НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Хмарне програмне забезпечення для моніторингу освітлення міста

Виконав студент IV курсу, групи ІП-96
(шифр групи)
Бурятов Олексій Олексійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ст.викл. Вітковська Ірина Іванівна _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації ст.викл. Вітковська Ірина Іванівна _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент д.т.н., проф. Писарчук Олексій Олександрович _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Бурятовому Олексію Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Хмарне програмне забезпечення для моніторингу освітлення міста

керівник проєкту Вітковська Ірина Іванівна, ст.викл.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «31»травня 2023 р. №2101-с

2. Термін подання студентом проєкту « 17 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.
3) Моделювання програмного забезпечення: архітектура програмного забезпечення та приватної хмари, конструювання програмного забезпечення та аналіз безпеки
4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, побудова звітів.
5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема бази даних

3) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» березня 2023 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	21.02.2023	
2	Аналіз існуючих методів розв’язання задачі	15.03.2023	
3	Постановка та формалізація задачі	25.03.2023	
4	Розробка інформаційного забезпечення	15.04.2023	
5	Алгоритмізація задачі	25.04.2023	
6	Обґрунтування вибору використаних технічних засобів	30.04.2023	
7	Розробка програмного забезпечення	10.05.2023	
8	Налагодження програми	13.05.2023	
9	Виконання графічних документів	17.05.2023	
10	Оформлення пояснювальної записки	19.05.2023	
11	Подання ДП на попередній захист	25.05.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	17.06.2023	

Студент _____ Олексій БУРЯТОВ
(підпис) (ініціали, прізвище)

Керівник _____ Ірина ВІТКОВСЬКА
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з чотирьох розділів, містить 44 таблиці, 43 рисунки та 12 джерел – загалом 88 сторінок.

Дипломний проект присвячений розробці хмарного програмного забезпечення для моніторингу освітлення міста.

Мета: підвищення рівня механізму моніторингу за освітлювальними приборами в місті, зменшення ризиків зберігання даних на наземних дата центрах шляхом впровадження хмарної інфраструктури

Об'єкт дослідження: програмне забезпечення для моніторингу світлом у місті.

Предмет дослідження: технологія «розумного міста», що передбачає збір інформації з систем освітлення та її обробка; моніторинг – при виявленні критичних показників або інших невідомих станів, відбувається сповіщення для усунення інцидентів та віртуалізація – використання хмарних технологій для високої доступності та продуктивності системи.

У розділі «Моделювання та конструювання програмного забезпечення», було описано архітектуру хмарної інфраструктури, розроблено концепцію сховища даних та процес передачі та отримання даних…

Розділ «Аналіз якості та тестування програмного забезпечення» присвячений тестуванню програмного забезпечення на вразливості з метою усунення недоліків. Було проведено мануальне тестування, а також аналіз коду, який допоміг усунути незначні помилки.

Четвертий розділ пояснювальної записки «Впровадження та супровід програмного забезпечення» описує процес взаємодії програміста з хмарною платформою Microsoft Azure, покрокові дії розгортання інфраструктури й компонент в хмарі та інструменти захисту даних від потенційних зловмисників.

Програмне забезпечення впроваджено в хмарній платформі Microsoft Azure, використовуючи методологію Infrastructure as code та Terraform.

Результати роботи пройшли апробацію на IV Міжнародній конференції «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2023)» та опубліковані в доменній зоні kpi.ua.

КЛЮЧОВІ СЛОВА: IOT, ARDUINO, МОНІТОРИНГ, SMART CITY, MICROSOFT AZURE, СХОВИЩЕ ДАНИХ, ETL

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 44 tables, 43 figures and 12 sources – in total 87 pages.

The purpose of the diploma project is development of cloud-based software for city lighting monitoring.

Goal: increasing level of the monitoring mechanism for street light devices in the city, reducing the risks of data storage in terrestrial data centers through the implementation of cloud infrastructure.

Object of research: software for light monitoring in the city.

The subject of research: "smart city" technology, which involves the collection of information from lighting systems and its processing; monitoring - when critical indicators or other unknown conditions are detected and notification which occurs to eliminate incidents; and virtualization - the use of cloud technologies for high availability and system performance.

In the section "Software modeling and construction" was described the architecture of the cloud infrastructure, the concept of data storage and the process of extract, transform and load data into data warehouse.

The "Software Quality Analysis and Testing" section is devoted to testing software for vulnerabilities in order to eliminate defects and fix them.

Manual testing was performed, as well as code analysis, which helped to eliminate minor bugs.

The fourth section of the explanatory note "Software implementation and maintenance" describes the process of interaction of the programmer with the Microsoft Azure cloud platform, step-by-step deployment of infrastructure and components in the cloud, and data protection tools from potential hackers.

The software is implemented in the Microsoft Azure cloud platform using the Infrastructure as code and Terraform methodology.

The results of the work were approved at the IV International Conference "Software Engineering and Advanced Information Technologies (SoftTech-2023)" and published in the domain zone kpi.ua.

KEYWORDS: IOT, ARDUINO, MONITORING, SMART CITY, MICROSOFT AZURE, DATA WAREHOUSE, ETL.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-9604.045430.01.91	Технічне завдання	15	
3	А4	КПІ.ІП-9604.045430.02.81	Пояснювальна записка	83	
4	А4	КПІ.ІП-9604.045430.03.12	Текст програми	48	
5	А4	КПІ.ІП-9604.045430.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІП-9604.045430.05.34	Керівництво користувача	19	
7	А3	КПІ.ІП-9604.045430.06.99	Схема структурна варіантів	1	
			використань		
8	А3	КПІ.ІП-9604.045430.07.99	Схема бази даних	1	
9	А3	КПІ.ІП-9604.045430.08.99	Креслення вигляду екранних	1	
			форм		

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІП-9604.045430.00.90		
Розроб.		Бурятов О.О.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Вітковська І.І.					
Н.контр.		Вітковська І.І.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-96	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ХМАРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ОСВІТЛЕННЯ МІСТА

Технічне завдання

КПІ.ІП-9604.045440.01.91

“ПОГОДЖЕНО”
Керівник проекту:
_____ Ірина ВІТКОВСЬКА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Олексій БУРЯТОВ

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ 3
2 ПІДСТАВА ДЛЯ РОЗРОБКИ 4
3 ПРИЗНАЧЕННЯ РОЗРОБКИ 5
4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 6
4.1 Вимоги до функціональних характеристик 6
4.1.1 Користувацького інтерфейсу 6
4.1.2 Для користувача: 8
4.1.3 Для адміністратора системи: 8
4.1.4 Додаткові вимоги: 9
4.2 Вимоги до надійності 9
4.3 Умови експлуатації 9
4.3.1 Вид обслуговування 9
4.3.2 Обслуговуючий персонал 9
4.4 Вимоги до складу і параметрів технічних засобів 10
4.5 Вимоги до інформаційної та програмної сумісності 10
4.5.1 Вимоги до вхідних даних 11
4.5.2 Вимоги до вихідних даних 11
4.5.3 Вимоги до середовища розробки 12
4.5.4 Вимоги до представленню вихідних кодів 12
4.6 Вимоги до маркування та пакування 12
4.7 Вимоги до транспортування та зберігання 12
4.8 Спеціальні вимоги 12
5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ 13
5.1 Попередній склад програмної документації 13
5.2 Спеціальні вимоги до програмної документації 13
6 СТАДІЇ І ЕТАПИ РОЗРОБКИ 14
7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ 15

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: ХМАРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ОСВІТЛЕННЯ МІСТА.

Галузь застосування:

Наведене технічне завдання поширюється на розробку хмарного програмного забезпечення освітлювальних приладів, котра використовується для моніторингу освітленням у місті та отриманні повідомлень у разі недоступності.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки хмарного програмного забезпечення для моніторингу освітлення в місті є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для моніторингу світлових приладів та керування ними у разі критичної необхідності.

Метою розробки є покращення ефективності освітлення в місті шляхом розробки системи моніторингу для світлових приладів та мікроконтролерів типу Arduino UNO, ESP8266 та WS2812B.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

– отримання персонального доменного імені для веб застосунку;
– доступність застосунку з інтернету;
– передача інформації через HTTPS протокол;
– перегляд усіх доступних датчиків та їхній актуальний стан;

Прототипи екранних форм наведено на рисунках 4.1 – 4.4.

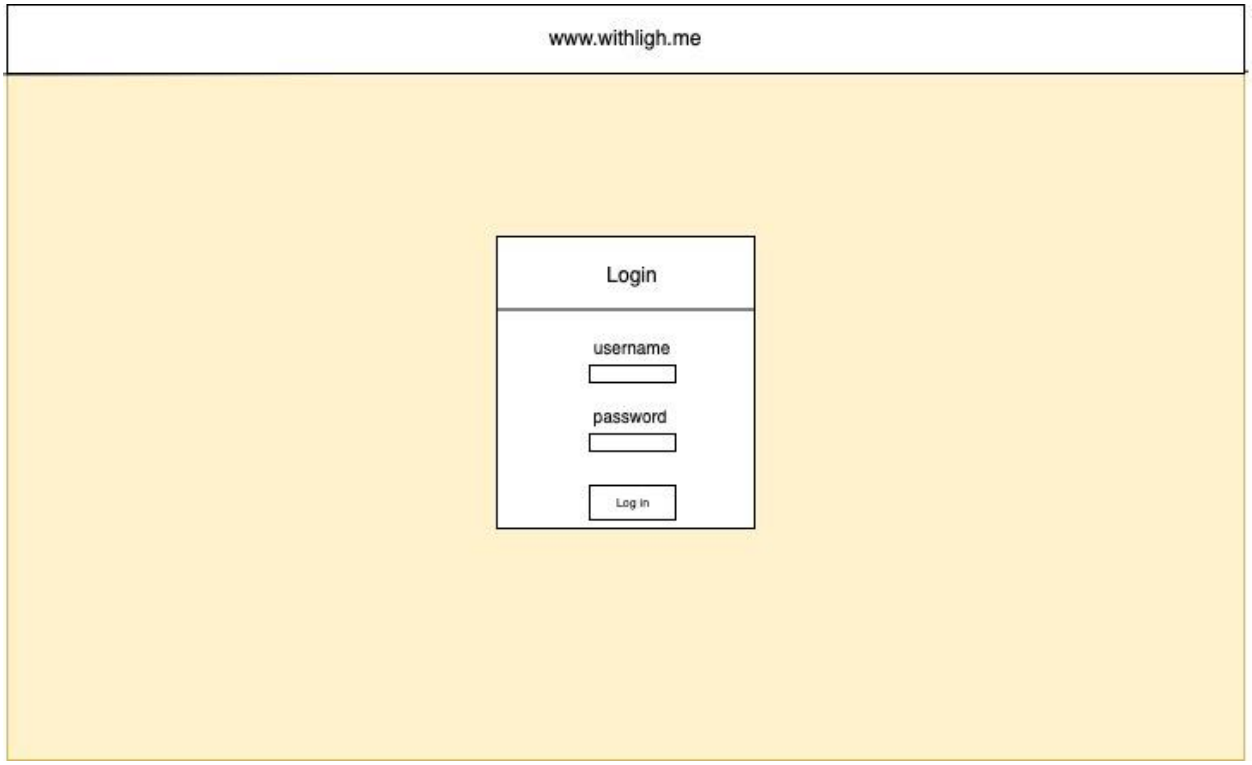


Рисунок 4.1 – Вікно авторизації до веб-застосунку

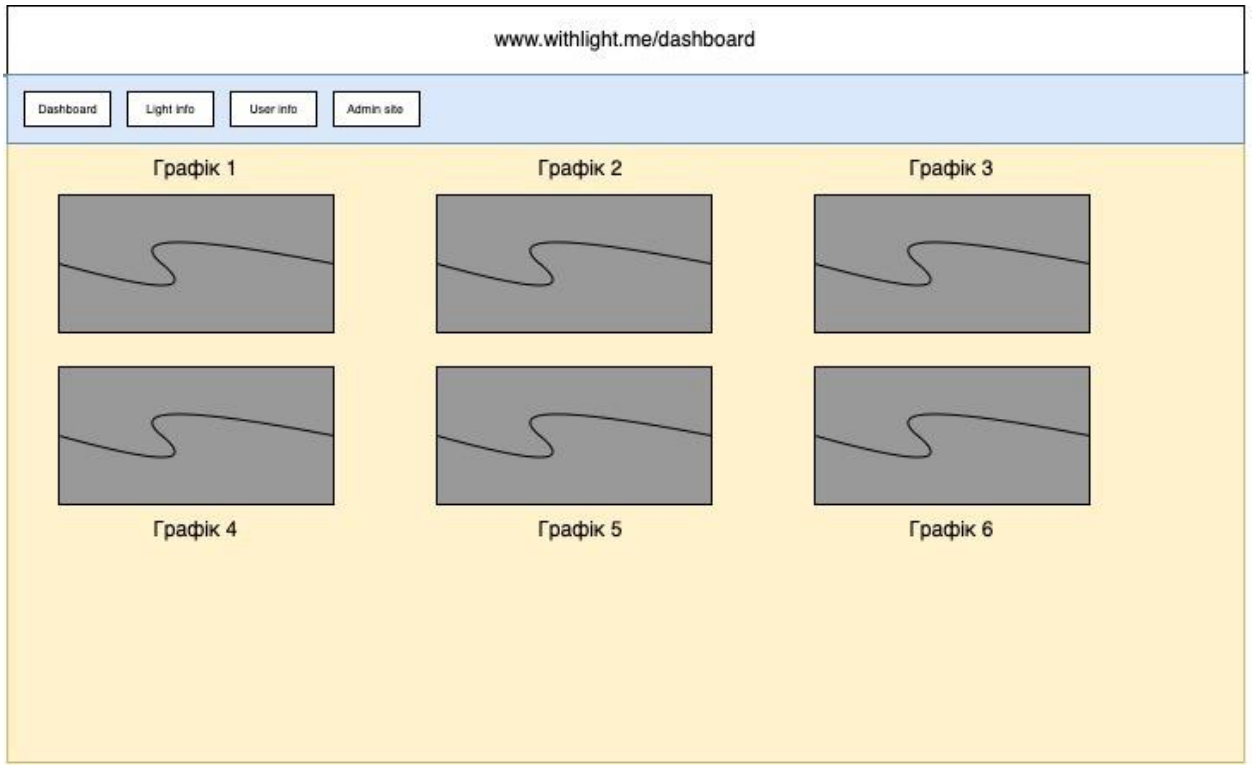


Рисунок 4.2 – Вікно перегляду графіків стану мікроконтролерів

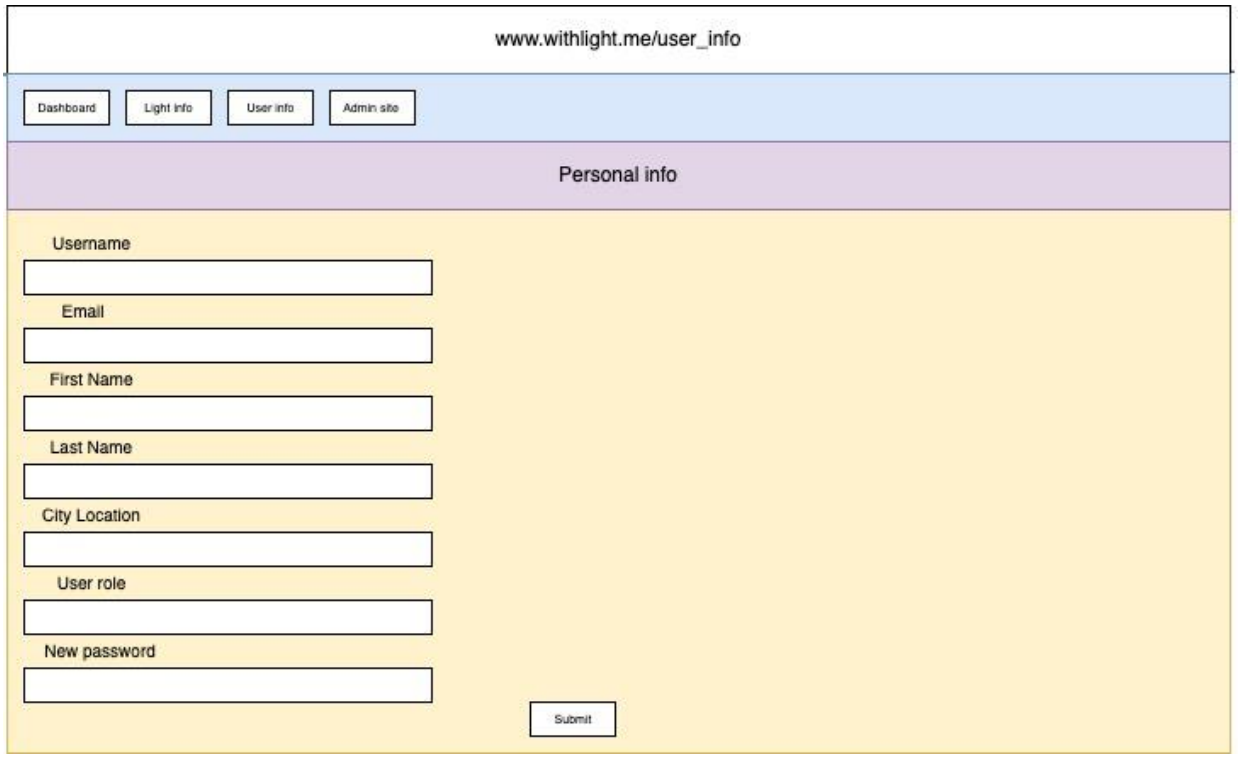


Рисунок 4.3 – Вікно перегляду з персональною інформацією

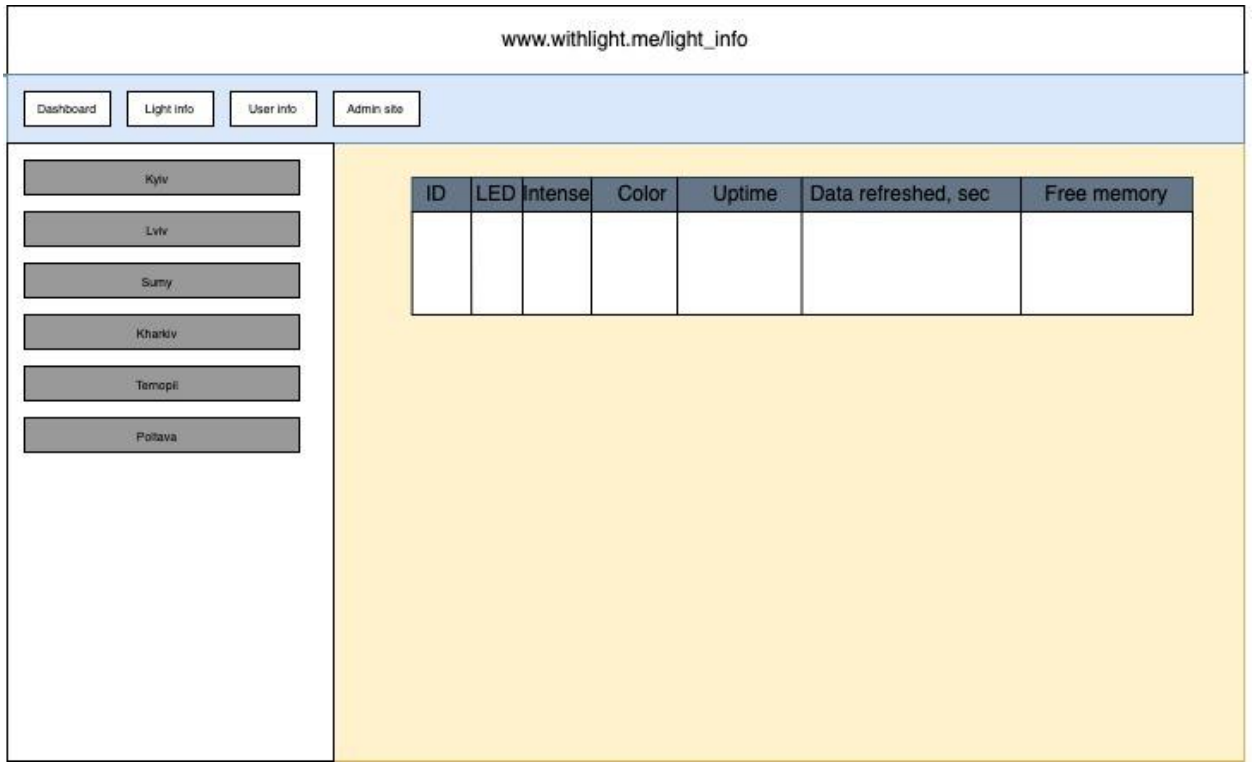


Рисунок 4.4 – Вікно перегляду стану мікроконтролерів у розрізі міста

4.1.2 Для користувача:

– авторизація користувача;

– перегляд метрик кожного освітлювального прибору, до якого він має доступ;

– перегляд та оновлення персональної інформації;

– отримання повідомлень у разі недоступності освітлювальних приладів;

4.1.3 Для адміністратора системи:

– авторизація користувача;

– перегляд метрик кожного освітлювального прибору;

– перегляд та оновлення персональної інформації;

– отримання повідомлень у разі недоступності освітлювальних приладів;

– створення нових користувачів;

– увімкнення режиму «небезпеки», при якому світло буде горіти червоним кольором;

4.1.4 Додаткові вимоги:

– захист передачі інформації за допомогою SSL сертифікатів та Bearer токенів

– розробка сховища даних, використовуючи реляційну базу даних Postgres та time-series базу даних Prometheus

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних. У разі відсутності автономного електропостачання до освітлювальних приборів, створити сповіщення про відсутність зв’язку з цим прибором, а також реалізувати механізм відновлення системи після усунення проблем. Організувати час зберігання логів датчиків протягом 180 днів.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Є необхідність в обслуговуванні хмарних сервісів, які використовуються під даний проєкт. Потрібно періодично переглядати стан сервісів, які розгорнуті в хмарі, а також слідкувати за бюджетом підписки, в якій ці сервіси розгорнуті.

4.3.2 Обслуговуючий персонал

Обслуговуючий персонал (DevOps інженери та System адміністратори) повинен мати досвід роботи з хмарними провайдерами (у даному випадку обраний Microsoft Azure), хмарними сервісами та інструментами моніторингу.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення може бути встановлене на будь-якому UNIX-подібному сервері, а також може бути розгорнуте, використовуючи послуги хмарного провайдера Microsoft Azure.

Мінімальна конфігурація технічних засобів:

Для Azure Kubernetes Services:

– SKU (Stock-Keeping Unit) – Standard_D2_v5;

– CPU Architecture – x64;

– об‘єм ОЗП: 8 Гб;

– vCPUs: 2

– тип процесору: Intel(R) Xeon(R) Platinum 8370C CPU @ 2.80GHz

– об‘єм ОЗП: 4 Гб;

– підключення до мережі Інтернет зі швидкістю від 100 мегабайт;

Для мікроконтролерів:

– Arduino UNO: 32 КБ flash пам’яті, SRAM пам’яті – 2 КБ, 5V напруга

– ESP8266: 32 КБ flash пам’яті, SRAM пам’яті – 2 КБ, 5V напруга, 2.4 ГГц WiFi підключення;

– освітлювальний прилад WS2812B: 256 led ліхтарів, максимальна споживча потужність яких складає 0,3 Вт кожна.

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i7;

– об‘єм ОЗП: 8 Гб;

– підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Unix.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі:


```
{
    "up": 1597,
    "fMemBytes": 98989898,
    "ledCount": 256,
    "timestamp": 1683821536
}
```


Вхідні дані представлені у форматі JSON.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі:


```
{
    "colorHex": "0x5F07EF",
    "deviceId": 1,
    "errorCode": 200,
    "lightIntense": 1,
    "status": "True"
}
```


Вихідні дані представлені у форматі JSON

Вимоги до мови розробки

Розробку програмного забезпечення для програмування мікроконтролерів Arduino UNO та ESP8266 виконати на мові програмування C.

Розробку веб-сервісу для API менеджера виконати на мові програмування Python.

Розробку веб-застосунку виконати на мові програмування Node.JS

4.5.3 Вимоги до середовища розробки

Розробку програмного забезпечення для програмування мікроконтролерів Arduino UNO та ESP8266 виконати на платформі Arduino IDE.

Розробку API менеджера та веб-застосунку виконати за допомогою Visual Studio Code.

4.5.4 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді файлів з наступними розширеннями: .ino, .py, .sql, .tf, .js, .css, .ejs тощо.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Розгортання сервісів у хмарному провайдері Microsoft Azure виконати з допомогою інструменту автоматизації IaC (Infrastructure as a Code – з англ. – інфраструктура за допомогою коду), використовуючи Terraform.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– текст програми;
– програма та методика тестування;
– керівництво користувача;
– керівництво адміністратора.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна компонентів програмного забезпечення;
– схема бази даних;
– архітектура програмного забезпечення;
– креслення вигляду екранних форм;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	21.02	
2.	Розробка технічного завдання	19.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	30.03	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	15.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	25.04	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проекту

на тему: Хмарне програмне забезпечення для моніторингу освітлення міста

КПІ.ІП-9604.045430.02.81

Київ – 2023

ЗМІСТ

ВСТУП 21
1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 23
1.1 Загальні положення 23
1.2 Змістовний опис і аналіз предметної області 23
1.3 Аналіз існуючих успішних ІТ-проєктів 24
1.3.1 Аналіз відомих технічних рішень 24
1.3.2 Аналіз допоміжних програмних засобів та засобів розробки 27
1.3.3 Аналіз відомих програмних продуктів 31
1.4 Аналіз вимог до програмного забезпечення 34
1.4.1 Розроблення функціональних вимог 43
1.4.2 Розроблення нефункціональних вимог 46
1.5 Постановка задачі 47
Висновки до розділу 47
2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 49
2.1 Моделювання та аналіз програмного забезпечення 49
2.2 Архітектура програмного забезпечення 56
2.3 Конструювання програмного забезпечення 57
2.4 Аналіз безпеки даних 64
Висновки до розділу 65
3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 67
3.1 Аналіз якості ПЗ 67
3.2 Опис процесів тестування 70
3.3 Опис контрольного прикладу 75
Висновки до розділу 86
4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ . 87
4.1 Розгортання програмного забезпечення 87

4.2 Підтримка програмного забезпечення 95
Висновки до розділу 96
ВИСНОВКИ 98
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 101
ДОДАТОК А ЗВІТ ПОДІБНОСТИ 103

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Інтегроване середовище розробки
API	–	Інтерфейс програмування веб-сервісів
MCU	–	Microcontroller unit
Big Data	–	(з англ. – великі дані) – сучасний напрям обробки великих масивів даних
MS Azure	–	Microsoft Azure – хмарне середовище з обчислювальних ресурсів
SKU	–	Stock-keeping unit – перелік схожих одиниць в хмарі однієї категорії різного типу.
LB	–	Load balancer – балансувальник трафіку
AKS	–	Azure Kubernetes Services
IoT	–	Internet of Things – інтернет речей
IaC	–	Infrastructure as a code – інфраструктура як код
LRS	–	Local redundant storage – сховище, дані яких є надлишковими у рамках одного дата центру
ZRS	–	Zone redundant storage – сховище, дані яких є надлишковими між зонами однієї країни
GRS	–	Geo redundant storage – сховище, дані яких є надлишковими між країнами
HA	–	High availability – можливість системи оперувати неперервно без простоїв у разі виходу з ладу одного із компонентів
PaaS	–	Platform as a Service – платформа як сервіс – можливість використання систем розробки без необхідності підтримки hardware частини
HTTP	–	Hypertext Transfer Protocol – протокол передачі гіпертекстових документів

HTTPS	–	Hypertext Transfer Protocol Secure – протокол передачі гіпертекстових документів, захищаючи SSL сертифікатами
SSL	–	Secure Sockets Layer – протокол, який забезпечує захищену передачу інформації через мережу Інтернет
Pay-as-you-go	–	Можливість оплати за використані ресурси в хмарі по факту витрат, зазначених у бюджеті
КМ	–	Контрольний механізм
RBAC	–	Role-based access control – метод регуляції доступів до сервісів та/або мережі всередині Kubernetes
ETL	–	Extract, transform and load – процес, який використовується при заборі даних із різних джерел, оброблюється та перекладається в DWH
DWH	–	Data Warehouse – сховище даних, яке зберігає дані з різних джерел.
БД	–	База даних

ВСТУП

Із всесвітнім прогресом розвитку нових технологій, людство має на меті полегшити життя та замінити однотипну й монотонну роботу, яку може виконати за неї робот, «розумний дім» тощо. Усе частіше можна зустріти в побуті автономні прилади, роботи-пилососи. Нещодавно взагалі був прорив штучного інтелекту. Тому постає питання в тому, щоб звичайну роботу, результат якої можна спрогнозувати, віддати автоматизованим системам, уникаючи можливого людського фактору.

Протягом останнього року гостро постало питання в збереженні природних ресурсів, які необхідні для людського існування, а саме: вода, тепло, електроенергія. Також, у зв’язку з постійними повітряними тривогами, є потреба в альтернативних рішеннях сповіщення населення в разі загрози ракетних ударів з повітря. Саме ці теми наразі актуальні, тому що є необхідність в контролі ресурсів та альтернативних методів сповіщення.

Даний дипломний проект ставить за мету підвищення рівня управління громадським освітленням, зниження вартості за обслуговування, а також використання розрахунків обробки та аналізу сенсорної інформації для підтримки та прийняття рішень у разі несправності елементів освітлення [1].

Для вирішення проблеми були визначені задачі:

– розробити програмне забезпечення для мікроконтролерів Arduino UNO, ESP8266 та LED стрічки WS8212B;

– розгорнути приватну хмару з інструментами доступу до сервісів;

– створити сховище даних та автоматизувати ETL процеси отриманої інформації від мікроконтролерів;

– розробити додаткове програмне забезпечення для користувачів, що будуть відслідковувати стан освітлювальних приладів;

– впровадити відмовостійкість системи.

Беручи до уваги світові рішення віддаленим моніторингом світла в місті за допомогою IoT, можна навести приклад китайської компанії IOTCOMM,

яка вже в далекому 2010 році побудувала перший прототип вуличного ліхтаря з IoT модулями. За допомогою постійного зв'язку з датчиками та передачі інформації, вони автоматизували процес увімкнення, вимкнення та проведення технічних робіт.

Наразі зберігання даних та їх резервне копіювання гостро постало в Україні, адже наземні дата центри компаній не є надійним сховищем, бо в будь-який момент можна зазнати втрату даних та доступу до дата центру. Тому більшість розвиваючих корпорацій почали віддавати перевагу хмарним провайдерам, які зараз користуються популярністю. Основними на ринку є: Microsoft Azure, Amazon Web Services, Google Cloud Platform. Головною перевагою використання хмарних сервісів є:

– копіювання даних між зонами дата центрів (ZRS, LRS, GRS);

– використання high availability кластеру з можливістю неперервного користування сервісами без простоїв у часі;

– постійне оновлення PaaS сервісів;

– ранжування об'ємами необхідних віртуальних машин, дисків, мереж тощо;

– погодинна або річна оплата за користування платформою;

– зручність та швидкість розгортання сервісів тощо.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Досліджуючи актуальність даної теми та хмарне впровадження IoT технологій в буденне життя людей, хочеться зазначити, що не всі готові до таких рішень. По-перше, це ресурсовитратний процес, адже необхідна компетенція розробки, впровадження та підтримки цих систем. Проте, навіть незважаючи на ці недоліки, є багато переваг, про які й буде йти мова далі.

Усі ми спостерігали, що в місті часто не працюють ліхтарі, або працюють дуже тьмяно. Це призводить до того, що приходиться ходити повз темні вулиці, що є небезпечним у наш час. Також, протягом останнього року, спостерігалося, що використання світлових приладів на вулиці було необґрунтовано необхідністю, адже іноді вони світили навіть вдень, що є марною витратою електроенергії, при її дефіциті. Тому постала необхідність виправити це неподобство та створити такий світловий прилад, який буде:

- справним;
- віддалено відслідковуватися;
- корисним під час небезпек тощо.

Виходячи з цих критеріїв буде побудований прилад з усіма цими функціями, як от, наприклад, зробила компанія IOTCOMM, яка розробила «розумний» ліхтар, використовуючи IoT технології.

1.2 Змістовний опис і аналіз предметної області

Для того, щоб втілити проект в працюючий стан необхідно проаналізувати всі компоненти та визначити подальші кроки. Перелічимо компоненти, які використовуються в даному проекті [3]:

- мікроконтролери типу Arduino Uno, ESP8266 та WS2812B;
- майданчик для розгортання – хмара Microsoft Azure з партнерською підпискою або підпискою pay-as-you-go;

– середовище для виконання розробки та імплементації застосунків;

– сховище для зберігання коду та конфігураційних файлів розгортання Terraform.

Узагальнюючи, щоб створити End-To-End готовий проект необхідно розробити програмне забезпечення як для мікроконтролерів, так і WEB частини; скомпонувати їх між собою; розгорнути інструменти та сервіси в хмарі та розробити тестування.

Наразі, моніторинг освітленням в основному відбувається вручну й без датчиків енергозбереження, кольору освітлення, а це набагато ускладнює моніторинг та швидке реагування у разі збою з електромережою, виходу з ладу ліхтарів або інших приладів. Окрім цього, все обладнання контролю, а саме: сервери, сховища даних, мережеве обладнання знаходиться на наземних дата центрах, що наразі доволі ризиковано.

Використовуючи ж даний підхід, який впроваджується за допомогою датчиків контролю, а також зберігання всіх даних у хмарі, спрощує моніторинг освітленням, процес масштабування і резервного копіювання (бекап) даних.

1.3 Аналіз існуючих успішних ІТ-проєктів

Було проаналізовано, відоме на сьогодні технічні рішення, а також аналоги, що допоможуть у реалізації “Хмарного програмного забезпечення для моніторингу освітлення міста”. Нижче буде розглянуто допоміжні програмні засоби, а також засоби розробки та готове програмне рішення.

1.3.1 Аналіз відомих технічних рішень

Даний проект націлений на розробку програмного забезпечення для взаємодії датчиків освітлення з основним контрольним механізмом, а також з наявним програмним забезпеченням, яке відповідає за зберігання та відображання логів, а також для повідомлень про можливу аварію на одному з приладів.

Перш за все, необхідно обрати хмарну платформу для використання та розгортання в хмарі. Основними хмарними провайдерами є: Microsoft Azure, Amazon Web Services та Google Cloud Platform [2].

Таблиця 1.1 – Порівняльна таблиця хмарних провайдерів .

Функціонал	Можливості Microsoft Azure	Можливості Amazon Web Services	Можливості Google Cloud Platform
Заявлена доступність сервісів	99.99%	99.95%	99.9%
Цінова політика	Середня (порівняння йде з мінімальною віруальною машиною - 1 vCPU, 1 GB RAM, вартість якої становить $9.67)	Завищена (порівняння йде з мінімальною віруальною машиною - 1 vCPU, 1 GB RAM, вартість якої становить $24.53)	Завищена (порівняння йде з мінімальною віруальною машиною - 1 vCPU, 1 GB RAM, вартість якої становить $13.80)

Так як цінова політика та доступність є одним із найважливіших критеріїв вибору хмари для даного проекту – зупинимося на Microsoft Azure.

Так як сервісів і компонент у даному проекті є багато, то необхідно визначити як саме та де саме буде це ПЗ працювати. Із відомих платформ у Microsoft Azure є AKS. Azure Kubernetes Service – це механізм керування застосунків, сервісів, мереж, які можна розгорнути за допомогою *.yaml* конфігураційних файлів, або автоматизовано, за допомогою Terraform та відповідних провайдерів. Обравши даний компонент, було вирішено низку питань, а саме:

– обмеження ресурсів, використовуючи одним застосунком;

– ізоляція сервісів із метою неможливості впливу одного застосунку на іншого;

– повний контроль мережевих доступів та відкритих портів;

– можливість перенаправлення трафіку з приватного мережевого сегменту Kubernetes на балансувальник даних. При такій конфігурації достатньо всього лише одного зарезервованого публічного IP для всіх сервісів, а не декілька;

– налаштування HTTPS трафіку за допомогою вбудованого в Kubernetes Ingress сервісу, який перенаправляє весь HTTP трафік через SSL;

– можливість розділення доступів усередині Kubernetes за допомогою RBAC;

– плавне масштабування у разі необхідності додавання worker машин у AKS.

Як альтернативу можна було використати Docker, але функціонал Docker не дозволяє таку гнучкість, як Kubernetes. Порівняння Kubernetes з Docker наведено в таблиці 1.2.

Таблиця 1.2 – Порівняльна таблиця Kubernetes з Docker

Функціонал	Можливості Docker.	Можливості Kubernetes
Ізоляція ресурсів	Дозволяють.	Дозволяють.
Контроль мережевих доступів	Не дозволяють. Необхідно обмежувати доступ на створення контейнерів зайвим користувачам.	Дозволяють.
Розділення доступів за допомогою RBAC	Не дозволяють. Усі користувачі, які мають доступ до комп’ютеру з Docker, мають повний доступ до всіх компонент.	Дозволяють.

Продовження таблиці 1.2

Безпечне збереження чутливих даних	Не дозволяють. Необхідно обмежувати права на файл, де будуть зберігатися sensitive дані.	Дозволяють. Додатково можливо примонтувати secret-об’єкти у вигляді файлу всередині контейнеру.
Масштабування ресурсів	Не дозволяють. Необхідно виконувати вручну або через стороннє ПЗ – Docker Swarm.	Дозволяють. Необхідні дії можна виконати на сторінці Microsoft Azure або через Terraform.

Отже, порівнюючи між Docker та Kubernetes, варто зробити висновок, що більш оптимальний варіант є Kubernetes, який і буде використовуватися в цьому проекті.

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Даний проект залучає та функціонує із допоміжними ПЗ. Усе програмне забезпечення є некомерційне та з відкритим кодом у мережі. Нижче буде короткий опис кожного.

Варто розпочати з інструментів, які необхідні для розробки та розгортання програмного забезпечення. Наприклад, для цього проекту використовується IDE – Microsoft Visual Studio Code. Цей інструмент «легкий» у споживанні ресурсів; має можливість змінювати інтерфейс під особливості кожного користувача окремо; є безкоштовні модулі для взаємодії з різними застосунками, розроблені групою людей, які цікавляться вдосконаленням даного продукту. Як альтернативу можна розглянути Sublime Text. На противагу Visual Studio Code, Sublime Text не є настільки популярним, що впливає на розробку додаткових функцій, різних модулів тощо, хоча є теж безкоштовним.

Основними компонентами даного проекту є база даних, API сервіс, веб-застосунок, інструменти для зберігання логів, моніторингу логів та сповіщення про можливу недоступність компонент або мікроконтролерів.

Таблиця 1.3 – Порівняння бази даних Postgres, Microsoft SQL Server та Oracle

Функціонал	Postgres	Microsoft SQL Server	Oracle
Наявність процедурної мови програмування	Вбудована	Вбудована	Вбудована
Зручність розгортання та підтримки БД. Наявність частих оновлень	Може розгортатися за допомогою Helm маніфестів, що дуже зручно. Підтримка досить проста, бо є можливість керувати за допомогою extensions	Правильне технічне розгортання SQL Server є трудомістким процесом, адже некомерційна версія негнучка. Сама база даних виживається, бо на заміну їй йде інша версія – Azure SQL Database.	Правильне технічне розгортання Oracle є трудомістким процесом, адже некомерційна версія негнучка. Часті оновлення доступні в комерційній версії.

Продовження таблиці 1.3

Вартість	Повністю некомерційна	Комерційна версія бази даних досить дорога і може складати від $1000 і вище.	Комерційна версія бази даних досить дорога і може складати від $1000 і вище.
Міграція SQL з інших баз даних	Мова SQL подібна до інших баз даних, проте є несуттєві власні позначення, що не співпадають з іншими БД	Мова SQL подібна до інших баз даних, проте є власні позначення, що не співпадають з іншими БД	Мова SQL подібна до інших баз даних, проте є власні позначення, що не співпадають з іншими БД

Так як вартість проекту повинна бути доступною для кожного, хто буде користуватися ним, а також з часто оновлюваними компонентами, підходить лише некомерційна база даних типу Postgres.

Розглянувши сучасні мови програмування, було обрано декілька з них. Наразі, в даному проекті використовуються 3 основні мови програмування – C, Python, JavaScript.

Мовою програмування C написаний програмний код для мікроконтролера Arduino UNO та ESP8266. Наразі ця мова програмування є єдиним варіантом, адже альтернатив немає. Проте, говорячи про переваги цієї мови, можна віднести наступне:

– відноситься до низькорівневих мов програмування, що є перевагою в розділенні ресурсів та використання пам’яті;

– структуроване написання коду.

Для розробки веб-сервісу, з метою доступу до бази даних (GET, PUT та POST запити) використовується мова програмування Python. Була обрана ця мова програмування тому, що:

– модуль Flask, який використовується для API сервісів, є легким при утилізації ресурсів;

– швидке та зручне розгортання;

– доступні всі конектори, які потрібні для комунікації між базою даних та застосунками;

– є необхідний модуль для pushgateway, який використовується для забору даних з API менеджера та відправляється до Prometheus.

Для розробки веб-застосунків було обрано розробку на мові програмування HTML + JavaScript. Обираючи цей шлях програмування, можна досягти успіхів у:

– потужного кластеру Frontend + Backend;

– вирішенні питань безпеки, а саме забезпечення HTTP over SSL, використовуючи публічні сертифікати;

– можливості авторизації для WEB API за допомогою Bearer токену;

– модульність програмного забезпечення з шаблонами *.ejs*.

Що стосується збирання логів та їх графічного відображення, було взято за увагу Prometheus + Grafana. Ці обидва застосунки є унікальними в даній сфері розробки, адже має багато переваг, на противагу іншим аналогам, а саме:

– безкоштовність;

– потужність забирання логів з періодичністю;

– власна Time-series база даних, яка є зручною у використанні при розробці (звичайна реляційна база даних не дозволяє швидко відобразити дані, та й постійне балансування, індексування, кешування збільшує час звернення до неї);

– реалізована можливість додаткових push-повідомлень, які необхідні для моніторингу систем. У разі несправності компонент або

недоступності – автоматично генерується повідомлення, яке отримує користувач системи.

Отже, підбиваючи вищевказане, головною перевагою використаних інструментів є безкоштовність та зручність у використанні.

1.3.3 Аналіз відомих програмних продуктів

Наразі є декілька схожих проектів, пов’язаних з віддаленим контролем освітлення в місті. Розглянемо два з наявних, а саме: IOTCOMM та ZGSM. Обидві ці компанії родом з Китаю, що є очевидним, адже китайські технології на голову вище за наших наявних.

Порівняння аналогів наведено в таблиці 1.4.

Таблиця 1.4 – Порівняння аналогів

Функціонал	Хмарне програмне забезпечення для моніторингу освітлення міста	IOTCOMM	ZGSM	Пояснення
Розгортання та збереження компонент в хмарі або в наземному дата центрі	Даний проект розрахований для розгортання та збереження компонент в хмарі.	Даний проект розрахований для розгортання та збереження компонент в хмарі.	Даний проект розрахований для розгортання та збереження компонент в хмарі.	Оскільки дані знаходяться в хмарі, є інструменти для резервних копій та конфігурації НА кластеру

Продовження таблиці 1.4

Комунікація через мережу Інтернет	Комунікація відбувається через Wi-Fi модуль, який розрахований на 2,4 Ггц	Комунікація відбувається через Ethernet/GPRS/ 3G модулі	Комунікація відбувається через 5G модуль	Найкращим варіантом є комбінація технологій, адже у разі недоступності одного компоненту, можна використати інший резервний.
Реалізація застосунку моніторингу	Передбачається комбінація веб-застосунку та API менеджера, у тому числі зовнішні розробки	Розроблений власний веб-застосунок і десктоп версія	Розроблений власний веб-застосунок і десктоп версія	Найкращим варіантом є веб-застосунок, адже всі власники гаджетів мають можливість користуватися браузером. Як відмінна риса мати десктоп версію – це перевага, але не головна.

Продовження таблиці 1.4

Сповіщення	Передбачені	Передбачені	Передбачені	Сповіщення необхідні для того, щоб максимально швидко зреагувати на недоступність системи та налагодити її.
Собівартість	До $500 за хмарні сервіси + до $100 за 1 комплект датчиків.	Ціна стартує від $1000 за 1 комплект датчиків	Ціна стартує від $1000 за 1 комплект датчиків	Собівартість готових рішень на порядок вища за дипломний проект. Це можна пояснити якістю мікроконтролерів, наявних додаткових сервісів у хмарі, вартість робочої сили тощо.

1.4 Аналіз вимог до програмного забезпечення

На рисунку 1.1 наведено Use Case діаграма:

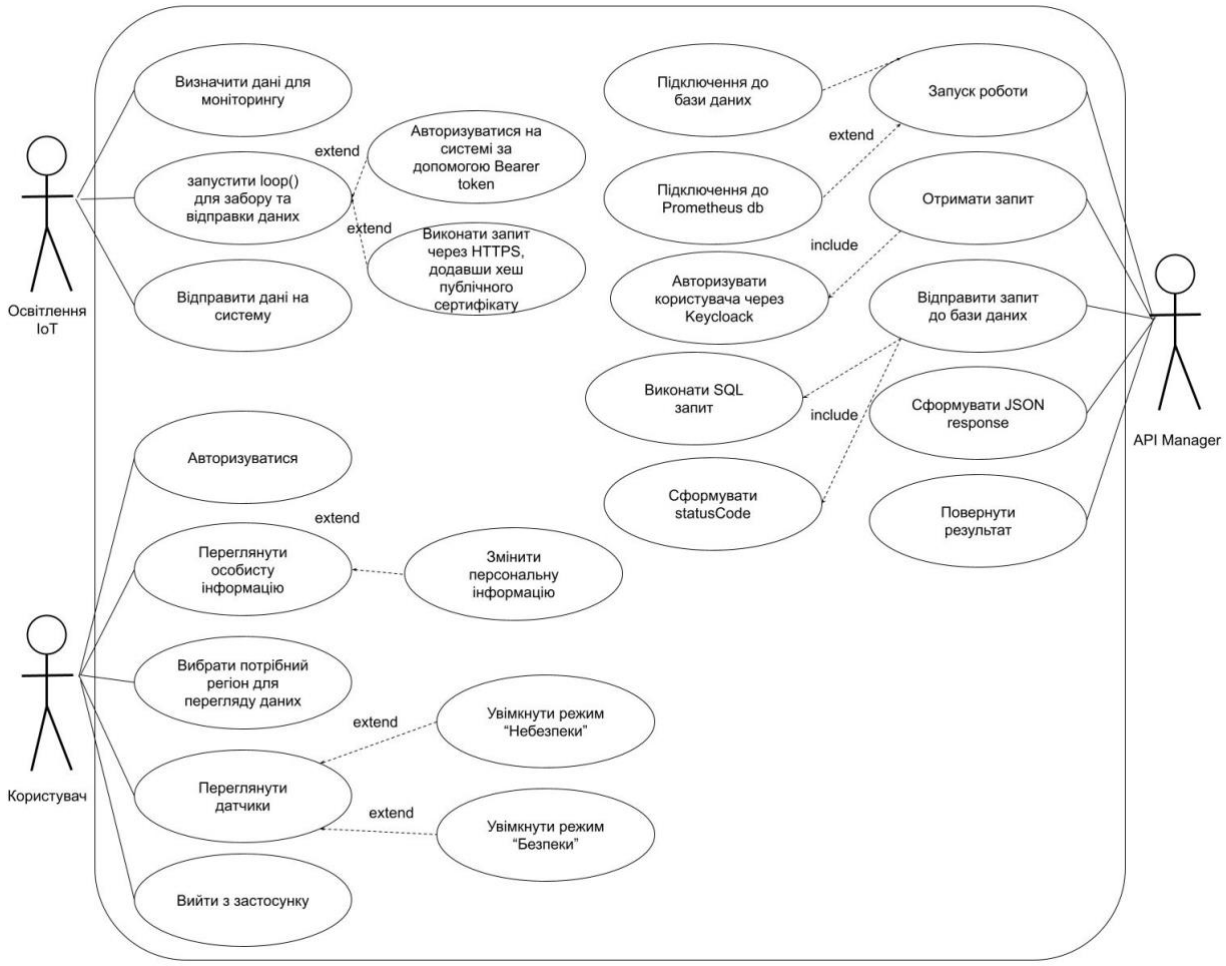


Рисунок 1.1 – Use case діаграма

Розглянемо модель вимог у вигляді багаторівневого списку:

№	Опис вимоги	Код	Пріоритет	Ризик
1	Можливість мікроконтролера отримувати та передавати дані до системи	FR-6	1	Високий
1.1	Можливість авторизації системами керування світлом до API менеджера	NFR-5	2	Середній

1.2	Передача даних через захищений протокол HTTPS	FR-5	1	Високий
1.3	Можливість масштабування обчислювальних ресурсів у разі потреби	NFR-1	3	Низький
2	Можливість користувача авторизації в системі «Withlight»	FR-1	1	Високий
2.1	Можливість розділення ролей в системі «Withlight»	FR-4	2	Середній
2.2	Можливість оновлення персональної інформації користувача	FR-3	3	Низький
3	Можливість керування освітлювальним приладом	FR-8	1	Високий
3.1	Автоматичне увімкнення/вимкнення оствілення	FR-7	1	Високий
3.2	Можливість переведення адміністраторим системи в режим «Небезпеки/Безпеки»	FR-7	3	Низький

В таблицях 1.5 - 1.15 наведені основні варіанти використання програмного забезпечення.

Таблиця 1.5 - Варіант використання UC-1

Use case name	Вхід користувача до системи
Use case ID	UC-01
Goals	Вхід користувача в систему
Actors	Зареєстрований користувач, знаходиться в базі даних

Продовження таблиці 1.5

Trigger	Користувач хоче зайти до системи з метою перегляду інформації
Pre-conditions	-
Flow of Events	Користувач знаходить на головну сторінку веб-застосунку. Перед ним з'являється два поля – для вводу логіну та пароля. У поле логін користувач вводить унікальний нікнейм, а в поле пароль – власний пароль. Після цього користувач натискає кнопку «Log in». У разі успішної авторизації, користувач переходить на головне меню веб-застосунку.
Extension	У випадку введення некоректних даних, при натисканні на кнопку «Log in», користувач отримає сповіщення та підсвітяться некоректні поля для вводу. Авторизація здійснена не буде.
Post-Condition	Авторизація користувача, перехід на головне меню.

Таблиця 1.6 - Варіант використання UC-2

Use case name	Перехід користувача на меню перегляду активних світлових приладів
Use case ID	UC-02
Goals	Переглянути всі активні світлові прилади
Actors	Зареєстрований користувач з правами «Reader» або «Admin» знаходиться в базі даних
Trigger	Користувач хоче переглянути всі активні світлові прилади
Pre-conditions	Користувач повинен бути авторизованим
Flow of Events	Користувач знаходиться на головній сторінці веб-застосунку. Він знаходить на головній сторінці кнопку «Light info». Користувач натискає на цю кнопку. У разі, якщо права дозволяють йому переглядати інформацію, він буде перенаправлений на сторінку з інформацією про світлові прилади.

Продовження таблиці 1.6

Extension	У випадку, якщо заявлених прав у користувача не буде, він не зможе переглянути дану інформацію.
Post-Condition	Перехід на меню з інформацією про світлові пристрої

Таблиця 1.7 - Варіант використання UC-3

Use case name	Перехід користувача на меню перегляду персональної інформації
Use case ID	UC-03
Goals	Переглянути власної персональної інформації
Actors	Зареєстрований користувач з правами «Reader» або «Admin» знаходиться в базі даних
Trigger	Користувач хоче переглянути власну персональну інформацію
Pre-conditions	Користувач повинен бути авторизованим
Flow of Events	Користувач знаходиться на головній сторінці веб-застосунку або на будь-якій іншій сторінці, після сторінки авторизації. Він знаходить кнопку «User info». Користувач натискає на цю кнопку. У разі, якщо права дозволяють йому переглядати інформацію, він буде перенаправлений на сторінку з персональною інформацією.
Extension	У випадку, якщо заявлених прав у користувача не буде, він не зможе переглянути дану інформацію.
Post-Condition	Перехід на меню з персональною інформацією

Таблиця 1.8 - Варіант використання UC-4

Use case name	Перехід користувача на панель керування адміністратора
Use case ID	UC-04

Продовження таблиці 1.8

Goals	Перехід на панель керування адміністратора
Actors	Зареєстрований користувач з правами «Admin» знаходиться в базі даних
Trigger	Користувач хоче перейти до панелі керування адміністратора
Pre-conditions	Користувач повинен бути авторизованим та мати права «Admin»
Flow of Events	Користувач знаходиться на головній сторінці веб-застосунку або на будь-якій іншій сторінці, після сторінки авторизації. Він знаходить кнопку «Admin site». Користувач натискає на цю кнопку. У разі, якщо права дозволяють йому переглядати інформацію, він буде перенаправлений на панель керування адміністратора
Extension	У випадку, якщо заявлених прав у користувача не буде, він не зможе переглянути дану інформацію та не матиме відповідної кнопки.
Post-Condition	Перехід на панель керування адміністратора

Таблиця 1.9 – Варіант використання UC-5

Use case name	Створення нового користувача
Use case ID	UC-05
Goals	Створити нового користувача в системі
Actors	Зареєстрований користувач з правами «Admin» знаходиться в базі даних
Trigger	Уповноважена особа хоче зареєструватися в системі
Pre-conditions	Адміністратор повинен бути авторизованим та мати права «Admin»

Продовження таблиці 1.9

Flow of Events	Користувач знаходиться на панелі керування адміністратора. На панелі екрана знаходиться кнопка «Admin site». Він повинен натиснути на цю кнопку. Перед ним справа з’являться поля для вводу персональних даних користувача. Адміністратор заповнює всі поля та натискає «Add new user»
Extension	У випадку, якщо адміністратор некоректно ввів дані про нового користувача, він отримає сповіщення та підсвічені поля, які є неправильно введеними.
Post-Condition	Сповіщення про створення нового користувача. Новий користувач з’явився в базі даних.

Таблиця 1.10 - Варіант використання UC-6

Use case name	Оновлення інформації у приладі освітлення
Use case ID	UC-06
Goals	Додати новий прилад
Actors	Зареєстрований користувач з правами «Admin» знаходиться в базі даних
Trigger	Є необхідність оновити дані у приладі освітлення
Pre-conditions	Адміністратор або інша уповноважена особа повинна бути авторизованим та мати права «Admin»
Flow of Events	Користувач знаходиться на панелі відображення інформації про освітлювальні прибори, яка зображена у вигляді таблиці. Для користувачів з правами «Admin» доступно редагувати поля про відображення кольору освітлення. Користувач повинен оновити поле та натиснути кнопку «ОК».
Extension	У випадку, якщо адміністратор некоректно ввів дані, він отримає сповіщення та підсвічені поля, яке є неправильно введеним.
Post-Condition	Сповіщення про оновлення інформації приладів. Нова інформація з’явилась в базі даних.

Таблиця 1.11 - Варіант використання UC-7

Use case name	Заведення в систему нового приладу освітлення
Use case ID	UC-07
Goals	Додати новий прилад
Actors	Зареєстрований користувач з правами «Admin»
Trigger	Є необхідність оновити дані у приладі освітлення
Pre-conditions	Адміністратор повинен бути авторизованим та мати права «Admin»
Flow of Events	Користувач знаходиться на панелі керування адміністратора. У лівому боці екрана знаходиться кнопка «Add new device». Він повинен натиснути на цю кнопку. Перед ним з'являться поле для вводу даних нового приладу Адміністратор заповнює всі поля та натискає «Create»
Extension	У випадку, якщо адміністратор некоректно ввів дані, він отримає сповіщення та підсвічені поля, яке є неправильно введеними.
Post-Condition	Сповіщення про успішне заведення освітлювального приладу. Нова інформація з'явилась в базі даних.

Таблиця 1.12 - Варіант використання UC-8

Use case name	Авторизація в системі "Grafana"
Use case ID	UC-08
Goals	Авторизуватися в системі "Grafana" для модифікації до графіків приладів
Actors	Уповноважена особа з правами «Admin»
Trigger	Уповноважена особа хоче авторизуватися в системі «Grafana»
Pre-conditions	Адміністратор повинен бути заведений та мати права «Admin» у системі "Grafana"

Продовження таблиці 1.12

Flow of Events	Користувач знаходиться на сторінці авторизації в застосунку. Перед ним з'являється два поля – для вводу логіну та пароля. У поле логін користувач вводить унікальний нікнейм, а в поле пароль – власний пароль. Після цього користувач натискає кнопку «Log in». У разі успішної авторизації, користувач переходить на головне меню веб-застосунку.
Extension	У випадку, якщо користувач некоректно ввів дані, він отримає сповіщення та підсвічені поля, яке є неправильно введеними.
Post-Condition	Сповіщення про успішну авторизацію користувача. Користувач перейшов до головного меню.

Таблиця 1.13 - Варіант використання UC-9

Use case name	Перегляд дашборду з поточним станом освітлювальних приладів
Use case ID	UC-9
Goals	Переглянути метрики з поточним станом освітлювальних приладів
Actors	Уповноважена особа з доступом до системи з правами «Reader» або «Admin»
Trigger	Уповноважена особа хоче переглянути метрики освітлювальних приладів
Pre-conditions	Користувач повинен бути авторизованих
Flow of Events	Користувач знаходиться головному меню. У лівому боці екрана знаходиться кнопка «Dashboards» Він повинен натиснути на цю кнопку. Перед ним з'являться перелік усіх панелей, які є в системі. Для збільшення масштабу в розрізі часу, користувач натискає на одну з панелей та протягує з ліва направо той проміжок часу, який цікавить користувача.
Extension	У випадку, якщо користувач не знаходить жодного дашборду, система не відображає нічого.
Post-Condition	Користувач побачить панелі з поточними метриками.

Таблиця 1.14 - Варіант використання UC-10

Use case name	Отримання даних приладом, використовуючи веб-сервіс
Use case ID	UC-10
Goals	Оновити поточні дані приладом, який звертається до веб-сервісу
Actors	Мікроконтролер ESP8266
Trigger	Запрограмована точка входу мікроконтролера до веб-сервісу
Pre-conditions	Система повинна бути увімкнена з доступом до мережі Інтернет.
Flow of Events	ESP8266 відправляє GET запит з поміткою власного deviceid, де deviceid – унікальний ідентифікатор приладу освітлення. Мікроконтролер надає Bearer токен, за допомогою якого відбувається авторизація. Після отримання даних з веб-сервісу, мікроконтролер отримує “responseCode”: “200”, і оброблює дані згідно з логікою
Extension	У випадку, якщо у токену закінчився термін дії, або якщо GET запит був сформований неправильно, то веб-сервіс поверне “errorCode” :”404” – NOT FOUND.
Post-Condition	Система освітлення оновить інформацію та у разі змін – світло та/або колір світла зміниться

Таблиця 1.15 - Варіант використання UC-11

Use case name	Надсилання даних приладом, використовуючи веб-сервіс
Use case ID	UC-11
Goals	Відправити поточний стан приладів, який звертається до веб-сервісу
Actors	Мікроконтролер ESP8266
Trigger	Запрограмована точка входу мікроконтролера до веб-сервісу
Pre-conditions	Система повинна бути увімкнена з доступом до мережі Інтернет.

Продовження таблиці 1.15

Flow of Events	ESP8266 відправляє PUT запит з поміткою власного deviceid, де deviceid – унікальний ідентифікатор приладу освітлення. Мікроконтролер надає Bearer токен, за допомогою якого відбувається авторизація. Після отримання даних з веб-сервісу, мікроконтролер отримує “responseCode”: “200”, і оброблює дані згідно з логікою
Extension	У випадку, якщо у токену закінчився термін дії, або якщо PUT запит був сформований неправильно, то веб-сервіс поверне “errorCode” :”404” NOT FOUND.
Post-Condition	База даних оновить інформацію та буде доступна для перегляду з веб-застосунку

1.4.1 Розроблення функціональних вимог

Дане програмне забезпечення комплексне. Є frontend частина, backend частина, API менеджер, комунікація мікроконтролера ESP8266 з API менеджером, передача даних до time-series бази даних, а також відображення даних системою «Grafana» з time-series бази даних. Усе це розділено на окремі модулі. Кожен із модулів має низку функцій. У таблицях 1.16 – 1.24 наведено опис функціональних вимог. Матриця трасування наведемо у таблиці 1.25.

Таблиця 1.16 – Функціональна вимога FR-1

Назва	Авторизація користувача
Опис	Система повинна надати можливість авторизації користувача шляхом введення унікально username та пароль до нього.

Таблиця 1.17 – Функціональна вимога FR-2

Назва	Створення користувача
Опис	Система повинна надати можливість створити користувача з допомогою адміністратора. Так як це критична система, то реєстрація не може відбуватися потенційним клієнтом – лише адміністратором.

Таблиця 1.18 – Функціональна вимога FR-3

Назва	Оновлення інформації користувача
Опис	Система повинна надати можливість оновити інформацію користувача. Оновлення інформації відбувається клієнтом.

Таблиця 1.19 – Функціональна вимога FR-4

Назва	Градація прав доступів до системи
Опис	Система повинна мати категорії користувачів – Reader та Admin. Це необхідно для того, щоб розмежовувати права та обов’язки кожного з користувачів.

Таблиця 1.20 – Функціональна вимога FR-5

Назва	Авторизація API менеджера
Опис	API менеджер повинен бути захищений від зловмисників та мати повну безпеку даних

Таблиця 1.21 – Функціональна вимога FR-6

Назва	Передача даних API менеджером
Опис	API сервіс повинен отримувати та передавати дані лише у форматі JSON. Статус коди повинні бути, додатково і в JSON повідомленнях у полі “errorCode”: 200/404/503/515 тощо

Таблиця 1.22 – Функціональна вимога FR-7

Назва	Переведення освітлювальних приладів у режим «небезпеки» та «безпеки»
Опис	У разі повітряної тривоги, адміністратор системи має змогу увімкнути режим «небезпеки», а у разі завершення повітряної тривоги, вимкнути режим «небезпеки» та увімкнути режим «безпеки»

Таблиця 1.23 – Функціональна вимога FR-8

Назва	Редагування інформації контролерів
Опис	Не всі поля можуть бути оновлені та змінені. Дані можуть бути змінені лише для кольору світла та години роботи світлових приладів

Таблиця 1.24 – Функціональна вимога FR-9

Назва	Перегляд світлових приладів в режимі «читання»
Опис	Забезпечити перегляд даних у режимі «читання» усім користувачам, без залежності до його ролі

Таблиця 1.25 – Матриця трасування вимог

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9
UC-1	+			+		+	+		
UC-2		+		+		+	+		+
UC-3				+		+	+		
UC-4				+		+	+		
UC-5				+		+	+		
UC-6			+	+		+	+		
UC-7				+		+	+	+	
UC-8	+	+							
UC-9				+					+
UC-10						+	+		
UC-11						+	+		

1.4.2 Розроблення нефункціональних вимог

Розглядаючи нефункціональні вимоги, потрібно звернути увагу на масштабованість системи, технічне обслуговування та надійність передачі даних. У наступних таблицях будуть визначені нефункціональні вимоги.

Таблиця 1.25 – Нуфункціональна вимога NFR-1

Назва	Масштабованість системи
Опис	Забезпечити масштабованість Kubernetes кластеру у разі нехватки об’єму пам’яті, CPU або дискового простору.

Таблиця 1.26 – Нуфункціональна вимога NFR-2

Назва	Автоматизація розгортання хмарних сервісів
Опис	Забезпечити автоматизоване розгортання хмарних сервісів, такі як: віртуальні машини, адресний простір, сховище даних за допомогою підходу IaC та Terraform.

Таблиця 1.27 – Нуфункціональна вимога NFR-3

Назва	Шифрування чутливих даних у сховищі
Опис	Забезпечити шифрування чутливих дані при передачі до сховища з метою унеможливлення витоку даних з персональною інформацією. Виконати за допомогою RSA приватного та публічного ключа.

Таблиця 1.28 – Нуфункціональна вимога NFR-4

Назва	Використання доменного імені
Опис	Забезпечити доступ до усіх сервісів виключно через доменне ім’я для забезпечення заміни ip адреси, конфігурацію ssl тунелю тощо за необхідністю

Таблиця 1.29 – Нуфункціональна вимога NFR-5

Назва	Авторизація до API менеджера за допомогою Bearer токену
Опис	Забезпечити авторизацію до API менеджеру за допомогою Bearer токену

1.5 Постановка задачі

Узагальнюючи все вищесказане, коротко опишемо систему, сервіси, дії тощо.

У даному проекті передбачається проведення моніторингу освітлювальних приборів міста. Розробити програмне забезпечення світлових пристроїв з мікроконтролерами типу Arduino UNO та Wi-Fi модуль ESP8266. За допомогою API менеджеру, MCU буде отримувати та відправляти актуальну інформацію, використовуючи реляційне сховище даних. У свою чергу, необхідно розробити time-series сховище даних для подальшої побудови графіків та метрик. У разі спрацювання тригеру по недоступності системи буде надсилатися повідомлення про помилку. Додатково до цих компонент необхідно розробити веб-застосунок, який виступає у ролі КМ. За його допомогою, користувач може переглядати інформацію про актуальний стан освітлювальних приладів.

Висновки до розділу

Отже, описуючи аналіз вимог до програмного забезпечення, можна зробити декілька висновків. По-перше, усе програмне забезпечення буде сконфігуровано та розгорнуто в хмарному просторі Microsoft Azure. Автоматизація розгортання та модифікація систем буде виконуватися за допомогою Terraform. По-друге, було обрано стороннє програмне забезпечення, яке буде використовуватися в цьому проекті, а саме: Postgres у якості реляційної бази даних; Prometheus – у якості time-series бази даних; Grafana – у якості побудови графіків з time-series бази даних. Усі ці

компоненти є некомерційними та з відкритим кодом, що дозволяє убезпечити себе від несанкціонованого доступу, а в разі можливих кібер-злочинів, сервіси забезпечені балансувальником даних з DNS іменем та SSL сертифікатами.

По-третє, зазначається, що буде створено веб-застосунок для конфігурації світлових приладів та веб-сервіс у ролі backend частини для обміну даними.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Дане програмне забезпечення потребує опису бізнес процесу. Наведемо BPMN моделі (рисунки 2.1 – 2.7).

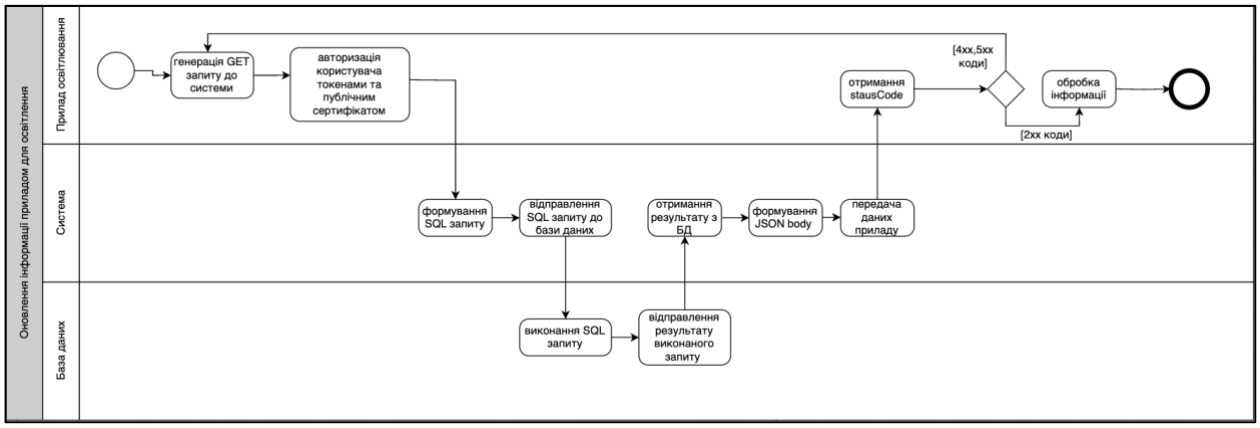


Рисунок 2.1 – Схема бізнес-процесу щодо оновлення інформації приладом для освітлення

Опис послідовності бізнес-процесу щодо оновлення інформації приладом для освітлення:

– мікроконтролером генерується GET запит;

– мікроконтролер додає токен до Заголовків запиту та відправляє до системи;

– система бере в роботу обробку запиту;

– система формує SQL запит та відправляє до бази даних;

– база даних виконує SQL скрипт та повертає результат до системи;

– система отримує результат виконаного SQL скрипту та формує JSON;

– система передає дані приладу та формує статус код, за яким прилад буде оброблювати отриману інформацію;

– прилад отримує JSON відповідь та статус код. У разі, якщо статус код не дорівнює “2xx” коду – прилад знову генерує запит до системи. У разі позитивного статусу – прилад оновлює інформацію у світловому приладі.

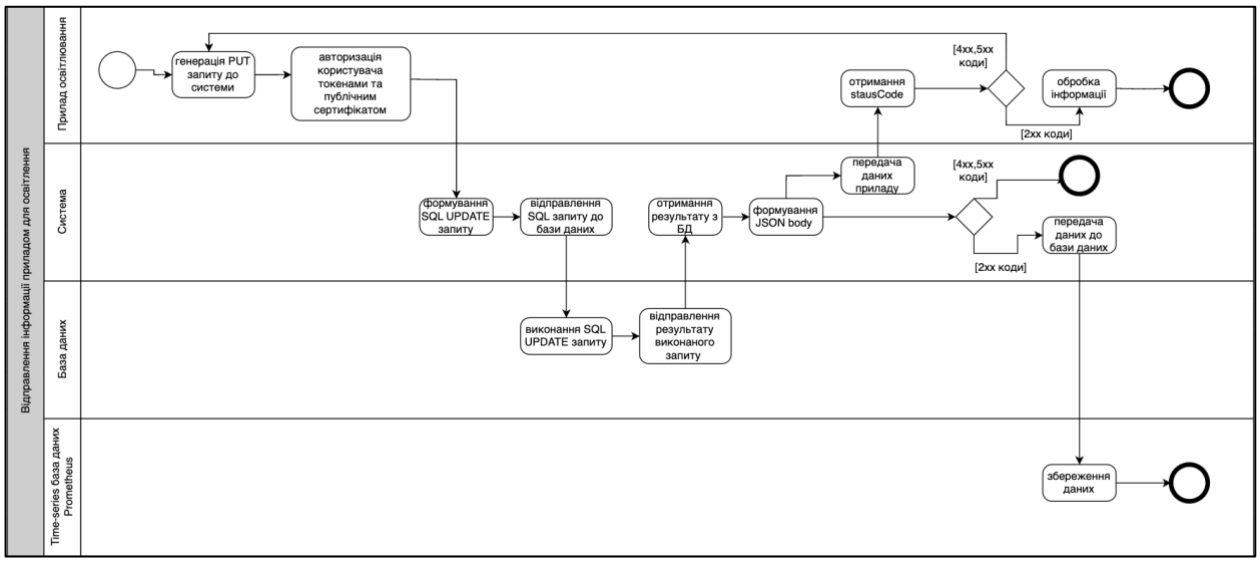


Рисунок 2.2 – Схема бізнес-процесу при відправленні інформації приладом для освітлення

Опис послідовності бізнес-процесу при відправленні інформації приладом для освітлення (рисунок 2.2):

– мікроконтролером генерується PUT запит;

– мікроконтролер додає токен до Заголовків запиту та відправляє до системи;

– система бере в роботу обробку запиту;

– система формує SQL запит на основі вхідних даних типу JSON та відправляє до бази даних;

– база даних виконує SQL скрипт та повертає результат до системи;

– система отримує результат виконаного SQL скрипту та формує JSON;

– система передає дані приладу та формує статус код, за яким прилад буде оброблювати отриману інформацію;

– у випадку коли статус код дорівнює “2xx” коду, дані відправляються до time-series бази даних за допомогою pushgateway.

– прилад отримує JSON відповідь та статус код. У разі, якщо статус код не дорівнює “2xx” коду – прилад знову генерує запит до системи. У разі позитивного статусу – прилад оновлює інформацію у світловому приладі.

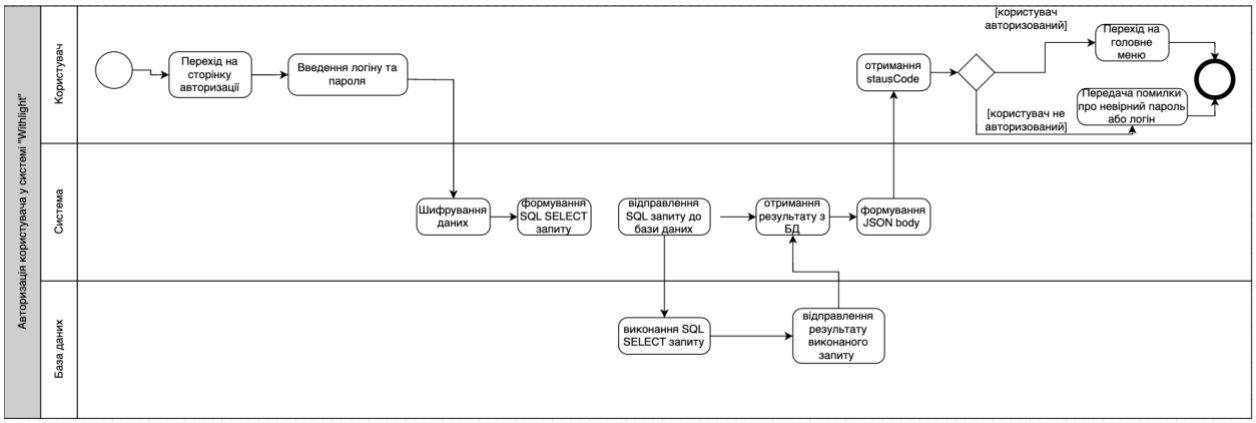


Рисунок 2.3 – Схема бізнес-процесу при авторизації користувача

Опис послідовності бізнес-процесу при авторизації користувача у системі Withlight (рисунок 2.3):

– користувач переходить на сторінку авторизації;

– користувач вводить логін та пароль;

– користувач натискає кнопку «Log in» і в той момент дані передаються до системи

– система бере в роботу обробку запиту;

– система шифрує дані;

– система формує SQL запит на основі вхідних даних типу JSON та відправляє до бази даних;

– база даних виконує SQL скрипт та повертає результат до системи;

– система отримує результат виконаного SQL скрипту та формує JSON;

– система передає дані та формує статус код;

– у випадку коли статус код дорівнює “2xx” коду, користувач успішно авторизований. В усіх інших випадках користувач отримує повідомлення про помилку. Користувачеві необхідно повторно авторизуватися.

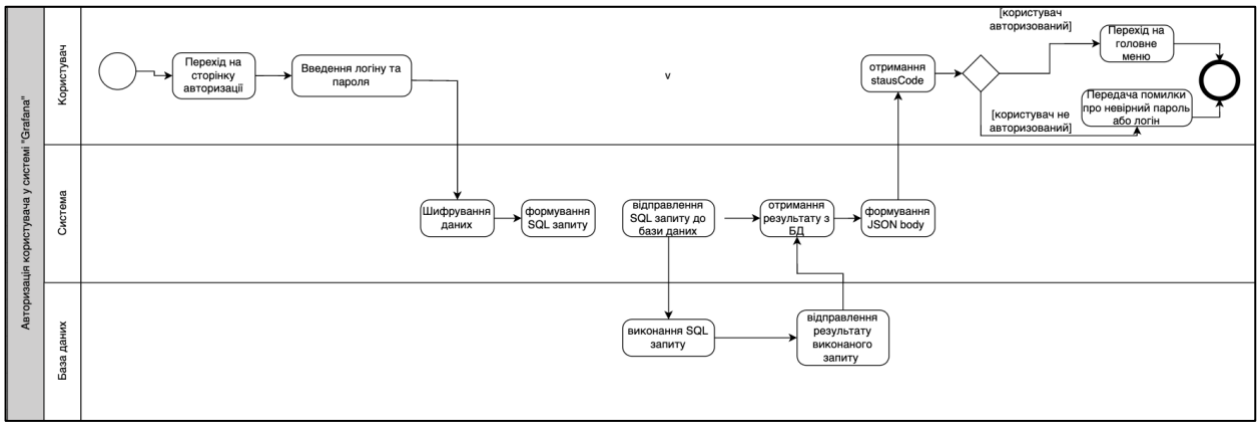


Рисунок 2.4 – Схема бізнес-процесу при авторизації адміністратора у системі “Grafana”

Опис послідовності бізнес-процесу при авторизації адміністратора у системі Grafana (рисунок 2.4):

– користувач переходить на сторінку авторизації;

– користувач вводить логін та пароль;

– користувач натискає кнопку «Log in» і в той момент дані передаються до системи

– система бере в роботу обробку запиту;

– система шифрує дані;

– система формує SQL запит на основі вхідних даних типу JSON та відправляє до бази даних;

– база даних виконує SQL скрипт та повертає результат до системи;

– система отримує результат виконаного SQL скрипту та формує JSON;

– система передає дані та формує статус код;

– у випадку коли статус код дорівнює “2xx” коду, користувач успішно авторизований. В усіх інших випадках користувач отримує повідомлення про помилку. Користувачеві необхідно повторно авторизуватися.

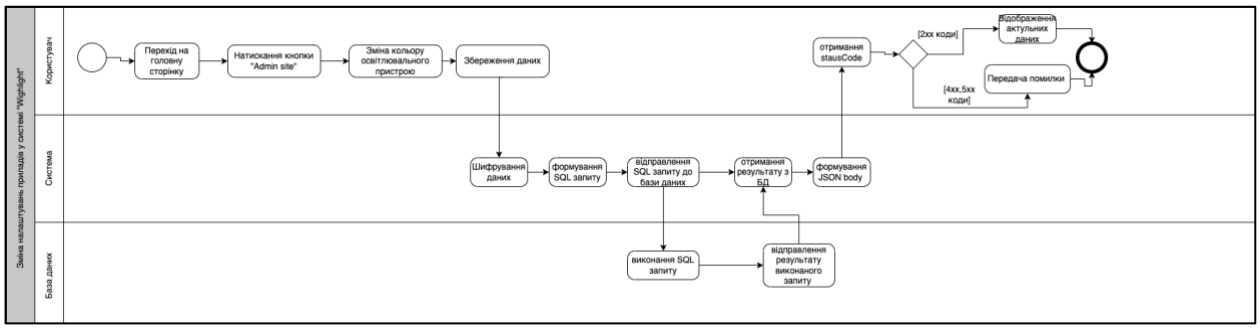


Рисунок 2.5 – Схема бізнес-процесу при зміні кольору освітлення приладу

Опис послідовності бізнес-процесу при зміні налаштувань приладів у системі (рисунок 2.5):

– користувач переходить на головну сторінку;

– користувач натискає кнопку “Admin site”;

– користувач змінює режим освітлення світильників, натискаючи на одну з двох кнопок: «Air raid alert» або «Air safety» та натискає кнопку “OK”. У цей час дані передаються до системи;

– система бере в роботу обробку запиту;

– система шифрує дані;

– система формує SQL запит на основі вхідних даних типу JSON та відправляє до бази даних;

– база даних виконує SQL скрипт та повертає результат до системи;

– система отримує результат виконаного SQL скрипту та формує JSON;

– система передає дані та формує статус код;

– у випадку коли статус код дорівнює “2xx” коду, налаштування успішно змінені. В усіх інших випадках користувач отримує повідомлення про помилку. Адміністратору необхідно повторити дії ще раз.

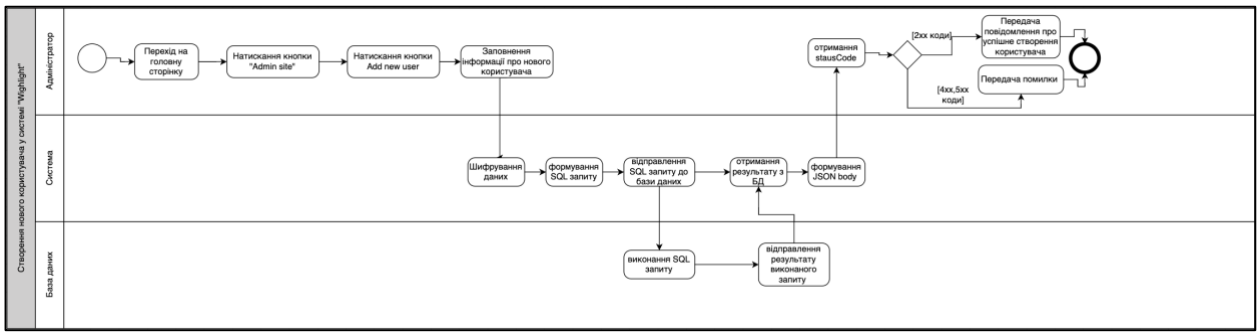


Рисунок 2.6 – Схема бізнес-процесу при додаванні нового користувача

Опис послідовності бізнес-процесу при додаванні нового користувача у системі “Withlight” (рисунок 2.6):

– адміністратор переходить на головну сторінку;

– адміністратор натискає кнопку “Admin site”;

– administrator переходить на вкладу;

– адміністратору відображається поля для введення всіх персональних даних нового користувача. Адміністратор заповнює поля та натискає кнопку «Add new user». У цей час дані передаються до системи;

– система шифрує дані;

– система формує SQL запит на основі вхідних даних типу JSON та відправляє до бази даних;

– база даних виконує SQL скрипт та повертає результат до системи;

– система отримує результат виконаного SQL скрипту та формує JSON;

– система передає дані та формує статус код;

– у випадку коли статус код дорівнює “2xx” коду, користувач успішно створений. В усіх інших випадках адміністратор отримує повідомлення про помилку. Адміністратору необхідно відкоригувати введені дані.

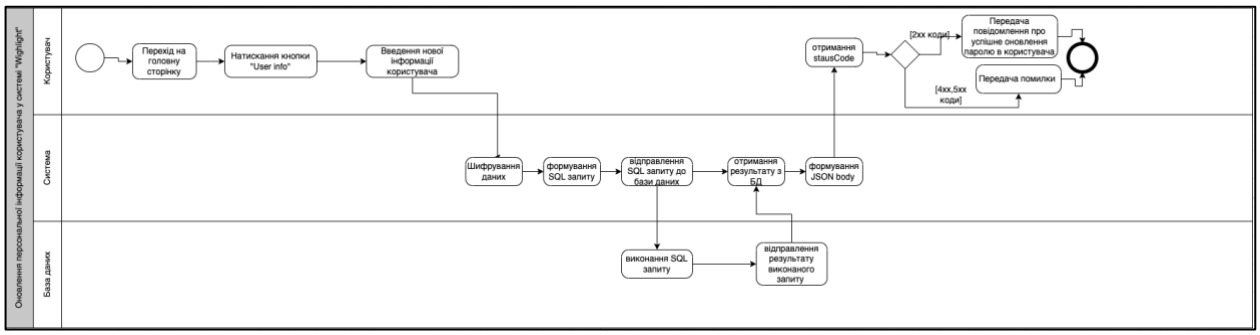


Рисунок 2.7 – Схема бізнес-процесу при оновлені інформації користувача

Опис послідовності бізнес-процесу при оновлені персональної інформації користувача (рисунок 2.7):

– користувач переходить на головну сторінку;

– користувач натискає кнопку “User info”;

– користувач переходить на вкладу для користувача та бачить персональну інформацію;

– користувач заповнює поля та натискає кнопку «Submit». У цей час дані передаються до системи;

– система шифрує дані;

– система формує SQL запит на основі вхідних даних типу JSON та відправляє до бази даних;

– база даних виконує SQL скрипт та повертає результат до системи;

– система отримує результат виконаного SQL скрипту та формує JSON;

– система передає дані та формує статус код;

– у випадку коли статус код дорівнює “2xx” коду, налаштування успішно змінені. В усіх інших випадках адміністратор отримує повідомлення про невідомого користувача або інше повідомлення. Адміністратору необхідно відкоригувати введені дані.

2.2 Архітектура програмного забезпечення

Опишемо архітектуру даного програмного забезпечення. На рисунку 2.8 зобразимо загальну архітектуру програмного забезпечення, з урахуванням хмарних компонентів.

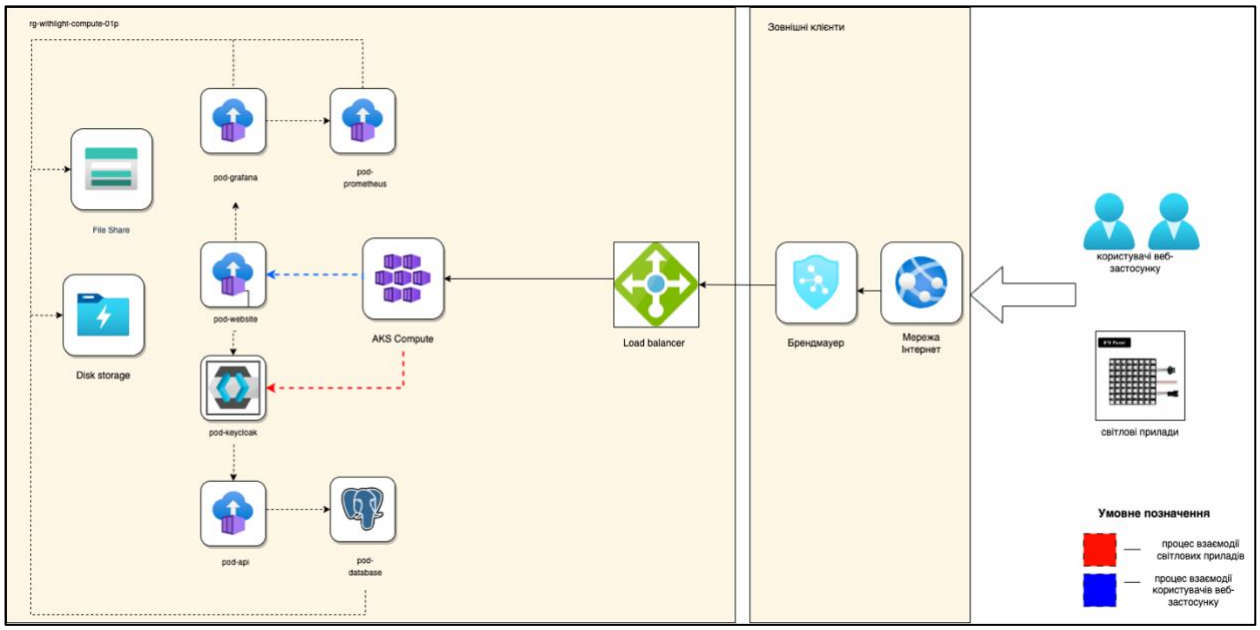


Рисунок 2.8 – Архітектура програмного забезпечення зі збереженням хмари, користувачів та освітлювальних приладів

Розіб’ємо архітектуру на три частини – сховища даних, API менеджер та веб-застосунок.

Розпочинаючи опис сховища даних, потрібно зазначити, що використовується RDBMS та нереляційна Time-series бази даних. Перша необхідна для того, щоб зберігати актуальну інформації освітлювальних приладів та дані про користувачів і друга база даних потрібна для забору логів і їх зберігання. У даному випадку зберігання логів та швидкий доступ до них буде дуже ускладнений, якщо для цього використовувати RDBMS.

API менеджер представлений у вигляді Python застосунку, ER сутностей якого відображений на рисунку 2.9.

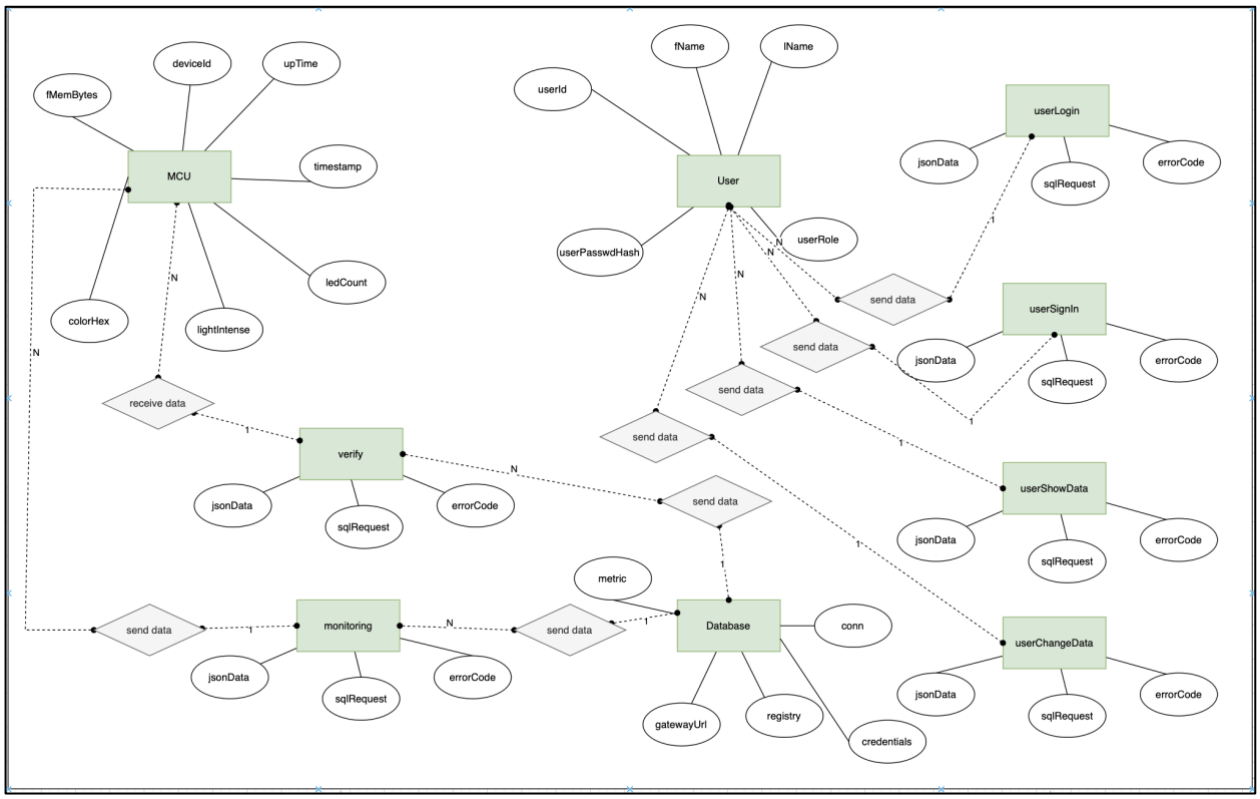


Рисунок 2.9 – ER діаграма сутностей Controller, User, Prometheus

Веб-застосунок написаний мовою програмування JavaScript з використанням фреймворку Express. У самому використовуються .ejs шаблони, які допомагають розділити кожну сторінку окремо. Завдяки цьому зв'язок між користувачем та сервісами є розділеним.

2.3 Конструювання програмного забезпечення

У якості реляційної бази даних використовується Postgres. База даних є основою роботи всіх процесі. На рисунку 2.10 зображена модель бази даних.

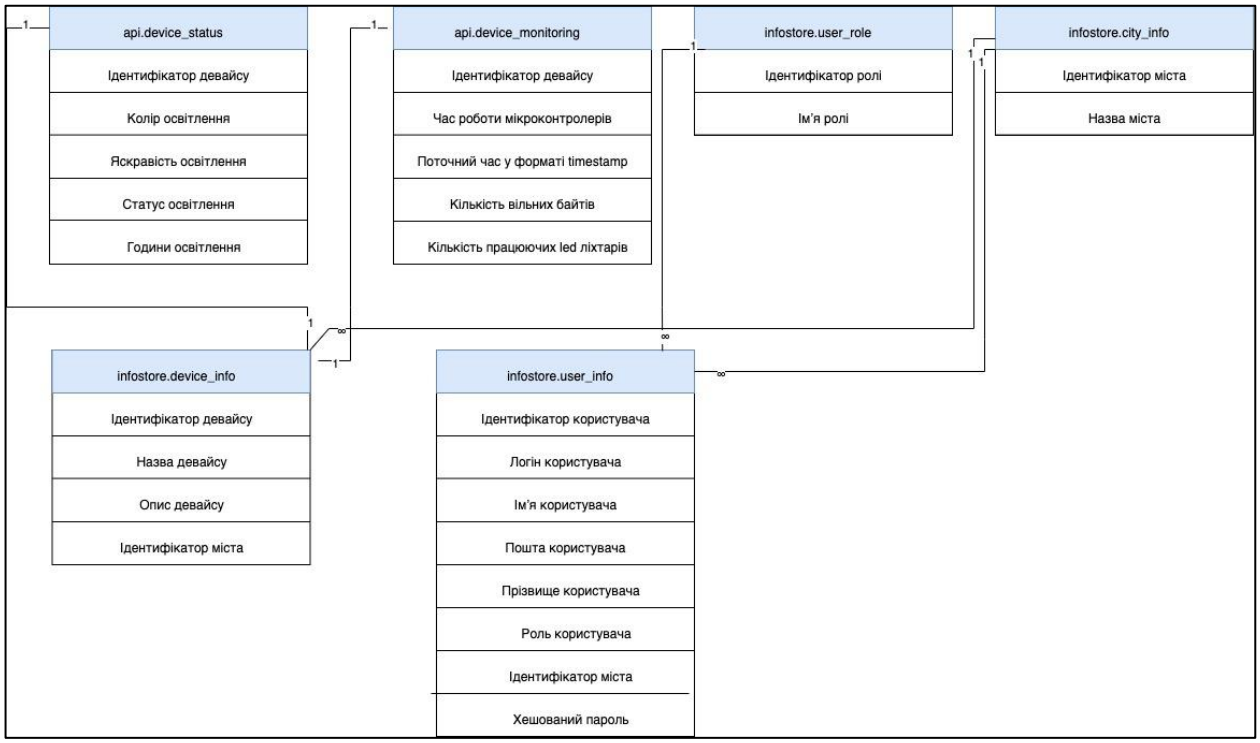


Рисунок 2.10 – Модель бази даних

Опис таблиці бази даних у таблицях 2.1 – 2.6. Модель бази даних наведено на рисунку 2.10.

Таблиця 2.1 – Опис таблиці api.device_status

Таблиця	Назва поля	Тип даних	Опис
device_status	device_id	serial	ідентифікаційний номер приладу освітлення
	color_hex	varchar (6)	ідентифікатор кольору у вигляді hex
	light_intense	varchar ()	яскравість кольору ліхтаря
	status	boolean	статус девайса – працює світло чи ні
	work_hour_from	time	час початку освітлення

Продовження таблиці 2.1

	work_hour_from	time	час до якого працює освітлення

Таблиця 2.2 – Опис таблиці api.device_monitoring

Таблиця	Назва поля	Тип даних	Опис
device_status	device_id	serial	ідентифікаційний номер приладу освітлення
	uptime	real	кількість секунд з останнього перезавантаження мікроконтролеру в секундах
	timestamp	timestamp	відображення часу в форматі timestamp для перевірки затримки мікроконтролера
	f_mem_bytes	int	кількість вільної пам’яті в байтах
	led_light_count	int	кількість увімкнених led ліхтарів

Таблиця 2.3 – Опис таблиці infostore.user_role

Таблиця	Назва поля	Тип даних	Опис
infostore.user_role	role_id	int	ідентифікатор ролі користувача
	rolename	text	назва ролі користувача

Таблиця 2.4 – Опис таблиці infostore.city_info

Таблиця	Назва поля	Тип даних	Опис
infostore.city_info	city_id	int	ідентифікатор міста
	cityname	text	назва міста

Таблиця 2.5 – Опис таблиці infostore.user_info

Таблиця	Назва поля	Тип даних	Опис
infostore.user_info	user_id	int	ідентифікатор користувача
	username	text	ім’я користувача

Продовження таблиці 2.5

	usermail	text	пошта користувача
	fname	text	ім’я користувача
	lname	text	прізвище користувача
	userrole	int	роль користувача
	citylocation	int	місто, за яким закріплений користувач
	passwdhash	text	хеш пароля користувача

Таблиця 2.6 – Опис таблиці infostore.device_info

Таблиця	Назва поля	Тип даних	Опис
infostore.device_info	device_id	int	ідентифікатор девайсу
	device_name	text	назва девайсу
	device_desc	text	опис девайсу
	citylocation	int	місцезнаходження девайсу

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.7.

Таблиця 2.7 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Microsoft Visual Studio Code	Основне середовище для розробки програмного забезпечення, автоматизації та розгортання сервісів.
2	Arduino IDE	Середовище для розробки, тестування та компілювання коду мовою програмування C для подальшого керування мікроконтролерами.
3	DataGrip	Середовище для розробки бази даних та графічний інтерфейс до бази даних
4	Kubernetes Lens	Програмне забезпечення у вигляді графічного інтерфейсу для Kubernetes сервісів, де можна переглянути всі елементи та сконфігурувати за допомогою Kubernetes manifest файлів або вручну.
5	Terminal	Термінал для комунікацій з віртуальними машинами, а також для тестування API через curl.
6	Prometheus	Некомерційне програмне забезпечення з відкритим вихідним кодом, яке представлене у вигляді веб-застосунку та нереляційної – time-series – бази даних з метою зберігання логів та подальшої обробки і побудови графіків.

Продовження таблиці 2.7

8	Grafana	Некомерційне програмне забезпечення з відкритим вихідним кодом, за допомогою якого будуються графіки з подальшим аналізом. Є безпосередній конектор між Grafana та Prometheus, що дозволяє моніторити у режимі реального часу.
9	Фреймворк Express	Програмна реалізація серверної частини для вебзастосунків для Node.js. За допомогою цього було реалізовано backend-частину програмного забезпечення
10	Бібліотека Crypto	Бібліотека з відкритим програмним кодом, для шифрування потенційно чутливих даних за допомогою RSA ключів
11	Бібліотека prometheus_client	Бібліотека з відкритим програмним кодом, відправлення логів до time-series бази даних Prometheus
12	Бібліотека psycopg2	Бібліотека з відкритим програмним кодом, яка призначена для взаємодії з реляційною базою даних Postgres
13	Бібліотека base64	Бібліотека з відкритим програмним кодом, яка призначена шифрування/дешифрування даних в base64
14	Бібліотека json	Бібліотека з відкритим програмним кодом, яка призначена автоформатування даних у форматі JSON
14	Бібліотека configparser	Бібліотека з відкритим програмним кодом, яка призначена для передачі чутливих даних із зовнішнього файлу безпосередньо до програмного забезпечення

Продовження таблиці 2.7

15	Бібліотека Flask	Бібліотека з відкритим програмним кодом, яка призначена для реалізації API менеджера у середовищі Python

2.4 Аналіз безпеки даних

Так як програмне забезпечення націлене на довготривалу розробку та вдосконалення функціоналу, гостро постає питання щодо встановлення методів захисту від витоку даних.

Так як частина програмного забезпечення є веб-застосунок, необхідно захистити персональні дані користувачів та чутливу інформацію, яка отримується та оброблюється. При розробці програмного забезпечення було враховано такі аспекти безпеки за стандартом GDPR:

– конфеденційність – веб-застосунок шифрує дані при передачі між системою та клієнтом, обмежує доступ до персональних даних шляхом розмежування прав;

– цілісність – веб-застосунок забезпечує цілісність передачі даних (використання SSL сертифікатів);

– доступність – веб-застосунок забезпечений резервним копіюванням у приватній хмарі Microsoft Azure.

У даному дипломному проєкті було застосовано декілька політик проти витоку даних. По-перше, при розробці бекенд веб-сервісу, за допомогою якого відбувається обмін даними між мікроконтролерами, застосунками тощо, був установлений додатковий захист, а саме забезпечення авторизації за допомогою Bearer токену. Цей процес було організовано за допомогою Keycloak.

Keycloack – некомерційний інструмент для автентифікації та менеджменту доступів [4]. За допомогою нього можна примусово змусити користувачів API застосунків авторизуватися, у протилежному випадку

доступності до даних не буде. Цей інструмент дозволяє створювати середовища, користувачів, термін дії токену тощо. Зручний веб інтерфейс дозволяє це зробити максимально зручно.

Іншим важливим аспектом є авторизація користувачів у самому веб-застосунку. Так як нам потрібно десь зберігати паролі, то необхідно проробити логіку шифрування та дешифрування даних. На допомогу приходить досвід роботи з приватними та публічними RSA ключами. Цей підхід дозволяє безпечно передавати дані через інтернет та зберігати в шифрованому форматі, не хвилюючись у разі неправомірного доступу до самої бази даних (у нашому випадку – це Postgres). Для того, щоб зашифрувати дані потрібен публічний ключ. Цим ключем ми шифруємо пароль і передаємо до веб-сервісу, який у свою чергу дешифрує пароль, який надав користувач з тим, що зберігається в базі даних. У разі, якщо пароль не сходиться – користувач отримує помилку і йому необхідно ввести пароль ще раз. Пам’ятаймо про те, що навіть якщо зловмисник матиме публічний ключ, він не зможе дешифрувати дані, а лише за допомогою приватного ключа. Так, як це сервіс захищений токеном, то неправомірного користування не передбачається.

Проте, навіть якщо передається нечутлива інформація, ми повинні бути впевнені, що дані не можуть бути перехоплені через Інтернет. Для цього потрібно захистити HTTP протокол SSL сертифікатами, щоб передавати дані через HTTPS.

Висновки до розділу

У даному розділі було проведено моделювання та аналіз програмного забезпечення. Під час розробки компонент програмного забезпечення було побудовано сховище даних, менеджер API та веб-застосунки разом з розробленими основними схемами бізнес-процесами та ER діаграмами сутностей.

Було визначено, що для даного проекту потрібно дві бази даних, одна з них реляційна, а інша – Time-series. Це означає, що залежності між сутностями

Database, User та MCU перетинаються в точці передачі логів з девайсів до сховища.

Також було встановлено, що є необхідність у використання додаткових утиліт і бібліотек, які використовуються в даному програмному забезпеченні для того, щоб авторизувати користувача, переглядати графіки та отримувати сповіщення.

Отже, під час написання даного розділу, було проведений комплексний аналіз структури проекту, розроблено та зображено сховище, процес ETL, а також основні бізнес кейси.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для даного програмного забезпечення було проведено тестування. Основними вимогами до тестування є: наявність основних перевірок для веб-застосунку; перевірка працездатності програмного забезпечення у хмарі, аналіз якості написаного програмного коду. Для того, щоб перевірити основні механізми роботи програмного забезпечення, було обрано мануальне тестування; для перевірки працездатності програмного забезпечення у хмарі було обрано Readiness & Liveness probes, а для аналізу якості написаного програмного коду було обрано аналізатор коду.

3.1 Аналіз якості ПЗ

Для аналізу якості ПЗ було використано Community версію Sonarqube. Дане програмне забезпечене націлене на виявлення багів, вразливостей, політики безпеки тощо. На рисунку 3.1 зображено результати роботи щодо написаного коду до даного проекту.

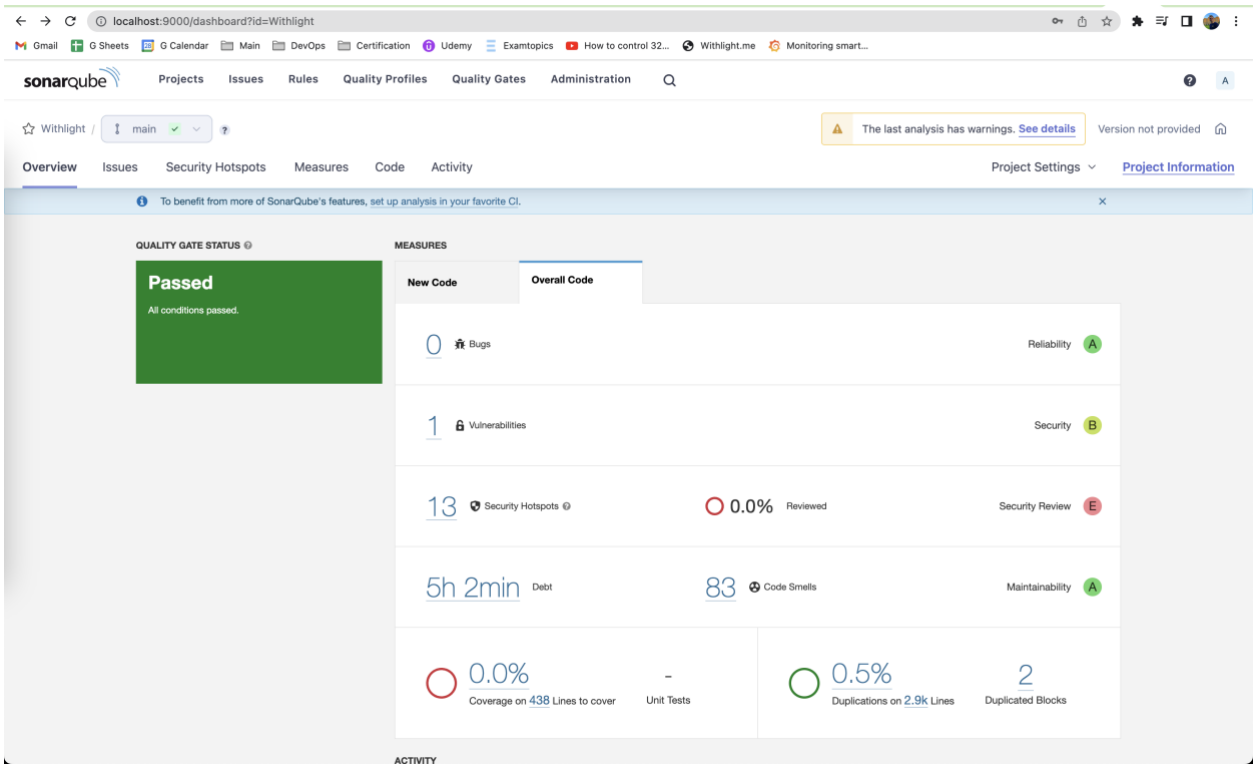


Рисунок 3.1 – Оцінка якості написаного коду програмного забезпечення

Проглядаючи результати даного тестування, можна дійти висновків, що перевірку пройдено:

– reliability – A;
– security – B;
– security review – E;
– maintainability – A.

Проте є певні недоліки, які необхідно в подальшому усунути задля покращення роботи ПЗ. Із основних недоліків написаного програмного забезпечення – це коментарі, які залишені в написаній програмі та є зайвими, а також не дотримання правил найменування об’єктів (рисунок 3.2).

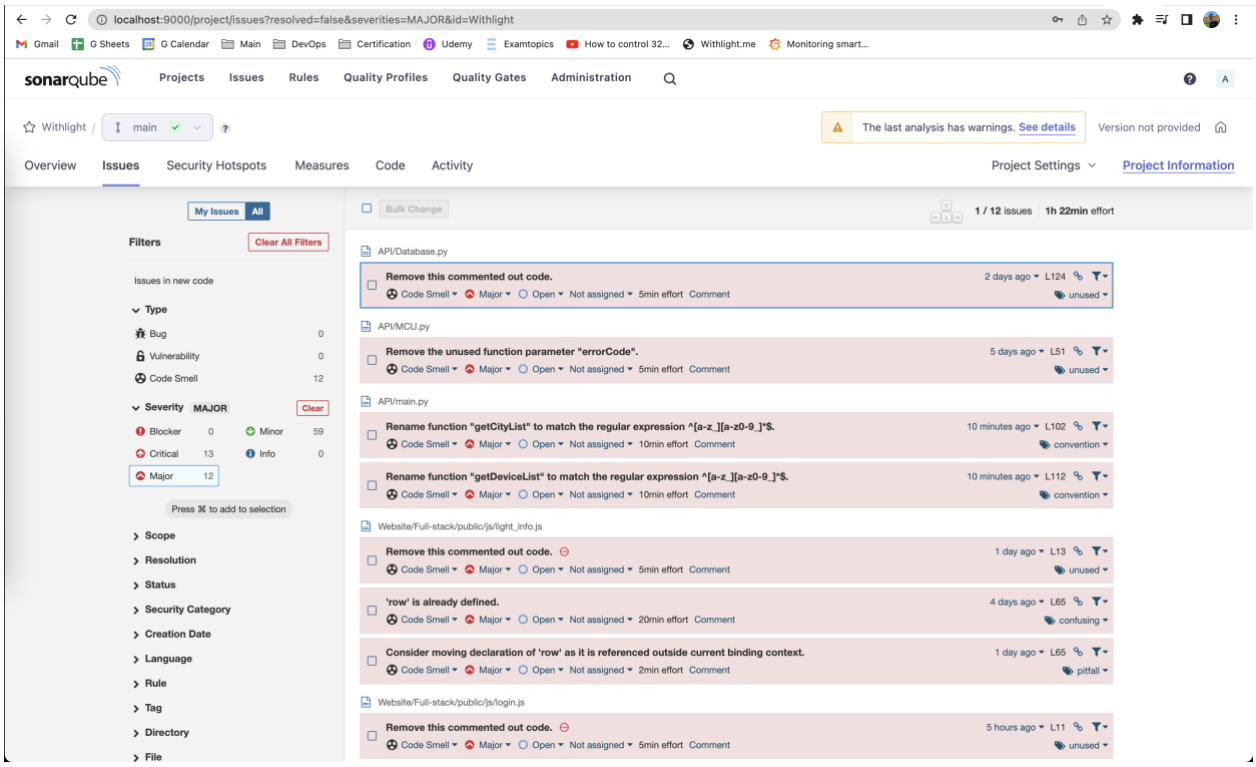


Рисунок 3.2 – Деталізація проблем, які виявив аналізатор коду

Якщо розглядати деталізацію Security Review з оцінкою E, то можна детальніше розглянути зауваження (рисунок 3.3):

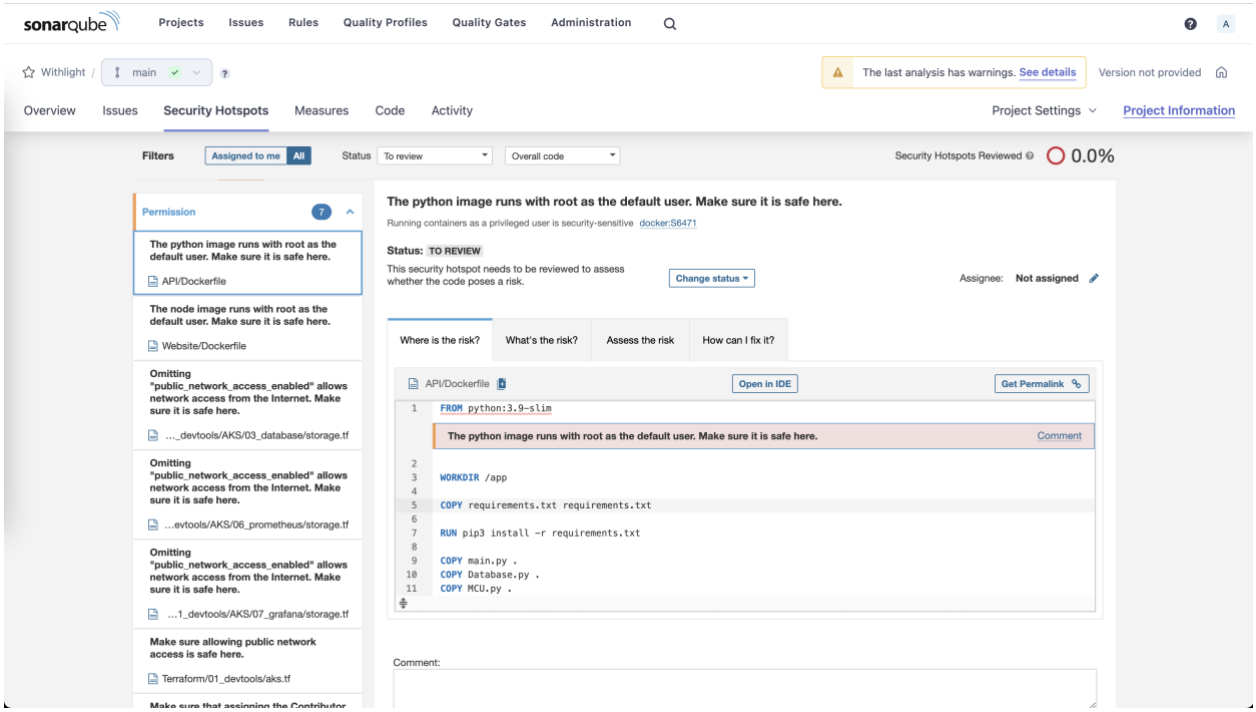


Рисунок 3.3 – Деталізація Security Review

Аналізатор коду розглядає ймовірність злому контейнерів в AKS як критичним і через це сильно понизив оцінку. Проте, проясняючи дану ситуацію, можна стверджувати що це практично нереально через додатковий захист системи, а саме:

– встановлення брандмауера поверх усіх застосунків, що унеможливлює прямого доступу до Kubernetes;

– встановлення балансувальника трафіку задля використання окремих DNS назв та публічної IP адреси, замість IP адреси AKS;

– додатковий захист зі сторони віртуальної машини, маючи доступ до якої виключно через SSH ключі;

– додатковий захист зі сторони AKS, де, для доступу до кластеру необхідній *kubeconfig* конфігураційний файл, який є виключно у системних адміністраторів системи.

Зважаючи на вищесказане, зауваження прийняті до уваги та будуть виправлені в подальшому.

3.2 Опис процесів тестування

Опишемо мануальне тестування програмного забезпечення, яке наведено в таблицях 3.1 – 3.20.

Таблиця 3.1 – Тест 1.1

Тест	Авторизація користувача
Модуль	Авторизація користувача
Номер тесту	1.1
Початковий стан системи	Сторінка для авторизації

Продовження таблиці 3.1

Вхідні данні	Логін, пароль
Опис проведення тесту	У поля для логіну та пароля вводяться: коректний логін користувача, який вже зареєстрований в даній системі та його особистий пароль. Під цими полями натискається кнопка «Увійти».

Таблиця 3.2 – Тест 1.2

Тест	Перегляд стану освітлювальних приборів
Модуль	Основне меню застосунку
Номер тесту	1.2
Початковий стан системи	Головна сторінка застосунку
Вхідні данні	-
Опис проведення тесту	На головному меню користувач натискає кнопку «Light info». Зліва від нього він повинен отримати список міст України. При натисканні на будь-яке місто України, користувач повинен побачити актуальну інформацію по місту. У разі, якщо в якомусь з міст освітлювальних приборів ще немає, користувач бачить справа пусте вікно.

Таблиця 3.3 – Тест 1.3

Тест	Перегляд панелей з інформацією по всім обчислювальним приладам
Модуль	Основне меню застосунку
Номер тесту	1.3
Початковий стан системи	Головна сторінка застосунку

Продовження таблиці 3.3

Вхідні данні	-
Опис проведення тесту	На головному меню користувач натискає кнопку «Dashboards». Користувач перейде на сторінку відображення панелей з усіма відображеними світловими приладами. У разі недоступності системи, користувач повинен отримати напис на панелі про помилку

Таблиця 3.4 – Тест 1.4

Тест	Оновлення персональної інформації
Модуль	Інформація про користувача
Номер тесту	1.4
Початковий стан системи	Авторизований користувач знаходиться на меню з персональною інформацією
Вхідні данні	Ім’я, прізвище користувача, email та пароль
Опис проведення тесту	Користувач, у разі бажання оновити персональну інформацію, змінює поля з ім’ям, прізвищем, email та паролем. Ім’я та прізвище користувача з забороною до вводу спецсимволів (окрім апострофу), пошта – лише по стандарту введення пошти, а пароль від 8 до 16 символів. Після вводу всіх даних, користувач натискає «Submit». Дані оновилися

Таблиця 3.5 – Тест 1.5

Тест	Переведення систем освітлення в режим «небезпеки»
Модуль	Модуль адміністратора
Номер тесту	1.5

Продовження таблиці 3.5

Початковий стан системи	Авторизований адміністратор знаходиться на меню адміністратора
Вхідні данні	Сигнал небезпеки
Опис проведення тесту	Уповноважена особа, знаходячись на сторінці для адміністраторів, натискає кнопку «Air raid alert». У цей момент їй приходить повідомлення про те, що успішно переведено освітлювальні прибори в режим «небезпеки». LED світильник почав світити червоним кольором

Таблиця 3.6 – Тест 1.6

Тест	Реєстрація нового користувача
Модуль	Модуль адміністратора
Номер тесту	1.6
Початковий стан системи	Авторизований адміністратор знаходиться на меню адміністратора
Вхідні данні	Персональні дані користувача
Опис проведення тесту	Уповноважена особа, знаходячись на сторінці для адміністраторів бачить справа від себе сторінку для реєстрації користувача. Адміністратор вводить всі персональні дані, а саме: логін, який містить виключно латиницю; пароль, який містить від 8 до 16 символів; прізвище та ім’я, яке не містить пробілів та спец символів, окрім тире. Адміністратор обирає роль користувача (Reader або Admin), місто, за яким закріплений користувач та натискає кнопку «Submit». Новий користувач повинен з’явитися в системі «Withlight»

Таблиця 3.7 – Тест 1.7

Тест	Переведення систем освітлення в режим «безпеки»
Модуль	Модуль адміністратора
Номер тесту	1.7
Початковий стан системи	Авторизований адміністратор знаходиться на меню адміністратора
Вхідні данні	Сигнал небезпеки
Опис проведення тесту	Уповноважена особа, знаходячись на сторінці для адміністраторів, натискає кнопку «Air safety». У цей момент їй приходить повідомлення про те, що успішно переведено освітлювальні прибори в режим «безпеки». LED світильник почав світити білим кольором

Таблиця 3.8 – Тест 1.8

Тест	Отримання сповіщення на пошту про недоступність системи
Модуль	Модуль сповіщення
Номер тесту	1.8
Початковий стан системи	Недоступний елемент освітлення
Вхідні данні	-
Опис проведення тесту	Уповноважена особа вимикає прилад освітлення. У цей система спостерігає недоступність елементу освітлення та чекає 5 хвилин допоки не відправить push повідомлення про недоступність системи. Через 5 хвилин на пошту приходить сповіщення всім користувачам, які закріплені за містом, а також адміністратори системи про недоступність

3.3 Опис контрольного прикладу

Варто розпочати зі сторінки авторизації системи “Withlight” (рисунок 3.4):

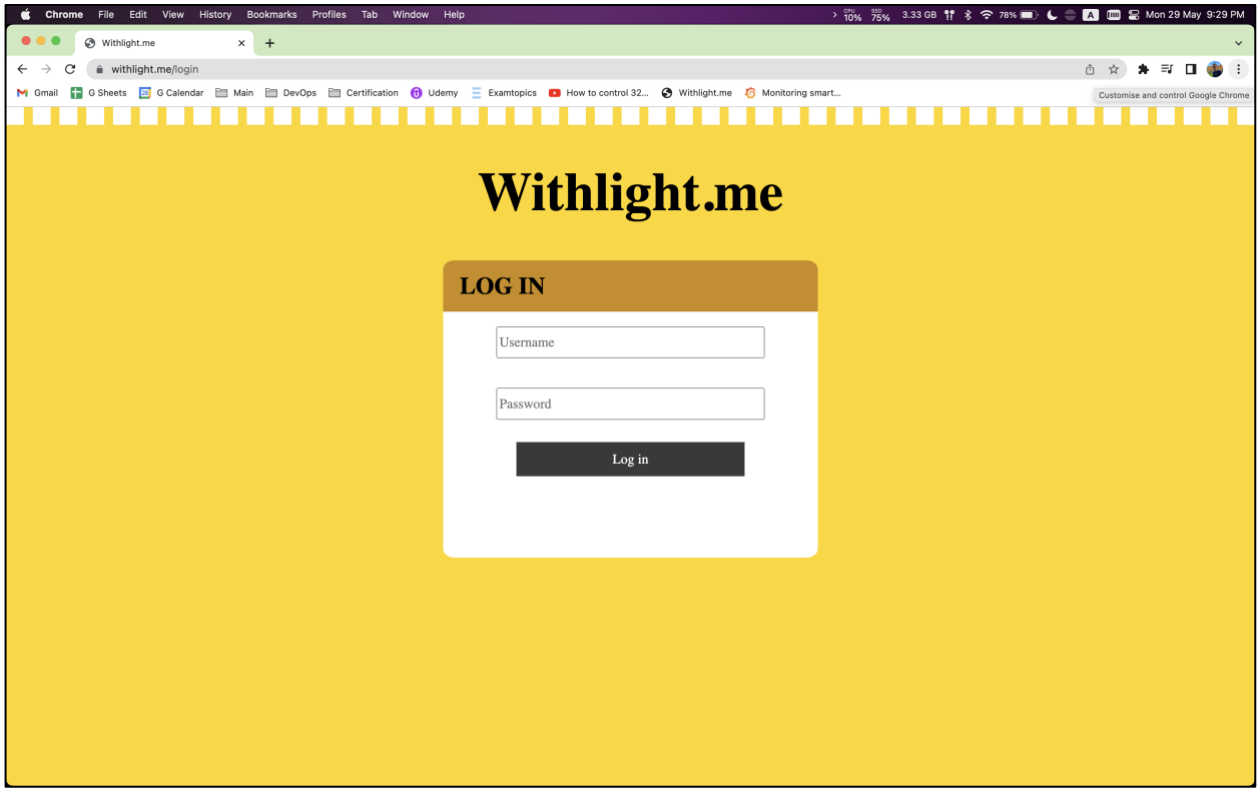


Рисунок 3.4 – Сторінка авторизації веб-застонку “Withlight”

Необхідно ввести логін та пароль до даної системи та авторизуватися (рисунок 3.5):

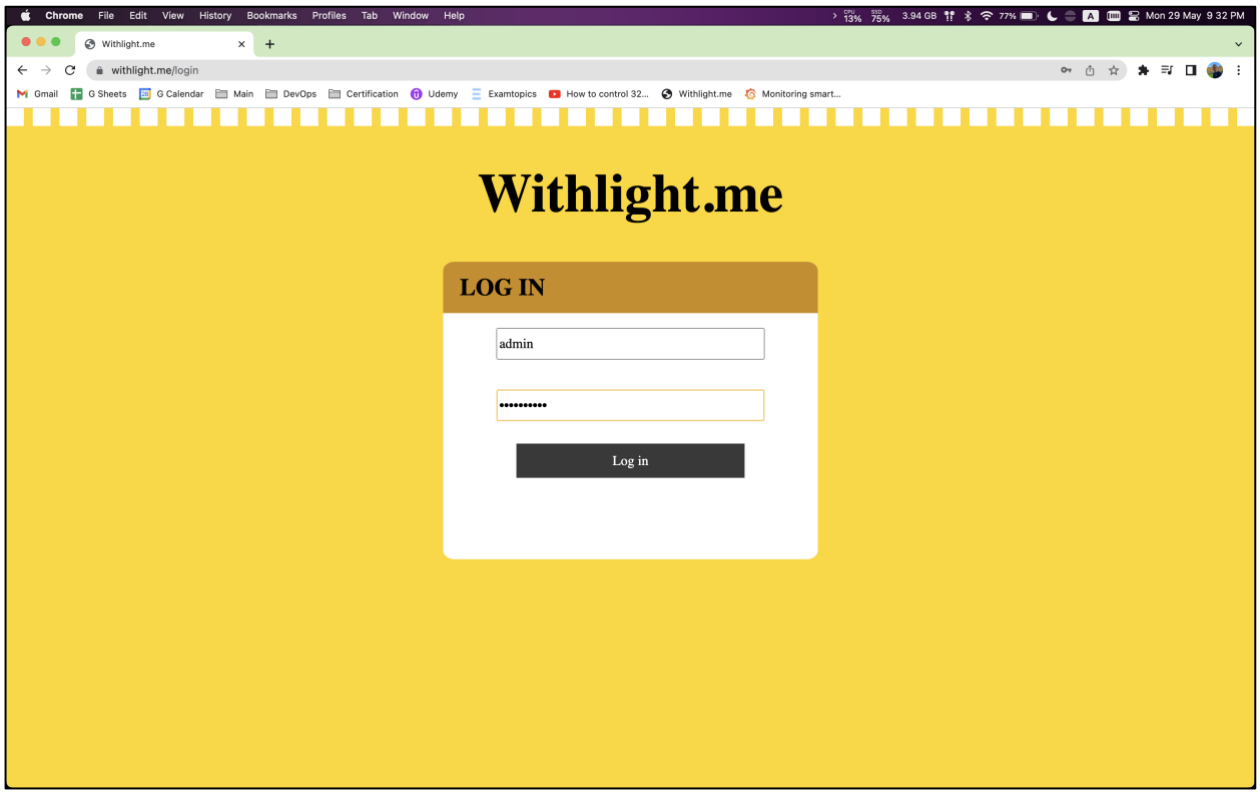


Рисунок 3.5 – Введені дані на сторінці авторизації

Паралельно з цим до освітлювального пристрою підключається елемент живлення для відображення актуальних метрик (рисунок 3.6):

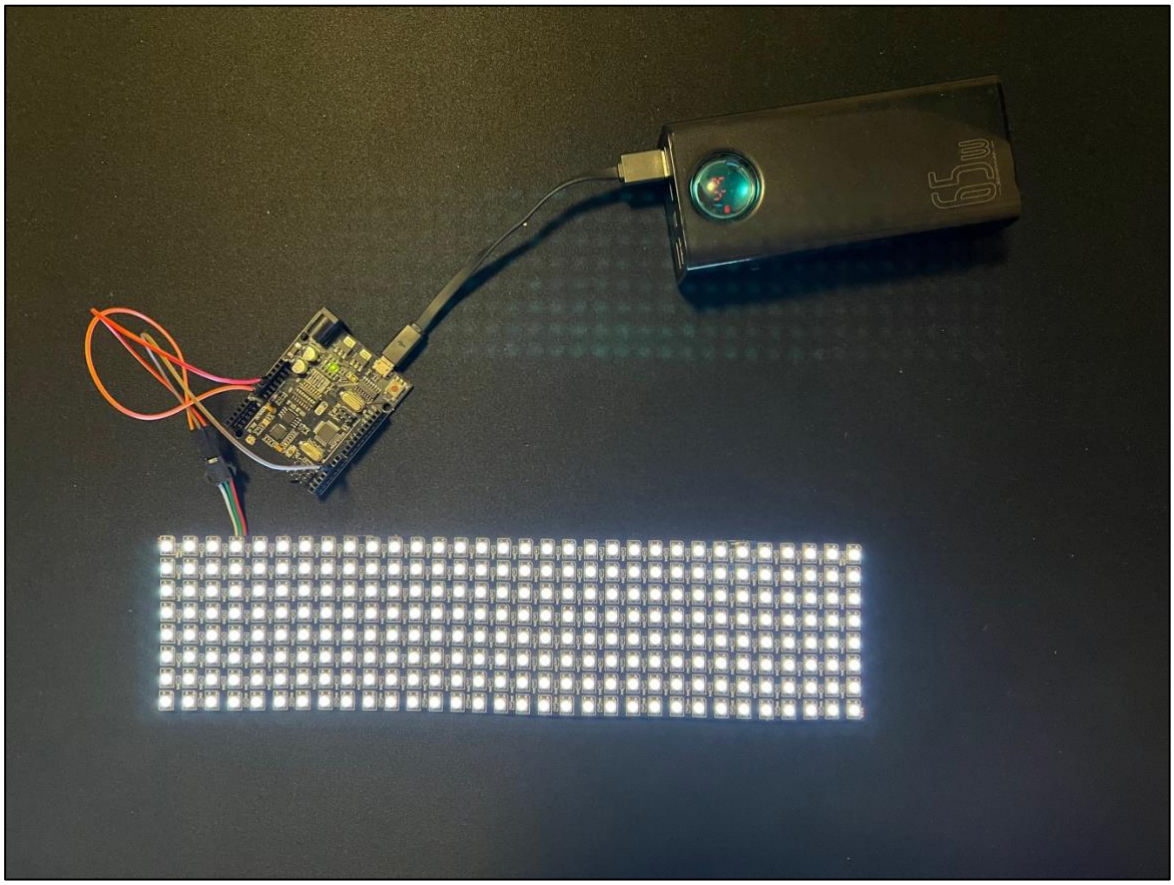


Рисунок 3.6 – Освітлювальний пристрій, увімкнений до елементу живлення (Power Bank)

Для того щоб оцінити стан пристрою у веб-застосунку, треба перейти у вкладку «Dashboard» (рисунок 3.7):

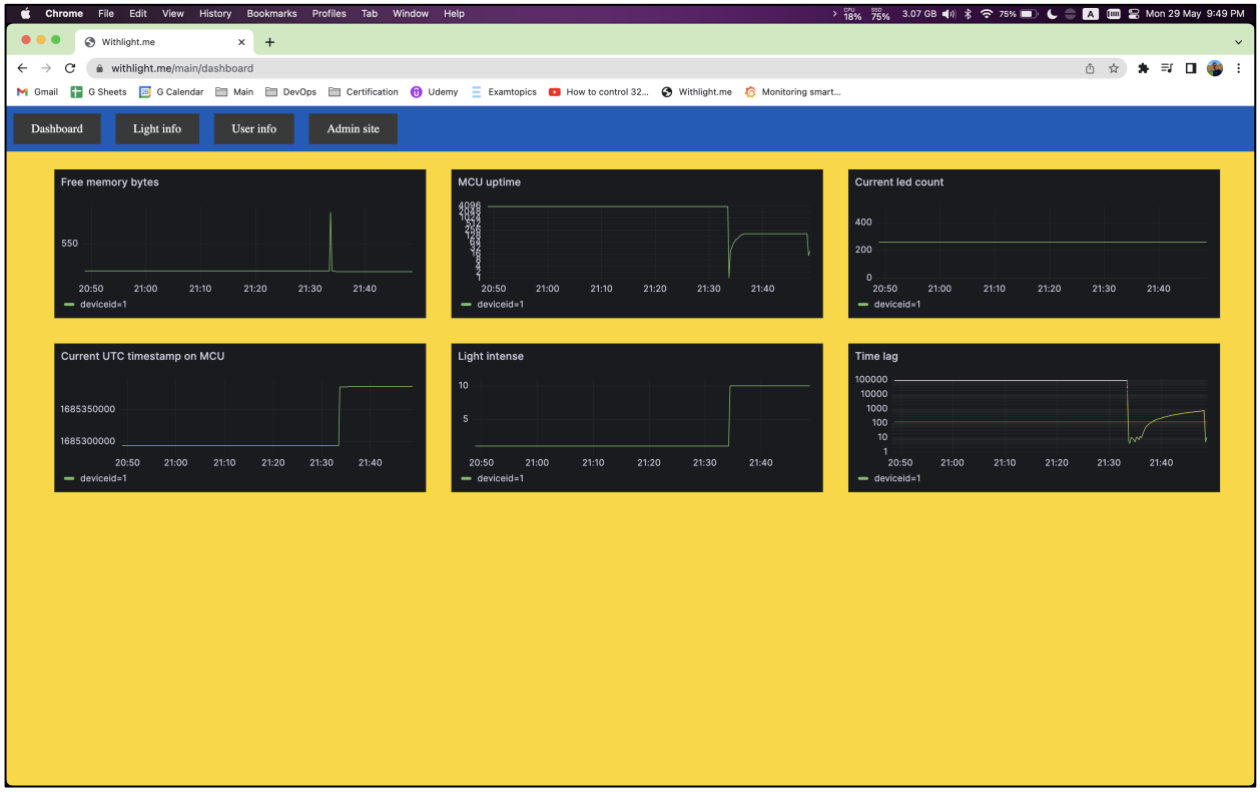


Рисунок 3.7 – Відображення метрик освітлювального пристрою

Бачимо, що пристрій працює справно. На даній панелі з метриками можна відслідкувати деякі параметри: кількість вільної пам’яті в Arduino UNO, час роботи мікроконтролера, кількість увімкнених led ліхтарів тощо. Основним із усіх метрик є time lag – це по суті відображення часової затримки мікроконтролера.

Перейдемо до інформації про світлові пристрої в розрізі міст. Натиснувши кнопку «Light info» можна побачити сторінку з інформацією про світлові пристрої в розрізі міст (рисунок 3.8):

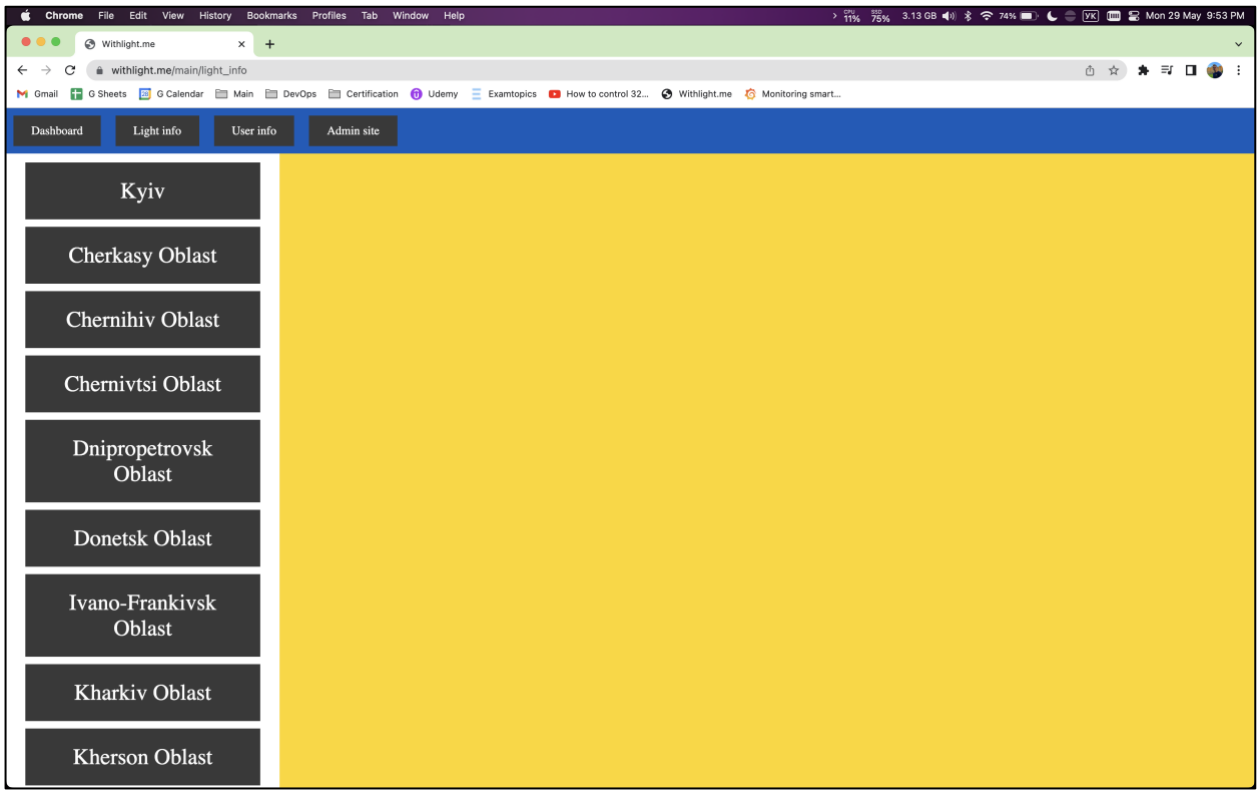


Рисунок 3.8 – Перелік міст та поле для відображення інформації в розрізі

Даний елемент освітлення сконфігурований та записаний як той, що знаходиться в Києві, тому натискаємо на місто Київ (рисунок 3.9):

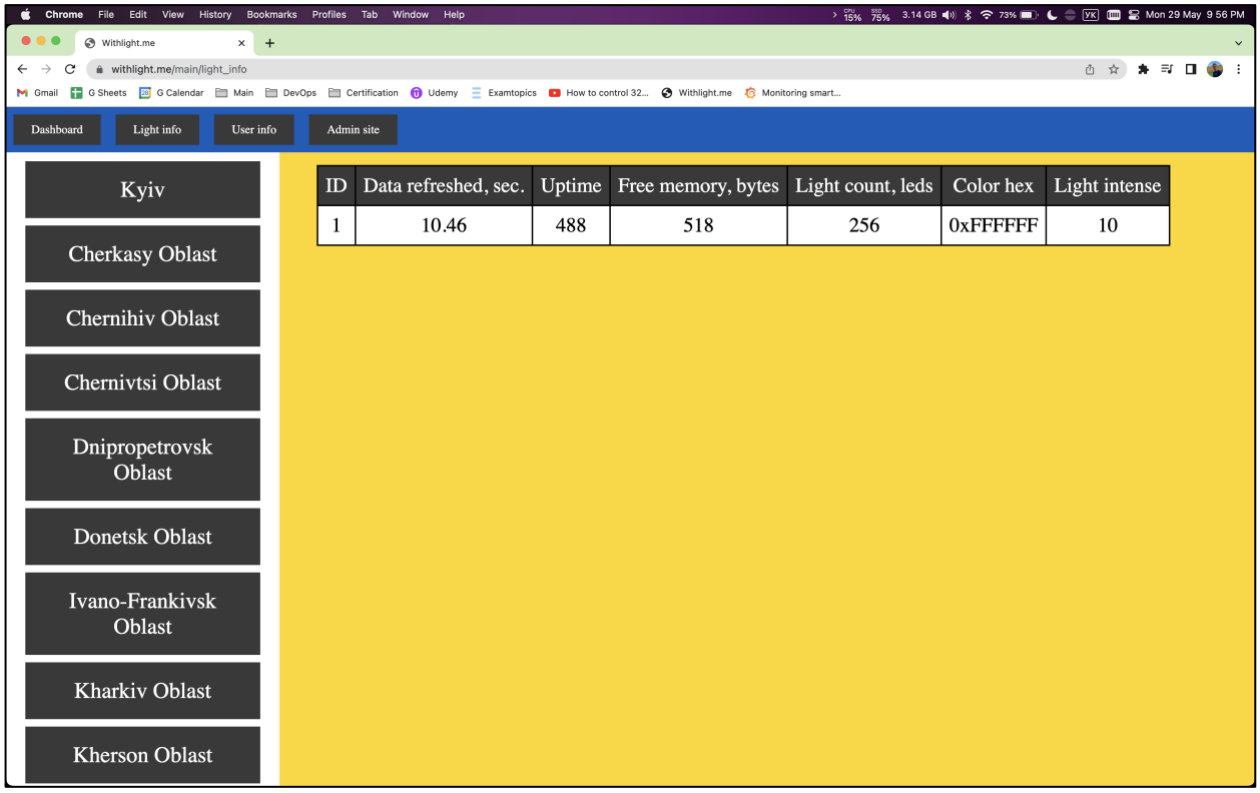


Рисунок 3.9 – Відображення останньої свіжої інформації про елемент освітлення

Як бачимо – основна інформація про освітлювальний пристрій присутня. Тут можна переглянути яскравість освітлювального пристрою, колір, яким зараз світиться пристрій тощо.

Перейдемо до персональної інформації користувача та змінимо власне ім’я та прізвище. Для цього натискаємо на кнопку «User info» (рисунок 3.10):

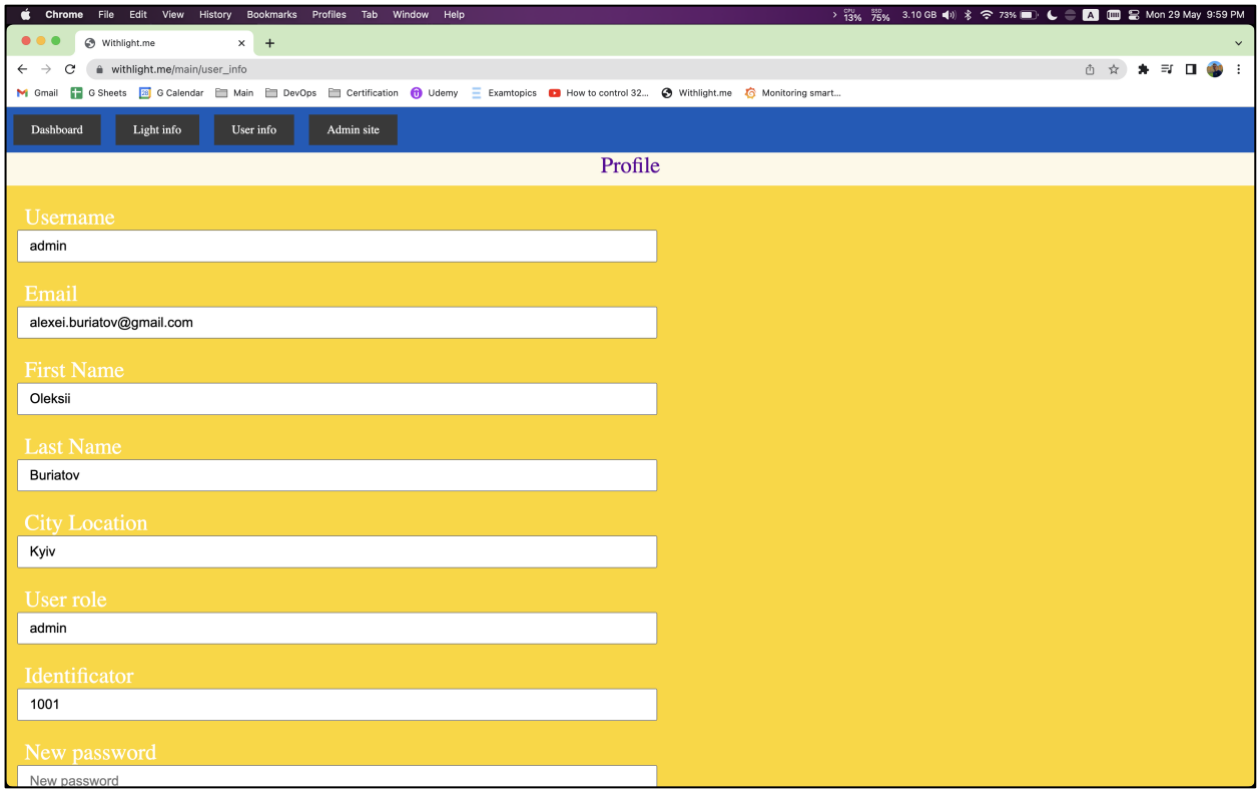


Рисунок 3.10 – Персональна інформація користувача

На цьому екрані зображена вся персональна інформація користувача та поля, в яких можна змінити інформацію. Не всі поля доступні для редагування (як-от username, city location та id змінити не можна, а власне ім’я, прізвище, пошту, та пароль можна). Змінимо власне прізвище, ім’я та пароль на інше (рисунок 3.11):

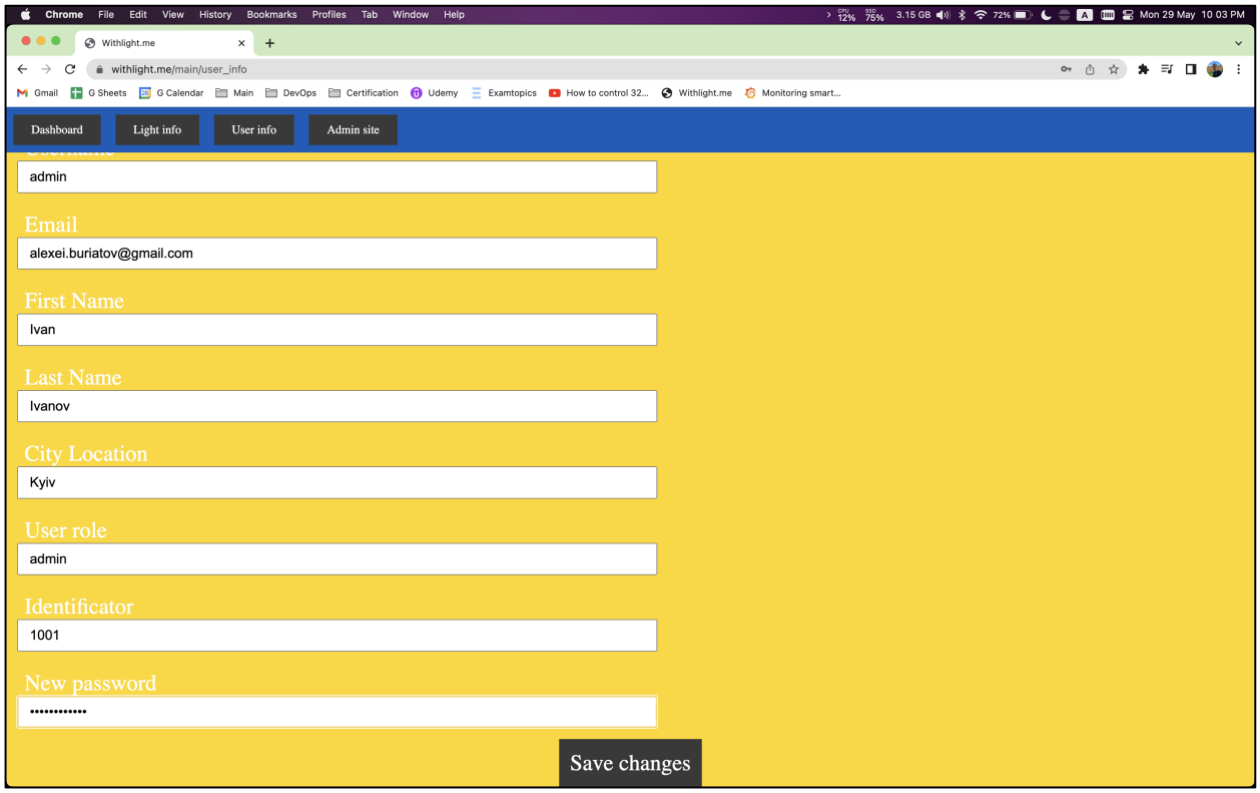


Рисунок 3.11 – Оновлення персональних даних користувача

Після вводу даних натискаємо «Save changes» та перевіряємо чи змінилися дані. Оновлюємо сторінку (рисунок 3.12):

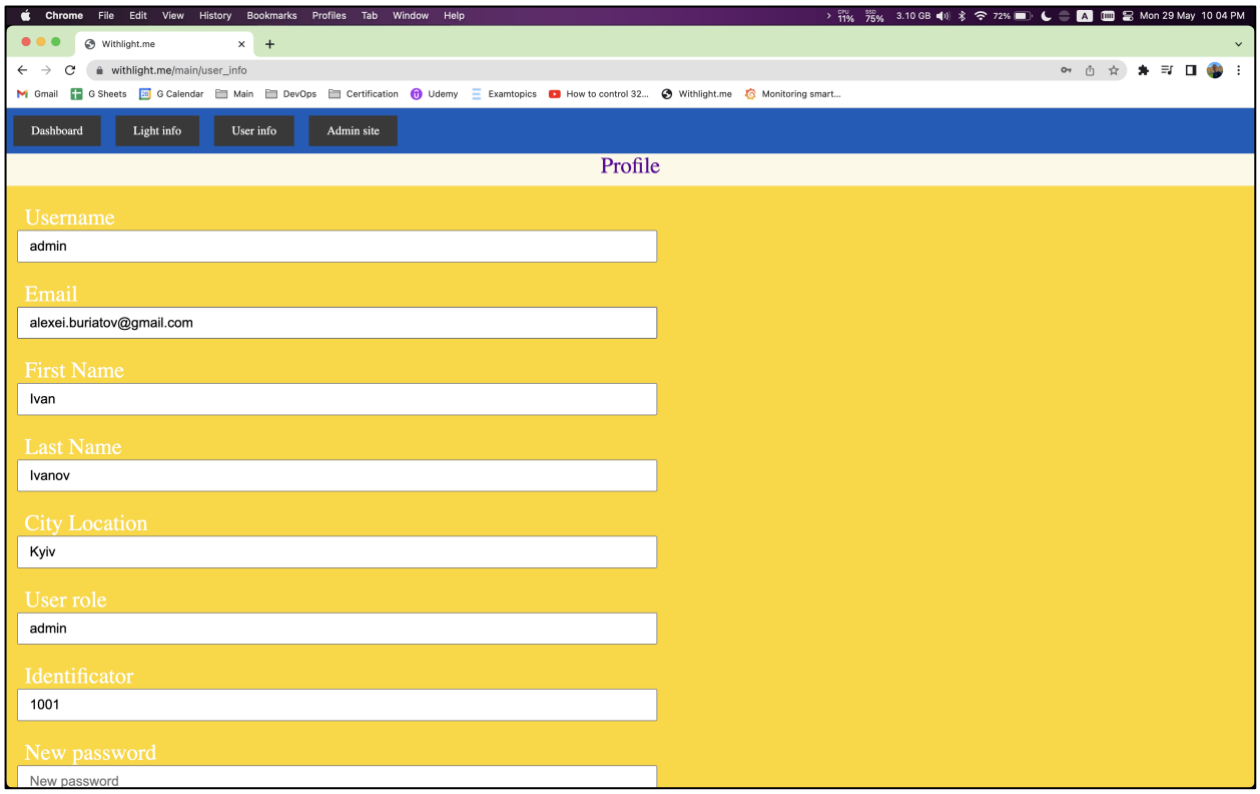


Рисунок 3.12 – Перевірка оновлення даних

Дані дійсно змінилися, тому все добре.

Спробуймо перевести систему освітлення в режим «небезпеки». Для цього натискаємо «Admin site» та переходимо до панелі керування адміністратора (рисунок 3.13):

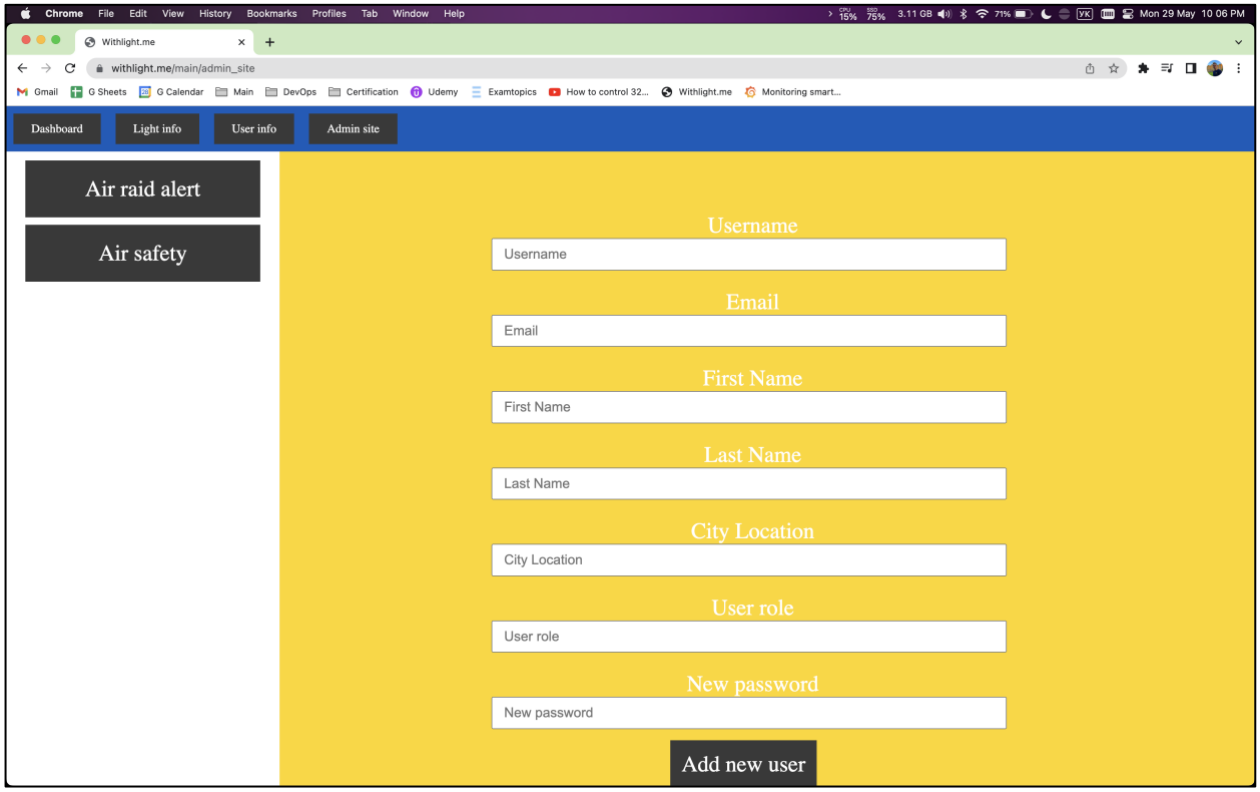


Рисунок 3.13– Панель керування адміністратора

Дана панель несе декілька функцій – переведення в режим «небезпеки» та режим «безпеки» освітлювальні прилади, а також створення нового користувача.

Для того, щоб перевести в режим «небезпеки» - натискаємо «Air raid alert». Отримуємо сповіщення (рисунок 3.14):

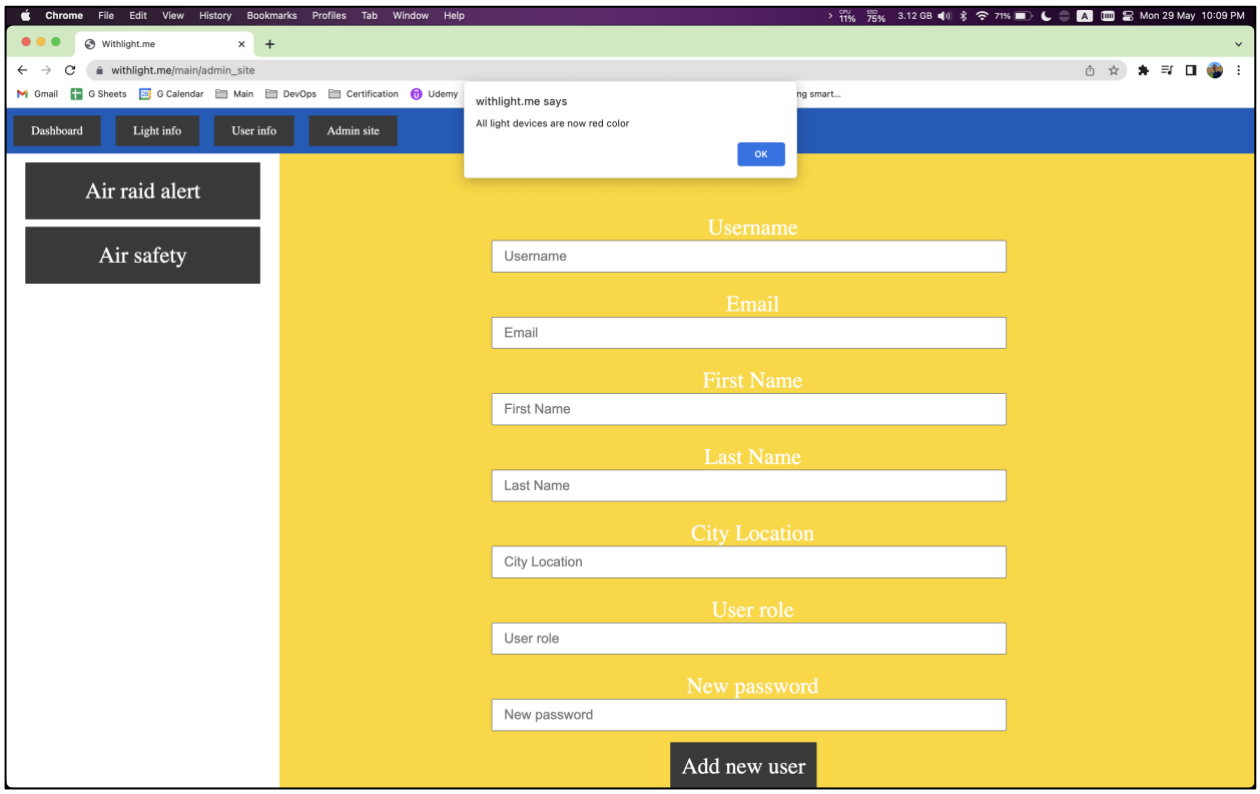


Рисунок 3.14 – Сповіщення про увімкнення режиму «небезпеки»

Через 5-10 секунд наш освітлювальний прилад змінив колір з білого на червоний (рисунок 3.15):

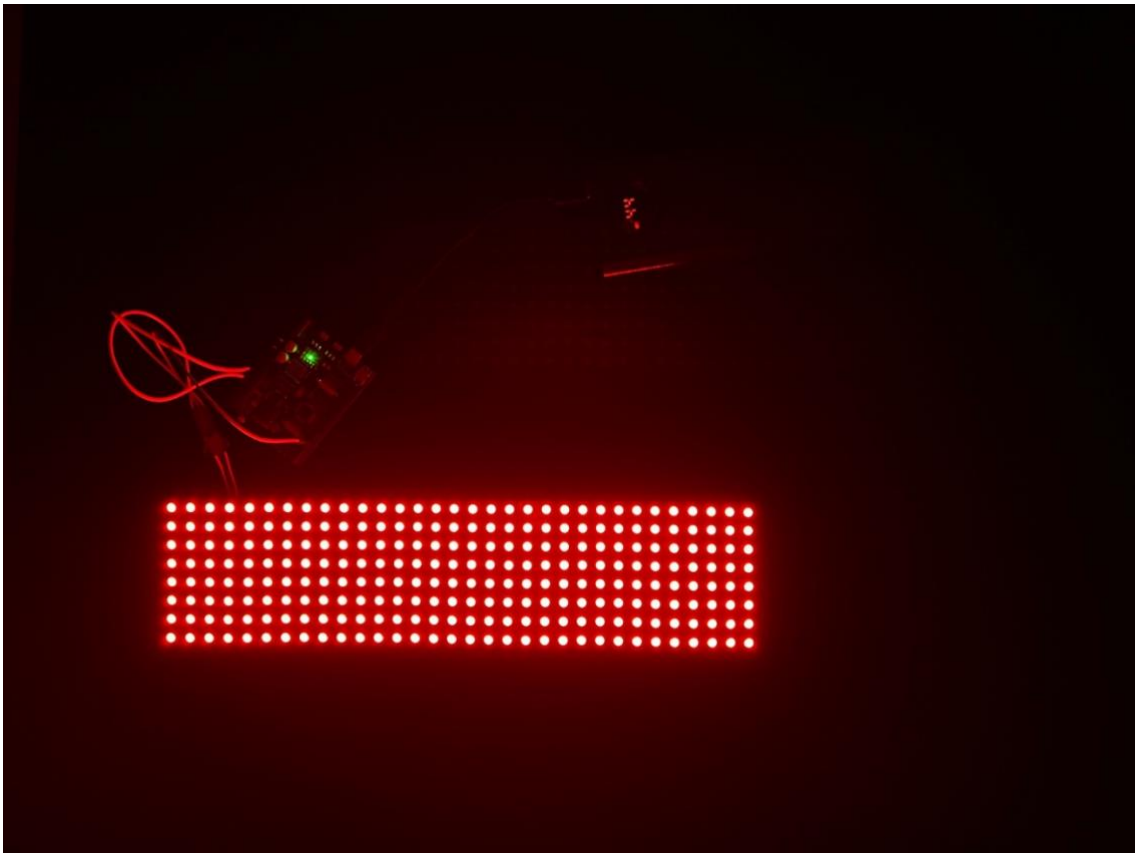


Рисунок 3.15 – Освітлення при режимі «небезпеки»

Бачимо, що дійсно колір змінився на червоний. Саме це може допомогти жителям міст (а також віддалених частин України, де сповіщення майже не чути) вберегти своє життя.

Додатково можемо переглянути на панелі «Light info», що система змінила колір з білого на червоний (рисунок 3.16):

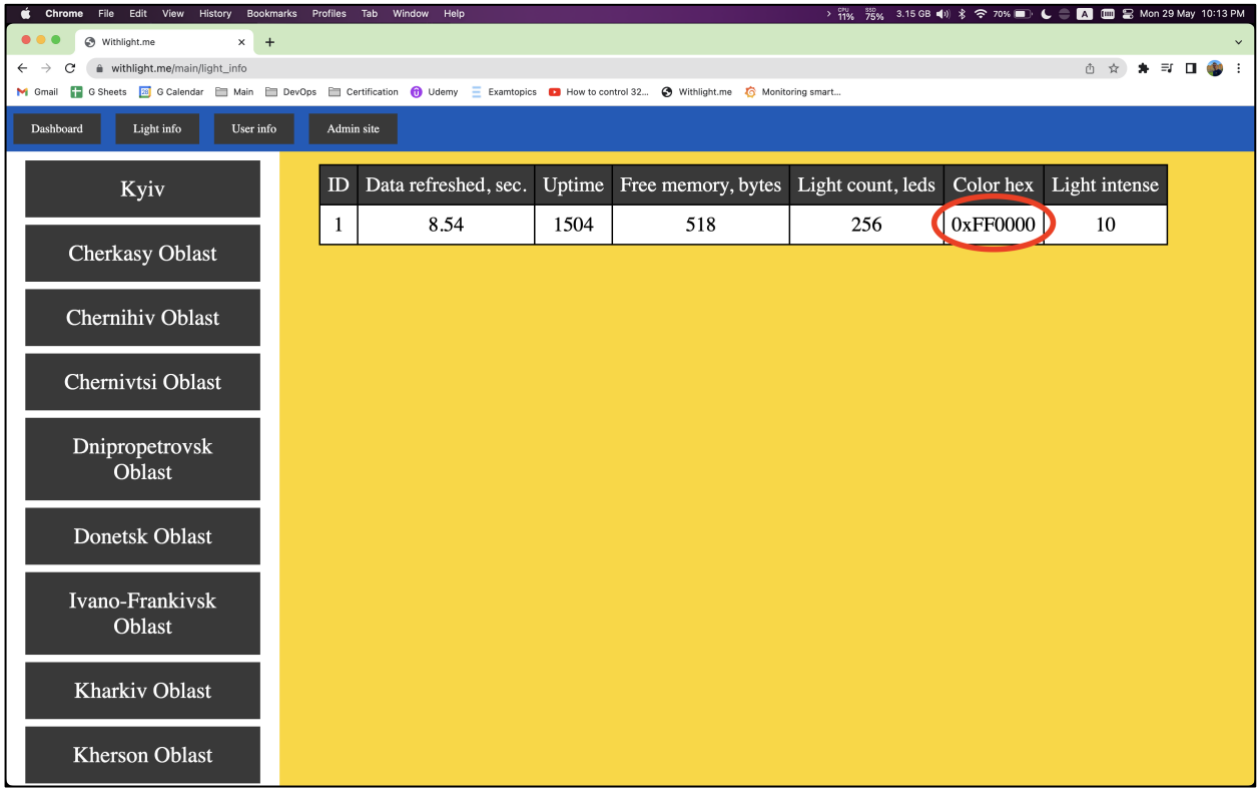


Рисунок 3.16 – Відображення інформації про освітлювальний прилад в режимі «небезпеки»

Перемкнемо на «безпечний» стан освітлювальних приладів та спробуємо створити користувача з правами «Reader». Для цього переходимо знову на панель керування адміністратора та заповнюємо всю інформацію про користувача (рисунок 3.17):

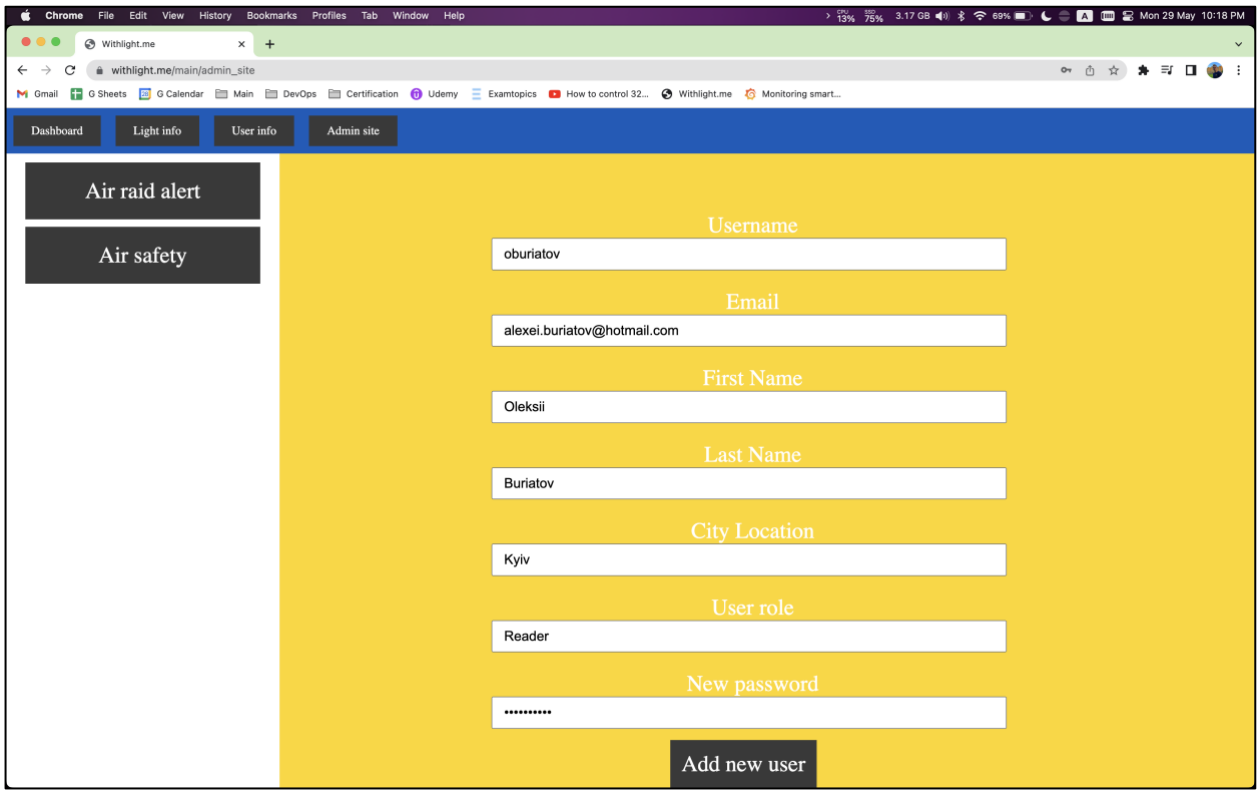


Рисунок 3.17 – Створення нового нового користувача

Натискаємо кнопку «Add new user» та можемо авторизуватися з-під нього.

А поки, можемо відключити світлове обладнання від постачання електроживлення та «зімітуємо» несправність. Перейдемо на панель керування «Dashboard» (рисунок 3.18):

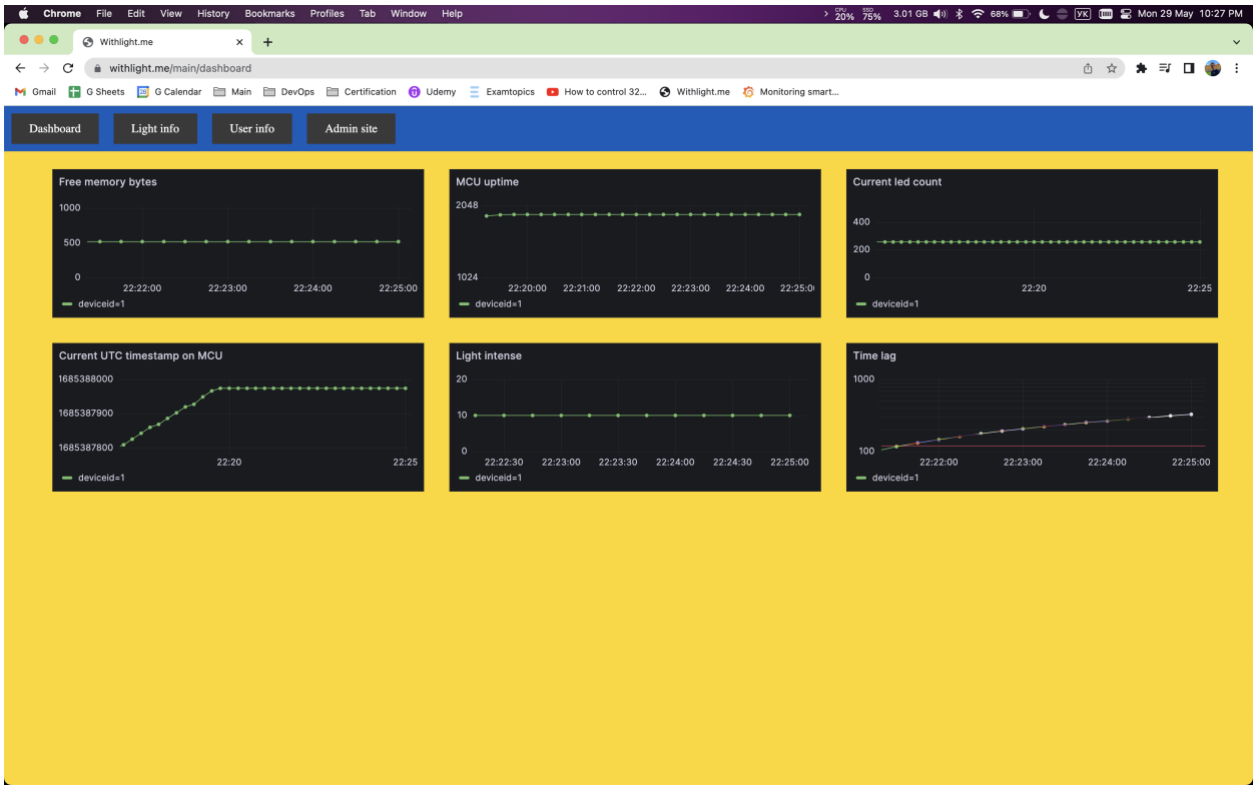


Рисунок 3.18 – Перегляд даних освітлюваних приборів у режимі несправності

Бачимо, що дані перестали оновлюватися – це значить, що мікроконтролер не отримує та не відправляє дані. Цю подію можна взяти за основу моніторингу та відправки повідомлень про недоступність системи (рисунок 3.19):

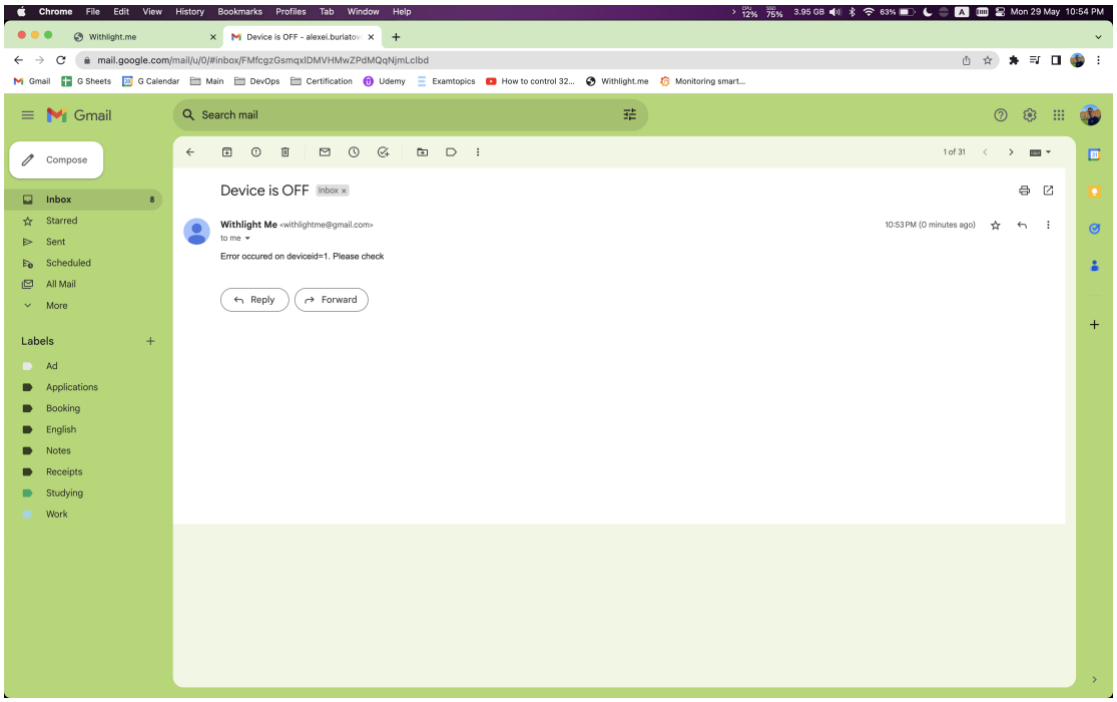


Рисунок 3.19 – Надіслане повідомлення про несправність системи освітлення

Після отримання даного повідомлення можна швидко зреагувати та виправити недоступність системи. Наприклад, перезавантажити мікроконтролери дуже часто допомагає вирішити проблему. У разі, якщо це не допомагає, скоріше за все є проблема з підключенням до мережі Інтернет. У такому випадку необхідно проаналізувати стан мережі та виправити неполадку.

Висновки до розділу

Під час написання даного розділу, було проведено аналіз програмного забезпечення. Для успішної роботи ПЗ та всіх компонентів вартує провести тестування ПЗ, а також виявити недоліки самого коду та виправити їх, що й було зроблено. За допомогою аналізатору коду Sonarqube було проведено дослідження на вразливість, правильність написання коду тощо. Дана система вказала перелік проблем, які не є критичними.

Наступною частиною написання даного розділу – це проведення мануального тестування. Було описано основні тести, які покривають більшу частину написаного програмного коду. Такий стан ПЗ є очікуваним і можна дійти висновку, що загальний стан системи є задовільний.

Отже, перевагою написання цього розділу є опис контрольного прикладу, за допомогою якого можна описати повний цикл роботи з даним програмним забезпеченням. Ілюстрації, які наведені в контрольному прикладі, є вагомими для розуміння роботи всього програмного забезпечення.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Кожне програмне забезпечення повинно бути розгорнуто та впроваджено. Розділимо цей процес на три етапи: підготовка компонент для хмари, розгортання приватної хмари з сервісами та веб-застосунків, програмування мікроконтролерів.

Етап перший:

Перш за все, для того щоб мати доступ до API менеджера, веб-застосунку або інший хмарних сервісів – необхідно мати DNS ім’я та SSL сертифікати (для безпечної передачі даних). Так як для студентів є можливість безкоштовно отримати DNS ім’я та SSL сертифікати з допомогою GitHub Students Pack – скористаймося цією можливістю та зареєструємося на Namecheap.com та отримаймо безкоштовне ім’я. Отримане ім’я зображено на рисунку 4.1

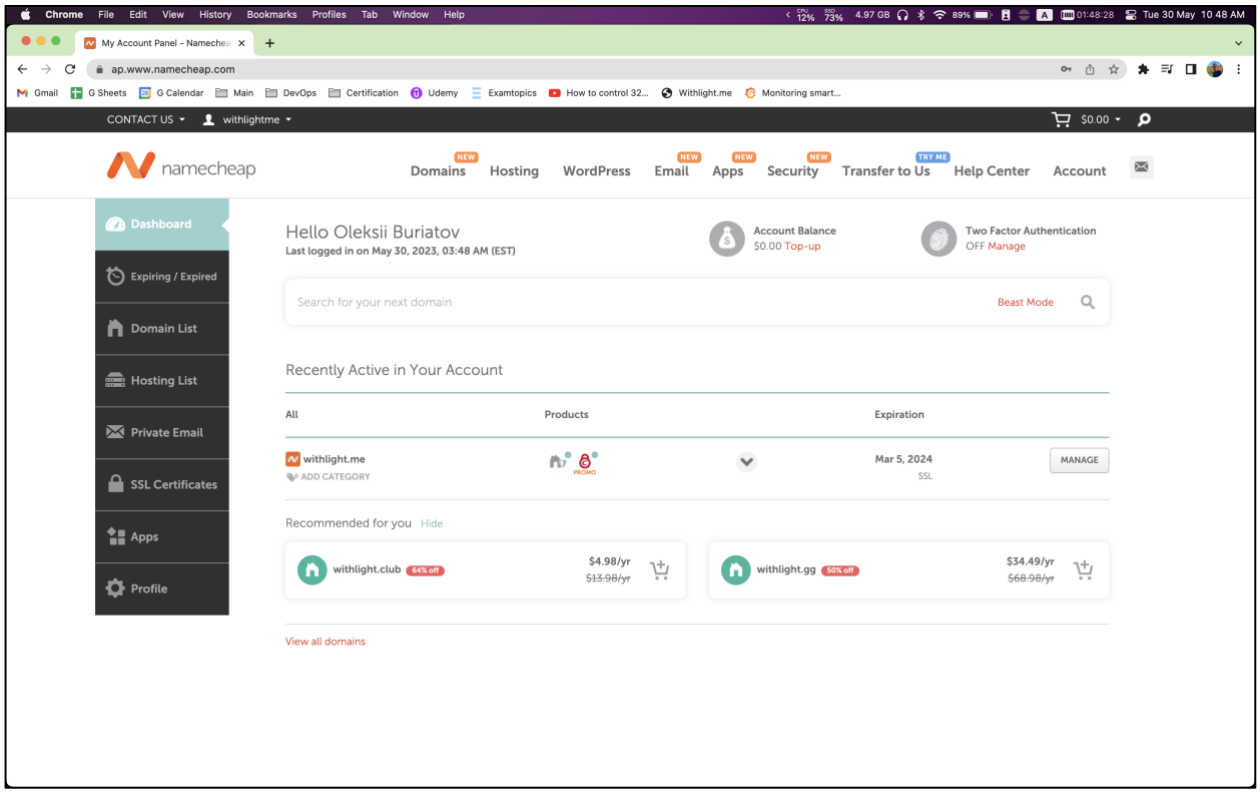


Рисунок 4.1 – Отримане DNS ім’я та SSL сертифікати

Наступним кроком потрібно підготувати середовище до розгортання хмарних компонентів у MS Azure. Для цього потрібно мати підписку, з якої будуть списуватися гроші (pay-as-you-go/партнерська тощо). Переходимо на https://portal.azure.com та перевіряємо чи є активна підписка. У даному випадку адміністратор акаунту має декілька партнерських підписок і варто обрати одну з них, наприклад – DMP_Sandbox_Buriatov (рисунок 4.2):

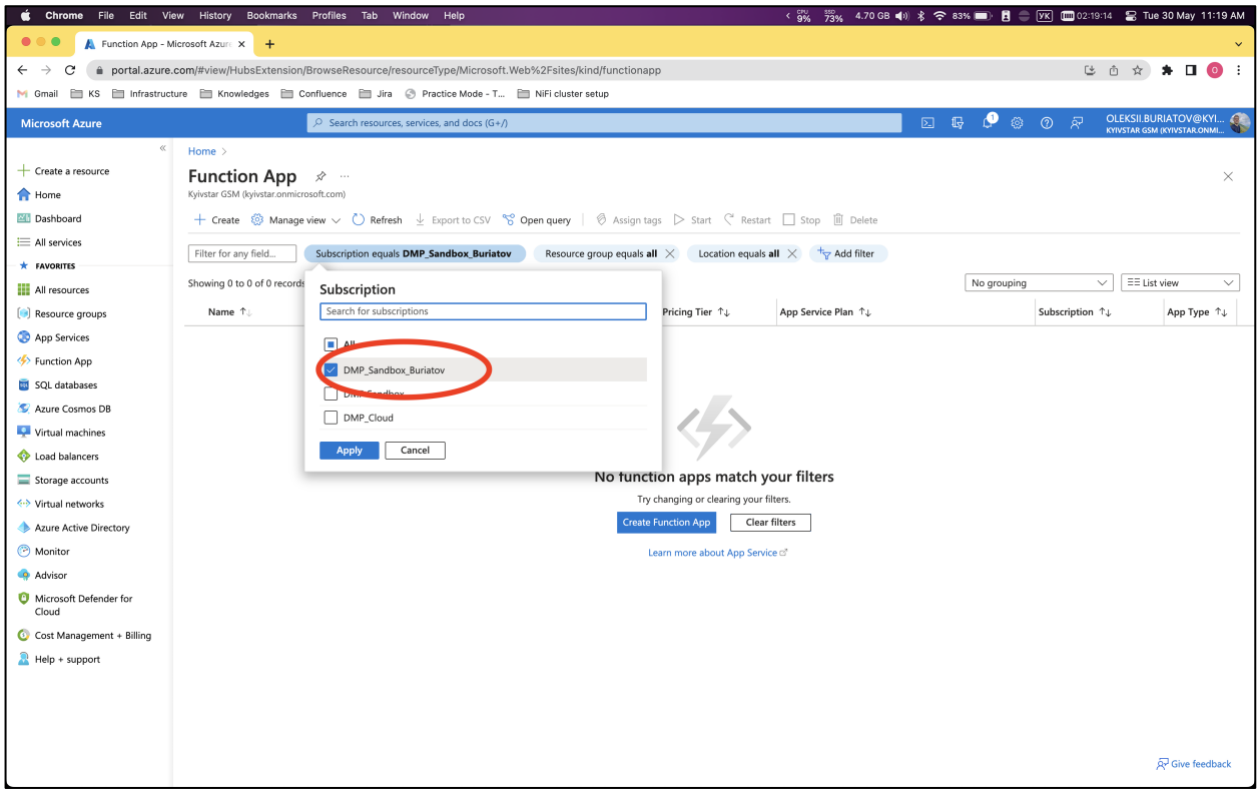


Рисунок 4.2 – Портал MS Azure та активні підписки

Цих кроків, на жаль, ще недостатньо, аби розгортати приватну хмару. Необхідно додатково встановити два компоненти на робочий комп'ютер, аби мати змогу використовувати IaC та Terraform, щоб віддалено розгортати всі компоненти. Для цього переходимо в Terminal та встановлюємо пакети azure-cli та terraform (приклад наведений для ОС MacOS). Виконуємо команду (рисунок 4.3):

brew install terraform azure-cli

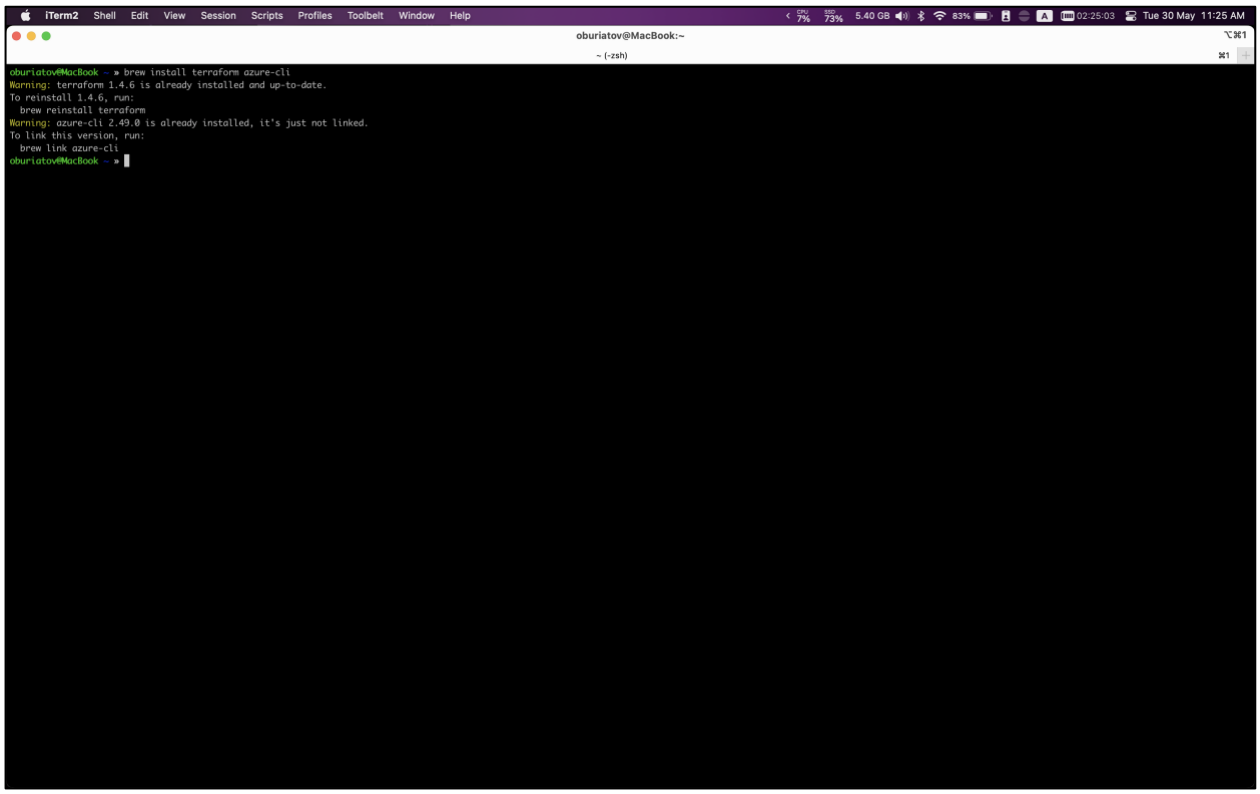


Рисунок 4.3 – Встановлення пакетів на робочий комп’ютер

Так як на даному прикладі вже встановлені ці компоненти, менеджер пакетів brew не виконує ніяких дій, проте якщо б їх не було, то вони були б встановлені.

Наступним кроком потірбно авторизуватися через azure-cli. Виконуємо наступні команди:

az login

Сервіс перенаправляє до браузеру для авторизації (рисунок 4.4):

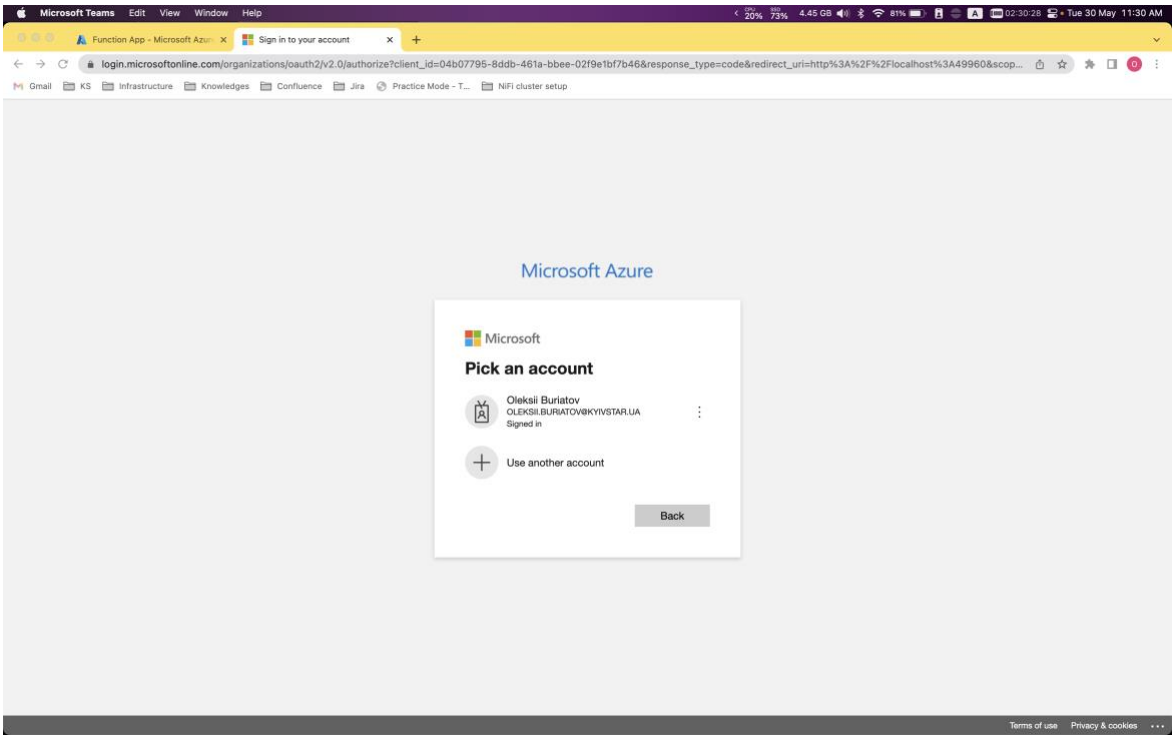


Рисунок 4.4 – Авторизація для az служби через веб-браузер

Необхідно підтвердити свій акаунт. Після успішної авторизації обираємо ту підписку, з якою буде відбуватися розгортання компонентів. Виконуємо команду, яка відобразить усі підписки:

az account list --output table

Задля безпеки, SubscriptionId на даному прикладі схований (рисунок 4.5):

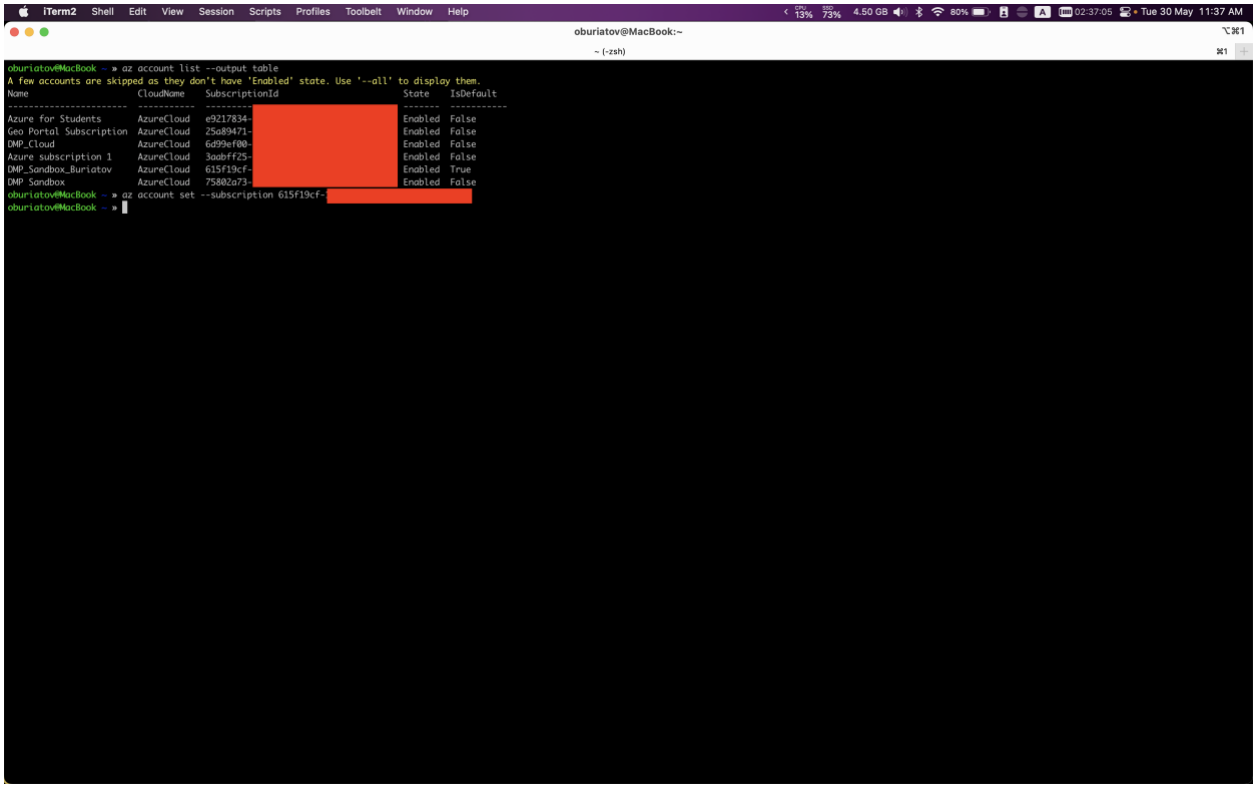


Рисунок 4.5 – Перевірка всіх підписок

За допомогою *az account set –subscription xxxxx-xxxxx-xxxxx-xxxx* обираємо підписку, з якою будемо працювати через Terraform.

Етап другий:

Переходимо в свій проект з наявними файлами *.tf* та виконуємо наступні команди:

terraform init

terraform apply

За допомогою цих команд ініціалізується середовище Terraform та виконується програмний код середовища (рисунок 4.6):

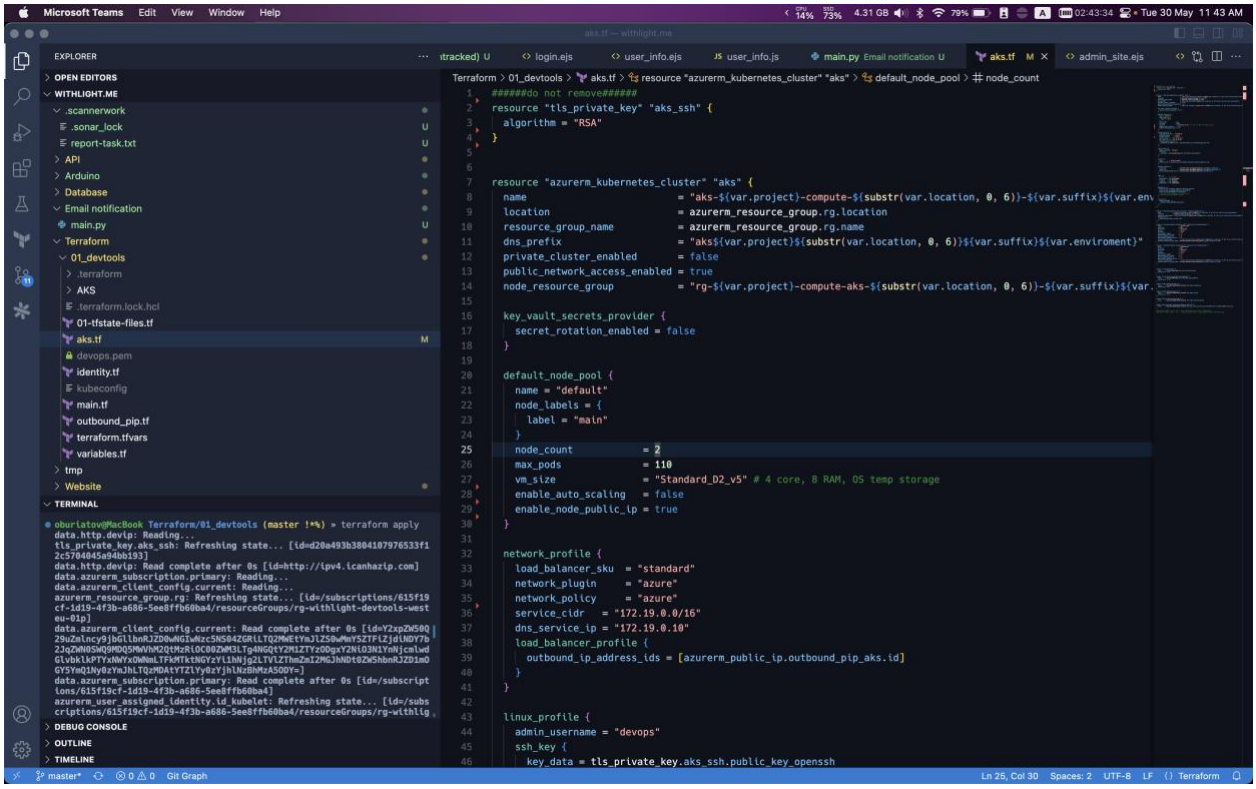


Рисунок 4.6 – Виконання коду Terraform

На цьому етапі розгортаються всі необхідні компоненти, а саме: AKS, дисковий простір, Azure Load Balancer, усі веб-застосунки та API Manager та резервуємо публічний IP адрес, який треба вказати в Namecheap для DNS розпізнавання (рисунок 4.7):

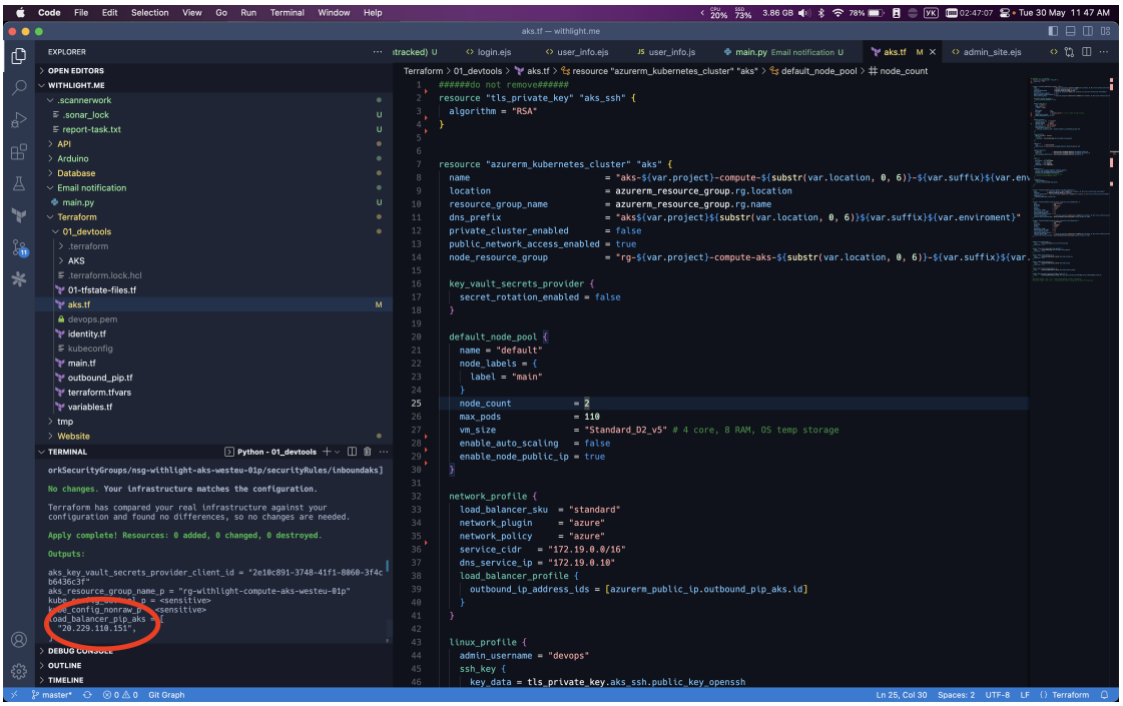


Рисунок 4.7 – Зарезервована IP адреса для DNS розпізнавання

Переходимо на сайт https://ap.www.namecheap.com/Domains/DomainControlPane та вписуємо IP адресу з нашим ім'ям (рисунок 4.8):

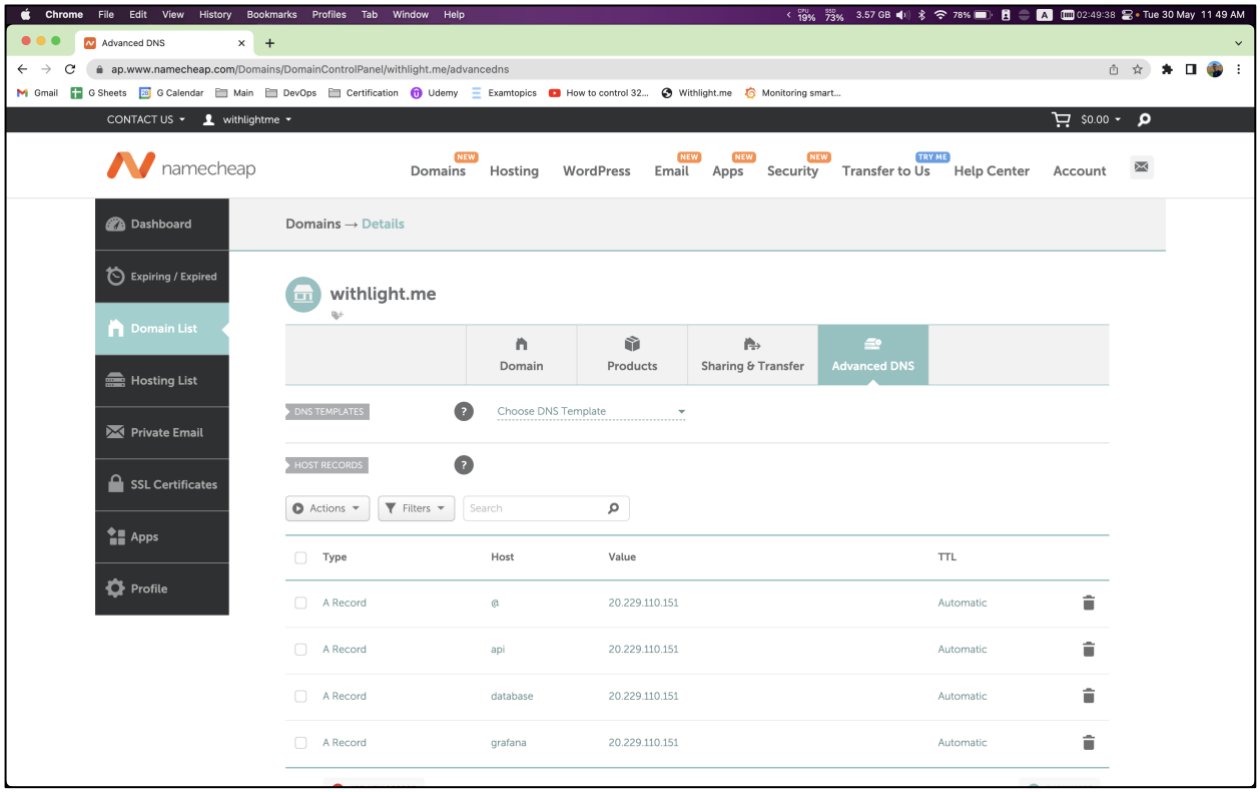


Рисунок 4.8 – Додавання IP адреси до доменного імені

Через деякий час ми отримаємо доступ до веб-застосунку та компонент проекту. Перевіримо через мережевий протокол *telnet* (рисунок 4.9):

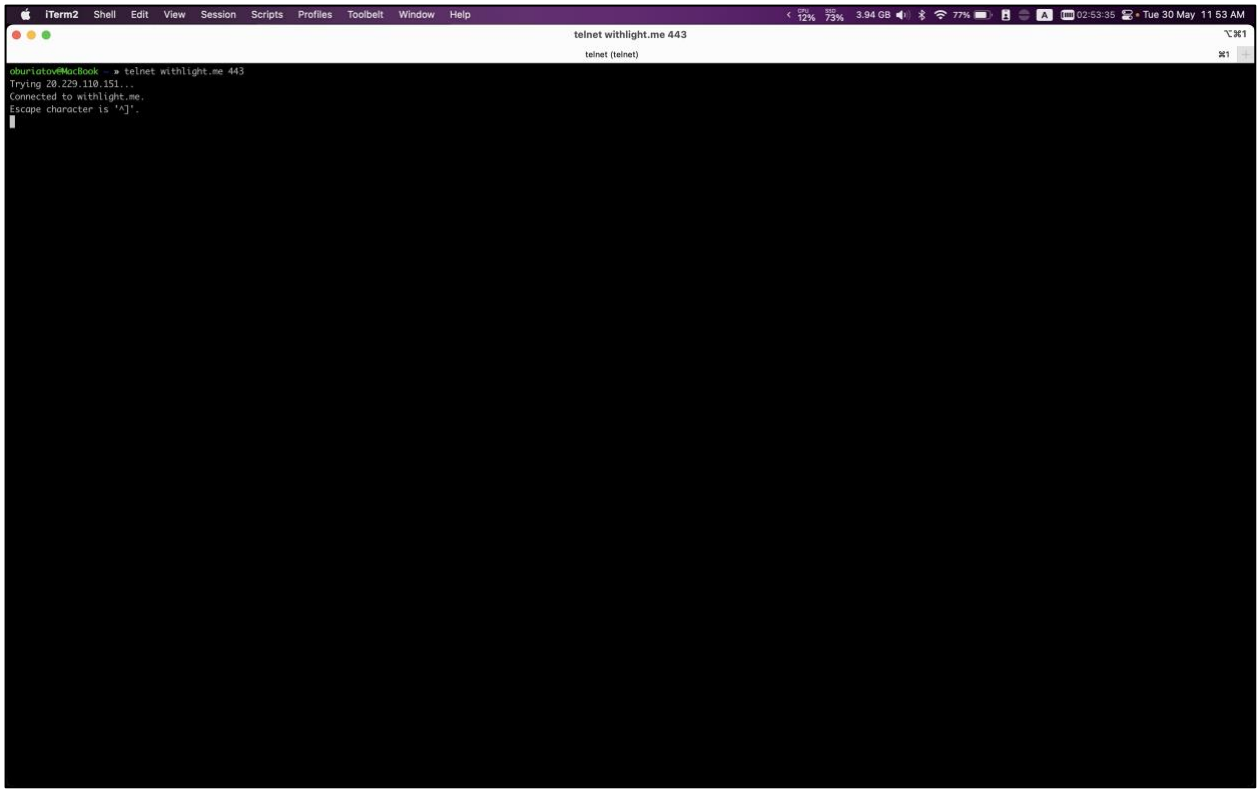


Рисунок 4.9 – Перевірка доступності компонентів у хмарі

Як бачимо, ми звертаємося до нашого застосунку не через IP адресу віртуальної машини, на якій розгорнутий AKS, а через Load Balancer, який додатково захищений брандмауером.

Етап третій:

На цьому етапі нам потрібно запрограмувати мікроконтролери Arduino UNO та WiFi модуль ESP8266. Підключаємо MCU до нашого комп’ютеру та відкриваємо Arduino IDE (рисунок 4.10):

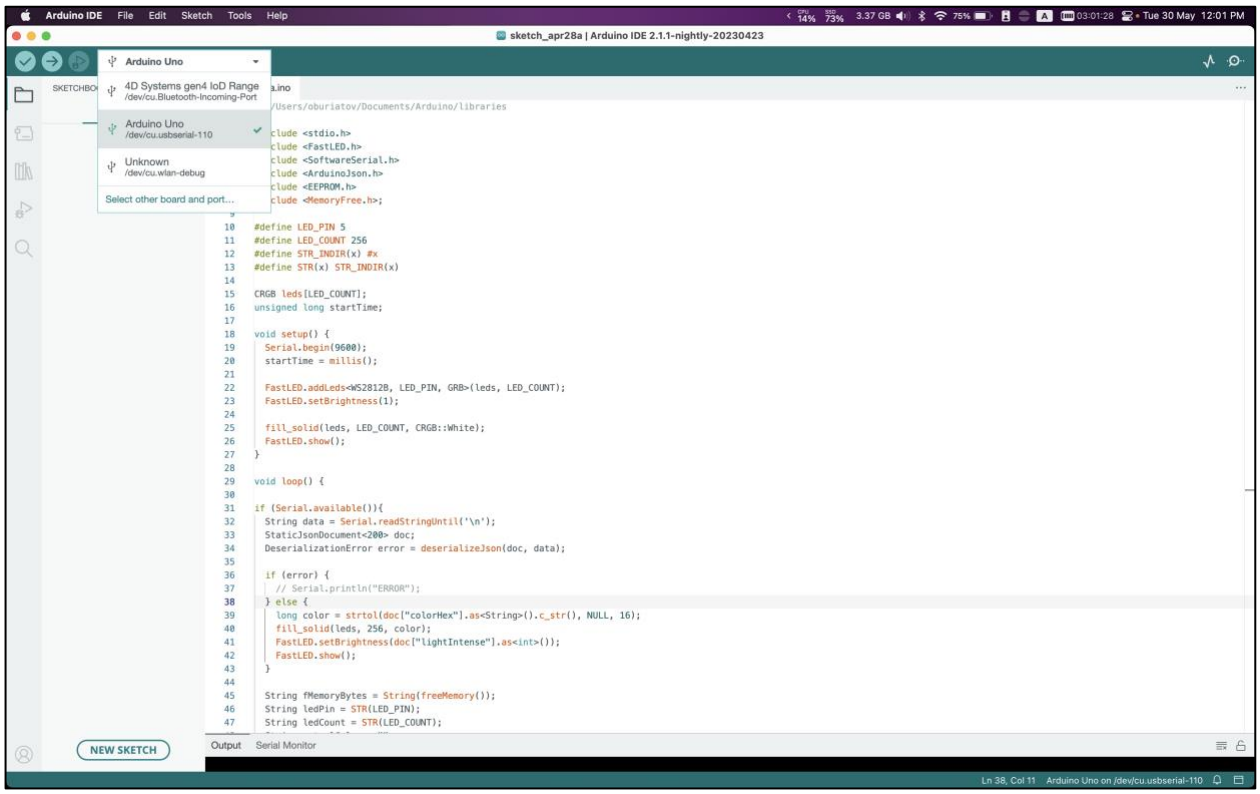


Рисунок 4.10 – Arduino IDE та програмований код

Щоб MCU був запрограмований – перевіряємо його через Arduino UNO та завантажуємо його (рисунок 4.11 – 4.12):

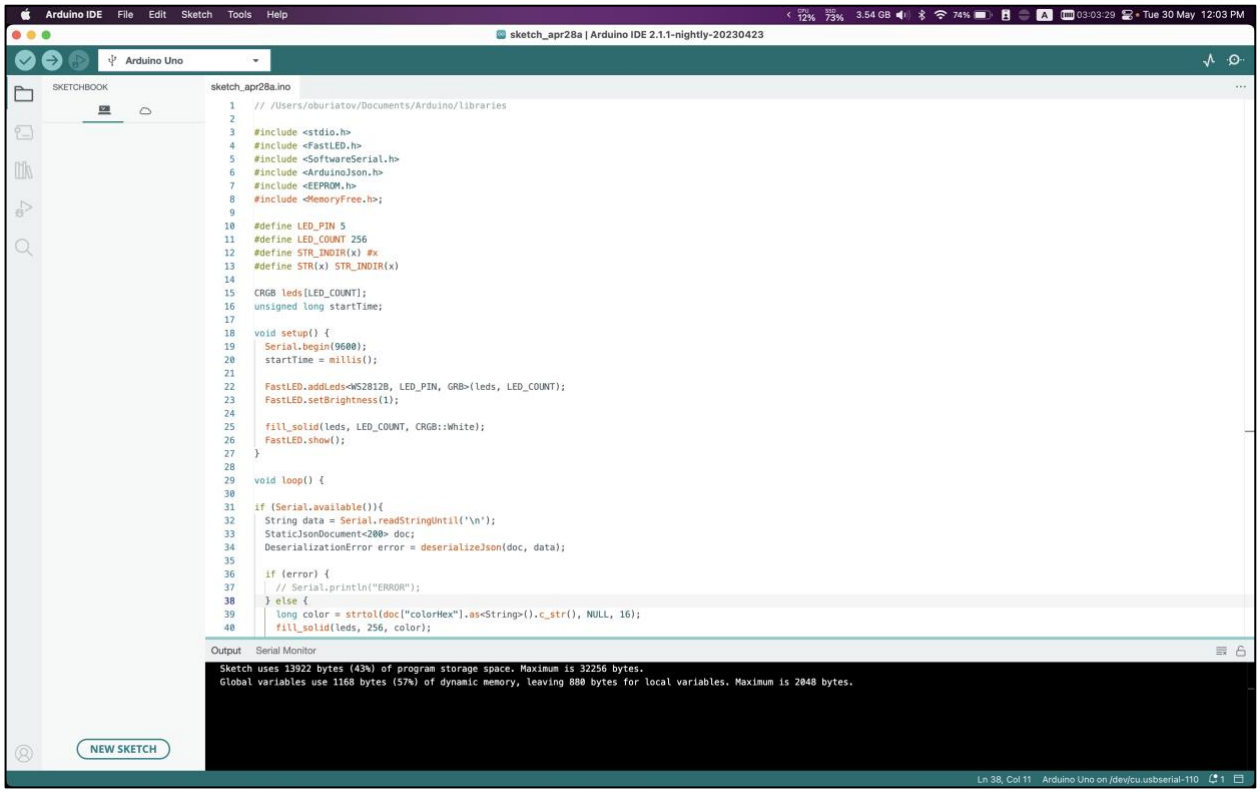


Рисунок 4.11 – Запрограмований мікроконтролер та статус виконання

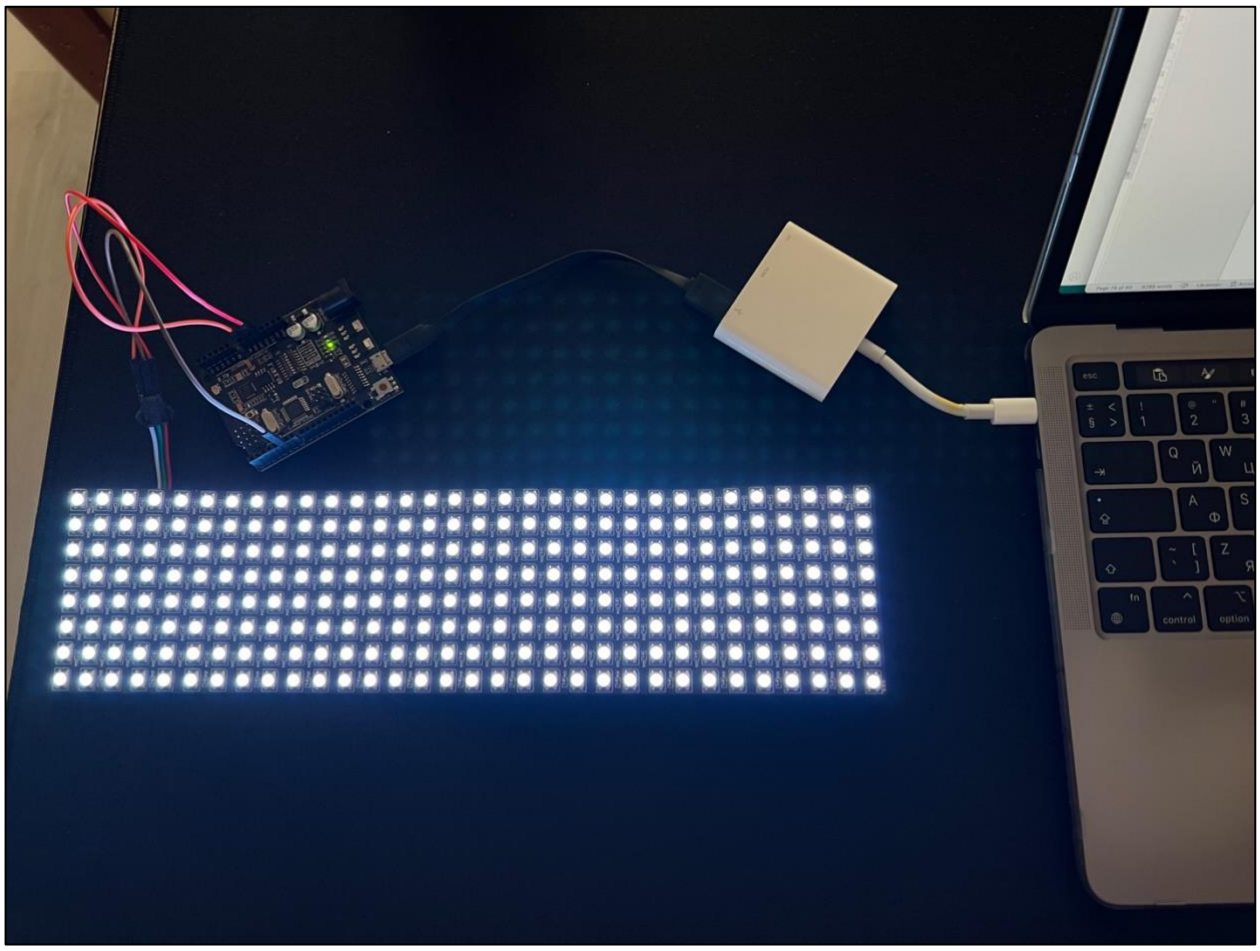


Рисунок 4.12 – Перевірка працездатності MCU

Повторюємо ті самі дії з ESP8266. На цьому етапі процес розгортання завершений.

4.2 Підтримка програмного забезпечення

Дане програмне забезпечення потребує підтримки. Так як це ПЗ зачіпає декілька напрямів підтримки, а саме: підтримка хмарних компонентів, підтримка веб-застосунку та API менеджера і підтримка працездатності MCU, то необхідний додатковий персонал у вигляді системного адміністратора, DevOps інженера та інженера освітлювальний приборів. Системний адміністратор відповідає за стан хмарних компонентів та підтримки дискового простору та AKS. DevOps інженер відповідальний за оновлення програмного забезпечення. Інженер освітлювальних приборів відповідає за безперебійне постачання електроживлення та виїзд до місць аварій.

Усе оновлення програмного забезпечення відбувається через IaC – Terraform. Для цього необхідно в програмному коду файлу *.tf* змінити версію, яка новіша за попередню того чи іншого застосунку та виконати команду (рисунок 4.13):

terraform apply

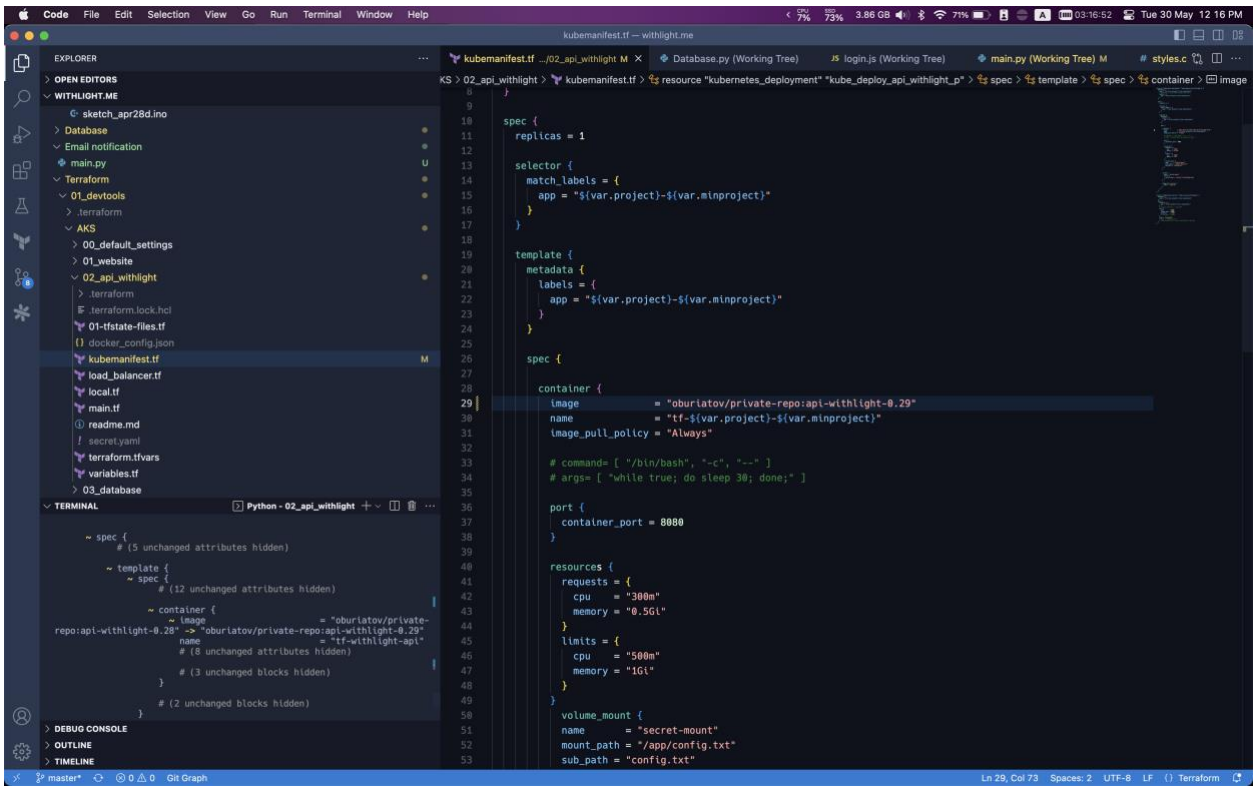


Рисунок 4.13 – Оновлення нових версій застосунків

Бачимо, що Terraform помітив зміни та виконає розгортання сервісу в хмарі з оновленою версією.

Висновки до розділу

Даний розділ був присвячений розгортанню та підтримці програмного забезпечення. Під час розробки даного проекту, було впроваджено новітні технології по автоматизації, а саме використання методології IaC. Цей підхід дозволяє описувати хмарні компоненти за допомогою коду, що дозволяє отримувати консистентність системи при перевстановленні хмарних інструментів або при інших випадках.

Також було впроваджено безпечну передачу інформації за допомогою SSL сертифікатів та доменного імені. Це допомагає не запам’ятовувати IP адресу балансувальника даних, а використовувати ім’я. Процес отримання доменного був описаний в даному розділі.

Отже, процес розгортання програмного забезпечення потребує додаткових знань, якими повинні володіти персонал з обслуговування та підтримки цієї системи.

ВИСНОВКИ

Хмарне програмне забезпечення для моніторингу освітлення міста розроблялося відповідно до поставленої мети та задач.

На самому початку розробки програмного забезпечення була оцінена предметна область з якою ця тема перетинається. Упродовж досліджень, було знайдено схожі проекти, які, до речі, є актуальними, що допомогли більш глибоко вивчити проблематику та поставити конкретні й визначені дії для досягнення мети.

Спершу, аби починати розробку програмного забезпечення, постала необхідність в обранні мікроконтролерів, аби втілити задумане. Arduino UNO, WiFi модуль ESP8266 та LED стрічка WS8212B стали основою подальшої реалізації проекту. Основною перепоною для побудови багаторівневої логіки роботи мікроконтролерів стало обмеження в ресурсах – а саме RAM пам’яті. У даних моделях є обмеження в ресурсах – 4КБ вільного простору, 20% з яких стає зайнятим при запуску. Проте було змінено підхід до написання коду та завдяки цьому було вивільнено частину пам’яті, якої вистачає для роботи.

Наступним кроком необхідно було вибрати хмарного провайдера, де будуть впроваджені всі сервіси. Основними трьома гравцями на ринку хмарних послуг – Microsoft Azure, Amazon Web Services та Google Cloud Platform. Кожен з них має свої переваги та недоліки, проте Microsoft Azure виграв у порівнянні з AWS та GCP вартістю AKS та швидкістю відгуку до сервісів. Було прийнято рішення описувати, використовуючи IaC та розгортати хмарну інфраструктуру в Azure.

Основною частиною розробки програмного забезпечення це розробка API менеджера, сховищ даних та веб-застосунку. Так як основними клієнтами даного рішення є мікроконтролер та люди, тому було прийнято рішення впровадження клієнт-серверної архітектури, де API менеджер виступає в ролі серверної сторони і забирає/віддає дані до DWH, а в ролі клієнтів виступають мікроконтролери та люди, що користуються веб-застосунком.

У якості API менеджера було обрано Python застосунком з бібліотекою Flask. Це рішення суттєво полегшило взаємодію з реляційними та time-series базами даних, адже швидкодія + легкість застосунку підвищила продуктивність та час отримання результатів.

У якості DWH було розроблено комплексне рішення у вигляді реляційної бази даних Postgres та нереляційної бази даних Prometheus. Postgres використовується в якості зберігання інформації по користувачам, мікроконтролерам та актуальному стану датчиків. Дане сховище розроблено в 2НФ.

Prometheus виступає в якості довготривалого зберігання даних. Ця нереляційна база даних дозволяє дуже швидко отримувати метрики в заданому часовому просторі.

У якості веб-застосунку було використано фреймворк Express для backend та EJS шаблони. Завдяки цьому програмне забезпечення використовує сучасний підхід до розробки. Після реалізації веб-застосунку, додаток було протестовано на екрані з розмірністю 1920x1080 пікселів. Так як аудиторія даного веб-застосунку є вузькою, то була націленість на десктопний формат екрану.

У свою чергу додаток був розроблений лише для моніторингу освітлювальними приладами, а не для керування, проте була втілена ідея додаткової функції «Небезпека». Мається на увазі, що у разі якоїсь загрози або повітряної тривоги, адміністратор системи має змогу змінити колір освітлювальних приладів з білого на червоний і таким чином попередити мешканців міста про загрозу.

Проте, незважаючи на те, що керування освітленням обмежене, було створено автоматизовані процеси, які дозволяють в автоматичному режимі вмикати та вимикати освітлення згідно з заданими параметрами, що в свою чергу збільшує енергоефективність та зменшує викиди вуглекислого газу в атмосферу.

Отже, підбиваючи все вищесказане, можна дійти до висновку, що з поставленою задачею розробки хмарного програмного забезпечення для моніторингу освітлення міста справилися. Наступним етапом є розвиток та додавання функціоналу як до мікроконтролерів (наприклад, встановлення датчиків руху та LTE модулі) та і для веб-застосунку (впровадження нових каналів сповіщення, розробка нових метрик моніторингу тощо).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Intelligent Street Lighting System Using Gsm [Електронний ресурс] // International Journal of Engineering Science Invention. – 2013. – Режим доступу до ресурсу: https://www.idc-online.com/technical_references/pdfs/electronic_engineering/Intelligent%20Street%20Lighting.pdf.

2) COMPARATIVE STUDY OF CLOUD PLATFORMS - MICROSOFT AZURE, GOOGLE CLOUD PLATFORM AND AMAZON EC2 [Електронний ресурс] // JREAS. – 2020. – Режим доступу до ресурсу: https://pdfs.semanticscholar.org/10bf/f6d5dee2c2dd62f85eac3ea1900045cae820.pdf.

3) Dlnya A. A. Webserver Based Smart Monitoring System Using ESP8266 Node MCU Module [Електронний ресурс] / Abdulahad Aziz Dlnya // International Journal Of Scientific & Engineering Research. – 2018. – Режим доступу до ресурсу: https://www.researchgate.net/profile/Dlnya-Aziz-4/publication/326672970_Webserver_Based_Smart_Monitoring_System_Using_ESP8266_Node_MCU_Module/links/5b5cae8e458515c4b2501919/Webserver-Based-Smart-Monitoring-System-Using-ESP8266-Node-MCU-Module.pdf.

4) Keycloak - Identity and Access Management for Modern Applications – UK: Packt Publishing Ltd., 2021. – 316 с.

5) Arduino IDE [Електронний ресурс] // Arduino. – Режим доступу до ресурсу: https://www.arduino.cc/

6) Kubernetes [Електронний ресурс] // The Kubernetes. – Режим доступу до ресурсу: https://kubernetes.io/

7) Prometheus [Електронний ресурс] // Prometheus Authors. – Режим доступу до ресурсу: https://prometheus.io/

8) Grafana [Електронний ресурс] // Grafana Labs. – Режим доступу до ресурсу: https://grafana.com/

9) ExpressJS [Електронний ресурс] // OpenJS Foundation. – Режим доступу до ресурсу: https://expressjs.com/

10) Flask API [Електронний ресурс] // Pallets. – 2010 – Режим доступу до ресурсу: https://flask.palletsprojects.com/en/2.3.x/api/

11) Postgres library [Електронний ресурс] // PyPi. – 2023. – Режим доступу до ресурсу: https://pypi.org/project/psycopg2/

12) Crypto JS [Електронний ресурс] // GitHub. – 2023. – Режим доступу до ресурсу: https://github.com/brix/crypto-js

ДОДАТОК А ЗВІТ ПОДІБНОСТИ

UNICHECK

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1015369925

Дата перевірки:
01.06.2023 15:19:18 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
01.06.2023 15:48:15 EEST

ID користувача:
76913

Назва документа: **ІП-96_Бурятов_ПЗ**

Кількість сторінок: **76** Кількість слів: **10578** Кількість символів: **77582** Розмір файлу: **19.61 MB** ID файлу: **1015036232**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

3.69%
Схожість

Найбільша схожість: **1.14%** з джерелом з Бібліотеки (ID файлу: **1014971150**)

1.63% Джерела з Інтернету	103	Сторінка 78
3.57% Джерела з Бібліотеки	215	Сторінка 78

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	12 сторінок

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ХМАРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ОСВІТЛЕННЯ МІСТА

Текст програми

КПІ.ІП-9604.045430.03.12

“ПОГОДЖЕНО”
Керівник проекту:
_____ Ірина ВІТКОВСЬКА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Олексій БУРЯТОВ

Київ – 2023

Файл sketch_apr28a.inu

```
#include <stdio.h>
#include <FastLED.h>
#include <SoftwareSerial.h>
#include <ArduinoJson.h>
#include <EEPROM.h>
#include <MemoryFree.h>;

#define LED_PIN 5
#define LED_COUNT 256
#define STR_INDIR(x) #x
#define STR(x) STR_INDIR(x)

CRGB leds[LED_COUNT];
unsigned long startTime;

void setup() {
  Serial.begin(9600);
  startTime = millis();

  FastLED.addLeds<WS2812B, LED_PIN, GRB>(leds, LED_COUNT);
  FastLED.setBrightness(1);

  fill_solid(leds, LED_COUNT, CRGB::White);
  FastLED.show();
}

void loop() {

if (Serial.available()){
  String data = Serial.readStringUntil('\n');
  StaticJsonDocument<200> doc;
  DeserializationError error = deserializeJson(doc, data);

  if (error) {
  } else {
    long color = strtol(doc["colorHex"].as<String>().c_str(), NULL, 16);
    fill_solid(leds, 256, color);
    FastLED.setBrightness(doc["lightIntense"].as<int>());
    FastLED.show();
  }
```

```
        String fMemoryBytes = String(freeMemory());
        String ledPin = STR(LED_PIN);
        String ledCount = STR(LED_COUNT);
        String actualColor = "";
        String errorParseJson = "";
        unsigned long uptime = millis()/1000 - startTime;

        String postDataSend = "\"up\": " + String(uptime) + ", \"fMemBytes\":
" + fMemoryBytes + ", \"ledCount\": " + ledCount;
        Serial.println(postDataSend);
      }
      }
```

Файл sketch_apr28d.ino

```
      #include <ESP8266WiFi.h>
      #include <ESP8266HTTPClient.h>
      #include <ArduinoJson.h>
      #include <WiFiClientSecure.h>
      #include <SoftwareSerial.h>
      #include <ArduinoJson.h>
      #include <WiFiUdp.h>
      #include <NTPClient.h>

      WiFiUDP ntpUDP;
      NTPClient timeClient(ntpUDP);

      const char* ssid = "";
      const char* password = "";

      const char* endpointUrl = "api.withlight.me";
      const char* bearerToken = "";
      const char fingerprint[] PROGMEM= "";
      String requestHeaders = "Authorization: Bearer " + String(bearerToken) +
"\r\n";
      String postRequestHeaders = "Bearer " + String(bearerToken);

      void setup() {
        Serial.begin(9600);

        delay(10);
        WiFi.begin(ssid, password);
```

```
    while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }

  timeClient.begin();
}

void loop() {
  WiFiClientSecure client;
  client.setFingerprint(fingerprint);

  if (!client.connect(endpointUrl, 443)) {
    return;
  }
  String response = "";

  if (Serial.available()){
    timeClient.update();
    String timestamp = String(timeClient.getEpochTime());
    String postData = "{ " + Serial.readString() + ", \"timestamp\": " +
timestamp +" }";
    HTTPClient http;
    http.begin(client, endpointUrl, 443, "/mcu/monitoring/deviceid=1");
    http.addHeader("Authorization", postRequestHeaders);
    http.addHeader("Content-Type", "application/json");
    int httpResponseCode = http.POST(postData);
    response = http.getString();
  }

  response = "";
  client.println("GET /mcu/verify/deviceid=1 HTTP/1.1");
  client.println("Host: api.withlight.me");
  client.println(requestHeaders);

  while (client.connected()) {
    String line = client.readStringUntil('\n');
    if (line == "\r") {
      break;
    }
  }
  while (client.available()) {
    String line = client.readStringUntil('\r');
    response += line;
```

```
  }
  Serial.println(response);
}
```

Файл main.py

```
from flask import Flask, jsonify, request,abort
from prometheus_client import CollectorRegistry, Gauge, push_to_gateway
import psycopg2
import configparser
import json
import base64

from Database import Database;
from MCU import MCU;
from User import User;
from Metadata import Metadata;

app = Flask(__name__)

@app.route('/mcu/verify/deviceid=<deviceid>', methods=['GET'])
def verify(deviceid):
    microcontroller = MCU();

    sql_query = str("SELECT \
        device_id,\
        color_hex,\
        light_intense,\
        status \
    FROM api.device_status\
    WHERE device_id ="+deviceid)

    sql_result = db.runSelectSQL(sql_query);
    microcontroller.setVerifyStatus(sql_result)

    db.setMetricLightIntenseToPrometheus(deviceid,sql_result[0],
int(sql_result[1][2]))
    return microcontroller.getVerifyStatus()

@app.route('/mcu/monitoring/deviceid=<deviceid>', methods=['POST'])
def monitoring(deviceid):
```

```
            response = request.get_json();
            microcontroller = MCU();
            microcontroller.setMonitoringInput(deviceid, response);

            sql_query = str("UPDATE api.device_monitoring SET\
                        up_time= " +microcontroller.upTime +
                        ", timestamp= to_timestamp(" + microcontroller.timestamp
+
                        "), led_light_count= " + microcontroller.ledCount +
                        ", f_mem_bytes=" + microcontroller.fMemBytes +
                    " WHERE device_id ="+ microcontroller.deviceId
                    );
            result = db.runInsertSQL(sql_query);


db.setMetricToPrometheus(deviceid,result[0],microcontroller.upTime,microcontr
oller.fMemBytes,microcontroller.timestamp,microcontroller.ledCount)
            return microcontroller.getMonitoringStatus(result);

      @app.route('/withlight-rest-api/login/username=<username>',
methods=['PUT'])
      def userlogin(username):

                response = request.get_json();
                user = None;

                sql_query = str("SELECT ui.userid,\
                            ui.username,\
                            ui.usermail, \
                            ui.fname,\
                            ui.lname,\
                            ur.rolename,\
                            ci.cityname,\
                            ui.passwdhash \
                            FROM  infostore.user_info ui\
                            JOIN  infostore.city_info  ci  ON  ui.citylocation  =
ci.cityid\
                            JOIN   infostore.user_role   ur   ON   ui.userrole   =
ur.roleid\
                            WHERE username= '" + username + "'")

                result = db.runSelectSQL(sql_query);
                user = User(result)
```

```
            return user.getUserInfo();

      @app.route('/withlight-rest-api/user/updateinfo=<username>',
methods=['PUT'])
      def updateinfo(username):

            response = request.get_json();
            print(response)

            sql_query = str("UPDATE infostore.user_info\
                        SET usermail ='" +response['email'] +"',\
                        fname ='"+response['fname'] +"',\
                        lName ='"+response['lname'] +"'\
                        WHERE username= '" + username +"'"
            )

            sql_result = db.runInsertSQL(sql_query);

            if(sql_result[0]==200):
                return json.dumps({
                    'errorCode': sql_result[0]
                });
            elif(sql_result[0]==404):
                abort(sql_result[0])

      @app.route('/withlight-rest-api/city-list', methods=['GET'])
      def getCityList():
            sql_query = str("SELECT cityid, cityname\
                        FROM  infostore.city_info\
                        ORDER BY cityname ASC")

            result = db.runSelectMultipleSQL(sql_query);
            metadata = Metadata("getCityList",result[1], result[0])
            return metadata.getFullCityList()

      @app.route('/withlight-rest-api/device-list=<cityname>',
methods=['GET'])
      def getDeviceList(cityname):
            microcontroller = MCU();

            sql_query = str("SELECT dm.device_id,\
```

```
                        round(extract(epoch from now()) -extract(epoch from
dm.timestamp),2) AS time_sec_diff,\
                        dm.up_time,dm.f_mem_bytes,\
                        dm.led_light_count,\
                        ds.color_hex,ds.light_intense\
                        FROM api.device_monitoring dm\
                        JOIN         api.device_status         ds         ON
ds.device_id=dm.device_id\
                        JOIN        infostore.device_info        di        ON
di.deviceid=ds.device_id\
                        JOIN        infostore.city_info        ci        ON
ci.cityid=di.citylocation\
                        WHERE ci.cityname = '"+ cityname + "'")

            result = db.runSelectMultipleSQL(sql_query);

            return microcontroller.getAllDeviceData(result[1], result[0])

    @app.route('/mcu/monitoring/alert=<status>', methods=['GET'])
    def airalert(status):
        microcontroller = MCU();

        if(status == "on"):
            sql_query = str("UPDATE api.device_status SET\
                        color_hex= '0xFF0000'"
                    );
            result = db.runInsertSQL(sql_query);
        elif(status == "off"):
            sql_query = str("UPDATE api.device_status SET\
                        color_hex= '0xFFFFFF'"
                    );
            result = db.runInsertSQL(sql_query);

        return microcontroller.getMonitoringStatus(result);

    if __name__ == '__main__':

        config = configparser.ConfigParser()
        config.read('config.txt')

        db = Database(
        config.get('withlight','host'),
        config.get('withlight', 'port'),
```

```
        config.get('withlight', 'name'),
        config.get('withlight', 'user'),
        config.get('withlight','password'),
        config.get('withlight','pushgateway_url'),
        config.get('withlight','api_instance_url')
        )
        app.run(debug=True, host='0.0.0.0', port='8080')
```

Файл Database.py

```
import psycopg2
import json
from prometheus_client import CollectorRegistry, Gauge, push_to_gateway

class Database:

    conn = None;
    dbHost = None;
    dbPort = None;
    dbName = None;
    dbPassword = None;
    dbUser = None;

    registry = CollectorRegistry()
    pushgatewayUrl = None;
    apiInstanceUrl = None;

    metricUptime = None;
    metricFMemBytes = None;
    metricLedCount = None;
    metricTimestamp = None;
    metricLightIntense = None;

    def __init__(self,
dbHost,dbPort,dbName,dbUser,dbPassword,pushgatewayUrl,apiInstanceUrl):
        self.dbHost = dbHost
        self.dbPort = dbPort
        self.dbName = dbName
        self.dbUser = dbUser
        self.dbPassword = dbPassword
```

```
            self.pushgatewayUrl= pushgatewayUrl;
            self.apiInstanceUrl = apiInstanceUrl;

            self.metricUptime    =    Gauge('api_monitoring_uptime_seconds',
'Current    uptime    in    seconds    after    last    reboot    of    Arduino
UNO',['instance','device_id'], registry=self.registry)
            self.metricFMemBytes   =   Gauge('api_monitoring_fmemory_bytes',
'Current  free  memory  in  bytes  of  Arduino  UNO',['instance','device_id'],
registry=self.registry)
            self.metricLedCount = Gauge('api_monitoring_led_count', 'Current
ledcount of Arduino UNO',['instance','device_id'], registry=self.registry)
            self.metricTimestamp = Gauge('api_monitoring_timestamp', 'Actual
timestamp        of        Arduino        ESP8266',['instance','device_id'],
registry=self.registry)
            self.metricLightIntense  =  Gauge('api_monitoring_light_intense',
'Actual  light  intense  of  Arduino  UNO  +  WS8212B',['instance','device_id'],
registry=self.registry)

            self.conn = psycopg2.connect(
                host=self.dbHost,
                port=self.dbPort,
                database=self.dbName,
                user=self.dbUser,
                password=self.dbPassword
            )

        def runSelectSQL(self, sql_query):
            try:
                cur= self.conn.cursor();
                cur.execute(sql_query);
                self.conn.commit();
                result = cur.fetchone();
                cur.close();

                if(result==None):
                    return [404,"None"]
                else:
                    return [200,result];
            except Exception as e:
                print("runSelectMultipleSQL was not run well" + dbName)
                print (e)
                return [404,"None"]
            finally:
```

```
            self.conn.rollback();
            cur.close();

    def runSelectMultipleSQL(self, sql_query):
        try:
            cur= self.conn.cursor();
            cur.execute(sql_query);
            self.conn.commit();
            result = cur.fetchall();
            cur.close();

            if(result==None):
                return [404,"None"]
            else:
                return [200,result];
        except Exception as e:
            print("runSelectMultipleSQL was not run wellm" +dbName)
            print (e)
            return [404,"None"]
        finally:
            self.conn.rollback();
            cur.close();

    def runInsertSQL(self, sql_query):
        try:

            cur= self.conn.cursor();
            cur.execute(sql_query);
            rowcount= cur.rowcount;
            self.conn.commit();
            cur.close();
            if(rowcount==0):
                return [404]
            else:
                return [200];
        except Exception as e:
            print("runSelectMultipleSQL was not run well" +dbName)
            print (e)
            return [404]
        finally:
            self.conn.rollback();
            cur.close();
```

```
        def                    setMetricToPrometheus(self,deviceid,sql_status,
uptime,fMemBytes,mcuTimestamp,ledCound):
            if(sql_status==200):

self.metricUptime.labels(instance=self.apiInstanceUrl,device_id=deviceid).set
(uptime)

self.metricFMemBytes.labels(instance=self.apiInstanceUrl,device_id=deviceid).
set(fMemBytes)

self.metricTimestamp.labels(instance=self.apiInstanceUrl,device_id=deviceid).
set(mcuTimestamp)

self.metricLedCount.labels(instance=self.apiInstanceUrl,device_id=deviceid).s
et(ledCound)

                push_to_gateway(self.pushgatewayUrl,
job='api_monitoring_scrape', registry=self.registry)

        def
setMetricLightIntenseToPrometheus(self,deviceid,sql_status,lightIntense):
            if(sql_status==200):

self.metricLightIntense.labels(instance=self.apiInstanceUrl,device_id=devicei
d).set(lightIntense)

                push_to_gateway(self.pushgatewayUrl,
job='api_monitoring_scrape', registry=self.registry)
```

Файл MCU.py

```
import json
from flask import abort

class MCU:
    deviceId = None;
    colorHex = None;
    lightIntense = None;
    status = None;
    errorCode = None;
```

```
upTime = None;
timestamp = None;
fMemBytes = None;
ledCount = None;

def setVerifyStatus(self, sql_result):
    if(sql_result[0]==200):
        self.errorCode= sql_result[0];
        self.deviceId = int(sql_result[1][0]);
        self.colorHex = str(sql_result[1][1]);
        self.lightIntense = int(sql_result[1][2]);
        self.status = str(sql_result[1][3]);
    elif(sql_result[0]==404):
        abort(sql_result[0])
        self.errorCode= sql_result[0];

def getVerifyStatus(self):
    return json.dumps({
        'deviceId': self.deviceId,
        'colorHex': self.colorHex,
        'lightIntense': self.lightIntense,
        'status':self.status
    })

def setMonitoringInput(self, deviceId, json_resp):
        self.deviceId=deviceId;
        self.upTime= str(json_resp['up']);
        self.timestamp= str(json_resp['timestamp']);
        self.ledCount= str(json_resp['ledCount']);
        self.fMemBytes=str(json_resp['fMemBytes']);

def getMonitoringStatus(self, sql_result):
    if(sql_result[0]==200):
        return json.dumps({
            'status': sql_result[0]
        });
    elif(sql_result[0]==404):
        abort(sql_result[0])
        self.errorCode= sql_result[0];

def getAllDeviceData(self,sql_result,errorCode):
    combineJson = None;
```

```
        if(len(sql_result)>0):
            combineJson = "{\"devices\":[";

            for row in sql_result:
                combineJson+="{\"deviceid\":"
                combineJson+=str(row[0]);
                combineJson+=str(",\"timesecdiff\":")
                combineJson+=str(row[1]);
                combineJson+=str(",\"uptime\":")
                combineJson+=str(row[2]);
                combineJson+=str(",\"fmembytes\":")
                combineJson+=str(row[3]);
                combineJson+=str(",\"ledlightcount\":")
                combineJson+=str(row[4]);
                combineJson+=str(",\"colorhex\":\"")
                combineJson+=str(row[5]);
                combineJson+=str("\",\"lightintense\":")
                combineJson+=str(row[6]);
                combineJson+=" },";
            combineJson = combineJson[:-1];
            combineJson+="],\"errorCode\": \"";
            combineJson+=str("200");
            combineJson+="\" }";
        else:
            combineJson=str("{\"errorCode\": \"404\"}");
        return combineJson;
```

Файл Metadata.py

```
import json

class Metadata:

    objName = None;
    sqlResult = None;
    errorCode = None;

    def __init__(self, obj_name, sql_result,errorCode):
        self.objName= obj_name;
```

```
        self.sqlResult = sql_result;
        self.errorCode=errorCode;

    def getFullCityList(self):
        combineJson = "{\"";

        for row in self.sqlResult:
            # print(row[1])
            combineJson+=str(row[0]);
            combineJson+="\": \"";
            combineJson+=str(row[1]);
            combineJson+="\", \"";

        combineJson = combineJson[:-1];
        combineJson+="\"errorCode\": \"";
        combineJson+=str(self.errorCode);
        combineJson+="\"}"

        return combineJson;
```

Файл User.py

```
import json

class User:
    username = None;
    userLoginStatus = None;
    userFname = None;
    userLname = None;
    userCityName = None;
    userRoleName = None;
    userPasswdHash = None;
    userId = None;
    usermail = None;

    def __init__(self,sql_result):
        if(sql_result[1]!="None"):
            self.userLoginStatus = sql_result[0];
            self.userId = sql_result[1][0];
            self.username = sql_result[1][1]
            self.usermail = sql_result[1][2];
            self.userFname = sql_result[1][3];
```

```
            self.userLname = sql_result[1][4];
            self.userRoleName = sql_result[1][5];
            self.userCityName = sql_result[1][6];
            self.userPasswdHash = sql_result[1][7];
        else:
            self.userLoginStatus = sql_result[0];

    def getUserInfo(self):
        return json.dumps({
            'userid': self.userId,
            'username': self.username,
            'usermail': self.usermail,
            'fname': self.userFname,
            'lname': self.userLname,
            'city': self.userCityName,
            'userrole': self.userRoleName,
            'passwd': self.userPasswdHash,
            'errorCode': self.userLoginStatus
        });
```

Файл requirements.txt

```
flask
configparser
psycopg2-binary
prometheus_client
```

Файл Dockerfile_API

```
FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt requirements.txt

RUN pip3 install -r requirements.txt

COPY main.py .
COPY Database.py .
COPY MCU.py .
COPY User.py .
COPY Metadata.py .

CMD ["python3", "main.py"]
```

Файл styles.css

```
body {
    background-color: #ffd500;
    margin: 0;
    font-family: Serif;
}

table {
    position: relative;
    background-color: white;
    border-color: black;
    text-align: center;
    border-collapse: collapse;
}

td,
th {
    padding: 10px;
    border: 2px solid black;
    font-weight: normal;
}

th {
    padding: 10px;
    background: rgb(57, 57, 57);
    border-radius: 0;
    color: white;
    top: 0;
    position: sticky;
}

tr {
    color: black;
}

.dashed_head{
    border: white 12px dashed;
}

.text_withlight{
    text-align: center;
```

```
        font-family: Serif;
        font-size: 8vh;
    }

    .login_container{
        background-color: white;
        border: 10;
        border-color: #ffffff;
        border-radius: 3%;
        width: 30%;
        height: 400px;
        margin: auto;
        text-align: center;
        overflow: auto;
    }

    .login_header{
        padding-left: 0.7em;
        padding-top: .5em;
        padding-bottom: .5em;
        background-color: rgb(203, 139, 18);
        font-size: 2em;
        text-align: left;

    }

    .profile{
        font-size: 30px;
        color: white;
        top: 55px;
        align-items: center;
        justify-content: center;
        position: relative;
        flex-direction: row;
        width: 100%;
    }
    .profile_header{
        text-align: center;
        margin: auto;
        padding-top: 10px;
        padding-bottom: 10px;

        width: 100%;
```

```
        background-color: #fef9e7;
        color: #5c0099;
    }

    .profile_data_header{
        padding-top: 25px;
        padding-left: 25px;

    }
    .profile_data_input{
        margin-left: 15px;
        padding-left: 15px;
        padding-top: .5em;
        padding-bottom: .5em;
        width: 50%;
        height: 30%;
        font-size: 2vh;
    }


    .login_input{
        margin: auto;
        padding-top: 20px;
        padding-bottom: 20px;

    }

    .form-control{
        padding-top: .5em;
        padding-bottom: .5em;
        width: 70%;
        height: 30%;
        font-family: Serif;
        font-size: 2vh;
    }

    .btn-submit{
        margin-top: 10%;
        width: 50%;
        font-family: Serif;
        font-size: 2vh;
    }
```

```
.main_header{
    position: fixed;
    top: 0px;
    left: 0px;
    right: 0px;
    z-index: 30;
    width: 100%;
    background-color: #005bbb;
    display: flex;
    flex-direction: row;

}

.btn_light_info,
.btn_admin_site,
.btn_user_info,
.btn_dashboard,
.btn_send_login,
.btn_save_changes,
.btn_air_alert,
.btn_air_safety{
    top: 10px;
    margin: 10px;
    color: #fff;
    text-decoration: none;
    user-select: none;
    background: rgb(57, 57, 57);
    padding: .7em 1.5em;
    outline: none;
}

.btn_light_info:hover,
.btn_admin_site:hover,
.btn_user_info:hover,
.btn_dashboard:hover,
.btn_send_login:hover,
.btn_save_changes:hover,
.btn_air_alert:hover,
.btn_air_safety:hover {
    background: rgba(232, 94, 76, 0.28);
    cursor: pointer;
}
```

```
.btn_light_info:active,
.btn_admin_site:active,
.btn_user_info:active
.btn_dashboard:active,
.btn_send_login:active,
.btn_save_changes:active,
.btn_air_alert:active,
.btn_air_safety:active{
    background: rgb(172, 92, 76);
}
.btn_save_changes{

    padding: 15px;
    margin: auto;
    margin-top: 15px;
    margin-bottom:  15px;
    width: fit-content;
    cursor: pointer;
}
.btn_send_login {
    cursor: pointer;
    margin-left: auto;
    margin-right: auto;
}

.text_about{
    background-color: white;
    border: 10;
    border-color: #ffffff;
    width: 70%;
    height: 400px;
    margin: auto;
    text-align: left;
    overflow: auto;
    font-family: Serif;
    margin-top: 10%;
    font-size: 1.3em;
    vertical-align: baseline    ;
}

.corner_fulldown {
    text-align: center;
    font-size: 30px;
```

```
        color: white;
        top: 55px;
        bottom: 0px;
        overflow-y: scroll;
        align-items: center;
        justify-content: center;
        position: fixed;
        padding: 8px 16px;
        flex-direction: column;
        }
    .corner_fulldown::-webkit-scrollbar {
        display: none;
    }

    .left {
        left: 0;
        width: 20%;
        background-color: white;
    }

    .right {
        right: 0;
        width: 80%;
    }

    .corner_fulldown_right {
        text-align: center;
        font-size: 30px;
        color: pink;
        padding: 10px;
        background-color: white;
        height: 100%;
        width: 20%;
    }

    .btn_left_corner
     {
        top: 10px;
        margin: 10px;
        text-decoration: none;
        user-select: none;
        background: rgb(57, 57, 57);
        padding: .7em 1.5em;
```

```
        outline: none;
    }


    .btn_left_corner:hover
    {
        background: rgba(232, 94, 76, 0.28);
        cursor: pointer;
    }

    .btn_light_info:active
    {
        background: rgb(172, 92, 76);
    }

    .mcu_list  {
        font-size: 3vh;
        border-color: black;
        border: 10;
        margin-left: auto;
        margin-right: auto;
        margin-top: 15px ;
    }
    .userinfo_list {
        width: 40%;
        margin: auto;
        text-align: left;
        overflow: auto;
        font-family: monospace;
        margin-top: 10%;
        font-size: 1.8em;
        vertical-align: baseline;
    }

    iframe{
        padding: 15px;
        margin: auto;
    }
    .grafana_integration{
        margin-left: 50px;
        margin-right: auto;
        margin-top: 70px;
        margin-bottom: 15px;
```

```
    display: block;
}
```

Файл index.js

```
const express = require('express');
const app = express();
const path = require('path');
const session = require('express-session');
var crypto = require("crypto");


let host = 'https://api.withlight.me'
let port = 443

app.use(express.json())

app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
app.use(express.static(path.join(__dirname, 'public')))
app.use(session({
    secret: process.env.PRIV_KEY_PEM,
    resave: false,
    saveUninitialized: true
}));

const headers = {
    'Authorization': `Bearer ${process.env.TOKEN_WITHLIGHT}`,
    'Content-Type': 'application/json'
};
let userdata;


app.get('/', (req, res) => {
    res.redirect('/login');
})

app.get('/login', (req, res) => {
    res.render('login', { name: 'Withlight.me' })
})
app.get('/main', (req, res) => {
    res.render('main', { name: 'Withlight.me' })
})
```

```
app.get('/main/light_info', (req, res) => {
    res.render('light_info', { name: 'Withlight.me' })
})

app.get('/main/user_info', (req, res) => {
    res.render('user_info', { name: 'Withlight.me' })
})

app.get('/main/admin_site', (req, res) => {
    res.render('admin_site', { name: 'Withlight.me' })
})

app.get('/main/dashboard', (req, res) => {
    res.render('dashboard', { name: 'Withlight.me' })
})

app.listen(3000, () => {
})

app.post('/withlight-rest-api/login', async (req, res) => {

    const jsonData = req.body;

    const encryptedPasswd = crypto.publicEncrypt({
        key: process.env.PUB_KEY_PEM,
        padding: crypto.constants.RSA_PKCS1_OAEP_PADDING,
        oaepHash: "sha256",
    }, Buffer.from(jsonData.password)).toString("base64");

    const POSTdata = {
        login: jsonData.login,
        password: encryptedPasswd
    };

    const apiUrl = host + ':' + port + '/withlight-rest-
api/login/username=' + jsonData.login;

    await fetch(apiUrl, {
        method: "PUT",
        headers: headers,
        body: JSON.stringify(POSTdata)
    }).then(response => response.json())
        .then((data) => {
```

```
                req.session.username = jsonData.login;
                userdata = data;
                res.status(data.errorCode).json({ status: data.errorCode });
            });
    });


    app.get('/withlight-rest-api/city-list', async (req, res) => {
        const apiUrl = host + ':' + port + '/withlight-rest-api/city-list';
        await fetch(apiUrl, {
            method: "GET",
            headers: headers
        }).then(response => response.json())
            .then((data) => {
                res.status(data.errorCode).json(data);
            });
    });

    app.post('/withlight-rest-api/device-list', async (req, res) => {
        const jsonData = req.body;
        const apiUrl = host + ':' + port + '/withlight-rest-api/device-list='+
jsonData.city;
        await fetch(apiUrl, {
            method: "GET",
            headers: headers
        }).then(response => response.json())
            .then((data) => {
                res.status(data.errorCode).json(data);
            });
    });

    app.get('/withlight-rest-api/userinfo', async (req, res) => {
        res.status(userdata.errorCode).json(userdata);
    });

    app.put('/withlight-rest-api/userinfo', async (req, res) => {
        const jsonData = req.body;

        const encryptedPasswd = crypto.publicEncrypt({
            key: PUB_KEY_PEM,
            padding: crypto.constants.RSA_PKCS1_OAEP_PADDING,
            oaepHash: "sha256",
        }, Buffer.from(jsonData.passwd)).toString("base64");
```

```
            const PUTdata = {
                fname: jsonData.fname,
                lname: jsonData.lname,
                email: jsonData.usermail,
                password: encryptedPasswd
            };
            const apiUrl = host + ':' + port + '/withlight-rest-
api/user/updateinfo=' + jsonData.username;

            await fetch(apiUrl, {
                method: "PUT",
                headers: headers,
                body: JSON.stringify(PUTdata)
            }).then(response => response.json())
                .then((data) => {
                    res.status(data.errorCode).json({ status: data.errorCode });
                });
            res.status(200).json(jsonData.userid);
        });

        app.get('/withlight-rest-api/alert/on', async (req, res) => {
            const apiUrl = host + ':' + port + '/mcu/monitoring/alert=on';
            await fetch(apiUrl, {
                method: "GET",
                headers: headers
            }).then(response => response.json())
                .then((data) => {
                    res.status(data.status).json(data);
                });
        });

        app.get('/withlight-rest-api/alert/off', async (req, res) => {
            const apiUrl = host + ':' + port + '/mcu/monitoring/alert=off';
            await fetch(apiUrl, {
                method: "GET",
                headers: headers
            }).then(response => response.json())
                .then((data) => {
                    res.status(data.status).json(data);
                });
        });
```

Файл light_info.js

```
let host = 'https://withlight.me'
let port = 443

const headers = {
    'Content-Type': 'application/json'
};

document.addEventListener('DOMContentLoaded', async function () {
    const apiUrl = host + ':' + port + '/withlight-rest-api/city-list';
    await fetch(apiUrl, {
        method: "GET"
    }).then(response => response.json())
        .then((data) => {
            var element = document.getElementById("city_list_fulldown")

            for (const [key, value] of Object.entries(data)) {
                if (`${ key }` != "errorCode"){
                    let link = document.createElement("div")
                    link.classList.add("btn_left_corner");
                    link.textContent = `${value}`
                    link.id = `city_${key}`
                    element.appendChild(link);
                }
            }
        });

})

window.onclick =  async function (event) {
    if (event.target.id.substring(0, 5) == 'city_') {
        const tablelist = document.getElementById('tablelist');

        const data = {
            city: event.target.textContent
        };

        const apiUrl = host + ':' + port + '/withlight-rest-api/device-
list';

        await fetch (apiUrl, {
```

```
                    method: "POST",
                    headers: headers,
                    body: JSON.stringify(data)
                }).then(response => response.json())
                    .then((data) => {
                        for (let i = tablelist.rows.length - 1; i >= 0; i--) {
                            tablelist.deleteRow(i);
                        }

                        if(data.errorCode == '200') {
                            var row = tablelist.insertRow(0);
                            row.insertCell(0).outerHTML = "<th>ID</th>"
                            row.insertCell(1).outerHTML  =  "<th>Data  refreshed,
sec.</th>"
                            row.insertCell(2).outerHTML = "<th>Uptime</th>"
                            row.insertCell(3).outerHTML   =   "<th>Free   memory,
bytes</th>"
                            row.insertCell(4).outerHTML   =   "<th>Light   count,
leds</th>"
                            row.insertCell(5).outerHTML = "<th>Color hex</th>"
                            row.insertCell(6).outerHTML       =       "<th>Light
intense</th>"

                        for (let i = 0; i < data.devices.length; i++) {
                            var row = tablelist.insertRow(i+1);

                            var cell1 = row.insertCell(0);
                            var cell2 = row.insertCell(1);
                            var cell3 = row.insertCell(2);
                            var cell4 = row.insertCell(3);
                            var cell5 = row.insertCell(4);
                            var cell6 = row.insertCell(5);
                            var cell7 = row.insertCell(6);

                            cell1.innerHTML = data.devices[i].deviceid;
                            cell2.innerHTML = data.devices[i].timesecdiff;
                            cell3.innerHTML = data.devices[i].uptime;
                            cell4.innerHTML = data.devices[i].fmembytes;
                            cell5.innerHTML = data.devices[i].ledlightcount;
                            cell6.innerHTML = data.devices[i].colorhex;
                            cell7.innerHTML = data.devices[i].lightintense;
                        }
                    }
```

```
                });
            }

        }
```

Файл admi_site.js

```
let host = 'https://withlight.me'
let port = 443


const headers = {
    'Content-Type': 'application/json'
};

window.onclick = async function (event) {
    if (!sessionStorage.getItem('isLoggedIn')) {
        document.location.href = "/login";
    }
    else {
    if (event.target.id == 'btn_air_alert') {
        const apiUrl = host + ':' + port + '/withlight-rest-api/alert/on';
        await fetch(apiUrl, {
            method: "GET"
        }).then(response => response.json())
            .then((data) => {
                alert("All light devices are now red color")
            });
    }
    else if (event.target.id == 'btn_air_safety') {
        const apiUrl = host + ':' + port + '/withlight-rest-
api/alert/off';
        await fetch(apiUrl, {
            method: "GET"
        }).then(response => response.json())
            .then((data) => {
                alert("All light devices are now white color")
            });
    }
}
```

```
}

document.addEventListener('DOMContentLoaded', async function () {
    if (!sessionStorage.getItem('isLoggedIn')) {
        document.location.href = "/login";
    }
})
```

Файл user_info.js

```
let host = 'https://withlight.me'
let port = 443

const headers = {
    'Content-Type': 'application/json'
};

document.addEventListener('DOMContentLoaded', async function () {
    if (!sessionStorage.getItem('isLoggedIn')) {
        document.location.href = "/login";
    }
    else{
    const apiUrl = host + ':' + port + '/withlight-rest-api/userinfo';
    await fetch(apiUrl, {
        method: "GET"
    }).then(response => response.json())
        .then((data) => {
            let profile_userid =
document.getElementById("profile_input_id");
            profile_userid.value = data.userid
            let profile_username =
document.getElementById("profile_input_username");
            profile_username.value = data.username
            let profile_usermail =
document.getElementById("profile_input_email");
            profile_usermail.value = data.usermail
            let profile_fname =
document.getElementById("profile_input_fname");
            profile_fname.value = data.fname
            let profile_lname =
document.getElementById("profile_input_lname");
            profile_lname.value = data.lname
```

```
                let                         profile_city                        =
document.getElementById("profile_input_city");
                profile_city.value = data.city
                let                       profile_userrole                      =
document.getElementById("profile_input_userrole");
                profile_userrole.value = data.userrole
            });
        }
    })

    window.onclick = async function (event) {
        if (event.target.className == 'btn_save_changes') {
            let                        profile_username                         =
document.getElementById("profile_input_username");
            let                        profile_usermail                         =
document.getElementById("profile_input_email");
            let                         profile_fname                           =
document.getElementById("profile_input_fname");
            let                         profile_lname                           =
document.getElementById("profile_input_lname")
            let                         profile_passwd                          =
document.getElementById("profile_input_password");

            const data = {
                username: profile_username.value,
                usermail: profile_usermail.value,
                fname: profile_fname.value,
                lname: profile_lname.value,
                passwd: profile_passwd.value
            };

            const apiUrl = host + ':' + port + '/withlight-rest-api/userinfo';
            fetch(apiUrl, {
                method: "PUT",
                headers: headers,
                body: JSON.stringify(data)
            }).then(response => response.json())
                .then((respdata) => {
                    alert(respdata.status)
                });

        }
```

```
}
```

Файл login.ejs

```
<body>
    <div class="content">
        <div class="dashed_head">
        </div>
        <h1 class="text_withlight">
            Withlight.me
        </h1>
        <div class="login_container">
            <div class="login_header">
                <strong>LOG IN</strong>
            </div>
            <div>
                <div class="login_input">
                    <input type="text" class="form-control"
id="input_username" name="name" placeholder="Username">
                </div>
                <div class="login_input">
                    <input type="password" class="form-control"
id="input_password" name="password" placeholder="Password">
                </div>
                <div class="btn btn-submit btn_send_login"
onclick="sendloginUser()">Log in</div>
            </div>
        </div>
        </div>
</body>
```

Файл admin_site.ejs

```
<body>
    <div class="content">
        <header class="main_header">
            <a href="/main/dashboard"
class="btn_dashboard">Dashboard</a>
            <a href="/main/light_info" class="btn_light_info">Light
info</a>
            <a href="/main/user_info" class="btn_user_info">User
info</a>
```

```
                <a    href="/main/admin_site"    class="btn_admin_site">Admin
site</a>
            </header>
            <div class="corner_fulldown left">
                <div id="city_list_fulldown">
                    <div  class="btn_air_alert"  id="btn_air_alert">Air  raid
alert</div>
                    <div    class="btn_air_safety"    id="btn_air_safety">Air
safety</div>
                </div>
            </div>
            <div class="corner_fulldown right">
            <div class="profile">
                <div class="profile_data_header">Username</div>
                <input        type="text"        class="profile_data_input"
id="profile_input_username" name="name" placeholder="Username">
                <div class="profile_data_header">Email</div>
                <input        type="text"        class="profile_data_input"
id="profile_input_email" name="name" placeholder="Email">
                <div class="profile_data_header">First Name</div>
                <input        type="text"        class="profile_data_input"
id="profile_input_fname" name="name" placeholder="First Name">
                <div class="profile_data_header">Last Name</div>
                <input        type="text"        class="profile_data_input"
id="profile_input_lname" name="name" placeholder="Last Name">
                <div class="profile_data_header">City Location</div>
                <input        type="text"        class="profile_data_input"
id="profile_input_city" name="name" placeholder="City Location">
                <div class="profile_data_header">User role</div>
                <input        type="text"        class="profile_data_input"
id="profile_input_userrole" name="name" placeholder="User role"
                    readonly>
                <div class="profile_data_header">New password</div>
                <input      type="password"      class="profile_data_input"
id="profile_input_password" name="name" placeholder="New password"
                    type="password">
                <div class="btn_save_changes">Add new user</div>
            </div>
            </div>

        </div>
    </body>
```

Файл dashboard.ejs

```
<body>
    <div class="content">
        <header class="main_header">
            <a       href="/main/dashboard"       class="btn_dashboard"
id="dashboard">Dashboard</a>
            <a      href="/main/light_info"      class="btn_light_info"
id="lightinfo">Light info</a>
            <a     href="/main/user_info"     class="btn_user_info">User
info</a>
            <a    href="/main/admin_site"    class="btn_admin_site">Admin
site</a>
        </header>
        <div class="grafana_integration">
            <iframe
                src="https://grafana.withlight.me"
                width="500" height="200" frameborder="0"></iframe>
            <iframe
                src="https://grafana.withlight.me"
                width="500" height="200" frameborder="0"></iframe>
            <iframe
                src="https://grafana.withlight.me"
                width="500" height="200" frameborder="0"></iframe>
            <iframe
                src="https://grafana.withlight.me"
                width="500" height="200" frameborder="0"></iframe>
            <iframe
                src="https://grafana.withlight.me"
                width="500" height="200" frameborder="0"></iframe>
            <iframe src="https://grafana.withlight.me"
                width="500" height="200" frameborder="0"></iframe>
        </div>

    </div>
</body>

</html>
```

Файл light_info.ejs

```
    <body>
        <div class="content">
            <header class="main_header">
                <a                                    href="/main/dashboard"
class="btn_dashboard">Dashboard</a>
                <a  href="/main/light_info"  class="btn_light_info"  >Light
info</a>
                <a     href="/main/user_info"     class="btn_user_info">User
info</a>
                <a   href="/main/admin_site"   class="btn_admin_site">Admin
site</a>
            </header    >
            <div class="corner_fulldown left">
                <div id="city_list_fulldown">
                </div>
            </div>
            <div class="corner_fulldown right">
                <div class=""></div>
                <table class="mcu_list" id="tablelist">
                </table>
            </div>

        </div>
    </body>
```

Файл light_info.ejs

```
    <body>
        <div class="content">
            <header class="main_header">
                <a                                    href="/main/dashboard"
class="btn_dashboard">Dashboard</a>
                <a   href="/main/light_info"   class="btn_light_info">Light
info</a>
                <a     href="/main/user_info"     class="btn_user_info">User
info</a>
                <a   href="/main/admin_site"   class="btn_admin_site">Admin
site</a>
            </header>
            <div class="profile">
                <div class="profile_header">Profile</div>
                <div class="profile_data_header">Username</div>
```

```
                <input        type="text"        class="profile_data_input"
id="profile_input_username" name="name" placeholder="Username" readonly>
                <div class="profile_data_header">Email</div>
                <input        type="text"        class="profile_data_input"
id="profile_input_email" name="name" placeholder="Email">
                <div class="profile_data_header">First Name</div>
                <input        type="text"        class="profile_data_input"
id="profile_input_fname" name="name" placeholder="First Name">
                <div class="profile_data_header">Last Name</div>
                <input        type="text"        class="profile_data_input"
id="profile_input_lname" name="name" placeholder="Last Name">
                <div class="profile_data_header">City Location</div>
                <input        type="text"        class="profile_data_input"
id="profile_input_city" name="name" placeholder="City Location" readonly>
                <div class="profile_data_header">User role</div>
                <input        type="text"        class="profile_data_input"
id="profile_input_userrole" name="name" placeholder="User role" readonly>
                <div class="profile_data_header">Identificator</div>
                <input        type="text"        class="profile_data_input"
id="profile_input_id" name="name" placeholder="Id" readonly>
                <div class="profile_data_header">New password</div>
                <input      type="password"      class="profile_data_input"
id="profile_input_password"     name="name"     placeholder="New     password"
type="password">
                <div class="btn_save_changes">Save changes</div>
            </div>
        </div>
    </body>
```

Файл Dockerfile_website

```
FROM node:20

WORKDIR /usr/src/app

COPY Full-stack/public ./public
COPY Full-stack/views ./views
COPY Full-stack/index.js ./
COPY Full-stack/package* ./

RUN npm install
```

```
EXPOSE 3000
CMD [ "node", "index.js" ]
```

Файл infostore.sql

```
CREATE USER withlight WITH PASSWORD '';

GRANT CONNECT ON DATABASE withlight TO withlight;
GRANT ALL PRIVILEGES ON SCHEMA api TO withlight;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA api TO withlight;
GRANT SELECT,UPDATE, INSERT ON ALL TABLES IN SCHEMA api TO withlight;

CREATE DATABASE withlight;

CREATE SCHEMA api;
CREATE TABLE api.device_status (
    device_id int not null,
    color_hex varchar(7) not null,
    light_intense varchar (3) not null,
    status boolean not null
);

CREATE TABLE api.device_monitoring (
    device_id int not null,
    up_time real not null,
    timestamp timestamp not null,
    f_mem_bytes int not null,
    led_light_count int not null
);

CREATE SCHEMA infostore;
GRANT ALL PRIVILEGES ON SCHEMA infostore TO withlight;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA infostore TO withlight;
GRANT SELECT,UPDATE, INSERT ON ALL TABLES IN SCHEMA infostore TO withlight;

CREATE TABLE infostore.user_role (
    roleid int not null primary key,
    rolename text not null
);

CREATE SEQUENCE user_role_sequence
  start 1000
  increment 1;
```

```
      CREATE                    TABLE                    infostore.city_info(
    cityid                  int           not     null     primary     key,
    cityname                  text                  not                  null
);

CREATE                        SEQUENCE                        city_info_sequence
  start                                                                  1000
  increment 1;

      CREATE             TABLE             infostore.user_info             (
    userid         int         not         null         primary         key,
    username                  text                  not                  null,
    usermail                  text                  not                  null,
    fname                   text                  not                  null,
    lname                   text                  not                  null,
    userrole        int        REFERENCES        infostore.user_role(roleid),
    citylocation       int       REFERENCES       infostore.city_info(cityid),
    passwdhash                text                  not                  null
);


CREATE                        SEQUENCE                        user_info_sequence
  start                                                                  1000
  increment 1;

      CREATE             TABLE             infostore.device_info             (
    deviceid         int         not         null         primary         key,
    deviceName                text                  not                  null,
    deviceDesc                text                  not                  null,
    citylocation       int       REFERENCES       infostore.city_info(cityid)
                              );


CREATE                        SEQUENCE                        device_info_sequence
  start                                                                    1
  increment 1;
```

Файл outbound_pip.tf

```
      resource "azurerm_public_ip" "outbound_pip_aks" {
        name                         = "pip-outbound-aks-${var.project}-compute-
${substr(var.location, 0, 6)}-${var.suffix}${var.enviroment}"
```

```
    resource_group_name = azurerm_resource_group.rg.name
    location            = azurerm_resource_group.rg.location
    allocation_method   = "Static"
    sku                 = "Standard"

    tags = {
      Enviroment = var.enviroment
      Project    = var.project
      Location   = var.location
      Subproject = var.subproject
    }
  }

  resource "azurerm_public_ip" "load_balancer_pip_aks" {
    name                  = "pip-load-balancer-aks-${var.project}-compute-
${substr(var.location, 0, 6)}-${var.suffix}${var.enviroment}"
    resource_group_name         =         "rg-${var.project}-compute-aks-
${substr(var.location, 0, 6)}-${var.suffix}${var.enviroment}"
    location            = azurerm_resource_group.rg.location
    allocation_method   = "Static"
    sku                 = "Standard"

    tags = {
      Enviroment = var.enviroment
      Project    = var.project
      Location   = var.location
      Subproject = var.subproject
    }

    depends_on = [
      azurerm_kubernetes_cluster.aks
    ]
  }
```

Файл identity.tf

```
  resource "azurerm_user_assigned_identity" "identity_general" {
    name               = "id-${var.project}-compute-${substr(var.location,
0, 6)}-${var.suffix}${var.enviroment}"
    resource_group_name = azurerm_resource_group.rg.name
    location            = var.location
  }
```

```
resource "azurerm_role_assignment" "contribrole" {
  scope                = azurerm_resource_group.rg.id
  role_definition_name = "Contributor"
  principal_id                                                    =
azurerm_user_assigned_identity.identity_general.principal_id
}

resource "azurerm_user_assigned_identity" "id_kubelet" {
  name                              =  "id-${var.project}-compute-kubelet-
${substr(var.location, 0, 6)}-${var.suffix}${var.enviroment}"
  resource_group_name = azurerm_resource_group.rg.name
  location            = azurerm_resource_group.rg.location
}
```

Файл aks.tf

```
  ######do not remove######
resource "tls_private_key" "aks_ssh" {
  algorithm = "RSA"
}

resource "azurerm_kubernetes_cluster" "aks" {
  name                                    = "aks-${var.project}-compute-
${substr(var.location, 0, 6)}-${var.suffix}${var.enviroment}"
  location                        = azurerm_resource_group.rg.location
  resource_group_name             = azurerm_resource_group.rg.name
  dns_prefix                    = "aks${var.project}${substr(var.location,
0, 6)}${var.suffix}${var.enviroment}"
  private_cluster_enabled       = false
  public_network_access_enabled = true
  node_resource_group                  =  "rg-${var.project}-compute-aks-
${substr(var.location, 0, 6)}-${var.suffix}${var.enviroment}"

  key_vault_secrets_provider {
    secret_rotation_enabled = false
  }

  default_node_pool {
    name = "default"
    node_labels = {
      label = "main"
    }
```

```
          node_count            = 2
          max_pods              = 110
          vm_size                 = "Standard_D2_v5" # 4 core, 8 RAM, OS temp
storage
          enable_auto_scaling   = false
          enable_node_public_ip = true
        }

        network_profile {
          load_balancer_sku  = "standard"
          network_plugin     = "azure"
          network_policy     = "azure"
          service_cidr   = "172.19.0.0/16"
          dns_service_ip = "172.19.0.10"
          load_balancer_profile {
            outbound_ip_address_ids = [azurerm_public_ip.outbound_pip_aks.id]
          }
        }

        linux_profile {
          admin_username = "devops"
          ssh_key {
            key_data = tls_private_key.aks_ssh.public_key_openssh
          }
        }

        identity {
          type         = "UserAssigned"
          identity_ids = [azurerm_user_assigned_identity.identity_general.id]
        }

        kubelet_identity {
          client_id                                                 =
azurerm_user_assigned_identity.id_kubelet.client_id
          object_id                                                 =
azurerm_user_assigned_identity.id_kubelet.principal_id
          user_assigned_identity_id                                 =
azurerm_user_assigned_identity.id_kubelet.id
        }

        tags = {
          Enviroment = var.enviroment
          Project    = var.project
```

```
    Location   = var.location
    Subproject = var.subproject
  }

  depends_on = [
    azurerm_user_assigned_identity.identity_general,
    azurerm_user_assigned_identity.id_kubelet,
    azurerm_role_assignment.contribrole
  ]

  lifecycle {
    ignore_changes = [
      default_node_pool[0].node_count,
    ]
  }
}

resource "azurerm_network_security_group" "nsg_aks_accept_all" {
  name                = "nsg-${var.project}-aks-${substr(var.location, 0,
6)}-${var.suffix}${var.enviroment}"
  location            = azurerm_resource_group.rg.location
  resource_group_name          =          "rg-${var.project}-compute-aks-
${substr(var.location, 0, 6)}-${var.suffix}${var.enviroment}"
  depends_on = [
    azurerm_kubernetes_cluster.aks
  ]
}

resource "azurerm_network_security_rule" "security_rule_outbound_aks" {
  name                        = "outboundaks"
  priority                    = 100
  direction                   = "Outbound"
  access                      = "Allow"
  protocol                    = "Tcp"
  source_port_range           = "*"
  destination_port_range      = "*"
  source_address_prefix       = "*"
  destination_address_prefix  = "*"
  resource_group_name                  =  "rg-${var.project}-compute-aks-
${substr(var.location, 0, 6)}-${var.suffix}${var.enviroment}"
  network_security_group_name                                          =
azurerm_network_security_group.nsg_aks_accept_all.name
}
```

```
resource "azurerm_network_security_rule" "security_rule_inbound_aks" {
  name                        = "inboundaks"
  priority                    = 100
  direction                   = "Inbound"
  access                      = "Allow"
  protocol                    = "Tcp"
  source_port_range           = "*"
  destination_port_range      = "*"
  source_address_prefix       = "*"
  destination_address_prefix  = "*"
  resource_group_name                 =  "rg-${var.project}-compute-aks-
${substr(var.location, 0, 6)}-${var.suffix}${var.enviroment}"
  network_security_group_name                                          =
azurerm_network_security_group.nsg_aks_accept_all.name
}
```

Файл 01-tf-state-files.tf

```
terraform {
  backend "azurerm" {
    resource_group_name  = "rg-terraform-state"
    storage_account_name = "terraformwithlight01p"
    container_name       = "terraformwithlight01p"
    key                  = "devtools-01p.terraform.tfstate"
  }
}
```

Файл load_balancer.tf

```
resource "kubernetes_ingress_v1" "kube_ingress_website_withlight_p" {
  wait_for_load_balancer = true
  metadata {
    name = "tf-${var.project}-${var.minproject}"
    labels = {
      app = "${var.project}-${var.minproject}"
    }

    annotations = {
      "kubernetes.io/ingress.class" = "nginx"
      "nginx.ingress.kubernetes.io/ssl-redirect" = "true"
```

```
            "nginx.ingress.kubernetes.io/session-cookie-name" = "route"
            "nginx.ingress.kubernetes.io/session-cookie-hash" = "sha1"
            "nginx.ingress.kubernetes.io/session-cookie-expires" = "172800"
            "nginx.ingress.kubernetes.io/session-cookie-max-age" = "172800"
            "nginx.ingress.kubernetes.io/affinity-mode" = "persistent"

          }
        }
        spec {
          tls {
            hosts = ["withlight.me"]
            secret_name = "tls-withlight-me"
          }
          rule {
            host = "withlight.me"
            http {
              path {
                path = "/withlight-rest-api/login"
                backend {
                  service {
                    name                                                    =
kubernetes_service.kube_svc_website_withlight_rest_api_p.metadata.0.name
                    port {
                      number = 3000
                    }
                  }
                }
              }
              path {
                path = "/withlight-rest-api/city-list"
                backend {
                  service {
                    name                                                    =
kubernetes_service.kube_svc_website_withlight_rest_api_p.metadata.0.name
                    port {
                      number = 3000
                    }
                  }
                }
              }
              path {
                path = "/withlight-rest-api/device-list"
                backend {
```

```
                service {
                  name                                                    =
kubernetes_service.kube_svc_website_withlight_rest_api_p.metadata.0.name
                  port {
                    number = 3000
                  }
                }
              }
            }
            path {
              path = "/withlight-rest-api/userinfo"
              backend {
                service {
                  name                                                    =
kubernetes_service.kube_svc_website_withlight_rest_api_p.metadata.0.name
                  port {
                    number = 3000
                  }
                }
              }
            }
            path {
              path = "/withlight-rest-api/alert/on"
              backend {
                service {
                  name                                                    =
kubernetes_service.kube_svc_website_withlight_rest_api_p.metadata.0.name
                  port {
                    number = 3000
                  }
                }
              }
            }
            path {
              path = "/withlight-rest-api/alert/off"
              backend {
                service {
                  name                                                    =
kubernetes_service.kube_svc_website_withlight_rest_api_p.metadata.0.name
                  port {
                    number = 3000
                  }
                }
```

```
                }
              }
              path {
                path = "/"
                backend {
                  service {
                    name                                                    =
kubernetes_service.kube_svc_website_withlight_p.metadata.0.name
                    port {
                      number = 3000
                    }
                  }
                }
              }
            }
          }
        }
      }
```

Файл local.tf

```
      locals {
        resource_group_name                                                 =
data.terraform_remote_state.compute_aks.outputs.aks_resource_group_name_p

        kube_host                                                           =
data.terraform_remote_state.compute_aks.outputs.kube_config_nonraw_p.0.host
        kube_username                                                       =
data.terraform_remote_state.compute_aks.outputs.kube_config_nonraw_p.0.userna
me
        kube_password                                                       =
data.terraform_remote_state.compute_aks.outputs.kube_config_nonraw_p.0.passwo
rd
        kube_cluster_ca_certificate                                         =
base64decode(data.terraform_remote_state.compute_aks.outputs.kube_config_nonr
aw_p.0.cluster_ca_certificate)
        kube_client_key                                                     =
base64decode(data.terraform_remote_state.compute_aks.outputs.kube_config_nonr
aw_p.0.client_key)
        kube_client_certificate                                             =
base64decode(data.terraform_remote_state.compute_aks.outputs.kube_config_nonr
aw_p.0.client_certificate)
```

```
    load_balancer_pip_aks                                                    =
data.terraform_remote_state.compute_aks.outputs.load_balancer_pip_aks.0

}
```

Файл script.sh

```
#!/bin/sh
if  [ ! -f /mnt/azurefile/grafana.ini ]
then
  cp /var/lib/grafana/* /mnt/azuredisk/
  cp /etc/grafana/* /mnt/azurefile/
  cp /tmp/grafana.ini /mnt/azurefile/grafana.ini

  chmod 700 -R /mnt/azurefile
  echo "Changed recursively chmod +x to /mnt/azurefile and copy file"
  chmod 700 -R /mnt/azuredisk
  echo "Changed recursively chmod +x to /var/lib/grafana/ and copy file"

  chown -R 1000:1000 /mnt/azurefile
  echo "Changed recursively to 1000:1000 for /mnt/azurefile"
  chown -R 1000:1000 /mnt/azuredisk
  echo "Changed recursively to 1000:1000 for /mnt/azuredisk"

else
  echo "Do nothing"
fi
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ХМАРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ОСВІТЛЕННЯ МІСТА

Програма та методика тестування

КПІ.ІП-9604.045430.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Ірина ВІТКОВСЬКА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Олексій БУРЯТОВ

Київ – 2023

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ 154
2 МЕТА ТЕСТУВАННЯ..... 155
3 МЕТОДИ ТЕСТУВАННЯ 156
4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ 157

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є хмарне програмне забезпечення для моніторингу освітлення міста. Випробування проводиться для UNIX-подібних операційних систем та для мікроконтролерів Arduino UNO, ESP8266 та LED стрічки WS8212B. Тестується правильність отримання та відправлення даних до API менеджера, коректність відображення показників та коректність реагування на режим «Небезпеки».

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка отримання даних від API менеджера;

– перевірка сумісності веб-додатку з останніми версіями сучасних браузерів (Chrome, Opera, Firefox);

– перевірка сумісності застосунку з різними UNIX-подібними системами

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

– тестування на відповідність функціональним вимогам;

– тестування на виведення повідомлень про помилку, коли це необхідно;

– тестування працездатності програми у випадку відсутності з’єднання до мережі;

– тестування інтерфейсу користувача;

– тестування зручності використання;

– тестування отримання повідомлень у разі недоступності освітлювальних приладів;

– тестування захищеності передачі даних за допомогою Bearer токену та SSL сертифікатів;

– тестування увімкнення та вимкнення режиму «Небезпеки» на освітлювальних приладах

– тестування отримання та передачі даних через мікроконтролери та API менеджер, використовуючи сховища даних.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ХМАРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ОСВІТЛЕННЯ МІСТА

Керівництво користувача

КПІ.ІП-9604.045430.05.34

“ПОГОДЖЕНО”
Керівник проекту:
_____ Ірина ВІТКОВСЬКА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Олексій БУРЯТОВ

Київ – 2023

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ 160
2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ. 163
2.1 Системні вимоги для коректної роботи 163
2.2 Завантаження застосунку 163
2.3 Перевірка коректної роботи 164
3 ВИКОНАННЯ ПРОГРАМИ 165

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

Хмарне програмне забезпечення для моніторингу освітлення міста – це комплекс програм, яке:

– призначене для програмування мікроконтролерів типу Arduino UNO, WiFi модулю ESP8266 та LED стрічки WS8212B;

– описує ETL механізм передачі та отримання даних з використанням сховища даних;

– призначене для розробки веб-застосунку для моніторингу освітлення міста.

Дане програмне забезпечення націлене на взаємодію з хмарним провайдером Microsoft Azure, що дозволяє утримувати всі компоненти програмного забезпечення в хмарі з метою відмови від наземних рішень збереження даних та додаткового захисту від втрати даних.

Програмне забезпечення складається з двох частин – програмний код, розроблений безпосередньо для програмування мікроконтролерів, сховища даних, API менеджера для веб-застосунку та програмний код, розроблений для побудови компонентів у приватній хмарі Microsoft Azure з доступом до мережі Інтернет через балансувальник даних. Перша частина складається з файлів типу.py, .js, .ejs, .ino, .css, .sql. На рисунку 1.1 зобрежена файлова структура першої частини.

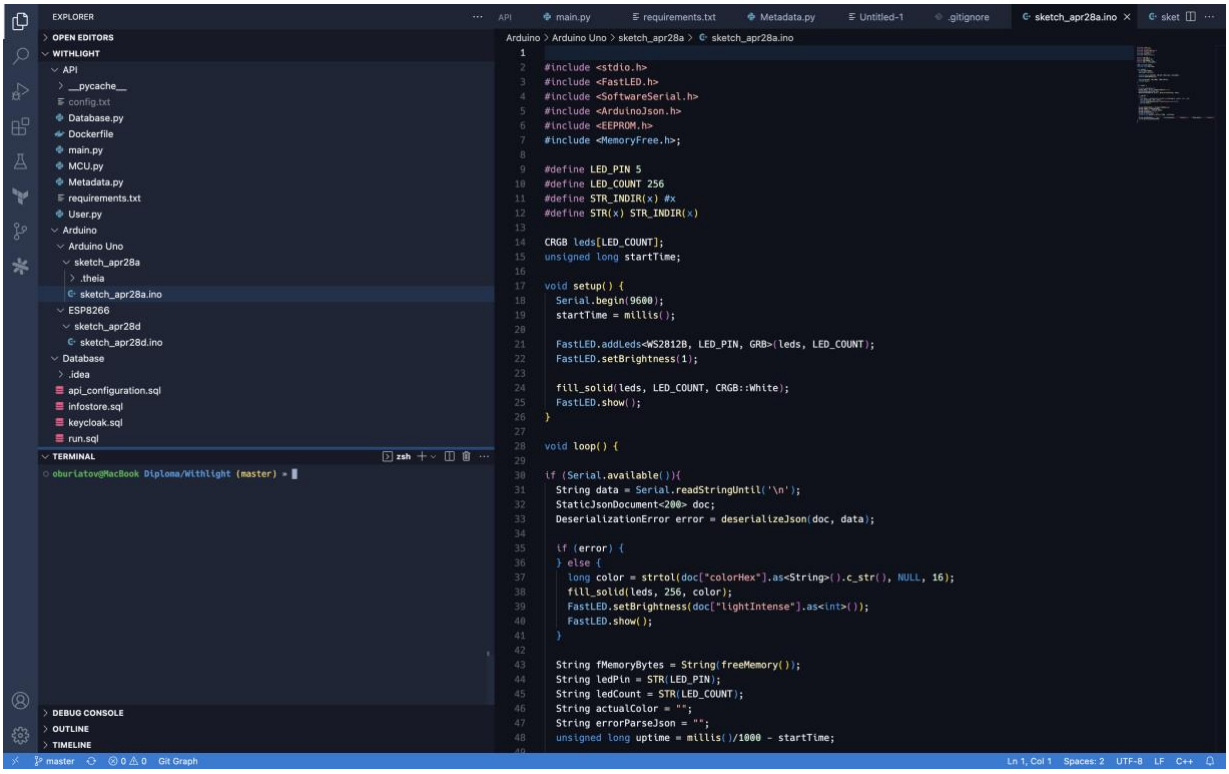


Рисунок 1.1 – Файлова структура програмного забезпечення

До другої частини відносяться файли типу .tf, що написані для використання методологією Інфраструктура як код (Iac) за допомогою Terraform (рисунок 1.2):

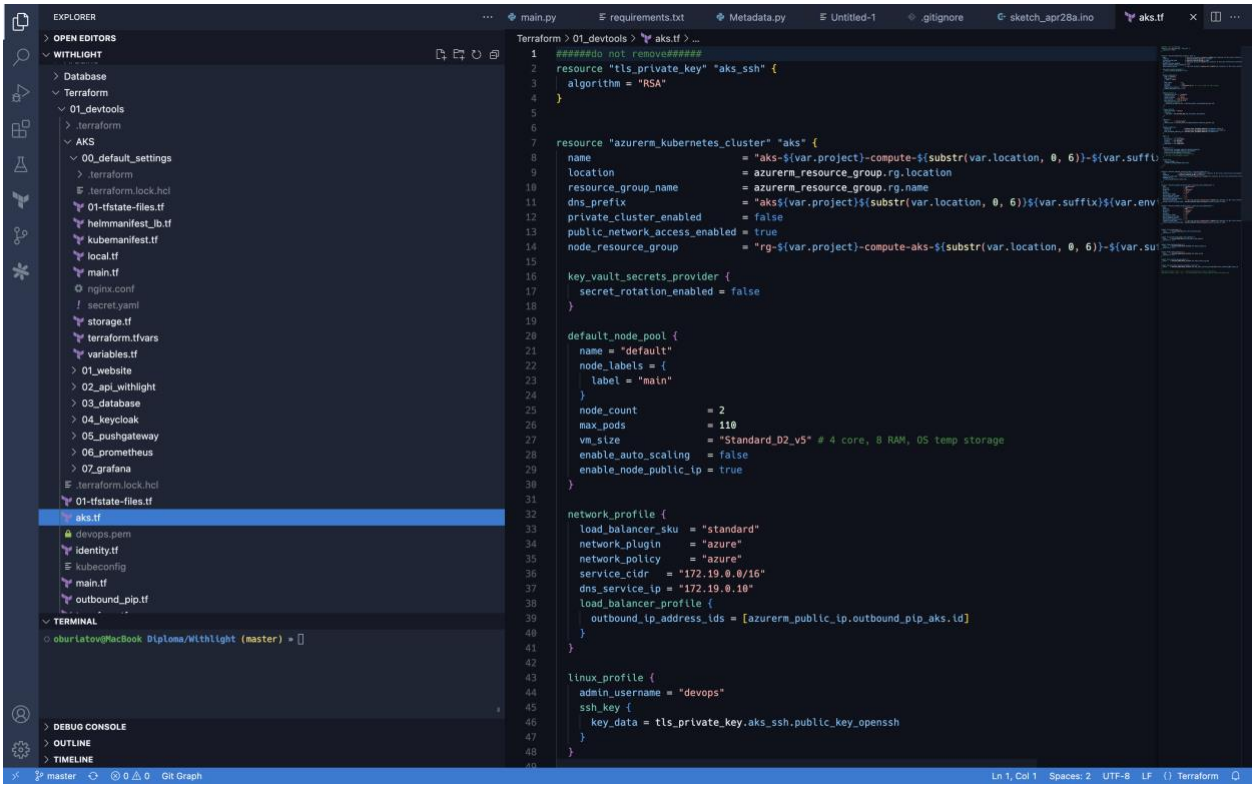


Рисунок 1.2 – Файлова структура інструменту для розгортання програмного забезпечення та компонентів Terraform

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для того, щоб програмне забезпечення для мікроконтролерів працювало коректно та безперебійно, необхідно:

– наявність вільної ROM пам’яті не менше 32 КБ та RAM пам’яті не менше 2 КБ;

– наявність доступу до Інтернету;

Для того, щоб програмне забезпечення, яке розгортається в хмарі, працювало коректно та безперебійно, необхідно:

– наявність дійсної підписки в приватній хмарі;

– використання віртуальних машин для побудови кластеру Kubernetes з наступними характеристиками: 4 CPU, 8 RAM та щонайменше 50 ГБ вільного простору;

– наявність доменного імені та балансувальника даних.

Для того, щоб веб-застосунок, працював коректно та безперебійно, необхідно:

– наявність персонального комп’ютеру або ноутбуку з будь-якою операційною системою;

– наявність веб-браузера;

– наявність доступу до мережі Інтернет.

2.2 Завантаження застосунку

Для того, щоб запустити веб сторінку програмного забезпечення на персональному комп’ютері достатньо встановити веб-браузер та перейти за посиланням на сторінку веб-застосунку.

Для того, щоб запустити програмне забезпечення, встановлене на мікроконтролерах, достатньо мати елемент живлення у вигляді підживлювача з напругою 2V та 5A та кабель типу USB A до micro USB або зарядний кабель з DC роз’ємом та напругою 7-12V та 1A.

2.3 Перевірка коректної роботи

По завершенню встановлення програмного забезпечення до мікроконтролерів, світло на LED стрічці типу WS8212B засвітиться тьмяним білим кольором. Після кожного перезапуску програмного забезпечення, освітлення буде загорятися білим кольором, допоки актуальна інформація з API менеджера не буде взята. Також, одразу після налаштування контакту між мікроконтролерами та освітлювальними приладами, на графіках буде відображено обнулення метрик, як-от на рисунку 2.1.

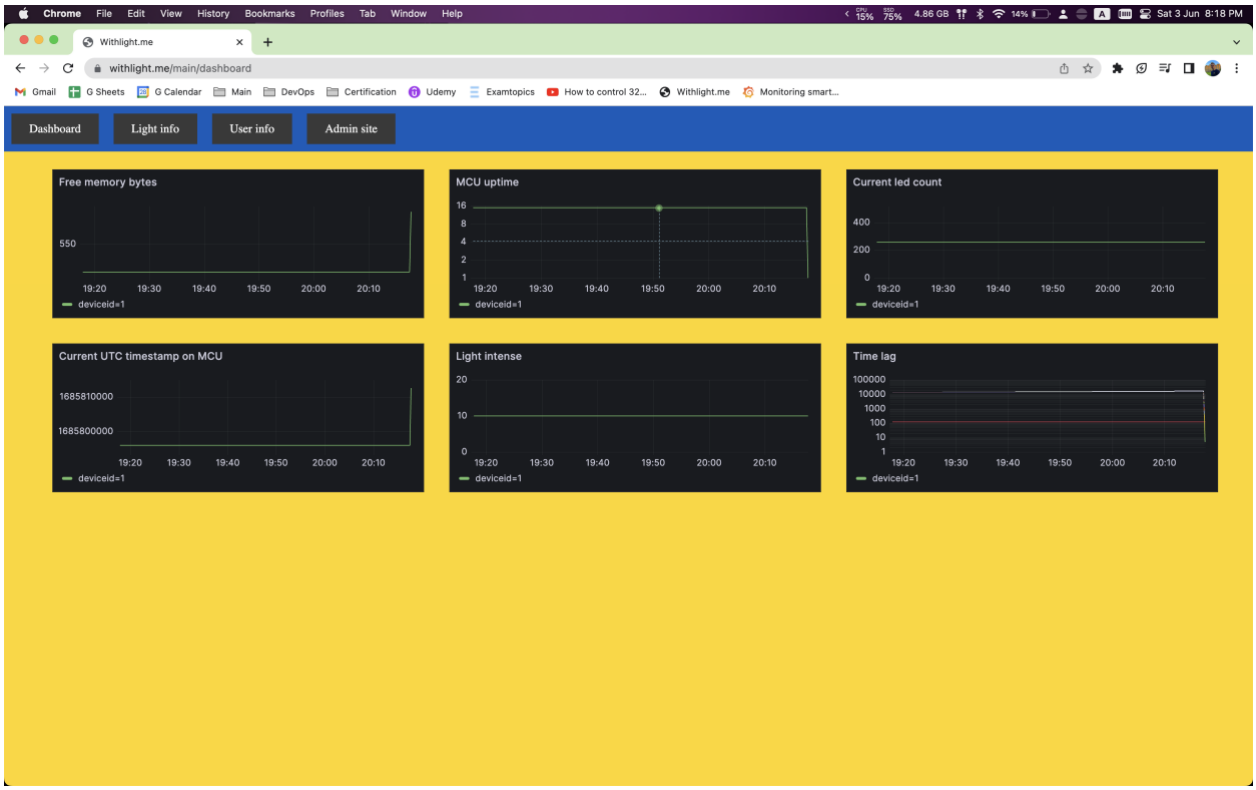


Рисунок 2.1 – Обнулення метрик після перезавантаження мікроконтролерів та відображення даних на графіках

3 ВИКОНАННЯ ПРОГРАМИ

3.1 Керівництво користувача

Для перегляду стану за освітлювальними приладами, необхідно перейти на веб-сторінку застосунку. Після переходу, користувачеві запропонують авторизуватися (рисунок 3.1).

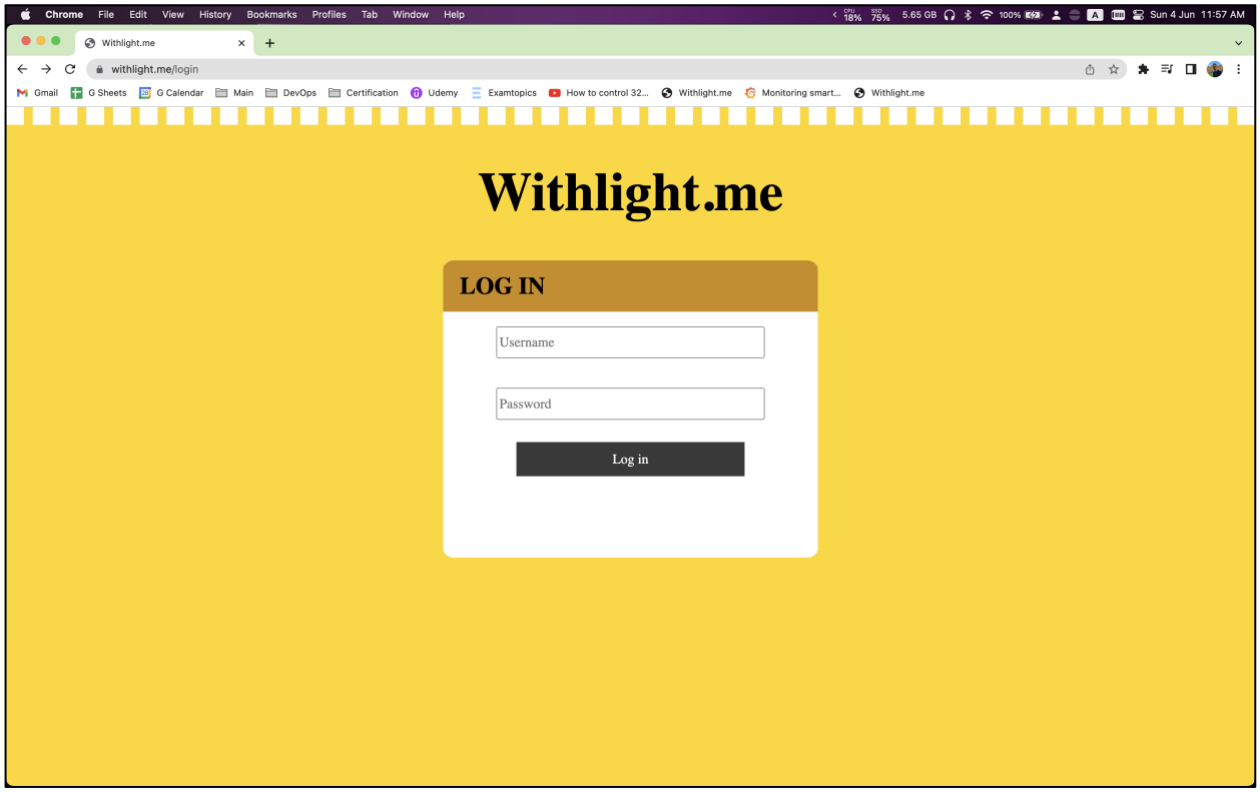


Рисунок 3.1 – Авторизація користувача до системи

Користувач не може самостійно зареєструватися, адже ця система є потенційно критичною і неправомірний доступ до інформації є недопустимою ситуацією. Користувач вводить логін і пароль і натискає кнопку «Log in» (рисунок 3.2).

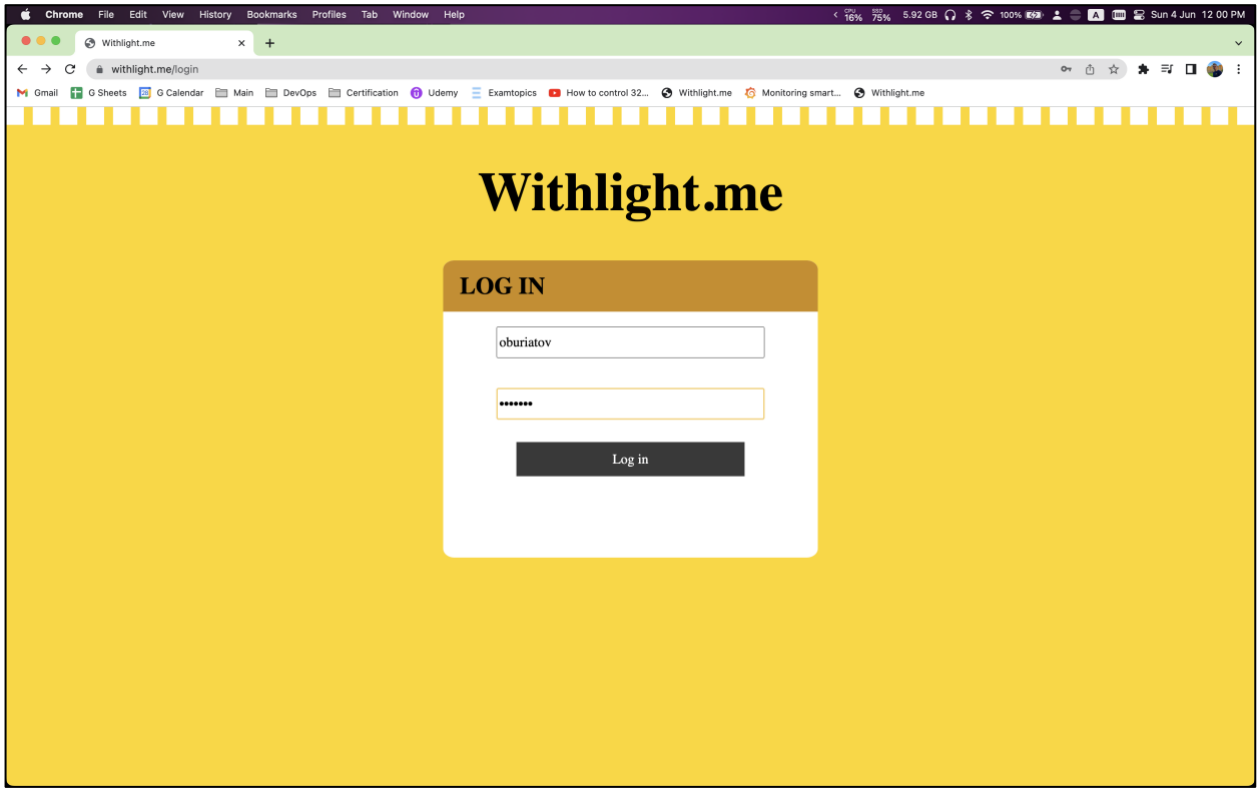


Рисунок 3.2 – Введений логін та пароль на сторінці авторизації

Після успішної авторизації до системи, користувач одразу опиняється на сторінці перегляду графіків та досліджує інформацію (рисунок 3.3).

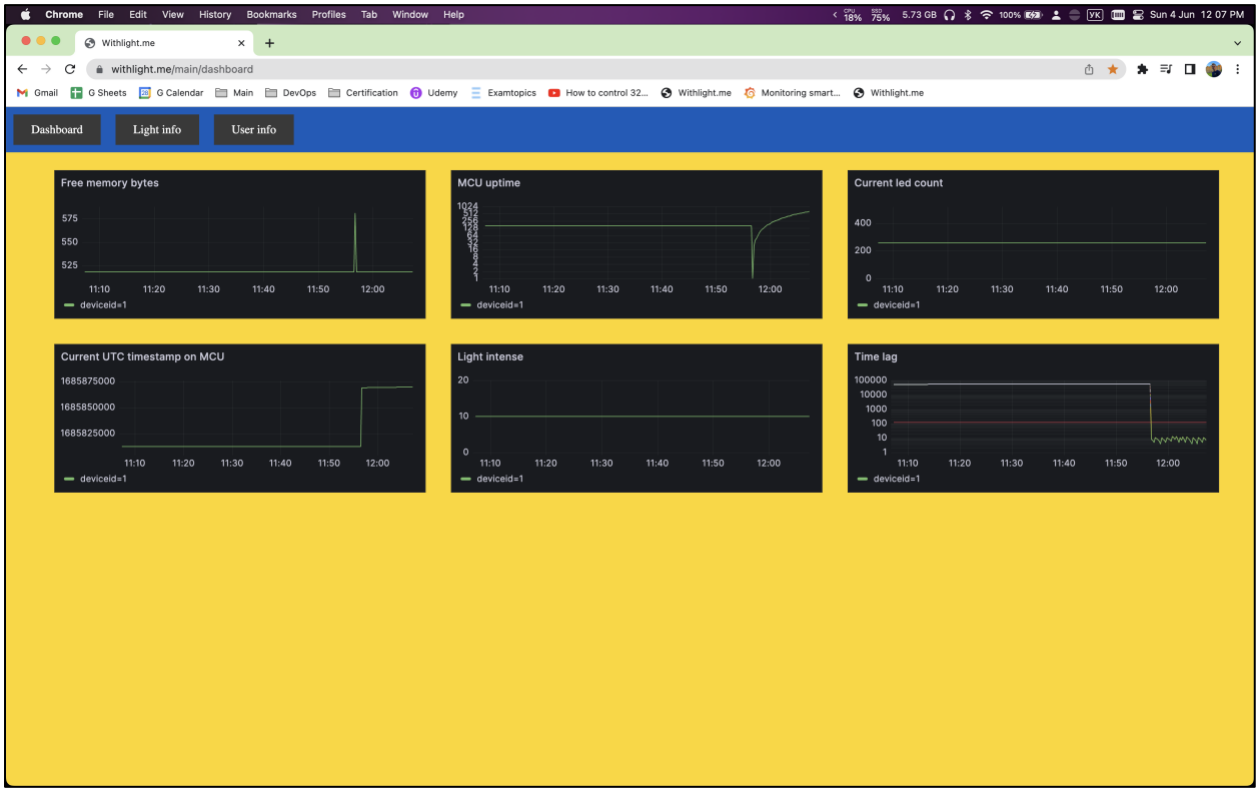


Рисунок 3.3 – Перегляд графіків поточного стану та в розрізі часу за всіма приладами

Для того, щоб користувач міг переглянути актуальну інформацію в розрізі міста, за яким користувач закріплений, необхідно натиснути на панелі керування кнопку «Light info». Перед користувачем зліва буде відображатися перелік усіх міст України (рисунок 3.4)

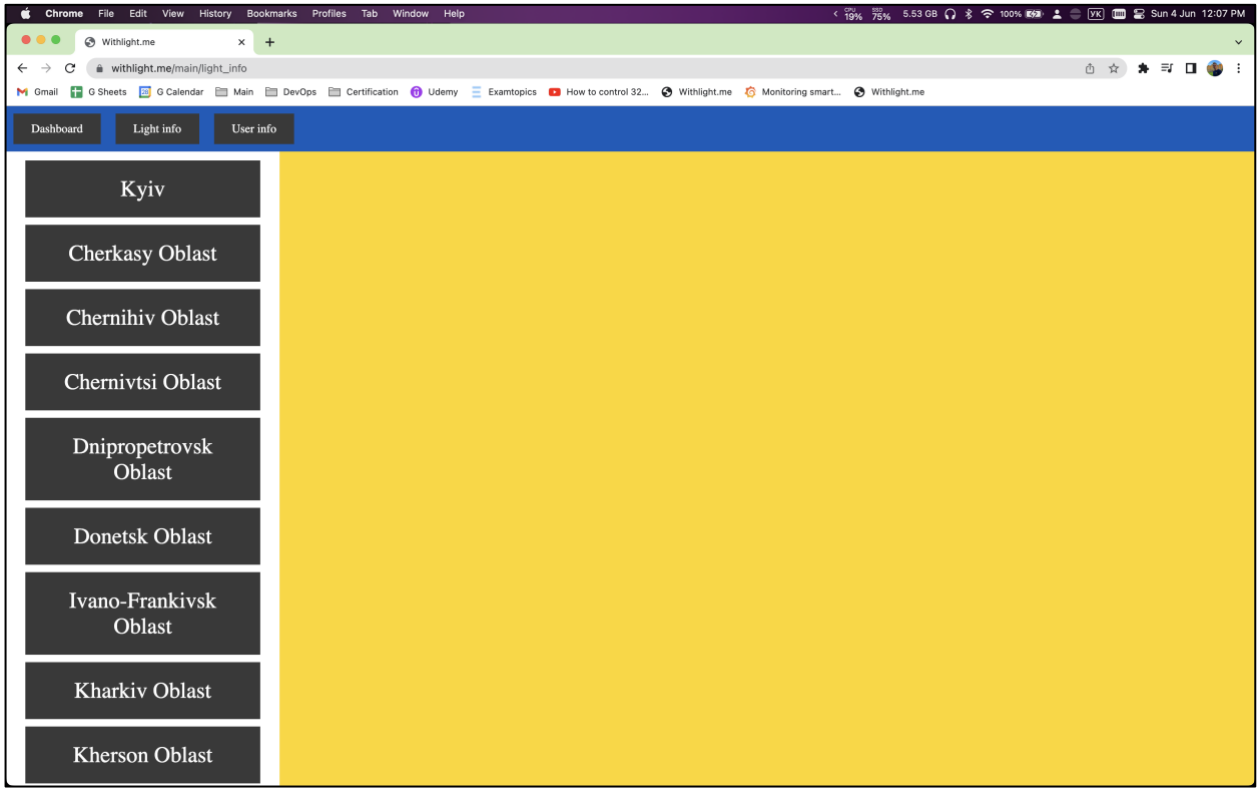


Рисунок 3.4 – Перелік міст України на вкладці «Light info»

Користувач натискає на кнопку того міста, за яким він закріплений у системі (далі буде описано як знайти назву міста за яким користувач закріплений). На рисунку 3.5 зображено найактуальнішу інформацію у за всіма пристроями міста Київ.

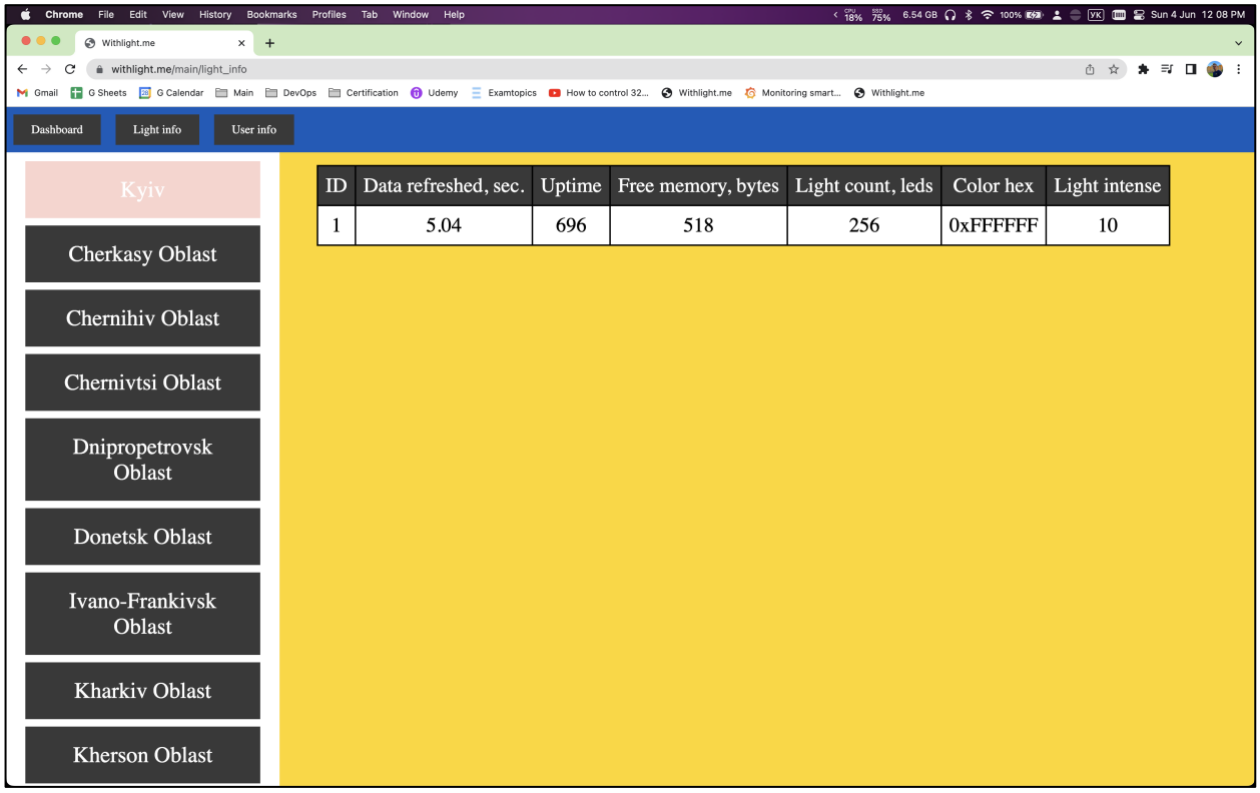


Рисунок 3.5 – Актуальна інформація по місту Київ

На попередньому рисунку можна дізнатися інтенсивність освітлення, колір освітлення, час роботи освітлювального приладу тощо.

Для того, щоб користувач міг переглянути персональну інформацію, достатньо на панелі керування знайти кнопку «User info». Користувачеві відкриється сторінка з персональною інформацією (рисунок 3.6).

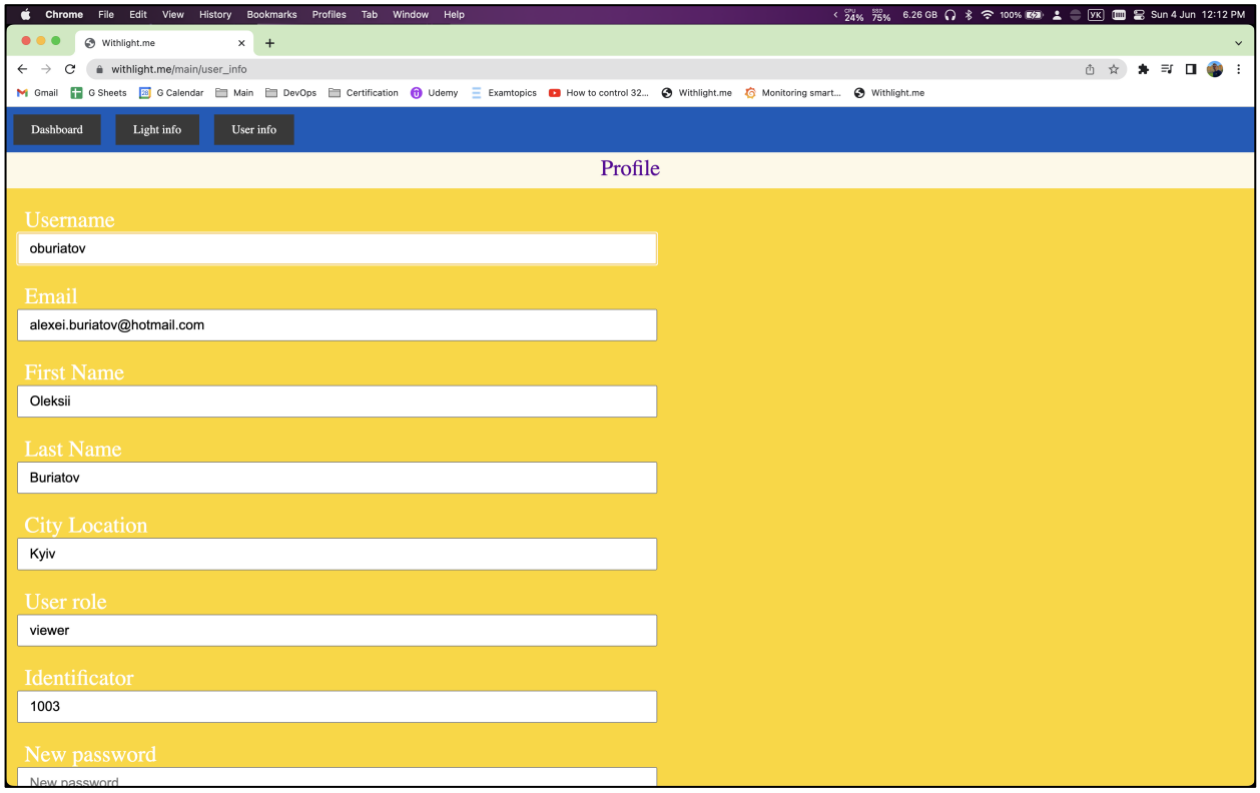


Рисунок 3.6. Персональна інформація користувача в системі

У разі, якщо користувач хоче змінити персональну інформацію, таку як: прізвище або ім’я або пошту – для нього активні дані поля і він може оновити інформацію. Обов’язковою умовою є введення нового паролю при кожній зміні персональної інформації. Наприклад, якщо користувач з нікнеймом «oburiatov» хоче змінити пошту – достатньо вписати нову пошту та ввести пароль і натиснути кнопку «Save changes» (рисунок 3.7)

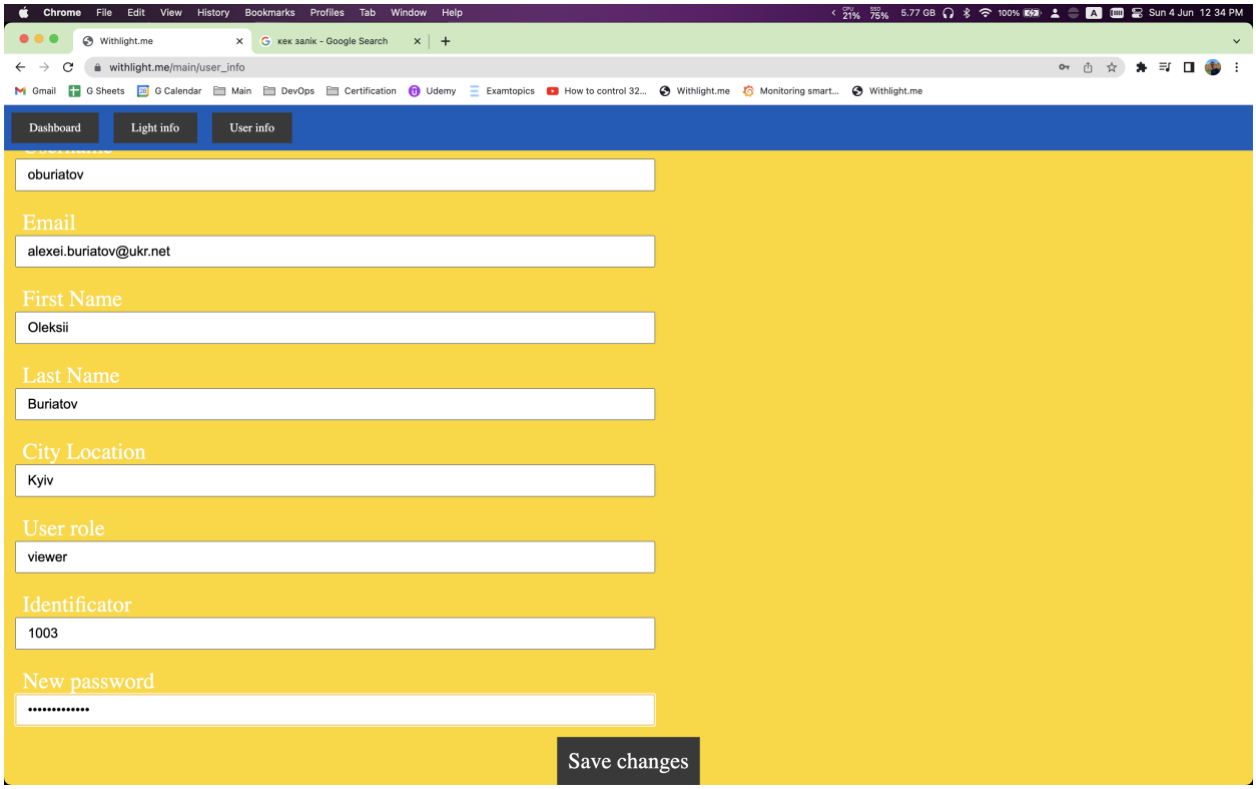


Рисунок 3.7. Оновлення пошти користувача «oburiatov»

Оновлена інформація на рисунку 3.8

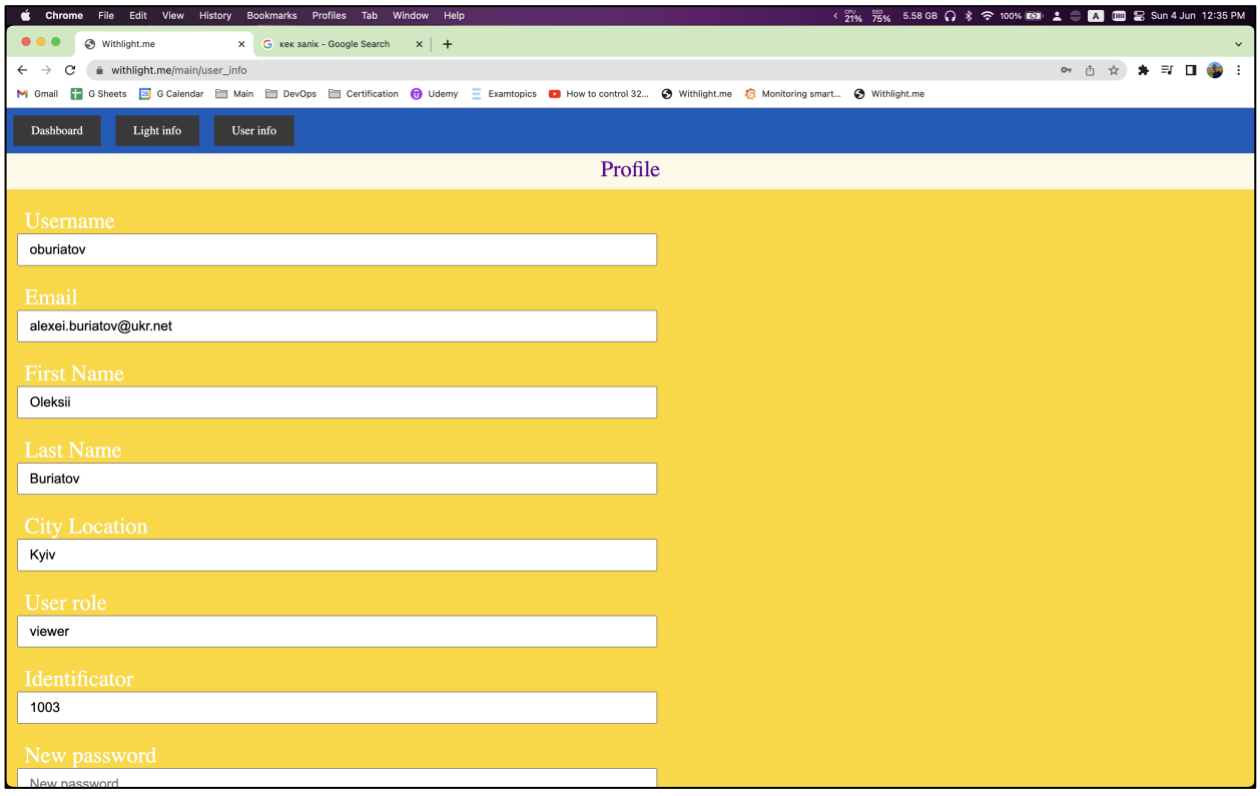


Рисунок 3.8. Актуальна інформація по користувачеві «oburiatov»

3.2 Керівництво адміністратора

У даному програмному забезпеченні є градація прав користувачів – роль адміністратора та звичайного користувача (переглядач). Якщо в користувача є права адміністратора – йому будуть доступні деякі додаткові функції. На рисунку 3.9. адміністратор авторизується в систему.

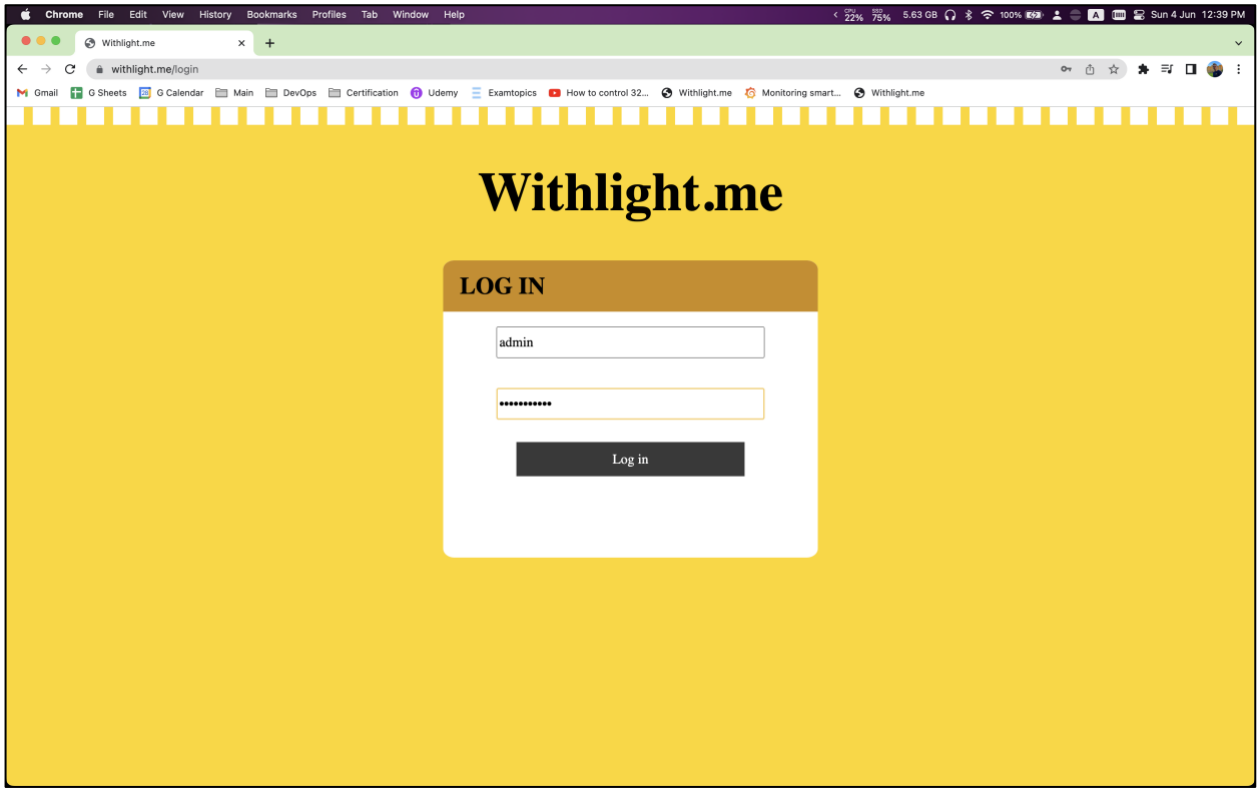


Рисунок 3.9. Авторизація користувача з правами адміністратора до системи

При успішній авторизації, адміністратор матиме додаткову кнопку «Admin site» на панелі керування (рисунок 3.10)

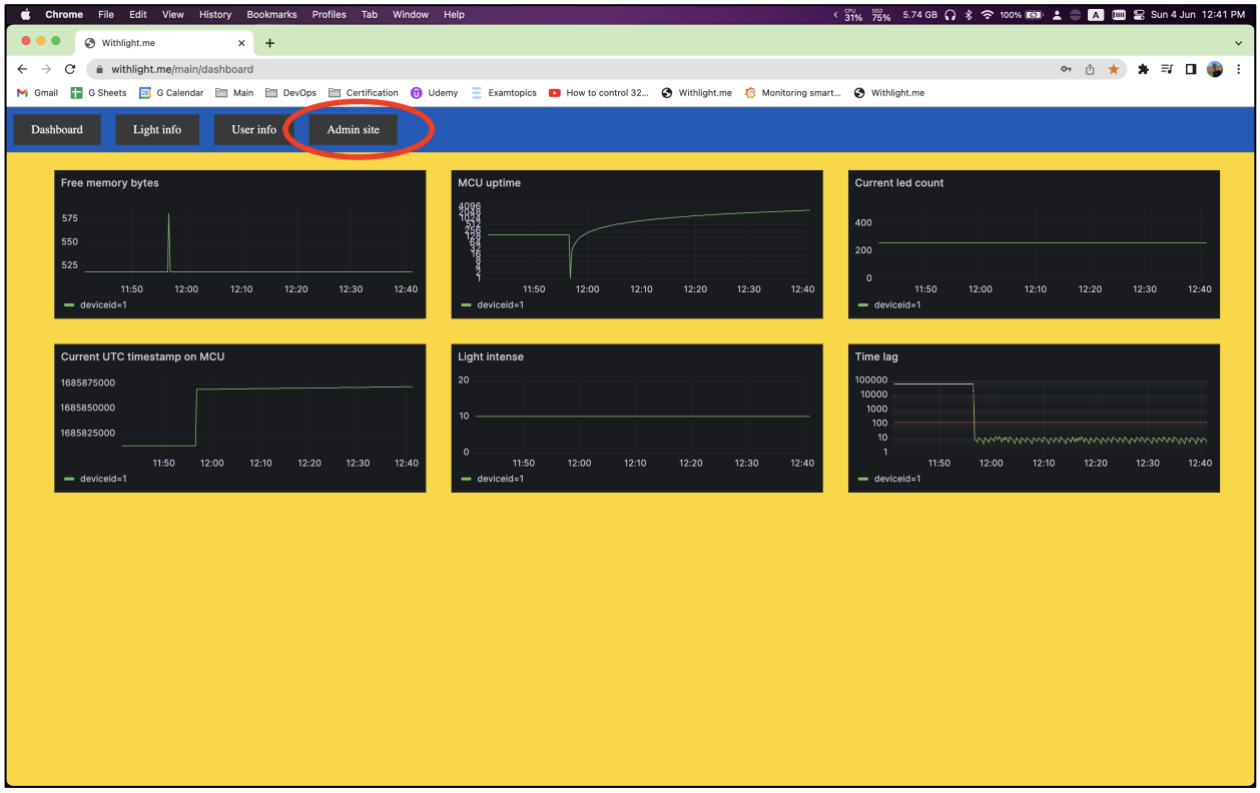


Рисунок 3.10. Додатковий функціонал для адміністратора системи

Для того, щоб переконатися, що даний користувач має права адміністратора, достатньо перейти на сторінку з персональною інформацією, натиснувши кнопку «User info»

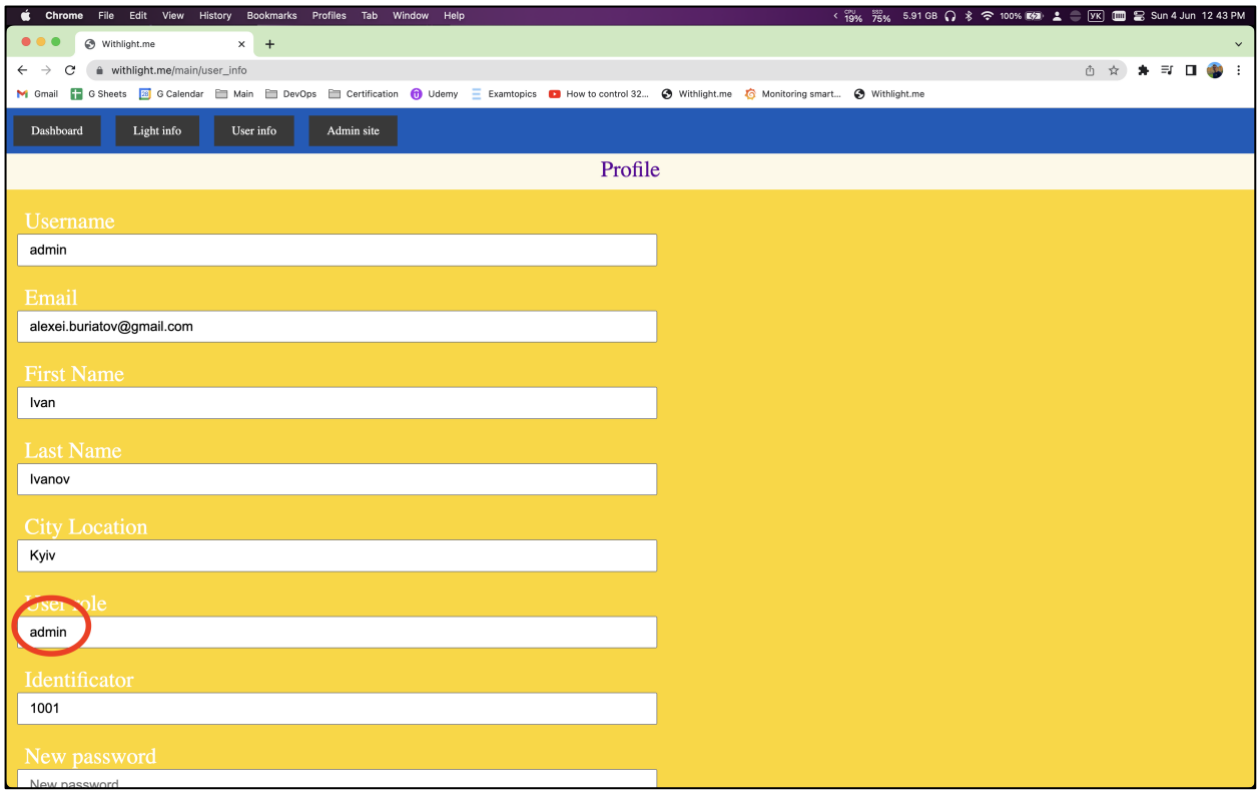


Рисунок 3.11. Перегляд персональної інформації адміністратора

Для того, щоб скористатися додатковим функціоналом адміністратора, необхідно натиснути кнопку «Admin site». Користувач перейде на сторінку адміністратора (рисунок 3.12).

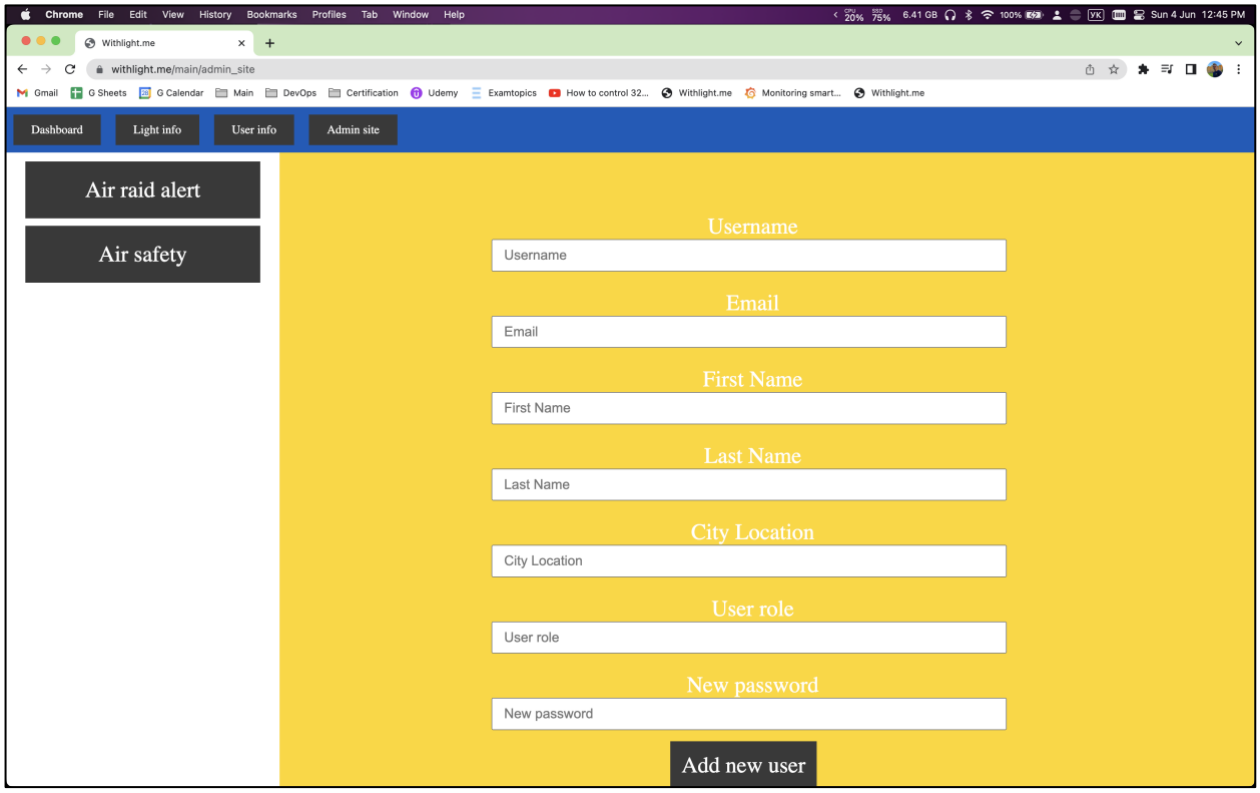


Рисунок 3.12. Сторінка адміністратора

На даній сторінці адміністратору доступно декілька дій: створення нового користувача, а також увімкнення та вимкнення режиму «Небезпеки». Режим «Небезпеки» - це режим освітлювальних приладів, при якому вони починають світитися яскраво червоним кольором. Натиснемо кнопку «Air raid alert». На рисунку 3.13 зображено стан освітлювального приладу.

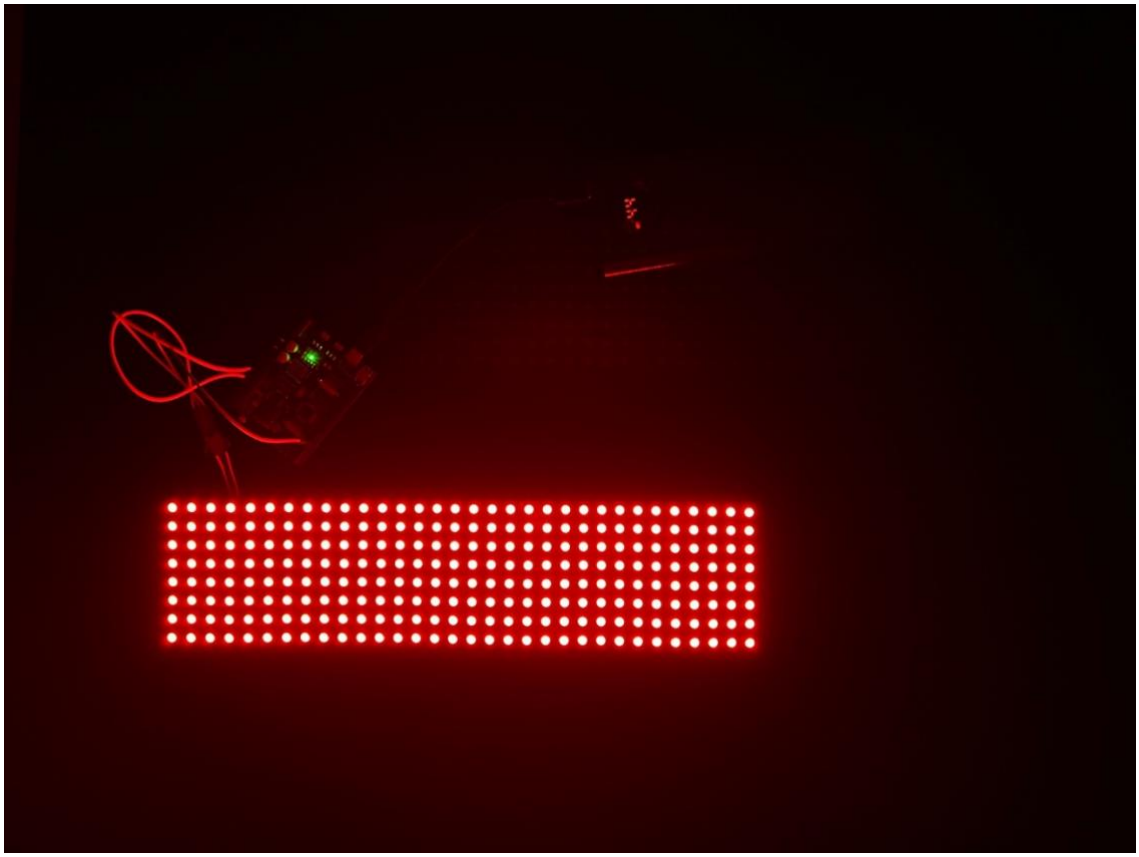


Рисунок 3.13. Увімкнений режим «Небезпеки» на освітлювальному приладі

Щоб перевірити оновлену інформацію на панелі світлових приладів, необхідно на сторінці «Light info» натиснути на місто Київ та переконатися, що колір освітлювального приладу змінився (рисунок 3.14)

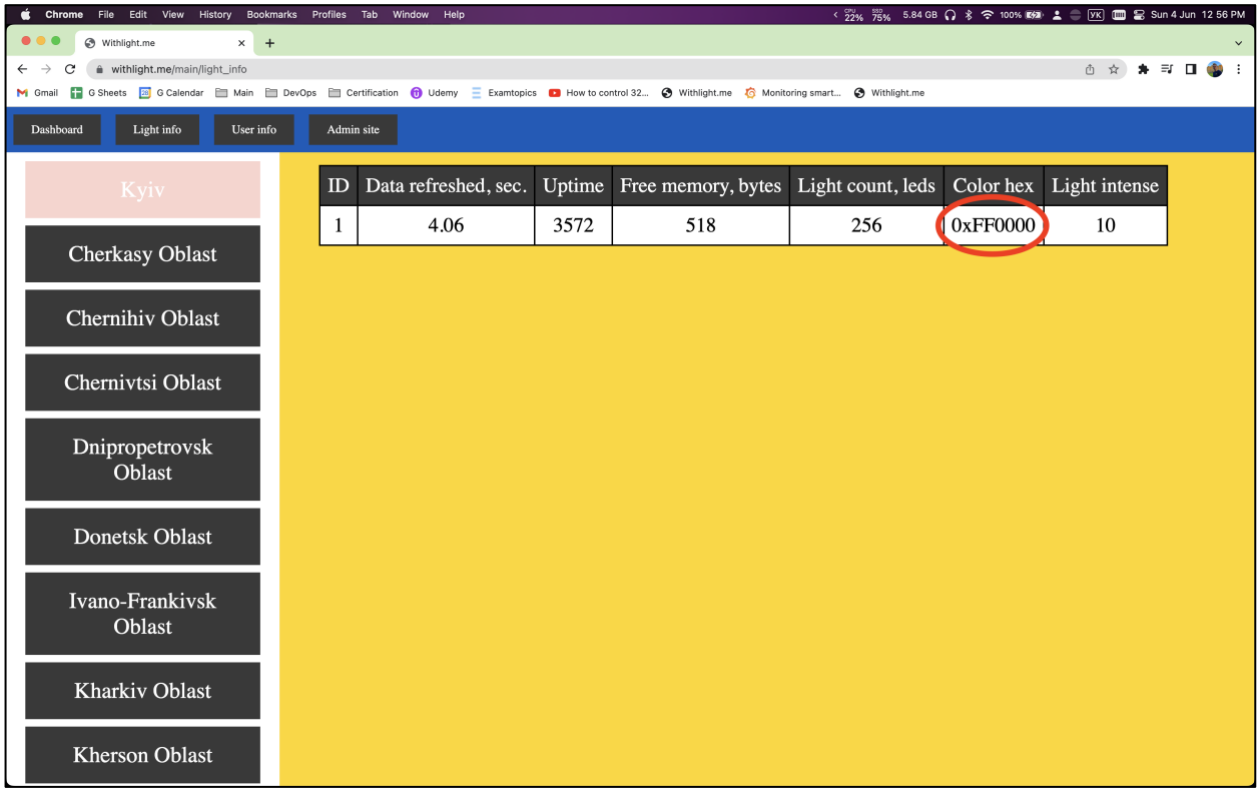


Рисунок 3.14. Перегляд оновленої інформації на сторінці зі світловими приладами

Щоб створити нового користувача з правами «Viewer», необхідно перейти знову на панель керування адміністратора та заповнити всю інформацію про користувача (рисунок 3.15):

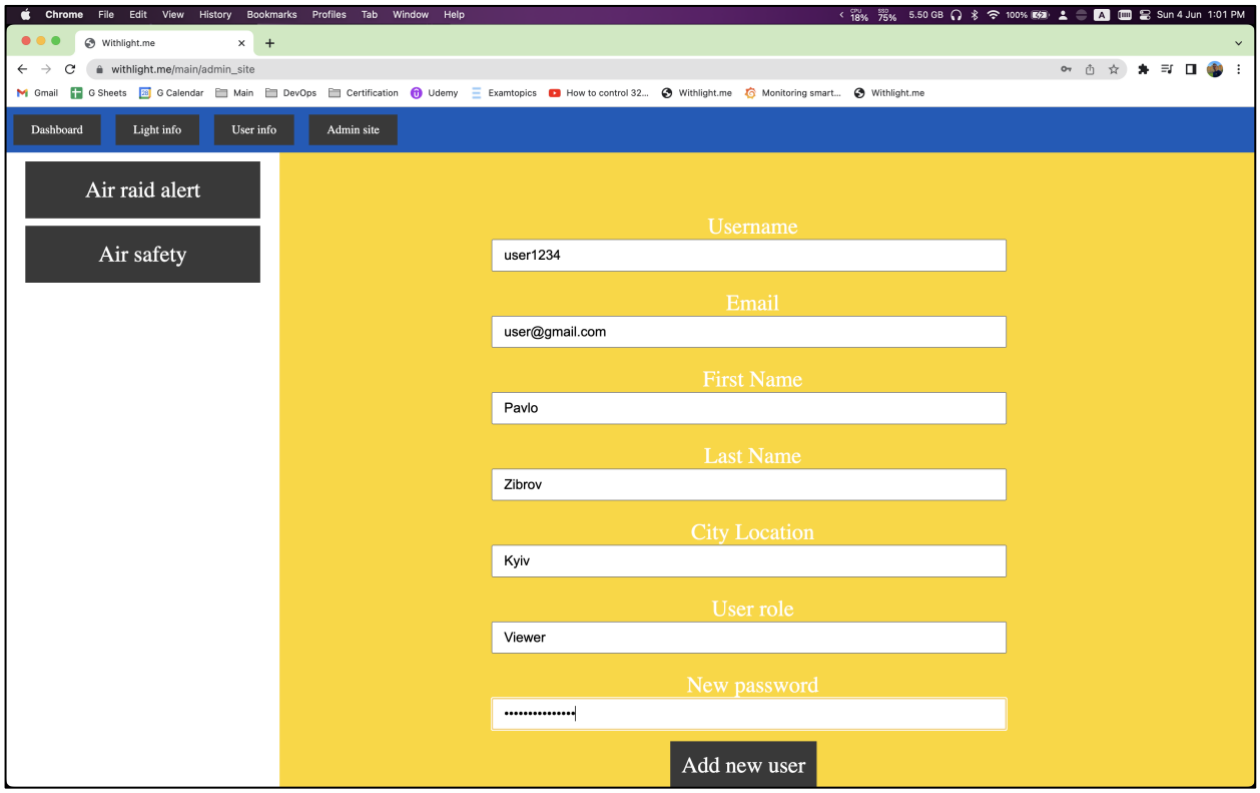


Рисунок 3.15 – Створення нового нового користувача

Після натискання кнопку «Add new user» можна авторизуватися з-під нього.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ХМАРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ОСВІТЛЕННЯ МІСТА

Графічний матеріал

КПІ.ІП-9604.045430.06.99

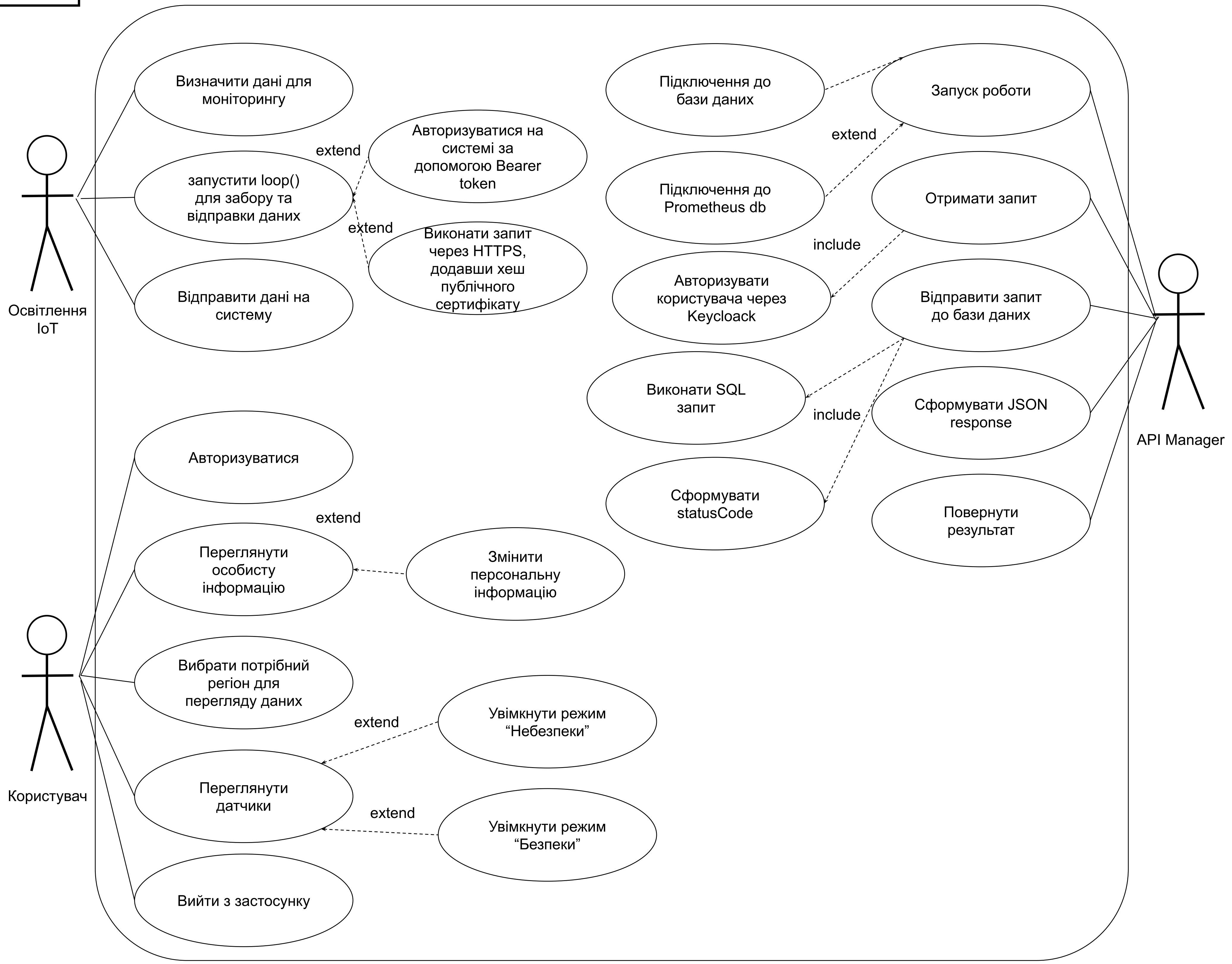
“ПОГОДЖЕНО”
Керівник проекту:
_____ Ірина ВІТКОВСЬКА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

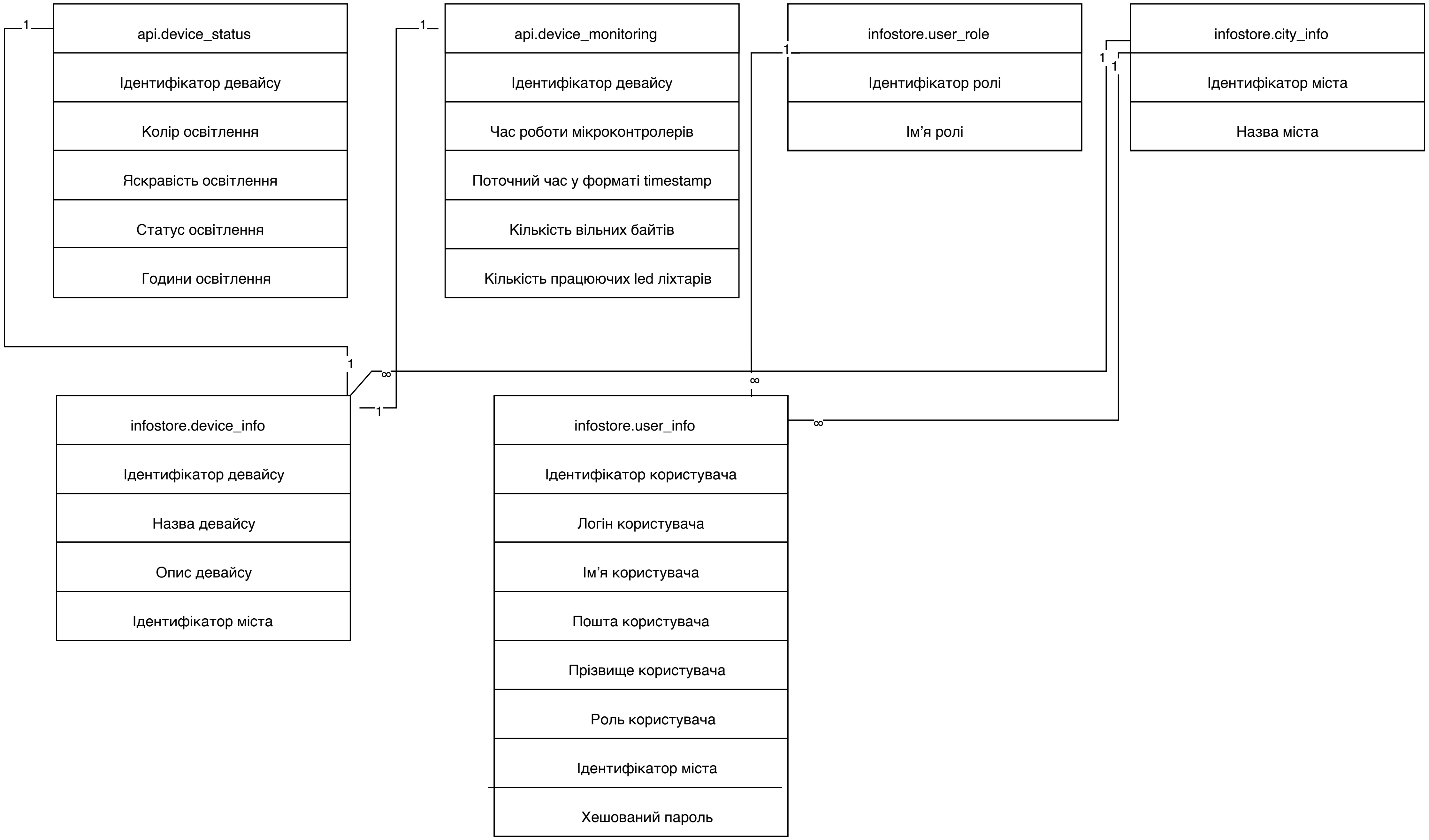
Виконавець:
_____ Олексій БУРЯТОВ

Київ – 2023

КПІ.ІП-9604.045430.09.99.СБД



					КПІ.ІП-9604.045430.06.99.СБД		
Зм.	Арк.	№ документа	Підпис	Дата	Діаграма варіантів використання	Літера	Маса	Масштаб
Розробив		Бурятов О.О.					
Перевірив		Вітковська І.І.				Аркуш	Аркушів
Т. кон.					Хмарне програмне забезпечення для моніторингу освітлення міста	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-96	
Н. кон.		Вітковська І.І.					
Затвердив		Жаріков Е.В.					



Зм.	Арк.	№ документа	Підпис	Дата	КПІ.ІП-9604.045430.07.99.СБД
Розробив		Бурятов О.О.			Схема бази даних
Перевірив		Вітковська І.І.			Літера / Маса / Масштаб
Т. кон.					Аркуш / Аркушів
Н. кон.		Вітковська І.І.			Хмарне програмне забезпечення для моніторингу освітлення міста
Затвердив		Жаріков Е.В.			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-96

www.withlight.me

Login

username

password

Log in

www.withlight.me/light_info

Dashboard | Light info | User info | Admin site

Kyiv | Lviv | Sumy | Kharkiv | Ternopil | Poltava

www.withlight.me/admin_site

Dashboard | Light info | User info | Admin site

Air raid alert

Air safety

Username

Email

First Name

Last Name

City Location

User role

New password

www.withlight.me/dashboard

Dashboard | Light info | User info | Admin site

Графік 1

Графік 2

Графік 3

Графік 4

www.withlight.me/user_info

Dashboard | Light info | User info | Admin site

Personal info

Username

Email

First Name

Last Name

City Location

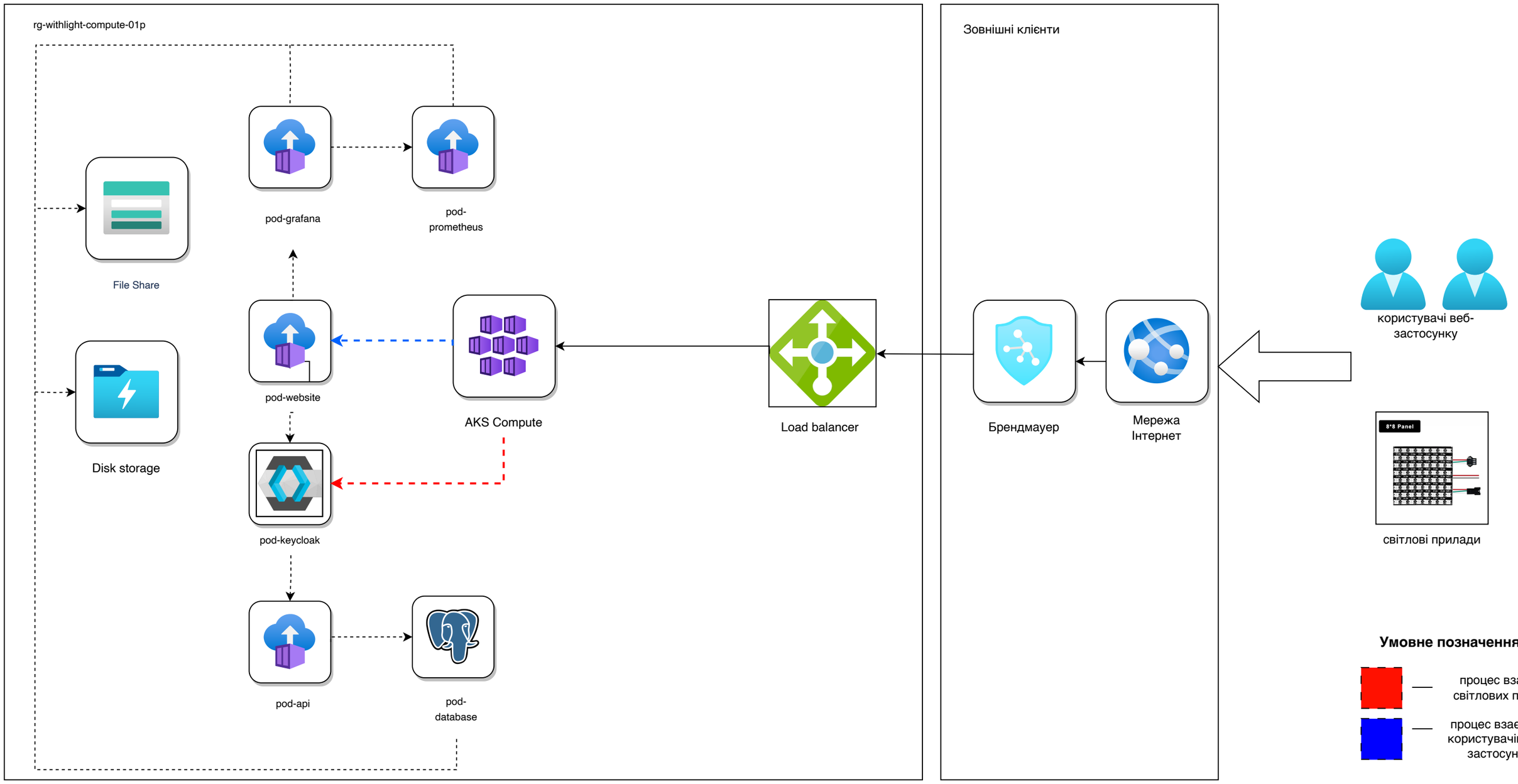
User role

Password

Зм.	Арк.	№ докум.	Підп.	Дата	КПІ.ІП-9604.045430.08.99.КЕ		
Розроб.	Бурячок О.О.				Хмарне програмне забезпечення для моніторингу освітлення міста	Літ.	Арк.	Аркушів
Перев.	Вінковська І.І.						1	
Т. Кон.						Аркуш	Аркушів
Н. Кон.	Вінковська І.І.					КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-96	
Затв.	Жарікова Е.В.						

Кріплення вузлових екранних форм

Схема архітектури програмного забезпечення



Демонстраційний плакат до дипломного проекту

Хмарне програмне забезпечення для моніторингу освітлення міста

Керівник Вітковська І.І.