність та відмовостійкість, необхідна продуктивність для обробки даних у режимі, близькому до реального часу, комплексний підхід до забезпечення безпеки даних та системи в цілому, а також гнучкість та сумісність для легкої адаптації та інтеграції в існуючу міську інфраструктуру.

Завершальним та синтезуючим етапом роботи в рамках даного розділу стала чітка, обґрунтована та формалізована постановка задачі на дослідження та розробку серверної частини кіберфізичної системи «Розумні ліхтарі». Ця постановка задачі не є простою констатацією мети, а є результатом синтезу виявлених у ході аналізу проблем та сформульованих вимог. Вона перетворює теоретичні викладки та аналітичні дані у конкретний, вимірюваний та досяжний план дій, що є основою та дорожньою картою для всієї подальшої роботи над дипломним проєктом. Вона чітко окреслює межі дослідження, визначає ключові результати, які необхідно отримати, та задає вектор для вибору архітектурних рішень та технологічного стеку.

Таким чином, результати, отримані в першому розділі, а саме проведений глибокий аналіз предметної області, визначення ролі та структури кіберфізичної системи, формулювання деталізованих та вичерпних вимог, а також чітка постановка конкретних завдань, створюють необхідну та достатню теоретичну, аналітичну та методологічну базу. Ця база є не просто вступною частиною роботи, а міцним фундаментом, на якому будуватимуться всі наступні етапи кваліфікаційної роботи, включаючи безпосереднє проєктування архітектури, розробку ключових програмних компонентів, реалізацію алгоритмів та проведення експериментального тестування серверної частини системи «Розумні ліхтарі».

2 ПРОЄКТУВАННЯ СЕРВЕРНОЇ ЧАСТИНИ КІБЕРФІЗИЧНОЇ СИТЕМИ «РОЗУМНІ ЛІХТАРІ»

2.1 Опис архітектури кіберфізичної системи

У цьому підрозділі здійснюється фундаментальне обґрунтування вибору архітектурного підходу до побудови серверної частини кіберфізичної системи «Розумні ліхтарі», а також вибору відповідного технологічного стеку, що включає мову програмування, програмний фреймворк, систему управління базами даних та протоколи взаємодії. Прийняті на цьому етапі рішення є критично важливими, оскільки вони закладають основу для всієї подальшої розробки та безпосередньо впливають на такі ключові нефункціональні характеристики системи, як її надійність, продуктивність, гнучкість, можливість подальшого масштабування та легкість супроводу. Кожен вибір є результатом ретельного аналізу специфічних вимог проєкту, збалансованого з урахуванням сучасних інженерних практик та доцільності на етапі створення прототипу.

При проєктуванні серверної частини для складних розподілених систем, до яких беззаперечно належить кіберфізична система «Розумні ліхтарі», двома домінуючими архітектурними парадигмами є монолітна та мікросервісна архітектури. Монолітна архітектура передбачає реалізацію всього можливого функціоналу системи, включаючи логіку збору даних, їх обробку, алгоритми керування, взаємодію з базою даних та надання програмного інтерфейсу (API), в межах єдиного, цілісного застосунку. Такий підхід характеризується спрощеною комунікацією між компонентами, оскільки вона відбувається через прямі виклики функцій в межах одного процесу. На противагу, мікросервісний підхід передбачає декомпозицію системи на набір невеликих, незалежних та слабкозв'язаних сервісів. Кожен такий сервіс відповідає за виконання вузькоспеціалізованої бізнес-функції (наприклад, сервіс автентифікації, сервіс прийому телеметрії, сервіс керування пристроями) і може бути розроблений, розгорнутий та

масштабований незалежно від інших, взаємодіючи з ними через мережеві інтерфейси, такі як REST API або асинхронні черги повідомлень.

Зважаючи на те, що в межах даної кваліфікаційної роботи реалізується саме прототип серверної частини, а не повномасштабне промислове рішення, було прийнято обґрунтоване рішення використовувати монолітну архітектуру. Такий вибір є прагматичним та доцільним на поточному етапі, оскільки він значно спрощує процеси розробки та розгортання, а також полегшує налагодження та тестування системи завдяки єдиній кодовій базі та відсутності мережевих затримок і складнощів розподіленої взаємодії. Однак, щоб уникнути потенційних проблем, пов'язаних з ростом складності класичного моноліту, було вирішено реалізувати його з чітким внутрішнім поділом на функціональні модулі. Цей підхід, відомий як «модульний моноліт», дозволяє зберегти логічну структуру та чіткі межі між різними частинами системи, що в майбутньому, у разі потреби масштабування системи до рівня реального міста, значно спростить процес поступової та контрольованої трансформації окремих модулів у незалежні мікросервіси без необхідності кардинальної перебудови всієї бізнес-логіки.

Для практичної реалізації спроєктованої серверної частини було обрано мову програмування Python у поєднанні з мікрофреймворком Flask. Цей вибір обґрунтований низкою вагомих причин. По-перше, Python, завдяки своєму чистому, лаконічному та інтуїтивно зрозумілому синтаксису, значно знижує когнітивне навантаження на розробника та прискорює цикл розробки, що є надзвичайно важливим на етапі прототипування. По-друге, Python володіє величезною та зрілою екосистемою сторонніх бібліотек, що дозволяє вирішувати практично будь-які завдання без необхідності "винаходити велосипед". Зокрема, існують високоякісні бібліотеки для роботи з MQTT, для взаємодії з базами даних (SQLAlchemy), для обробки та аналізу даних (Pandas, NumPy), що є критично важливим для функціоналу нашої системи. Фреймворк Flask, у свою чергу, є ідеальним інструментом для швидкого створення веб-серверів та RESTful API. Його мінімалістичне ядро не нав'язує розробнику жорстких структурних

обмежень, надаючи повну свободу у виборі інструментів та архітектурних рішень, а його розширюваність дозволяє легко інтегрувати будь-які необхідні компоненти.

Вибір системи управління базами даних є одним із найважливіших рішень, оскільки дані є серцем будь-якої інформаційної системи. Для зберігання даних було вирішено використати PostgreSQL як основну реляційну СУБД. Цей вибір зумовлений високою надійністю, стабільністю, відповідністю стандартам ACID та потужними можливостями PostgreSQL, що включають підтримку складних SQL-запитів, транзакцій, а також гнучку систему розширень. PostgreSQL ідеально підходить для зберігання структурованої інформації, такої як дані про ліхтарі, їхні конфігурації, інформація про користувачів та їхні ролі доступу. Однак, ключовою особливістю даних у системі «Розумні ліхтарі» є наявність великих обсягів часових рядів (time-series data) – показників від датчиків, що надходять безперервним потоком. Для ефективної роботи з такими даними було прийнято рішення використовувати TimescaleDB – розширення з відкритим вихідним кодом, яке перетворює PostgreSQL на повноцінну та високопродуктивну базу даних часових рядів. TimescaleDB використовує механізм гіпертаблиць, автоматично розподіляючи великі таблиці з часовими даними на менші, керовані частини за часовим та, за потреби, іншими атрибутами. Такий підхід кардинально підвищує продуктивність операцій вставки нових даних та ефективність виконання запитів до великих обсягів історичних даних, що є типовими для IoT-систем. Таким чином, поєднання PostgreSQL з розширенням TimescaleDB дозволяє створити єдине, потужне та гнучке сховище даних, яке поєднує переваги реляційної моделі для структурованих даних з високою оптимізацією для часових рядів сенсорних показників, задовольняючи всі потреби системи.

Для забезпечення надійного та ефективного обміну даними між тисячами потенційних ліхтарів та центральним сервером, було обрано протокол MQTT (Message Queuing Telemetry Transport). Він був спеціально розроблений для Інтернету речей та ідеально підходить для пристроїв з обмеженими обчислювальними ресурсами та для роботи в мережах з низькою пропускною

здатністю та можливими перебоями зв'язку. Його легковаговість, що проявляється в мінімальному розмірі заголовків повідомлень, модель «видавець-підписник», яка забезпечує слабку зв'язаність та гнучкість системи, а також підтримка трьох рівнів якості обслуговування (QoS), що дозволяє гарантувати доставку критичних команд керування, роблять його безальтернативним вибором для даного проєкту. Функція «Last Will and Testament» додатково дозволяє серверу миттєво реагувати на несподіване відключення пристроїв, підвищуючи надійність моніторингу.

Для організації взаємодії між серверною частиною та клієнтськими застосунками, такими як веб-інтерфейс адміністратора, було обрано архітектурний стиль REST з використанням формату обміну даними JSON. REST є де-факто стандартом для побудови сучасних веб-сервісів, а його використання на основі стандартних HTTP-методів (GET, POST, PUT, DELETE) забезпечує чітку, зрозумілу та передбачувану модель взаємодії. Формат JSON, у свою чергу, є легкою, текстовою та легко інтерпретованою як людиною, так і машиною структурою, що гарантує високий рівень сумісності між серверною та клієнтською частинами, незалежно від технологій їх реалізації. Такий підхід сприяє чіткому архітектурному розділенню відповідальностей, де сервер концентрується на бізнес-логіці, а клієнт – на її представленні, та створює умови для легкої масштабованості та інтеграції з іншими інформаційними системами.

Нарешті, при виборі платформи для розгортання серверної частини розглядалися два основні підходи: локальне розгортання та використання хмарних платформ. Локальне розгортання на власних серверах надає повний контроль над інфраструктурою, проте вимагає значних капітальних витрат та наявності кваліфікованого персоналу. Хмарні платформи (AWS, Azure, GCP) пропонують високу гнучкість, масштабованість та модель оплати за фактичне використання, що є надзвичайно привабливим. Для даної кваліфікаційної роботи, на етапі розробки та тестування прототипу, передбачається використання локального середовища. Однак, спроєктована архітектура та вибір технологій,

зокрема використання контейнеризації за допомогою Docker, закладають можливість легкого та безшовного перенесення системи як на локальні сервери замовника, так і в будь-яку хмарну інфраструктуру в майбутньому.

Обраний технологічний та архітектурний підхід дозволяють сформувати наступну таку узагальнену архітектуру серверної частини кіберфізичної системи «Розумні ліхтарі», яка представлена на рис. 2.1

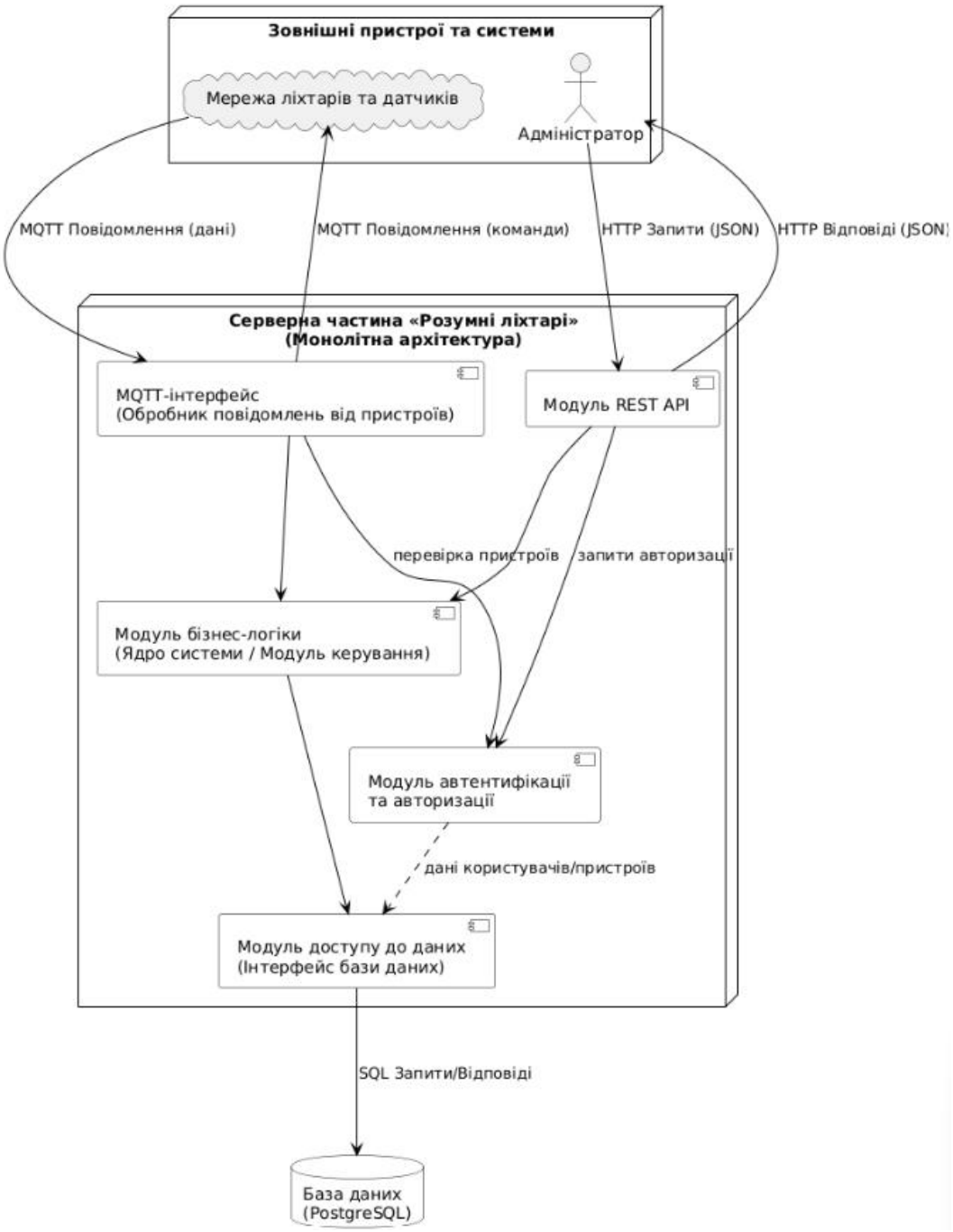


Рисунок 2.1 − запропонована архітектура серверної частини кіберфізичної системи «Розумні ліхтарі»

На рис. 2.1 зображена серверна частина, яка реалізована за монолітною архітектурою з чітким логічним поділом на функціональні модулі, що забезпечує як простоту розробки прототипу, так і можливість подальшої структуризації. Основними компонентами архітектури є зовнішні пристрої та системи, до яких належать:

1. Мережа ліхтарів та датчиків, що взаємодіє з серверною частиною переважно через протокол MQTT, а також адміністратор (або інші користувачі системи), який отримує доступ до функціоналу через клієнтські застосунки.

2. Серверна частина «Розумні ліхтарі» (монолітна архітектура): це ядро системи , що містить наступні ключові модулі, такі як:

1) MQTT-інтерфейс (обробник повідомлень від пристроїв): відповідає за прийом телеметричних даних (стан ліхтарів, показники датчиків) від кінцевих пристроїв через MQTT-брокер, а також за відправку команд керування на пристрої.

2) Модуль REST API: надає зовнішній програмний інтерфейс на основі архітектурного стилю REST для взаємодії з клієнтськими застосунками (наприклад, інтерфейсом адміністратора). Обробляє HTTP-запити та формує відповіді у форматі JSON.

3) Модуль автентифікації та авторизації: забезпечує перевірку прав доступу для користувачів, що взаємодіють через REST API, та автентифікацію пристроїв, що підключаються через MQTT; модуль бізнес-логіки (ядро системи / модуль керування).

4) Центральний компонент, в ньому реалізовані основні алгоритми функціонування системи: обробка та аналіз даних, що надходять від пристроїв; прийняття рішень щодо режимів освітлення відповідно до заданих сценаріїв, розкладів та показників датчиків; формування команд для виконання пристроїв.

5) Модуль доступу до даних (Інтерфейс бази даних): інкапсулює всю логіку взаємодії з базою даних, виконуючи операції читання, запису, оновлення та видалення інформації.

6) База даних (PostgreSQL): центральне сховище для всієї системної інформації, включаючи дані про конфігурацію ліхтарів та датчиків, історичні показники сенсорів, журнали подій, інформацію про користувачів та налаштування системи. Передбачається використання можливостей PostgreSQL для ефективної роботи з різними типами даних, включаючи часові ряди (потенційно з використанням розширення TimescaleDB).

Взаємодія між модулями всередині монолітного застосунку здійнсюється через прямі виклики функцій та методів, що спрощує розробку на етапі прототипування. Така архітектура забезпечує необхідний функціонал для реалізації поставлених завдань та є основою для подальшого проєктування окремих компонентів системи.

2.2 Розробка структури бази даних та програмного інтерфейсу системи

Ефективне зберігання та управління даними є важливими для функціонування серверної частини системи «Розумні ліхтарі». Як було обґрунтовано в підрозділі 2.1, для цих цілей обрано реляційну систему управління базами даних PostgreSQL. У даному пункті буде детально описано спроєктовану структуру бази даних, включаючи основні таблиці, їх атрибути та зв’язки між ними, що необхідні для зберігання інформації про ліхтарі, показники датчиків, користувачів та інші системні дані. Також буде розглянуто проєктування програмного інтерфейсу, який забезпечить взаємодію серверної частини з клієнтськими застосунками.

Для забезпечення функціональності було визначено наступні ключові сутності, для кожної з яких розроблено відповідну таблицю в базі даних PostgreSQL: Ліхтарі, як streetlights, показання датчиків, як sensor_readings,

користувачі, як users. Взаємозв'язки між таблицями забезпечують цілісність даних, зокрема, за рахунок зовнішніх ключів. Таким чином, структура дозволяє ефективно реалізувати логіку збирання, обробки та аналізу даних.

Таблиця 2.1 – таблиця сутностей бази даних ліхтарів

Поле	Тип Даних	Обмеження	Опис
Id	Serial	PK	Унікальний ідентифікатор ліхтаря
Device_eui	VARCHAR(255)	UNIQUE, NOT NULL	Унікальний апаратний ідентифікатор пристрою
name	VARCHAR(255)	NOT NULL	Назва або опис ліхтаря
latitude	DOUBLE PRECISION	NOT NULL	Географічна широта
longitude	DOUBLE PRECISION	NOT NULL, DEFAULT (active)	Географічна довгота
status	VARCHAR(50)	DEFAULT 0, CHECK (0-100)	Поточний стан Ліхтаря
Current_brightness	INTEGER	DEFAULT CURRENT_TIMESTAMP	Рівень яскравості в %
Installed_at	TIMESTAMP		Дата

	WITH		встановлення
Last_communication_at	TIME ZONE		Останній сеанс зв’язку

Було також зроблено таблицю показань датчиків, яка призначена для зберігання часових даних від датчиків, асоційовних з ліхтарями.

Таблиця 2.2 – таблиця сутностей бази даних показань датчиків

Поле	Тип даних	Обмеження	Опис
id	BIGSERIAL	PK	Унікальний ідентифікатор запису
Streetlight_id	INTEGER	NOT FULL, FK	Посилання на streetlights (id)
Sensor_type	VARCHAR(50)	NOT NULL	Тип датчика (рух, освітленість тощо)
Value_numeric	DOUBLE PRECISION		Числове значення
Value_text	VARCHAR (255)		Текстове значення
timestamp	TIMESTAMP WITH TIME ZONE	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Час отримання показника

Таблиця бази даних користувачів призначена для зберігання інформації про користувачів системи, які мають доступ до інтерфейсу адміністратора.

Таблиця 2.3 – таблиця сутностей бази даних користувачів

Поле	Тип даних	Обмеження	Опис
id	SERIAL	PK	Ідентифікатор користувача
username	VARCHAR(100)	UNIQUE,NOT NULL	Логін користувача
Password_hash	VARCHAR(255)	NOT NULL	Хеш пароля
email	VARCHAR(255)	UNIQUE	Електронна пошта
Full_name	VARCHAR(255)		Повне ім’я
role	VARCHAR(50)	NOT NULL, DEFAULT (operator)	Роль в системі
Created_at	TIMESTAMP WITH TIME ZONE	DEFAULT CURRENT_TIMESTAMP	Дата створення облікового запису
Is_active	BOOLEAN	DEFAULT TRUE	Активність облікового запису

На основі визначених сутностей та їх атрибутів була розроблена логічна схема бази даних, яка візуалізує структуру таблиць та зв’язки між ними. Ця схема представлена на рисунку 2.2. Вона відображає ключові таблиці, необхідні для зберігання інформації про ліхтарі, показання датчиків та користувачів системи. Взаємозв’язки між таблицями забезпечують цілісність даних, зокрема,

за рахунок зовнішніх ключів. Така структура дозволяє ефективно реалізувати логіку збирання, обробки та аналізу даних у системі «Розумні ліхтарі».

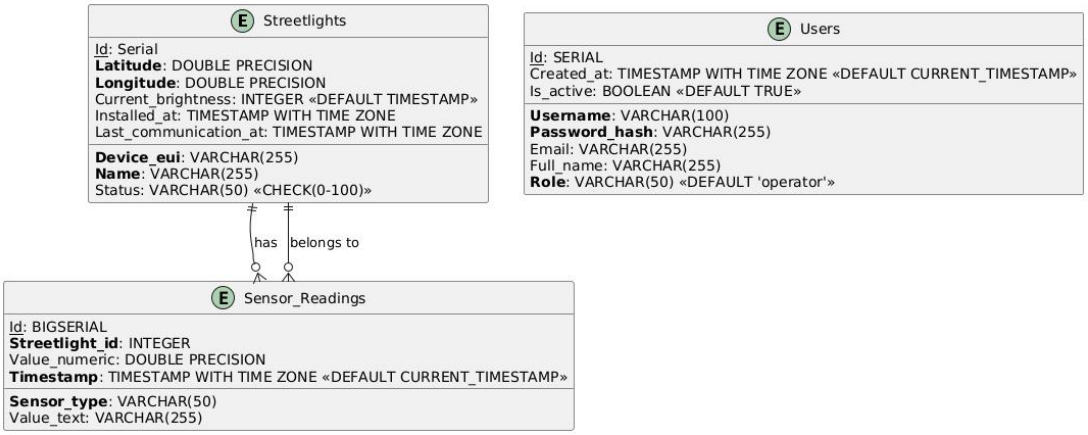


Рисунок 2.2 – Логічна схема бази даних системи «Розумні Ліхтарі»

На рисункуу показано основні таблиці бази даних та зв’язки між ними. Таблиця streetlights є центральною і зберігає інформацію про кожен освітлюваний прилад. З нею пов’язана таблиця Sensor_Readings (зв’язок один-до-багатьох), через зовнішній ключ streetlights_id у таблиці sensor_readings, що дозволяє зберігати історію показників для кожного ліхтаря. Таблиця users призначена для зберігання облікових даних користувачіів системи та управління їх доступом.

Така реляційна структура з трьох основних таблиць забезпечує необхідний рівень нормалізації даних для уникнення їх дублювання,підтримку цілісності інформації та можливість виконання ефективних запитів для аналізу стану системи освітлення та управління нею на етапі прототипування.

2.3 Проєктування програмного інтерфейсу системи

Для забезпечення взаємодії між серверною частиною системи «Розумні ліхтарі» та клієнтськими застосунками (наприклад, вебінтерфейсом адміністратора), а також для потенційної інтерграції з іншими міськими системами розроблено програмний інтерфейс. Як було обґрунтовано в підрозділі 2.1, для цього обрано архітектурний стиль REST з обміном даними у форматі

JSON. Основними ресурсами, якими оперує API, є ліхтарі, дані їх сенсорів та користувачі. Для кожного ресурсу визначено набір стандартних операцій. Передбачається використання токен-базованої автентифікації (наприклад, JWT) для захисту доступу до API, що забезпечує передачу токена в заголовках запитів після успішного входу користувача в систему.

Ключові ендпоінти API, спроєктовані для управління системою «Розумні ліхтарі», включають наступні групи:

Автентифікація користувачів: POST /api/auth/login: Призначений для автентифікації користувачів системи. При успішній автентифікації (перевірці логіна та пароля) сервер генерує та повертає клієнту токен доступу (наприклад, JWT), який в подальшому використовується для авторизації запитів до захищених ресурсів API;

Управління ліхтарями (базовий шлях /api/lights):

GET /api/lights: возволяє отримати список всіх зареєстрованих ліхтарів у системі. Відповідь сервера містить масив об'єктів, кожен з яких представляє ліхтар з його основними атрибутами. Передбачена можливість пагінації для роботи з великою кількістю пристроїв та фільтрації (наприклад, за статусом або назвою).

POST /api/lights: використовується для реєстрації нового ліхтаря в системі. Тіло запиту має містити необхідну інформацію про ліхтар, таку як device_eui, name, географічні координати (latitude, longitude). При успішному створенні сервер повертає дані створеного об’єкта та статус 201(Created).

GET /api/lights/{light_id}: надає детальну інформацію про конкретний ліхтар за його унікальним ідентифікатором (light_id).

PUT /api/lights/{light_id}: дозволяє оновити інформацію про існуючий ліхтар (наприклад, змінити його назву, опис, статус). Дані для оновлення передаються в тілі запиту.

DELETE /api/lights/{light_id}: призначений для видалення ліхтаря з системи за його ідентифікатором.

POST /api/lights/{light_id}/control: забезпечує можливість надсилання команд керування конкретним ліхтарем. Наприклад, через тіло запиту можна передати новий бажаний рівень яскравості або команду на увімкнення/вимкнення. Робота з даними датчиків (базовий шлях /api/lights/{light_id}/readings):

GET /api/lights/{light_id}/readings: дозволяє отримати історію показників датчиків для вказаного ліхтаря. API може підтримувати параметри запиту для фільтрації даних за типом датчика та часовим проміжком (наприклад, sensor_type=motion&start_date=…&end_date=…).

Керування користувачами (базовий шлях: /api/users - доступно для користувачів з роллю адміністратора):

GET /api/users: отримання списку всіх користувачів системи;

POST /api/users: створення нового облікового запису користувача (з вказанням логіна, пароля, ролі);

PUT /api/users/{user_id}: оновлення даних існуючого користувача;

DELETE /api/users/{user_id}: видалення облікового запису користувача.

Таблиця 2.4 – основні ендпоінти API серверної частини кіберфізичної системи «Розумні ліхтарі»

HTTP	URL-шлях (Ендпоінт)	Призначення/Опис
POST	/api/auth/login	Автентифікація користувача,отримання токена доступу
GET	/api/lights	Отримати список всіх ліхтарів
POST	/api/lights	Додати новий ліхтар
GET	/api/lights/{id}	Отримати інформацію про ліхтар за ID
PUT	/api/lights/{id}	Оновити інформацію про ліхтар за ID

Кінець таблиці 2.4.

DELETE	/api/lights/{id}	Видалити ліхтар за ID
POST	/api/lights/{id}/control	Надіслати команду керування ліхтарем.
GET	/api/lights/{id}/readings	Отримати показники датчиків ліхтаря за ID
GET	/api/users	Отримати список користувачів (доступно адміністраторам)
POST	/api/users	Створення нового облікового запису користувача

Представлений набір ендпоінтів забезпечує необхідний базовий функціонал для моніторингу стану, управління освітлювальними приладами та адміністрування користувачів системи «Розумні ліхтарі» через клієнтські застосунки. Спроєктований API є гнучкими та може бути розширений у майбутньому для підтримки нового функціоналу або інтеграції з іншими міськими інформаційними сервісами.

2.3 Висновки до другого розділу

У другому розділі даної кваліфікаційної роботи було виконано фундаментальний етап проєктування серверної частини кіберфізичної системи «Розумні ліхтарі», який є логічним продовженням проведеного в першому розділі аналізу предметної області та сформульованих вимог. На цьому етапі було розроблено та всебічно обґрунтовано ключові архітектурні та технологічні рішення, а також детально спроєктовано структуру бази даних та програмний

інтерфейс (API) системи, що в сукупності створює міцний технічний фундамент для подальшої програмної реалізації.

У підрозділі 2.1 було проведено глибоке обґрунтування вибору архітектурного підходу та технологічного стеку. Після ретельного аналізу переваг та недоліків монолітної та мікросервісної архітектур, було прийнято зважене рішення обрати для реалізації прототипу системи монолітну архітектуру. Такий вибір є доцільним на поточному етапі, оскільки він значно спрощує процеси розробки, тестування та розгортання, що є критичним в умовах обмежених ресурсів дипломного проєкту.

Водночас було наголошено на необхідності застосування принципів модульності та чіткого внутрішнього поділу на функціональні компоненти. Такий підхід, відомий як «модульний моноліт», не лише підвищує якість коду та його підтримуваність, але й створює стратегічну основу для потенційної та менш болісної трансформації системи в повноцінну мікросервісну архітектуру в майбутньому, у міру її масштабування та ускладнення.

В якості основного технологічного стеку для реалізації серверної частини було обрано мову програмування Python у поєднанні з мікрофреймворком Flask. Цей вибір обґрунтований високою продуктивністю Python у задачах обробки даних, його багатим екосистемом бібліотек для роботи з мережевими протоколами та базами даних, а також гнучкістю та мінімалістичністю Flask, що ідеально підходить для створення RESTful API. Для зберігання даних було обрано систему управління базами даних PostgreSQL як надійне, потужне та гнучке рішення для реляційних даних (інформація про ліхтарі, користувачів, конфігурації).

Окремо було підкреслено доцільність потенційного використання розширення TimescaleDB, яке перетворює PostgreSQL на повноцінну time-series базу даних, що є оптимальним для ефективного зберігання та обробки великих обсягів телеметричних даних від датчиків.

Для забезпечення комунікації з кінцевими пристроями було обрано протокол MQTT як де-факто стандарт для Інтернету речей, що гарантує надійну та легкомасштабовану доставку повідомлень. Взаємодія з клієнтськими застосунками (веб-інтерфейс адміністратора) буде реалізована через REST API з використанням універсального формату обміну даними JSON. Запропоновану комплексну архітектуру серверної частини, що візуалізує взаємодію всіх цих компонентів, було детально представлено на відповідній схемі (рис. 2.1).

У підрозділі 2.2 було зосереджено увагу на детальному проєктуванні двох ключових компонентів системи: бази даних та програмного інтерфейсу. Було розроблено логічну структуру бази даних PostgreSQL, що включає три основні таблиці.

Таблиця streetlights призначена для зберігання статичної та конфігураційної інформації про кожен ліхтар як фізичний об'єкт: його унікальний ідентифікатор, географічні координати, модель, дата встановлення тощо.

Таблиця sensor_readings спроєктована для зберігання динамічних часових рядів даних, що надходять від датчиків, таких як поточний статус, рівень яскравості, показники енергоспоживання та події детектування руху, з обов'язковою прив'язкою до часу та ідентифікатора ліхтаря.

Таблиця users відповідає за зберігання інформації про користувачів системи, їхні облікові дані та ролі доступу, що є основою для майбутньої реалізації механізмів безпеки. Логічну схему розробленої бази даних, що наочно відображає всі атрибути таблиць, типи даних та зв’язки між ними, було представлено у вигляді ER-діаграми (рис. 2.2).

Також в рамках цього підрозділу було ретельно спроєктовано програмний інтерфейс (API) серверної частини на основі архітектурного стилю REST. Було визначено ключові ресурси системи та розроблено набір ендпоінтів для забезпечення всього необхідного функціоналу.

Цей набір включає операції для повного життєвого циклу управління ліхтарями (створення, читання, оновлення, видалення), ендпоінти для отримання

історичних та поточних телеметричних даних для візуалізації на графіках та картах, а також маршрути для адміністрування користувачів системи. Основні ендпоінти, їхні HTTP-методи, очікувані параметри та призначення було детально описано та узагальнено у відповідній таблиці (табл. 2.4), що слугуватиме специфікацією для подальшої розробки.

Таким чином, у другому розділі було створено комплексну, детальну та обґрунтовану проєктну основу для серверної частини кіберфізичної системи «Розумні ліхтарі».

Розроблені архітектурні рішення, обраний технологічний стек, спроєктована структура бази даних та специфікація програмного інтерфейсу формують цілісний та завершений технічний проєкт. Цей проєкт є необхідним і достатнім підґрунтям для переходу до наступного, практичного етапу кваліфікаційної роботи − безпосередньої програмної реалізації, імплементації алгоритмів та всебічного тестування спроєктованої системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СЕРВЕРНОЇ ЧАСТИНИ КІБЕРФІЗИЧНОЇ СИСТЕМИ «РОЗУМНІ ЛІХТАРІ»

3.1 Реалізація архітектури серверної частини кіберфізичної системи

Основоположним компонентом, що визначає не лише весь спектр функціональних можливостей, але й фундаментальні якісні атрибути, такі як надійність, масштабованість, безпека та інтелектуальність спроєктованої кіберфізичної системи «Розумні ліхтарі», є її програмно-серверна частина. В рамках даної кіберфізичної системи сервер виконує роль центрального вузла управління, обробки та аналізу, виступаючи в якості обчислювального та логічного ядра, яке інтегрує тисячі потенційно географічно розподілених фізичних пристроїв в єдину, злагоджено функціонуючу, адаптивну та керовану екосистему. На нього покладено виконання цілого комплексу ключових функціональних обов'язків. Перш за все, це безперервний прийом та обробка даних, що включає отримання, синтаксичну та семантичну валідацію, очищення від шумів та артефактів, а також первинну обробку та нормалізацію безперервних потоків телеметричної інформації. Ця інформація, що надходить від різноманітних сенсорів та мікроконтролерів, які є невід'ємною частиною мережі освітлення, формує основу для прийняття всіх подальших рішень.

Не менш важливою є функція зберігання інформації, яка забезпечує довготривале, структуроване, належним чином індексоване та безпечне збереження як історичних даних телеметрії, так і статичних конфігураційних даних у реляційній базі даних. Це створює надійний фундамент для подальшого глибокого аналізу, побудови аналітичних панелей, візуалізації трендів, проведення аудиту системних подій та генерації звітності для адміністративного персоналу. Критично важливою для всієї концепції «розумного» освітлення є реалізація складної бізнес-логіки. Цей компонент відповідає за виконання складних математичних та евристичних алгоритмів інтелектуального керування, які лежать в основі адаптивної роботи системи. Саме ця логіка дозволяє системі

динамічно реагувати на мінливі умови в міському середовищі, оптимізуючи енергоспоживання та підвищуючи рівень безпеки. Нарешті, сервер надає чітко документовані та стандартизовані програмні інтерфейси доступу, які слугують універсальним шлюзом для безпечної та контрольованої взаємодії з різноманітними клієнтськими застосунками, такими як адміністративні веб-панелі та мобільні додатки для технічного персоналу. Більше того, наявність відкритого API відкриває широкі можливості для потенційної інтеграції з іншими міськими інформаційними системами в рамках більш глобальної концепції «Розумного міста».

Фундаментом для спроєктованої архітектури слугує класична та перевірена часом триярусна модель взаємодії, що базується на парадигмі «клієнт–сервер». Ключова перевага цієї моделі полягає в ефективній сегрегації (розділенні) функціональних обов'язків між рівнем представлення, рівнем бізнес-логіки та рівнем дани. Клієнтські додатки, такі як веб-інтерфейси чи мобільні додатки, повністю відповідають за ініціалізацію запитів до сервера та за візуалізацію отриманих даних для кінцевого користувача, не маючи прямого доступу до бази даних чи складної логіки. Сервер, у свою чергу, інкапсулює всю складну логіку обробки цих запитів, керування загальним станом системи, взаємодію з базою даних та, що найважливіше, забезпечення безпеки та розмежування прав доступу. Такий чіткий розподіл відповідальності дозволяє незалежно розробляти, тестувати, масштабувати та оновлювати клієнтську та серверну частини, залучаючи спеціалізованих розробників для кожної з них, що значно підвищує якість та швидкість розробки.

Для програмної реалізації серверної логіки було обрано полегшений програмний каркас, відомий як мікрофреймворк, Flask на мові програмування Python. На відміну від більш монолітних та комплексних рішень, таких як Django, які пропонують готовий та жорстко інтегрований набір інструментів «з коробки» (ORM, панель адміністратора, система автентифікації), Flask свідомо дотримується філософії мінімалізму та розширюваності. Він пропонує лише

базовий, але надзвичайно потужний та гнучкий інструментарій для маршрутизації HTTP-запитів та обробки шаблонів, надаючи розробнику максимальну свободу у виборі та інтеграції сторонніх бібліотек для вирішення конкретних завдань. Такий підхід виявився оптимальним для даного проєкту, оскільки він дозволив уникнути надлишкового функціонального навантаження та створити високопродуктивне та спеціалізоване рішення, точно адаптоване під потреби системи. Важливою перевагою Flask є його виняткова зручність у побудові RESTful API, що використовує стандартні методи протоколу HTTP (GET, POST, PUT, DELETE) для маніпуляції системними ресурсами. Це забезпечує універсальність, стандартизацію та передбачуваність комунікації з будь-якими клієнтськими застосунками, незалежно від платформи їх реалізації.

В якості системи управління базами даних було обрано PostgreSQL. Це рішення обґрунтоване бездоганною репутацією PostgreSQL як однієї з найнадійніших, найпотужніших та найбільш функціонально багатих реляційних СУБД з відкритим кодом. Вона забезпечує строгу відповідність стандартам ACID (атомарність, узгодженість, ізольованість, довговічність), що є абсолютно критичним для гарантування цілісності та консистентності даних у системі, де кожна транзакція має бути виконана коректно. Розширені можливості PostgreSQL для вертикального та горизонтального масштабування, підтримка складних аналітичних запитів з використанням віконних функцій та Common Table Expressions, а також наявність спеціалізованих типів даних, таких як JSONB для ефективного індексованого зберігання напівструктурованих даних від сенсорів, та геопросторових типів даних через розширення PostGIS для роботи з координатами ліхтарів, роблять її чудовим вибором для систем, що оперують значними та різнорідними обсягами даних.

Взаємодія між програмним кодом додатку на Python та СУБД PostgreSQL здійснюється не напряму, а через потужну бібліотеку SQLAlchemy, яка реалізує шаблон проєктування Object-Relational Mapping (ORM). Застосування ORM дозволяє повністю абстрагуватися від прямої роботи з мовою SQL та

низькорівневих драйверів бази даних. Замість написання складних та потенційно вразливих до помилок SQL-конструкцій, розробник оперує даними на високому рівні абстракції, використовуючи звичні об'єкти мови Python. Це не тільки суттєво підвищує читабельність, підтримуваність та тестовність коду, але й значно прискорює процес розробки та, що надзвичайно важливо, мінімізує ризики, пов'язані з атаками типу SQL-ін'єкцій, оскільки SQLAlchemy автоматично екранує всі вхідні дані, що передаються до бази даних.

Для організації комунікації з кінцевими пристроями, тобто безпосередньо з «розумними ліхтарями», було передбачено глибоку інтеграцію з протоколом MQTT. Даний протокол, що функціонує за асинхронною моделлю «видавець-підписник» через центрального посередника, є загальновизнаним промисловим стандартом для рішень в галузі Інтернету речей. Його ключові характеристики, такі як мінімальне споживання мережевого трафіку завдяки компактним бінарним заголовкам, надзвичайна стійкість до нестабільних каналів зв'язку завдяки механізмам QoS та здатність до передачі даних в режимі реального часу, роблять його ідеальним та практично безальтернативним рішенням для отримання телеметрії та відправки команд керування на тисячі пристроїв одночасно.

Розроблена серверна частина має виражену модульну архітектуру, побудовану на фундаментальному принципі розмежування повноважень (Separation of Concerns). Такий підхід, що є наріжним каменем сучасної інженерії програмного забезпечення, передбачає, що кожен компонент системи має одну, чітко визначену та ізольовану відповідальність. Це сприяє не лише гнучкості та підтримуваності, але й можливості повторного використання коду в інших проєктах. Як продемонстровано на рисунку 3.1, проєкт декомпозовано на кілька логічних та функціональних модулів, що взаємодіють між собою через чітко визначені інтерфейси.

Центральним елементом та точкою входу в додаток є ініціалізаційний модуль (__init__.py), який реалізує шаблон проєктування. Його головне завдання − це динамічне створення та конфігурування екземпляру додатку Flask. Саме тут

відбувається ініціалізація всіх необхідних розширень, зокрема SQLAlchemy для роботи з базою даних та Flask-Migrate для управління міграціями схеми, завантаження конфігураційних параметрів з модуля config.py та реєстрація модулів маршрутизації, які містять логіку обробки API-запитів.

Модуль конфігурації (config.py) виконує роль централізованого та ізольованого сховища для всіх конфігураційних даних системи, які можуть змінюватися залежно від середовища розгортання (розробка, тестування, продакшн). У ньому містяться такі критично важливі параметри, як рядок підключення до бази даних PostgreSQL, налаштування для підключення до MQTT-брокера, секретні ключі для шифрування сесій та інші змінні. Винесення конфігурації в окремий файл дозволяє легко та безпечно змінювати налаштування системи без необхідності втручання в основну програмну логіку.

З метою покращення структури коду, його логічної організації та можливості перевикористання, логіка обробки запитів до API інкапсульована в самостійних модулях за допомогою механізму Flask Blueprints. Наприклад, файл lights.py містить всю логіку, що стосується ресурсу «ліхтар». Він обробляє HTTP-запити GET, POST, PUT та DELETE, що надходять на відповідні маршрути, валідує вхідні дані та організовує взаємодію з базою даних через моделі для виконання відповідних операцій.

Модуль моделей даних (models.py) є декларативним описом структури бази даних у вигляді Python-класів, використовуючи синтаксис SQLAlchemy. Цей модуль слугує єдиним джерелом правди про схему даних для всього додатку, забезпечуючи консистентність та унеможливлюючи розбіжності.

Хоча в поточній версії прототипу компонент MQTT-клієнта може бути реалізований у вигляді набору функцій, у повноцінній системі він функціонуватиме як окремий фоновий процес або сервіс, щоб не блокувати основний потік веб-сервера. Його відповідальність полягає у встановленні постійного з'єднання з MQTT-брокером, підписці на відповідні топіки для отримання даних від пристроїв та їх подальшій передачі на обробку в основний

додаток, наприклад, через внутрішню чергу повідомлень. Такий глибокий та продуманий підхід до декомпозиції та вибору технологій створює надійну, гнучку та масштабовану основу для майбутнього розвитку та вдосконалення системи.

3.2 Процес налаштування середовища та розгортання серверної частини

Етап розгортання та налаштування робочого середовища є критично важливим та фундаментальним процесом у життєвому циклі розробки будь-якого програмного продукту, оскільки саме він закладає основу для подальшої стабільної, безпечної та ефективної роботи. Для серверної частини проєкту «Розумні ліхтарі» цей процес було реалізовано з неухильним дотриманням сучасних принципів та найкращих практик інженерії програмного забезпечення, зокрема, ключової концепції reproducible development (відтворювана розробка). Основна мета, що переслідувалася на даному етапі, полягала у створенні такого програмного оточення, яке можна було б легко, швидко, автоматизовано та, що найголовніше, з гарантованою ідентичністю відтворити на будь-якій іншій машині розробника, тестовому стенді чи робочому сервері з мінімальними ручними зусиллями.

Такий підхід є абсолютною запорукою ефективної командної роботи, унеможливлюючи виникнення поширеної проблеми «у мене на машині все працює», та забезпечує безперебійний, контрольований перехід від етапу розробки до етапів тестування, інтеграції та подальшої експлуатації. Нижче детально описано послідовність виконаних етапів, що в сукупності формують цілісний та завершений процес підготовки системи до функціонування.

На початковому, архітектурному етапі було закладено логічну та фізичну структуру каталогів та файлів проєкту. Ця, на перший погляд, формальна дія має глибоке методологічне значення. Організація кодової бази була виконана у суворій відповідності до загальноприйнятих конвенцій та найкращих практик, рекомендованих для додатків, розроблених на фреймворку Flask. Такий

структурований підхід забезпечує не лише високу читабельність та інтуїтивну зрозумілість коду, але й значно спрощує навігацію по проєкту для нових розробників, скорочуючи час їхнього входження в проєкт.

Крім того, він підтримує фундаментальний принцип розмежування відповідальності (Separation of Concerns), коли кожен файл та каталог має чітко визначене призначення. Основна директорія проєкту smart_lights/ інкапсулює абсолютно всі компоненти системи, слугуючи єдиною точкою входу. Ключова логіка додатку логічно зосереджена в підпапці app/, що містить такі невід'ємні елементи, як ініціалізаційний файл __init__.py, який виконує роль "фабрики додатку" і відповідає за створення та конфігурування екземпляра Flask; модуль models.py, що містить декларативний опис ORM-моделей, які відображають структуру бази даних; централізований файл конфігурації config.py; та директорію routes/ з окремими модулями для кожного набору API-маршрутів, що дозволяє уникнути хаосу в одному великому файлі.

Окрім цього, у кореневій директорії проєкту знаходяться такі важливі файли, як requirements.txt, що декларує всі зовнішні залежності проєкту, та run.py, який слугує зручною точкою входу для запуску застосунку в режимі розробки з увімкненим відлагоджувачем.

Для забезпечення повної та надійної ізоляції проєкту від глобального системного середовища Python та для уникнення потенційних, часто важко діагностованих, конфліктів версій бібліотек, було створено віртуальне середовище за допомогою вбудованого модуля venv. Ця технологія є абсолютним та незаперечним стандартом у сучасній Python-розробці та дозволяє кожному окремому проєкту мати власний, незалежний набір залежностей, не впливаючи на інші проєкти на тій самій машині.

Після успішної активації віртуального середовища було виконано встановлення всіх необхідних програмних пакетів, точний перелік та версії яких були заздалегідь визначені та зафіксовані у файлі requirements.txt. Цей файл виступає маніфестом залежностей проєкту, дозволяючи повністю автоматизувати

процес налаштування робочого середовища за допомогою єдиної консольної команди pip install -r requirements.txt. Такий підхід гарантує, що кожен розробник у команді та кожне середовище розгортання (тестове, робоче) буде використовувати абсолютно ідентичні версії бібліотек, що повністю унеможливлює виникнення помилок, пов'язаних з їхньою несумісністю, та забезпечує детермінованість поведінки програми.

Одним із ключових аспектів побудови безпечної, гнучкої та легко підтримуваної архітектури є винесення всіх конфігураційних даних за межі основного коду програми. У даному проєкті цю критично важливу функцію виконує модуль config.py. Він містить усі чутливі та змінні параметри, зокрема рядок підключення до бази даних SQLALCHEMY_DATABASE_URI, який інкапсулює в собі ім'я користувача, пароль, мережеву адресу хоста, порт та назву бази даних, а також потенційні налаштування для підключення до MQTT-брокера.

Такий підхід дозволяє легко та безболісно змінювати параметри підключення, наприклад, при перенесенні проєкту з локального середовища розробки на робочий сервер, не втручаючись у програмну логіку та не перезбираючи додаток. Для підвищення рівня безпеки в промислових системах такі дані, особливо паролі та секретні ключі, зазвичай не зберігаються безпосередньо у файлі конфігурації у відкритому вигляді, а завантажуються динамічно зі змінних оточення операційної системи, що запобігає їхньому випадковому потраплянню до системи контролю версій.

Для забезпечення персистентного зберігання даних системи було розгорнуто та налаштовано спеціалізовану базу даних SmartLightsDB у середовищі СУБД PostgreSQL. Процес було реалізовано за допомогою повнофункціонального графічного інтерфейсу pgAdmin, який надав зручні та інтуїтивно зрозумілі інструменти для створення самої бази даних, а також для створення окремого користувача з наданням йому необхідних та достатніх прав доступу, що відповідає принципу найменших привілеїв.

На основі ORM-моделей, які були декларативно описані у файлі models.py за допомогою синтаксису SQLAlchemy, було спроєктовано та створено необхідні таблиці в базі даних: users, lights та light_logs. Цей етап гарантує, що фізична структура бази даних у середовищі PostgreSQL повністю відповідає логічній моделі даних, визначеній у програмному коді, що унеможливлює розбіжності та помилки на рівні взаємодії з даними.

У промислових проєктах процес створення та оновлення структури таблиць зазвичай автоматизується за допомогою спеціалізованих інструментів міграції, таких як Alembic, що дозволяє версіонувати зміни схеми бази даних аналогічно до того, як система контролю версій Git версіонує зміни коду.

Після завершення всіх ретельних підготовчих етапів було здійснено запуск серверної частини. Цей процес ініціюється через стандартний інтерфейс командного рядка, який надається фреймворком Flask. У разі успішного запуску, вбудований веб-сервер, що є частиною Flask і призначений для розробки, починає прослуховувати вказаний локальний порт (зазвичай 5000), роблячи RESTful API додатку доступним для прийому HTTP-запитів з локальної машини. Консольне повідомлення про успішний старт сервера, а також відсутність у логах будь-яких помилок, пов'язаних з підключенням до бази даних чи ініціалізацією інших сервісів, слугує первинним підтвердженням коректності всіх виконаних налаштувань та готовності системи до повноцінної роботи. На рисунку 3.1 показано запуск Flask API сервера.

Фінальним, верифікаційним етапом розгортання є всебічна перевірка функціональності та коректності роботи розробленого програмного інтерфейсу (API). Для цієї мети було використано спеціалізовану утиліту для тестування API – Postman, яка є галузевим стандартом для таких завдань. Було створено окрему колекцію запитів, що охоплює всі ключові ендпоінти, та проведено серію тестових сценаріїв для перевірки всіх основних CRUD-операцій (Create, Read, Update, Delete) над ресурсом «ліхтар». Було ретельно перевірено можливість

успішного отримання повного списку ліхтарів через GET-запит та валідацію структури отриманої відповіді.

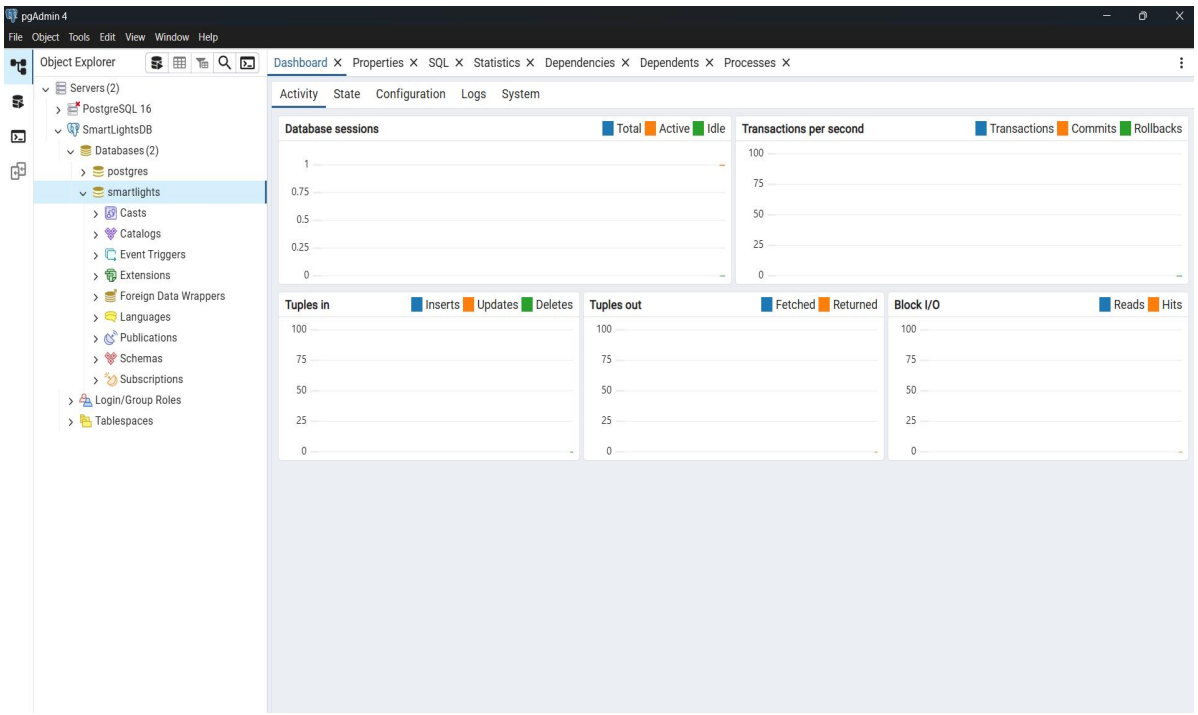


Рисунок 3.1 – запуск Flask API сервера у терміналі

Було проведено тестування створення нового запису шляхом передачі коректних JSON-даних у тілі POST-запиту та перевірки успішного створення об'єкта в базі даних. Було перевірено коректність оновлення інформації про існуючий ліхтар через PUT-запит, а також верифіковано можливість його повного видалення з системи через DELETE-запит. Успішне проходження всіх цих тестів, що підтверджується отриманням очікуваних кодів стану HTTP (200 OK, 201 Created, 404 Not Found тощо) та коректними даними у відповідях, стало остаточним підтвердженням повної працездатності та коректності роботи розробленого API.

Таким чином, у межах даного підрозділу було детально описано послідовний, структурований та методологічно обґрунтований процес розгортання серверної частини кіберфізичної системи «Розумні ліхтарі». Застосування таких сучасних практик, як віртуальні середовища для ізоляції

залежностей, чітка та логічна структура проєкту для підтримуваності коду, а також централізоване управління конфігурацією, забезпечує високий рівень відтворюваності, стабільності та безпеки системи. Проведене ретельне тестування ключових ендпоінтів API за допомогою спеціалізованих інструментів підтвердило повну працездатність розробленого функціоналу та його готовність до подальшої інтеграції з клієнтськими застосунками та іншими компонентами системи.

3.3 Функціональне тестування та верифікація працезданості API

Функціональне тестування є невід'ємним, критично важливим та методологічно обґрунтованим етапом життєвого циклу розробки будь-якого програмного забезпечення. Цей процес дозволяє провести емпіричну верифікацію та перевірити відповідність реалізованого програмного продукту заздалегідь визначеним функціональним вимогам, специфікаціям та очікуваній поведінці. Після успішного завершення етапу розгортання серверної частини проєкту «Розумні ліхтарі» та налаштування всього необхідного робочого оточення, було проведено комплексне та всебічне функціональне тестування розробленого програмного інтерфейсу (API).

Основною метою даного етапу було не просто формальне підтвердження працездатності, а глибока верифікація коректності обробки HTTP-запитів, перевірка стабільності, надійності та консистентності взаємодії з системою управління базами даних PostgreSQL, а також підтвердження цілісності даних під час виконання повного циклу базових операцій створення, зчитування, оновлення та видалення (CRUD). Цей етап є ключовим для забезпечення впевненості у якості розробленого прототипу перед переходом до подальшого розширення його функціоналу.

Для досягнення поставлених цілей тестування було обрано комбінований підхід, що поєднував у собі симуляцію клієнтських запитів за принципом «чорної скриньки» та безпосередній моніторинг стану бази даних за принципом «білої

скриньки», що дозволило отримати повну та об'єктивну картину роботи системи. Для цього було залучено спеціалізований та загальновизнаний у галузі інструментарій.

Програма Postman була використана як основний та потужний інструмент для інтерактивного тестування RESTful API. Це багатофункціональне середовище дозволяє гнучко та з високою точністю конструювати, надсилати та детально аналізувати HTTP-запити і відповіді від сервера, ефективно імітуючи поведінку реального клієнтського застосунку, такого як веб-панель адміністратора. Водночас, клієнт pgAdmin 4 було застосовано як засіб для адміністрування, моніторингу та прямої взаємодії з СУБД PostgreSQL.

Цей графічний клієнт надав безцінну можливість в режимі реального часу відстежувати та аналізувати зміни в таблицях бази даних, що дозволило безпосередньо верифікувати успішність та коректність виконання кожної операції на рівні персистентного збереження даних, підтверджуючи, що зміни, ініційовані через API, дійсно були зафіксовані в базі даних.

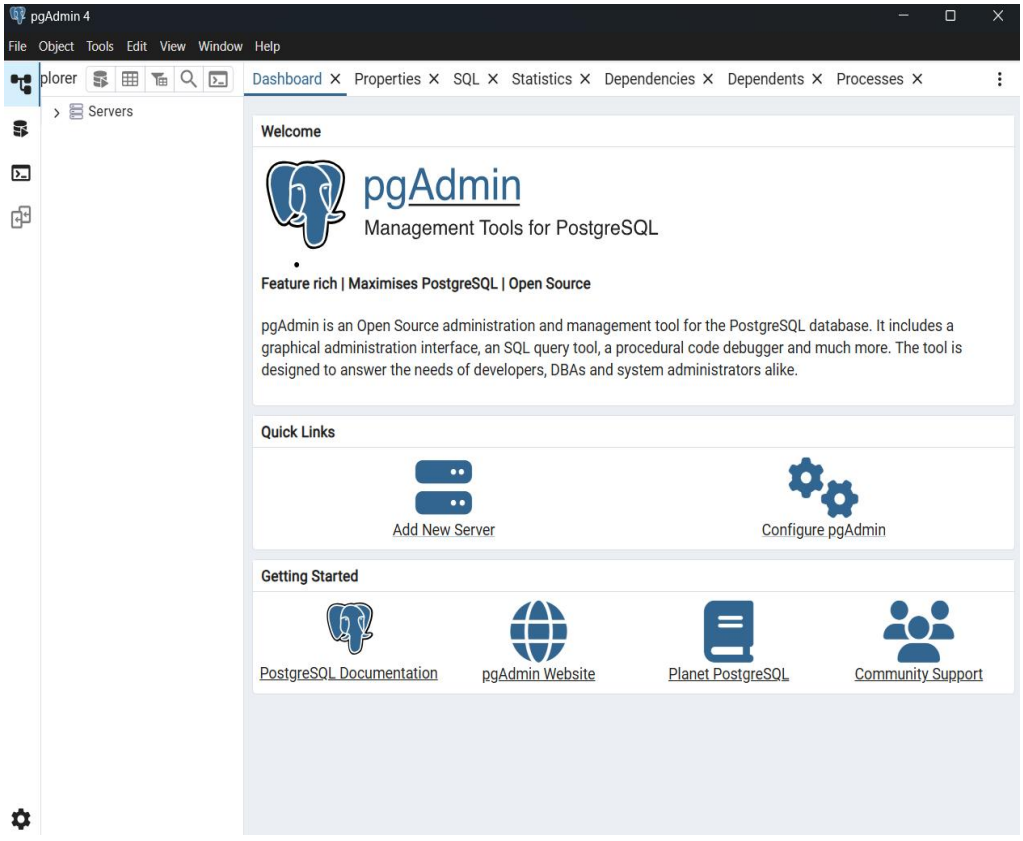


Рисунок 3.2 − Вікно pgAdmin з відкритою базою даних

Процес тестування було логічно структуровано та зосереджено на ендпоінтах, що відповідають за управління ключовим ресурсом системи – lights. Він послідовно охоплював усі чотири основні CRUD-операції, що є фундаментом будь-якого RESTful API. Першим кроком стала ретельна верифікація базової операції зчитування даних. Було сформовано та надіслано GET-запит до маршруту /lights. Очікуваним результатом, згідно зі спроєктованою специфікацією API, була відповідь від сервера з кодом стану 200 OK, що свідчить про успішне виконання запиту, та тілом відповіді у форматі JSON, що містить масив об'єктів, кожен з яких представляє один ліхтар з бази даних. Фактичний результат, отриманий у Postman, повністю збігся з очікуваним, що стало першим важливим підтвердженням коректної роботи логіки маршруту, правильності налаштування з'єднання з базою даних та здатності системи коректно вибирати та серіалізувати дані у потрібний формат.

Наступним, не менш важливим кроком, було тестування функціональності створення нових сутностей у системі. Було сформовано POST-запит до того ж маршруту /lights з передачею в тілі запиту JSON-об'єкта, що містив поля для нового ліхтаря, такі як його місцезнаходження, початковий статус та яскравість. У відповідь на цей запит сервер, як і очікувалося, повернув код стану 201 Created, що є стандартом для успішного створення нового ресурсу, та тіло відповіді, що містило новостворений об'єкт. Важливо відзначити, що цьому об'єкту система автоматично та коректно присвоїла унікальний ідентифікатор, що продемонстровано на відповідному скріншот. Цей позитивний результат довів, що серверна частина коректно приймає, валідує, обробляє вхідні дані та успішно створює новий запис у базі даних.

Для перевірки логіки оновлення існуючих записів, що є однією з найчастіших операцій у реальних системах, було виконано PUT-запит до специфічного маршруту /lights/1, де 1 − це ідентифікатор цільового об'єкта, який було створено на попередньому кроці. У тілі запиту було передано JSON-об'єкт з

оновленими даними, наприклад, зі зміненим статусом або рівнем яскравості. Сервер успішно обробив цей запит, повернувши код стану 200 OK та оновлену версію об'єкта у відповіді, що підтвердило коректну роботу механізму пошуку запису за ідентифікатором та його подальшої модифікації. Фінальним тестовим сценарієм у цьому циклі стала перевірка критично важливої операції видалення через надсилання DELETE-запиту до того ж маршруту /lights/1. Успішна відповідь від сервера з кодом 200 OK та відповідним повідомленням засвідчила, що запит було коректно оброблено, а відповідний запис − безповоротно видалено з бази даних.

Паралельно з тестуванням через Postman, після виконання кожної мутуючої операції (тобто POST, PUT та DELETE), проводився ретельний аудит стану таблиці lights через графічний інтерфейс pgAdmin. Такий двосторонній контроль, що поєднує функціональне тестування з тестуванням на рівні даних, дозволив візуально та однозначно переконатися, що всі зміни, ініційовані через API, коректно, атомарно та консистентно відображаються на рівні бази даних. Було підтверджено, що після успішного виконання POST-запиту в таблиці з'являється новий рядок з відповідними даними; після PUT-запиту оновлюються значення у відповідних стовпцях цільового рядка; а після DELETE-запиту відповідний рядок повністю та безслідно видаляється. Це гарантує відсутність будь-якої розсинхронізації між станом додатку та станом його персистентного сховища даних, що є запорукою цілісності та надійності всієї системи.

Проведений комплекс ретельних тестових заходів дозволив зробити впевнені та обґрунтовані висновки про повну функціональну відповідність та високу працездатність розробленої системи на даному етапі. Було емпірично підтверджено, що реалізована серверна частина коректно обробляє всі основні типи HTTP-запитів, забезпечуючи повний та функціональний набір CRUD-операцій, що є основою для будь-якого подальшого розвитку. З'єднання з базою даних є стабільним, а механізм ORM SQLAlchemy забезпечує коректне, надійне та своєчасне відображення змін у базі даних, абстрагуючи розробника від

складнощів SQL. Розроблене API повністю відповідає принципам архітектурного стилю REST, що робить його універсальним, передбачуваним та готовим до легкої інтеграції з різноманітними клієнтськими застосунками, розробленими на будь-яких технологіях.

Таким чином, етап функціонального тестування повністю підтвердив працездатність, надійність та коректність роботи розробленої серверної частини API. Було продемонстровано успішну, стабільну та надійну інтеграцію між програмною логікою на фреймворку Flask та системою управління базами даних PostgreSQL. Успішне завершення цього етапу верифікації формує міцну, надійну та перевірену основу для подальшого ітеративного розширення функціональності системи, такого як реалізація більш складних алгоритмів керування, додавання механізмів автентифікації та інтеграція з MQTT-брокером для отримання даних від реальних пристроїв.

3.4 Налаштування MQTT-зв’язку та симуляція роботи з пристроями

Однією з найважливіших задач кіберфізичної системи «Розумні ліхтарі» є налагодження надійного, ефективного та масштабованого каналу зв'язку між центральним сервером та кожним окремим освітлювальним приладом. Для вирішення цього завдання було обрано та інтегровано спеціалізований протокол MQTT, який функціонує за асинхронною моделлю «видавець–підписник». На відміну від синхронного протоколу HTTP, MQTT був розроблений спеціально для середовища Інтернету речей, де пристрої можуть мати обмежені ресурси, а мережеве з'єднання може бути нестабільним.

Центральним елементом цієї архітектури є MQTT-брокер, який виступає посередником для всіх повідомлень. Кожен ліхтар («видавець») надсилає свої телеметричні дані на певну адресу, що називається топіком, наприклад, lights/telemetry. Серверна частина, у свою чергу, виступає в ролі «підписника», який прослуховує цей топік. Як тільки на ньому з'являється нове повідомлення,

брокер миттєво доставляє його серверу. Такий підхід забезпечує високу ефективність завдяки мінімальному розміру повідомлень, надійність через підтримку рівнів якості обслуговування та гнучкість, оскільки серверу не потрібно встановлювати пряме з'єднання з кожним пристроєм.

Для інтеграції MQTT-функціоналу в серверну частину на Flask було використано бібліотеку paho-mqtt. У програмному коді було створено спеціальний MQTT-клієнт, який виконує підключення до брокера, підписується на необхідні топіки для отримання даних та має можливість публікувати команди керування у відповідні топіки. Таким чином, сервер одночасно виступає і як підписник, і як видавець.

Оскільки на етапі розробки доступ до реальних фізичних пристроїв був відсутній, для тестування комунікаційного каналу було розроблено скрипти-емулятори на Python. Кожен такий емулятор симулював поведінку справжнього «розумного ліхтаря»: підключався до MQTT-брокера, періодично генерував та надсилав повідомлення з телеметрією у відповідний топік та одночасно прослуховував свій персональний топік для отримання команд керування від сервера. Такий підхід дозволив повністю протестувати всю логіку обміну даними.

Ключовим моментом тестування стала перевірка наскрізного потоку даних. У коді Flask-сервера було реалізовано функцію-обробник on_message, яка автоматично активується при отриманні нового повідомлення через MQTT. Ця функція розпаковує дані з формату JSON та зберігає отриману інформацію в базу даних, створюючи відповідний запис у таблиці light_logs. Під час тестування, після запуску емуляторів, ми спостерігали, як кожне надіслане повідомлення миттєво з'являлося у вигляді нового запису в базі даних, що підтвердило повну працездатність системи.

Таким чином, у цьому підрозділі було успішно налаштовано та верифіковано двосторонній зв'язок між сервером та пристроями за допомогою протоколу MQTT. Симуляція роботи фізичних об'єктів дозволила повністю перевірити та налагодити всю логіку обміну даними в режимі реального часу, що

доводить гнучкість, масштабованість та готовність системи до інтеграції з реальними «розумними ліхтарями».

3.5 Локальне розгортання та комплексне налагодження системи

Завершальним етапом практичної реалізації проєкту стало повне розгортання всіх розроблених компонентів системи в єдиному локальному середовищі. Метою цього етапу було об'єднання серверного застосунку, бази даних та MQTT-брокера в одне ціле та перевірка їхньої злагодженої спільної працездатності. Цей процес є своєрідною генеральною репетицією перед можливим розгортанням на реальному сервері, що дозволяє виявити та виправити потенційні проблеми взаємодії між компонентами в контрольованому середовищі.

Першим кроком стала ретельна підготовка інфраструктури. Було сформовано фінальну структуру проєкту, створено та активовано ізольоване віртуальне середовище Python, в яке було встановлено всі необхідні залежності. Паралельно, за допомогою інструменту pgAdmin було створено та налаштовано базу даних SmartLightsDB в СУБД PostgreSQL. У конфігураційному файлі config.py було прописано коректний рядок підключення до цієї бази даних. Фінальним підготовчим кроком стало налаштування та запуск локального MQTT-брокера Mosquitto, який виконував роль посередника для обміну повідомленнями.

Коли всі компоненти були налаштовані, було здійснено перший запуск всієї системи. Запуск Flask-сервера супроводжувався детальними логами в консолі, які підтвердили, що сервер не тільки успішно стартував, але й встановив стабільне з'єднання як з базою даних PostgreSQL, так і з MQTT-брокером. Це стало первинним підтвердженням коректності всіх конфігураційних параметрів.

Далі було перевірено наскрізну логіку роботи. Після запуску скрипта-емулятора, який надсилав тестове повідомлення з телеметрією через MQTT, ми спостерігали в логах сервера, що функція-обробник on_message успішно спрацювала, отримала дані та виконала запис у базу даних.

Це було негайно перевірено через pgAdmin. Наостанок було проведено повторну серію тестових запитів через Postman до API-ендпоінтів, які також відпрацювали коректно, підтвердивши працездатність API-рівня в умовах комплексної роботи всієї системи, на прикладі рисунків 3.3–3.6. Тестування охоплювало основні CRUD-операції для роботи з даними ліхтарів, користувачів і сенсорних показань. Кожен запит повертав очікувану відповідь сервера, включно зі статусом HTTP 200 або відповідними кодами для помилок. Особливу увагу було приділено перевірці автентифікації та авторизації користувачів, щоб гарантувати надійний доступ до адміністративного функціоналу. Завдяки використанню Postman вдалося візуалізувати структуру відповідей API у зручному форматі, що значно спростило аналіз результатів тестування. Усі тести підтвердили стабільну роботу системи на рівні обміну даними між клієнтом і сервером.

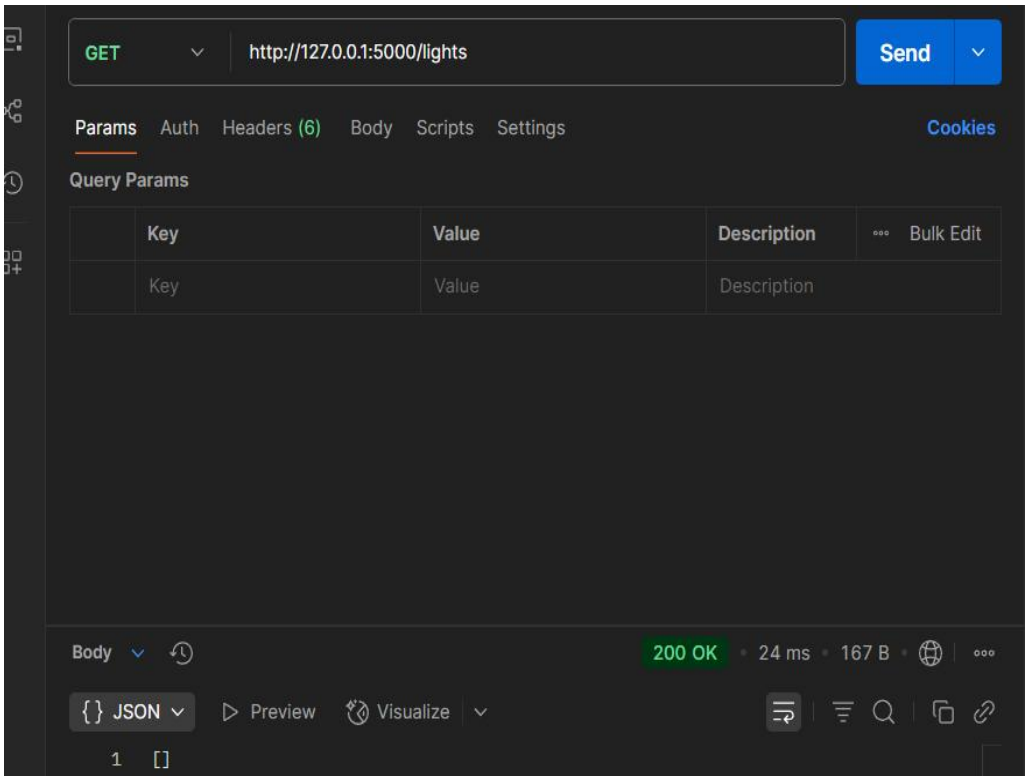


Рисунок 3.3 – Результат виконання GET-запиту до ресурсу /lights отримання всіх ліхтарів у системі; порожній список свідчить про відсутність записів на момент запиту

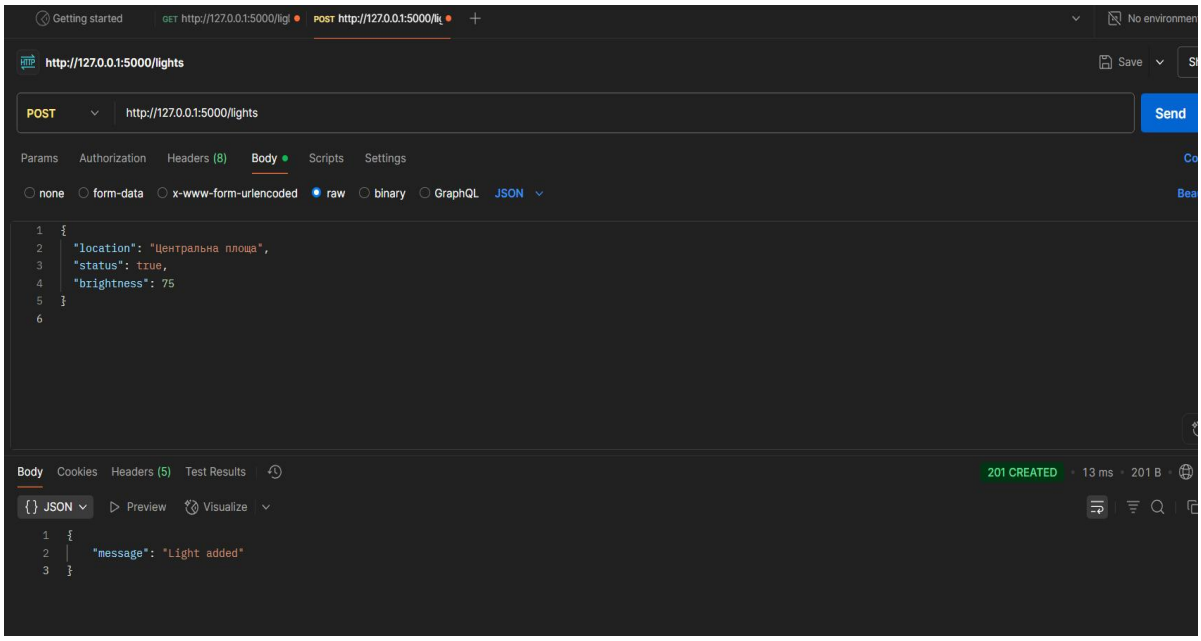


Рисунок 3.4 – результат виконання POST-запиту до ресурсу /lights додавання нового ліхтаря до бази даних; сервер повертає повідомлення про успішне створення об’єкта

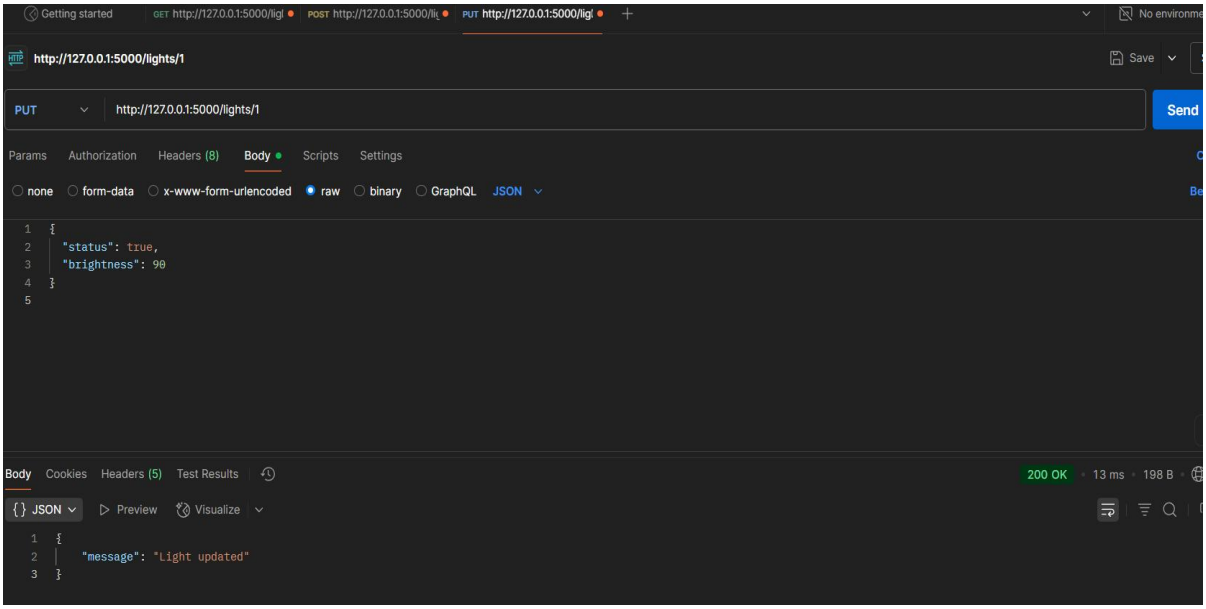


Рисунок 3.5 – результат виконання PUT-запиту до ресурсу /lights/1 (оновлення даних вже створеного ліхтаря, зокрема зміна його яскравості)

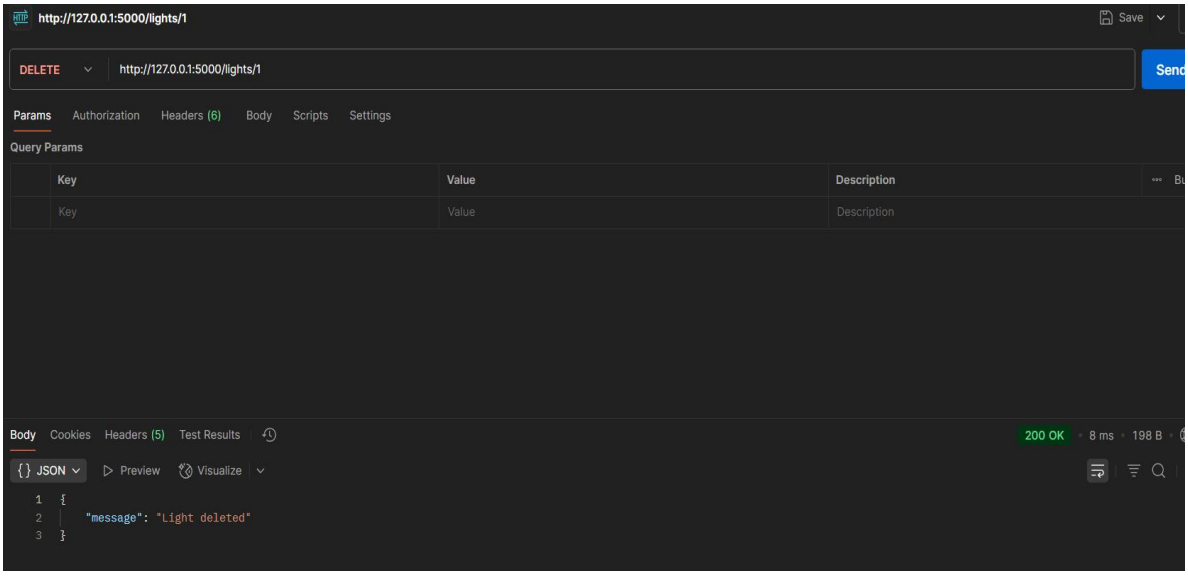


Рисунок 3.6 − результат виконання DELETE-запиту до ресурсу /lights/1 (видалення ліхтаря з бази даних; сервер підтверджує видалення відповідним повідомленням)

Проведені роботи з локального розгортання та налагодження дозволили зробити впевнені висновки про готовність системи. Результати цього етапу повністю підтвердили, що всі розроблені модулі та обрані технології коректно працюють разом. Було налагоджено стабільний двосторонній канал зв'язку через MQTT, підтверджено здатність сервера одночасно обробляти API-запити та асинхронні повідомлення, та створено повністю функціонуючий прототип системи, готовий до наступних етапів розробки. Цей етап став логічним завершенням практичної частини, довівши життєздатність обраної архітектури та правильність її реалізації.

За підсумками роботи, виконаної у третьому розділі, було створено та всебічно перевірено повноцінний програмний прототип серверної частини для кіберфізичної системи «Розумні ліхтарі». Цей розділ охопив увесь практичний цикл розробки: від детального проєктування архітектури та написання програмного коду до розгортання, комплексного налагодження та фінального тестування працездатності розробленого рішення.

Насамперед, в якості фундаменту для всієї системи була закладена гнучка модульна архітектура, що дозволило чітко розмежувати логічні компоненти та в майбутньому значно спростить підтримку та розширення функціоналу. На цій основі, за допомогою фреймворку Flask, було реалізовано повноцінний RESTful API, який надає весь необхідний інструментарій для управління об'єктами системи через стандартні CRUD-операції, забезпечуючи легку та стандартизовану інтеграцію з будь-якими клієнтськими застосунками.

Особливу увагу було приділено організації надійної та ефективної роботи з даними. Було успішно налаштовано взаємодію з промисловою базою даних PostgreSQL, а використання технології ORM SQLAlchemy дозволило побудувати безпечний та абстрагований міст між бізнес-логікою додатку та реляційною структурою зберігання інформації. Для забезпечення життєво важливого для кіберфізичної системи зв'язку з кінцевими пристроями було інтегровано ключовий комунікаційний канал на базі протоколу MQTT. Це рішення забезпечило можливість миттєвого двостороннього обміну повідомленнями та гарантує високу масштабованість системи в майбутньому.

Оскільки тестування на реальному обладнанні на даному етапі було неможливим, було розроблено спеціалізовані скрипти-емулятори. Вони повністю імітували поведінку "розумних ліхтарів", надсилаючи телеметрію та реагуючи на команди. Завдяки цьому, а також за допомогою інструментів Postman та pgAdmin, було проведено комплексне наскрізне тестування. Ми переконалися, що кожен компонент системи − від API-ендпоінтів до механізму запису даних у базу після отримання MQTT-повідомлення − працює коректно та стабільно. Процес локального розгортання пройшов успішно, підтвердивши правильність налаштувань та взаємодії всіх частин системи.

Таким чином, у третьому розділі було створено цілісну, працездатну та стабільну програмну платформу. Розроблена серверна частина є готовим до впровадження прототипом, який довів свою ефективність і гнучкість та є

надійною основою для подальшого розвитку проєкту в рамках концепції «Розумного міста».

3.6 Висновки до третього розділу

У третьому розділі кваліфікаційної роботи представлено комплексне рішення щодо практичної реалізації серверної частини кіберфізичної системи «Розумні ліхтарі». Робота охоплювала етапи від обґрунтування вибору технологічного стеку та проєктування програмної архітектури до розгортання, налаштування робочого середовища, функціонального тестування та верифікації працездатності розробленого програмного інтерфейсу. Ключовим результатом стало створення повноцінного програмного прототипу, що забезпечує централізоване управління, обробку даних та взаємодію з кінцевими пристроями.

На етапі розробки програмного забезпечення для системи детально обґрунтовано вибір мікрофреймворку Flask на мові Python для реалізації серверної логіки. Застосовано модульну архітектуру з використанням Flask Blueprints. Для взаємодії з базою даних обрано PostgreSQL та бібліотеку SQLAlchemy, що реалізує шаблон ORM. Описано функціональне призначення ключових програмних модулів.

У підрозділі, присвяченому процесу налаштування середовища та розгортання, детально описано кроки для створення відтворюваного робочого оточення, включаючи організацію структури проєкту, використання віртуальних середовищ, управління залежностями та конфігурацією. Продемонстровано процес створення та налаштування бази даних у PostgreSQL.

Критично важливим етапом стало функціональне тестування розробленого API. За допомогою інструменту Postman проведено перевірку CRUD-операцій для ресурсу «ліхтар». Підтверджено коректність обробки HTTP-запитів та взаємодії з базою даних.

Налаштування MQTT-зв’язку та симуляція роботи з пристроями продемонстрували можливості системи для взаємодії з елементами Інтернету

речей. Інтегровано бібліотеку paho-mqtt та розроблено скрипти-емулятори, що дозволило перевірити наскрізний потік даних через MQTT-брокера.

Етап локального розгортання та комплексного налагодження об'єднав усі компоненти в єдине функціонуюче середовище. Проведено фінальну перевірку їхньої злагодженої роботи, що засвідчило готовність розробленого прототипу.

Таким чином, у третьому розділі було успішно реалізовано та всебічно протестовано серверну частину кіберфізичної системи. Розроблений програмний прототип забезпечує необхідний інструментарій для управління даними, взаємодії з пристроями та надає стандартизований API, що створює основу для подальшого розвитку системи.

ВИСНОВКИ

У ході виконання даної дипломної роботи було успішно досягнуто поставленої мети, яка полягала в розробці та дослідженні програмної серверної частини для кіберфізичної системи «Розумні ліхтарі». Виконана робота охопила повний цикл створення програмного продукту: від глибокого теоретичного аналізу предметної області та формулювання вимог до проєктування архітектури, безпосередньої програмної реалізації та всебічного тестування розробленого прототипу.

На першому етапі роботи було проведено комплексний аналіз сучасного стану систем інтелектуального освітлення. Було виявлено та систематизовано ключові недоліки традиційних підходів, зокрема їхню низьку енергоефективність, високі експлуатаційні витрати та відсутність гнучкості в управлінні. Це дозволило обґрунтувати високу актуальність та практичну значущість переходу до інтелектуальних рішень на основі концепції кіберфізичних систем. Було детально розглянуто архітектуру та принципи функціонування кіберфізичної системи, що стало теоретичним підґрунтям для подальшого проєктування. Результатом цього аналітичного етапу стало формування вичерпного переліку функціональних та нефункціональних вимог до серверної частини, які охопили аспекти збору даних, керування, аналітики, безпеки, масштабованості та надійності, а також чітка постановка задачі на розробку.

Другий етап було присвячено проєктуванню архітектури та ключових компонентів системи. На основі сформульованих вимог та з урахуванням специфіки прототипування було обрано модульну монолітну архітектуру, яка забезпечує простоту розробки та розгортання на поточному етапі, однак закладає можливість легкої трансформації в мікросервісну архітектуру в майбутньому. Було обґрунтовано вибір технологічного стеку, в основі якого лежить мова програмування Python у поєднанні з мікрофреймворком Flask, що забезпечило необхідну гнучкість та швидкість розробки. В якості системи управління базами даних було обрано PostgreSQL, що гарантує високу надійність та цілісність даних,

а для взаємодії з кінцевими пристроями − протокол MQTT, який є стандартом для Інтернету речей. Було детально спроєктовано логічну структуру бази даних, що включає таблиці для зберігання інформації про ліхтарі, їхні історичні стани та користувачів системи. Також було розроблено специфікацію програмного інтерфейсу (API) на основі архітектурного стилю REST, що визначає набір ендпоінтів для управління всіма ключовими ресурсами системи.

Третій, практичний етап роботи, охоплював програмну реалізацію, розгортання та тестування спроєктованої системи. Було створено програмний код серверної частини відповідно до розробленої модульної архітектури. Особливу увагу було приділено налаштуванню робочого середовища з використанням віртуальних середовищ та чітким управлінням залежностями, що забезпечує відтворюваність та стабільність розробки. Було проведено комплексне функціональне тестування розробленого RESTful API за допомогою інструменту Postman. В ході тестування було верифіковано коректність обробки всіх основних CRUD-операцій, підтверджено надійність взаємодії з базою даних PostgreSQL та цілісність даних на всіх етапах. Для перевірки комунікаційного каналу з пристроями було налаштовано взаємодію з MQTT-брокером та розроблено скрипти-емулятори, що симулювали поведінку «розумних ліхтарів». Наскрізне тестування підтвердило, що система коректно отримує телеметричні дані через MQTT та зберігає їх у базу даних в режимі реального часу. Фінальним кроком стало комплексне локальне розгортання всіх компонентів системи, яке довело їхню злагоджену та стабільну спільну роботу.

Таким чином, в результаті виконання дипломної роботи було створено повноцінний, працездатний та всебічно перевірений програмний прототип серверної частини для кіберфізичної системи «Розумні ліхтарі». Розроблена система демонструє здатність виконувати ключові функції: керування освітлювальними приладами через стандартизований API та отримання телеметричних даних через спеціалізований IoT-протокол. Практичне значення отриманих результатів полягає в тому, що створений програмний продукт може

бути використаний як надійна основа для подальшого розвитку та впровадження повномасштабних систем інтелектуального освітлення.

Напрямками для подальшого розвитку проєкту можуть стати: реалізація більш складних адаптивних алгоритмів керування на основі машинного навчання для аналізу патернів руху та предиктивного обслуговування; розробка повноцінного графічного веб-інтерфейсу для адміністраторів та операторів системи; посилення механізмів безпеки, включаючи впровадження рольової моделі доступу та шифрування даних; а також перехід від монолітної до мікросервісної архітектури для забезпечення вищої масштабованості та відмовостійкості при розгортанні на рівні цілого міста.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Abarro C. C., Caliwag A. C., Valverde E. C., Lim W., Maier M. Implementation of IoT-based low-delay smart streetlight monitoring system. *IEEE Internet of Things Journal*. 2022. Vol. 9, No. 19. P. 18461–18472.

2. Anitha C., Tellur A., Rao K. B. V. B., Kumbhar V., Gopi T., Jadhav S., Vidhya R. G. Enhancing CyberPhysical Systems Dependability through Integrated CPS IoT Monitoring. *International Research Journal of Multidisciplinary Scope*. 2024. Vol. 5, No. 2. P. 706–713.

3. Aravind S., Deepakumaran S., Zubair A. M., Selvam M. E., Manish M. P., Venkatasamy B. Intelligent Street Lighting System Using RF Communication. *2025 8th International Conference on Trends in Electronics and Informatics (ICOEI)*. 2025. P. 104–110.

4. Avotins A., Adrian L. R., Porins R., Apse-Apsitis P., Ribickis L. Smart city street lighting system qualityand control issues to increase energy efficiency and safety. *The Baltic Journal of Road and Bridge Engineering*. 2021. Vol. 16, No. 4. P. 28–57.

5. Baharudin N. H., Mansur T. M. N. T., Ali R., Sobri N. F. A. Smart lighting system control strategies for commercial buildings: A review. *International Journal of Advanced Technology and Engineering Exploration*. 2021. Vol. 8, No. 74. P. 45.

6. Bauer M., Sanchez L., Song J. IoT-enabled smart cities: Evolution and outlook. *Sensors*. 2021. Vol. 21, No. 13. P. 4511.

7. Bertoldi P., Zissis G. Smart Lighting systems: Dream or reality. *12th International Conference on Improving Energy Efficiency in Commercial Buildings and Smart Communities (IEECB&SC'24) and European ESCO Conference*. 2024.

8. Bhattacharya S., Somayaji S. R. K., Gadekallu T. R., Alazab M., Maddikunta P. K. R. A review on deep learning for future smart cities. *Internet Technology Letters*. 2022. Vol. 5, No. 1. P. 187.

9. Cassottana B., Roomi M. M., Mashima D., Sansavini G. Resilience analysis of cyber-physical systems: A review of models and methods. *Risk Analysis*. 2023. Vol. 43, No. 11. P. 2359–2379.

10. Chen J., An J., Yan D., Zhou X. Future trends in intelligent lighting control systems: Integrated technologies, multi-system linkage and AI integration. *Building Simulation*. 2024. P. 1–24.

11. Chen Z., Sivaparthipan C. B., Muthu B. IoT based smart and intelligent smart city energy optimization. *Sustainable Energy Technologies and Assessments*. 2022. Vol. 49. P. 101724.

12. Chiang S.-B., Tien C.-H. Smart Lighting Control Built on AI Architecture. *International Conference on Technologies and Applications of Artificial Intelligence.* 2024. P. 262–275.

13. Chiradeja P., Yoomak S. Development of public lighting system with smart lighting control systems and internet of thing (IoT) technologies for smart city. *Energy Reports*. 2023. Vol. 10. P. 3355–3372.

14. De Paz J. F., Bajo J., Rodríguez S., Villarrubia G., Corchado J. M. Intelligent system for lighting control in smart cities. *Information Sciences*. 2016. Vol. 372. P. 241–255.

15. Deepaisarn S., Yiwsiw P., Chaisawat S., Lerttomolsakul T., Cheewakriengkrai L., Tantiwattanapaibul C., Buaruk S., Sornlertlamvanich V. Automated street light adjustment system on campus with AI-assisted data analytics. *Sensors*. 2023. Vol. 23, No. 4. P. 1853.

16. Dizon E., Pranggono B. Smart streetlights in Smart City: a case study of Sheffield. *Journal of Ambient Intelligence and Humanized Computing*. 2022. P. 1–16.

17. Ferrell S. S., Lunsford R. CPS Energy: Leading the way to a sustainable future. *Journal of Business Case Studies*. 2015. Vol. 11, No. 4. P. 177.

18. Fu R., Zou A., Chen C., Guan X., Ma Y. Mesh Network Scheduling Based on Cyber-Physical Sensitivity for Wireless Control Systems. *2025 IEEE 31st Real-*

Time and Embedded Technology and Applications Symposium (RTAS). 2025. P. 123–136.

19. Füchtenhans M., Grosse E. H., Glock C. H. Smart lighting systems: state-of-the-art and potential applications in warehouse order picking. *International Journal of Production Research*. 2021. Vol. 59, No. 12. P. 3817–3839.

20. Garcia E. AI-Driven Smart Lighting Systems for Energy-Efficient and Adaptive Urban Environments.

21. Gowda V. D., Annepu A., Ramesha M., Kumar K. P., Singh P. IoT enabled smart lighting system for smart cities. *Journal of Physics: Conference Series*. 2021. Vol. 2089, No. 1. P. 012037.

22. Hawkins-Stark H., Sheldon F., Abdel-Rahim A. Internet of Things (IoT) Platforms for Smart City Implementation in Rural and Urban Communities: A Comprehensive Review. *Pacific Northwest Transportation Consortium*. 2022.

23. Hintaw A. J., Manickam S., Aboalmaaly M. F., Karuppayah S. MQTT vulnerabilities, attack vectors and solutions in the internet of things (IoT). *IETE Journal of Research*. 2023. Vol. 69, No. 6. P. 3368–3397.

24. Jenzeri D., Chehri A. Data analysis for IoT system using 6LoWPAN and constrained applicationprotocol for environmental monitoring. *Procedia Computer Science*. 2022. Vol. 207. P. 1472–1481.

25. Kashef M., Visvizi A., Troisi O. Smart city as a smart service system: Human-computer interaction and smart city surveillance systems. *Computers in Human Behavior*. 2021. Vol. 124. P. 106923.

26. Liang J., Ku N. C. LED Roadway Lighting Control Scheme Based on TALQ Protocol. *Advanced Materials Research*. 2014. Vol. 1044. P. 1541–1544.

27. Ma C. Smart city and cyber-security; technologies used, leading challenges and future recommendations. *Energy Reports*. 2021. Vol. 7. P. 7999–8012.

28. Marino F., Leccese F., Pizzuti S. Adaptive street lighting predictive control. *Energy Procedia*. 2017. Vol. 111. P. 790–799.

29. Nair R. Lighting as a Service. *Lighting Design Application*. 2021. Vol. 51, No. 8. P. 52–53.

30. Ntentos E., Amiri A., Warnett S., Zdun U. Decision-making support for data integration in cyberphysical-system architectures. *International Conference on Service-Oriented Computing*. 2023. P. 137–152.

31. Nounou A., Jaber H., Aydin R. A cyber-physical system architecture based on lean principles formanaging industry 4.0 setups. *International Journal of Computer Integrated Manufacturing*. 2022. Vol.35, No. 8. P. 890–908.

32. Omar A., AlMaeeni S., Attia H., Takruri M., Altunaiji A., Sanduleanu M., Shubair R., Ashhab M. S., Al Ali M., Al Hebsi G. Smart city: Recent advances in intelligent street lighting systems based on IoT. *Journal of Sensors*. 2022. Vol. 2022, No. 1. P. 5249187.

33. Pardo-Bosch F., Blanco A., Sesé E., Ezcurra F., Pujadas P. Sustainable strategy for the implementationof energy efficient smart public lighting in urban areas: case study in San Sebastian. *Sustainable Citiesand Society*. 2022. Vol. 76. P. 103454.

34. Petkovic M., Bajovic D., Vukobratovic D., Machaj J., Brida P., McCutcheon G., Stankovic L., Stankovic V. Smart dimmable LED lighting systems. *Sensors*. 2022. Vol. 22, No. 21. P. 8523.

35. Kugele S., Hutzelmann T., Beffart T., Bergemann S., Pretschner A. Defining adaptivity and logical architecture for engineering (smart) self-adaptive cyber-physical systems. *Information and Software Technology*. 2022. Vol. 147. P. 106866.

36. Pivoto D. G. S., De Almeida L. F. F., da Rosa Righi R., Rodrigues J. J. P. C., Lugli A. B., Alberti A. M.Cyber-physical systems architectures for industrial internet of things applications in Industry 4.0: A literature review. *Journal of manufacturing systems*. 2021. Vol. 58. P. 176–192.

37. Potenza G., Baglivo C., Bonomolo M., Ribino P. Self-adaptive and Self-learning Lighting System: Integrating LSTM and RL for Energy Efficiency and

Personalized Visual Comfort. *Advances in Signal Processing and Artificial Intelligence*. 2025. P. 125.

38. Rajeswaran N., Swarupa M. L., Maddula R., Alhelou H. H., Kesava Vamsi Krishna V. A study on cyberphysical system architecture for smart grids and its cyber vulnerability. *Power systems cybersecurity: Methods, concepts, and best practices*. 2023. P. 413–427.

39. Rehan H. Internet of Things (IoT) in smart cities: Enhancing urban living through technology. *Journal of Engineering and Technology*. 2023. Vol. 5, No. 1. P. 1–16.

40. Saeed M. M., Al-Hamoud R. A., Adam A. Intelligent Lighting Control System Using AI and IoT. *2024th International Conference on Ubiquitous Computing and Intelligent Information Systems (ICUIS)*. 2024. P. 1701–1705.

41. Shaheen H. I., Gapit M., Gireesan S., Lešić V. Model predictive control of street lighting based on spatial points of interest. *IFAC-PapersOnLine*. 2023. Vol. 56, No. 2. P. 9417–9422.

42. Shahidinejad A., Ghobaei-Arani M., Souri A., Shojafar M., Kumari S. Light-edge: A lightweight authentication protocol for IoT devices in an edge-cloud environment. *IEEE consumer electronics magazine*. 2021. Vol. 11, No. 2. P. 57–63.

43. Soe R.-M., Ruohomäki T., Patzig H. Urban open platform for borderless smart cities. *Applied Sciences*. 2022. Vol. 12, No. 2. P. 700.

44. Soheilian M., Fischl G., Aries M. Smart lighting application for energy saving and user well-being in the residential environment. *Sustainability*. 2021. Vol. 13, No. 11. P. 6198.

45. Soudbakhsh D., Annaswamy A. M., Wang Y., Brunton S. L., Gaudio J., Hussain H., Vrabie D., Drgona J., Filev D. Data-driven control: Theory and applications. *2023 American Control Conference (ACC)*. 2023.P. 1922–1939.

46. Stolojescu-Crisan C., Crisan C., Butunoi B.-P. An IoT-based smart home automation system. *Sensors*. 2021. Vol. 21, No. 11. P. 3784.

47. Swain S., Pattanayak B. K., Mohanty M. N., Senapati C. Analytical Performance Comparison of IoT Communication Protocols MQTT and CoAP from Security Perspective. *2024 3rd Odisha International Conference on Electrical Power Engineering, Communication and Computing Technology (ODICON)*. 2024. P. 1–5.

48. Vashishtha K., Saad A., Faieghi R., Xi F. Intelligent adaptive lighting algorithm: Integrating reinforcement learning and fuzzy logic for personalized interior lighting. *Engineering Applications of Artificial Intelligence*. 2024. Vol. 133. P. 108512.

49. Voets S. J. J., Den Ouden E., Van E., Steenbergen V. R., Valkenburg R. Smart Lighting Adoption Within Dutch Municipalities. 2023.

50. Wang H. Research on real-time reliability evaluation of CPS system based on machine learning. *Computer communications*. 2020. Vol. 157. P. 336–342.

51. Wu P., Zhang Z., Peng X., Wang R. Deep learning solutions for smart city challenges in urban development. *Scientific Reports*. 2024. Vol. 14, No. 1. P. 5176.

52. Zhang J., Chen Z., Wang A., Li Z., Wan W. Intelligent personalized lighting control system for residents. *Sustainability*. 2023. Vol. 15, No. 21. P. 15355.

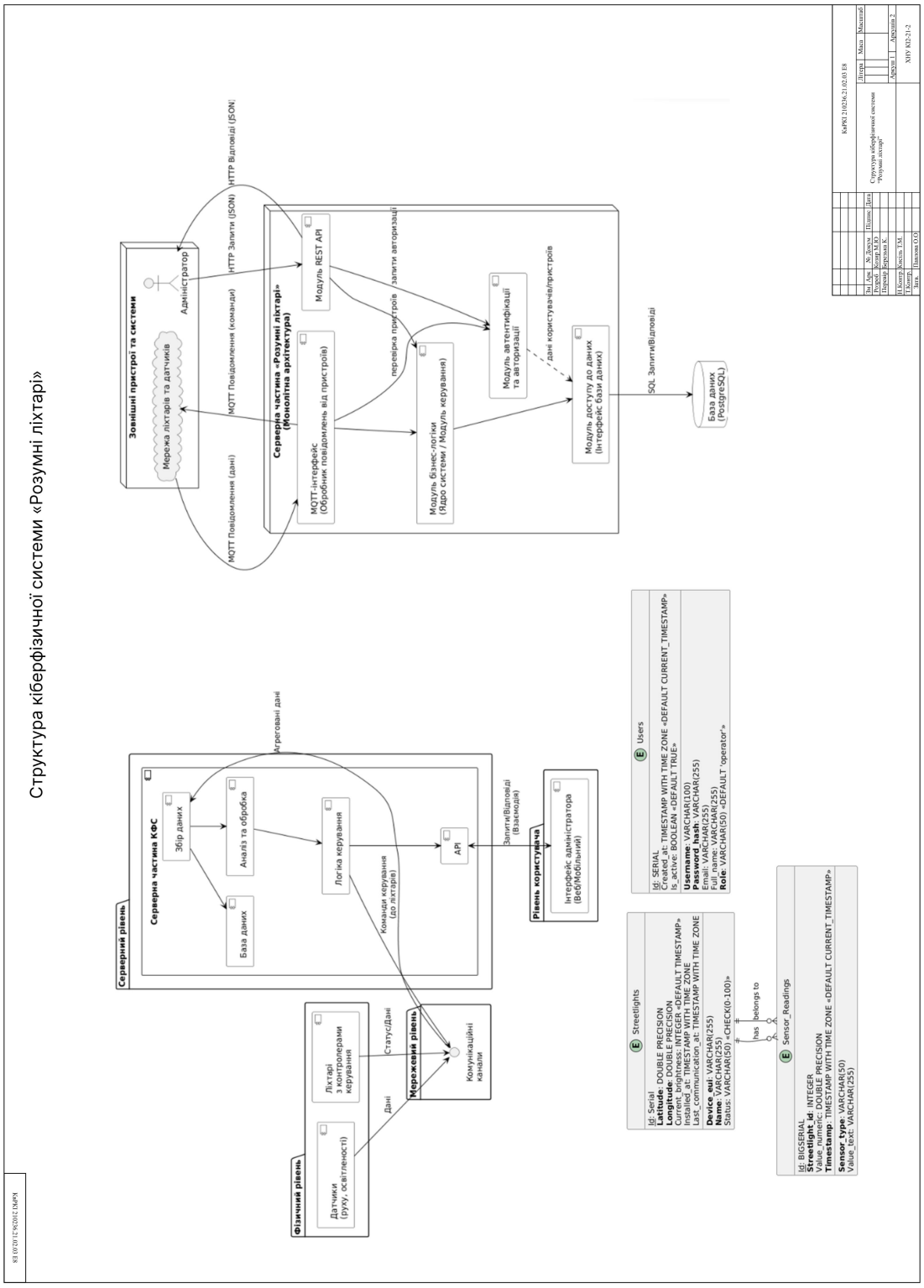
53. Zhang Y., Beetz J. Building-CPS: cyber-physical system for building environment monitoring. *Proc. Of the Conference CIB W78*. 2021. Vol. 2021. P. 11–15.

54. Zhou J., Sun J., Zhang M., Ma Y. Dependable scheduling for real-time workflows on cyber-physical cloud systems. *IEEE Transactions on Industrial Informatics*. 2020. Vol. 17, No. 11. P. 7820–7829.

ДОДАТОК А

(обов’язковий)

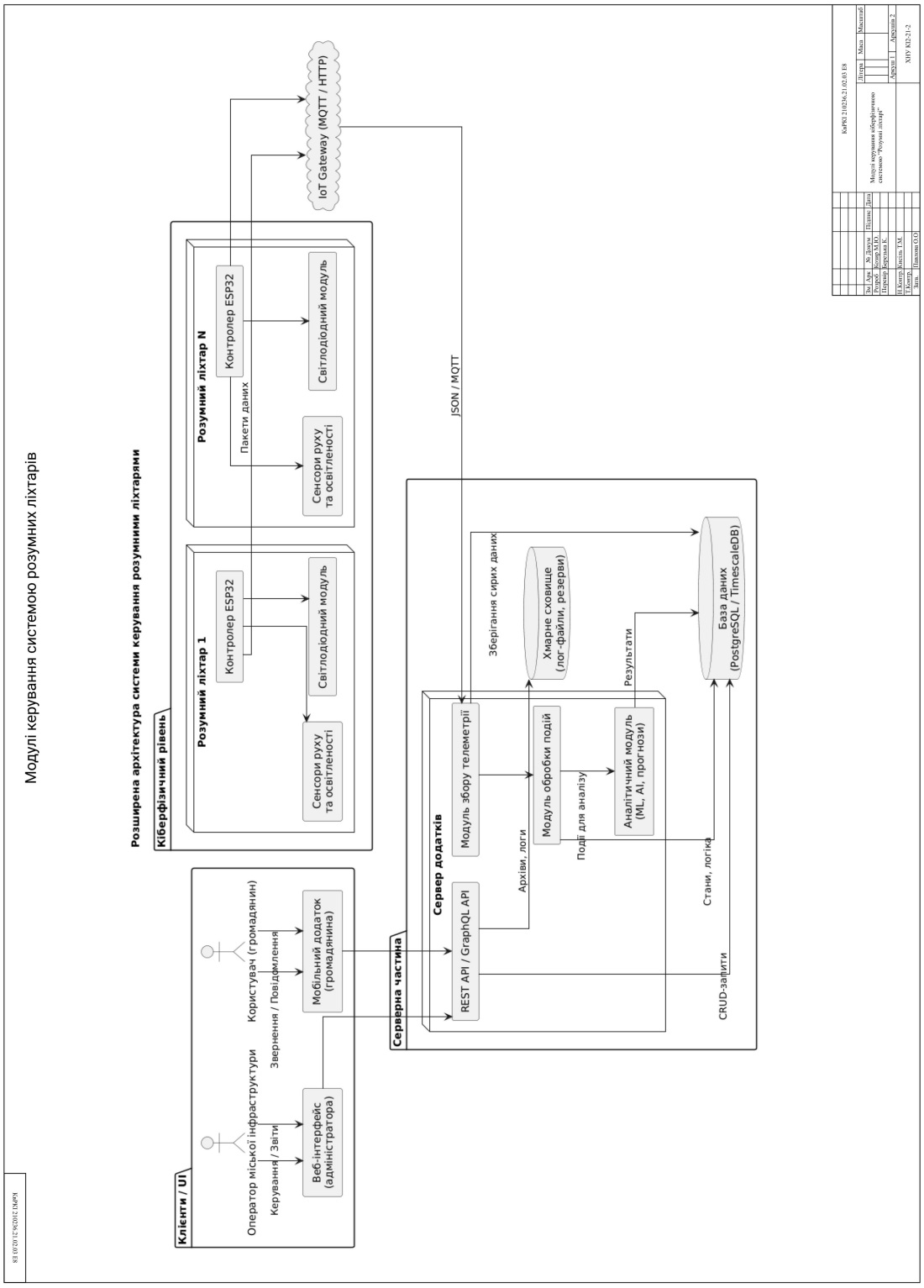
КОПІЯ КРЕСЛЕННЯ «СТРУКТУРА КІБЕРФІЗИЧНОЇ СИСТЕМИ «РОЗУМНІ ЛІХТАРІ»»



ДОДАТОК Б

(обов’язковий)

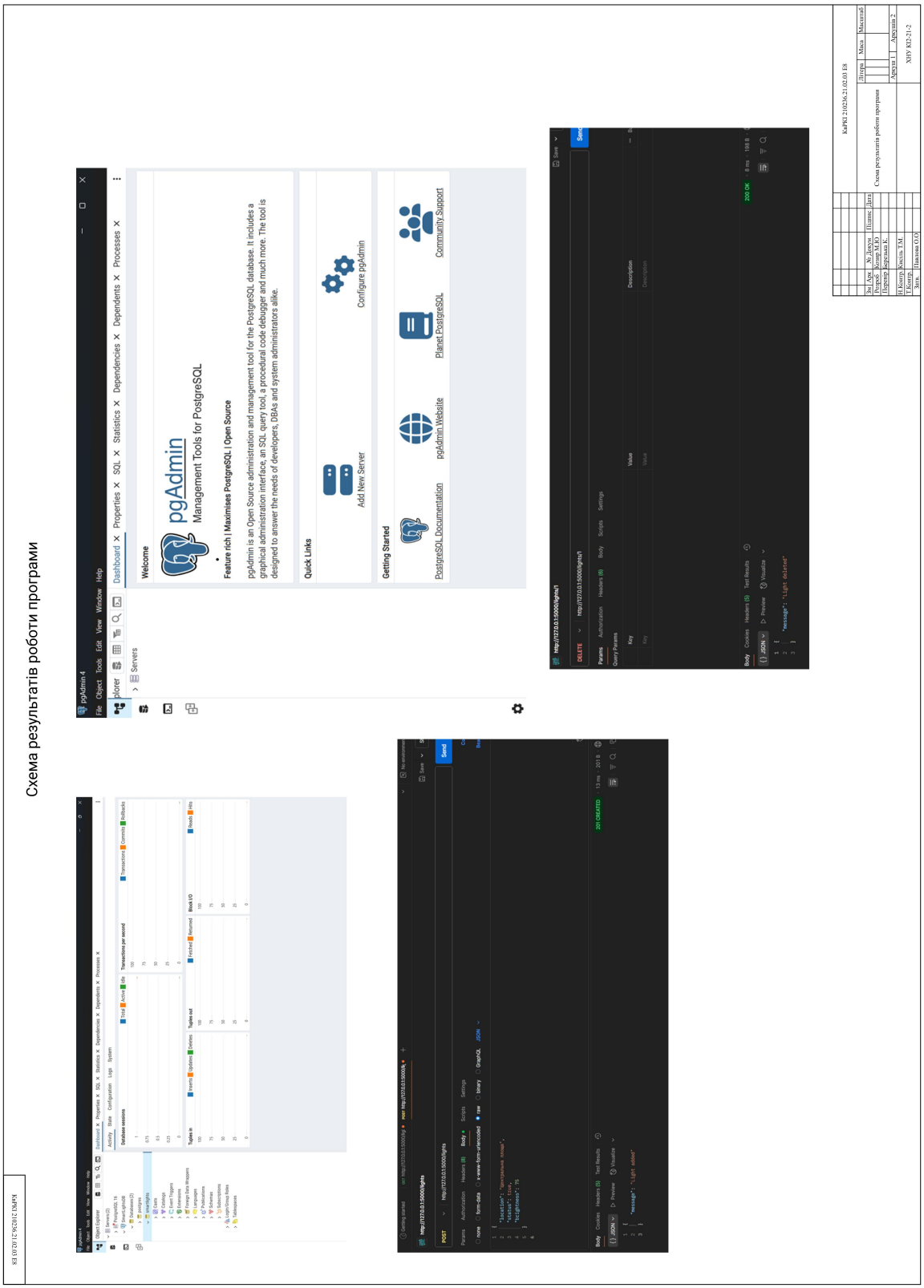
КОПІЯ КРЕСЛЕННЯ «МОДУЛІ КЕРУВАННЯ СИСТЕМОЮ РОЗУМНИХ ЛІХТАРІВ»



ДОДАТОК В

(обов'язковий)

КОПІЯ КРЕСЛЕННЯ «СХЕМА РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМИ»



Fri Jun 13 16:54:27 EEST 2025, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 0.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 13%**

ID: 245758 Title: БКР Кіберфізична система "Розумні ліхтарі". Серверна частина Added in a DB: 2025-06-13 Authors: Максим КОЗИР Heads: Катерина БЕРЕЗЬКА Consultants: Opponents:	Document		Sum coincidence on the DB	
	Symbols	Lexemes	Symbols	Lexemes
	100975	634	1163 (1%)	19 (3%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Максим КОЗИР

Співавтор:

Назва: Козир_Кіберфізична система "Розумні ліхтарі". Серверна частина

Експерт:

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:6.1%

Коефіцієнт подібності 2:2.7%

Мікропробіли: 7

Заміна букв: 0

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-13 18:26:35.0

***Після аналізу* Звіту подібності констатую наступне:**

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2025-06-13 Доцент Андрій Нічепорук

Дата експерт

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Максим КОЗИР

Тема: Кіберфізична система "Розумні ліхтарі". Серверна частина

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки _____

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є розробка та дослідження серверної частини кіберфізичної системи «Розумні ліхтарі», яка забезпечує прийом, обробку та зберігання даних від освітлювальних пристроїв і датчиків.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено аналіз предметної області кіберфізичної системи «Розумні ліхтарі». В другому розділі кваліфікаційної роботи виконано проєктування серверної частини кіберфізичної системи розумного ліхтаря. В третьому розділі кваліфікаційної роботи описано програмну реалізацію та тестування серверної частини кіберфізичної системи «Розумні ліхтарі».

4. Позитивні сторони роботи: Висока практична цінність роботи.

5. Негативні сторони роботи:

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на високому інженерно-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре (4.25/B)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Гаусов Михайло Васильович доктор філософії,
ст. викладач кафедри кібербезпеки

"__"_____2025 р. _____(підпис)

Завідувачу кафедри КІІС
д-р. філософії, доц. Ользі ПАВЛОВІЙ

Максима Козира
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-21-2

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Strike-Plagiarism та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

2025 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Кіберфізична система "Розумні Ліхтарі". Серверна частина

Автор: Максим КОЗИР

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Катерина БЕРЕЗЬКА, к.т.н, доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) Запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, що є загальним оглядом літератури та не стосуються безпосередньо авторського дослідження чи оригінальних результатів роботи.
2) Усі запозичення є фрагментарними та належним чином оформлені відповідно до академічних вимог, з наданням точних посилань на джерела, з яких вони були отримані.
3) Окремі збіги, зафіксовані системою, є загальновживаними фразами або термінами, що часто використовуються в галузі, про що свідчить численні збіги з 10-40 джерелами на один фрагмент тексту. Такі фрази не є результатом авторського плагіату, оскільки належать до загальновизнаних термінів.
4) В окремих випадках система зафіксувала послідовності чотиризначних двійкових кодів, що є типовими вхідними даними для множини задач, і не можуть розглядатися як об'єкт авторських прав, оскільки ці послідовності є стандартними елементами для математичних чи технічних розрахунків.
5) Всі зафіксовані системою ознаки модифікації тексту пов'язані з комбінуванням латинських символів зі скороченнями українських індексів у формулах. Це не є модифікацією змісту, оскільки такі скорочення є стандартною практикою в математичній та технічній літературі і не порушують авторських прав.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 2% і адресується до 401 першоджерела; та системою Anti-Plagiarism складає 1%, що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Катерина БЕРЕЗЬКА

Гарант ОП _____ Андрій НІЧЕПОРУК

Завідувач кафедри КПС _____ Ольга ПАВЛОВА