

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ВЕБПОРТАЛУ ДЛЯ ПІДТРИМКИ ЗЕЛЕНОЇ
СТРАТЕГІЇ РОЗВИТКУ УНІВЕРСИТЕТУ
DEVELOPMENT OF A WEB PORTAL TO SUPPORT THE
UNIVERSITY'S GREEN DEVELOPMENT STRATEGY

Виконав(ла): студент(ка) 4 курсу, групи КНТ-221

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

КІРІСНКО А.С.

(ПРИЗВИЩЕ та ініціали)

Керівник ПАРХОМЕНКО А.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент ЗЕЛЕНЬОВА І.Я.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 122 Комп'ютерні науки
(код і найменування)
 Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КІРІЄНКА Андрія Сергійовича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка вебпорталу для підтримки зеленої стратегії розвитку університету. Development of a Web Portal to Support the University's Green Development Strategy

керівник проєкту (роботи) к.т.н., доцент, ПАРХОМЕНКО Анжеліка Володимирівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програмного забезпечення. 3. Розробка програмного забезпечення. 4. Експлуатація, тестування, та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ПАРХОМЕНКО А.В., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання « 14 » квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)


(підпис)

Андрій КІРІЄНКО
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)


(підпис)

Анжеліка ПАРХОМЕНКО
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 105 с., 16 табл., 32 рис., 3 дод., 41 джерело.

ВЕБПОРТАЛ, ЗЕЛЕНА СТРАТЕГІЯ РОЗВИТКУ УНІВЕРСИТЕТУ, ЗБІР МЕТЕОДАНИХ, ПОРІВНЯЛЬНИЙ АНАЛІЗ, NODE.JS, REACT.

Об'єкт дослідження – програмна підтримка стратегії зеленого розвитку університету.

Предмет дослідження – методи та програмні засоби збору, обробки та візуалізації результатів моніторингу глобальних та локальних метеорологічних даних.

Мета роботи – розробка вебпорталу, що забезпечить збирання, збереження, візуалізацію та порівняльний аналіз локальних та глобальних метеорологічних даних для підтримки зеленої стратегії розвитку університету.

Матеріали, методи та технічні засоби: шаблон MVC, REST API, засоби візуалізації Chart.js, Leaflet; інструменти Docker, Git, Node.js, React, MongoDB.

Результати. Реалізовано вебпортал, що дозволяє збирати глобальні та локальні метеорологічні дані для подальшого прогнозування генерації енергії з відновлюваних джерел. Система забезпечує зручний інтерфейс для користувачів і розробників, надаючи наочну візуалізацію розташування датчиків на мапі, а також отриманих даних у вигляді графіків. Забезпечено інтеграцію з зовнішнім метеосервісом та екологічним чатботом.

Висновки. Портал має стати платформою для підвищення енергоефективності та залучення університетської спільноти до екологічних заходів, що сприятиме підвищенню позиції університету в рейтингу UI GreenMetric.

Галузь використання – університети, освітні платформи, дослідницькі центри.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 105 pages, 16 tables, 32 figures, 3 appendixes, 41 sources.

WEB PORTAL, GREEN UNIVERSITY DEVELOPMENT STRATEGY, METEOROLOGICAL DATA COLLECTION, COMPARATIVE ANALYSIS, NODE.JS, REACT.

Object of research – software support for the university's green development strategy.

The subject of the study is methods and software tools for collecting, processing and visualizing the results of monitoring global and local meteorological data.

The purpose of the study is to develop a web portal that will provide collection, storage, visualization and comparative analysis of local and global meteorological data to support the green development strategy of the university.

Materials, methods and techniques: MVC template, REST API, Chart.js, Leaflet visualization tools; Docker, Git, Node.js, React, MongoDB tools.

Results. A web portal has been implemented that allows collecting global and local meteorological data for further forecasting energy generation from renewable sources. The system provides a user-friendly interface for users and developers, providing a visualization of the location of sensors on the map, as well as the data obtained in the form of graphs. Integration with an external weather service and an environmental chatbot is provided.

Conclusions. The portal should become a platform for improving energy efficiency and involving the university community in environmental activities, which will help to improve the university's position in the UI GreenMetric ranking.

Application area: universities, educational platforms, research centers.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Огляд особливостей сталого розвитку та ролі університетів у «зеленому переході»	10
1.2 Дослідження існуючих цифрових рішень	21
1.3 Аналіз теоретичних основ побудови систем екологічного моніторингу	29
1.4 Постановка завдань дипломної роботи.....	34
2 Проєктування програмного забезпечення	35
2.1 Аналіз вимог	35
2.2 Архітектура програмного забезпечення	36
2.3 Обґрунтування вибору технологій	40
2.4 Висновки до розділу 2	42
3 Розробка програмного забезпечення.....	43
3.1 Загальна структура програмної системи	43
3.2 Реалізація клієнтської частини	44
3.3 Реалізація серверної частини	47
3.4 Робота з базою даних	49
3.5 Обробка та візуалізація екологічних даних.....	50
3.6 Інтеграція чатбота у вебпортал.....	52
4 Експлуатація, тестування та експериментальне дослідження програми	53
4.1 Інструкція користувача.....	53
4.2 Інструкція для розробника	62
4.3 Тестування компонентів.....	64
4.4 Висновки до розділу 4	66
Висновки	67

Перелік джерел посилань	68
Додаток А Технічне завдання	71
Додаток Б Текст програми	77
Додаток В Слайди презентації.....	98

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AI	– Artificial Intelligence;
EUA	– European University Association;
GIS	– Geographic Information System;
IoT	– Internet Of Things;
MVC	– Model-View-Controller;
UML	– Unified Modeling Language;
ДЗЗ	– дистанційне зондування землі;
ООН	– Організація Об'єднаних Націй;
СЕМ	– система екологічного моніторингу;
ЦСР	– цілі сталого розвитку.

ВСТУП

У сучасному світі університети виступають не лише освітніми та науковими центрами, а й активними учасниками екологічних трансформацій. Вони дедалі частіше впроваджують ініціативи, спрямовані на зменшення впливу на довкілля та популяризацію принципів сталого розвитку серед студентської спільноти. Багато університетів створюють спеціалізовані вебпортали для інформування громадськості про свої екологічні ініціативи та досягнення.

UI GreenMetric – це рейтинг, що враховує різноманітні показники, зокрема стан інфраструктури, енергозбереження, кліматичні зміни, управління відходами та водними ресурсами, розвиток транспортної системи, а також екологічну освіту і дослідження. Високі позиції у цьому рейтингу свідчать про системну екологічну політику університету та його прихильність до ідей сталого розвитку. Але Національний університет «Запорізька політехніка» поки не представлений у цьому рейтингу.

Розробка вебпорталу, присвяченого підтримці зеленої стратегії розвитку університету, дозволить централізовано представити інформацію про екологічні ініціативи, заходи та досягнення закладу освіти. Такий портал сприятиме підвищенню обізнаності студентів, викладачів та співробітників щодо екологічних питань, залученню їх до участі в екологічних програмах та підвищенню загального рівня екологічної культури в університеті. Портал має стати платформою для обміну інформацією, залучення спільноти до екологічних заходів та сприяти підвищенню позиції університету в рейтингу UI GreenMetric. Тому, тема роботи є актуальною.

Мета роботи – розробка вебпорталу, що забезпечить збирання, збереження, візуалізацію та порівняльний аналіз локальних та глобальних метеорологічних даних для підтримки зеленої стратегії розвитку університету.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд особливостей сталого розвитку та ролі університетів у «зеленому переході»

Потреба у впровадженні в Україні Стратегії сталого розвитку Стратегія обумовлена чинниками внутрішнього і зовнішнього характеру, такі як [1]:

- потребою в актуалізації Стратегії сталого розвитку «Україна – 2020» у зв'язку з ухваленням на Саміті Організації Об'єднаних Націй (ООН) зі сталого розвитку 17 глобальних цілей сталого розвитку на період до 2030 року відображено в таблиці 1.1 [1];

- у зв'язку з виявленим рівнем економічного розвитку і добробуту населення, який не відповідає природному, науково-технічному, аграрно-промисловому потенціалу та кваліфікаційно-освітньому рівню населення, соціально-історичним і культурним традиціям народу України [1];

- фактом домінування терміну упродовж багатьох років ресурсо- та енергоємних галузей та технологій, сировинної орієнтації експорту, надмірної концентрації виробництва у промислових регіонах України небезпечна структура управління розвитком, яка загалом є неефективною та екологічно небезпечною [1];

- Україна має такі міжнародні зобов'язання: сталий розвиток, визначений як стратегія документи ООН [1];

- основа впровадження інновацій можливість для України трансформуватися в напрямку сталого розвитку – це Україна і Європейський Союз [1];

- при розробці та затвердженні стратегічних загальнодержавних, галузевих та регіональних документах не повною мірою відображають цілі та завдання сталого розвитку [1];

- українські вчені підготували наукову теорію переходу України до сталого розвитку та опублікували відповідний паблік [1].

Таблиця 1.1 – Цілі сталого розвитку [1]

№	Назва цілі	Стислий опис
1	Ні бідності	Усунути крайню бідність і забезпечити соціальний захист
2	Ні голоду	Забезпечити продовольчу безпеку та сталий розвиток сільського господарства
3	Гарне здоров'я	Знизити смертність, покращити медичні послуги
4	Якісна освіта	Забезпечити доступ до інклюзивної та рівної освіти
5	Гендерна рівність	Досягти різних прав і можливостей для жінок і чоловіків
6	Чиста вода та належні санітарні умови	Забезпечити доступ до безпечної води та санітарії
7	Відновлювальна енергія	Забезпечити доступ до сучасних джерел енергії
8	Гідна праця та економічне зростання	Сприяти економічному зростанню та продуктивній зайнятості
9	Інновації та інфраструктура	Розвивати інноваційні технології, промисловість та інфраструктуру
10	Зменшення нерівності	Сприяти рівним можливостям для всіх
11	Сталий розвиток міст та спільнот	Зробити міста безпечними, сталими та інклюзивними
12	Відповідальне споживання	Зменшити відходи, підвищити ефективність ресурсів
13	Боротьба зі зміною клімату	Вжити термінових заходів щодо зміни клімату
14	Збереження морських екосистем	Захистити океани, моря та морські ресурси
15	Збереження екосистем суші	Охороняти ліси, боротьба з опустелюванням та втратою біорізноманіття
16	Мир та справедливість	Зміцнити правопорядок, скоротити насильство та корупцію
17	Партнерство заради стійкого розвитку	Посилити глобальне партнерство і міжнародне співробітництво

Основою для формування Стратегії сталого розвитку України стали 17 глобальних Цілей сталого розвитку до 2030 року, Стратегія сталого розвитку «Україна – 2020», а також Оновлена стратегія сталого розвитку Європейського Союзу. У процесі розробки документа були проведені SWOT-аналіз та аналіз прогалин у політиці, законодавстві, діючих програмах і планах у таких ключових сферах: сільське господарство, економіка, охорона довкілля, соціальна політика, природокористування та енергетика, транспорт,

регіональний розвиток, освіта і наука. Проект Стратегії обговорювався упродовж червня – грудня 2016 року на регіональному та національному рівнях. У консультаціях брали участь представники державної влади, органів місцевого самоврядування, народні депутати, науковці, освітяни, громадські організації, професійні спілки, бізнес, медіа та міжнародні експерти [1].

Головною метою Стратегії є досягнення збалансованого соціально-економічного та екологічного розвитку країни, забезпечення українського населення відповідним умовам якісного життя для нинішнього та майбутнього поколінь [1].

Основним механізмом реалізації визначено Національний план дій (дорожню карту), що буде ухвалений після затвердження самої Стратегії. Документ встановлює систему стратегічних та операційних цілей, а також визначає інституційні механізми впровадження, напрями міжгалузевої взаємодії, ключові інструменти та фактори реалізації. Для забезпечення поетапного впровадження Стратегії передбачено три часові етапи: 2017–2020, 2021–2025 та 2026–2030 роки [1].

До основних принципів, що лежать в основі Стратегії сталого розвитку, належать [1]:

- дотримання прав людини як у природньому, так і в соціальному середовищі;
- відкритість управлінського процесу та відповідальність усіх учасників управління;
- забезпечення участі громадян і зацікавлених сторін у процесі прийняті рішень;
- інтеграція політичного, соціального та регіонального рівнів задля підвищення сталого розвитку;
- задоволення потреби теперішнього покоління без шкоди для можливостей майбутніх поколінь;
- використання сучасних наукових знань та технологій для прийняття рішень у сфері економіки та екології;

- застосування превентивних заходів для зменшення негативного впливу на довкілля та здоров'я людей;
- зобов'язати компанії або люди які використовують природні ресурси платити за їх використання та оновлення в майбутньому;
- запровадити відповідальність тих хто забруднює довкілля або шкодить здоров'ю людей у вигляді компенсацій для відновлення природи або здоров'ю населення.

Сталий розвиток налічує в собі 7 основних національних цілей.

Підтримка тривалості збалансованого зростання та надання достойної зайнятості кожного, що несе за собою наступні завдання [1]:

- забезпечення щорічного зростання валового внутрішнього продукту (ВВП) на рівні не менше ніж 4%;
- підвищення частки експорту продукції з високою додатною вартістю;
- шляхом технічної модернізації та диверсифікації досягнути збільшення кількості робочих місць;
- збільшення фінансування наукових досліджень з усіх доступних джерел;
- реалізація політики, що сприяє якісним умовам праці та підвищенню кваліфікації населення;
- гарантування рівної оплати праці за рівнозначну роботу незалежно від статі чи фізичних можливостей.

Забезпечити розвиток в усіх галузях економіки та регіонах, з особливою увагою до сільського господарства, в межах реалізації цією цілі полягають наступні завдання [1]:

- підвищення продуктивності праці та рівня доходів у сфері сільського господарства;
- забезпечення збалансованого виробництва продуктів харчування відповідно до потреб населення;
- розширення площ земель сільськогосподарського призначення;

- збільшення обсягів фінансування в сільськогосподарську діяльність;
- забезпечити збереження наявних природних ресурсів, пов'язаних із сільськогосподарською діяльністю.

Боротьба з бідністю, зменшення соціальної та гендерної нерівності має наступні завдання [1]:

- ліквідація крайньої бідності серед населення;
- забезпечення права громадян на володіння, користування та розпорядження землею, іншими формами власності, природними ресурсами та фінансовими послугами;
- досягнення стабільного зростання доходів для найбідніших верств населення;
- удосконалення бюджетно-податкової політики у сфері регулювання заробітної плати;
- впровадження гендерної складової в усі сфери діяльності;
- підвищення рівня зайнятості жінок, які мають дітей віком до 6 років.

Гарантування якісного здоров'я, добробуту та освіти, а також безпечного життя у містах і селах, передбачають наступні завдання [1]:

- зменшення рівня смертності серед новонароджених і дітей до 5 років;
- зниженні захворюваності на туберкульоз, ВІЛ/СНІД, парентеральний гепатит та інші небезпечні вірусні інфекції;
- скоротити кількість смертей і травм унаслідок дорожньо-транспортних пригод;
- забезпечення доступу до якісної дошкільної освіти для всіх дітей, незалежно від здоров'я, соціально-економічного статусу родини чи місце проживання;
- усунення гендерної нерівності в сфері освіти;

- забезпечення широкого доступу до безпечних, доступних, недорогих і екологічно збалансованих транспортних систем.

Перехід до відповідального споживання і виробництва, ефективного природокористування та активної кліматичної політики передбачають такі завдання [1]:

- забезпечення повного та комплексного впровадження систем регулювання споживання води, тепла та інших енергоресурсів;
- збільшення частки твердих побутових відходів, що спалюються, утилізуються або переробляються до рівня не менше 50%;
- гарантування доступу всього населення до безпечної та недорогої питної води – як у сільській, так і в міській місцевості;
- ліквідування несанкціонованих сміттєзвалищ, особливо в прибережних зонах;
- зменшення обсягів викидів парникових газів в атмосферу в усіх секторах економіки до рівня, який перевищує 60% від базового показника;
- поліпшення екологічної освіти, інформування населення, розширення можливостей установ і громадян щодо запобігання змінам клімату, адаптації до них та своєчасного реагування.

Збереження природного різноманіття суші та морів, сталий підхід до використання їхніх ресурсів передбачають наступні завдання [1]:

- збільшення рівня лісистості території до 20% від загальної площі країни;
- забезпечення цілісності екосистем і біорізноманіття шляхом ефективного планування на національному, регіональному та місцевому рівнях;
- встановлення прибережних та водоохоронних захисних смуг уздовж морського узбережжя, а також гарантування їх охорони та цільового використання;
- зменшення процесів деградації природного середовища та відновлення пошкоджених територій;

- вжити заходів для боротьби з браконьєрством і незаконною торгівлею видами флори та фауни.

Гарантування безпеки, доступу до справедливості, формулювання підзвітних і інклюзивних інституцій передбачають наступні завдання [1]:

- зниження поширення усіх форм насильства та зниження насильницької смертності шляхом посилення охорони правопорядку;
- боротьба з усіма видами злочинності, включаючи незаконну торгівлю в усіх сферах ринку;
- забезпечення рівного доступу всіх громадян до правосуддя, зокрема через можливість звернення до національних та міжнародних судів;
- зниження рівня корупції у всіх формах шляхом підвищення ефективності системи запобігання і контролю;
- сприяння прозорому та відповідальному ухваленню рішень органами влади всіх рівнів із залученням представників усіх верств населення.

Європейський зелений курс — це стратегічна ініціатива, спрямована на досягнення кліматичної нейтральності в Європі до 2050 року. Вона охоплює комплекс політик і регуляторних механізмів, орієнтованих насамперед на вуглецево-інтенсивні галузі з метою трансформації моделей виробництва та споживання при збереженні динаміки економічного зростання. Екологічна стійкість та «зелений» перехід визначені як ключові пріоритети для країн Європейського Союзу, а отже — і як стратегічний виклик для європейських закладів вищої освіти. Європейська асоціація університетів (EUA) офіційно задекларувала свою прихильність до досягнення вуглецевої нейтральності та бере активну участь у ініціативах, спрямованих на поліпшення кліматичних показників у межах вищої освіти. EUA також підтримує розвиток партнерських зв'язків між членами асоціації, сприяючи обміну знаннями та інноваційними підходами у сфері сталого розвитку. Це спрямовано на посилення впливу університетів на формування екологічно відповідального суспільства як на локальному, так і на глобальному рівнях. Усі ці зусилля утверджують важливість інтеграції цілей сталого розвитку в освітні програми

університетів і сприяють формуванню нової парадигми університетської освіти [2].

У подкасті, присвяченому інтеграції Цілей сталого розвитку в освітні структури, Вікторія Галан-Мурос у співрозмові з Бальжаном Оразбаєвим виокремила 11 ключових етапів, які дозволяють ефективно впроваджувати принципи сталого розвитку у діяльність закладів вищої освіти. Ці етапи охоплюють як організаційно-управлінські процеси, так і стратегічні, комунікаційні та ресурсні аспекти змін [3]:

- етап 0 «Концептуальне узгодження»: першочерговим кроком є формування спільного розуміння базових понять, таких як «стійкість» та «сталий розвиток», серед усіх учасників освітнього процесу та зацікавлених сторін. Це забезпечує єдність у візії змін [3];

- етап 1 «Підвищення обізнаності»: формування усвідомлення причин та важливості переходу до сталої моделі функціонування університету, спільнота повинна розуміти, як зміни вплинуть на неї безпосередньо [3];

- етап 2 «Лідерське зобов'язання»: активна участь керівництва – як у вираженні волі до змін, так і у виділенні ресурсів – є критично важливою, йдеться про політичну та організаційну волю до трансформації [3];

- етап 3 «Залучення зацікавлених сторін»: інклюзія внутрішніх (студенти, викладачі, адміністрація) та зовнішніх (партнери, громада) акторів до процесу впровадження, з метою підвищення рівня їх участі та відповідальності [3];

- етап 4 «Розробка стратегії»: формування спільно схваленого стратегічного документа, який містить цілі, пріоритети та механізми реалізації цілей сталого розвитку (ЦСР) в університеті [3];

- етап 5 «Освітній супровід змін»: учасники освітнього середовища повинні бути поінформовані про очікувані зміни, їхній зміст, наслідки та вигоди – як під час переходу, так і в постперехідний період [3];

- етап 6 «Аналіз поточних ініціатив»: оцінка вже наявних практик, проєктів і програм, які прямо або опосередковано сприяють досягненню ЦСР у діяльності закладу [3];
- етап 7 «Визначення пріоритетів для майбутніх ініціатив»: на основі аналізу визначаються напрями, які ще не реалізуються, але мають потенціал для позитивного впливу [3];
- етап 8 «Забезпечення ресурсами»: визначення та мобілізація необхідних ресурсів – людських, матеріальних, фінансових – для реалізації окреслених ініціатив [3];
- етап 9 «Реалізація ініціатив»: планування та запуск конкретних проєктів у межах загальної стратегії сталого розвитку, процес має бути інтегрованим, із належною координацією в усіх підрозділах установи [3];
- етап 10 «Забезпечення довготривалої стійкості»: контроль і підтримка ініціатив у довгостроковій перспективі, що передбачає інституційну сталість, системну підтримку, моніторинг і адаптацію [3].

На рисунку 1.1 відображено схематично етапи впровадження принципів сталого розвитку у діяльність закладів вищої освіти [3].

Цей підхід формує цілісну рамку для університетів, які прагнуть інтегрувати ЦСР на всіх рівнях своєї діяльності – від стратегії до щоденної практики викладання, досліджень і взаємодії з суспільством [3].

В таблиці 1.2 наведено огляд та порівняння вебпорталів «зелених» університетів [4]-[7].

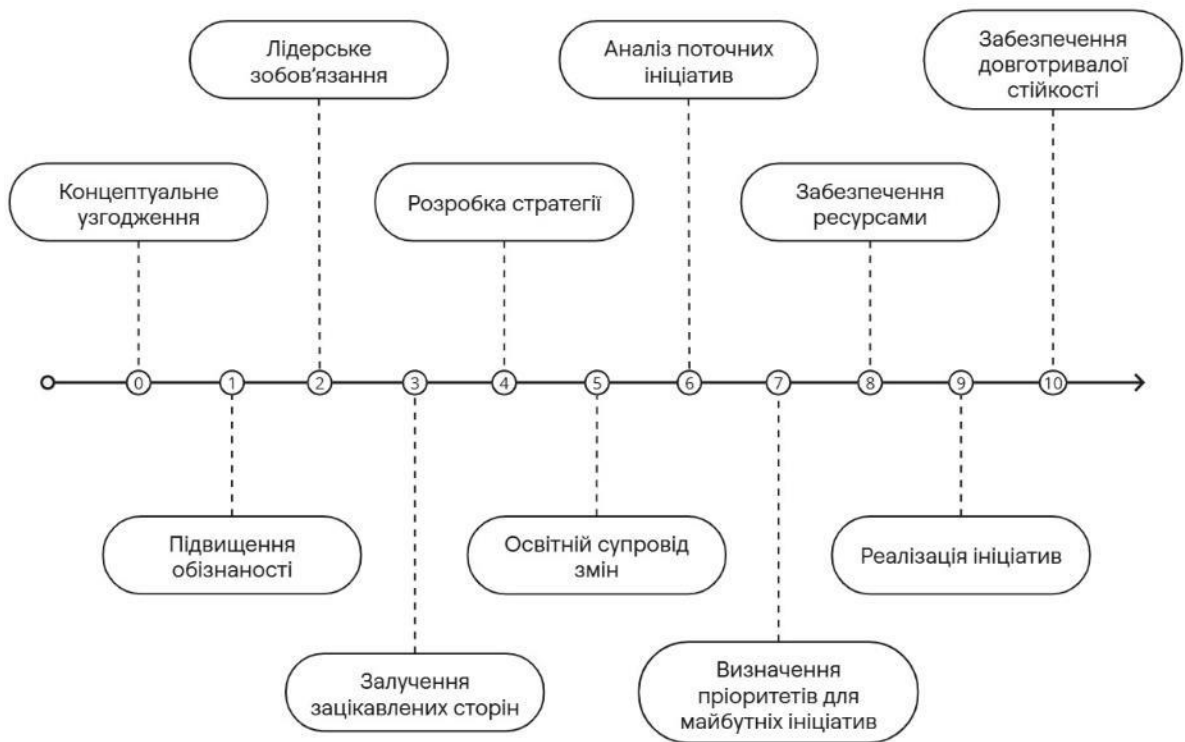


Рисунок 1.1 – Етапи впровадження принципів сталого розвитку у діяльність закладів вищої освіти [3]

Таблиця 1.2 – Порівняльна таблиця «зелених» університетських ініціатив [4]-[7]

Критерій	ЗУНУ (ЕкоБот) [4]	КРОК (ЕКО «ЛОТОС») [5]	George Mason University [6]	University of Pennsylvania [7]
1	2	3	4	5
Наявність окремого розділу	Окремий портал «ЕкоБот» (рис. 1.2)	Розділ «ЕКО ЛОТОС» (рис. 1.3)	Сайт Green GMU (рис. 1.4)	Sustainability at Penn (рис. 1.5)
Доступ до даних у реальному часі	Графіки з сенсорів температури, вологості, якості води (рис. 1.2)	Даних у реальному часі немає	Дані про компост, споживання води, електрики (рис. 1.4)	Є доступ до даних про енергію, мобільність
Опис проєктів та ініціатив	Розписано реалізацію системи моніторингу	Проєкти екологічного виховання (рис. 1.3)	Patriot Green Fund, енергетичні ініціативи (рис. 1.4)	Climate & Sustainability Action Plan

Кінець таблиці 1.2 [4]-[7]

1	2	3	4	5
Можливості для студентів	Не зазначено	Участь у заходах ЕКО-клубу (рис. 1.3)	Волонтерство, фінансування ідей	Стажування, дослідження, конкурси (рис. 1.5)
Інтеграція у навчальні програми	Не зазначено	Немає підтвердження	Є тематичні курси та проекти	Інтегровано в академічні програми
Візуальні або інтерактивні елементи	Дані з сенсорів (рис. 1.2)	Немає інтерактиву	Інтерактивні мапа (рис. 1.4)	Дашборди, онлайн-аналітика (рис. 1.5)

На рисунках 1.2-1.4 наведено приклади вебпорталів «зелених» університетів [4]-[7].

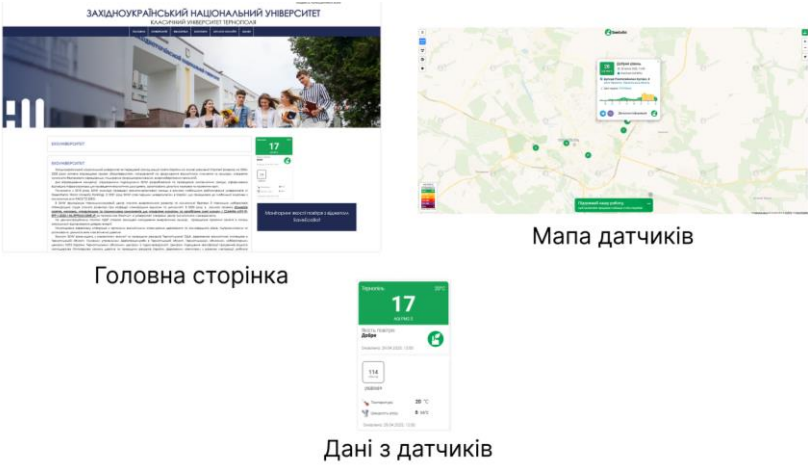


Рисунок 1.2 – Скріншоти екологічного порталу ЗУНУ [4]

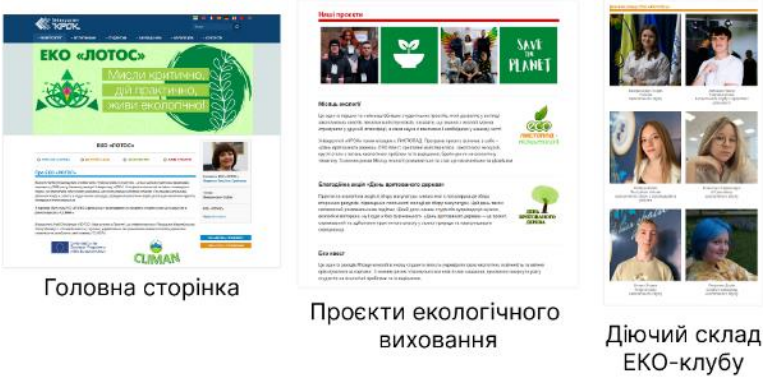


Рисунок 1.3 – Скріншоти екологічного порталу «ЕКО ЛОТОС» [5]

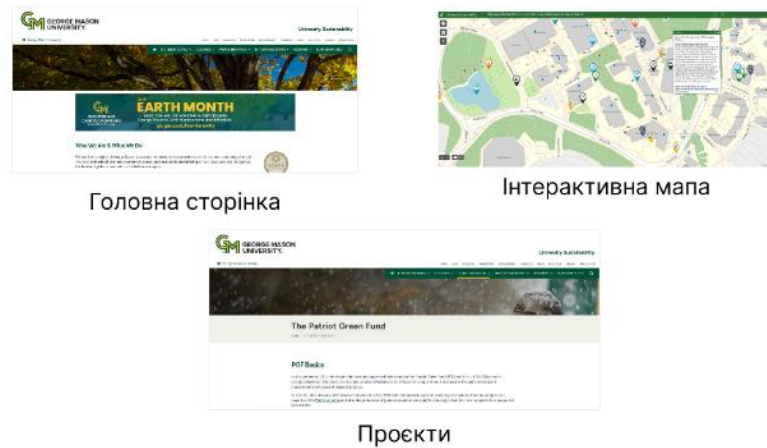


Рисунок 1.4 – Скріншоти екологічного порталу George Mason University [6]

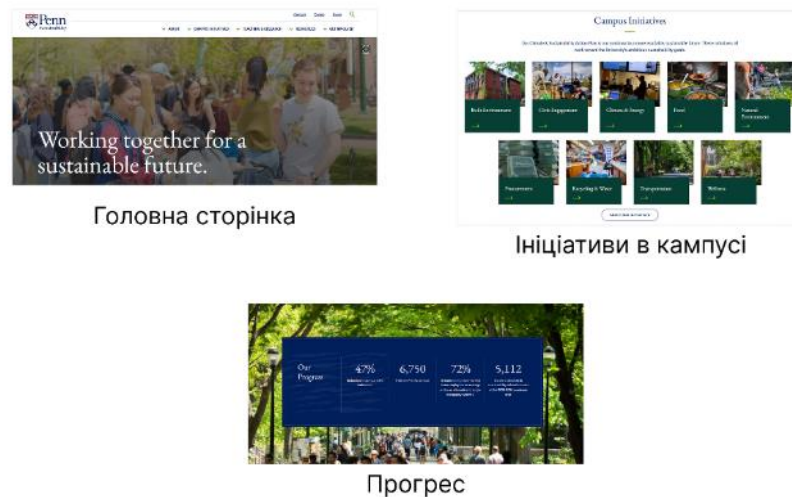


Рисунок 1.5 – Скріншоти екологічного порталу University of Pennsylvania [7]

1.2 Дослідження існуючих цифрових рішень

Цифрові технології відіграють визначальну роль у підвищенні ефективності екологічного моніторингу та сталого управління природними ресурсами. Особливо актуальним у цьому контексті є впровадження сучасних технологій, таких як Інтернет речей (Internet of Things, IoT) [8], геоінформаційні системи (GIS) [9], штучний інтелект (AI) [10] та технології обробки великих даних (Big Data) [11]. Ці рішення надають змогу здійснювати безперервне спостереження за станом навколишнього середовища в

реальному часі, оперативно аналізувати отриману інформацію та приймати обґрунтовані управлінські рішення щодо захисту екосистем [8].

Показовим прикладом успішної реалізації системи екологічного моніторингу на основі IoT [8] є досвід компанії BlueGraph, яка базується в Сінгапурі. Її розробка знайшла застосування в аграрному секторі, промисловому виробництві та міському плануванні, де особливо відчувалася потреба в точному та оперативному моніторингу екологічних параметрів. Компанія визначила критично важливим впровадження інтелектуальної системи, здатної забезпечувати контроль над показниками стану довкілля у складних умовах сучасного господарювання [8].

Метою створення системи було забезпечення безперервного збору, обробки та візуалізації екологічних даних. Зокрема, система охоплювала моніторинг таких параметрів, як [8]:

- якість повітря;
- температурні показники;
- рівень вологості;
- швидкість та напрямок вітру;
- інтенсивність та спектр сонячного освітлення;
- параметри якості води.

З технічної точки зору система реалізовувалася у кілька послідовних етапів [8]:

- збір функціональних і нефункціональних вимог до системи моніторингу;
- проєктування архітектури системи з урахуванням масштабованості та безпеки;
- вибір та інтеграція сенсорних модулів, відповідно до параметрів, що підлягали вимірюванню;
- розробка IoT-шлюзу для консолідації даних із сенсорів та їх передачі до хмари;

- налаштування хмарної інфраструктури, зокрема для обробки, зберігання та візуалізації інформації;
- інтеграція аналітичного інтерфейсу, який дозволяє будувати звіти, діаграми та прогнозні моделі.

Завдяки інтеграції із сучасними візуалізаційними інструментами користувачі мали змогу отримувати екологічну інформацію в реальному часі, переглядати динаміку показників за різні часові інтервали (від хвилин до тижнів), а також користуватися персоналізованими інформаційними панелями залежно від посадової ролі [8].

Особлива увага у проєкті була приділена інформаційній безпеці: дані шифрувалися під час збору, передачі та зберігання, що гарантувало надійний захист екологічної інформації. Завдяки інтеграції IoT із сучасними алгоритмами аналітики компанія змогла не лише оперативно реагувати на зміни у навколишньому середовищі, а й підвищити ефективність своєї діяльності, оптимізувати ресурси та сформувати передумови для сталого природокористування [8].

Таким чином, впровадження інтелектуальної системи екологічного моніторингу на базі IoT у компанії BlueGraph стало успішним прикладом цифрової трансформації управління довкіллям, що може бути адаптований і до умов інших країн, зокрема України, де потреба у подібних системах стає дедалі актуальнішою у зв'язку з екологічними викликами [8].

IoT у поєднанні з GIS є однією з ключових технологічних комбінацій, що сприяє цифровій трансформації міст та регіонів у напрямку їх сталого розвитку. IoT дозволяє об'єднувати фізичні об'єкти (так звані «речі»), обладнані сенсорами, мікропроцесорами та передавачами, у взаємопов'язану інфраструктуру, яка функціонує через мережі передачі даних, найчастіше – через Інтернет. Ці «речі» можуть включати побутові прилади, транспортні засоби, пристрої контролю стану довкілля, енергетичні системи тощо [9].

За оцінками аналітиків, вже до 2025 року кількість IoT-пристроїв у глобальному масштабі перевищить 21 мільярд, що свідчить про глибоке

проникнення цієї технології у всі сфери повсякденного життя. Таким чином, вплив Інтернету речей на соціально-економічні та екологічні процеси стає невід'ємним компонентом сучасного функціонування інфраструктури [9].

У свою чергу GIS є потужними інструментами для збору, обробки, зберігання, аналізу та візуалізації просторових (географічних) даних. Вони дозволяють ефективно виявляти просторові закономірності, підтримувати прийняття рішень, моделювати та прогнозувати зміни у навколишньому середовищі [9].

Основна мета інтеграції IoT [8] та GIS [9] полягає в синергетичному поєднанні переваг обох підходів для побудови гнучких, масштабованих і розумних систем моніторингу, керування та оптимізації у різноманітних галузях. Станом на сьогодні, існує низка напрямів, де така інтеграція вже успішно застосовується [9]:

- міська інфраструктура: у зв'язку з глобальними тенденціями до урбанізації, значна кількість міст активно впроваджує цифрові рішення для підвищення якості життя мешканців. IoT [8] у цьому контексті виконує функцію сенсорного шару, що забезпечує збір даних у реальному часі про різні аспекти міського середовища (транспорт, енергетика, безпека, екологія). У свою чергу, GIS [9] виступає засобом просторового аналізу та візуалізації зібраних даних, що критично важливо на етапах планування, проектування, будівництва й експлуатації міської інфраструктури [9];

- транспорт: поєднання IoT [8] та GIS [9] у сфері транспорту забезпечує оптимізацію маршрутів, безпечне паркування, зменшення заторів та підвищення ефективності логістичних процесів. GPS-трекери, датчики трафіку й геоаналітичні інструменти дозволяють в реальному часі контролювати пересування транспорту та оперативно реагувати на критичні ситуації [9];

- управління катастрофами: інтеграція зазначених технологій сприяє підвищенню готовності до надзвичайних ситуацій, зокрема на етапах моніторингу, прогнозування та відновлення після стихійних лих або

техногенних аварій. GIS [9] дозволяє моделювати розповсюдження катастрофічних подій, а IoT [8] забезпечує збирання первинних даних з місця події [9];

- екологічний моніторинг: контроль за якістю повітря, води, ґрунтів та іншими параметрами довкілля є неможливим без постійного моніторингу. Сенсори IoT фіксують екологічні показники, а GIS [9] забезпечує географічну прив'язку, аналітичну обробку та відображення даних на інтерактивних картах. Такий підхід дає змогу не лише виявляти джерела забруднення, а й формувати цілісну картину стану екосистем [9];

- сільське господарство: використання супутникових знімків у поєднанні з польовими IoT-сенсорами дозволяє отримувати дані про стан ґрунтів, вологість, наявність шкідників тощо. Це забезпечує прецизійне землеробство, оптимізацію витрат та збільшення врожайності [9];

- комунальні послуги: управління водопостачанням, вуличним освітленням, збором сміття та іншими сервісами значно спрощується завдяки впровадженню систем з дистанційним моніторингом і автоматичним управлінням. GIS [9] дозволяє формувати просторові моделі мереж, а IoT [8] – забезпечує оперативне реагування на несправності або аварії [9];

- охорона здоров'я: GIS-аналіз у поєднанні з IoT [8] дає змогу ефективно відстежувати поширення захворювань, планувати розміщення медичних установ та забезпечувати віддалений моніторинг стану пацієнтів. Це сприяє підвищенню доступності, швидкості реагування та якості медичної допомоги [9].

Таким чином, інтеграція IoT та GIS відкриває нові горизонти для побудови гнучких, адаптивних і високотехнологічних систем, що забезпечують стабільне функціонування різних сфер життєдіяльності людини. GIS виступає не лише платформою для візуалізації даних, але й аналітичним ядром, яке дозволяє обробляти великі обсяги просторової інформації в режимі реального часу. У подальших дослідженнях доцільно зосередитися на розширенні сфер застосування цієї інтеграції, удосконаленні архітектури

наявних систем, подоланні бар'єрів сумісності й безпеки, а також на глибшому аналізі темпоральних (часових) закономірностей у просторово-орієнтованих даних [8], [9].

У контексті цифровізації екологічного управління, особливої уваги заслуговує поєднання технологій AI та Big Data. Ці інструменти є ефективними у вирішенні складних аналітичних завдань, які пов'язані з обробкою та інтерпретацією великих масивів екологічної інформації [10], [11].

AI-технології здатні аналізувати дані, отримані з численних джерел – сенсорів, супутникових систем, хмарних сервісів тощо – з метою виявлення трендів, виявлення аномалій, прогнозування змін довкілля та формування оптимальних управлінських рішень. Алгоритми машинного навчання дозволяють автоматично визначати причинно-наслідкові зв'язки між екологічними показниками та факторами впливу, що суттєво підвищує якість прийняття рішень у реальному часі [10].

Один із практичних прикладів застосування таких технологій демонструє екосистема екологічного моніторингу у Великій Британії: на узбережжях та в місцях для купання було впроваджено системи контролю якості води в режимі реального часу. За допомогою AI-аналізу та обробки великих даних система може оперативно виявляти погіршення якості води та миттєво інформувати населення про потенційні загрози для здоров'я. Такий підхід значно підвищує рівень екологічної безпеки та соціальної відповідальності [10].

Впровадження цифрових технологій, зокрема AI [10], Big Data [11], IoT [8] і GIS [9], у сферу екологічного моніторингу має низку суттєвих переваг (рис. 1.6):

- оперативність: системи, що працюють у режимі реального часу, забезпечують швидке виявлення змін у довкіллі, що дозволяє здійснювати негайне реагування на потенційні загрози;

- **точність:** використання високотехнологічних сенсорів і аналітичних моделей дозволяє проводити більш глибокий і точний аналіз стану навколишнього середовища, включаючи динаміку змін;
- **прогнозування:** аналітичні моделі, засновані на історичних даних і машинному навчанні, дають змогу передбачити можливі екологічні тенденції, а отже – попередити кризові ситуації або адаптувати до них інфраструктуру;
- **прозорість:** автоматизований збір, зберігання та публікація екологічних даних у відкритому доступі сприяють підвищенню рівня громадської обізнаності, залученості та довіри до екологічної політики.

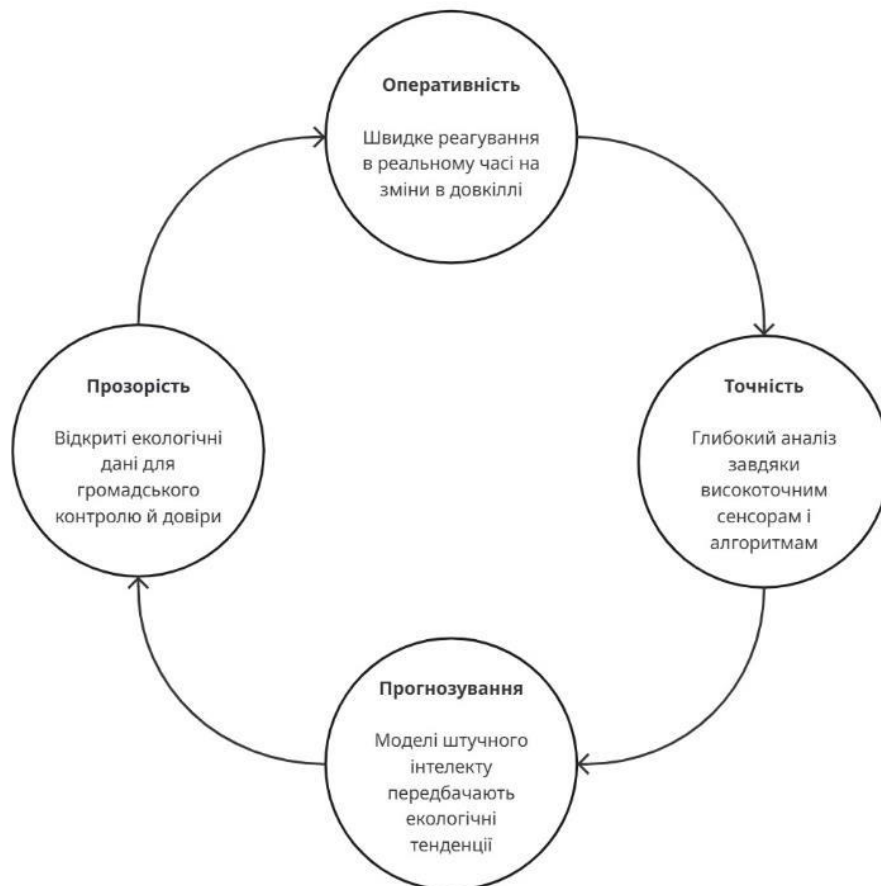


Рисунок 1.6 – Функціональні переваги впровадження AI, Big Data, IoT і GIS у CEM [8]-[11]

В таблиці 1.3 відображено порівняння використовуваних технологій AI, Big Data, IoT і GIS [8]-[11].

Таблиця 1.3 – Порівняння використовуваних технологій у сфері екологічного моніторингу

Технологія	Призначення	Переваги	Приклади застосування
IoT [8]	Збір даних із сенсорів у реальному часі	Автоматизація, оперативність висока точність	Моніторинг температури, вологості, якості повітря
GIS [9]	Аналіз і візуалізація просторових даних	Просторове моделювання, побудова карт, локалізація подій	Картографування зон забруднення, аналіз локацій
AI [10]	Обробка та аналіз великих обсягів даних, прогнозування	Автоматичне виявлення закономірностей, адаптивність	Прогноз погоди, виявлення аномалій у даних моніторингу
Big Data [11]	Зберігання, обробка та аналіз масивів даних з різних джерел	Масштабованість, аналіз у реальному часі, гнучкість	Обробка даних з сенсорів, супутників, метеостанцій

З огляду на загострення екологічної ситуації в Україні, зокрема внаслідок промислового забруднення, воєнних дій та незбалансованого природокористування, необхідність впровадження сучасних цифрових систем моніторингу є стратегічно важливою. Такі технології дозволять здійснювати більш точний контроль за станом довкілля, оптимізувати екологічне управління на національному та місцевому рівнях, а також сприятимуть адаптації України до європейських екологічних стандартів та інтеграції в міжнародні ініціативи сталого розвитку [10].

Крім того, впровадження цифрових рішень сприятиме розширенню екологічної компетентності громадян, підвищенню рівня екологічної культури та забезпеченню прозорого, науково обґрунтованого прийняття рішень у сфері охорони довкілля.

1.3 Аналіз теоретичних основ побудови систем екологічного моніторингу

Екологічна ситуація в сучасному світі, особливо в контексті стрімкої урбанізації, масштабної індустріалізації, а також прогресуючих глобальних кліматичних змін, вимагає перегляду та модернізації підходів до забезпечення екологічної стабільності. У цьому контексті особливої актуальності набуває створення та впровадження високоефективних систем екологічного моніторингу (СЕМ), які здатні в реальному часі забезпечувати надійний контроль за станом природного середовища, своєчасно виявляти потенційно небезпечні процеси та формувати науково обґрунтовані управлінські рішення [12].

Сучасна господарська діяльність людини – від масового видобутку природних ресурсів до зростаючого рівня викидів промислових забруднень – спричиняє інтенсивну трансформацію екосистем, що виявляється у деградації ґрунтів, втраті біорізноманіття, зниженні якості атмосферного повітря та води. Антропогенна активність посилює не лише локальні загрози, але й провокує масштабні екологічні катастрофи глобального масштабу. У таких умовах системи моніторингу середовища виконують критичну функцію – стають інструментом раннього попередження, прогнозування та реакції на негативні тенденції у природному середовищі [12].

Протягом останніх десятиліть світове населення зросло більш ніж удвічі, що спричинило вибухоподібне зростання попиту на природні ресурси, включаючи енергію, харчові продукти, прісну воду, території для проживання та ведення господарства. Зростання масштабів людської діяльності прямо корелює з рівнем впливу на довкілля, тому виникає потреба у всеохопному та системному збиранні екологічної інформації, її кваліфікованій обробці, аналізі та застосуванні для прийняття управлінських рішень на всіх рівнях – від локального до глобального [12].

СЕМ представляють собою складні техніко-інформаційні комплекси, які об'єднують технічні засоби, програмне забезпечення, організаційні процедури та нормативно-правове забезпечення, спрямовані на безперервне спостереження, оцінювання та прогнозування стану навколишнього середовища. Основна функція СЕМ полягає у своєчасному виявленні небезпечних змін параметрів довкілля, аналізі динаміки екологічних процесів, побудові прогнозів на основі достовірних даних та забезпеченні інформаційної підтримки процесів прийняття рішень в екологічній політиці та управлінні природокористуванням. Типова архітектура СЕМ включає низку взаємозалежних функціональних блоків, зокрема (рис. 1.7) [12]:

- польові сенсори та датчики, які безпосередньо здійснюють вимірювання фізичних, хімічних та біологічних параметрів середовища: температура, вологість, рівень рН, концентрації вуглекислого газу, важких металів, радіоактивних ізотопів тощо;
- системи збору, передачі та попередньої обробки даних, які забезпечують швидке та надійне передавання екологічної інформації до центрів аналізу через дротові або бездротові канали зв'язку;
- аналітичні платформи, що здійснюють фільтрацію, нормалізацію, структурування даних, а також розрахунок індикаторів стану довкілля. До таких платформ часто інтегрують елементи GIS [9], AI [10], баз даних та експертних систем;
- інтерфейси візуалізації для користувачів: інтерактивні карти, графіки, дашборди, звіти;
- керуючі модулі, які автоматизують функції попередження, формування сценаріїв реагування та управлінських рішень, зокрема сповіщення відповідальних служб, формування звітів, запуск регуляторних механізмів.

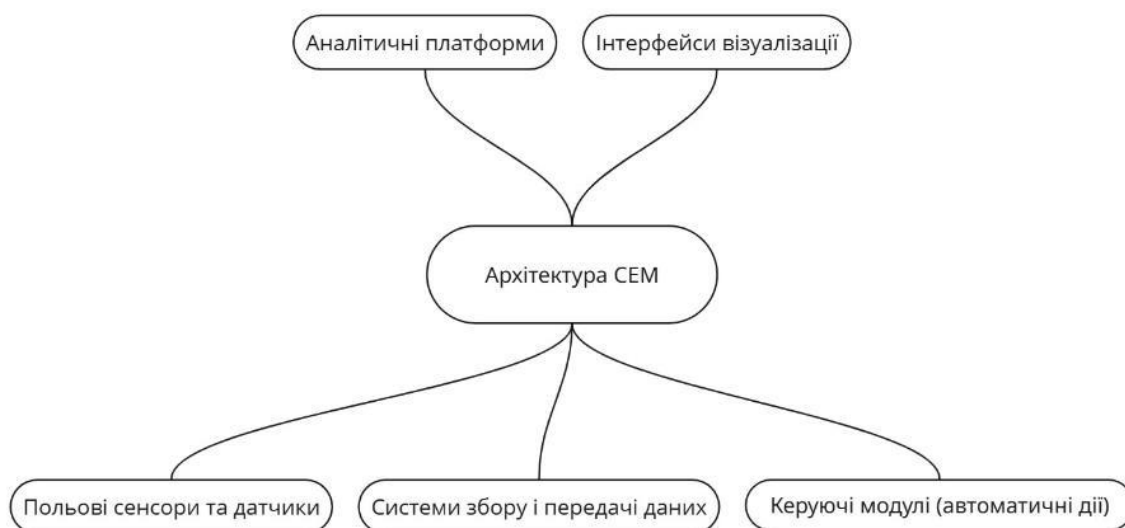


Рисунок 1.7 – Архітектура СЕМ [12]

Такі системи можуть охоплювати окремі природні середовища – атмосферу, водні об’єкти, ґрунтові покриви – або функціонувати комплексно, що особливо актуально у випадках забруднень змішаного типу чи природно-техногенних катастроф [12].

За класифікацією, запропонованою Günther et al., при побудові екологічних інформаційних систем використовують декілька основних типів моделей [12]:

- детерміністичні моделі ґрунтуються на фундаментальних фізико-математичних закономірностях, описують процеси з максимальною точністю та використовують рівняння переносу речовин, дифузії, теплообміну. Вони потребують значного обсягу початкових даних та значних обчислювальних ресурсів, проте дозволяють отримати високоточні результати;

- стохастичні моделі відображають імовірнісний характер природних процесів, які не піддаються однозначному математичному опису. Такі моделі особливо корисні в умовах невизначеності або неповноти інформації, наприклад, при прогнозуванні атмосферного забруднення, динаміки фітопланктону або біорізноманіття;

- гібридні моделі поєднують переваги детерміністичного підходу з гнучкістю стохастичних методів, що дозволяє адаптувати систему до змінних умов і враховувати як структуровану, так і непередбачувану інформацію;
- геоінформаційні моделі дозволяють інтегрувати просторову інформацію з екологічними показниками, створювати цифрові карти ризиків, моделювати поширення забруднення з урахуванням географічних особливостей;
- інтелектуальні моделі, зокрема штучні нейронні мережі, нечітка логіка, алгоритми машинного навчання, здатні навчатися на великих обсягах даних і виявляти складні нелінійні залежності між параметрами без попереднього формального моделювання.

Сучасні СЕМ побудовані за принципом багаторівневої обробки даних, яка включає наступні стадії [12]:

- первинний збір: за допомогою сенсорних систем, метеостанцій, супутників дистанційного зондування Землі, мобільних лабораторій;
- агрегація та стандартизація: включає фільтрацію шумів, заповнення пропущених значень, нормалізацію параметрів до єдиної шкали;
- зберігання: використання реляційних і нереляційних баз даних, хмарних сервісів, сховищ великих даних, з обов'язковою часовою й географічною прив'язкою;
- аналітика: побудова прогнозних сценаріїв, індексів, статистичних і математичних моделей, інтерпретація результатів і їх подання у вигляді зручних для сприйняття форм.

На практиці системи екологічного моніторингу реалізуються у вигляді комплексних проєктів. Наприклад, у федеральній землі Баден-Вюртемберг (Німеччина) функціонує проєкт UIS, який об'єднує екологічні бази даних, аналітичні модулі, супутникові знімки, лабораторні дослідження та інтерфейси користувача в єдину архітектуру. Така система дозволяє органам управління отримувати оперативну інформацію та забезпечує координацію між міністерствами, муніципалітетами й дослідницькими інститутами [12].

Крім того, у рамках досліджень інституту FAW Ulm були створені [12]:

- RESEDA: платформа обробки супутникових знімків із функціями автоматичного розпізнавання змін;
- ZEUS: система моніторингу ґрунтових вод із використанням адаптивних моделей рівня забруднення;
- WANDA: інтерфейс візуалізації екологічної динаміки в режимі реального часу;
- SCOPE: глобальна система оцінювання екологічного впливу антропогенних факторів на основі мультифакторного аналізу.

Окреме місце у сучасному екологічному моніторингу займає технологія дистанційного зондування Землі (ДЗЗ), яка дозволяє отримувати об'єктивну інформацію без безпосереднього контакту з об'єктами. До основних компонентів ДЗЗ належать [12]:

- платформи: супутники серій Landsat, Sentinel, безпілотники (дрони), пілотовані літальні апарати;
- сенсори: мультиспектральні, гіперспектральні, тепловізійні, радарні прилади;
- режими знімання: пасивні (відбите сонячне світло) і активні (власне випромінювання, наприклад, радіолокаційні системи).

Дані з ДЗЗ використовуються для моніторингу вирубки лісів, ерозії ґрунтів, зміни водного покриву, спостереження за урбанізаційними процесами, а також для оцінки рівнів забруднення у віддалених регіонах [12].

В умовах глибокої екологічної трансформації, яку переживає Україна внаслідок військових дій, руйнування інфраструктури, знищення природних об'єктів і масштабних техногенних аварій, створення ефективної системи екологічного моніторингу набуває стратегічного значення. Зокрема, в контексті реалізації екологічної стратегії України до 2030 року, необхідно забезпечити впровадження інтегрованих СЕМ на національному рівні, що включають супутниковий моніторинг, мобільні лабораторії, цифрові платформи аналізу та міжнародну координацію [12].

Екологічна безпека повинна стати частиною загальнонаціональної політики відновлення та сталого розвитку. Її основою має бути міждисциплінарний підхід, поєднання знань в екології, інформатиці, математичному моделюванні, інженерії, праві та управлінні. Впровадження таких систем стане ключем до попередження екологічних катастроф, збереження природного багатства та забезпечення здорового середовища для майбутніх поколінь [12].

1.4 Постановка завдань дипломної роботи

Як показали проведені дослідження більшість українських вищих навчальних закладів не мають цифрових платформ, які б дозволяли оперативно збирати, аналізувати та візуалізувати метеорологічні дані, що ускладнює прогнозування генерації енергії з відновлюваних джерел та прийняття обґрунтованих рішень щодо її оптимального використання. Тому, тема роботи є актуальною.

Мета роботи - розробка вебпорталу, що забезпечить збирання, збереження, візуалізацію та порівняльний аналіз локальних та глобальних метеорологічних даних для підтримки зеленої стратегії розвитку університету.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати існуючі підходи у сфері екологічного моніторингу;
- обґрунтувати вибір технологій для реалізації вебпорталу;
- розробити структуру та функціональні модулі програмної системи;
- забезпечити базову інтеграцію з базами локальних та глобальних метеорологічних даних, а також з екологічним чатботом;
- підготувати рекомендації щодо впровадження вебпорталу.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз вимог

У процесі розробки інформаційної системи особливу увагу слід приділяти детальному аналізу вимог до програмного забезпечення, оскільки саме вони формують основу для архітектурних рішень, вибору технологій та структури даних. Аналіз вимог охоплює два основні компоненти: функціональні та нефункціональні вимоги.

Функціональні вимоги описують, що саме система повинна виконувати. У межах проєкту вебпорталу для підтримки «зеленої стратегії» університету до основних функціональних вимог належать:

- збір екологічних даних з сенсорів, розміщених на території університету (температура, вологість, сонячна радіація, швидкість вітру тощо);
- візуалізація даних у вигляді графіків, таблиць і діаграм у вебінтерфейсі користувача;
- інтеграція з метеорологічними службами через API для порівняння локальних і зовнішніх показників;

Нефункціональні вимоги визначають якісні характеристики системи:

- надійність: система повинна стабільно функціонувати у випадку втрати зв'язку з окремими сенсорами;
- продуктивність: інтерфейс має швидко відповідати на запити користувача;
- масштабованість: підтримка додавання нових типів сенсорів без кардинальних змін у структурі системи;
- сумісність: забезпечення коректної роботи в основних браузерах та на різних пристроях.

Для візуального представлення сценаріїв взаємодії користувача із системою використовується UML-діаграма (Unified Modeling Language) прецедентів (рис. 2.1).

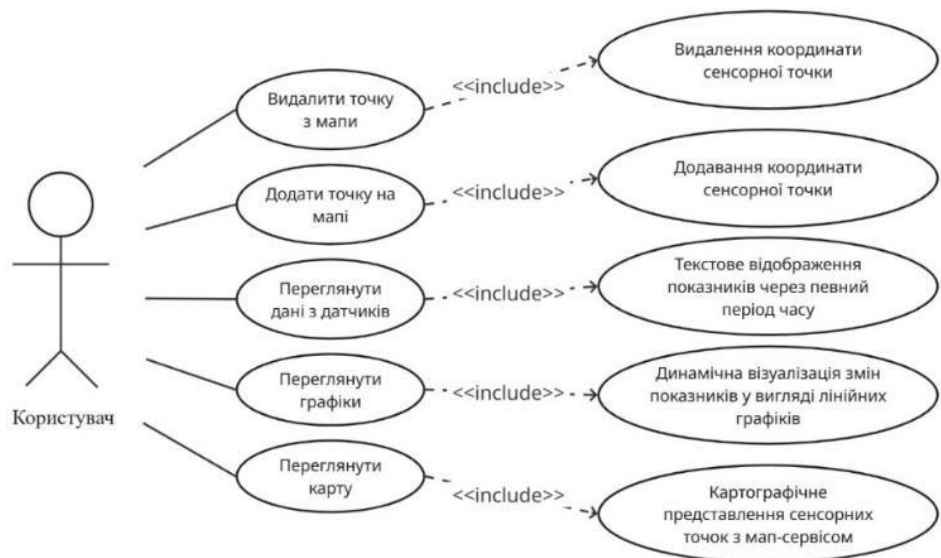


Рисунок 2.1 – UML-діаграма прецедентів

2.2 Архітектура програмного забезпечення

У процесі проєктування вебпорталу для екологічного моніторингу було обрано архітектурний шаблон MVC (Model – View – Controller) (рис. 2.2) як такий, що найкраще відповідає структурі задач і принципам розділення обов’язків у системах із багатокомпонентною логікою. Шаблон MVC дозволяє забезпечити високу гнучкість, масштабованість та підтримуваність системи, що особливо важливо для проєктів, де відбувається активне оновлення інтерфейсу в режимі реального часу [13].

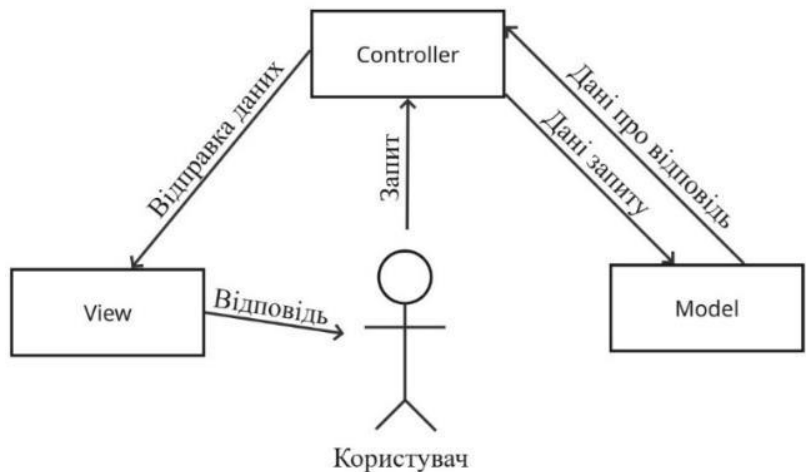


Рисунок 2.2 – Візуальна архітектура шаблону MVC [13]

В таблиці 2.1 показано порівняльні характеристики з іншими архітектурними шаблонами [13]-[16].

Таблиця 2.1 – Порівняння архітектурних шаблонів [13]-[16]

Критерій	MVC [13]	MVP [14]	MVVM [15]	Microservices [16]
Розділення обов'язків	Добре	Помірно	Добре	Чудово
Масштабованість	Висока	Середня	Висока	Дуже велика
Ускладнення реалізації	Помірна	Помірна	Складніша	Висока
Придатність для SPA	Ідеальна	Менш зручна	Ідеальна	Не основне застосування
Придатність для IoT/REST API	Підходить	Обмежено	Підходить	Чудово
Швидкість розгортання	Швидко	Швидко	Повільніше	Довго
Приклад застосування	Вебпортали	Десктоп-додатки	SPA, мобільні	Складні хмарні платформи

На основі таблиці 2.1, можна зробити такий висновок, що з огляду на потребу у швидкому створенні масштабованого вебпорталу з клієнтським інтерфейсом, орієнтованим на інтерактивну візуалізацію, архітектурний шаблон MVC є найоптимальнішим вибором. Його перевагами є чітке розділення відповідальностей між компонентами, висока гнучкість для командної розробки, можливість інтеграції з REST API та простота масштабування. Інші шаблони хоч і мають свої переваги, є надлишковими або складними для задач цього проєкту [13].

У даному проєкті компоненти розподіляються наступним чином:

- model: відповідає за роботу з базою даних MongoDB [17], включаючи структури зберігання показників [13];
- view: інтерфейс користувача, реалізований за допомогою React.js [18], який дозволяє переглядати візуалізовані дані, взаємодіяти з дашбордами [13];

– controller: логіка обробки запитів на стороні сервера (Node.js [19] + Express [20]), включаючи маршрутизацію, перевірку авторизації, обробку API-запитів до метеостанцій [13].

На рисунку 2.3 зображено архітектуру IoT-системи екологічного моніторингу.

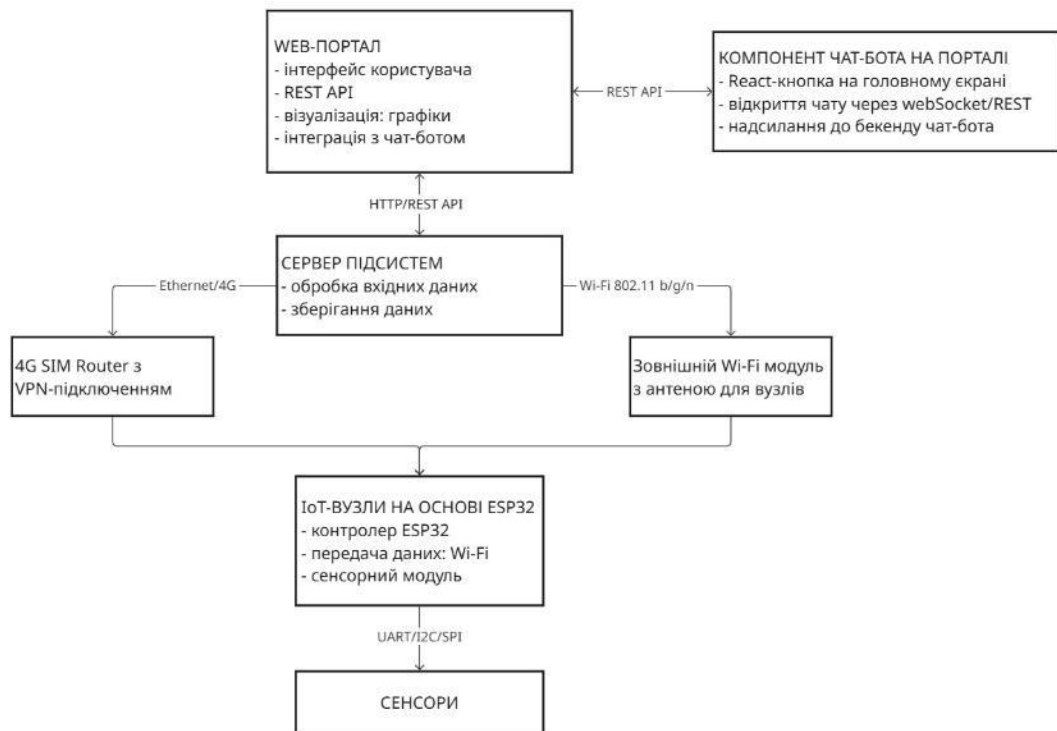


Рисунок 2.3 – Архітектура IoT-системи екологічного моніторингу

Також система включає:

- модуль збору даних: отримує дані з сенсорів через IoT-протоколи;
- інтеграційний модуль: забезпечує підключення до зовнішніх метеослужб;
- сервіс візуалізації: побудова графіків і діаграм з використанням бібліотек Chart.js [21].

Таким чином, застосування шаблону MVC дозволяє не лише ефективно розділити відповідальність між логікою, інтерфейсом та даними, але й підвищує стабільність, надійність і прозорість розробки системи, що критично

важливо у межах програмного продукту з довготривалим життєвим циклом [13].

Також у межах проєкту вебпорталу екологічного моніторингу важливо передбачити можливості інтеграції як з внутрішніми інформаційними ресурсами університету, так і з зовнішніми екосистемами цифрових сервісів. Така інтеграція розширює функціональність системи, забезпечує автоматизацію обміну даними та сприяє досягненню міжсистемної сумісності:

а) інтеграція з внутрішньою IT-інфраструктурою університету:

1) аутентифікація користувачів: система може бути підключена до корпоративної служби ідентифікації, що забезпечить реалізацію механізму єдиного входу для студентів, викладачів та персоналу;

2) публічне розміщення порталу: інтеграція з основним сайтом закладу вищої освіти сприятиме популяризації екологічних ініціатив та забезпечить відкритий доступ до результатів моніторингу;

б) інтеграція з зовнішніми цифровими сервісами:

1) API метеорологічних служб, які забезпечують актуальні погодні дані. Дані отримуються у форматі JSON через REST-запити та використовуються для порівняння з локальними сенсорними показниками;

2) сервіси картографії: для візуалізації просторових даних використовуються інтерактивні платформи, що дозволяють розміщувати маркери сенсорів на мапі, будувати зони впливу, переглядати стан об'єктів у географічному контексті;

в) можливість масштабування: система проєктувалась із дотриманням принципів модульності та гнучкої архітектури, що дає змогу:

1) легко інтегрувати нові типи сенсорів або зовнішні джерела даних;

2) розширити функціональність за рахунок мобільного додатку або хмарного інтерфейсу;

3) підтримувати зв'язок з іншими ініціативами «розумного кампусу» на базі закладу.

2.3 Обґрунтування вибору технологій

Розробка вебпорталу екологічного моніторингу потребує вибору оптимального стеку технологій, який забезпечить стабільність, масштабованість, адаптивність до реального часу та зручність візуалізації даних. Нижче наведено порівняльний аналіз ключових інструментів, які могли бути застосовані в проєкті, та обґрунтовано вибір конкретних рішень. В таблиці 2.2 [18], [22]-[23] показано порівняльні характеристики фронтенд-технологій в таблиці 2.3 [19]-[20], [24]-[25] показано порівняльні характеристики серверних платформ.

Таблиця 2.2 – Порівняння фронтенд-технологій [18], [22]-[23]

Критерій	React.js [18]	Vue.js [22]	Angular [23]
Продуктивність	Висока	Висока	Середня
Крива навчання	Помірна	Низька	Висока
Компонентна структура	Так	Так	Так
Розмір спільноти	Дуже велика	Велика	Велика
Гнучкість інтеграції	Висока	Висока	Складніша інтеграція
Підтримка екосистеми	Багата	Обмежена	Вбудовані рішення

Таблиця 2.3 – Порівняння серверних платформ [19]-[20], [24]-[25]

Критерій	Node.js [19] + Express[20]	Diango [24]	Laravel [25]
Асинхроність	Так	Ні	Обмежена
Продуктивність	Продуктивність	Середня	Середня
Гнучкість	Висока	Висока	Середня
Складність налаштування	Низька	Помірна	Середня
Спільнота	Велика	Середня	Низька
REST API з коробки	Так	Так	Так

В таблицях 2.4-2.7 показано порівняльні характеристики баз даних, бібліотек візуалізації, засобів контейнеризації та систем контролю версій.

Таблиця 2.4 – Порівняння баз даних [17], [26]-[27]

Критерій	MongoDB [17]	PostgreSQL [26]	MySQL [27]
Підтримка JSON	Так	Так	Часткова
Масштабованість	Висока	Середня	Середня
Гнучкість схеми	Дуже гнучка	Менш гнучка	Менш Гнучка
Продуктивність при IoT	Висока	Середня	Середня

Таблиця 2.5 – Порівняння бібліотек візуалізації [21], [28]-[29]

Критерій	Chart.js [21]	D3.js [28]	Google Charts [29]
Простота використання	Висока	Низька	Середня
Підтримка анімації	Так	Так	Так
Інтерактивність	Обмежена	Висока	Обмежена
Час роботи	Короткий	Довгий	Середній
Підтримка React	Так	Так	Обмежена

Таблиця 2.6 – Порівняння засобів контейнеризації [30]-[32]

Критерій	Docker/Compose [30]	Vagrant [31]	Kubernetes [32]
Простота налаштування	Висока	Середня	Низька
Швидкість розгортання	Висока	Середня	Висока
Придатність до локальної розробки	Так	Так	Ні

Таблиця 2.7 – Порівняння систем контролю версій [33]-[35]

Критерій	Git [33]	Subversion [34]	Mercurial [35]
1	2	3	4
Тип системи	Розподілена	Централізована	Розподілена
Продуктивність	Висока	Середня	Висока

Кінець таблиці 2.7 [33]-[35]

1	2	3	4
Робота офлайн	Так	Ні	Так
Гнучкість гулкування	Висока	Обмежена	Середня
Популярність та підтримка	Найпопулярніша	Менша	Низька
Інтеграція з Github/GitLab	Повна	Часткова	Обмежена
Навчальна база/документація	Обширна	Обмежена	Обмежена

З представлених у таблицях варіантів було обрано такі технології: React.js [18], Node.js [19] з Express [20], MongoDB, Chart.js [21], Docker [30] та Git [33]. Ці інструменти забезпечують гнучкість архітектури, адаптацію до динамічних даних, масштабованість, простоту інтеграції з сенсорами IoT та сторонніми API, а також сприяють швидкому розгортанню проєкту в різних середовищах. Порівняльний аналіз доводить, що альтернативи поступаються за ключовими характеристиками, що критично важливо для стабільної роботи системи екологічного моніторингу.

2.4 Висновки до розділу 2

У результаті аналізу було сформульовано функціональні й нефункціональні вимоги до програмного забезпечення, що розробляється. Запропоновано архітектурну модель, яка забезпечує ефективну організацію обміну даними між компонентами системи, та обґрунтовано вибір відповідних технологій для реалізації кожного з елементів.

Вебпортал екологічного моніторингу дозволить інтегрувати цифрові дані з локальних сенсорів та метеорологічних служб, забезпечити їх зберігання, обробку й візуалізацію, а також надати користувачам зручні інструменти для аналітики та прийняття управлінських рішень у межах «зеленої стратегії» університету.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Загальна структура програмної системи

Розроблена система екологічного моніторингу побудована як багаторівнева вебплатформа, що складається з клієнтської частини (frontend), серверної частини (backend), бази даних, сенсорної підсистеми збору даних та окремого компонента чат-бота. Кожен компонент виконує окремі функції та взаємодіє з іншими за допомогою REST API.

Основні рівні системи (рис. 3.1):

- frontend (React.js [18]): відповідає за побудову інтерфейсу користувача, відображення даних з сенсорів у вигляді графіків, таблиць та мап;
- backend (Node.js [19] + Express [20]): обробляє запити клієнта, виконує логіку отримання та агрегації даних, формує відповіді у форматі JSON;
- база даних (MongoDB [17]): зберігає значення екологічних параметрів, отриманих з IoT-сенсорів, а також інформацію про точки моніторингу;
- IoT-вузли: сенсорні пристрої на базі ESP32 передають дані на сервер через Wi-Fi або інші бездротові інтерфейси;
- зовнішні API: використовується сторонній метеосервіс для порівняння локальних екологічних показників із загально регіональними;
- чат-бот (Laravel [25] + MySQL [27]): інтегрований у портал у вигляді окремого модуля; забезпечує навігацію по інформаційним розділам за допомогою кнопок та пошуку.

Завдяки архітектурі типу MVC система залишається масштабованою та модульною — кожен компонент може розвиватися незалежно.

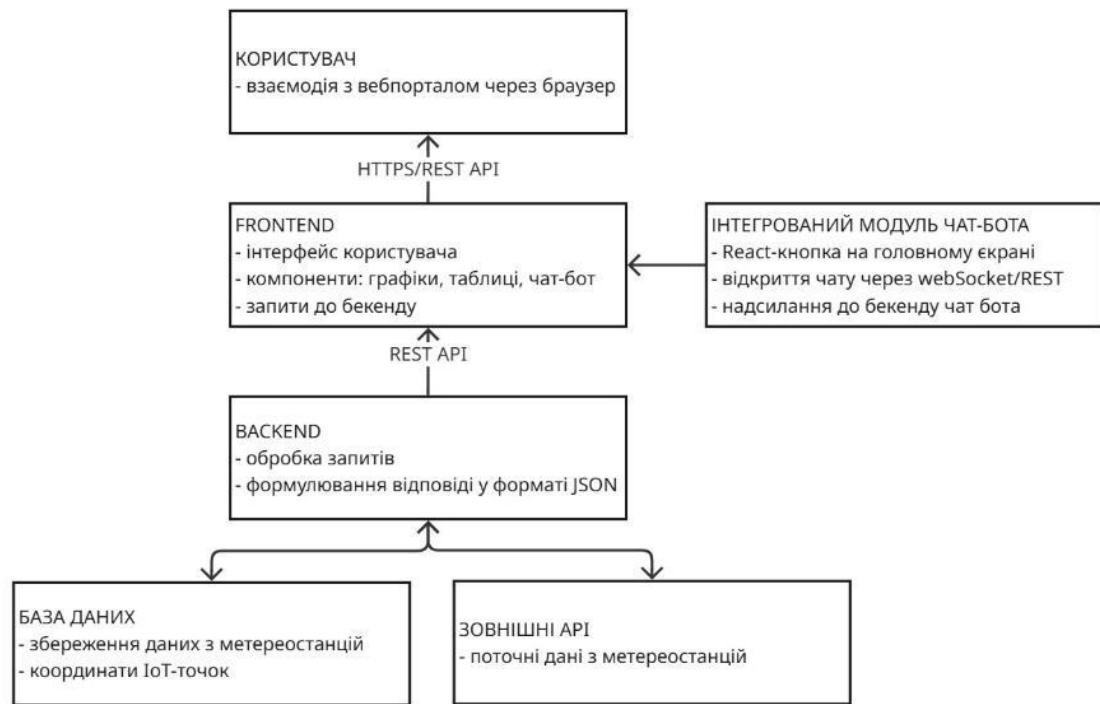


Рисунок 3.1 – Структурна схема системи

3.2 Реалізація клієнтської частини

Клієнтська частина системи реалізована з використанням бібліотеки React.js, що дозволяє побудувати гнучкий, динамічний та адаптивний інтерфейс [18].

Ключові компоненти:

- головний екран: відображає загальний стан екосистеми, включаючи останні дані з сенсорів;
- компонент карти: показує розташування сенсорних точок;
- графіки та аналітика: реалізовані з використанням бібліотеки Chart.js [21], забезпечують візуалізацію температури, швидкості та напрямку вітра, хмарності, атмосферного тиску.
- таблиці: відображення останніх сім записів та останні сім днів в табличному вигляді;
- чат-бот: реалізовано як окремий React-компонент, вбудований у нижній кут інтерфейсу, що відкриває діалогове вікно з навігацією по екологічній інформації.

Взаємодія з backend здійснюється за допомогою Axios [36], запити надсилаються до REST API, відповіді приходять у форматі JSON і обробляються у відповідних компонентах.

У таблиці 3.1 відображено детальний опис компонентів клієнтської частини (Додаток Б.1-Б.7).

Таблиця 3.1 – Опис компонентів клієнтської частини

Назва компонента	Призначення	Функціональність	Взаємодія з API
1	2	3	4
Graphs	Головний компонент для візуалізації екологічних даних у вигляді графіків. Він агрегує та відображає дані з різних джерел, таких як локальні метеорологічні сенсори та OpenWeatherMap	- ініціалізує запити до API для отримання історичних та поточних даних; - відображає графіки температури, швидкості вітру, напрямку вітру, хмарності, атмосферного тиску; - дозволяє користувачу зробити аналіз 30 днів; - використовує бібліотеку Chart.js для побудови графіків	Отримує дані з бекенду через маршрути GET /api/localMeteorological/period/30-days та GET /api/openWeatherMap/period/30-days
LastSevenDays	Компонент для відображення середніх значень екологічних параметрів за останні 7 днів	- надсилає запит до API для отримання агрегованих даних за 7 днів; - відображає дані у вигляді таблиці; - може включати показники, такі як середня температура, середня швидкість вітру, середній напрямок вітру, середню хмарність, середнє значення сонячної радіації, середнє значення точки роси, середнє значення атмосферного тиску	Отримує дані з маршруту GET /api/localMeteorological/period/7-days
LastSevenEntries	Компонент для відображення останніх 7 записів екологічних даних	- надсилає запит до API для отримання останніх 7 записів; - відображає дані у вигляді таблиці з часовими мітками. - дозволяє користувачу швидко переглянути останні зміни в екологічних параметрах	Отримує дані з маршруту GET /api/localMeteorological/period/7-latest
LocalMeteorological	Компонент для відображення поточних локальних метеорологічних даних, отриманих з Home Assistant	-надсилає запит до API для отримання поточних даних; - відображає параметри, такі як температура, швидкість вітру, напрям вітру, хмарність, сонячна радіація, точка роси, атмосферний тискЮ останнє оновлення; - включає візуальні іконки погодних умов для покращення сприйняття даних	Отримує дані з маршруту GET /api/localMeteorological/

Кінець таблиці 3.1

1	2	3	4
OpenWeatherMap	Компонент для відображення поточних погодних даних, отриманих з OpenWeatherMap API	- надсилає запит до API для отримання поточних погодних даних; - відображає параметри, такі як температура, відчутна температура, швидкість вітру, напрям вітру, пориви вітру, атмосферний тиск, тиск на рівні землі, тиск на рівня моря, вологість повітря, хмарність, видимість, останнє оновлення; - включає візуальні іконки погодних умов для покращення сприйняття даних	Отримує дані з маршруту GET /api/openWeatherMap
Map	Компонент для відображення мапи з точками моніторингу екологічних даних	- відображає інтерактивну мапу з маркерами, що представляють точки моніторингу; - дозволяє користувачу взаємодіяти з маркерами (перегляд, додавання, видалення); - використовує бібліотеку Leaflet для роботи з мапами	- отримує дані про точки моніторингу з маршруту GET /api/point; - додає нову точку моніторингу по маршруту POST /api/point; - видаляє точку моніторингу по маршруту DELETE /api/point/:id
GreenStrategyPage	Компонент виконує функцію центрального хабу для візуалізації, порівняння та моніторингу екологічних даних. Він поєднує всі основні функціональні компоненти порталу, забезпечуючи зручний та узгоджений інтерфейс користувача для перегляду даних, графіків, мап та погодних звітів	- відображає: 1) поточні локальні метеодані з Home Assistant (через компонент LocalMeteorological); 2) дані з OpenWeatherMap для зовнішнього порівняння (OpenWeatherMap); 3) останні 7 записів із сенсорів (LastSevenEntries); 4) агреговані дані за 7 днів (LastSevenDays); 5) інтерактивні графіки з аналітикою (Graphs); 6) мапу з точками моніторингу (Maps). - дозволяє: 1) перемикає мову інтерфейсу (інтеграція з i18n); 2) оцінювати відхилення між локальними та зовнішніми метеоданими; 3) переглядати історичні значення у вигляді графіків; 4) інтерактивно працювати з картою (перегляд, додавання/видалення точок)	- поточні локальні дані GET /api/localMeteorological; - поточні дані з Open Weather Map: GET /api/openWeatherMap; - останні 7 записів локальних даних: GET /api/localMeteorological/period/7-latest; - агреговані дані за певний період для локальних даних: GET /api/localMeteorological/period/:period; - агреговані дані за певний період для Open Weather Map даних: GET /api/openWeatherMap/period/:period; - отримує дані про точки моніторингу з маршруту GET /api/point; - додає нову точку моніторингу по маршруту POST /api/point; - видаляє точку моніторингу по маршруту DELETE /api/point/:id

3.3 Реалізація серверної частини

Серверна логіка реалізована за допомогою Node.js [19] у поєднанні з Express.js [20], що дозволяє швидко створювати масштабовані API-сервіси.

Основні функції:

- обробка запитів з клієнтської частини: маршрутизація запитів, формування відповідей;
- зв'язок з базою даних MongoDB [17];
- інтеграція з метеосервісами: періодичні запити до OpenWeatherMap [37];
- обробка даних із сенсорів: система отримує нові записи від IoT-модулів через POST-запити, зберігає їх у БД.

Архітектура забезпечує розділення відповідальності за бізнес-логіку, роботу з базою та зовнішніми джерелами.

У таблиці 3.2 відображено детальний опис функцій на серверній частині (Додаток Б.8-Б.16).

Таблиця 3.2 – Опис функцій

Назва функції	Маршрут	Вхідні параметри	Джерело даних/логіка	Формат відповіді
1	2	3	4	5
getLocalMeteorologicalData	GET /api/localMeteorological/	–	Отримує дані з Home Assistant, знаходиться потрібну entity_id, витягує атрибути	JSON: - позитивний результат: data: {state, last_updated, attributes: { temperature, wind_speed, wind_bearing, cloud_coverage, uv_index, dew_point, pressure, }, } - негативний результат: { error: 'Forecast data not found' } або { error: 'Local Meteorological request error' }

Продовження таблиці 3.2

1	2	3	4	5
addLocalMeteorologicalDataToBD	–	–	Отримує поточні погодні дані з Home Assistant і записує в MongoDB модель LocalMeteorological	JSON - позитивний результат: { message: 'Local data saved successfully' } - негативний результат: { error: 'Forecast data not found' } або { error: 'Local Meteorological request error' }
getLocalMeteorologicalDataByPeriod	GET /api/localMeteorological/period/:period	period	За константними змінами SEVEN_LATEST, SEVEN_DAYS, THIRTY_DAYS робить вибірку або агрегацію з MongoDB відповідно	JSON - позитивний результат: { data: { data } } - негативний результат: { error: 'Invalid period parameter' } або { error: 'Server error' }
getOpenWeatherMapData	GET /api/openWeatherMap	–	Звертається до OpenWeatherMap API (місто: Zaporizhia) через API ключ	JSON - позитивний результат: { data: { data } } - негативний результат: { error: data.message } або { error: 'OpenWeatherMap API request error' }
addOpenWeatherMapDataToDB	–	–	Отримує погодні дані через OpenWeather API, парсить, трансформує, зберігає у MongoDB модель OpenWeatherMap	JSON - позитивний результат: { message: 'OpenWeatherMap data saved successfully' } - негативний результат: { error: data.message } або { error: 'OpenWeatherMap API request error', details: err.message }
getOpenWeatherMapDataByPeriod	GET /api/openWeatherMap/period/:period	period	За константною змінною THIRTY_DAYS робить агрегацію з MongoDB	JSON - позитивний результат: { data: data } - негативний результат: { error: 'Invalid period parameter' } або { error: 'Server error' }
createPoint	POST /api/point	body	Створює нову точку моніторингу за вхідними параметрами (position: {lat, lng}, sensor, street)	JSON - позитивний результат: { data: createPoint }

Кінець таблиці 3.2

1	2	3	4	5
getPoints	GET /api/point	–	Отримує список усіх точок моніторингу з колекції Point	JSON - позитивний результат: { data: points }
deletePoint	DELETE /api/point/:id	id	Видаляє точку з MongoDB колекції Point по її ObjectId	JSON - позитивний результат: { data: deletePoint } - негативний результат: { message: 'Point not found' }

3.4 Робота з базою даних

Для зберігання екологічних даних використовується MongoDB [17] — документоорієнтована база, що добре підходить для записів із сенсорів.

У таблиці 3.3 наведено опис MongoDB-колекцій (Додаток Б.17-Б.19).

Таблиця 3.3 – Опис MongoDB-колекцій

Назва колекції	Структура	Призначення	Особливості
1	2	3	4
LocalMeteorological	- state: <i>String</i> – загальний стан погоди - temperature: <i>Number</i> – температура - windSpeed: <i>Number</i> – швидкість вітру - windBearing: <i>Number</i> – напрям вітру - cloudCoverage: <i>Number</i> – хмарність - solarRadiation: <i>Number</i> – індекс сонячної радіації - dewPoint: <i>Number</i> – точка роси - pressure: <i>Number</i> – атмосферний тиск - lastUpdated: <i>Date</i> – час останнього оновлення	Зберігання локальних метеорологічних даних з Home Assistant. Використовуються для порівняння з зовнішніми джерелами	Дані надходять від IoT-серверу університету у форматі JSON через REST-запит
OpenWeatherMap	- state: <i>String</i> – загальний стан погоди - temperature: <i>Number</i> – температура - temperatureFeelsLike: <i>String</i> – температура по відчуттях - windSpeed: <i>Number</i> – швидкість вітру - windBearing: <i>Number</i> – напрям вітру - windGust: <i>Number</i> – пориви вітру - pressure: <i>Number</i> – атмосферний тиск - pressureAtGroundLevel: <i>Number</i> – атмосферний тиск на рівні землі - pressureAtSeaLevel: <i>Number</i> – атмосферний тиск на рівні моря - humidity: <i>Number</i> – вологість повітря - cloudCoverage: <i>Number</i> – хмарність - visibility: <i>Number</i> – видимість у метрах - lastUpdated: <i>Date</i> – час останнього оновлення	Зберігання метеоданих із зовнішнього API OpenWeatherMap. Використовується для порівняльного аналізу з локальними даними	Дані приходять з публічного API у форматі JSON; підтримується агрегація по періодах

Кінець таблиці 3.3

1	2	3	4
Point	- position.lat: <i>Number</i> – широта точки - position.lng: <i>Number</i> – довгота точки - sensor: <i>String</i> – тип сенсора - street: <i>String</i> – назва вулиці або локації	Містить координати встановлених сенсорів для відображення на інтерактивній мапі	Підтримується CRUD-операції для редагування на карті через інтерфейс користувача

3.5 Обробка та візуалізація екологічних даних

Ефективна робота екологічного порталу передбачає не лише збір сенсорних показників, але й їхню своєчасну обробку, агрегацію, аналіз та зручне представлення кінцевому користувачу. З огляду на великий обсяг даних, що надходить із сенсорів у реальному часі, виникає потреба в структурованому процесі трансформації сирової інформації у наочні й аналітично цінні візуалізації. Така обробка має забезпечити як миттєве ознайомлення з актуальним станом мікроклімату, так і історичний аналіз динаміки змін.

Обробка екологічних даних у розробленій системі реалізована у кілька ключових етапів:

- отримання з БД: дані вибираються з колекцій MongoDB залежно від параметрів запиту користувача (наприклад, останні 7 значень, середні показники за 7 днів або 30 днів). Відбір здійснюється за діапазоном часу з урахуванням часу останнього оновлення `lastUpdated`;
- попередня агрегація: для економії ресурсів на клієнтській стороні агрегування даних виконується вже на рівні сервера. За допомогою механізмів MongoDB Aggregation Pipeline [17] проводиться групування показників по годинах, днях або інших інтервалах. Це дозволяє формувати узагальнені статистичні значення, зокрема середню температуру, тиск, вологість тощо за певний період.

- передача у frontend: підготовлені дані надсилаються клієнтській частині у форматі масиву об'єктів JSON, кожен з яких містить ключі з екологічними параметрами та часовими мітками.

Після отримання з сервера, оброблені дані відображаються у вигляді:

- графіків: за допомогою бібліотеки Chart.js [21] реалізовано побудову лінійних графіків, які дозволяють користувачу зручно спостерігати зміну температури, швидкості вітру, напрямку вітру, хмарності та атмосферного тиску у динаміці;

- таблиць: для відображення числових значень використано адаптивні таблиці, побудовані на базі React-компонентів. Табличне представлення дозволяє переглядати детальні значення по днях або годинах;

- інтерактивної мапи: за допомогою Leaflet [38] реалізовано відображення розташування сенсорних точок на території університету. Мапа включає маркери з даними, а також heatmap для візуалізації інтенсивності показників.

Реалізований функціонал дозволяє користувачам:

- порівнювати дані з метеостанціями: користувач може бачити як локальні сенсорні показники, так і погодні дані з OpenWeatherMap чи локальної метеостанції, що дозволяє визначити локальні відхилення;

- переглядати історію значень: доступ до агрегованих історичних даних дає змогу аналізувати зміни у мікрокліматі впродовж тижня або місяця;

- бачити аномальні зміни чи пікові значення: завдяки графічному представленню можна швидко виявити сплески температури, перепади тиску або інші потенційно небезпечні ситуації.

Таким чином, обробка та візуалізація екологічних даних у проєкті дозволяє реалізувати інформативний, адаптивний та наочний інструмент для моніторингу довкілля, який може бути використаний як у навчальних цілях, так і для прийняття управлінських рішень.

На рисунку 3.3 зображено візуально етапи обробки даних

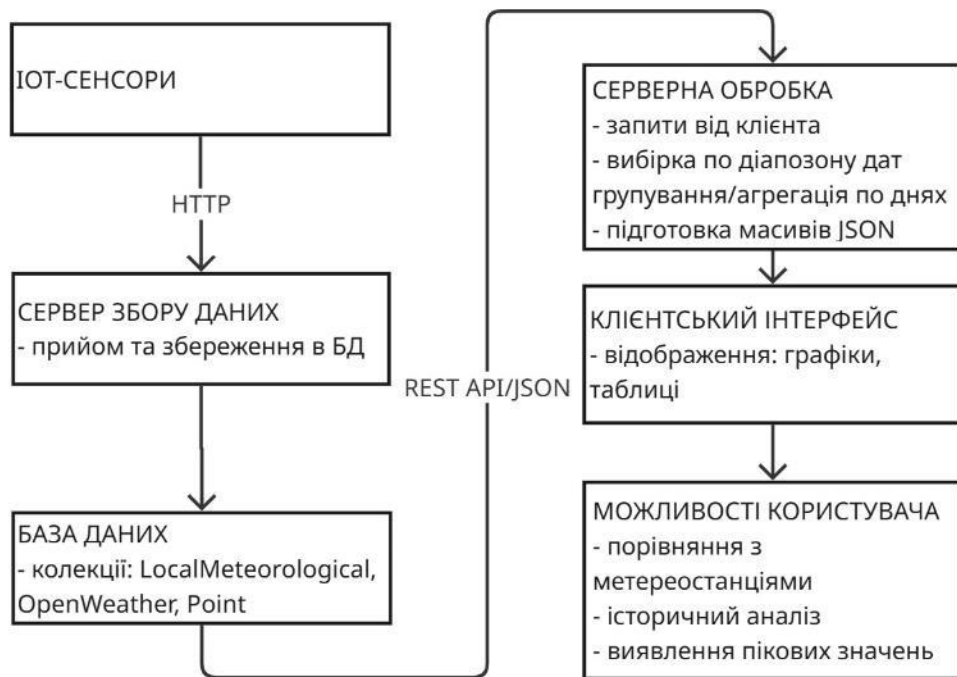


Рисунок 3.2 – Етапи обробки даних

3.6 Інтеграція чатбота у вебпортал

Чат-бот реалізовано як автономний компонент у межах порталу, що дозволяє інтегрувати його незалежно від основної логіки.

Особливості:

а) frontend:

- 1) React-компонент відкриває вікно чат-бота [18];
- 2) здійснює запити до бекенду чат-бота (Laravel API [25]);

б) backend чат-бота (Laravel [25]):

- 1) приймає запити через REST API;
- 2) формує відповіді з бази даних MySQL [27];

в) навігація:

- 1) динамічні кнопки: «FAQ», «Академ. ресурси»;
- 2) можливість пошуку за ключовими словами.

Завдяки окремому API, чат-бот може оновлюватися незалежно від основної частини порталу, а його функціональність — розширюватись з урахуванням потреб екологічної комунікації.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ
ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Інструкція користувача

Вебпортал для екологічного моніторингу призначений для кінцевих користувачів, які зацікавлені у перегляді даних про стан навколишнього середовища в межах території університету.

На рисунку 4.1 зображено функціонал зміни мови (англійська, українська) порталу, а саме через випадний список меню.

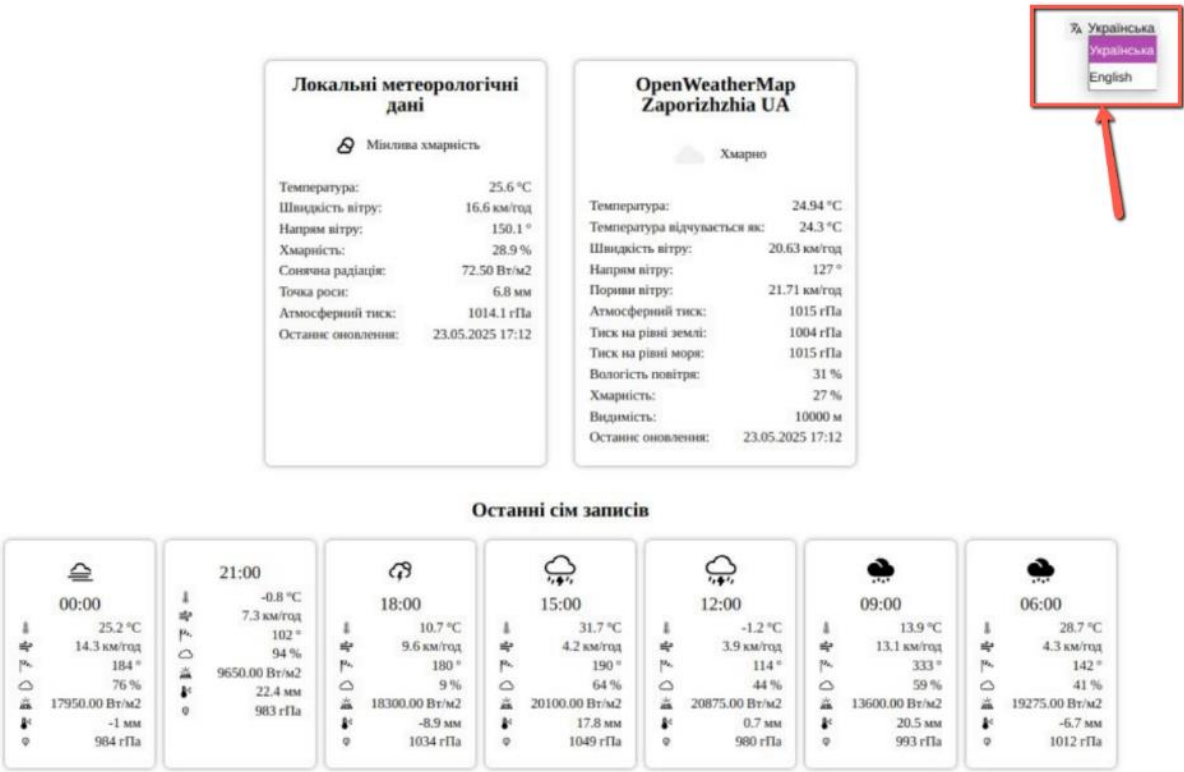


Рисунок 4.1 – Випадний список зі зміненням мови

Переглянути поточні локальні та Open Weather Map метеорологічні дані можна в першому блоці порталу (рис. 4.2-4.3).

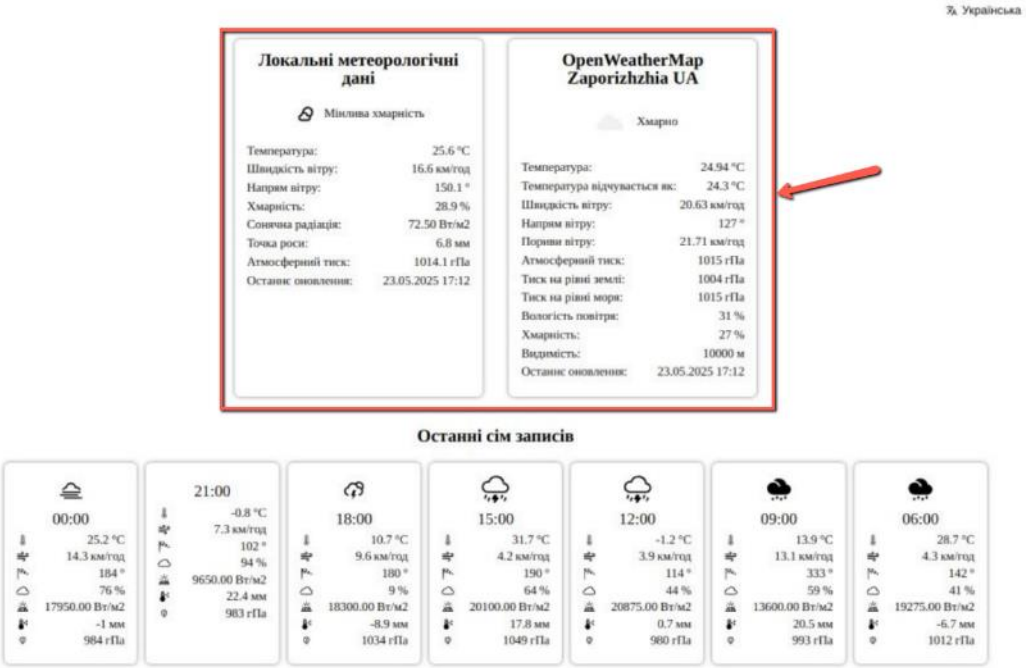


Рисунок 4.2 – Поточні дані (українською)

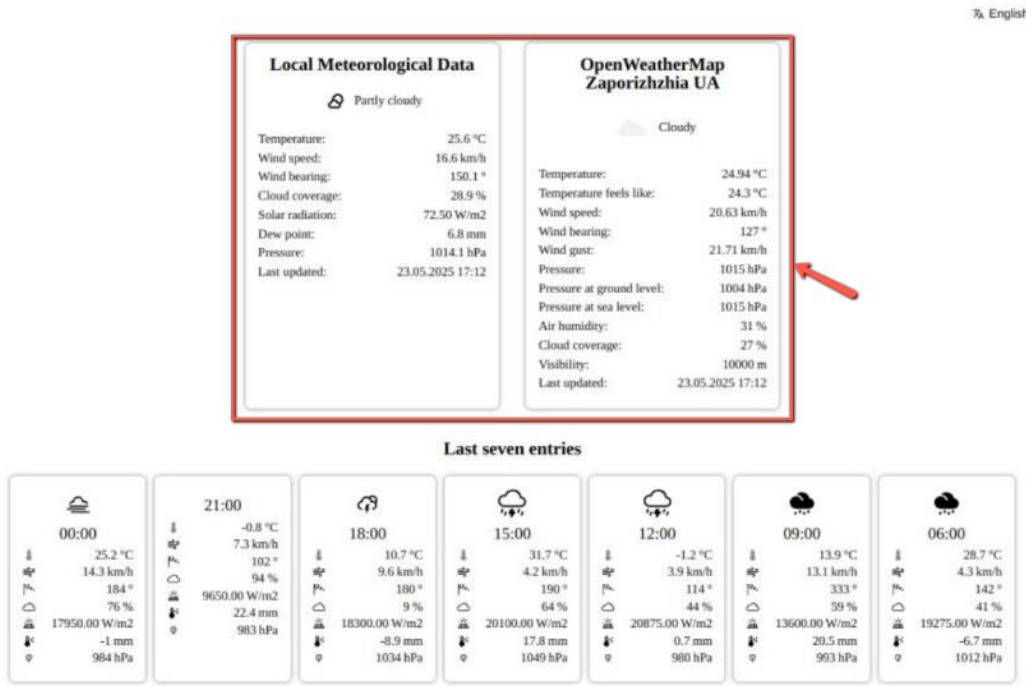


Рисунок 4.3 – Поточні дані (англійською)

В наступній секції вебдодатку можна переглянути та проаналізувати дані за останні 7 записів та за останні 7 днів (рис. 4.4-4.5).



Рисунок 4.4 – Дані за останні 7 записів та 7 днів (українською)



Рисунок 4.5 – Дані за останні 7 записів та 7 днів (англійською)

В наступному блоці застосунку можна побачити порівняльні графіки де відображено середні локальні та Open Weather Map дані за останні 30 днів (рис 4.6-4.7).

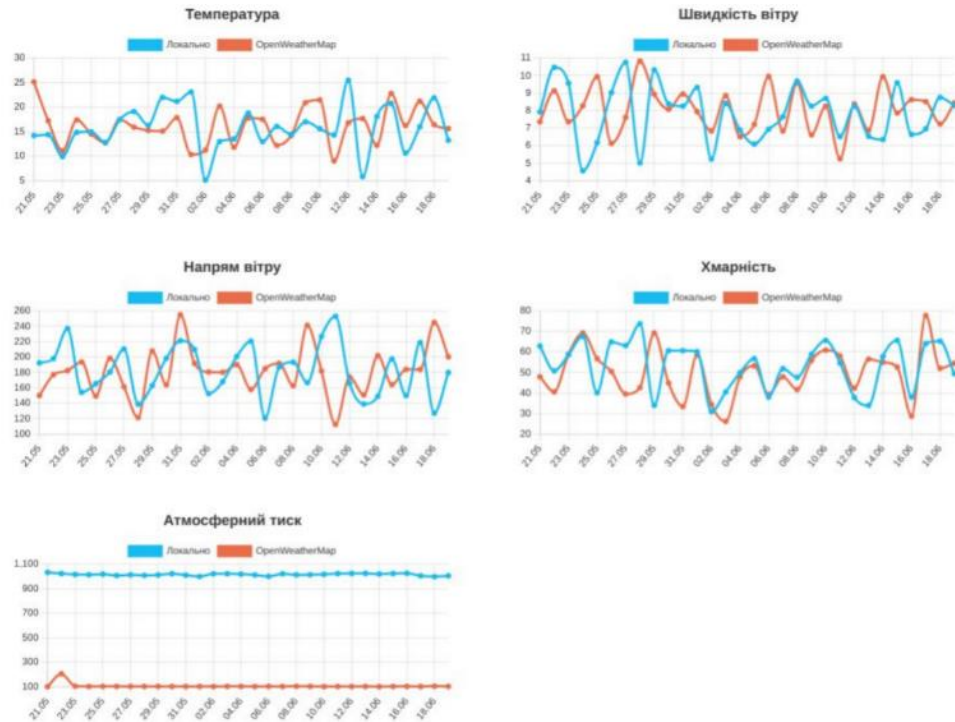


Рисунок 4.6 – Порівняльні графіки (українською)

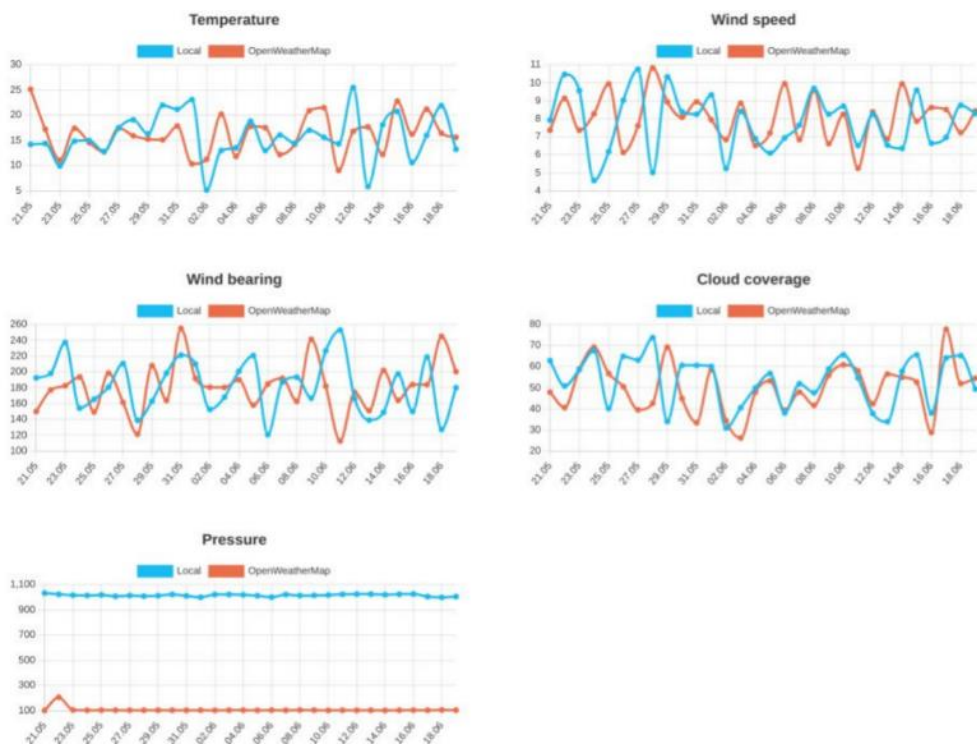


Рисунок 4.7 – Порівняльні графіки (англійською)

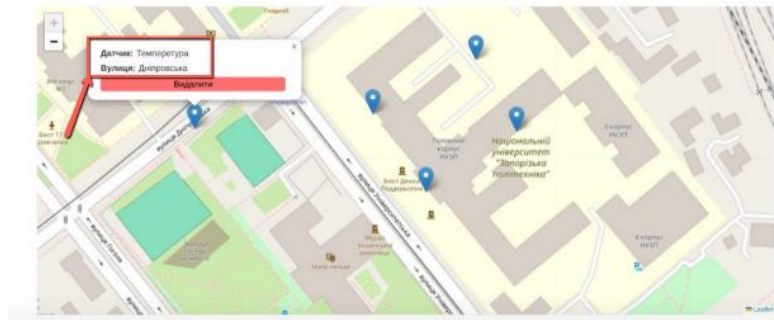


Рисунок 4.10 – Інформаційні поля точки (українською)

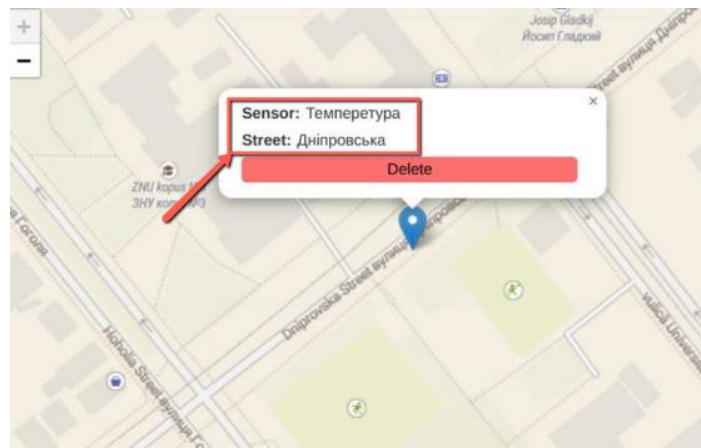


Рисунок 4.11 – Інформаційні поля точки (англійською)

На рисунках 4.12 та 4.13 відображено коректне заповнення полів для створення нової точки на карті. При натисненні кнопки «Зберегти» або «Save» в залежності від вибраної мови, точка збережеться в базу даних і буде відображена на карті де можна її буде переглянути (рис. 4.14-4.15).

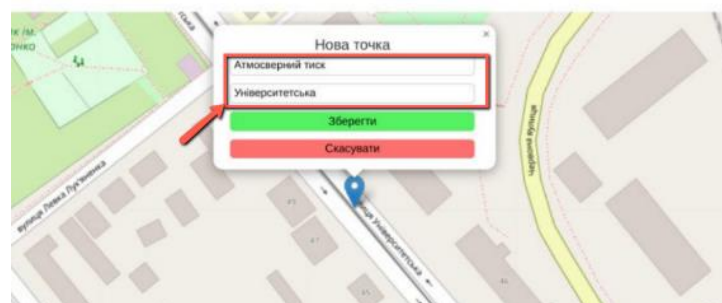


Рисунок 4.12 – Заповнення коректно полів форми на створення точки (українською)

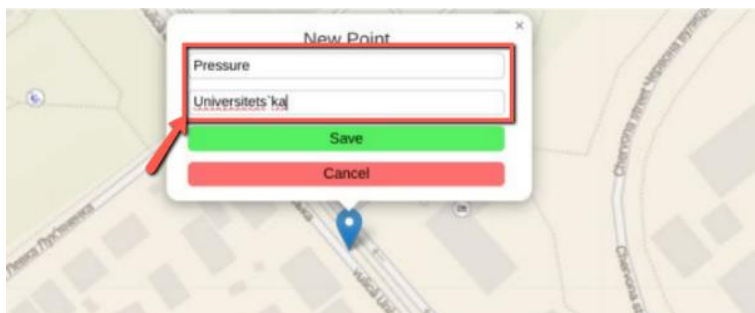


Рисунок 4.13 – Заповнення коректно полів форми на створення точки (англійською)

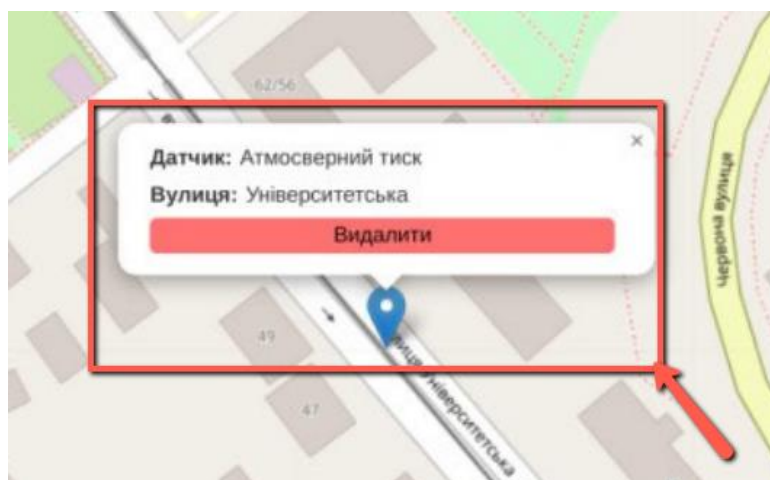


Рисунок 4.14 – Перегляд новоствореної точки (українською)

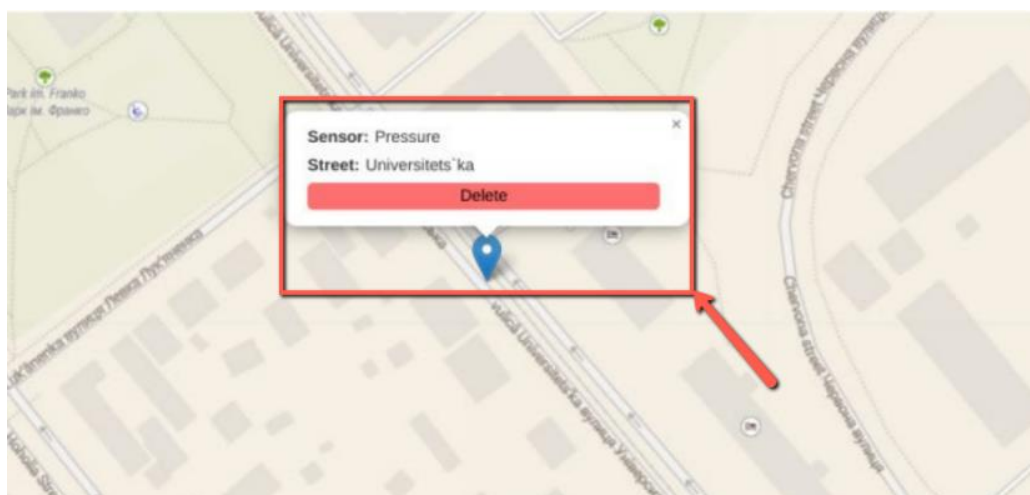


Рисунок 4.15 – Перегляд новоствореної точки (англійською)

Якщо поля форми створення точки залишити всі пустими або лишити одне поле пусте і натиснути кнопку «Зберегти» або «Save» в залежності від

вибраної мови, то з'явиться спливаюче повідомлення про попередження, щоб закрити його потрібно натиснути кнопку «OK» на цьому повідомленні (рис. 4.16-4.17).

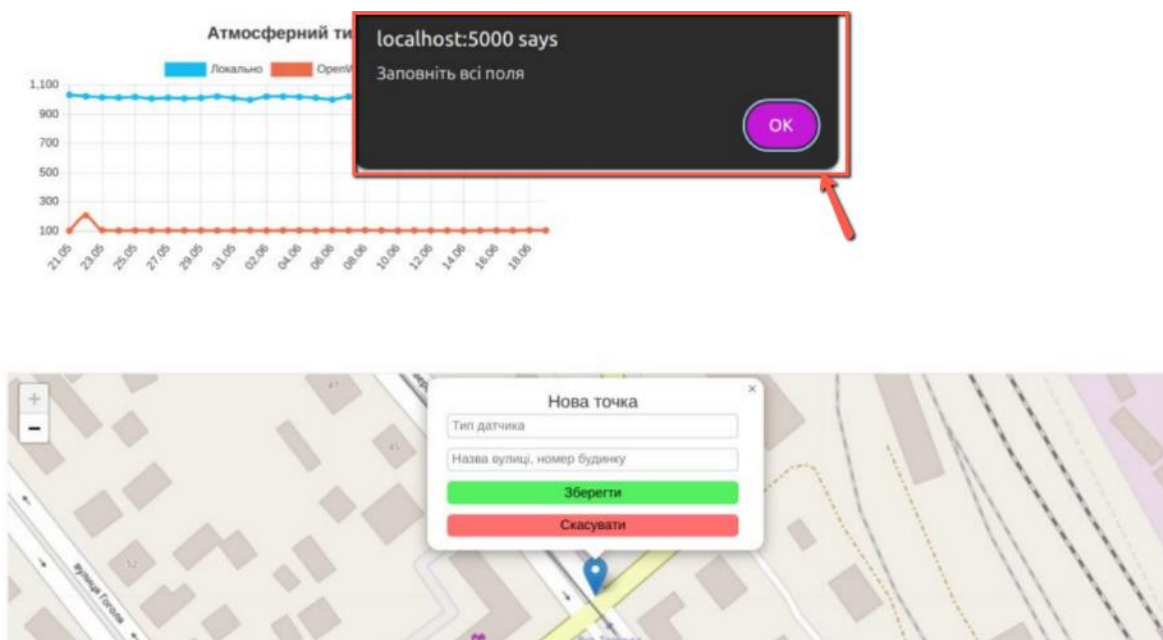


Рисунок 4.16 – Спливаюче попереджувальне повідомлення (українською)

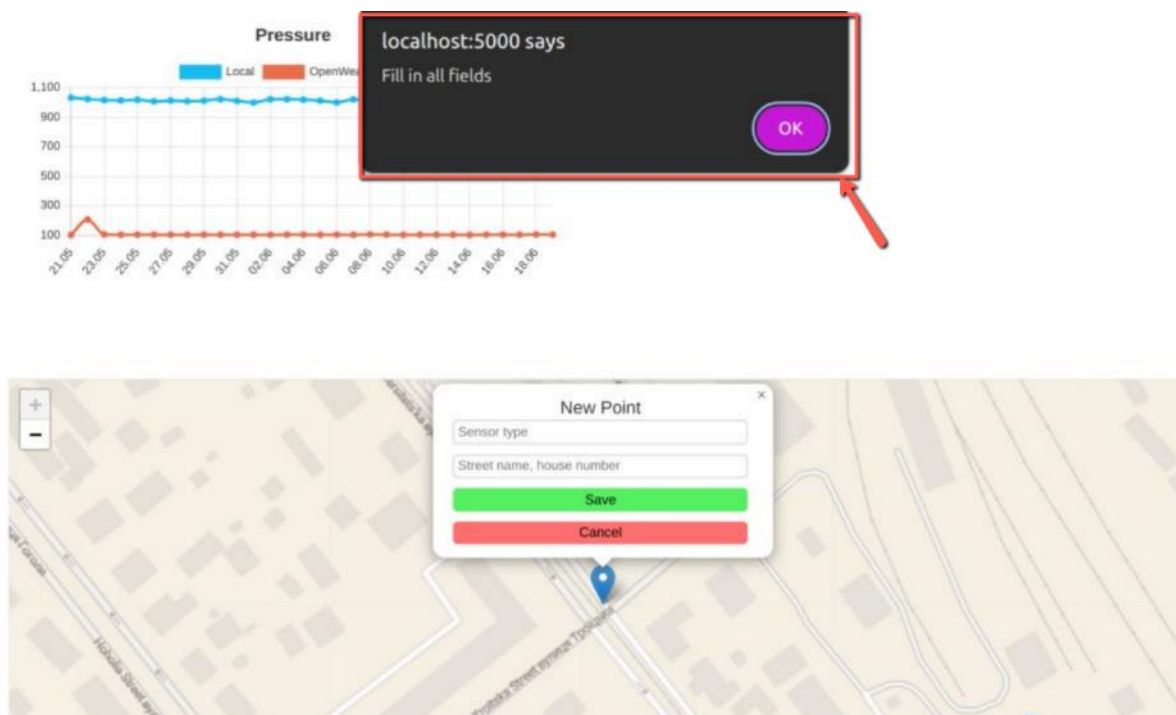


Рисунок 4.17 – Спливаюче попереджувальне повідомлення (англійською)

Для видалення точки з мапи та з самої бази даних, потрібно клікнути на саму точку, відкриється інформаційні поля про точку а під ними буде кнопка «Видалити» або «Delete», в залежності від вибраної мови, яка відповідає за видалення точки (рис. 4.18-4.19).

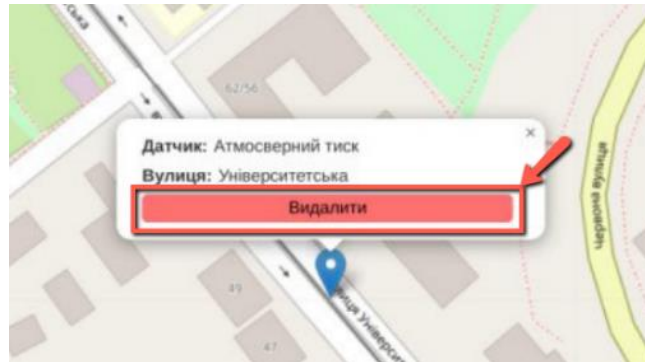


Рисунок 4.18 – Видалення точки з мапи (українською)

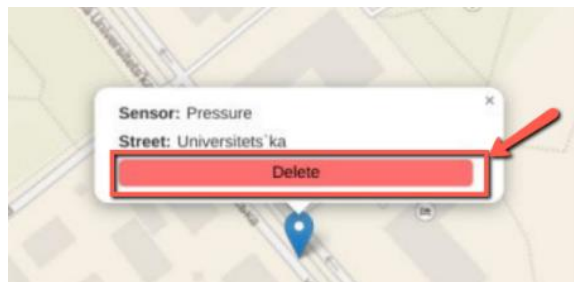


Рисунок 4.19 – Видалення точки з мапи (англійською)

Результат видалення точки зображено на рисунку 4.20.



Рисунок 4.20 – Видалена точка

4.2 Інструкція для розробника

Цей розділ призначений для розробників, які здійснюватимуть обслуговування, модифікацію або розширення вебпорталу екологічного моніторингу. Документ описує структуру проєкту, порядок розгортання, підключення до API зовнішніх метеосервісів, а також інструкції щодо доповнення та налаштування системи.

Структура репозиторію:

- /client: фронтенд, реалізований на React [18];
- /server: бекенд Node.js [19] + Express [20];
- /docker-compose-dev.yaml: конфігурація для контейнеризації за допомогою Docker Compose [30].

Порядок розгортання:

а) клонувати репозиторій: `git clone https://github.com/AndrKirienko/diploma.git;`

б) перейти в директорію проєкту: `cd diploma;`

в) виконати `docker compose -f docker-compose-dev.yaml up` для запуску всіх сервісів;

г) налаштувати токени доступу до метеосервісів;

1) для HomeAssistant:

– після запуску контейнерів перейдіть за адресою `http://localhost:8123;`

– авторизуйтеся → профіль → безпека → творити токен;

– збережіть токен та пропишіть його у `/server/src/constants.js` у змінну `TOKEN`;

2) для OpenWeatherMap:

– увійдіть на `https://home.openweathermap.org;`

– перейдіть у меню «API Keys», згенеруйте ключ;

– пропишіть його у файл `/server/src/constants.js` у змінну `OPENWEATHER_API_KEY`.

Перевірити доступність:

- вебінтерфейс: `http://localhost:5000`;
- сервер API: `http://localhost:3000`;
- чат-бот API: `http://localhost:8000/api`.

В таблиці 4.1 відображені основні REST API ендпоінти.

Таблиця 4.1 – Основні ендпоінти

Метод	Роут	Опис
GET	<code>/api/localMeteorological</code>	Отримання останніх локальних метеоданих
GET	<code>/api/localMeteorological/period/:period</code>	Дані за період (7-latest, 7-days, 30-days)
GET	<code>/api/openWeatherMap</code>	Отримання поточних даних із OpenWeatherMap
GET	<code>/api/openWeatherMap/period/:period</code>	Дані за період (30-days)
GET	<code>/api/point</code>	Отримання точок на мапі
POST	<code>/api/point</code>	Створення нової точки
DELETE	<code>/api/point/:id</code>	Видалення точки за ID

Для забезпечення підтримки масштабованості та гнучкого розширення, система розроблена за принципами чистої архітектури та використовує шаблон MVC.

Додавання нового ендпоінта API:

- додайте новий файл у `/server/src/ontrollers/`;
- опишіть логіку функції;
- зареєструйте маршрут у відповідному `router.js`;
- за потреби створіть відповідну модель у `/server/src/models/`.

Розширення бази даних:

- додайте нові поля у відповідний `schema`-файл.

Оновлення або зміна структури графіка:

- редагуйте компонент Graphs у `/client/src/components/Graphs`;
- змініть конфігурацію відображення в `Chart.js`;
- перевірте відповідність формату даних.

Додавання нової сторінки:

- створіть компонент у `/client/src/pages`;
- підключіть до React Router в `App.js`;
- за потреби реалізуйте логіку підвантаження даних через `axios`.

Налаштування середовища

- конфігураційні змінні зберігаються у `/server/src/constants.js`;
- усі підключення до API винесені в окремі функції.

Рекомендації з обслуговування та оновлення:

- оновлення бібліотек: регулярно перевіряйте наявність оновлень у `package.json`;
- безпека: не зберігайте токени в репозиторії; використовуйте `.env` для продакшену;
- журналювання: додавайте логування у `console.log`.

4.3 Тестування компонентів

Тестування проводилось з метою перевірки працездатності основних модулів вебпорталу, коректності обміну даними між компонентами та зручності використання інтерфейсу. Було реалізовано наступні типи тестування:

- функціональне тестування: перевірка відповідності роботи модулів заявленим функціональностям;
- UX-тестування: перевірка взаємодії користувача з інтерфейсом, адаптивності сторінок та зручності навігації.

Тестування охоплювало REST API та основні компоненти фронтенду, що взаємодіють з базою даних і візуалізують інформацію. Результати тестів подано у таблиці 4.2.

Таблиця 4.2 – Функціональне тестування основних сценаріїв

Сценарій	Вхідні умови	Очікуваний результат	Статус
Завантаження головної сторінки	Відкрити <code>http://localhost:5000</code>	Сторінка завантажується, показано всі блоки	+
Отримання поточних даних з Home Assistant	<code>GET /api/localMeteorological</code>	Повертається актуальний запис JSON з даними сенсорів	+
Отримання даних з Open Weather Map	<code>GET /api/openWeatherMap</code>	Повертається погодна інформація із зовнішнього API	+
Додавання точки на мапу	<code>POST /api/point</code> з валідними координатами	Точка успішно створюється, мапа оновлюється	+
Видалення точки з мапи	<code>DELETE /api/point/:id</code>	Вказана точка видаляється із БД і більше не відображається	+
Отримання даних за період	<code>GET /api/openWeatherMap/period/:period</code> та <code>GET /api/localMeteorological/period/:period</code>	Повертається запис JSON за певний період відповідно	+
Некоректний запит до API	<code>GET /api/point/undefined</code>	Повертається 404 або 500 з повідомленням про помилку	+

Тестування користувацького досвіду виконувалось вручну на різних пристроях з використанням браузерів Google Chrome [39], Mozilla Firefox [40], Safari [41] та мобільного емулятора DevTools (табл. 4.3).

Таблиця 4.3 – UX-тестування

Перевірка	Умова	Очікуваний результат	Статус
Відображення інтерфейсу на мобільних пристроях	Ширина екрану < 768 px	Всі блоки адаптуються, елементи не виходять за межі екрана	+
Відкриття карти та взаємодія з маркерами	Додавання, перегляд, видалення точок	Додаються, видаляються та відображаються точки на мапі	+
Перемикання мови інтерфейсу	Обрати англійську / українську	Зміна мови відображення без перезавантаження сторінки	+

4.4 Висновки до розділу 4

У четвертому розділі було здійснено комплексну підготовку системи до експлуатації, проведено тестування функціональних та інтерфейсних можливостей, а також створено супровідну документацію, необхідну для подальшого використання та обслуговування вебпорталу.

У підпункті 4.1 надано інструкцію для кінцевого користувача, яка містить опис основних функцій системи, включаючи навігацію сторінкою, перегляд графіків, взаємодію з мапою та базову інтерпретацію даних з екологічних сенсорів. Це дозволяє користувачам ефективно використовувати інтерфейс без попередньої технічної підготовки.

У підпункті 4.2 представлено інструкцію для розробника, де розглянуто архітектурну структуру репозиторію, порядок розгортання системи за допомогою Docker [30], налаштування ключів доступу до API метеосервісів та структуру REST API-ендпоінтів. Додатково надано рекомендації щодо масштабування системи, оновлення її компонентів, а також можливості розширення функціоналу, що є важливим для забезпечення життєвого циклу програмного продукту в умовах динамічної зміни вимог.

У підпункті 4.3 детально описано результати тестування основних компонентів системи. Проведене функціональне тестування підтвердило коректність обробки запитів до бази даних, відображення графіків, мапи та взаємодії з екосенсорами. UX-тестування довело адаптивність інтерфейсу до різних пристроїв та зручність навігації. Усі критичні сценарії тестування завершилися успішно, що підтверджено таблицями тест-кейсів.

Таким чином, за результатами розділу можна зробити висновок, що розроблена система є повністю готовою до використання. Вона відповідає функціональним вимогам, має інтуїтивно зрозумілий інтерфейс, задокументовану архітектуру та продемонструвала стабільну роботу в процесі тестування. Наявність інструкцій для користувачів і розробників спрощує подальше впровадження системи у навчальне або дослідницьке середовище.

ВИСНОВКИ

У роботі вирішено актуальну проблему цифровізації екологічного моніторингу в межах університетської інфраструктури.

Створено вебпортал, який забезпечує зручний доступ до екологічних даних у реальному часі, сприяє підвищенню прозорості екополітики навчального закладу та формуванню екосвідомості серед студентів і викладачів. Розроблена система дозволяє здійснювати порівняльний аналіз локальних показників з даними метеосервісів, візуалізувати історію змін у вигляді графіків, карт та таблиць.

У межах дослідження проаналізовано існуючі підходи до екологічного моніторингу, визначено роль інформаційних технологій у підтримці сталого розвитку та обґрунтовано вибір архітектури на основі шаблону MVC. Реалізація програмного рішення здійснена за допомогою стеку технологій React.js, Node.js, MongoDB, що забезпечує масштабованість, адаптивність та інтеграцію з зовнішніми API. Проведено тестування ключових компонентів системи, а також підготовлено інструкції для кінцевих користувачів і розробників з описом процесу розгортання та роботи з інтерфейсом.

Під час підготовки роботи використовувалися можливості штучного інтелекту для пошуку наукових джерел, редагування та граматичної перевірки текстів, переформулювання фрагментів у науковому стилі, а також для оптимізації коду та вибору відповідних бібліотек у практичній реалізації. Застосування ШІ дозволило значно покращити якість оформлення та реалізації проєкту.

Результати роботи було представлено на Щорічній науково-практичній конференції викладачів, науковців, молодих учених, аспірантів та студентів Національного університету «Запорізька політехніка» – «Тиждень науки-2025».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Стратегія сталого розвитку України до 2030 року [Електронний ресурс]. – Режим доступу : <https://www.undp.org/ukraine/publications/sustainable-development-strategy-ukraine-2030>.
2. European University Association «Green transition» [Electronic resource]. – Access mode : <https://www.eua.eu/our-work/topics/green-transition.html>.
3. Galán-Moros E. The role of universities in advancing the Sustainable Development Goals [Electronic resource]. – Access mode : <https://www.uiin.org/2024/06/25/the-role-of-universities-in-advancing-the-sustainable-development-goals/>.
4. ЗУНУ (ЕкоБот) [Електронний ресурс]. – Режим доступу : <https://www.wunu.edu.ua/ekobot/>.
5. КРОК (ЕКО «ЛОТОС») [Електронний ресурс]. – Режим доступу : <https://www.krok.edu.ua/ua/studentam/eko-lotos>.
6. George Mason University [Electronic resource]. – Access mode : <https://green.gmu.edu/>.
7. University of Pennsylvania [Electronic resource]. – Access mode : <https://www.sustainability.upenn.edu/>.
8. Environment Monitoring System Using IoT [Electronic resource]. – Access mode : https://www.technoduces.com/case-study/iot-based-environmental-monitoring?utm_source=chatgpt.com.
9. A Survey of GIS and IoT Integration: Applications and Architecture [Electronic resource]. – Access mode : https://www.mdpi.com/2076-3417/11/21/10365?utm_source=chatgpt.com#B118-applsci-11-10365.
10. AI Water Quality Monitors Deployed Across Southern England [Electronic resource]. – Access mode : https://kurrant.com/kurrantly-news/ai-water-quality-monitors-deployed-across-southern-england/?utm_source=chatgpt.com.

11. Великі дані [Електронний ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/Великі_дані.
12. Günther O. / Environmental Monitoring: Models, Methods, And Systems / O. Günther // Environmental Monitoring. – July 1999. – P. 2-24. (https://www.researchgate.net/publication/2819151_Environmental_Monitoring_Models_Methods_And_Systems)
13. Model – View – Controller [Electronic resource]. – Access mode : <https://en.wikipedia.org/wiki/Model-view-controller>.
14. MVP [Electronic resource]. – Access mode : https://en.wikipedia.org/wiki/Minimum_viable_product.
15. MVVM [Electronic resource]. – Access mode : <https://en.wikipedia.org/wiki/Model-view-viewmodel>.
16. Microservices [Electronic resource]. – Access mode : <https://en.wikipedia.org/wiki/Microservices>.
17. MongoDB [Electronic resource]. – Access mode : <https://www.mongodb.com>.
18. React [Electronic resource]. – Access mode: <https://react.dev>.
19. Node.js [Electronic resource]. – Access mode: <https://nodejs.org/en>.
20. Express [Electronic resource]. – Access mode: <https://expressjs.com>.
21. Chart.js [Electronic resource]. – Access mode: <https://www.chartjs.org>.
22. Vue.js [Electronic resource]. – Access mode: <https://vuejs.org>.
23. Angular [Electronic resource]. – Access mode: <https://angular.dev>.
24. Django [Electronic resource]. – Access mode : <https://www.djangoproject.com>.
25. Laravel [Electronic resource]. – Access mode: <https://laravel.com>.
26. PostgreSQL [Electronic resource]. – Access mode : <https://www.postgresql.org>.
27. MySQL [Electronic resource]. – Access mode : <https://www.mysql.com>.
28. D3.js [Electronic resource]. – Access mode: <https://d3js.org>.

29. Google Charts [Electronic resource]. – Access mode :
<https://developers.google.com/chart>.
30. Docker [Electronic resource]. – Access mode: <https://docs.docker.com>.
31. Vagrant [Electronic resource]. – Access mode :
<https://developer.hashicorp.com/vagrant>.
32. Kubernetes [Electronic resource]. – Access mode :
<https://kubernetes.io>.
33. Git [Electronic resource]. – Access mode: <https://git-scm.com>.
34. Subversion [Electronic resource]. – Access mode :
<https://www.visualsvn.com>.
35. Mercurial [Electronic resource]. – Access mode :
<https://www.mercurial-scm.org>.
36. Axios [Electronic resource]. – Access mode : <https://axios-http.com/docs/intro>.
37. OpenWeatherMap [Electronic resource]. – Access mode :
<https://openweathermap.org/api>.
38. Leaflet [Electronic resource]. – Access mode: <https://leafletjs.com>.
39. Google Chrome [Electronic resource]. – Access mode :
https://www.google.com/intl/en_uk/chrome/.
40. Mozilla Firefox [Electronic resource]. – Access mode :
<https://www.mozilla.org/en-US/firefox/new/>.
41. Safari [Electronic resource]. – Access mode :
<https://www.apple.com/ua/safari/>

ДОДАТОК А
Технічне завдання

Вступ

Метою роботи є розробка вебпорталу, що забезпечить збирання, збереження, візуалізацію та порівняльний аналіз локальних та глобальних метеорологічних даних для підтримки зеленої стратегії розвитку університету.

На поточному етапі університет не має централізованої цифрової платформи для відображення та аналізу екологічних показників. Дані, які потенційно можуть збиратися з сенсорів, не акумулюються, не зберігаються в єдиній базі та не візуалізуються для користувачів. Відсутній облік, аналіз або порівняння з офіційними метеорологічними станціями. Внутрішні системи не інтегровані з екологічними підсистемами.

Після реалізації проєкту буде створений вебпортал, що забезпечить відображення даних із сенсорів у реальному часі, збереження у базі даних, побудову графіків і таблиць, інтерацію з API метеостанцій, управління точками на мапі, а також адаптивний інтерфейс для всіх типів пристроїв, використання багатомовності. Система стане частиною «зеленої стратегії» ЗВО та зможе інтегруватися з головним сайтом університету або освітніми ресурсами

A.1 Підстави для розробки

Підставою для розробки є Наказ № 209 від 28 квітня 2025 року по Національному університету «Запорізька політехніка».

A.2 Основні вимоги до програми

A.2.1 Вимоги до інтеграції

Інтеграція майбутнього вебпорталу в екосистему університету передбачає реалізацію стандартних механізмів обміну даними між

клієнтською частиною, сервером та зовнішніми сервісами. Визначаються такі загальні вимоги до обміну:

- формат обміну: JSON;
- протоколи: HTTPS, REST API;
- частота оновлення: не рідше одного разу на 5 хвилин (настроюване);
- обсяг: дані сенсорів (температура, вологість, швидкість вітру, радіація тощо), довідкова інформація, графіки;
- інтерфейс: взаємодія з зовнішніми сервісами (наприклад, метеостанції) та можливість вивантаження інформації у відкритих форматах.

A.2.2 Функціональні вимоги

Система має забезпечувати такі функціональні можливості:

- перегляд екологічних даних у графічному та табличному вигляді;
- візуалізація точок на мапі;
- додавання або видалення точок на мапі;
- порівняння з даними метеостанцій;
- інтеграція з чатботом.

A.2.3 Технічні вимоги

У технічному плані система повинна відповідати загальним критеріям надійності, масштабованості та сумісності з уже існуючими цифровими системами ЗВО:

- можливість розгортання на університетських серверах або VPS;
- використання стандартного інтерфейсу обміну даними з сенсорами та метеосервісами;
- забезпечення адаптивності для десктопів і мобільних пристроїв;

- використання відкритих API для з'єднання із зовнішніми джерелами даних;
- висока продуктивність системи при роботі з великим обсягом даних.

A.3 Етапи виконання робіт

Реалізація проєкту передбачає проходження кількох етапів, які охоплюють повний цикл розробки від аналізу до впровадження:

- аналіз вимог та постановка задачі;
- проєктування архітектури та інформаційної моделі;
- реалізація основної функціональності (інтерфейс, база даних, інтеграція);
- тестування системи в умовах моделювання даних;
- документування результатів;
- навчання користувачів (адміністраторів);
- впровадження на платформі ЗВО.

A.4 Тестування

A.4.1 План тестування

Для перевірки працездатності системи заплановане проведення таких типів тестування:

- функціональне тестування ключових модулів;
- інтеграційне тестування між компонентами (база – інтерфейс – API);
- навантажувальне тестування при одночасному запиті великої кількості даних;
- тестування доступу користувачів з різними правами;

- кросбраузерне тестування для забезпечення сумісності з популярними браузерами.

A.4.2 Критерії приймання

Проект вважається успішно реалізованим за умови досягнення наступних технічних та функціональних результатів:

- дані з сенсорів обробляються і візуалізуються без затримки;
- графіки відображаються динамічно та коректно;
- система витримує навантаження не менше 10 одночасних запитів;
- дані з метеосервісів успішно інтегруються та порівнюються.

A.5 Необхідна документація

У межах завершеного проекту формується супровідна документація для користувачів, розробників і контролю якості:

- керівництво користувача: опис інтерфейсу, функціональності, пошуку, роботи з мапою та графіками;
- керівництво програміста: структура модулів, API, структура даних, приклади запитів і відповідей, логіка обробки даних;
- тестові звіти: результати тестування системи згідно з критеріями приймання.

A.6 Безпека

A.6.1 Захист даних

Система повинна відповідати сучасним стандартам безпеки:

- HTTPS-шифрування даних при передачі;
- використання JWT-токенів для авторизації;
- резервне копіювання даних щодня.

A.6.2 Конфіденційність

Система повинна забезпечувати дотримання базових принципів захисту персональних даних і конфіденційності.

Дані з сенсорів публікуються у знеособленому форматі.

ДОДАТОК Б
Текст програми

Б.1 Реалізація компонента з відображенням графіків

```

<div className={styles.graphsContainer}>
  {parametersToCompare.map(param => (
    <LineChart
      key={param}
      title={t(param)}
      localData={comparison[param].local}
      openData={comparison[param].open}
      labels={labels}
      t={t}
    />
  ))}
</div>

```

Б.2 Реалізація компонента відображення останніх сім днів

```

<div className={styles.lastSevenDaysContainer}>
  <h2
    className={styles.lastSevenDaysTitle}>{t('lastSevenDays')}</h2>
  <div className={styles.lastSevenDaysItemContainer}>
    {localMeteorologicalDataByPeriod?.data?.length ? (
      localMeteorologicalDataByPeriod.data.map((entries, index) =>
        (
          <div className={styles.entriesContainer} key={index}>
            <span className={styles.timeContainer}>
              <span>
                {entries._id.day}.{entries._id.month}
              </span>
            </span>
            <p className={styles.weatherItemContainer}>
              <CiTempHigh />
              <span className={styles.weatherItemData}>
                <span>{entries.temperature}</span>
                <span>{t('temperatureUnit')}</span>
              </span>
            </p>
            <p className={styles.weatherItemContainer}>
              <LuWind />
              <span className={styles.weatherItemData}>
                <span>{entries.windSpeed}</span>

```

```

        <span>{t('windSpeedUnit')}</span>
    </span>
</p>
<p className={styles.weatherItemContainer}>
    <GiWindsock />
    <span className={styles.weatherItemData}>
        <span>{entries.windBearing}</span>
        <span>{t('windBearingUnit')}</span>
    </span>
</p>
<p className={styles.weatherItemContainer}>
    <IoIosCloudOutline />
    <span className={styles.weatherItemData}>
        <span>{entries.cloudCoverage}</span>
        <span>{t('cloudCoverageUnit')}</span>
    </span>
</p>
<p className={styles.weatherItemContainer}>
    <PiSolarPanelThin />
    <span className={styles.weatherItemData}>
        <span>{ (entries.solarRadiation
25).toFixed(2) }</span>
        <span>{t('solarRadiationUnit')}</span>
    </span>
</p>
<p className={styles.weatherItemContainer}>
    <MdDewPoint />
    <span className={styles.weatherItemData}>
        <span>{entries.dewPoint}</span>
        <span>{t('dewPointUnit')}</span>
    </span>
</p>
<p className={styles.weatherItemContainer}>
    <WiBarometer />
    <span className={styles.weatherItemData}>
        <span>{entries.pressure}</span>
        <span>{t('pressureUnit')}</span>
    </span>
</p>
</div>
))
) : (
    <p className={styles.loading} styles>

```

*

```

        {t('loading')}
      </p>
    )}
  </div>
</div>

```

Б.3 Реалізація компонента відображення останніх сім записів

```

<div className={styles.lastSevenEntriesContainer}>
  <h2
    className={styles.lastSevenEntriesTitle}>{t('lastSevenEntries')}</h2>
  <div className={styles.lastSevenEntriesItemContainer}>
    {localMeteorologicalDataByPeriod?.data?.length ? (
      localMeteorologicalDataByPeriod.data.map((entries, index) =>
        (
          <div className={styles.entriesContainer} key={index}>
            <p className={styles.weatherStateContainer}>
              <StateWeather state={entries.state} isTitle={false} />
            </p>
            <span className={styles.timeContainer}>
              <span>{format(new Date(entries.lastUpdated),
                'HH:mm')}</span>
            </span>
            <p className={styles.weatherItemContainer}>
              <CiTempHigh />
              <span className={styles.weatherItemData}>
                <span>{entries.temperature}</span>
                <span>{t('temperatureUnit')}</span>
              </span>
            </p>
            <p className={styles.weatherItemContainer}>
              <LuWind />
              <span className={styles.weatherItemData}>
                <span>{entries.windSpeed}</span>
                <span>{t('windSpeedUnit')}</span>
              </span>
            </p>
            <p className={styles.weatherItemContainer}>
              <GiWindsock />
              <span className={styles.weatherItemData}>
                <span>{entries.windBearing}</span>

```

```

        <span>{t('windBearingUnit')}</span>
    </span>
</p>
<p className={styles.weatherItemContainer}>
    <IoIosCloudOutline />
    <span className={styles.weatherItemData}>
        <span>{entries.cloudCoverage}</span>
        <span>{t('cloudCoverageUnit')}</span>
    </span>
</p>
<p className={styles.weatherItemContainer}>
    <PiSolarPanelThin />
    <span className={styles.weatherItemData}>
        <span>{ (entries.solarRadiation
25).toFixed(2) }</span>
        <span>{t('solarRadiationUnit')}</span>
    </span>
</p>
<p className={styles.weatherItemContainer}>
    <MdDewPoint />
    <span className={styles.weatherItemData}>
        <span>{entries.dewPoint}</span>
        <span>{t('dewPointUnit')}</span>
    </span>
</p>
<p className={styles.weatherItemContainer}>
    <WiBarometer />
    <span className={styles.weatherItemData}>
        <span>{entries.pressure}</span>
        <span>{t('pressureUnit')}</span>
    </span>
</p>
</div>
))
) : (
    <p className={styles.loading} styles>
        {t('loading')}
    </p>
)
</div>
</div>

```

*

Б.4 Реалізація компонента по відображенню локальних метеорологічних даних в реальному часі

```

<section className={styles.weatherContainer}>
  <h2 className={styles.weatherTitle}>
    {t('localTitle')}
  <p className={styles.weatherState}>
    <StateWeather state={state} isTitle={true} />
  </p>
</h2>
<div className={styles.weatherItemContainer}>
  <span className={styles.weatherItemTitle}>{t('temperature')}:
</span>

  <span className={styles.weatherItemData}>
    <span>{temperature}</span>
    <span>{t('temperatureUnit')}</span>
  </span>
</div>
<div className={styles.weatherItemContainer}>
  <span className={styles.weatherItemTitle}>{t('windSpeed')}:
</span>

  <span className={styles.weatherItemData}>
    <span>{wind_speed}</span>
    <span>{t('windSpeedUnit')}</span>
  </span>
</div>
<div className={styles.weatherItemContainer}>
  <span className={styles.weatherItemTitle}>{t('windBearing')}:
</span>

  <span className={styles.weatherItemData}>
    <span>{wind_bearing}</span>
    <span>{t('windBearingUnit')}</span>
  </span>
</div>
<div className={styles.weatherItemContainer}>
  <span className={styles.weatherItemTitle}>{t('cloudCoverage')}:
</span>

  <span className={styles.weatherItemData}>
    <span>{cloud_coverage}</span>
    <span>{t('cloudCoverageUnit')}</span>
  </span>
</div>

```

```

    </div>
    <div className={styles.weatherItemContainer}>
      <span
        className={styles.weatherItemTitle}>{t('solarRadiation')}: </span>
      <span className={styles.weatherItemData}>
        <span>{(uv_index * 25).toFixed(2)} </span>
        <span>{t('solarRadiationUnit')} </span>
      </span>
    </div>
    <div className={styles.weatherItemContainer}>
      <span      className={styles.weatherItemTitle}>{t('dewPoint')}:
</span>

      <span className={styles.weatherItemData}>
        <span>{dew_point} </span>
        <span>{t('dewPointUnit')} </span>
      </span>
    </div>
    <div className={styles.weatherItemContainer}>
      <span      className={styles.weatherItemTitle}>{t('pressure')}:
</span>

      <span className={styles.weatherItemData}>
        <span>{pressure} </span>
        <span>{t('pressureUnit')} </span>
      </span>
    </div>
    <div className={styles.weatherItemContainer}>
      <span      className={styles.weatherItemTitle}>{t('lastUpdated')}:
</span>

      <span className={styles.weatherItemData}>
        {format(new Date(last_updated), 'dd.MM.yyyy HH:mm')}
      </span>
    </div>
  </section>

```

Б.5 Реалізація компонента по відображенню Open Weather Map метеорологічних даних в реальному часі

```

<section className={styles.weatherContainer}>
  <h2 className={styles.weatherTitle}>
    OpenWeatherMap{' '}
  <p>

```

```

        {name} {country}
    </p>
    <p className={styles.weatherState}>
        <OpenStateWeather state={state} />
    </p>
</h2>
<div className={styles.weatherItemContainer}>
    <span className={styles.weatherItemTitle}>{t('temperature')}:
</span>

    <span className={styles.weatherItemData}>
        <span>{temp}</span>
        <span>{t('temperatureUnit')}</span>
    </span>
</div>
<div className={styles.weatherItemContainer}>
    <span
        className={styles.weatherItemTitle}>
{t('tempFeelsLike')}: </span>
    <span className={styles.weatherItemData}>
        <span>{feels_like}</span>
        <span>{t('temperatureUnit')}</span>
    </span>
</div>
<div className={styles.weatherItemContainer}>
    <span
        className={styles.weatherItemTitle}>{t('windSpeed')}:
</span>

    <span className={styles.weatherItemData}>
        <span>{(speed * 3.6).toFixed(2)}</span>
        <span>{t('windSpeedUnit')}</span>
    </span>
</div>
<div className={styles.weatherItemContainer}>
    <span
        className={styles.weatherItemTitle}>{t('windBearing')}:
</span>

    <span className={styles.weatherItemData}>
        <span>{deg}</span>
        <span>{t('windBearingUnit')}</span>
    </span>
</div>
<div className={styles.weatherItemContainer}>
    <span
        className={styles.weatherItemTitle}>{t('windGust')}:
</span>

    <span className={styles.weatherItemData}>
        <span>{(gust * 3.6).toFixed(2)}</span>

```

```

        <span>{t('windSpeedUnit')}</span>
    </span>
</div>
<div className={styles.weatherItemContainer}>
    <span      className={styles.weatherItemTitle}>{t('pressure')}</span>:
</span>

    <span className={styles.weatherItemData}>
        <span>{pressure}</span>
        <span>{t('pressureUnit')}</span>
    </span>
</div>
<div className={styles.weatherItemContainer}>
    <span  className={styles.weatherItemTitle}>{t('grndPressure')}</span>:
</span>

    <span className={styles.weatherItemData}>
        <span>{grnd_level}</span>
        <span>{t('pressureUnit')}</span>
    </span>
</div>
<div className={styles.weatherItemContainer}>
    <span  className={styles.weatherItemTitle}>{t('seaPressure')}</span>:
</span>

    <span className={styles.weatherItemData}>
        <span>{sea_level}</span>
        <span>{t('pressureUnit')}</span>
    </span>
</div>
<div className={styles.weatherItemContainer}>
    <span      className={styles.weatherItemTitle}>{t('humidity')}</span>:
</span>

    <span className={styles.weatherItemData}>
        <span>{humidity}</span>
        <span>{t('humidityUnit')}</span>
    </span>
</div>
<div className={styles.weatherItemContainer}>
    <span className={styles.weatherItemTitle}>{t('cloudCoverage')}</span>:
</span>

    <span className={styles.weatherItemData}>
        <span>{clouds}</span>
        <span>{t('cloudCoverageUnit')}</span>
    </span>
</div>

```

```

<div className={styles.weatherItemContainer}>
  <span    className={styles.weatherItemTitle}>{t('visibility')}:
</span>

  <span className={styles.weatherItemData}>
    <span>{visibility}</span>
    <span>{t('visibilityUnit')}</span>
  </span>
</div>
<div className={styles.weatherItemContainer}>
  <span    className={styles.weatherItemTitle}>{t('lastUpdated')}:
</span>

  <span
className={styles.weatherItemData}>{formatUnixTime(dt)}</span>
</div>
</section>

```

Б.6 Реалізація компонента по відображенню мапи з точками моніторингу екологічних даних

```

<MapContainer
  center={DEFAULT_CENTER_MAP}
  zoom={DEFAULT_ZOOM_MAP}
  scrollWheelZoom={false}
  style={{ height: '500px' }}
>
  <TileLayer url={mapUrl()} />

  <LocationHandler createPoint={createPoint} />
  <Points deletePoint={deletePoint} points={points} />
</MapContainer>

```

Б.7 Реалізація компонента, який виконує функцію центрального хабу

```

<div className={styles.container}>
  <div className={styles.changeLanguageContainer}>
    <div className={styles.changeLanguage}>
      <IoLanguageOutline />
      <select

```

```

        className={styles.changeLanguageSelect}
        name='language'
        value={i18n.language}
        onChange={e => i18n.changeLanguage(e.target.value)}
      >
        <option value='uk'>Українська</option>
        <option value='en'>English</option>
      </select>
    </div>
  </div>
  <div className={styles.weatherContainer}>
    <LocalMeteorological />
    <OpenWeatherMap />
  </div>
  <LastSevenEntries />
  <LastSevenDays />
  <div>
    <Graphs />
  </div>
  <div>
    <Maps />
  </div>
</div>

```

Б.8 Метод отримання локальних метеорологічних даних

```

module.exports.getLocalMeteorologicalData = async (req, res, next) => {
  try {
    const response = await
    fetch('http://localMeteorological:8123/api/states', {
      headers: {
        Authorization: `Bearer ${TOKEN}`,
        Accept: 'application/json',
      },
    });
    const data = await response.json();

    const forecast = data.find(
      item => item.entity_id === 'weather.forecast_home_assistant'
    );
  }

```

```

    if (!forecast) {
        return res.status(404).json({ error: 'Forecast data not found' });
    }

    const { state = null, last_updated = null, attributes = {} } =
forecast;

    const {
        temperature = null,
        wind_speed = null,
        wind_bearing = null,
        cloud_coverage = null,
        uv_index = null,
        dew_point = null,
        pressure = null,
    } = attributes;

    res.send({
        data: {
            state,
            last_updated,
            attributes: {
                temperature,
                wind_speed,
                wind_bearing,
                cloud_coverage,
                uv_index,
                dew_point,
                pressure,
            },
        },
    });
} catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Local Meteorological request error'
});
}
};

```

Б.9 Метод додавання локальних метеорологічних даних до локальної бази даних

```

module.exports.addLocalMeteorologicalDataToBD = async (req, res, next)
=> {
    try {
        const response = await
fetch('http://localMeteorological:8123/api/states', {
    headers: {
        Authorization: `Bearer ${TOKEN}`,
        Accept: 'application/json',
    },
});
const data = await response.json();

const forecast = data.find(
    item => item.entity_id === 'weather.forecast_home_assistant'
);

if (!forecast) {
    return res.status(404).json({ error: 'Forecast data not found' });
}

const { state = null, last_updated = null, attributes = {} } =
forecast;

const {
    temperature = null,
    wind_speed = null,
    wind_bearing = null,
    cloud_coverage = null,
    uv_index = null,
    dew_point = null,
    pressure = null,
} = attributes;

const formattedDate = new Date(last_updated.replace(/\/\.\d{6}/, ''));

const entry = new LocalMeteorological({
    state,
    temperature,
    windSpeed: wind_speed,

```

```

        windBearing: wind_bearing,
        cloudCoverage: cloud_coverage,
        solarRadiation: +(uv_index * 25).toFixed(2),
        dewPoint: dew_point,
        pressure,
        lastUpdated: formattedDate,
    });

    await entry.save();

    res.status(200).json({ message: 'Local data saved successfully' });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Local Meteorological request error'
  });
}
};

```

Б.10 Метод отримання локальних метеорологічних даних за певних період

```

module.exports.getLocalMeteorologicalDataByPeriod = async (req, res,
next) => {
  try {
    const { period } = req.params;

    let data;

    switch (period) {
      case SEVEN_LATEST:
        data = await LocalMeteorological.find()
          .sort({ lastUpdated: -1 })
          .limit(7);
        break;

      case SEVEN_DAYS:
      case THIRTY_DAYS: {
        const days = period === SEVEN_DAYS ? 7 : 30;
        const fromDate = new Date(Date.now() - days * 24 * 60 * 60 *
1000);

```

```

data = await LocalMeteorological.aggregate([
  { $match: { lastUpdated: { $gte: fromDate } } },
  {
    $group: {
      _id: {
        month: { $month: '$lastUpdated' },
        day: { $dayOfMonth: '$lastUpdated' },
      },
      temperature: { $avg: '$temperature' },
      windSpeed: { $avg: '$windSpeed' },
      windBearing: { $avg: '$windBearing' },
      cloudCoverage: { $avg: '$cloudCoverage' },
      solarRadiation: { $avg: '$solarRadiation' },
      dewPoint: { $avg: '$dewPoint' },
      pressure: { $avg: '$pressure' },
    },
  },
  { $sort: { '_id.month': -1, '_id.day': -1 } },
  { $limit: days },
  { $sort: { '_id.month': 1, '_id.day': 1 } },
]);
break;
}

default:
  return res.status(400).json({ error: 'Invalid period parameter'
});

}

res.status(200).send({ data: { data } });
} catch (err) {
  console.error('Error fetching meteorological data:', err);
  res.status(500).json({ error: 'Server error' });
}
};

```

Б.11 Метод отримання Open Weather Map метеорологічних даних

```

module.exports.getOpenWeatherMapData = async (req, res, next) => {
  try {
    const response = await fetch(

```

```

    `https://api.openweathermap.org/data/2.5/weather?q=Zaporizhzhia&units=
    metric&appid=${OPENWEATHER_API_KEY}`
    );
    const data = await response.json();

    if (response.ok) {
        res.send({ data: { data } });
    } else {
        res.status(response.status).json({ error: data.message });
    }
} catch (err) {
    res.status(500).json({ error: 'OpenWeather API request error' });
}
};

```

Б.12 Метод додавання Open Weather Map метеорологічних даних до локальної бази даних

```

module.exports.addOpenWeatherMapDataToDB = async (req, res, next) => {
    try {
        const response = await fetch(

`https://api.openweathermap.org/data/2.5/weather?q=Zaporizhzhia&units=metric&
appid=${OPENWEATHER_API_KEY}`
        );
        const data = await response.json();
        const {
            visibility,
            dt,
            clouds: { all: clouds },
            wind: { deg, gust, speed },
            weather: {
                0: { main: state },
            },
            main: { temp, feels_like, grnd_level, humidity, pressure,
sea_level },
        } = data;
        if (response.ok) {
            const entry = new OpenWeatherMap({
                state,
                temperature: temp,

```

```

        temperatureFeelsLike: feels_like,
        windSpeed: speed,
        windBearing: deg,
        windGust: gust,
        pressure,
        pressureAtGroundLevel: grnd_level,
        pressureAtSeaLevel: sea_level,
        humidity,
        cloudCoverage: clouds,
        visibility,
        lastUpdated: new Date(dt * 1000),
    });

    await entry.save();

    res
        .status(200)
        .json({ message: 'OpenWeatherMap data saved successfully' });
    } else {
        res.status(response.status).json({ error: data.message });
    }
} catch (err) {
    res
        .status(500)
        .json({ error: 'OpenWeather API request error', details:
err.message });
    }
};

```

Б.13 Метод отримання Open Weather Map метеорологічних даних за певних період

```

module.exports.getOpenWeatherMapDataByPeriod = async (req, res, next)
=> {
    try {
        const { period } = req.params;

        let data;

        switch (period) {
            case SEVEN_LATEST:

```

```

        data = await OpenWeatherMap.find().sort({ lastUpdated: -1
    }).limit(7);

    break;

case SEVEN_DAYS:
case THIRTY_DAYS: {
    const days = period === SEVEN_DAYS ? 7 : 30;
    const fromDate = new Date(Date.now() - days * 24 * 60 * 60 *
1000);

    data = await OpenWeatherMap.aggregate([
        { $match: { lastUpdated: { $gte: fromDate } } },
        {
            $group: {
                _id: {
                    month: { $month: '$lastUpdated' },
                    day: { $dayOfMonth: '$lastUpdated' },
                },
                temperature: { $avg: '$temperature' },
                windSpeed: { $avg: '$windSpeed' },
                windBearing: { $avg: '$windBearing' },
                cloudCoverage: { $avg: '$cloudCoverage' },
                pressure: { $avg: '$pressure' },
                pressureAtGroundLevel: { $avg: '$pressureAtGroundLevel'
},

                pressureAtSeaLevel: { $avg: '$pressureAtSeaLevel' },
                humidity: { $avg: '$humidity' },
                visibility: { $avg: '$visibility' },
            },
        },
        { $sort: { '_id.month': -1, '_id.day': -1 } },
        { $limit: days },
        { $sort: { '_id.month': 1, '_id.day': 1 } },
    ]);
    break;
}

default:
    return res.status(400).json({ error: 'Invalid period parameter'
});
}

res.status(200).send({ data: data });

```

```

    } catch (err) {
      console.error('Error fetching OpenWeatherMap data:', err);
      res.status(500).json({ error: 'Server error' });
    }
  };
};

```

Б.14 Метод отримання списку усіх точок у локальну базу даних моніторингу

```

module.exports.getPoints = async (req, res, next) => {
  try {
    const points = await Point.find();
    res.status(200).send({ data: points });
  } catch (error) {
    next(error);
  }
};

```

Б.15 Метод додавання точки моніторингу у локальну базу даних

```

module.exports.createPoint = async (req, res, next) => {
  try {
    const createPoint = await Point.create(req.body);
    res.status(201).send({ data: createPoint });
  } catch (error) {
    next(error);
  }
};

```

Б.16 Метод видалення точки моніторингу з локальної бази даних

```

module.exports.deletePoint = async (req, res, next) => {
  try {
    const deletePoint = await Point.findByIdAndDelete(req.params.id);

    if (!deletePoint) {
      return res.status(404).send({ message: 'Point not found' });
    }
  }
};

```

```

    res.status(200).send({ data: deletePoint });
  } catch (error) {
    next(error);
  }
};

```

Б.17 Схема Mongo-колекції для локальних метеорологічних даних

```

const Schema = new mongoose.Schema({
  state: String,
  temperature: Number,
  windSpeed: Number,
  windBearing: Number,
  cloudCoverage: Number,
  solarRadiation: Number,
  dewPoint: Number,
  pressure: Number,
  lastUpdated: Date,
});

```

Б.18 Схема Mongo-колекції для Open Weather Map метеорологічних даних

```

const Schema = new mongoose.Schema({
  state: String,
  temperature: Number,
  temperatureFeelsLike: String,
  windSpeed: Number,
  windBearing: Number,
  windGust: Number,
  pressure: Number,
  pressureAtGroundLevel: Number,
  pressureAtSeaLevel: Number,
  humidity: Number,
  cloudCoverage: Number,
  visibility: Number,
  lastUpdated: Date,
});

```

Б.19 Схема Mongo-колекції для метеорологічних точок

```
const Schema = new mongoose.Schema({  
  position: {  
    lat: Number,  
    lng: Number,  
  },  
  sensor: String,  
  street: String,  
});
```

ДОДАТОК В
Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»
Кафедра програмних засобів

Дипломна кваліфікаційна робота на тему:
Розробка вебпорталу для підтримки зеленої стратегії розвитку
університету

Виконав
студент групи КНТ-221

Андрій КІРІЄНКО

Керівник
к.т.н., доцент

Анжеліка ПАРХОМЕНКО

2025

Рисунок В.1 – Слайд 1 «Титульний слайд»

Актуальність роботи

Порівняльна таблиця «зелених» університетських ініціатив

Критерій	ЗУНУ (ЕкоБот)	КРОК (ЕКО «ЛОТОС»)	George Mason University	University of Pennsylvania
Наявність окремого розділу	Окремий портал «ЕкоБот»	Розділ «ЕКО ЛОТОС»	Сайт Green GMU	Sustainability at Penn
Доступ до даних у реальному часі	Графіки з сенсорів температури, вологості, якості води	Даних у реальному часі немає	Дані про компост, споживання води, електрики	Є доступ до даних про енергію, мобільність
Опис проєктів та ініціатив	Розписано реалізацію системи моніторингу	Проекти екологічного виховання	Patriot Green Fund, енергетичні ініціативи	Climate & Sustainability Action Plan
Можливості для студентів	Не зазначено	Участь у заходах ЕКО-клубу	Волонтерство, фінансування ідей	Стажування, дослідження, конкурси
Інтеграція у навчальні програми	Не зазначено	Немає підтвердження	Є тематичні курси та проєкти	Інтегровано в академічні програми
Візуальні або інтерактивні елементи	Дані з сенсорів	Немає інтерактиву	Інтерактивні мапа	Дашборди, онлайн-аналітика



Слайд 2

Рисунок В.2 – Слайд 2 «Актуальність роботи»

Мета та завдання роботи

Мета роботи – розробка вебпорталу, що забезпечить збирання, збереження, візуалізацію та порівняльний аналіз локальних та глобальних метеорологічних даних для підтримки зеленої стратегії розвитку університету.

Завдання:

- проаналізувати існуючі підходи у сфері екологічного моніторингу;
- обґрунтувати вибір технологій для реалізації вебпорталу;
- розробити структуру та функціональні модулі програмної системи;
- забезпечити базову інтеграцію з базами локальних та глобальних метеорологічних даних, а також з екологічним чатботом;
- підготувати рекомендації щодо впровадження вебпорталу.

Слайд 3

Рисунок В.3 – Слайд 3 «Мета та завдання роботи»

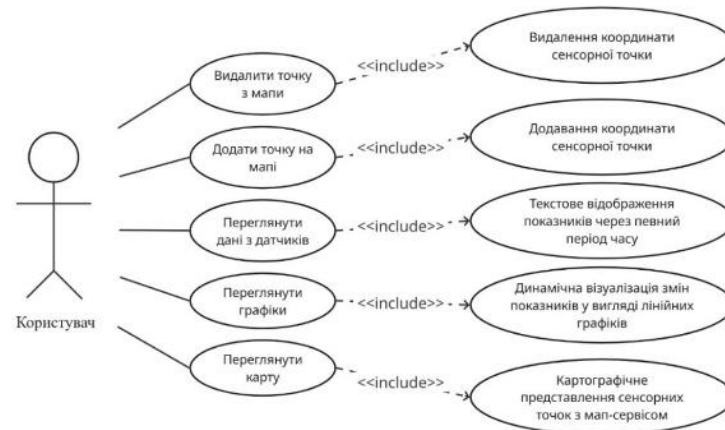
Порівняння використовуваних технологій у сфері екологічного моніторингу

Технологія	Призначення	Переваги	Приклади застосування
IoT	Збір даних із сенсорів у реальному часі	Автоматизація, оперативність висока точність	Моніторинг температури, вологості, якості повітря
GIS	Аналіз і візуалізація просторових даних	Просторове моделювання, побудова карт, локалізація подій	Картографування зон забруднення, аналіз локацій
AI	Обробка та аналіз великих обсягів даних, прогнозування	Автоматичне виявлення закономірностей, адаптивність	Прогноз погоди, виявлення аномалій у даних моніторингу
Big Data	Зберігання, обробка та аналіз масивів даних з різних джерел	Масштабованість, аналіз у реальному часі, гнучкість	Обробка даних з сенсорів, супутників, метеостанцій

Слайд 4

Рисунок В.4 – Слайд 4 «Порівняння використовуваних технологій екологічного моніторингу»

Проектування системи



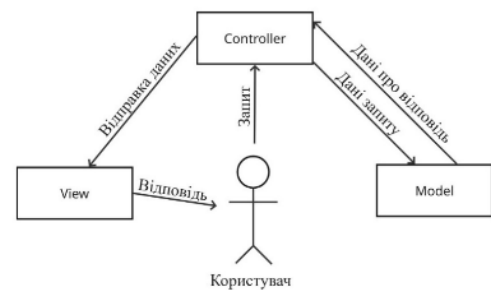
UML-діаграма прецедентів

Слайд 5

Рисунок В.5 – Слайд 5 «Проектування системи»

Вибір архітектурного шаблону

Критерій	MVC	MVP	MVVM	Microservices
Розділення обов'язків	Добре	Помірно	Добре	Чудово
Масштабованість	Висока	Середня	Висока	Дуже велика
Ускладнення реалізації	Помірна	Помірна	Складніша	Висока
Придатність для SPA	Ідеальна	Менш зручна	Ідеальна	Не основне застосування
Придатність для IoT/REST API	Підходить	Обмежено	Підходить	Чудово
Швидкість розгортання	Швидко	Швидко	Повільніше	Довго
Приклад застосування	Вебпортали	Десктоп-додатки	SPA, мобільні	Складні хмарні платформи



Слайд 6

Рисунок В.6 – Слайд 6 «Вибір архітектурного шаблону»

Архітектура IoT-системи екологічного моніторингу



Слайд 7

Рисунок В.7 – Слайд 7 «Архітектура IoT-системи екологічного моніторингу»

Порівняння та вибір технологій розробки

Фронтенд-технології			
Критерій	React.js	Vue.js	Angular
Продуктивність	Висока	Висока	Середня
Крива навчання	Помірна	Низька	Висока
Компонентна структура	Так	Так	Так
Розмір спільноти	Дуже велика	Велика	Велика
Гнучкість інтеграції	Висока	Висока	Складніша інтеграція
Підтримка екосистеми	Багата	Обмежена	Вбудовані рішення

Бекенд-технології			
Критерій	Node.js + Express	Django	Laravel
Асинхронність	Так	Ні	Обмежена
Продуктивність	Продуктивність	Середня	Середня
Гнучкість	Висока	Висока	Середня
Складність налаштування	Низька	Помірна	Середня
Спільнота	Велика	Середня	Низька
REST API з коробки	Так	Так	Так

Бібліотеки візуалізації			
Критерій	Chart.js	D3.js	Google Charts
Простота використання	Висока	Низька	Середня
Підтримка анімації	Так	Так	Так
Інтерактивність	Обмежена	Висока	Обмежена
Час роботи	Короткий	Довгий	Середній
Підтримка React	Так	Так	Обмежена

Засоби контейнеризації			
Критерій	Docker/Compose	Vagrant	Kubernetes
Простота налаштування	Висока	Середня	Низька
Швидкість розгортання	Висока	Середня	Висока
Придатність до локальної розробки	Так	Так	Ні

Бази даних			
Критерій	MongoDB	PostgreSQL	MySQL
Підтримка JSON	Так	Так	Часткова
Масштабованість	Висока	Середня	Середня
Гнучкість схеми	Дуже гнучка	Менш гнучка	Менш Гнучка
Продуктивність при IoT	Висока	Середня	Середня

Системи контролю версій			
Критерій	Git	Subversion	Mercurial
Тип системи	Розподілена	Централізована	Розподілена
Продуктивність	Висока	Середня	Висока
Робота офлайн	Так	Ні	Так
Гнучкість гнучкування	Висока	Обмежена	Середня
Популярність та підтримка	Найпопулярніша	Менша	Низька
Інтеграція з Github/GitLab	Повна	Часткова	Обмежена
Навчальна база/документація	Обширна	Обмежена	Обмежена

Слайд 8

Рисунок В.8 – Слайд 8 «Порівняння та вибір технологій розробки»

Структурна схема системи

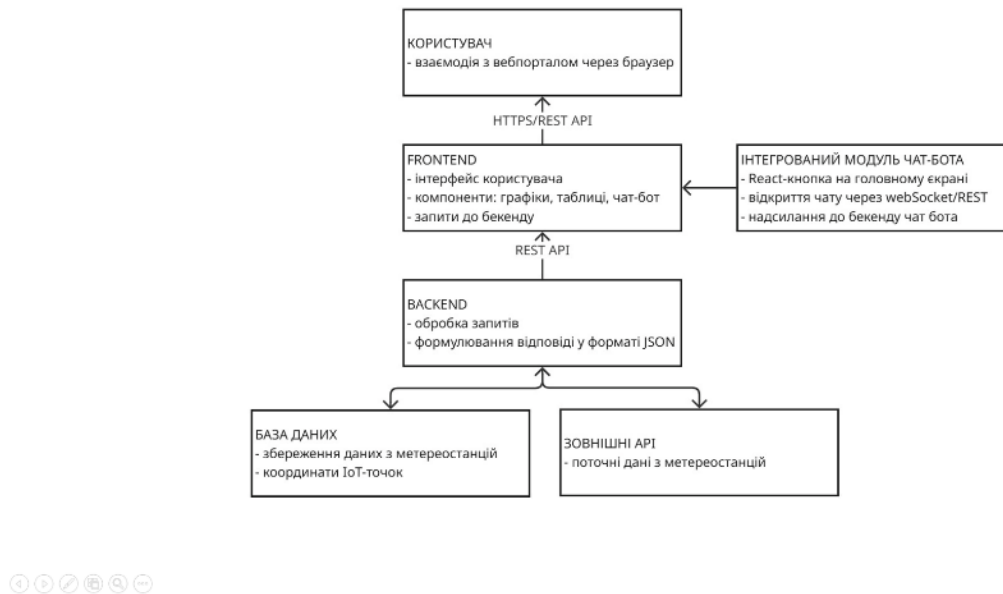


Рисунок В.9 – Слайд 9 «Структурна схема системи»

Основні рішення щодо програмної реалізації

Клієнтська частина		MongoDB-колекції		Опис функцій	
Назва компонента	Призначення	Назва колекції	Призначення	Назва функції	Джерело даних/портка
Graphs	Головний компонент для візуалізації екологічних даних у вигляді графіків. Він агрегує та відображає дані з різних джерел, таких як локальні метеорологічні сенсори та OpenWeatherMap	LocalMeteorological	Зберігання локальних метеорологічних даних з Home Assistant. Використовуються для порівняння з зовнішніми джерелами	getLocalMeteorologicalData	Отримує дані з Home Assistant, знаходиться потрібну entity_id, витягує атрибути
LastSevenDays	Компонент для відображення середніх значень екологічних параметрів за останні 7 днів	OpenWeatherMap	Зберігання метеоданих із зовнішнього API OpenWeatherMap. Використовуються для порівняльного аналізу з локальними даними	addLocalMeteorologicalDataToDB	Отримує поточні погодні дані з Home Assistant і записує в MongoDB модель LocalMeteorological
LastSevenEntries	Компонент для відображення останніх 7 записів екологічних даних	Point	Містить координати встановлених сенсорів для відображення на інтерактивній мапі	getLocalMeteorologicalDataByPeriod	За константними змінними SEVEN_LATEST, SEVEN_DAYS, THIRTY_DAYS робить вибірку або агрегацію з MongoDB відповідно
LocalMeteorological	Компонент для відображення поточних локальних метеорологічних даних, отриманих з Home Assistant			getOpenWeatherMapData	Звертається до OpenWeatherMap API (місто: Zaporizhzhia) через API ключ
OpenWeatherMap	Компонент для відображення поточних погодніх даних, отриманих з OpenWeatherMap API			addOpenWeatherMapDataToDB	Отримує погодні дані через OpenWeather API, парсить, трансформує, зберігає у MongoDB модель OpenWeatherMap
Map	Компонент для відображення мапи з точками моніторингу екологічних даних			getOpenWeatherMapDataByPeriod	За константною змінною THIRTY_DAYS робить агрегацію з MongoDB
GreenStrategyPage	Компонент виконує функцію центрального хабу для візуалізації, порівняння та моніторингу екологічних даних. Він поєднує всі основні функціональні компоненти порталу, забезпечуючи зручний та узгоджений інтерфейс користувача для перегляду даних, графіків, мап та погодніх звітів			createPoint	Створює нову точку моніторингу за вхідними параметрами (position: {lat, lng}, sensor, street)
				getPoints	Отримує список усіх точок моніторингу з колекції Point
				deletePoint	Видаляє точку з MongoDB колекції Point по її ObjectId

Слайд 10

Рисунок В.10 – Слайд 10 «Основні рішення щодо програмної реалізації»

Запуск проєкту

Порядок розгортання

- клонувати репозиторий: `git clone https://github.com/AndrKirienko/diploma.git`;
- перейти в директорию проєкту: `cd diploma`;
- виконати `docker compose -f docker-compose-dev.yaml up` для запуску всіх сервісів;
- налаштувати токени доступу до метеосервісів;

1) для HomeAssistant:

- після запуску контейнерів перейдіть за адресою `http://localhost:8123`;
- авторизуйтесь → профіль → безпека → творити токен;
- збережіть токен та пропишіть його у `/server/src/constants.js` у змінну `TOKEN`;

2) для OpenWeatherMap:

- увійдіть на `https://home.openweathermap.org`;
- перейдіть у меню «API Keys», згенеруйте ключ;
- пропишіть його у файл `/server/src/constants.js` у змінну `OPENWEATHER_API_KEY`.

Доступність

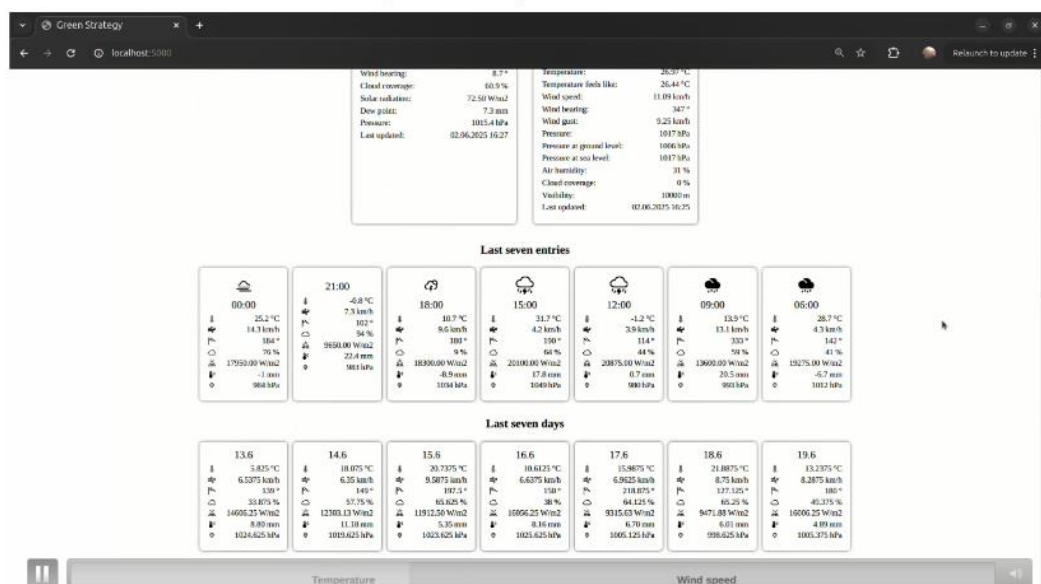
вебінтерфейс: `http://localhost:5000`;
сервер API: `http://localhost:3000`;
Home Assistant: `http://localhost:8123`.



Слайд 11

Рисунок В.11 – Слайд 11 «Запуск проєкту»

Демонстрація роботи системи



Слайд 12

Рисунок В.12 – Слайд 12 «Демонстрація роботи системи»

Висновки

У роботі вирішено актуальну проблему цифровізації екологічного моніторингу в межах університетської інфраструктури.

Створено вебпортал, який забезпечує зручний доступ до екологічних даних у реальному часі, сприяє підвищенню прозорості екополітики навчального закладу та формуванню екосвідомості серед студентів і викладачів.

Розроблена система дозволяє здійснювати порівняльний аналіз локальних показників з даними метеосервісів, візуалізувати історію змін у вигляді графіків, карт та таблиць.

Результати роботи було представлено на Щорічній науково-практичній конференції викладачів, науковців, молодих учених, аспірантів та студентів Національного університету «Запорізька політехніка» – «Тиждень науки-2025».

Слайд 13

Рисунок В.13 – Слайд 13 «Висновки»