МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

МОРОЗ ДАНИЛО РУСЛАНОВИЧ

Допускається до захисту:

В. о. завідувача кафедри

прикладної математики та кібербезпеки, доктор

філософії з математики,

_____ Луценко А.В.

«__»_____20__ р.

СТВОРЕННЯ ДОДАТКУ ДЛЯ ОПТИМІЗАЦІЇ РОБОТИ СМАРТ БУДИНКУ ЗА КОНЦЕПЦІЄЮ ВУГЛЕЦЕВОГО СЛІДУ

Спеціальність 105 Прикладна фізика та наноматеріали

Кваліфікаційна (магістерська) робота

Науковий керівник:

В.Г. Крижановский

д-р техн. наук, професор,

професор кафедри прикладної

математики та кібербезпеки

_____

(підпис)

Оцінка:___/___/_____

(бали за шкалою ЄКТС/за національною шкалою)

Голова ЕК:_____

(підпис)

Вінниця - 2025

ЗМІСТ

ВСТУП

Стрімкий розвиток інформаційних технологій, Інтернету речей (IoT) та кіберфізичних систем суттєво трансформує підходи до організації житлового простору [3] [5]. Концепція смарт-будинку передбачає інтеграцію сенсорів, виконавчих пристроїв, систем керування та аналітики з метою забезпечення комфортних, безпечних та енергоефективних умов для користувача [9]. Водночас, на фоні загострення проблеми зміни клімату та зростання обсягів викидів парникових газів, особливо актуальними стають питання оцінювання та зменшення вуглецевого сліду побутового сектору [4]. За оцінками міжнародних організацій, значну частку глобального споживання енергії та викидів CO_2 становлять саме будівлі – житлові та комерційні, – у тому числі за рахунок опалення, кондиціонування, освітлення та роботи побутових приладів [11]. Це формує суспільний та науковий запит на рішення, що дозволяють не лише автоматизувати керування інженерними системами, а й орієнтувати ці процеси на мінімізацію екологічного впливу.

Одним із ключових індикаторів такого впливу є вуглецевий слід – сумарний обсяг прямого та непрямого викиду парникових газів, виражений, як правило, в еквіваленті CO_2, який виникає внаслідок діяльності людини, функціонування будівель, інфраструктури та використання енергоресурсів [1] [7]. У контексті смарт-будинку вуглецевий слід формується, насамперед, за рахунок споживання електроенергії, теплової енергії, використання газу та інших палив. Точний облік споживання та коректне застосування коефіцієнтів емісії відкривають можливість для кількісної оцінки впливу конкретних сценаріїв роботи будинку, а також для пошуку оптимальних режимів функціонування, що поєднують комфорт користувача з екологічною відповідальністю. [12] [16]

Сучасні комерційні екосистеми для керування розумними будинками – такі як Google Nest, Apple HomeKit, Amazon Alexa Smart Home, Samsung SmartThings тощо – орієнтовані переважно на зручність, інтеграцію різних пристроїв та базову енергоефективність [18]. Проте в більшості випадків вони не надають користувачеві прозорого, кількісно обґрунтованого уявлення саме про

вуглецевий слід та не пропонують механізмів оптимізації, безпосередньо прив'язаних до викидів CO_2. Це створює нішу для спеціалізованих програмних рішень, які інтегрують методики розрахунку вуглецевого сліду з даними про реальне або змодельоване енергоспоживання смарт-будинку [13].

У цьому контексті актуальним є розроблення браузерного додатку, який дозволяє виконувати моніторинг енергоспоживання, розрахунок вуглецевого сліду та формування рекомендацій щодо оптимізації режимів роботи окремих пристроїв та систем будинку. Веб-орієнтований підхід забезпечує кросплатформеність, доступність із будь-якого пристрою, що має браузер, та спрощує оновлення й розгортання рішення. В умовах навчального та наукового середовища особливо цінною є можливість створити реалістичну, але водночас умовну (демонстраційну) модель такого додатку, яка б відображала принципи побудови архітектури, алгоритми розрахунку та базові можливості користувацького інтерфейсу, не вимагаючи повноцінної інтеграції з реальними IoT-пристроями.

Магістерська робота, присвячена розробці браузерного додатку для оптимізації роботи смарт-будинку за концепцією вуглецевого сліду, має на меті поєднати теоретичні засади екологічного менеджменту та енергоефективності з практичними підходами до проєктування веб-застосунків та IoT-орієнтованих систем. У рамках дослідження планується проаналізувати сучасний стан розвитку смарт-будинків та методів обчислення вуглецевого сліду, розробити концептуальну модель системи підтримки прийняття рішень для оптимізації роботи будинку та реалізувати прототип веб-додатку, що на прикладі демонструє можливість оцінювання та зниження вуглецевого сліду за рахунок зміни сценаріїв споживання енергоресурсів.

Структурно робота складається зі вступу, трьох розділів, висновків та списку використаних джерел. У першому розділі подано теоретичні відомості щодо смарт-будинків, концепції вуглецевого сліду та аналізу наявних рішень. У другому розділі описано методику розробки браузерного додатку, обґрунтовано вибір технологій та наведено алгоритми розрахунку показників. Третій розділ

присвячено безпосередній реалізації додатку, демонстрації його інтерфейсу та аналізу результатів роботи.

Мета даної роботи – розроблення концепції, методики та демонстраційного браузерного додатку, який дозволяє здійснювати моніторинг енергоспоживання смарт-будинку, розрахунок його вуглецевого сліду та оптимізацію роботи інженерних систем шляхом формування рекомендацій щодо зниження викидів CO_2.

Об'єкт дослідження – процеси функціонування смарт-будинку, зокрема взаємодія інтелектуальних пристроїв, систем керування енергоспоживанням та програмного забезпечення, що забезпечує автоматизацію побутових процесів.

Предмет дослідження – методи, алгоритми та програмні засоби оцінювання та оптимізації вуглецевого сліду смарт-будинку, а також архітектура веб-додатку, який реалізує ці можливості.

Завдання роботи:

1. Проаналізувати сучасні підходи до побудови смарт-будинків, структуру IoT-платформ та особливості інтеграції інтелектуальних пристроїв.
2. Дослідити методології оцінювання вуглецевого сліду, включаючи стандарти GHG Protocol, ISO 14064 та принципи формування коефіцієнтів викидів.
3. Опрацювати існуючі програмні рішення та екосистеми розумного дому, визначити їхні переваги та недоліки з точки зору екологічної оптимізації.
4. Сформувати концептуальну модель веб-додатку для моніторингу енергоспоживання та розрахунку викидів CO_2 на основі отриманих даних.
5. Обґрунтувати вибір технологій та інструментів для створення браузерного рішення (frontend, backend, БД, протоколи інтеграції).
6. Розробити алгоритми розрахунку вуглецевого сліду та сценарії оптимізації роботи систем смарт-будинку.

7. Реалізувати демонстраційний прототип веб-додатку, що містить ключові екрани, функції розрахунку та інтерфейс для користувача.
8. Виконати тестування та оцінити ефективність роботи створеного прототипу.
9. Сформувати висновки щодо можливостей, обмежень та перспектив удосконалення запропонованої системи.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ОПТИМІЗАЦІЇ СМАРТ-БУДИНКУ ТА КОНЦЕПЦІЯ ВУГЛЕЦЕВОГО СЛІДУ

1.1. Смарт-будинки: поняття, архітектура та принципи роботи

Сучасні смарт-будинки є одним із ключових напрямів розвитку кіберфізичних систем та Інтернету речей (IoT), оскільки поєднують у собі автоматизовані інженерні рішення, алгоритми обробки даних та інтелектуальні функції, орієнтовані на комфорт користувача та раціональне використання ресурсів. Смарт-будинок визначається як комплекс взаємопов'язаних пристроїв, сенсорів, виконавчих механізмів та програмних компонентів, які забезпечують моніторинг і керування процесами життєдіяльності будівлі в автоматичному або напівавтоматичному режимі. За визначенням ISO/IEC 30141, такі системи належать до розподілених IoT-архітектур, де фізичні пристрої взаємодіють із цифровими сервісами, формуючи єдине інтелектуальне середовище [5], [10].

У науковій літературі смарт-будинок розглядається як інтелектуальна система керування житловим простором, здатна аналізувати стан середовища, реагувати на зовнішні події та підтримувати оптимальні умови без активної участі людини. На відміну від традиційних будівель, оснащених окремими автоматизованими підсистемами, смарт-будинок передбачає інтеграцію всіх інженерних елементів у єдину екосистему, що забезпечує комплексне керування освітленням, опаленням, вентиляцією, кондиціонуванням, безпекою, побутовими приладами та мультимедійним обладнанням.

1.1.2. Архітектура смарт-будинку

Архітектура сучасних смарт-будинків традиційно поділяється на три структурні рівні, кожен з яких забезпечує певну частину функціонування системи.

Перший рівень — рівень пристроїв (Device Layer).
На цьому рівні знаходяться всі фізичні компоненти — сенсори, виконавчі механізми, IoT-пристрої, інтегровані побутові прилади. Саме вони здійснюють

первинне збирання даних і виконання команд. Пристрої мають обмежені обчислювальні ресурси та низьке енергоспоживання, що визначає використання спеціальних протоколів зв’язку.

Другий рівень — рівень мереж і шлюзів (Gateway & Network Layer). Пристрої рідко мають пряме підключення до хмарних сервісів, тому між ними використовується проміжний компонент — шлюз. Він збирає та маршрутизує дані, виконує перетворення протоколів і формує локальну мережу. У багатьох екосистемах (наприклад, Philips Hue або Xiaomi Home) шлюз є критичним елементом, що забезпечує стабільність роботи системи.

Третій рівень — рівень хмарних сервісів та застосунків (Cloud/Application Layer).

На цьому рівні відбувається обробка, зберігання та аналіз даних, формування статистики, реалізація користувацького інтерфейсу, створення інтелектуальних сценаріїв та аналітичних моделей. Саме тут працюють алгоритми машинного навчання, моделі прогнозування енергоспоживання та оптимізації.

1.1.3. Мережеві протоколи в системах смарт-будинку

Функціонування смарт-будинку ґрунтується на ефективній та стабільній комунікації між пристроями, що забезпечується набором спеціалізованих мережевих протоколів.

Wi-Fi є найпоширенішим протоколом завдяки високій пропускній здатності та широкій доступності. Він дозволяє передавати значні обсяги даних і ідеально підходить для камер, мультимедійних систем та пристроїв, які потребують стабільного інтернет-з’єднання. Однак через високе енергоспоживання Wi-Fi рідко використовується у малопотужних сенсорних пристроях. [14]

ZigBee — малопотужний протокол, оптимізований для сенсорних мереж. Його головною перевагою є підтримка топології "mesh", де кожен пристрій може ретранслювати сигнал, збільшуючи радіус дії. ZigBee використовується у

багатьох популярних системах, включаючи Philips Hue, IKEA Tradfri та Xiaomi Home.

Z-Wave працює на нижчих частотах (868/915 МГц), що дозволяє ефективно уникати перевантаженості діапазону 2.4 ГГц. Він має низьке енергоспоживання та відмінну проникність сигналу, завдяки чому широко застосовується у системах домашньої автоматизації, зокрема в Samsung Smart Things.

Thread є одним із найсучасніших протоколів і орієнтований на створення безпечних, енергоефективних та стійких до відмов IoT-мереж. Thread тісно пов'язаний зі стандартом Matter, який має на меті забезпечити універсальну сумісність пристроїв різних виробників [24].

MQTT — легковаговий протокол обміну повідомленнями за моделлю "publish/subscribe". Він використовується переважно на рівні серверних сервісів і підходить для систем, де важлива мінімальна затримка й гарантована доставка повідомлень.

1.1.4. Принципи роботи смарт-будинку

Робота смарт-будинку базується на низці концептуальних принципів, серед яких одними з найважливіших є автоматизація, централізоване керування, сценарність, аналітика та інтелектуальність.

Автоматизація передбачає, що пристрої здатні самостійно реагувати на зміни середовища. Наприклад, система освітлення може автоматично вимикатися за відсутності руху, а система клімат-контролю — оптимізувати температуру залежно від часу доби або зовнішніх умов.

Централізоване керування означає, що користувач може керувати всіма підсистемами через єдину платформу – мобільний додаток або вебінтерфейс. Це дозволяє створювати комплексні сценарії та інтегрувати різні види обладнання.

Сценарність – це можливість комбінувати дії різних пристроїв. Наприклад, сценарій "Немає вдома" може вимикати освітлення, переводити систему опалення у режим економії та активувати охоронні датчики.

Аналітика включає збирання та обробку даних про споживання ресурсів, виявлення аномалій, прогнозування пікових навантажень та формування рекомендацій для користувача. Завдяки цьому смарт-будинок не лише виконує команди, а й здатний покращувати власну ефективність.

Інтелектуальність реалізується через застосування алгоритмів машинного навчання, які дозволяють системі адаптуватися до поведінки користувачів, прогнозувати події та оптимізувати роботу пристроїв.

1.1.5. Смарт-будинки як інструмент зменшення вуглецевого сліду

Використання смарт-технологій у житлових будівлях сприяє зниженню енергоспоживання та, відповідно, викидів CO_2. Автоматизоване керування освітленням, опаленням, вентиляцією, кондиціонуванням та побутовими приладами дозволяє зменшувати непродуктивні витрати електроенергії. За даними Міжнародного енергетичного агентства, оптимізація роботи HVAC-систем за допомогою інтелектуальних алгоритмів може знизити загальне енергоспоживання будівлі на 20–40% залежно від її типу та кліматичних умов. [17]

Таким чином, смарт-будинки виступають потужною технологічною платформою для практичного впровадження концепції зменшення вуглецевого сліду, забезпечуючи підвищення ефективності використання ресурсів та зменшення екологічного навантаження.

1.2. Аналіз існуючих рішень та застосунків у сфері смарт-будинків

Ринок смарт-будинків активно зростає завдяки розвитку IoT-технологій, здешевленню сенсорики та поширенню бездротових стандартів зв’язку. У відповідь на потреби користувачів у підвищенні комфорту, безпеки та енергоефективності виникло кілька великих екосистем, що задають стандарти у сфері розумного дому. Серед них найбільш поширеними є Google Nest, Apple HomeKit, Amazon Alexa Smart Home та Samsung SmartThings. Хоча усі вони

забезпечують базові можливості автоматизації, між ними існують значні відмінності в архітектурі, функціональності, відкритості, підтримуваних пристроях і підходах до енергоефективності. Детальний аналіз цих рішень дозволяє визначити їхні переваги та недоліки в контексті завдання оптимізації вуглецевого сліду, а також сформувати підґрунтя для розроблення нової системи.

1.2.1. Google Nest

Google Nest є однією з найвідоміших платформ для автоматизації будинку, розробленою компанією Google з акцентом на інтелектуальному управлінні кліматом, безпекою та енергоспоживанням. Первинним продуктом екосистеми став Nest Learning Thermostat — інтелектуальний термостат, який здатний аналізувати поведінку користувача та будувати автоматизовані графіки нагрівання приміщення. Підхід Google ґрунтується на максимальному використанні хмарних технологій і машинного навчання: пристрої постійно передають дані в хмару, де вони аналізуються різноманітними алгоритмами.

Особливістю Google Nest є можливість відстеження споживання енергії, побудова історичних графіків і аналіз тенденцій теплового навантаження. Проте, незважаючи на розвинені функції аналізу, система не надає користувачеві прямих індикаторів вуглецевого сліду. Nest концентрується здебільшого на комфорті та економії ресурсів, залишаючи питання екологічної оцінки поза межами своєї основної функціональності. Для додаткової інтеграції Nest підтримує протокол Thread і є одним із перших пристроїв, сумісних зі стандартом Matter.

1.2.2. Apple HomeKit

Екосистема Apple HomeKit орієнтована на безпеку, надійність роботи та приватність користувачів. Однією з ключових особливостей HomeKit є локальна обробка даних: на відміну від більшості конкурентів, система може працювати без підключення до хмарних сервісів. Це підвищує швидкість реакції та безпеку,

однак обмежує широке використання складних алгоритмів аналізу, що потребують значної обчислювальної потужності.

HomeKit підтримує великий спектр пристроїв: розумні лампи, розетки, замки, датчики руху, камери тощо. Взаємодія між ними здійснюється за принципом "сцен" та "автоматизацій", що дозволяє створювати складні логіки роботи. Попри це, екосистема Apple практично не надає користувачам даних про енергоспоживання. Більшість пристроїв працює лише в режимі керування, а не моніторингу. Таким чином, HomeKit підходить для створення безпечної та стабільної платформи, проте як інструмент оцінки та оптимізації вуглецевого сліду він має суттєві обмеження.

1.2.3. Amazon Alexa Smart Home

Amazon Alexa Smart Home є однією з найбільш масштабованих платформ у сфері домашньої автоматизації. Alexa інтегрується з тисячами пристроїв від різних виробників, включаючи системи освітлення, HVAC, безпеки, мультимедіа та побутову техніку. Головною особливістю цієї екосистеми є використання голосового управління, що робить її однією з найбільш зручних для широкого кола користувачів.

На відміну від HomeKit, Alexa активно використовує хмарні обчислення, що дозволяє реалізовувати складні сценарії та аналізувати дані. Amazon пропонує функції базового моніторингу енергоспоживання через Alexa Energy Dashboard, але ця функціональність є обмеженою і не охоплює всі пристрої. Крім того, Alexa не виконує повноцінних розрахунків вуглецевого сліду, надаючи лише деякі узагальнені дані про споживання енергії.

1.2.4. Samsung SmartThings

Платформа SmartThings, що належить Samsung, є однією з найбільш відкритих та гнучких екосистем. Вона підтримує велику кількість пристроїв через протоколи ZigBee, Z-Wave, Thread, Wi-Fi та локальні API. SmartThings

відрізняється від інших платформ тим, що дозволяє глибоке кастомізоване програмування автоматизацій, надає доступ до сценаріїв через середовище SmartThings Edge та дозволяє створювати власні інтеграції.

Перевагою SmartThings є наявність засобів моніторингу енергоспоживання, включаючи окремі інтелектуальні розетки та енергометри. Проте, як і інші системи, SmartThings не підтримує прямі розрахунки вуглецевого сліду. Основний акцент зроблено на керуванні пристроями та забезпеченні сумісності між різними виробниками.

1.2.5. Аналітичне порівняння систем

Порівнюючи наведені екосистеми, можна зробити висновок, що всі вони успішно реалізують базову функціональність смарт-будинків: автоматизацію, централізоване керування, сценарності та зручні інтерфейси. Проте майже жодна з них не фокусується на кількісному аналізі енерговитрат у контексті вуглецевого сліду. Хоча деякі платформи, зокрема Google Nest та Samsung SmartThings, пропонують можливості моніторингу енергоспоживання, вони не інтегрують ці дані в аналітичні моделі оцінки CO_2.

Причини цього полягають у тому, що більшості комерційних систем важливо забезпечити універсальність та простоту для масового користувача. Додавання ж функцій екологічного аналізу потребує глибокої кастомізації, локальних коефіцієнтів викидів, складних інтерфейсів та розширених обчислень.

Отже, аналіз показує, що існуючі рішення не повністю закривають потребу у створенні інструменту для вимірювання та оптимізації вуглецевого сліду, а тому є підстави для розроблення спеціалізованого браузерного додатку, який би поєднував функціональність моніторингу, прогнозування та рекомендацій.

1.3. Концепція вуглецевого сліду

1.3.1. Визначення та сутність вуглецевого сліду

Вуглецевий слід (англ. *carbon footprint*) — це кількісна оцінка загального обсягу парникових газів (ПГ), викинутих у результаті діяльності людини, організації, продукту, процесу або системи. Традиційно обсяг викидів виражається у тоннах або кілограмах еквіваленту вуглекислого газу (CO_2e), що дозволяє враховувати не лише CO_2, але й інші ПГ, такі як метан (CH_4), закис азоту (N_2O), гідрофторвуглеці (HFCs), перфторвуглеці (PFCs), сірчистий гексафторид (SF_6), перераховані за потенціалом глобального потепління. [1] [7]

За визначенням IPCC (Міжурядова група з питань зміни клімату), вуглецевий слід є ключовою метрикою для оцінювання впливу діяльності людини на кліматичні процеси. Останні дослідження демонструють, що побутовий сектор є одним із найбільших джерел непрямих викидів: будинки споживають близько 30% глобальної електроенергії та продукують близько 20% пов'язаних викидів CO_2 (IEA, 2022).

У контексті смарт-будинку вуглецевий слід формується переважно за рахунок:

- споживання електроенергії;
- опалення та кондиціонування;
- використання газових систем;
- роботи побутових приладів;
- тепловтрат, спричинених неефективною експлуатацією будівлі.

Таким чином, точне вимірювання вуглецевого сліду житлового простору є критичним для оптимізації енергоспоживання та зменшення екологічного впливу.

1.3.2. Міжнародні стандарти оцінювання вуглецевого сліду

Методологія розрахунку вуглецевого сліду регламентується низкою міжнародних стандартів і протоколів, серед яких найважливішими є GHG Protocol та ISO 14064.

GHG Protocol (Greenhouse Gas Protocol)

GHG Protocol — це найбільш поширений глобальний стандарт для обліку та звітності викидів парникових газів. Він поділяє викиди на три категорії (Scopes) [2]:

- Scope 1 — прямі викиди (наприклад, спалювання палива в домашніх котлах, генераторах, автомобілях).
- Scope 2 — непрямі викиди від споживання електроенергії, тепла чи пари, вироблених іншими організаціями.
- Scope 3 — інші непрямі викиди (виробництво товарів, транспорт, утилізація відходів, постачальні ланцюги).

Для побутових систем найактуальнішими є Scope 2, а частково й Scope 1 (газові системи, пічне опалення).

ISO 14064

Стандарт ISO 14064 доповнює GHG Protocol і встановлює принципи:

- вимірювання,
- моніторингу,
- верифікації вуглецевих викидів.

Для наукових робіт важливе положення ISO 14064-1, яке регламентує оцінювання викидів для конкретних об’єктів (домогосподарств, підприємств, будівель). [1]

Ці стандарти становлять теоретичну основу для оцінки вуглецевого сліду у смарт-будинку.

1.3.3. Основні принципи розрахунку вуглецевого сліду

Процес розрахунку вуглецевого сліду для будівлі включає кілька основних етапів:

1. Визначення джерел споживання енергії

Для житлового сектора це:

- електроприлади;
- системи освітлення;
- HVAC-системи (опалення, вентиляція, кондиціонування);

- нагрів води;
- газові плити та котли.

2. Вимірювання або оцінка споживання енергоресурсів

Для електроенергії:

- показники лічильників,
- розрахунок на основі потужності приладів (Вт) і тривалості роботи,
- дані зі смарт-розеток або IoT-сенсорів.

3. Використання коефіцієнтів викидів (Emission Factors)

CO_2-викиди залежать від:

- енергетичного міксу регіону (вугілля, газ, ВДЕ),
- типу енергоресурсів,
- джерела виробництва електроенергії.

Для електроенергії України (станом на 2023–2024 рр.) середній коефіцієнт викидів становить приблизно:

0.35–0.45 кг CO_2e за 1 кВт·год,

але може варіювати залежно від сезону та частки ВДЕ.

4. Розрахунок вуглецевого сліду за основною формулою:

$$CO_2e = E \times EF$$

де

- E — спожита енергія (кВт·год),
- EF — коефіцієнт викидів (кг CO_2e/кВт·год).

Для газових систем використовуються інші коефіцієнти, враховуючи теплотворну здатність та хімічний склад газу.

1.3.4. Особливості оцінювання вуглецевого сліду у побутових системах

Розрахунок вуглецевого сліду в домогосподарствах має характерні особливості, пов’язані з варіативністю режимів роботи пристроїв та поведінковими патернами користувачів.

Змінність режимів споживання

Побутові прилади зазвичай мають динамічне навантаження:

- холодильники споживають енергію циклічно,
- кондиціонери — залежно від температури,
- освітлення — залежно від наявності людей у приміщенні.

Це робить складним точний прогноз без використання історичних даних.

Вплив поведінкових факторів

Дослідження показують, що від 20% до 40% енерговитрат у будинках пов’язано не з технічними параметрами приладів, а з користувацькими звичками (не вимикають світло, використовують техніку неекономічно тощо).

Смарт-будинок може зменшити цей вплив, автоматизуючи поведінкові сценарії.

Вплив тепловтрат і клімату

У холодних регіонах значну частину вуглецевого сліду формують системи опалення; у теплих — кондиціонування. Смарт-система, що керує HVAC, може найбільше вплинути на CO_2.

Неточність даних від користувачів у неавтоматизованих системах

Звичайні будинки не мають сенсорів споживання енергії для кожного приладу. Смарт-будинок вирішує це за допомогою:

- розумних лічильників,
- Wi-Fi та ZigBee енергомоніторів,
- даних про мікроспоживання.

Можливість автоматичної оптимізації

Смарт-системи можуть:

- визначати прилади з найбільшим CO_2-вкладом,
- формувати рекомендації щодо їх роботи,
- зменшувати роботу техніки у пікові години,
- обирати «вуглецево дешевші» часові проміжки для прання чи зарядки пристроїв.

1.3.5. Значення оцінки вуглецевого сліду у смарт-будинку

Оцінювання вуглецевого сліду у смарт-будинку має важливе екологічне та економічне значення. Воно дозволяє:

- підвищити енергоефективність житла,
- зменшити непродуктивні витрати ресурсів,
- оптимізувати сценарії роботи техніки,
- контролювати вплив поведінки користувачів,
- інтегрувати відновлювані джерела енергії,
- прогнозувати рахунки за електроенергію,
- знизити викиди CO_2.

Смарт-технології у поєднанні з аналізом вуглецевого сліду дозволяють будівлям наближатися до рівня Net Zero Home, де річний баланс викидів прагне до нуля.

1.4. Поточний стан досліджень оптимізації енергоспоживання у смарт-будинках

1.4.1. Моделі управління навантаженням та енергоспоживанням у житлових будівлях

У сучасних дослідженнях енергоменеджменту смарт-будинків ключова увага приділяється побудові моделей керування електричним навантаженням (load management) та систем домашнього енергоменеджменту (Home Energy Management Systems, HEMS). Такі моделі розглядають житло як динамічну систему, у якій електроприлади, системи опалення, кондиціонування, вентиляції, освітлення та заряджання електромобілів виступають окремими керованими навантаженнями, що можуть бути переміщені у часі, частково обмежені або адаптовані до умов енергосистеми. Ряд оглядових робіт вказує, що впровадження HEMS, інтегрованих з IoT-технологіями, дозволяє знизити енергоспоживання домогосподарств у середньому на 15–30 % порівняно з

традиційними будинками, за рахунок гнучкого керування навантаженням, врахування тарифів часу доби та поведінкових патернів користувачів.

Суттєвий розвиток отримали моделі керування попитом (Demand Side Management, DSM) та реакції попиту (Demand Response, DR), у яких смарт-будинок розглядається не лише як споживач, а як активний учасник енергосистеми. У таких моделях оптимізація графіка роботи приладів відбувається з урахуванням сигналів від енергопостачальної компанії, тарифів реального часу, обмежень мережі та комфорту мешканців. Окремі роботи пропонують комплексні DR-фреймворки для житлових будівель із підтримкою інтеграції електромобілів, фотоелектричних установок та накопичувачів енергії, що дозволяє суттєво зменшити пікові навантаження та підвищити стійкість мережі.

Актуальні систематичні огляди моделей енергоменеджменту у смарт-будинках демонструють, що більшість підходів базується на централізованому контролері, який отримує дані зі смарт-лічильників та сенсорів, прогнозує навантаження, а потім формує оптимальний розклад роботи приладів із урахуванням множини критеріїв: мінімізація витрат, зменшення енергоспоживання, зниження пікових потужностей, підвищення частки відновлюваних джерел у балансі.

1.4.2. Алгоритми оптимізації: від rule-based систем до методів машинного навчання

З позицій алгоритмічної реалізації виділяють кілька поколінь підходів до оптимізації енергоспоживання у смарт-будинках. Перші рішення базувалися переважно на простих rule-based системах, у яких поведінка приладів описувалася набором логічних правил: вмикати/вимикати обладнання при досягненні певних порогів температури, освітленості, часу доби або наявності мешканців. Ці системи відзначалися відносною простотою впровадження, але погано адаптувалися до складних сценаріїв, де потрібно враховувати тарифи, прогноз погоди, змінність поведінки користувачів та обмеження мережі.

Подальший розвиток привів до широкого застосування оптимізаційних методів: лінійного та цілочислового програмування, еволюційних алгоритмів, рою частинок, генетичних алгоритмів тощо. Ці підходи дозволяють формулювати задачу керування енергоспоживанням як задачу математичної оптимізації з обмеженнями, де критеріями можуть бути мінімізація сумарної вартості електроенергії, зменшення піків навантаження, максимізація самоспоживання енергії від фотоелектричних панелей тощо. Огляд сучасних оптимізаційних алгоритмів для домашніх енергоменеджмент-систем показує, що поєднання таких методів із DR-стратегіями дає значне зниження витрат та покращення профілю навантаження.

Останніми роками особливої популярності набули підходи, що використовують методи машинного навчання й глибинного навчання. Вони застосовуються для прогнозування коротко- та середньострокового енергоспоживання, класифікації типів навантажень, виявлення аномалій, побудови персоналізованих рекомендацій для користувачів. У ряді робіт пропонуються моделі на основі рекурентних нейронних мереж (зокрема, LSTM), здатні з високою точністю прогнозувати споживання енергії на рівні окремого домогосподарства, що дозволяє більш ефективно розв’язувати оптимізаційні задачі в режимі прогнозуючого керування.

Паралельно розвиваються персоналізовані системи оптимізації, у яких враховуються індивідуальні звички мешканців, їхні пріоритети щодо комфорту та готовність змінювати поведінку заради енергоефективності. Такі системи за допомогою адаптивних моделей машинного навчання аналізують детальні профілі споживання, і на цій основі пропонують або автоматично застосовують оптимізовані сценарії роботи пристроїв.

1.4.3. Інтеграція відновлюваних джерел енергії та розподіленої генерації

Значна частина сучасних досліджень у сфері смарт-будинків пов’язана з інтеграцією відновлюваних джерел енергії (ВДЕ) та технологій розподіленої генерації — насамперед фотоелектричних сонячних панелей та акумуляторних

систем зберігання енергії. У таких системах завдання енергоменеджменту ускладняються: необхідно не лише керувати споживанням, а й узгоджувати його з виробництвом енергії, що залежить від погодних умов, а також із процесами заряджання та розряджання накопичувачів.

Сучасні моделі смарт-будинків із сонячною генерацією часто використовують підходи до оптимізації навантаження, де метою є максимізація самоспоживання виробленої електроенергії, зменшення обмінних потоків з мережею та мінімізація витрат за рахунок використання гнучких тарифів. У низці робіт запропоновано системи планування роботи побутових приладів і заряджання електромобілів із урахуванням прогнозу сонячної генерації та обмежень зарядних пристроїв, що дозволяє істотно знизити як витрати, так і пов'язані викиди CO_2.

Паралельно ведуться дослідження інтеграції смарт-будинків до розумних мереж (smart grids), де будинки виступають активними учасниками енергосистеми – можуть гнучко змінювати навантаження, зберігати енергію, продавати надлишки у мережу. Комбіноване використання смарт-мереж, IoT-платформ та інтелектуальних алгоритмів управління, за результатами окремих досліджень, дозволяє знизити енергоспоживання на 20–25 %, а викиди CO_2 – до 30–35 %, одночасно підвищуючи надійність енергосистеми.

1.4.4. Перехід до вуглецево-орієнтованих та «carbon-aware» стратегій керування

Традиційно більшість досліджень у сфері енергоменеджменту смарт-будинків зосереджувалися на мінімізації вартості електроенергії та загального енергоспоживання. Однак у контексті посилення глобальної кліматичної політики дедалі більшої актуальності набувають так звані carbon-aware стратегії, у яких критерієм оптимізації виступає вже не лише кіловат-година чи грошова одиниця, а кілограми CO_2-еквіваленту [16] [11].

Новітні дослідження пропонують моделі керування навантаженням, що враховують вуглецеву інтенсивність електроенергії в різні періоди часу. Викиди, пов'язані з кВт·год, можуть змінюватися в залежності від того, які джерела

працюють у даний момент: вугільні, газові чи відновлювані. У таких системах побутові навантаження, які можна відкласти у часі (прання, сушіння, заряджання електромобіля), переносяться на ті інтервали, коли частка ВДЕ в енергобалансі є більшою, а вуглецева інтенсивність – нижчою.

З’являються моделі вуглецево-орієнтованого DR, де оптимізаційна задача формулюється з урахуванням обмежень комфорту користувача, можливостей гнучкого керування навантаженням і часово-залежних коефіцієнтів викидів. Такі підходи дозволяють досягати не лише економії коштів, а й відчутного скорочення вуглецевого сліду без істотної зміни звичного способу життя мешканців.

Водночас огляд наявних робіт свідчить, що практичні реалізації саме побутових інтерфейсів, орієнтованих на прозору візуалізацію вуглецевого сліду та інтеграцію екологічних показників у сценарії керування смарт-будинком, залишаються обмеженими. Більшість рішень зосереджені на рівні технічних моделей і прототипів, тоді як користувацькі додатки здебільшого пропонують лише базовий енергомоніторинг без прямої прив’язки до CO_2-метрик.

Ця прогалина формує наукове підґрунтя та практичну мотивацію для розроблення спеціалізованих веб-додатків, які б поєднували функції моніторингу енергоспоживання, оцінювання вуглецевого сліду та генерації рекомендацій щодо його зниження у доступній для кінцевого користувача формі. Саме до такого типу рішень належить демонстраційний браузерний додаток, розробці якого присвячено подальші розділи цієї магістерської роботи.

2. РОЗДІЛ. РОЗРОБКА БРАУЗЕРНОГО ДОДАТКУ ДЛЯ ОПТИМІЗАЦІЇ СМАРТ-БУДИНКУ ЗА ВУГЛЕЦЕВИМ СЛІДОМ

2.1. Постановка задачі та концепція роботи додатку

Сучасні системи автоматизації смарт-будинків забезпечують високий рівень комфорту, безпеки та дистанційного керування, однак мають істотні обмеження щодо екологічно орієнтованого управління енергоспоживанням. Попри широке поширення інтелектуальних термостатів, сенсорних мереж та централізованих контролерів, більшість комерційних рішень орієнтовані на покращення зручності або на часткову економію енергії, але не на комплексну оптимізацію вуглецевого сліду будівлі. Зокрема, практично відсутні системи, що надають користувачу інтегровані індикатори CO_2-викидів, пов'язаних із роботою побутових приладів, та не виконують оптимізацію сценаріїв роботи з урахуванням вуглецевої інтенсивності електроенергії та реальних даних енергоспоживання. [10]

Ураховуючи посилення глобальних екологічних вимог і прагнення зменшити антропогенний вплив на клімат, виникає потреба у створенні програмного забезпечення, яке дозволить користувачеві не лише контролювати прилади смарт-будинку, а й усвідомлено знижувати власний вуглецевий слід. Актуальність цього завдання підтверджується розширенням концепцій *carbon-aware computing*, впровадженням інтелектуальних мереж (*smart grids*) та переходом енергосистем до гнучких тарифів і високої частки відновлюваних джерел. У такому контексті смарт-будинок може виступати не лише об’єктом автоматизації, а й активним елементом екологічно орієнтованої енергомережі. [6]

З огляду на це, задачою даної магістерської роботи є розроблення браузерного додатку, який би забезпечував повний цикл аналізу енергоспоживання та оцінки вуглецевого сліду смарт-будинку, а також пропонував механізми оптимізації роботи приладів із урахуванням екологічних критеріїв. Демонстраційний прототип веб-додатку має дозволяти моделювати

різні сценарії роботи побутових систем, проводити порівняння екологічних показників для різних режимів, формувати рекомендації для зниження CO_2-викидів та візуалізувати результати у зручній формі.

Концепція роботи додатку

Браузерний додаток проєктується як інтерактивна система, яка виконує такі основні функції:

1. Моніторинг енергоспоживання — користувач вводить або отримує умовні дані про роботу різних пристроїв (час роботи, потужність, режими), після чого система автоматично перетворює їх у енергетичні показники (кВт·год).
2. Розрахунок вуглецевого сліду – на основі енергоспоживання та актуальних коефіцієнтів викидів додаток обчислює викиди CO_2 для кожного пристрою та для всієї системи загалом.
3. Візуалізація даних – результати подаються у вигляді графіків, діаграм і таблиць, що дає змогу оцінити внесок кожного приладу у формування загального вуглецевого сліду.
4. Оптимізація режимів роботи – додаток формує рекомендації щодо можливості зниження CO_2-викидів шляхом зміни часу роботи приладів, режимів використання HVAC-систем чи заміни неефективних пристроїв.
5. Сценарне моделювання – користувач може створювати альтернативні сценарії (наприклад, “Економія”, “Комфорт”, “Нічний тариф”, “Пікова генерація ВДЕ”) і порівнювати їхній вплив на викиди.
6. Інтерактивність та доступність – оскільки додаток є веб-базованим, він не потребує встановлення, доступний на будь-якому пристрої та може бути розширений до інтеграції з IoT у майбутньому.

Формалізація задачі

З формальної точки зору, задача оптимізації вуглецевого сліду смарт-будинку може бути подана як багатокритеріальна оптимізація, в якій необхідно:

$$\text{Мінімізувати } CO_2e = \sum_{i=1}^{n} (E_i \times EF)$$

за умов:

- обмеження на комфорт користувача (температура, час роботи приладу),
- можливість переносимості навантажень у часі,
- часові коефіцієнти вуглецевої інтенсивності електроенергії,
- обмеження на потужність та режими роботи пристроїв.

Результатом розв'язання цієї задачі є рекомендації та оптимізовані профілі роботи приладів, які дозволяють зменшити сумарний вуглецевий слід.

Обґрунтування застосування браузерного формату

Обрання веб-орієнтованого підходу зумовлене:

- доступністю (працює у будь-якому браузері без встановлення),
- простотою розповсюдження та оновлення,
- можливістю реалізації як локальної, так і хмарної логіки,
- зручністю інтеграції з REST API та MQTT у майбутньому,
- універсальністю інтерфейсу для будь-яких пристроїв (ПК, смартфон, планшет).

Таким чином, браузерний додаток є оптимальною платформою для демонстраційного реалізованого рішення в межах магістерської роботи.

2.2. Архітектура майбутнього рішення

Архітектура веб-додатку для оптимізації роботи смарт-будинку за вуглецевим слідом повинна бути побудована таким чином, щоб забезпечити модульність, масштабованість, просту підтримку та можливість розширення до повноцінного IoT-рішення. Оскільки додаток є браузерним, важливо забезпечити чітке розділення між клієнтською частиною (frontend), серверною логікою (backend) та підсистемами збору й обробки даних. Проєктоване рішення повинно ефективно працювати як у режимі ізольованої демонстрації, так і в перспективі інтеграції з датчиками, хмарними сервісами або реальними системами домашньої автоматизації [26].

2.2.1. Загальна структурна модель

Архітектуру додатку доцільно описати у вигляді трирівневої моделі, яка включає:

1. Frontend (клієнтська частина) – відповідає за інтерфейс, візуалізацію даних, взаємодію з користувачем і модель сценаріїв.
2. Backend (серверна частина) – виконує обчислення, логіку моделювання, зберігання даних і формування оптимізаційних рекомендацій.
3. Data Layer (джерела даних) – включає базу даних, модулі імпорту/експорту даних та потенційні канали IoT-взаємодії (у майбутньому).

Уся система побудована за принципами кластерної модульності, де кожен компонент може бути змінений або розширений без порушення логіки роботи інших частин.

2.2.2. Архітектура клієнтської частини (Frontend)

Frontend буде реалізовано як односторінковий веб-додаток (SPA), що забезпечує плавну, інтерактивну та швидку взаємодію користувача з системою. Для реалізації інтерфейсу доцільно застосувати сучасний фреймворк (React, Vue або Angular – у наступному підпункті 2.3 ми оберемо конкретний) [27].

У межах архітектури клієнтської частини виділяються такі ключові підсистеми:

Модуль інтерфейсу користувача

Відповідає за:

- візуальне подання даних (графіки, діаграми, таблиці),
- форму введення даних про прилади, режими роботи, потужність тощо,
- інструменти порівняння різних сценаріїв,
- відображення рекомендацій щодо оптимізації.

Особливо важливо забезпечити зрозумілу й інтуїтивну подачу інформації про вуглецевий слід – показники CO_2e повинні бути візуально простими для сприйняття.

Сценарний модуль

Забезпечує можливість створення та редагування сценаріїв:

- «Стандартний режим»
- «Економія енергії»
- «Нічний тариф»
- «Максимальна екологічність»
- «Кастомний сценарій»

Він дозволяє користувачеві задавати різні часові графіки, параметри роботи та пріоритети (комфорт, економія, CO_2).

Модуль обміну даними з backend

Забезпечує:

- передачу введених користувачем параметрів,
- запити до оптимізаційних сервісів,
- отримання результатів розрахунку CO_2,
- синхронізацію сценаріїв.

Комунікація відбувається через REST API, що робить додаток універсальним.

Лампи розжарювання

- *Ефективність:* Низька; більшість енергії перетворюється на тепло.
- *Викиди CO_2:* Високі через високе енергоспоживання.
- *Енергоспоживання:* Високе; наприклад, лампа потужністю 60 Вт споживає 60 Вт енергії.

Світлодіодні лампи (LED)

- *Ефективність*: Висока; споживають на 80–90% менше енергії порівняно з лампами розжарювання.
- *Викиди CO_2:* Низькі; зменшення енергоспоживання призводить до зниження викидів.
- *Енергоспоживання:* Низьке; наприклад, LED-лампа потужністю 10 Вт може замінити лампу розжарювання потужністю 60 Вт.

2.2.3. Архітектура серверної частини (Backend)

Backend виступає як логічне ядро додатку, яке відповідає за всі обчислення, моделювання, аналіз та формування рекомендацій. Він може бути реалізований на Node.js із використанням Express або Nest.js (вибір — у наступному підпункті).

Архітектура backend включає такі основні модулі:

Модуль енергетичного аналізу

Приймає параметри роботи пристроїв:

- номінальну потужність,
- час роботи,
- режим роботи,
- клас енергоефективності,
- коефіцієнт завантаження.

На основі цих параметрів система розраховує енергоспоживання:

$$E = P \times t$$

де

E — енергоспоживання (кВт·год),

P — потужність (кВт),

t — час роботи (год).

Модуль розрахунку вуглецевого сліду

Розраховує CO_2-викиди за формулою:

$$CO_2e = E \times EF$$

де EF — коефіцієнт викидів (залежить від регіону або встановлений користувачем).

Модуль підтримує:

- різні коефіцієнти для різних енергоресурсів,
- динамічну вуглецеву інтенсивність (carbon intensity by hour),
- можливість використання різних джерел даних.

Модуль оптимізації

Алгоритми оптимізації можуть включати:

- перенос роботи приладів на «вуглецево дешевші» години,
- пошук неефективних пристроїв,
- пропозиції щодо скорочення часу роботи,
- рекомендації щодо модернізації устаткування.

Система формує рекомендації у природній для користувача формі (наприклад: «Зміна режиму роботи кондиціонера з 18°C на 23°C зменшить викиди на 0.6 кг CO_2 за добу»).

Модуль зберігання даних

Використовується для збереження:

- сценаріїв,
- результатів розрахунків,
- історії дій,
- конфігурацій.

Доступ реалізується через ORM або нативні драйвери.

Природна вентиляція

- *Ефективність:* Залежить від погодних умов; може бути недостатньою для забезпечення якісного повітрообміну.
- *Викиди CO_2:* Можуть бути високими через необхідність додаткового опалення або охолодження.
- *Енергоспоживання:* Низьке, але може призводити до тепловтрат.

Механічна вентиляція з рекуперацією тепла (MVHR)

- *Ефективність:* Висока; дозволяє зберігати до 90% тепла з витяжного повітря.
- *Викиди CO_2:* Знижені за рахунок зменшення потреби в додатковому опаленні.
- *Енергоспоживання:* Помірне; енергоспоживання компенсується енергозбереженням.

2.2.4. Взаємодія компонентів: загальний сценарій роботи

1. Користувач відкриває браузерний додаток.
2. Вводить параметри роботи приладів або обирає попередньо налаштований сценарій.
3. Frontend передає дані на backend для розрахунку.
4. Backend:
    - обробляє дані,
    - виконує енергетичний аналіз,
    - розраховує CO_2,
    - формує рекомендації.
5. Результати повертаються у браузер у вигляді графіків, таблиць і текстових порад.
6. Користувач може:
    - змінити сценарій,
    - порівняти різні режими,
    - провести аналіз «до/після».

Таким чином, система працює як повноцінний інструмент екологічно орієнтованого енергоменеджменту.

2.3. Інструменти та технології

2.3.1. Вибір технологій для фронтенду

Клієнтська частина додатку є центральною точкою взаємодії користувача із системою, оскільки саме через браузер він переглядає результати розрахунків, вводить параметри роботи пристроїв та керує сценаріями. Для реалізації інтерфейсу існує широкий спектр технологій – від класичного JavaScript до сучасних фреймворків, таких як Angular, Vue.js та React. Однак найбільш доцільним вибором для даного проєкту є React у поєднанні з TypeScript. [21]

React вирізняється тим, що базується на компонентноорієнтованій архітектурі, що дозволяє розробляти інтерфейс у вигляді набору незалежних

модулів: форми, графіки, інформаційні картки, таблиці. Такий підхід особливо ефективний для додатків, які активно працюють із візуалізацією даних і потребують регулярного оновлення інтерфейсу. Завдяки використанню віртуального DOM React забезпечує високу продуктивність навіть у випадках значних обсягів інтерактивної графіки чи складних UI-компонентів. Це важливо, оскільки майбутній додаток повинен у реальному часі відображати результати розрахунків енергоспоживання, порівняння сценаріїв і динаміку CO_2-викидів [20].

Додатковою перевагою є величезна екосистема бібліотек та інструментів, що дозволяє легко поєднувати React з рішеннями для графіків (Recharts, D3, Chart.js), керування станом, маршрутизації та форм. Також важливим чинником вибору є використання TypeScript, що додає статичну типізацію та підвищує надійність коду, зменшуючи ризи помилок під час розробки та масштабування проєкту.

Таким чином, React у поєднанні з TypeScript стає оптимальним вибором для створення високоякісного, модульного, продуктивного та надійного інтерфейсу браузерного додатку.

2.3.2. Вибір серверної частини (Backend)

Серверна частина додатку відповідає за обробку даних, виконання математичних розрахунків, формування рекомендацій щодо оптимізації та управління сценаріями. У ролі бекендтехнології обрано Node.js, оскільки він дозволяє використовувати JavaScript/TypeScript як універсальну мову для всього проєкту. Використання єдиної мови в обох частинах системи значно спрощує підтримку додатку, забезпечує швидше навчання та прискорює процес розробки [22].

Node.js вирізняється своїм неблокуючим підходом до обробки запитів, що дозволяє серверу працювати з великою кількістю запитів одночасно без значних навантажень. Це особливо важливо для проєктів, які передбачають активний

обмін даними між клієнтом і сервером, а також можливість майбутньої інтеграції з реальними IoT-пристроями, які можуть працювати в потоковому режимі.

Для побудови API можуть бути використані два основні фреймворки: Express.js або Nest.js. Express забезпечує мінімалістичний підхід і простоту, а Nest.js – більш структурувану, модульну архітектуру, яка наслідує принципи Angular. У межах магістерської роботи можна використати Express.js як легший варіант, але при бажанні можна перейти до Nest.js для складніших систем. Гнучкість Node.js дозволяє реалізувати всі необхідні модулі: аналіз енергоспоживання, обчислення CO_2, управління сценаріями й формування рекомендацій. [25] [19]

2.3.3. База даних

Для зберігання результатів розрахунків, користувацьких сценаріїв та додаткових даних щодо налаштувань використовується база даних. Оптимальним рішенням для демонстраційного прототипу є MongoDB, оскільки вона працює з документноорієнтованою структурою – JSON-подібними документами. Така структура ідеально відповідає природі даних у системах цього типу: сценарії мають варіативний набір параметрів, дані про споживання енергії можуть мати динамічний формат, а структури пристроїв можуть змінюватися з часом [23].

MongoDB добре інтегрується з Node.js через бібліотеки Mongoose чи аналогічні ORM, що дозволяє створювати моделі даних та працювати з ними інтуїтивно. Документноорієнтований підхід полегшує розширення структури даних без міграцій бази й дозволяє швидко адаптувати систему під нові сценарії чи параметри, що виникнуть у ході дослідження.

2.3.4. Протоколи комунікації та інтеграції

Оскільки додаток є браузерним, основним способом комунікації між клієнтом і сервером виступає REST API. Такий підхід забезпечує просту та

прозору взаємодію: клієнт відправляє запит із даними про прилади або сценарії, сервер виконує розрахунки й повертає результат у форматі JSON. REST API легко інтегрується з будь-яким фронтендом і не потребує спеціальних адаптацій.

У перспективі система може бути розширена для підтримки реального часу (через WebSockets) або підключення до IoT-пристроїв через MQTT. REST залишається універсальним рішенням, доброю основою для майбутніх інтеграцій.

2.3.5. Інструменти для візуалізації даних

Оскільки важливою функцією додатку є наочне подання даних про енергоспоживання та вуглецевий слід, доцільно використовувати інструменти для побудови графіків і діаграм. Одним із найбільш гнучких рішень є бібліотека Recharts, яка добре інтегрується з React та дозволяє створювати інтерактивні, динамічні візуалізації. За необхідності можна застосовувати Chart.js або D3.js, однак Recharts забезпечує баланс між можливостями та простотою реалізації, що важливо в межах навчального проєкту.

2.3.6. Інструменти розробки та середовище виконання

Для написання коду використовуватиметься редактор Visual Studio Code, що має вбудовану підтримку JavaScript і TypeScript, а також інструменти для форматування коду, підказок і дебагу. Для управління бібліотеками використовуватиметься npm, а серверна частина запускатиметься в середовищі Node.js LTS. У якості системи контролю версій може бути обраний GitHub або GitLab.

2.3.7. Причини вибору описаних технологій

- Усі технології сучасні, актуальні та підтримувані великими спільнотами.
- Вони забезпечують швидкий час розробки.

- Легкість документування та демонстрації — важливо для магістерської.
- Архітектура залишається модульною і готовою до розширення.
- Frontend + backend на одній мові спрощує весь проєкт.
- Інструменти ідеально підходять для візуалізації CO_2 та енергетичних даних.
- У разі бажання додаток можна легко підключити до реальних IoT-пристроїв.

3. РОЗДІЛ. ЗАГАЛЬНА АРХІТЕКТУРА ТА СТРУКТУРА ДОДАТКУ

3.1. Архітектура розробленого веб-додатку

Розроблений веб-додаток базується на клієнт-серверній архітектурі, де кожен рівень виконує окрему функцію та взаємодіє з іншими через стандартизований REST API. Такий підхід забезпечує масштабованість, зрозумілість структури та можливість незалежного вдосконалення складових системи. [19]

Клієнтська частина (frontend), реалізована у технологіях React та TypeScript, відповідає за відображення інтерфейсу, взаємодію з користувачем і подання інформації у вигляді таблиць, карток та графічних компонентів. Усі введені користувачем дані — такі як параметри приладів, коефіцієнт емісії або цільові значення для оптимізації — надсилаються на сервер і обробляються вже на бекенді.

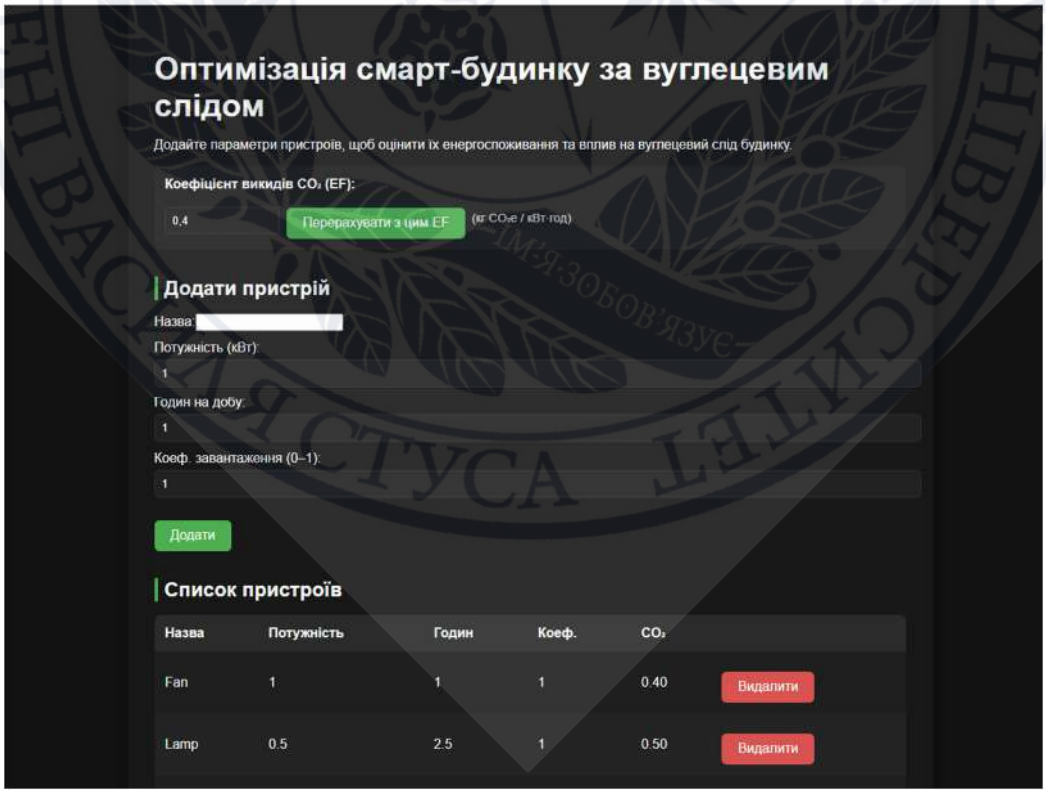


Рисунок 3.1

Серверна частина (backend), побудована на Node.js та Express, реалізує логіку обчислення енергоспоживання, визначення вуглецевих викидів, формування рекомендацій та обробку сценаріїв. Усі операції виконуються на стороні сервера для забезпечення стабільності, однорідності обчислень та можливості подальшої інтеграції з IoT-пристроями або зовнішніми API. Додатково сервер містить евристичний оптимізаційний модуль, який автоматично формує сценарій із зменшеним рівнем CO_2. [22] [25]

Рисунок 3.2

Зберігання інформації здійснюється у хмарній базі даних MongoDB Atlas, що дозволяє зберігати пристрої та сценарії у форматі документів. Це усуває потребу у власному сервері БД на комп'ютері користувача та надає можливість працювати зі сценаріями навіть після перезавантаження системи.

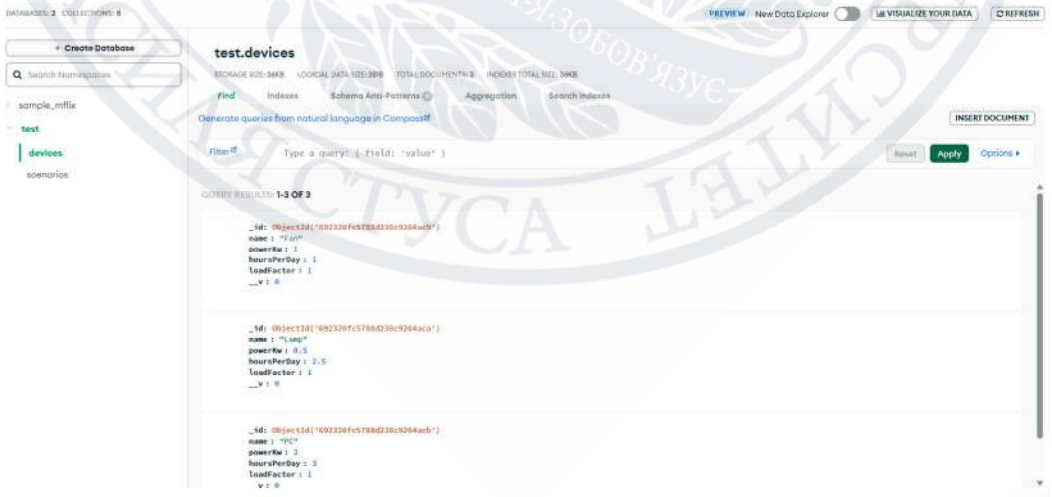


Рисунок 3.3

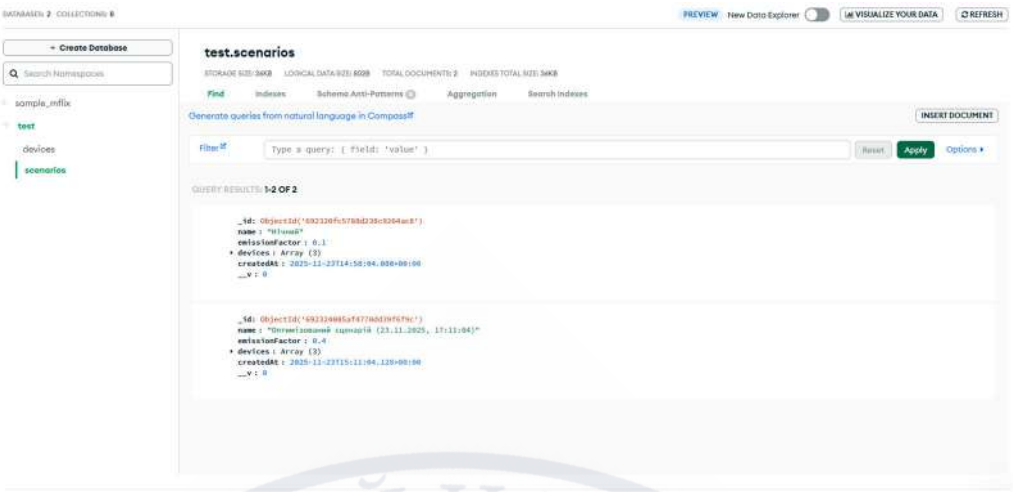


Рисунок 3.4

Архітектура додатку забезпечує плавний цикл роботи. Користувач вводить параметри пристроїв, після чого додаток розраховує енергоспоживання й формує діаграму розподілу CO_2. Далі користувач може створювати сценарії, порівнювати їх, застосовувати або генерувати оптимізований сценарій. Після кожної зміни інтерфейс фронтенду автоматично оновлюється, отримуючи нові дані через API.

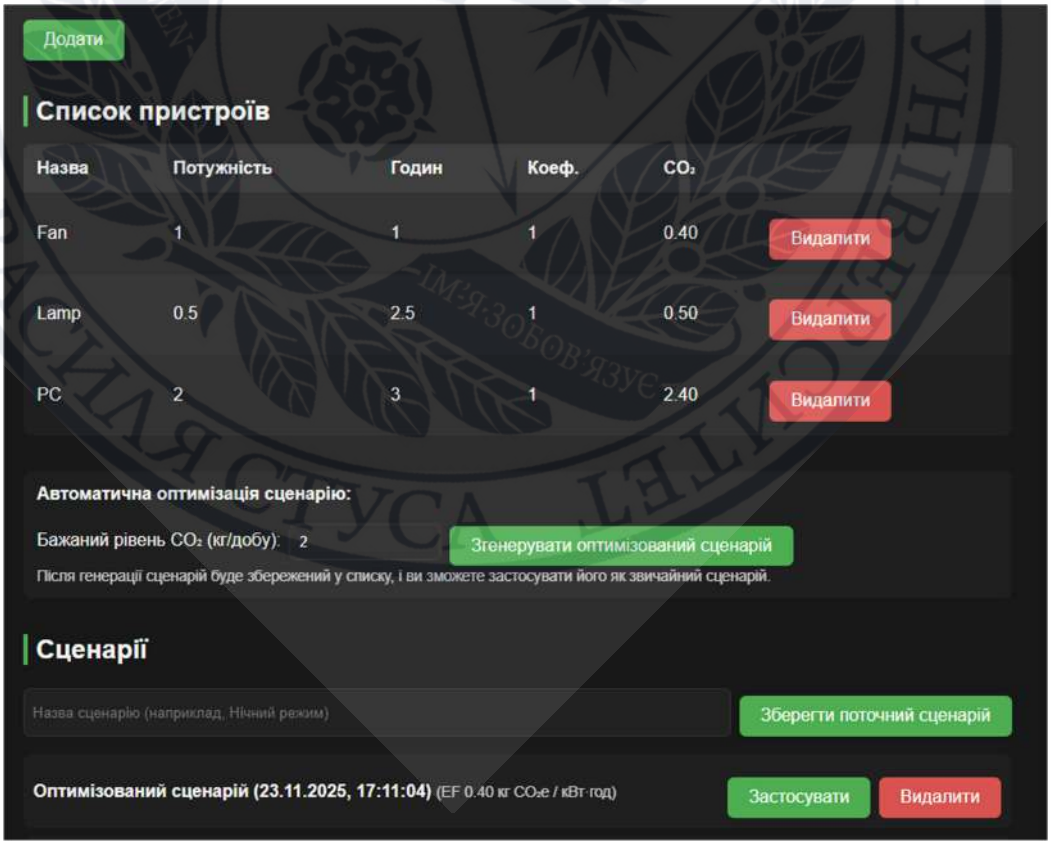


Рисунок 3.5

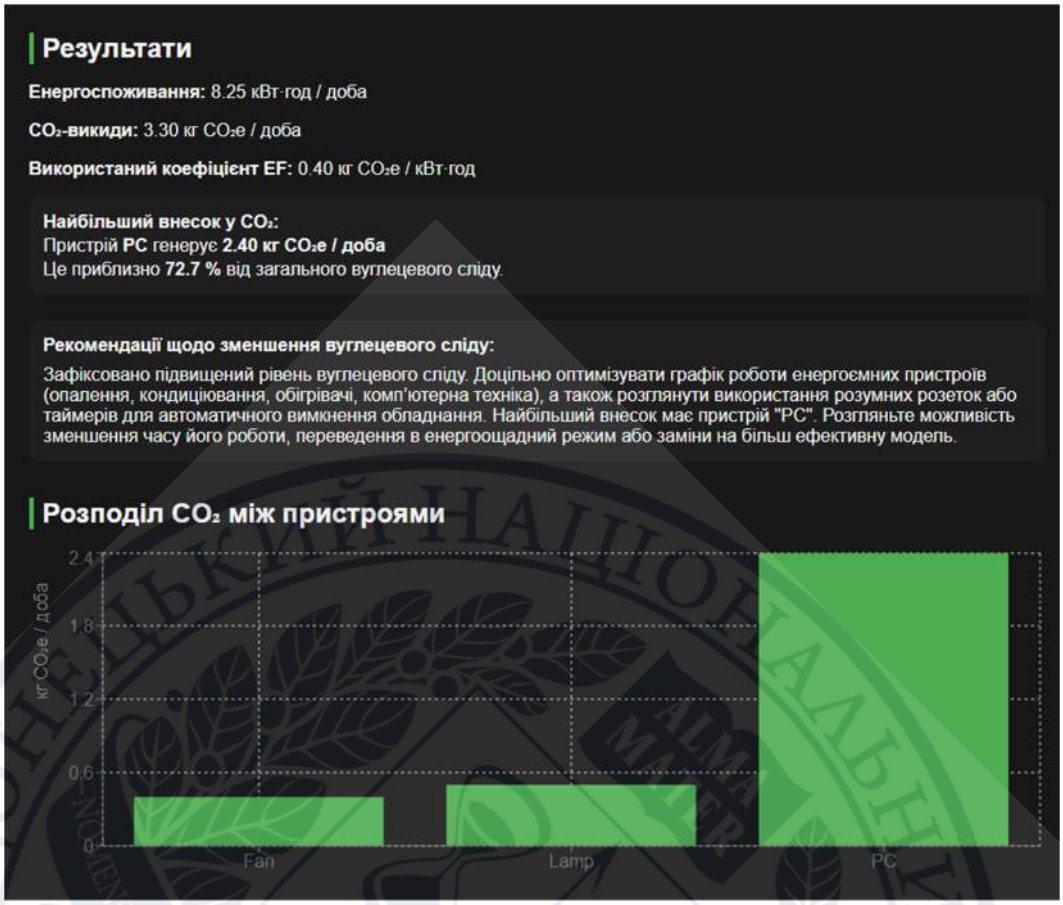
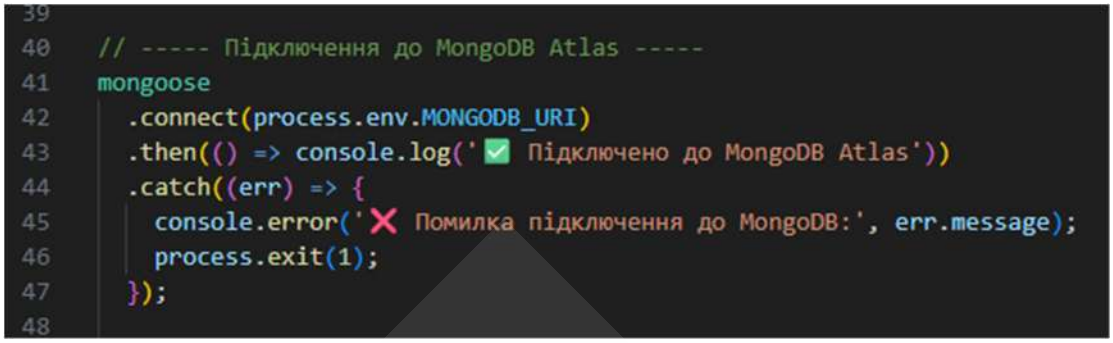


Рисунок 3.6

3.2. Реалізація серверної (бекенд) частини веб-додатку

Серверна частина розробленого веб-додатку реалізована на платформі Node.js із використанням фреймворку Express, що забезпечує просту та ефективну обробку HTTP-запитів, а також масштабовану структуру API. Бекенд відповідає за виконання ключових бізнес-операцій, включаючи обчислення енергоспоживання, визначення вуглецевих викидів, роботу з базою даних та формування автоматично оптимізованих сценаріїв.

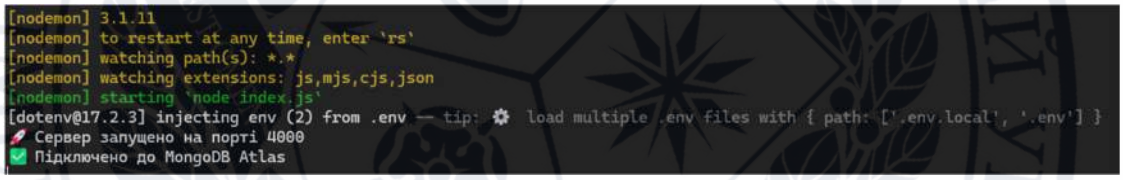
Структура бекенду включає основний файл index.js, у якому реалізовані налаштування сервера, підключення до MongoDB Atlas, визначення маршрутів і логіка обробки даних. Проєкт побудований так, щоб забезпечити максимально просту підтримку та можливість подальшої інтеграції з іншими сервісами або мікросервісами.



```
39
40  // ----- Підключення до MongoDB Atlas -----
41  mongoose
42    .connect(process.env.MONGODB_URI)
43    .then(() => console.log('✅ Підключено до MongoDB Atlas'))
44    .catch((err) => {
45      console.error('❌ Помилка підключення до MongoDB:', err.message);
46      process.exit(1);
47    });
48
```

Рисунок 3.7

Після запуску бекенд-додаток встановлює захищене підключення до хмарної бази даних MongoDB Atlas. Підключення реалізоване за допомогою бібліотеки Mongoose, яка забезпечує створення схем, валідацію даних і гнучкі CRUD-операції. Для підключення використовується файл конфігурації .env, у якому зберігається рядок підключення до кластера MongoDB.



```
[nodemon] 3.1.11
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
[dotenv@17.2.3] injecting env (2) from .env — tip: ⚙ load multiple .env files with { path: ['.env.local', '.env'] }
🚀 Сервер запущено на порті 4000
✅ Підключено до MongoDB Atlas
```

Рисунок 3.8

У серверній частині визначені моделі Device та Scenario, що описують структуру відповідних документів у базі. Модель пристрою включає назву, потужність, час роботи та коефіцієнт завантаження, тоді як модель сценарію містить набір пристроїв, коефіцієнт викидів EF і дату створення. Такий підхід дозволяє зберігати сценарії як готові конфігурації, що легко відтворюються в інтерфейсі користувача.

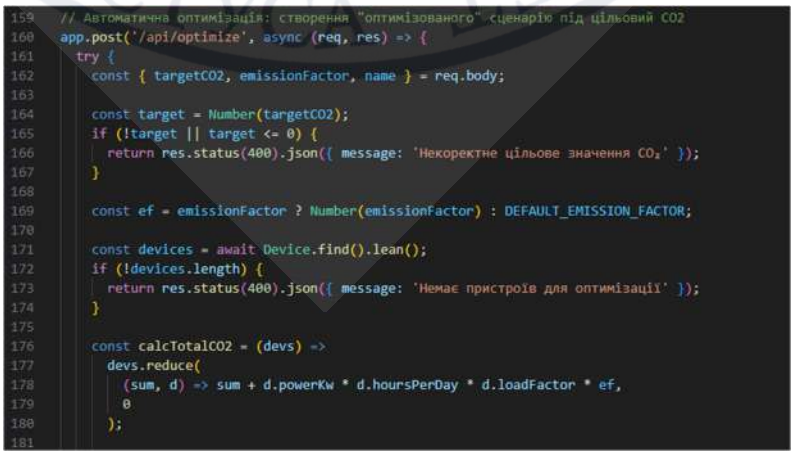
```
const deviceSchema = new mongoose.Schema({
  name: { type: String, required: true },
  powerKw: { type: Number, required: true },
  hoursPerDay: { type: Number, required: true },
  loadFactor: { type: Number, required: true },
});

const Device = mongoose.model('Device', deviceSchema);

const scenarioSchema = new mongoose.Schema({
  name: { type: String, required: true },
  emissionFactor: { type: Number, required: true },
  devices: [
    {
      name: String,
      powerKw: Number,
      hoursPerDay: Number,
      loadFactor: Number,
    },
  ],
  createdAt: { type: Date, default: Date.now },
});
```

Рисунок 3.9

API реалізовано у вигляді REST-маршрутів. Зокрема, маршрути /api/devices забезпечують роботу з пристроями, а /api/scenarios — створення, застосування та видалення сценаріїв. Особливе значення має маршрут /api/optimize, який реалізує евристичний алгоритм зменшення CO_2-викидів. Алгоритм аналізує пристрої, визначає ті, що мають найбільший внесок у викиди, і поступово знижує їхній час роботи до досягнення цільового рівня CO_2. На основі отриманих параметрів система автоматично створює новий оптимізований сценарій у базі даних.



```
// Автоматична оптимізація: створення "оптимізованого" сценарію під цільовий CO2
app.post('/api/optimize', async (req, res) => {
  try {
    const { targetCO2, emissionFactor, name } = req.body;

    const target = Number(targetCO2);
    if (!target || target <= 0) {
      return res.status(400).json({ message: 'Некоректне цільове значення CO₂' });
    }

    const ef = emissionFactor ? Number(emissionFactor) : DEFAULT_EMISSION_FACTOR;

    const devices = await Device.find().lean();
    if (!devices.length) {
      return res.status(400).json({ message: 'Немає пристроїв для оптимізації' });
    }

    const calcTotalCO2 = (devs) =>
      devs.reduce(
        (sum, d) => sum + d.powerKw * d.hoursPerDay * d.loadFactor * ef,
        0
      );
```

Рисунок 3.10

Комунікація між фронтенд-додатком і сервером здійснюється через JSON-запити. При додаванні пристрою або зміні EF сервер виконує обчислення енергоспоживання та викидів, після чого передає оновлені результати клієнту. Завдяки такій взаємодії інтерфейс реагує на зміни миттєво, а всі розрахунки виконуються централізовано на бекенді.

Таким чином, бекенд частина виконує роль аналітичного ядра системи, забезпечуючи точні розрахунки, надійне зберігання інформації та логіку оптимізації, що дозволяє користувачеві працювати зі сценаріями смарт-будинку на основі екологічних показників.

3.3. Реалізація користувацького інтерфейсу веб-додатку

Користувацький інтерфейс веб-додатку є одним із ключових компонентів системи, оскільки саме він забезпечує прямий взаємозв'язок користувача з інструментом оцінювання та оптимізації вуглецевого сліду смарт-будинку. Усі елементи інтерфейсу реалізовано на основі бібліотеки React із використанням TypeScript, що забезпечує високу стабільність, передбачуваність поведінки компонентів та розширення функціональності без зміни основної архітектури.

Після запуску додатку користувач потрапляє на головну сторінку, яка побудована у вигляді односторінкового застосунку. Інтерфейс організований таким чином, щоб відобразити всі необхідні елементи: форму для введення характеристик приладів, таблицю поточної конфігурації, блок управління коефіцієнтом викидів, розділ роботи зі сценаріями, панель результатів та інфографіку у вигляді діаграми.

Одним із основних елементів інтерфейсу є форма введення параметрів пристрою. Вона дозволяє користувачеві додати прилад, вказавши його назву, потужність у кіловатах, середню кількість годин роботи на добу та коефіцієнт завантаження. Реактивність інтерфейсу дає можливість миттєво оновлювати дані пристроїв та результати обчислень після їх надсилання на сервер. Це створює

максимальну зручність для користувача під час дослідження різних конфігурацій.

Рисунок 3.11

Після додавання приладу дані відображаються у структурованій таблиці. Кожен рядок містить інформацію про пристрій, а також кнопку для його видалення. Таблиця оновлюється автоматично після кожної зміни, оскільки фронтенд отримує оновлений перелік пристроїв від сервера. Такий механізм забезпечує цілісність даних і виключає можливість розбіжностей між інтерфейсом та серверною частиною.

Рисунок 3.12

Окремий блок інтерфейсу присвячений роботі зі сценаріями. Користувач може створювати сценарії, надаючи їм довільні назви, застосовувати їх, видаляти або зберігати декілька конфігурацій для подальшого порівняння. Сценарії зберігаються у хмарній базі MongoDB, що дозволяє працювати з ними навіть після перезапуску ПК чи браузера. Інтерфейс сценаріїв побудовано у вигляді

інтерактивного списку, кожен елемент якого супроводжується кнопками керування.

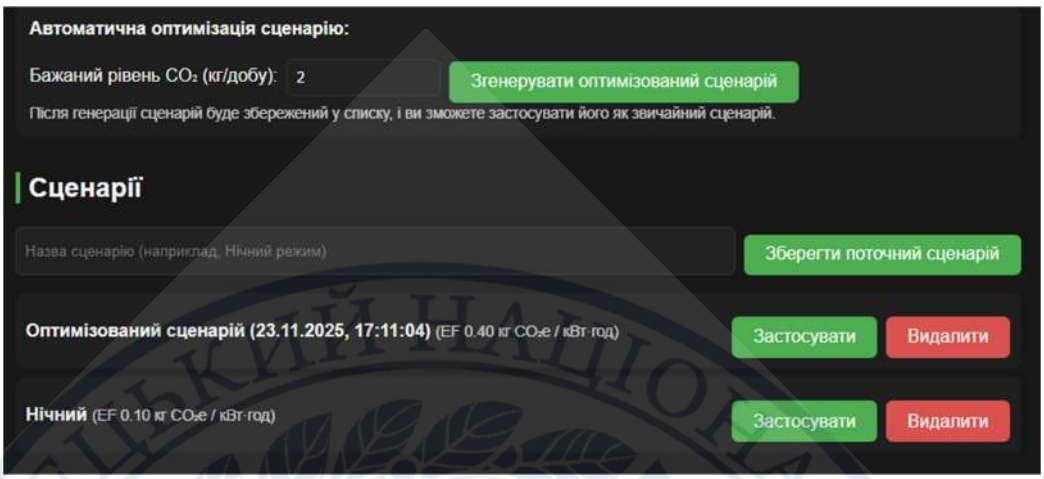


Рисунок 3.13

Ще одним важливим елементом є блок управління коефіцієнтом викидів CO_2. Користувач може ввести нове значення EF, після чого додаток оновлює розрахунок загальних викидів. Це дозволяє легко моделювати вплив різних енергетичних джерел або локальних тарифів. Крім того, система підтримує функцію автоматичної оптимізації, яка використовує евристичний підхід для зниження загальних викидів.

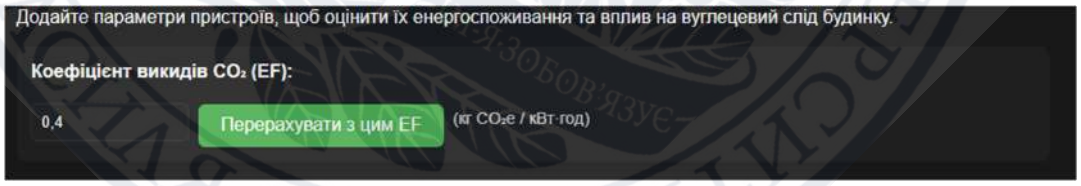


Рисунок 3.14

Для представлення результатів розрахунків застосовується панель з підсумковими показниками. Вона демонструє загальне енергоспоживання, сумарні викиди та ідентифікує пристрій, який є найбільшим джерелом CO_2. Така інформація допомагає користувачеві швидко оцінити вплив окремих приладів на загальний результат.

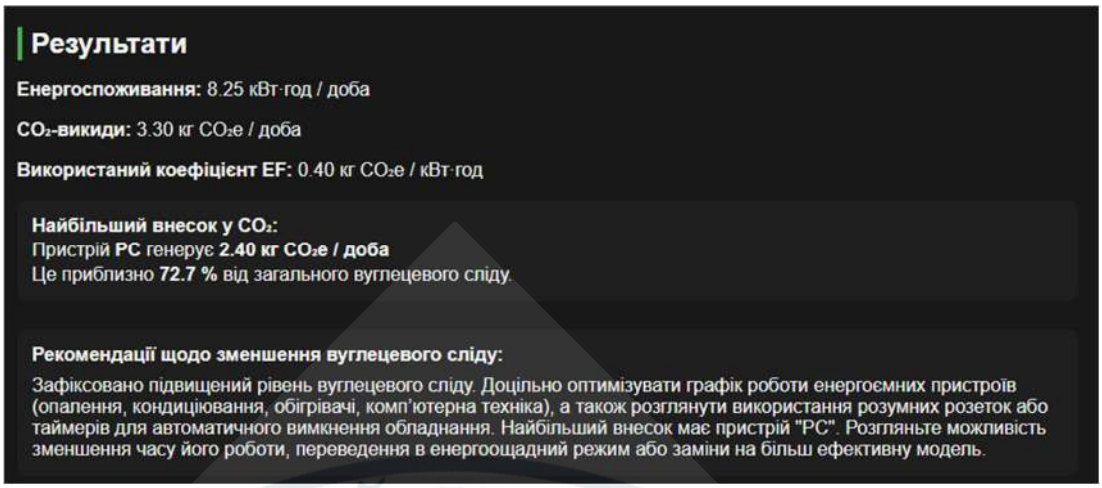


Рисунок 3.15

Візуалізація результатів також доповнюється графічним компонентом, що побудований на основі бібліотеки Recharts. Діаграма відображає внесок кожного пристрою у загальні викиди CO_2, дозволяючи порівняти їх між собою. Це підсилює аналітичну цінність системи та робить результати більш наочними для користувача.

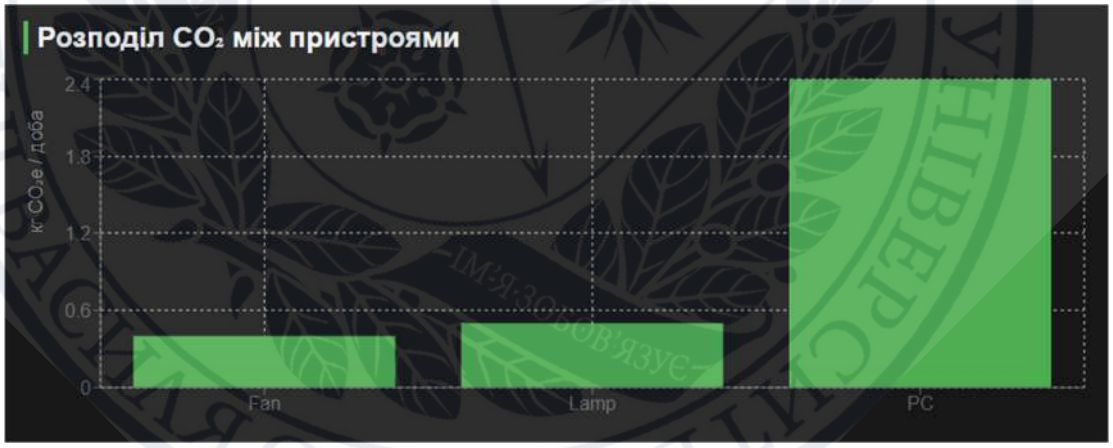


Рисунок 3.16

Таким чином, інтерфейс користувача виконує важливу роль інструменту аналізу та взаємодії. Він забезпечує простоту, логічну структурованість та швидкість роботи, що є ключовими вимогами сучасних веб-систем, орієнтованих на екологічну аналітику.

3.4. Реалізація алгоритму оптимізації та механізму сценаріїв у веб-додатку

Алгоритм оптимізації вуглецевого сліду та механізм роботи зі сценаріями становлять центральну частину програмного забезпечення, оскільки саме ці модулі дозволяють користувачеві не лише аналізувати наявні конфігурації пристроїв смарт-будинку, але й отримувати автоматично сформовані рекомендації для зниження викидів. У розробленій системі обидва модулі реалізовано на серверній частині, що гарантує коректність розрахунків і стабільність роботи незалежно від змін у користувацькому інтерфейсі [9], [11].

Сценарії у системі визначаються як окремі набори параметрів, що включають перелік пристроїв, їх потужність, тривалість роботи, коефіцієнт завантаження та значення коефіцієнта викидів. Кожен сценарій може бути створений користувачем вручну або автоматично – у результаті роботи евристичного алгоритму оптимізації. Зберігання сценаріїв у хмарній базі MongoDB Atlas дозволяє зберігати історію конфігурацій та відтворювати їх у будь-який час, що значно підвищує аналітичну цінність системи.

Рисунок 3.17

Алгоритм оптимізації базується на евристичному підході, завданням якого є поступове зменшення загальних викидів CO_2 до заданого користувачем рівня. На відміну від машинного навчання або математичного моделювання високої

складності, евристична модель має перевагу у зрозумілості, передбачуваності результатів та швидкості виконання. Її логіка полягає у послідовному зменшенні часу роботи тих пристроїв, які забезпечують найбільший внесок у загальний вуглецевий слід.

Після отримання від користувача бажаного значення CO_2-добових викидів сервер визначає поточний рівень емісії та порівнює його з цільовим. Якщо існуюча конфігурація перевищує встановлену норму, система формує копію переліку пристроїв і ініціює цикл оптимізації. На кожній ітерації алгоритм обирає пристрій із найбільшими викидами та зменшує його тривалість роботи на визначений крок. Після цього обчислення повторюється, і цикл триває доти, доки загальний рівень CO_2 не наблизиться до заданого значення або не буде досягнута межа можливого зниження. Такий підхід дозволяє оптимізувати конфігурацію без втрати функціональності системи — прилади продовжують працювати, але з меншим навантаженням.

Після завершення оптимізації сервер автоматично створює новий сценарій у базі даних. У цьому сценарії відображається скоригована тривалість роботи кожного пристрою, а також нові сумарні викиди, що дозволяє користувачеві порівняти попередню та оптимізовану конфігурації. Крім того, сценарій має часову позначку та назву, яка включає дату та час його формування, що полегшує ідентифікацію в інтерфейсі.

Рисунок 3.18

На стороні фронтенду сценарії відображаються у вигляді інтерактивного списку, де кожен сценарій можна застосувати, видалити або зберегти новий. Застосування сценарію викликає маршрути бекенду, які очищують поточну конфігурацію пристроїв і завантажують дані зі збереженого сценарію. Це

дозволяє користувачеві швидко перемикатися між конфігураціями та оцінювати їхнє енергетичне та екологічне значення.

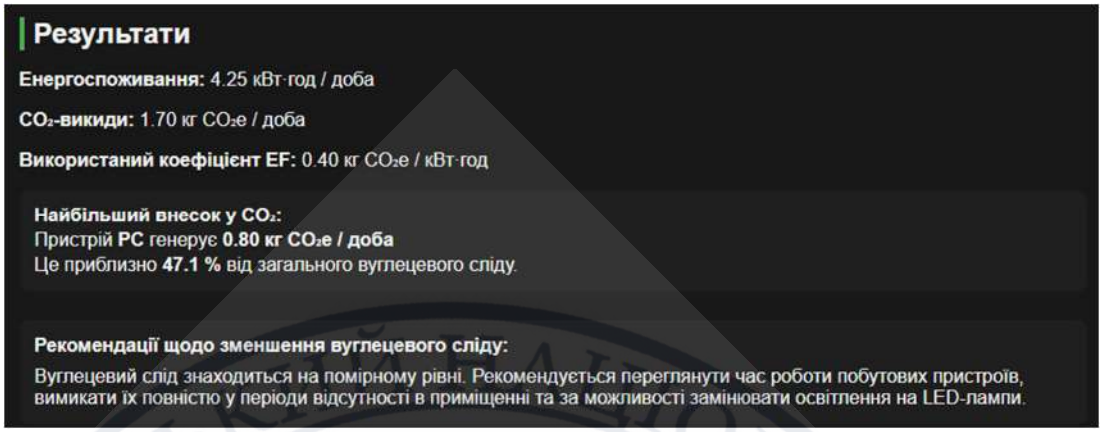


Рисунок 3.19

Реалізація механізму сценаріїв у поєднанні з евристичною оптимізацією створює можливість гнучкого управління енергоспоживанням у межах смарт-будинку. Користувач може не лише отримувати результати розрахунків, але й формувати моделі поведінки системи у реальних умовах. Проєктована функціональність може бути розширена підключенням реальних IoT-пристроїв, динамічним оновленням тарифів або інтеграцією з системами «розумної мережі» у майбутньому. [12] [17]

ВИСНОВКИ

У магістерській роботі досліджено методи оцінювання та зменшення вуглецевого сліду в системах розумного дому. На основі аналізу сучасних підходів до енергоменеджменту розроблено веб-додаток для оптимізації роботи побутових приладів з екологічної точки зору.

Програмний продукт реалізовано на базі сучасного стеку технологій MERN (MongoDB, Express, React, Node.js). Така клієнт-серверна архітектура забезпечила гнучкість системи, зручність інтерфейсу та надійність зберігання даних про енергоспоживання. Функціональні можливості додатку дозволяють користувачам моніторити енерговитрати, розраховувати викиди CO_2 та моделювати різні сценарії експлуатації техніки.

Створений прототип довів ефективність веб-орієнтованих інструментів для підтримки прийняття рішень у сфері екологічного енергоменеджменту. Система має практичну цінність та відкриту архітектуру для подальшої інтеграції з реальними IoT-сенсорами та впровадження методів машинного навчання

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ ISO 14064-1:2015. Парникові гази. Частина 1. Вимоги та настанови щодо кількісного визначення і звітності про викиди [Електронний ресурс]. – Режим доступу: http://online.budstandart.com/ua/catalog/doc-page?id_doc=63955
2. Global Protocol for Community-Scale Greenhouse Gas Emission Inventories [Електронний ресурс]. – Режим доступу: https://ghgprotocol.org/sites/default/files/standards/GHG_Protocol_for_Cities_0.pdf
3. ISO/IEC 30141:2018. Internet of Things (IoT) — Reference Architecture [Електронний ресурс]. – Режим доступу: https://www.iso.org/standard/65695.html
4. Методика розрахунку викидів парникових газів, пов'язаних із конфліктами та надзвичайними ситуаціями [Електронний ресурс]. – Режим доступу: https://ecoaction.org.ua/wp-content/uploads/2024/02/metodyka-otsinky-vykydiv.pdf
5. Аналіз технологій «Інтернет речей» як інструменту енергозбереження та підвищення комфорту [Електронний ресурс]. – Режим доступу: http://it-visnyk.kpi.ua/article/view/245123
6. Особливості інтеграції розумних мереж Smart Grid в енергетичну систему України [Електронний ресурс]. – Режим доступу: https://energy-security.org.ua/smart-grid-integration-2022
7. Вуглецевий слід продукту: методики оцінювання та перспективи впровадження в Україні [Електронний ресурс]. – Режим доступу: https://center-ltd.com.ua/novyny/ozrahuvaty-vugletsevyj-slid-produktu/
8. Напрями й інструментарій реалізації енергоефективних стратегій у будівництві [Електронний ресурс]. – Режим доступу: http://mdu.in.ua/zbirnyk-bud-tech-2024.pdf
9. Systematic Review of Optimization Methodologies for Smart Home Energy Management Systems (HEMS) [Електронний ресурс]. – Режим доступу: https://www.mdpi.com/1996-1073/18/19/5262

10. Smart Home Energy Management System: A Review of Algorithms and Architectures [Електронний ресурс]. – Режим доступу: https://www.tandfonline.com/doi/full/10.1080/17512549.2022.2045678
11. Carbon-Aware Demand Response for Residential Smart Buildings: Strategies and Algorithms [Електронний ресурс]. – Режим доступу: https://www.mdpi.com/journal/electronics/special_issues/carbon_aware_grid
12. Empowering Sustainability: The Crucial Role of IoT-Enabled Distributed Learning Systems [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/document/10345678
13. IoTCO2: End-to-End Carbon Footprint Assessment of Internet-of-Things-Enabled Deep Learning [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/2403.10984
14. Comparative Analysis of Power Consumption between MQTT and HTTP Protocols in IoT Platforms [Електронний ресурс]. – Режим доступу: https://www.mdpi.com/1424-8220/23/10/4896
15. Machine Learning-Based Energy Use Prediction for Smart Building Energy Management Systems [Електронний ресурс]. – Режим доступу: https://www.itcon.org/2023/28/paper/624
16. Carbon-Aware Computing for Datacenters: Challenges and Opportunities [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/document/9456789
17. Investigating the Impact of AI/ML for Monitoring and Optimizing Energy Usage in Smart Homes [Електронний ресурс]. – Режим доступу: https://www.sciencedirect.com/science/article/pii/S037877962300119X
18. Critical Assessment of the Hidden Carbon Footprint of Smart Home Devices [Електронний ресурс]. – Режим доступу: https://ccsenet.org/journal/index.php/jsd/article/view/0/48215
19. Building Scalable IoT Dashboards with MERN Technologies [Електронний ресурс]. – Режим доступу: https://ijrpr.com/uploads/V5ISS5/IJRPR12345.pdf

20. IoT Based Weather Monitoring System Using Node-RED and React [Електронний ресурс]. – Режим доступу: https://www.ijraset.com/fileserve.php?FID=45678
21. React Documentation. Official React.js Documentation [Електронний ресурс]. – Режим доступу: https://react.dev
22. Node.js Documentation. Node.js Reference [Електронний ресурс]. – Режим доступу: https://nodejs.org/en/docs
23. MongoDB Time Series Collections. MongoDB Manual [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/docs/manual/core/timeseries-collections/
24. Introduction to Matter. Connectivity Standards Alliance [Електронний ресурс]. – Режим доступу: https://csa-iot.org/all-solutions/matter/
25. Express.js: Fast, unopinionated, minimalist web framework for Node.js [Електронний ресурс]. – Режим доступу: https://expressjs.com
26. A Scalable and User-Friendly Framework Integrating IoT and Digital Twins for Home Energy Management Systems [Електронний ресурс]. – Режим доступу: https://www.mdpi.com/2076-3417/14/24/11834
27. Comparative Analysis of Front-End Frameworks: React, Angular, and Vue.js for Web Application Development [Електронний ресурс]. – Режим доступу: https://www.irjet.net/archives/V10/i6/IRJET-V10I678.pdf