

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
ІНТЕЛЕКТУАЛЬНИХ МЕТОДІВ КОНТРОЛЮ ЕНЕРГОСПОЖИВАННЯ
RESEARCH AND SOFTWARE IMPLEMENTATION OF
INTELLIGENT METHODS FOR ENERGY CONSUMPTION CONTROL

Виконав: студент 2 курсу, групи КНТ-121м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Горіченко Ю. Є.

(прізвище та ініціали)

Керівник Пархоменко А. В.

(прізвище та ініціали)

Рецензент Зеленьова І. Я.

(прізвище та ініціали)

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

Факультет комп'ютерних наук та технологій
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С. О. Субботін
«_____» _____ 2022 року

ЗАВДАННЯ

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Горіченка Юлія Євгенівна

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи): Дослідження та програмна реалізація інтелектуальних методів контролю енергоспоживання. Research and Software Implementation of Intelligent Methods for Energy Consumption Control

керівник проєкту (роботи): Пархоменко Анжеліка Володимирівна, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «04» листопада 2022 р. № 363

2. Строк подання студентом проєкту (роботи): 1 грудня 2022 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): 1 Аналіз предметної області. 2 Огляд існуючих інтелектуальних методів в сфері ефективного енергоспоживання. 3 Проєктування програмного забезпечення. 4 Конструювання програмного забезпечення. 5 Тестування розробленої підсистеми. 6 Керівництво програміста. 7 Керівництво користувача.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): Слайди презентації

6. Консультанти розділів проєкту (роботи):

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	Пархоменко А. В., доцент		
Нормоконтроль	Камінська Ж. К., асистент		

7. Дата видачі завдання «05» вересня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання
2	Аналіз предметної області	2-3 тиждень	Розділ 1
3	Огляд існуючих інтелектуальних методів в сфері ефективного енергоспоживання	4-5 тиждень	Розділ 2
4	Проектування програмного забезпечення	6 тиждень	Розділ 3
5	Конструювання програмного забезпечення	7-9 тижні	Розділ 4
6	Тестування розробленої підсистеми	10 тиждень	Розділ 5
7	Розробка програмної документації	11 тиждень	Розділ 6, 7
8	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12-13 тиждень	ПЗ, Додатки
9	Захист роботи	14 тиждень	

Студент

Горіченко Ю. Є.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

Пархоменко А. В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
177 с., 95 рис., 54 табл., 2 дод., 92 джерела.

СИСТЕМА ДОМАШНЬОЇ АВТОМАТИЗАЦІЇ, ШТУЧНІ НЕЙРОННІ
МЕРЕЖІ, ПРОГНОЗУВАННЯ, ОПТИМІЗАЦІЯ РОЗКЛАДУ,
ЕНЕРГОСПОЖИВАННЯ.

Об'єкт дослідження – процес контролю енергоспоживання в системі домашньої автоматизації.

Предмет дослідження – методи та засоби контролю енергоспоживання в системі домашньої автоматизації.

Мета роботи – дослідження та практична реалізація інтелектуальних методів контролю енергоспоживання для підвищення ефективності користування електроприладами та економії електроенергії в системі домашньої автоматизації.

Матеріали, методи та технічні засоби: платформа OpenHAB, мова програмування Java, вебфреймворк Spring, мова програмування Python, бібліотека Keras, система керування базами даних MongoDB.

Результати. Запропоновано комплексний інтелектуальний метод контролю енергоспоживання на основі прогнозування генерації електроенергії з максимальним використанням альтернативних джерел та оптимізації розкладу використання електроприладів.

Висновки. Розроблено підсистему інтелектуальної підтримки для системи домашньої автоматизації, яка виконує контроль генерації та споживання електроенергії для підвищення ефективності користування побутовими приладами та економії ресурсів.

Галузь використання – системи домашньої автоматизації на платформі OpenHAB.

ABSTRACT

Explanatory note to the final qualifying work of the master: 177 pages, 95 figures, 54 tables, 2 appendixes, 92 sources.

HOME AUTOMATION SYSTEM, ARTIFICIAL NEURAL NETWORKS, PREDICTION, SCHEDULE OPTIMIZATION, POWER CONSUMPTION.

The object of the study is a monitoring energy consumption process for a home automation system.

The subject of the study are the methods and tools of monitoring energy consumption for a home automation system.

The purpose of the work is a research and implementation of intelligent methods for controlling energy consumption in order to improve the efficiency of using household appliances and save electricity for a home automation system.

Materials, methods and tools: OpenHAB platform, Java programming language, Spring Web Framework, Python programming language, Keras library, MongoDB database management system.

Results. A hybrid intelligent method for monitoring energy consumption based on forecasting electricity generation with maximum use of alternative energy sources and optimal scheduling of household appliances was proposed.

Conclusions. An intelligent support subsystem for a home automation system has been developed, which takes control of electricity generation and consumption in order to improve the efficiency of using household appliances and save resources.

The field of use are the home automation systems using OpenHAB.

ЗМІСТ

	С.
ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК	9
ВСТУП	10
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	11
1.1 Концепція системи домашньої автоматизації	11
1.2 Дослідження архітектури реальної системи домашньої автоматизації	12
1.3 Дослідження енергоспоживачів системи домашньої автоматизації	15
1.4 Дослідження особливостей генерації електроенергії в системі домашньої автоматизації	21
1.5 Висновки до розділу	26
2 ОГЛЯД ІСНУЮЧИХ ІНТЕЛЕКТУАЛЬНИХ МЕТОДІВ В СФЕРІ ЕФЕКТИВНОГО ЕНЕРГОСПОЖИВАННЯ	28
2.1 Використання методу рою часток	28
2.2 Використання генетичного алгоритму	32
2.3 Використання моделі ARIMA	34
2.4 Використання штучних нейронних мереж	35
2.5 Висновки до розділу	40
3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	43
3.1 Аналіз вимог	43
3.2 Вдосконалення архітектури системи домашньої автоматизації	47
3.3 Реалізація комплексного інтелектуального методу прогнозування та оптимізації	49
3.4 Вибір метеосервісу	51
3.5 Вибір системи керування базами даних	53
3.6 Вибір інструментарію розробки	57
3.6.1 Вибір вебфреймворку для розробки підсистеми	57
3.6.2 Вибір мови програмування для машинного навчання	62
3.6.3 Вибір бібліотеки для машинного навчання	64
3.7 Висновки до розділу	66

4 КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	69
4.1 Основні рішення щодо модульної побудови підсистеми	69
4.2 Основні рішення щодо представлення даних	75
4.3 Основні розроблені алгоритми	79
4.3.1 Алгоритм оптимізації розкладу роботи електроприладів	79
4.3.2 Алгоритм моніторингу показників енергоспоживання	83
4.4 Опис особливостей програмної реалізації підсистеми	85
4.5 Реалізація нейромережевої моделі	100
4.5.1 Набір навчальних даних та їх структура	100
4.5.2 Порівняння та вибір найкращої нейромережевої моделі.....	105
4.6 Особливості реалізації графічного інтерфейсу.....	113
4.7 Висновки до розділу	115
5 ТЕСТУВАННЯ РОЗРОБЛЕНОЇ ПІДСИСТЕМИ	118
5.1 Функціональне тестування.....	118
5.2 Кросплатформенне тестування.....	124
5.3 Висновки до розділу	126
6 КЕРІВНИЦТВО ПРОГРАМІСТА	127
6.1 Призначення та умови застосування підсистеми	127
6.2 Характеристики підсистеми.....	128
6.3 Інсталяція та налаштування підсистеми.....	129
6.4 Звертання до підсистеми	131
6.5 Вхідні та вихідні дані.....	132
6.6 Повідомлення	132
7 КЕРІВНИЦТВО КОРИСТУВАЧА	135
7.1 Призначення підсистеми	135
7.2 Умови виконання підсистеми	135
7.3 Виконання підсистеми.....	136
7.3.1 Сторінка «Календар»	138
7.3.2 Сторінка «Планувальник розкладу»	141
7.3.3 Сторінка «Побутові прилади»	144

7.3.4 Сторінка «Про підсистему»	145
7.4 Повідомлення	146
ВИСНОВКИ.....	150
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	152
ДОДАТОК А ТЕКСТ ПРОГРАМИ	161
ДОДАТОК Б СЛАЙДИ ПРЕЗЕНТАЦІЇ.....	169

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AGPL	– Affero General Public License;
API	– Application Programming Interface;
BSD	– Berkeley Software Distribution;
CSV	– Comma-separated values;
FLOSS	– Free/Libre Open Source software;
GPL	– General Public License;
HDMI	– High Definition Multimedia Interface;
HTML	– HyperText Markup Language;
JRE	– Java Runtime Environment;
JSON	– JavaScript Object Notation;
JSP	– Java Server Pages;
LAN	– Local Area Network;
MVC	– Model-View-Controller;
OpenHAB	– Open Home Automation Bus;
PD	– public domain;
RMSE	– Root Mean Square Error;
SQL	– Structured query language;
USB	– Universal Serial Bus;
VGA	– Video Graphics Array;
ДКЧП	– довга короткочасна пам'ять;
РНМ	– рекурентна нейронна мережа;
СДА	– система домашньої автоматизації;
СКБД	– система керування базами даних.

ВСТУП

На сьогоднішній день існує багато різних платформ домашньої автоматизації (СДА), наприклад: OpenHAB, HomeAssistant, ioBroker тощо. Здебільшого їх функціонал спрямований на моніторинг та управління різноманітними процесами в будинках та квартирах. Домогосподарство, оснащене такою системою, пропонує більше комфорту, гнучкості та безпеки, але при цьому СДА не надає інтелектуальної підтримки користувачам, зокрема, з точки зору організації економного споживання електроенергії, що генерується альтернативними джерелами електроенергії (сонячні та вітрові електростанції) на основі оптимального використання різноманітних побутових приладів.

За останні роки спостерігається активний розвиток інтелектуальних технологій та штучних нейронних мереж, які активно впроваджуються в домогосподарства, що дозволяє вивести стандартну СДА на новий рівень. Використання інтелектуальних методів дозволяє вивчати звички мешканців будинку, передбачати їх поведінку та створювати оптимальні умови для життя. Однією з найважливіших сфер використання інтелектуальних технологій є економія електроенергії, що дозволить здобути незалежність від електромережі та заощадити кошти за комунальні послуги, використовуючи альтернативні джерела енергії.

Таким чином, для вирішення поставленої проблеми необхідно розробити механізм для управління оптимальним споживанням електроенергії з урахуванням інтересів та звичок мешканців будинку, що дозволить зробити наявну СДА дійсно «розумною».

Метою роботи є дослідження та практична реалізація інтелектуальних методів контролю енергоспоживання для підвищення ефективності користування електроприладами та економії електроенергії в системі домашньої автоматизації, що на сьогодні є значущою проблемою особливо в умовах нестабільної економічної та політичної ситуації в країні.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Концепція системи домашньої автоматизації

СДА являє собою централізовану систему, що поєднує в собі різні технології щодо моніторингу, а також управління користування приладами та датчиками, що підключені до неї. Будинок, оснащений такою системою, пропонує набагато більше комфорту, гнучкості та безпеки, але, що найбільш важливо, допомагає автоматизувати рутинні задачі. Так, СДА може виконувати безліч завдань, таких як управління шторами та вікнами без втручання людини, відкриття та закриття гаражних воріт, управління кліматом всередині будинку, забезпечення відповідного освітлення в кожній кімнаті, запуск розбризкувача, коли ґрунт занадто сухий тощо [1].

В останні роки спостерігається значне зростання популярності домашньої автоматизації у зв'язку з появою Інтернету речей. Різні пристрої та побутова техніка в будинку, такі як освітлення, кондиціонери, системи безпеки та розваг тепер можуть підключатися до мережі Інтернет, щоб ними можна було керувати віддалено (рис. 1.1). Це призвело до впровадження та активного використання різних інтелектуальних методів для забезпечення енергоефективності та зменшення витрат на комунальні послуги [1].

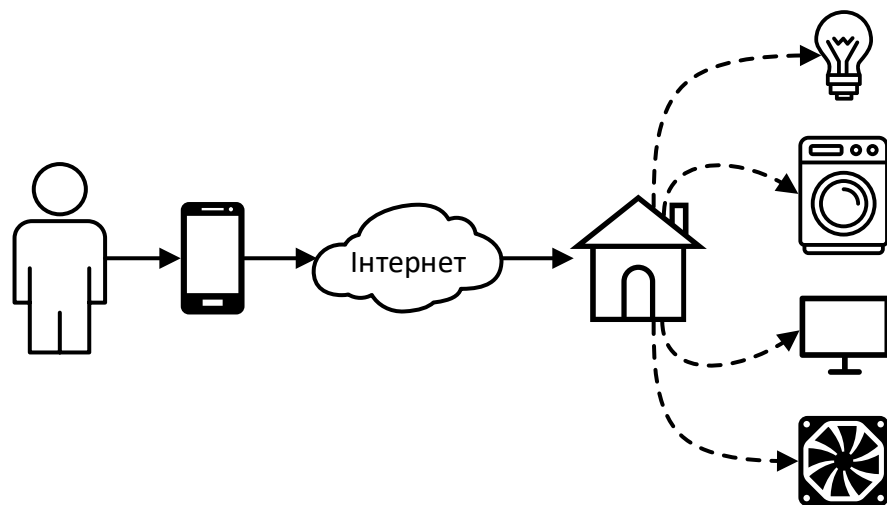


Рисунок 1.1 – Концепція системи домашньої автоматизації

1.2 Дослідження архітектури реальної системи домашньої автоматизації

Для аналізу накопичених даних з реальної СДА було використано приватний будинок, у якому проживає чотири людини. Архітектуру СДА для досліджуваного будинку зображено на рисунку 1.2.

Центром досліджуваної системи виступає міні-комп'ютер J1900. Це повністю безшумний та компактний системний блок невеликого розміру, який працює на базі чотирьох ядерного процесора Intel Celeron J1900 та має 4 ГБ оперативної пам'яті. Для підключення периферії він має наступні порти: Universal Serial Bus (USB) версії 2.0 та 3.0, аудіовихід, High Definition Multimedia Interface (HDMI) та Video Graphics Array (VGA) для виводу зображення на екран, а також Local Area Network (LAN) для підключення до локальної мережі або до мережі Інтернет. Тобто це повноцінний комп'ютер, який можна використовувати для різноманітних задач [2].

В ролі системи для створення домашньої автоматизації виступає програмне забезпечення Open Home Automation Bus (OpenHAB), що встановлене на міні-комп'ютері J1900.

OpenHAB являє собою платформу, яка дозволяє об'єднати різні технології з системами домашньої автоматизації, надаючи єдине рішення. Таким чином, можна підключити різні пристрої, датчики, актуатори до однієї системи та керувати ними віддалено, а також переглядати статистику різних показників. Крім того, OpenHAB є модульним програмним забезпеченням, яке масштабується за допомогою різних аддонів, що надають платформі широкий спектр можливостей. До них відносяться різні користувальницькі інтерфейси та бібліотеки для взаємодії з великою кількістю пристроїв, що підключаються. До того ж OpenHAB написано на мові програмування Java та це означає, що такий програмний засіб не залежить від операційної системи та його можна встановлювати на різні популярні платформи, такі як Linux, Windows та macOS [3].

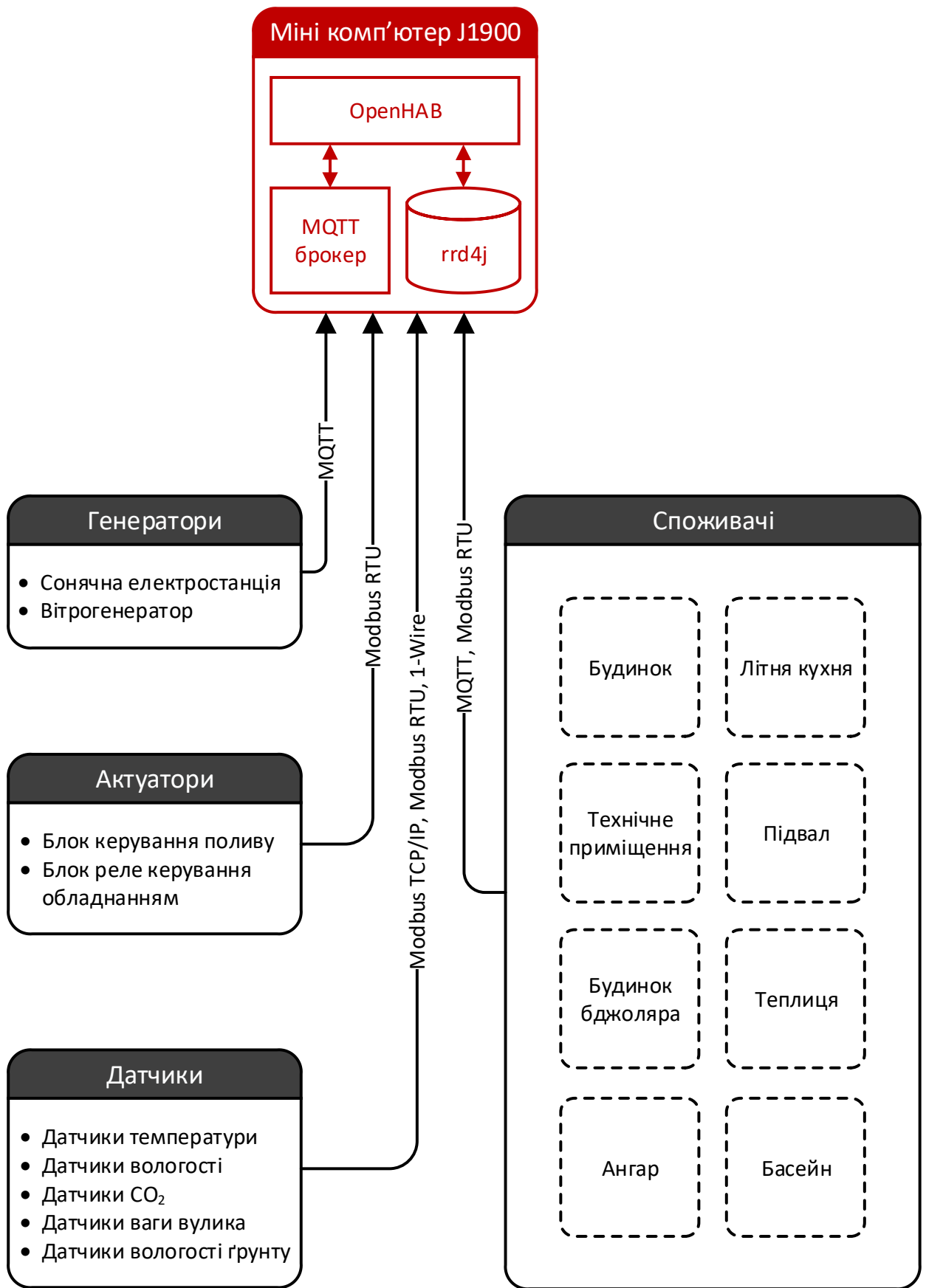


Рисунок 1.2 – Архітектура досліджуваної СДА

База даних rrd4j – це сховище даних, яке використовується за замовчуванням для платформи OpenHAB. На відміну від традиційних реляційних баз даних, rrd4j не збільшується у розмірах, бо воно має фіксований розмір. Це досягається за рахунок збереження обмеженої кількості точок даних та у подальшому їхнього стиснення. Це означає, що чим старіші дані, тим менше доступних значень, бо зберігається тільки середнє значення за деякий період [4].

Message Queue Telemetry Transport (MQTT) – це протокол обміну повідомленнями, який широко використовується для зв'язку пристроїв Інтернету речей. MQTT надає мережевим клієнтам з обмеженими ресурсами простий спосіб поширення телеметричної інформації в середовищах з низькою пропускнуою здатністю, що робить цей протокол ідеальним для використання в СДА [5], [6].

В архітектурі MQTT існує два типи систем: клієнти та брокери.

MQTT брокер – це сервер, з яким взаємодіють клієнти. Брокер отримує повідомлення від підключених до нього пристроїв та відправляє їх іншим пристроям або зберігає отриману інформацію у базі даних. Таким чином, клієнти не спілкуються безпосередньо один з одним, а підключаються до брокера, який оброблює всі запити [7].

Modbus – це протокол передачі даних, заснований на моделі «запит-відповідь». Він використовується для передачі інформації між пристроями, які підключені до мережі по послідовних лініях або Ethernet. Modbus RTU є найбільш поширеною версією цього протоколу та використовується для послідовних підключень. Modbus TCP/IP використовує протокол TCP/IP на транспортному рівні для передачі інформації по мережі Ethernet. Modbus RTU дозволяє простіше налаштувати послідовні з'єднання на відміну від Modbus TCP/IP, де для підключення по Ethernet необхідно встановити мережеву карту та деяке програмне забезпечення, а також налаштувати IP-адреси [8].

1-Wire – це шина передачі даних пристроїв, яка забезпечує передачу інформації та сигналів на низькій швидкості, а також живлення по одному провіднику. Зазвичай 1-Wire використовується для зв'язку з простими датчиками, такими як цифрові термометри або метеорологічні прилади [9].

У системі присутні два види пристроїв: актуатори та датчики.

Актуатор – це такий прилад, який приймає команди від OpenHAB та перетворює їх у фізичні події [10]. У даній системі блок керування обладнанням та блок поливу виступають актуаторами, що підключені до OpenHAB за допомогою протоколу Modbus RTU.

Датчик – це пристрій, який приймає дані з навколишнього середовища та передає їх у платформу OpenHAB для подальшого аналізу або збереження у базу даних, використовуючи наступні протоколи: Modbus TCP/IP, Modbus RTU, 1-Wire [10]. До системи підключені різні види датчиків, а саме: температури, вологості повітря, вуглекислого газу, ваги та вологості ґрунту. Отримані дані використовуються платформою OpenHAB для запуску, розроблених мешканцем, сценаріїв автоматизації.

1.3 Дослідження енергоспоживачів системи домашньої автоматизації

Досліджувана СДА поділяється на кілька основних зон, кожна з яких має власні споживачі електроенергії:

- будинок;
- літня кухня;
- технічне приміщення;
- підвал;
- будинок бджоляра;
- теплиця;
- ангар;
- басейн.

Здебільшого всі зони контролюються платформою OpenHAB, що дозволяє легко та зручно управляти електроприладами, а також переглядати показники різних датчиків, використовуючи графічний інтерфейс.

На рисунках 1.3 – 1.7 наведено інтерфейси для управління літньою кухнею, теплицею, басейном, підвалом, технічним приміщенням та поливом.

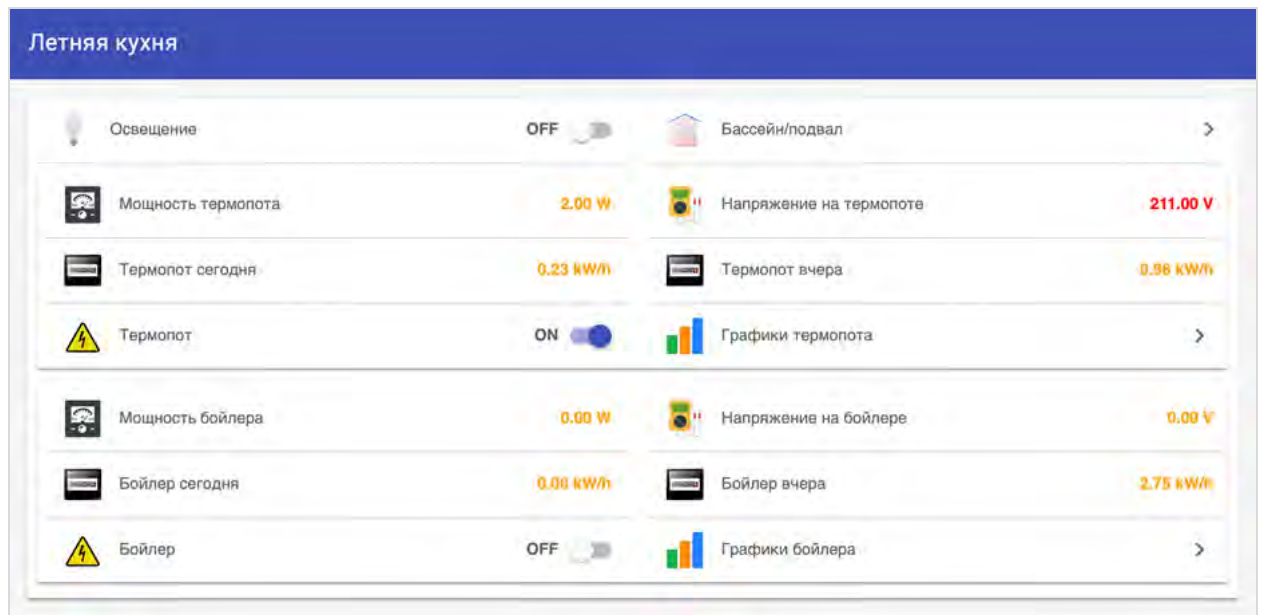


Рисунок 1.3 – Інтерфейс управління для літньої кухні

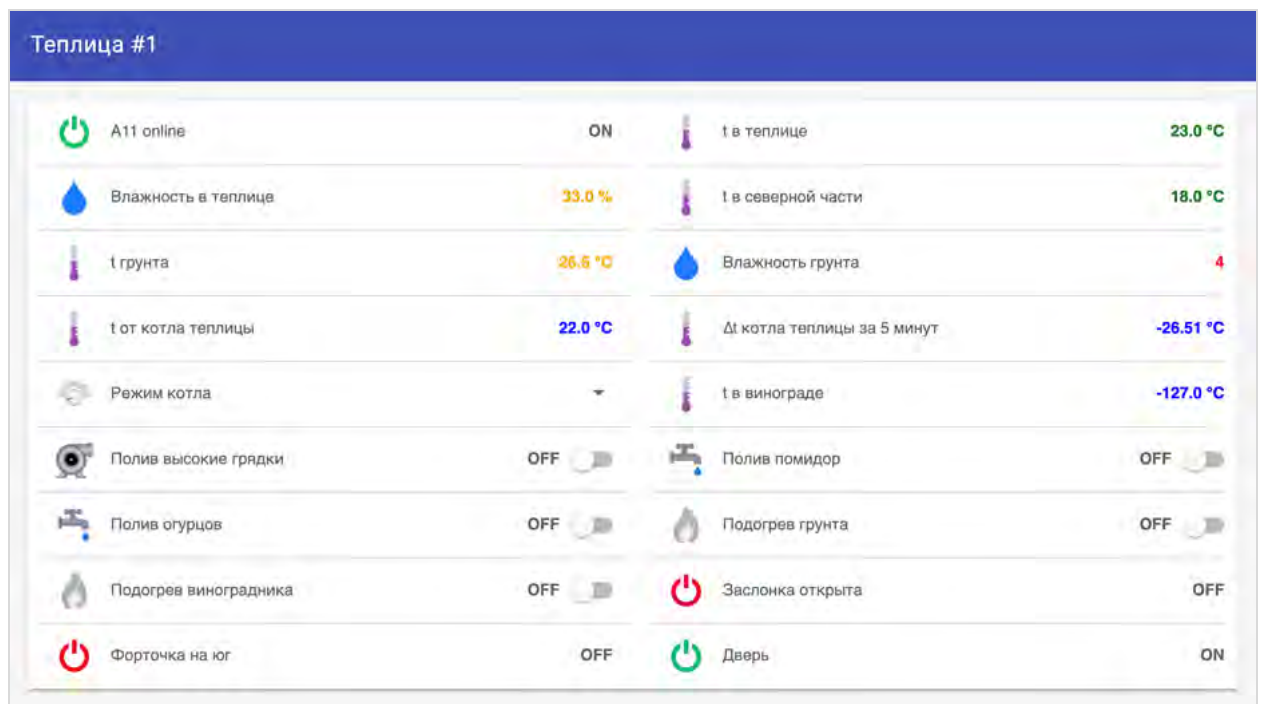


Рисунок 1.4 – Інтерфейс управління для теплиці

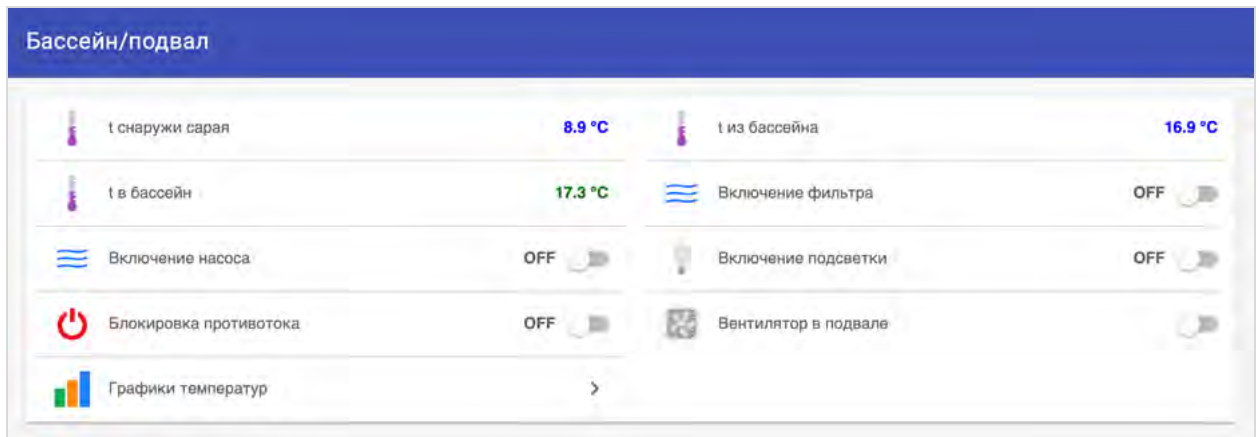


Рисунок 1.5 – Интерфейс управління для басейну та підвалу

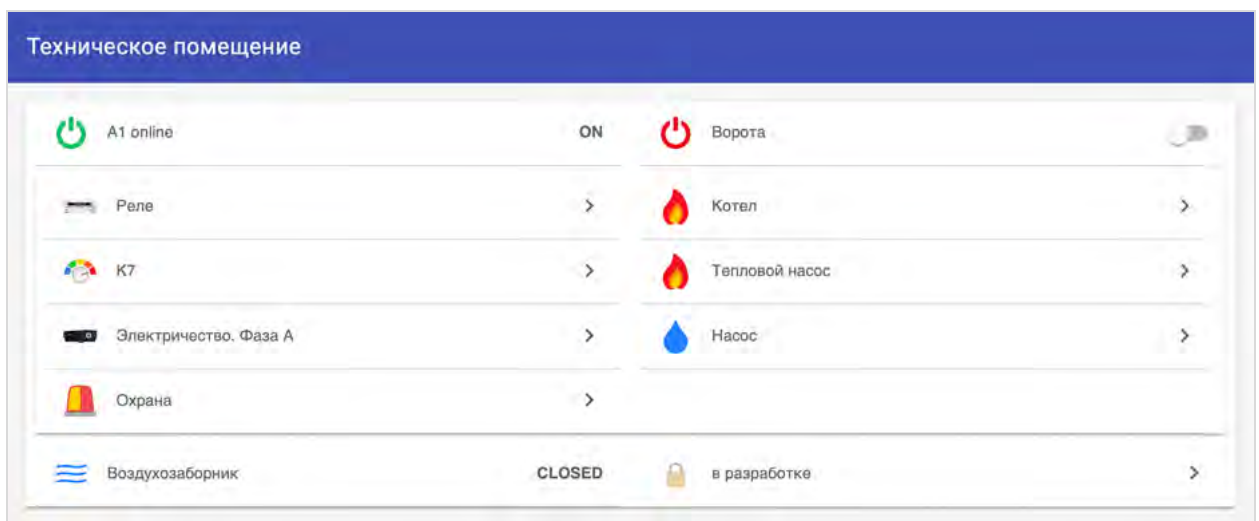


Рисунок 1.6 – Интерфейс управління для технічного приміщення

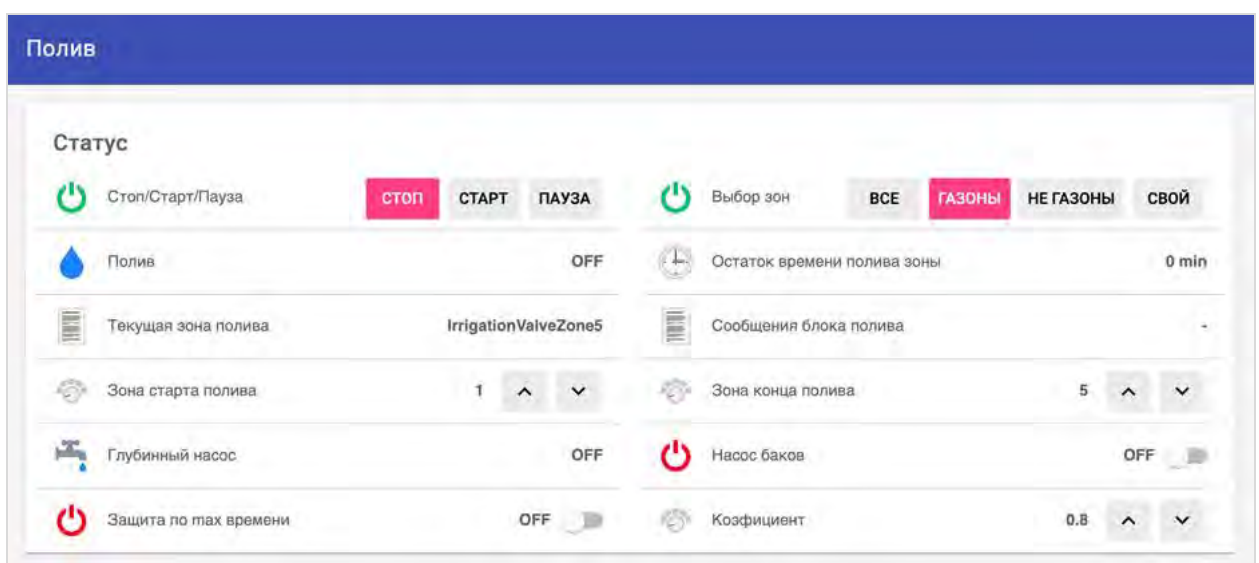


Рисунок 1.7 – Интерфейс управління поливом

На рисунках 1.8 – 1.9 зображено інтерфейси OpenHAB для управління клапанами поливу та будинком бджоляра.

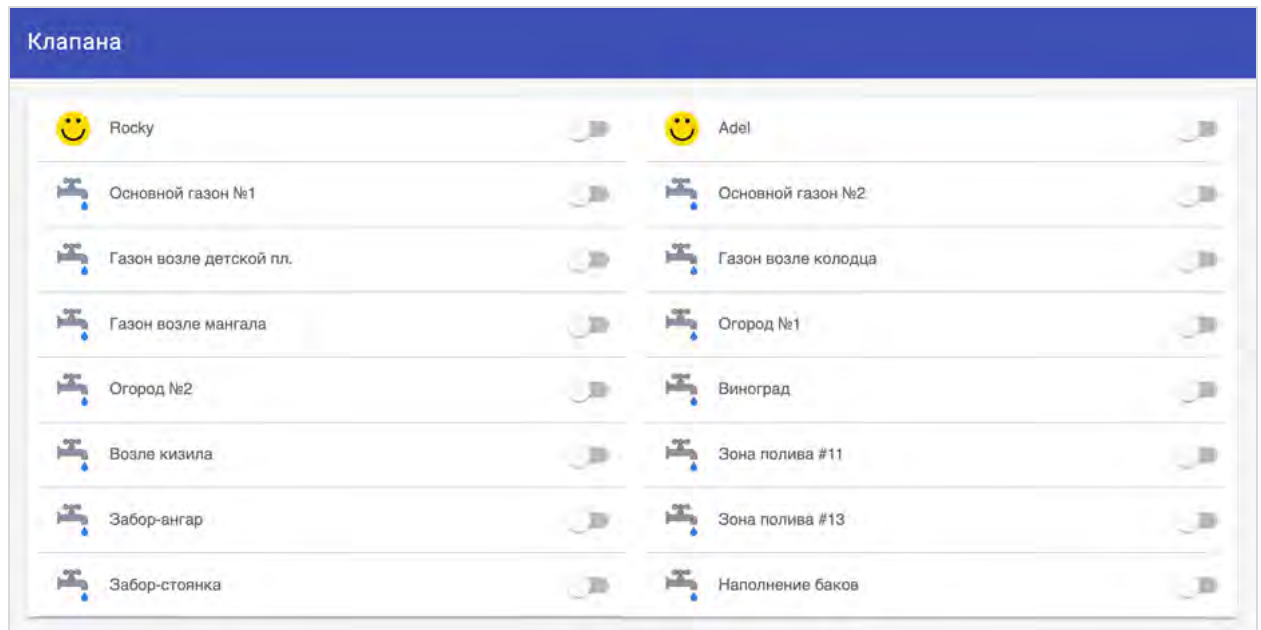


Рисунок 1.8 – Интерфейс управління клапанами поливу

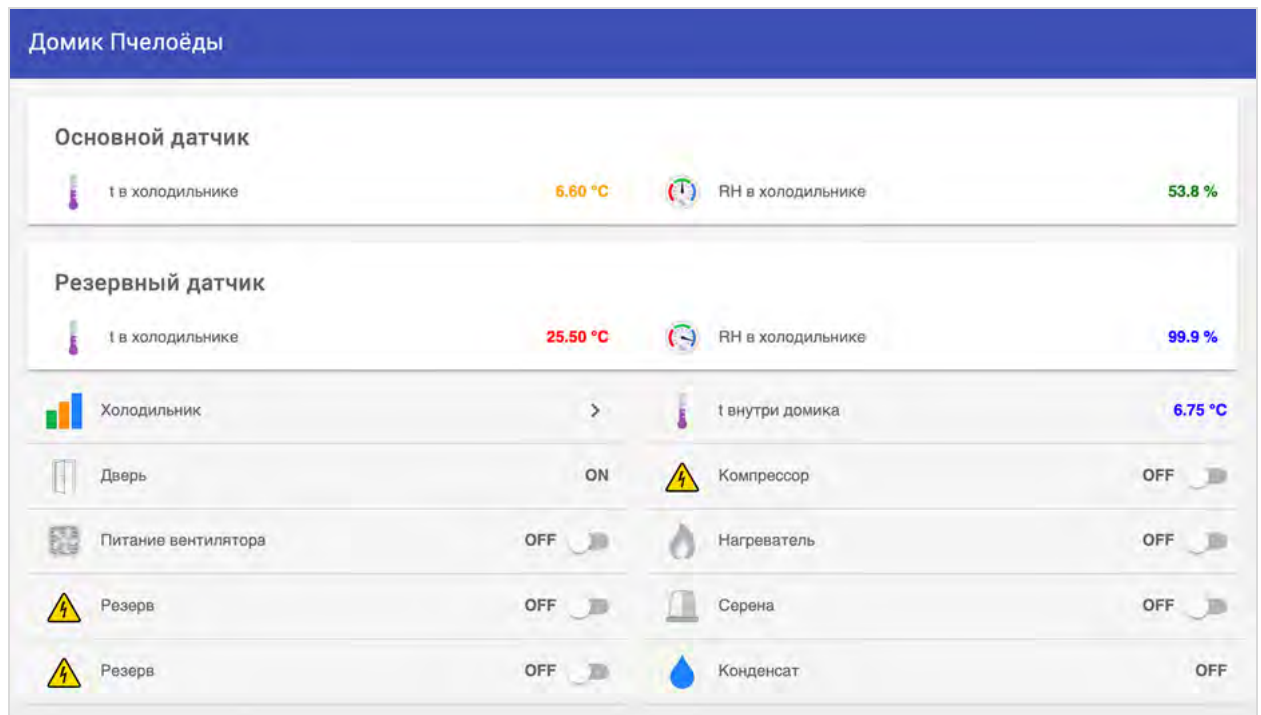


Рисунок 1.9 – Интерфейс управління для будинка бджоляра

Список основних споживачів, що використовуються, наведено у таблиці 1.1 відповідно до зон досліджуваної СДА.

Таблиця 1.1 – Основні споживачі відповідно до зон у СДА

Назва зони	Споживач	Можливість керування платформою OpenHAB
Будинок	Електроплита та духовка	—
	Мультиварка	—
	Піролізний котел	—
	Пральна машина	+
	Праска	—
	Рекуператор	+
	Термопот	+
	Циркуляційний насос опалення	+
Літня кухня	Бойлер	+
	Електроплита	—
	Термопот	+
Технічне приміщення	Газонокосарка	—
	Електропила	—
	Тример	—
Підвал	Вентилятор	+
	Глибинний насос	+
	Насос поливу з баків	—
Будинок бджоляра	Кондиціонер / Холодильник	+
Теплиця	Полив ґрунту	+
	Підігрів повітря	+
	Підігрів ґрунту	+
Ангар	Вентилятор печі	—
	Кондиціонер	—
	Нагрівач повітря	—
	Осушувач повітря	—
Басейн	Геліонасос	+
	Насос фільтра	+
	Підсвітка	+

Список приладів, які мають найбільше енергоспоживання, їхня номінальна потужність, а також можливість збереження статистики користування приладом у базу даних наведено у таблицю 1.2.

Таблиця 1.2 – Найпотужніші споживачі досліджуваної СДА

Споживач	Номінальна потужність	Збереження статистики використання
Електроплита та духовка	4 кВт	—
Кондиціонер (ангар)	3,5 кВт	—
Праска	3 кВт	—
Газонокосарка	2,5 кВт	—
Електропила	2,5 кВт	—
Тример	2,5 кВт	—
Пральна машина	2 кВт	+
Глибинний насос	2 кВт	+
Електроплита	2 кВт	—
Нагрівач повітря (ангар)	2 кВт	—
Бойлер	1,5 кВт	+
Підігрів ґрунту (теплиця)	1,5 кВт	+
Підігрів повітря (теплиця)	1,5 кВт	+
Мультиварка	1 кВт	—
Осушувач повітря (ангар)	1 кВт	—
Кондиціонер (будинок бджоляра)	750 Вт	+
Термопот	700 Вт	+
Насос фільтра (басейн)	500 Вт	+
Насос поливу з баків	500 Вт	—
Рекуператор	200 Вт	+
Циркуляційний насос опалення	150 Вт	+
Піролізний котел	150 Вт	—
Вентилятор печі	150 Вт	—
Геліонасос (басейн)	50 Вт	+

1.4 Дослідження особливостей генерації електроенергії в системі домашньої автоматизації

Як альтернативні джерела енергії для будинку використовуються сонячні електростанції та вітрогенератори, що дозволяють заощаджувати великі кошти та не залежати від тарифів на комунальні послуги з електроенергії. Схему підключення СДА до електромережі зображено на рисунку 1.10.

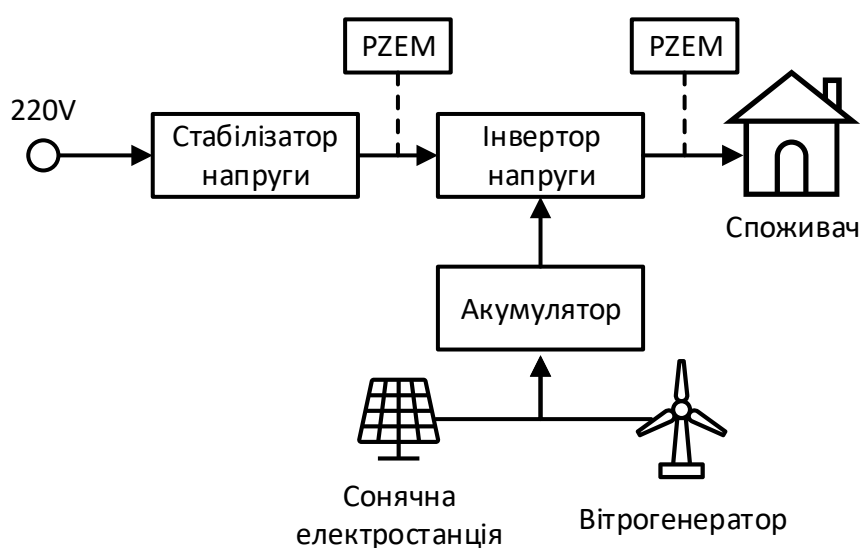


Рисунок 1.10 – Схема підключення альтернативних джерел енергії у СДА

Головним елементом в даній схемі виступає інвертор напруги. Це пристрій для перетворення постійного струму в змінний з можливістю зміни величини напруги. Таким чином, він живить будинок електроенергією, яку отримує з акумулятора, який в свою чергу заряджається від сонячних електростанцій та вітрогенераторів. Якщо альтернативної енергії виробляється недостатньо для роботи всіх необхідних побутових приладів, в такому випадку додатково використовується електроенергія з мережі, яка проходить через стабілізатор напруги до інвертора. Такий підхід дозволяє мати надійне джерело електроенергії для всіх приладів, використовуючи різні джерела живлення [11].

Після стабілізатора напруги та інвертора було підключено модулі PZEM-004T, які призначені для вимірювання вхідної напруги, споживаного навантаження струму та розрахунку споживаної потужності у реальному часі [12]. Ці показники передаються до платформи OpenHAB (рис. 1.11), використовуючи протокол MQTT, та зберігаються у базі даних.

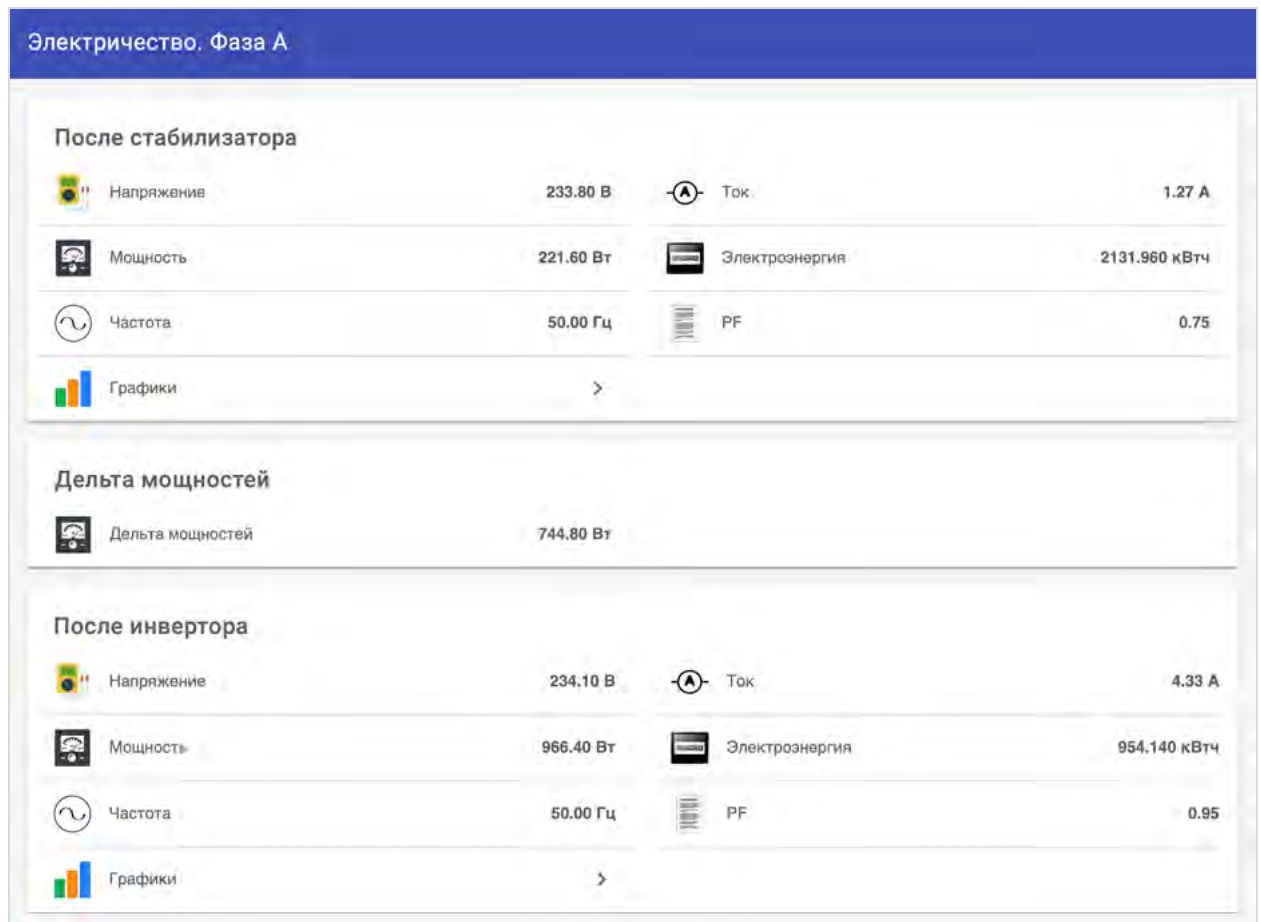


Рисунок 1.11 – Моніторинг показників споживання електроенергії

Згідно схеми підключення СДА до електромережі (рис. 1.10) було проаналізовано особливості під'єднання модулів PZEM-004T. Модуль, що підключено після інвертора, вимірює потужність загального споживання всіх приладів, а PZEM, який встановлено після стабілізатора напруги, вимірює потужність електроенергії, яка береться з мережі у випадку недостатньої генерації альтернативними джерелами.

Обчислення потужності електричного струму P_{Δ} , що генерується сонячними електростанціями та вітрогенераторами виконується за формулою (1.1):

$$P_{\Delta} = P_{inv} - P_{grid} \quad (1.1)$$

де P_{inv} – потужність електричного струму після інвертора;

P_{grid} – потужність електричного струму після стабілізатора.

Графіки енергоспоживання за тиждень для досліджуваної СДА наведено на рисунках 1.12 – 1.13.



Рисунок 1.12 – Графік споживання електроенергії за тиждень

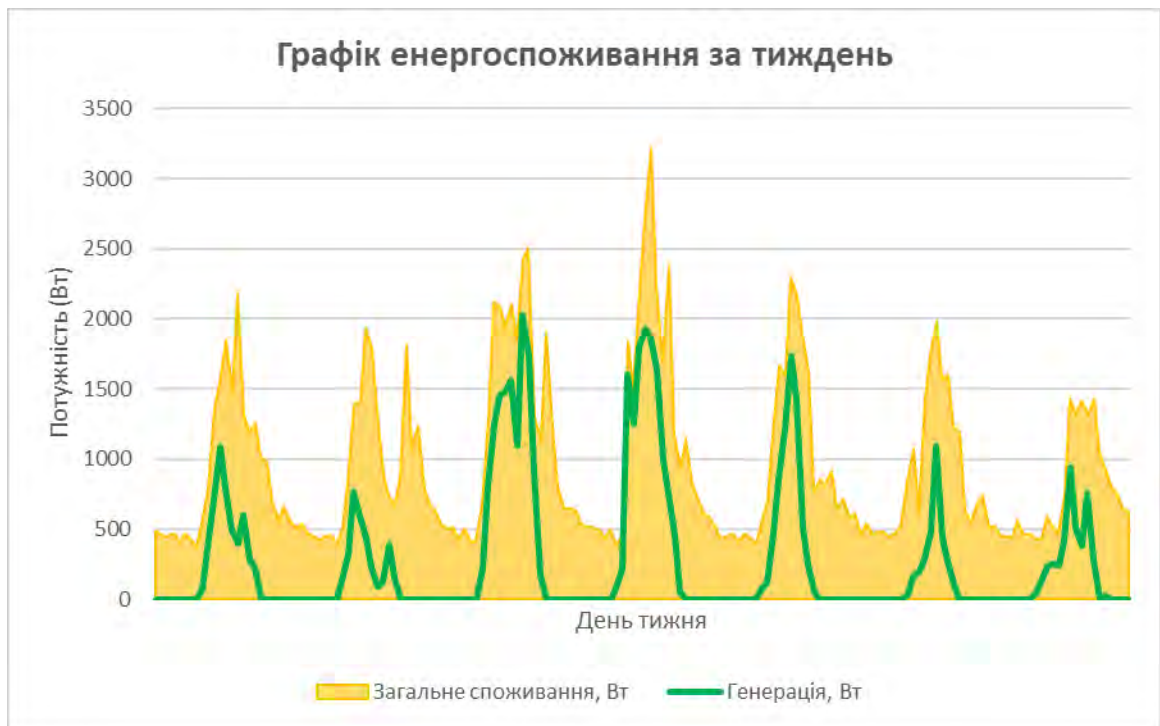


Рисунок 1.13 – Графік споживання та генерації електроенергії за тиждень

Фрагмент даних енергоспоживання, що отримуються за допомогою модулів PZEM-004T та які зберігаються у базі даних наведено у таблиці 1.3. Споживання з мережі та загальне споживання – це дані з модулів, які встановлені до та після інвертора відповідно. Дані про генерацію електроенергії було розраховано, використовуючи формулу 1.1.

Таблиця 1.3 – Дані енергоспоживання з модулів PZEM

Дата та час	Загальне споживання (Вт)	Споживання з електромережі (Вт)	Генерація (Вт)
1	2	3	4
2022-09-12 00:00:00	509.5991597	552.3952569	-42.79609725
2022-09-12 01:00:00	506.4512448	547.2560636	-40.80481881
2022-09-12 02:00:00	477.1804979	518.7225296	-41.54203172
2022-09-12 03:00:00	498.5725	539.1861111	-40.61361111
2022-09-12 04:00:00	490.8884937	533.1053465	-42.21685281
2022-09-12 05:00:00	452.9835417	492.0138277	-39.03028599
2022-09-12 06:00:00	454.9945263	496.015748	-41.02122172
2022-09-12 07:00:00	429.494617	466.0079051	-36.51328816
2022-09-12 08:00:00	581.4745868	366.0872832	215.3873035
2022-09-12 09:00:00	737.404321	351.1619231	386.2423979

Продовження таблиці 1.3.

1	2	3	4
2022-09-12 10:00:00	571.8670807	353.5831395	218.2839412
2022-09-12 11:00:00	1046.367006	951.2542471	95.11275901
2022-09-12 12:00:00	878.3468172	439.0242775	439.3225398
2022-09-12 13:00:00	868.2793878	419.7423077	448.5370801
2022-09-12 14:00:00	727.1804082	265.8023077	461.3781005
2022-09-12 15:00:00	780.1504065	408.1586408	371.9917657
2022-09-12 16:00:00	731.0350515	473.2773077	257.7577439
2022-09-12 17:00:00	1262.783164	997.9859885	264.7971758
2022-09-12 18:00:00	838.6594705	649.8461538	188.8133166
2022-09-12 19:00:00	857.9873706	839.9630058	18.02436482
2022-09-12 20:00:00	823.1340206	820.2488506	2.885170044
2022-09-12 21:00:00	874.0051867	886.6106796	-12.60549289
2022-09-12 22:00:00	683.5144654	693.1786127	-9.664147308
2022-09-12 23:00:00	699.7863918	714.1015414	-14.31514967

На основі даних з таблиці 1.3 було створено графік енергоспоживання за добу (рис. 1.14), який відображає інформацію про загальне споживання, споживання з мережі, а також генерацію електроенергії.

В період з 00:00 по 7:00 споживання з мережі перевищує загальне споживання, що на перший погляд є нелогічним. Такі показники є результатом того, що вночі генерація повністю відсутня, тому із зовнішньої мережі береться електроенергія, що необхідна для живлення інвертора (рис. 1.10), який споживає приблизно 40 Вт, а модуль PZEM, який встановлено після інвертора не враховує це споживання. Це пояснює той факт, чому генерація, яка була розрахована за формулою 1.1, вночі має від'ємні значення.

В період з 10:00 по 12:00 можна простежити різкий скачок споживання електроенергії з мережі. Така поведінка пояснюється тим, що працював якийсь потужний електроприлад та в цей період сильно зменшилася генерація, наприклад, небо стало хмарним, що вплинуло на роботу сонячних електростанцій. У зв'язку з тим, що генерації не вистачало на всі увімкнені прилади, то використовувалася електроенергія з мережі.

Зменшення споживання із зовнішньої електромережі можна побачити у період з 12:00 по 14:00, тому що значення генерації знову збільшилося та більшість побутових приладів працювало за рахунок альтернативної енергії.

Таким чином, проаналізувавши цей графік, можна побачити, що користувач неефективно використовує побутові прилади, в результаті чого збільшується вартість комунальних послуг на електроенергію.



Рисунок 1.14 – Графік споживання та генерації електроенергії за добу

1.5 Висновки до розділу

Було проведено аналіз архітектури СДА для реального будинку (рис. 1.2), який використовує платформу OpenHAB в ролі програмної частини для створення системи домашньої автоматизації. Також було проведено аналіз енергоспоживачів, які використовуються в даній системі та список найпотужніших побутових приладів зведено у таблицю 1.2. Головною особливістю досліджуваної СДА є те, що до її складу входять сонячні

електростанції та вітрогенератори (рис. 1.10), відповідно, генерація власної електроенергії дозволяє суттєво заощаджувати кошти та менше залежати від тарифів на комунальні послуги за користування електроенергією. Оскільки в системі задіяно велику кількість потужних електроприладів, власної генерації не завжди вистачає, тому система споживає частину електроенергії із зовнішньої мережі (рис. 1.14).

Як показали проведені дослідження, OpenHAB є потужною платформою, яка надає багато можливостей мешканцям будинку, збільшує комфорт та дозволяє автоматизувати рутинні задачі. Тобто така система може тільки інформувати власника про поточні події.

На жаль, за замовчуванням платформа OpenHAB не надає інтелектуальні рішення, які змогли б підказувати користувачу як можна знизити енергоспоживання, що на сьогодні є значущою проблемою особливо в умовах нестабільної економічної та політичної ситуації в країні, тому тема дипломної роботи є актуальною.

Метою роботи є дослідження та практична реалізація інтелектуальних методів контролю енергоспоживання для підвищення ефективності користування електроприладами та економії електроенергії в СДА.

Для досягнення поставленої мети потрібно вирішити наступні задачі:

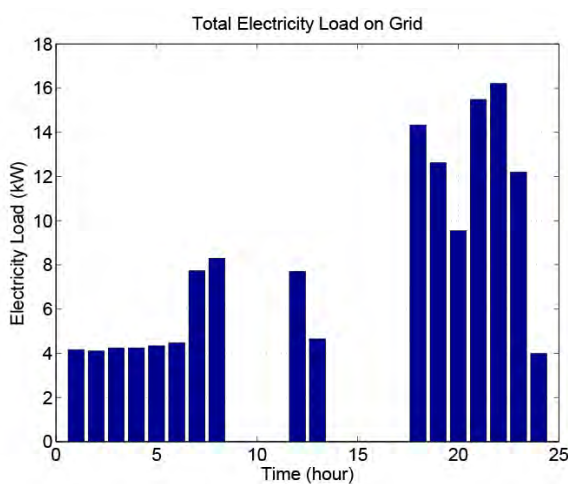
- огляд існуючих інтелектуальних методів в сфері ефективного споживання електроенергії;
- розробка комплексного інтелектуального методу контролю енергоспоживання на основі прогнозування генерації електроенергії з максимальним використанням альтернативних джерел та оптимізації розкладу використання електроприладів;
- проектування та конструювання підсистеми інтелектуальної підтримки;
- інтеграція з платформою OpenHAB та тестування розробленої підсистеми;
- розробка програмної документації.

2 ОГЛЯД ІСНУЮЧИХ ІНТЕЛЕКТУАЛЬНИХ МЕТОДІВ В СФЕРІ ЕФЕКТИВНОГО ЕНЕРГОСПОЖИВАННЯ

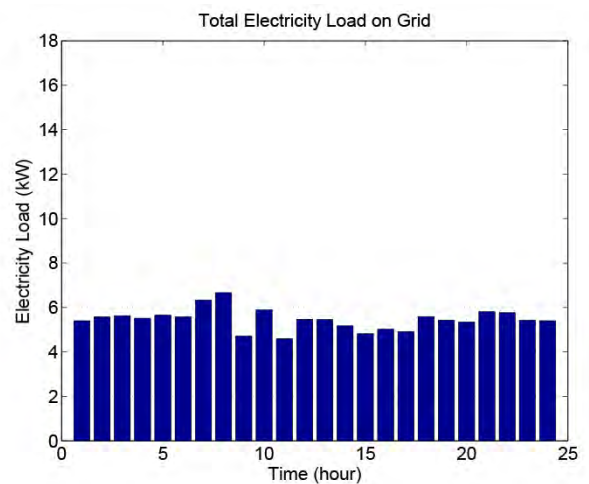
2.1 Використання методу рою часток

У роботах [13]-[17] для зниження витрат на використання електроенергії автори пропонують використовувати метод рою часток. Ідея цього методу полягає в оптимізації розкладу роботи приладів з можливістю зміни часу в СДА. Таким чином, цей метод вирішує проблему мінімізації пікового навантаження в мережі, що дозволяє знизити комунальні витрати на електроенергію.

Використовуючи цей метод, автори у роботі [13] досягли зниження вартості електроенергії на 6,61%. Це було зроблено за рахунок планування роботи приладів в періоди доби, коли електроенергія має низьку ціну (рис. 2.1). Наприклад, після оптимізації посудомийна машина повинна працювати в середині ночі, а пральна машина та сушарка для білизни – у другій половині дня, тому що вони можуть заважати мешканцям будинку, якщо будуть працювати вночі.



а)



б)

Рисунок 2.1 – Використання електроенергії до (а) та після (б) оптимізації роботи приладів [13]

Автори [14] використовували частково інший підхід для мінімізації витрат на електроенергію, хоча ідея залишилась той же самою. Вони запропонували використовувати акумулятор, який повинен заряджатися в періоди доби, коли ціна на електроенергію є низькою і, навпаки, коли ціна висока – використовувати заряд акумулятора для користування електроприладами. Таким чином, застосування методу оптимізації допомогло знизити витрати на електроенергію на 16,49%, що є досить непоганим показником.

Метод рою часток також було використано у роботі [15] для створення плану енергоспоживання, що спрямований на мінімізацію витрат електроенергії. У ході різних експериментів авторам вдалося знизити ціну на електроенергію на 18,43%. Такий результат було досягнуто через планування використання основних побутових приладів, а саме: мікрохвильової печі, індукційної плити, посудомийної машини, рисоварки, пральної машини, калориферу, витяжки та кондиціонера (рис. 2.2). Головним недоліком такого експерименту є те, що рівень комфорту мешканців було значно знижено, тому що використовувати більшу кількість електроприладів було дозволено тільки в певний час, що не завжди буває зручно.

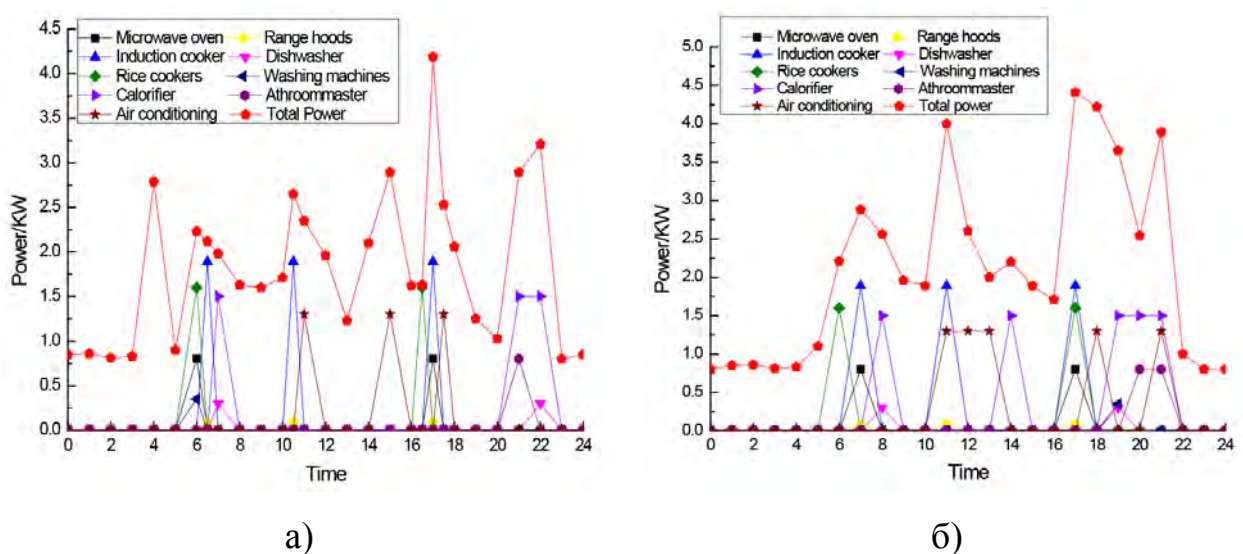


Рисунок 2.2 – План енергоспоживання до (а) та (б) після оптимізації [15]

Використання методу рою часток дозволило авторам [16] досягти певних результатів. Для проведення експериментів було використано три різні вибірки, у кожній з яких розглядається певна область інтелектуальної мережі з різними типами керованих електроприладів:

- житлова область: побутові прилади, які зазвичай знаходяться в будинках мають низьке енергоспоживання (пральна машина, посудомийна машина тощо);

- комерційна область: пристрої в цій зоні здебільшого мають більш високі показники енергоспоживання, ніж в житловому районі (кавоварка, диспенсер для води тощо);

- промислова область: ця зона складається з найменшої кількості керованих приладів з великим енергоспоживанням, ніж пристрої іншої зони (зварювальний апарат, водонагрівач тощо).

Також для кожної години доби було вказано пристрої, які можуть працювати в цей час. Після цього кількість доступних електроприладів оптимізується для того, щоб зменшити максимальне навантаження в конкретну годину. Таким чином, алгоритм забезпечує зміщення використання доступних приладів на інші години, коли ціна за електроенергію є нижчою.

Отримані результати моделювання свідчать, що алгоритм рою часток змінив криву навантаження таким чином, щоб вона максимально наблизилася до цільової кривої (рис. 2.3). Це означає, що пікова потреба для житлового району знизилась з 1363,6 кВт до 1046,5 кВт, що складає 23,25% та вартість комунальних послуг також знизилася на 7,24%. Для комерційної зони піковий попит знизився з 1818,2 кВт до 1497,95 кВт, що становить 17,61%, а витрати на електроенергію також знизилася приблизно на 9,65%. Вартість комунальних послуг для промислової зони знизилася на 18,71%, а пікове навантаження зменшено з 2727,31 кВт до 2273,63 кВт, що є на 16,63% менше.

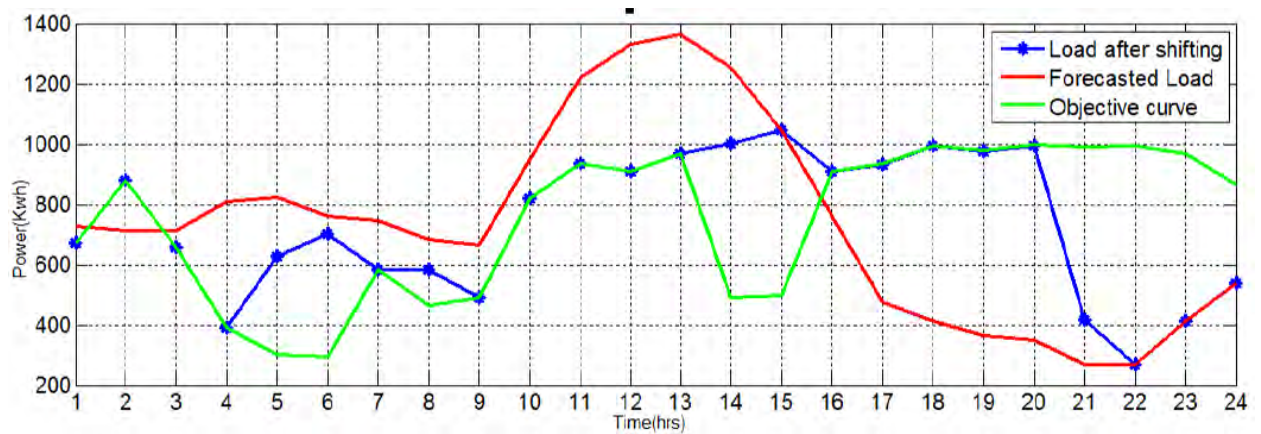


Рисунок 2.3 – Результати оптимізації для житлової області [16]

Аналогічний підхід для зниження витрат на електроенергію було запропоновано авторами у роботі [17]. Усю побутову техніку було поділено на три основні категорії. Перша група включає прилади, час роботи яких не може бути змінено, наприклад, холодильник, який має бути увімкнено впродовж 24 годин. Друга група містить електроприлади, роботу яких можна планувати заздалегідь, але вони мають працювати безперервно протягом деякого часу, наприклад, пральна або посудомийна машина. Третя група охоплює пристрої, які можна вмикати та вимикати в будь-який час, наприклад, кондиціонер.

За допомогою алгоритму рою часток створюється оптимальний розклад користування приладами для другої та третьої групи, враховуючи ціни на електроенергію. Якщо в певний час потужність використовуваних приладів є занадто високою, то користувач отримує повідомлення з проханням припинити або відкласти використання деякої побутової техніки.

Згідно отриманих результатів можна побачити, що авторам вдалося знизити вартість комунальних послуг на електроенергію на 42,5%. Але головним недоліком такого експерименту є те, що не було враховано звички мешканців, тому що відповідно до нового розкладу більшість електроприладів повинні використовуватися вночі (рис. 2.4).

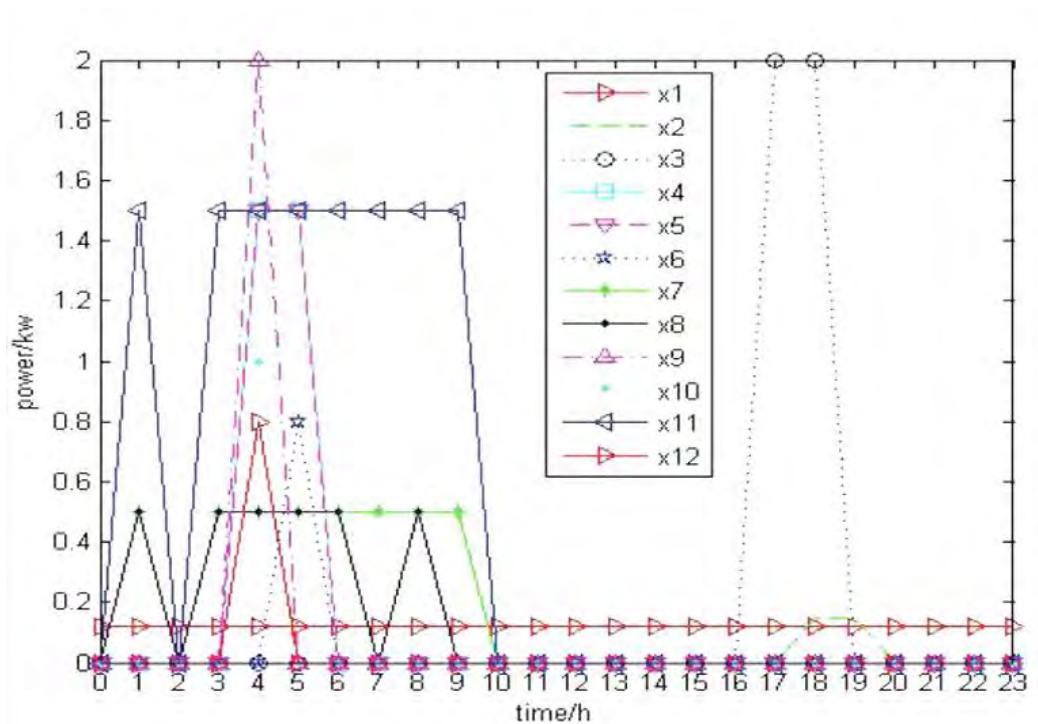


Рисунок 2.4 – Отриманий розклад за допомогою алгоритму рою часток [17]

2.2 Використання генетичного алгоритму

Автори [18]-[20] у своїх роботах пропонують використання генетичного алгоритму для зниження вартості комунальних послуг на електроенергію. Ідея такого підходу аналогічна методу рою часток – зміщення пікового навантаження в непікові години для поліпшення економії енергії в СДА.

Проведені експерименти у [18] показали, що генетичний алгоритм забезпечує оптимальне рішення для планування користування побутовими приладами за рахунок скорочення загальних витрат на споживану енергію та зменшення пікових значень. Оскільки ціна на електроенергію варіюється щогодини, то можна побачити, що з 11 по 14 години ціни на електроенергію є найвищими. У зв'язку з цим, генетичний алгоритм планує навантаження в часових інтервалах з 1 по 10 годину та з 15 по 24, щоб знизити витрати на споживання.

Отримані результати, після застосування генетичного алгоритму, ілюструють значне скорочення рахунків за електроенергію у разі планування без наявності альтернативних джерел, а також з використанням сонячної електростанції (рис. 2.5). Це свідчить про те, що генетичний алгоритм добре працює за будь-яких обставин. Таким чином, вартість електроенергії знизилась на 7,1%, в той час як використання сонячної електростанції дозволило знизити вартість на 55,62%.

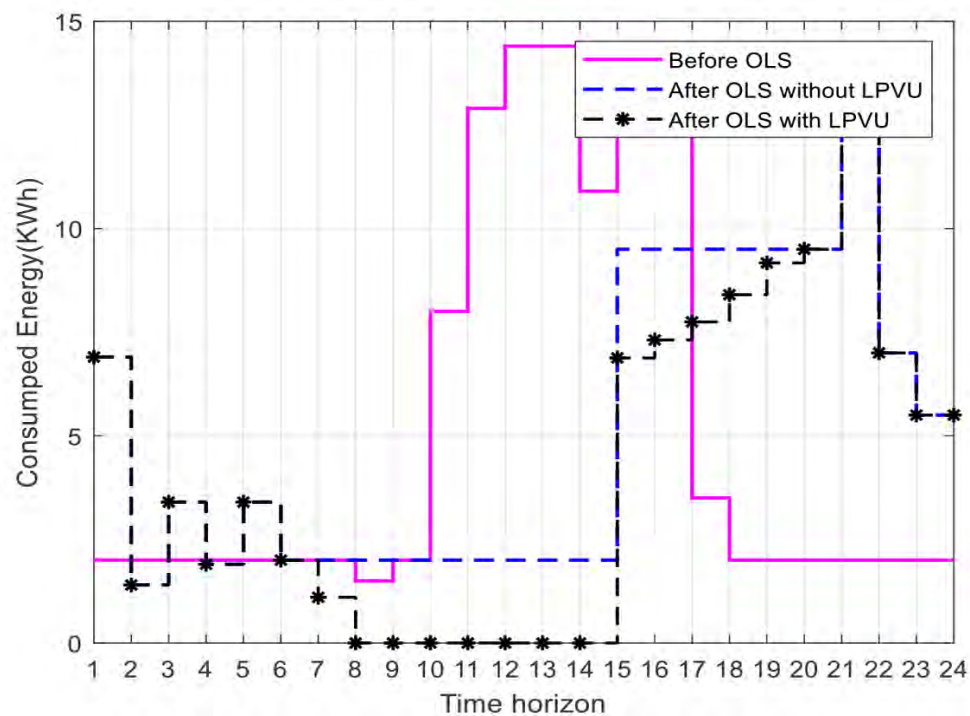


Рисунок 2.5 – Порівняння результатів споживання електроенергії до та після оптимізації [18]

У роботі [19] було продемонстровано використання генетичного алгоритму. Результати відображають, що планування з використанням цього методу дозволило знизити пікове навантаження шляхом переміщення його в непікові години. Таким чином, вдалося знизити ціну на електроенергію на 5%, що є невеликим показником.

Такий же підхід було використано у дослідженні [20]. Розроблена система використовує генетичний алгоритм для планування роботи приладів

у СДА відповідно до зміни цін на електроенергію. Оскільки піковий час являє собою період, протягом якого ціна на електроенергію стає високою, то мета алгоритму полягає в тому, що він створює оптимальний розклад, який не перевищує поріг навантаження у всіх часових інтервалах та намагається перемістити використання приладів у непікові години (рис. 2.6). Результати експериментів з різними умовами споживання електроенергії показують, що запропонований алгоритм знижує витрати на електроенергію в середньому від 27% до 36,4%.

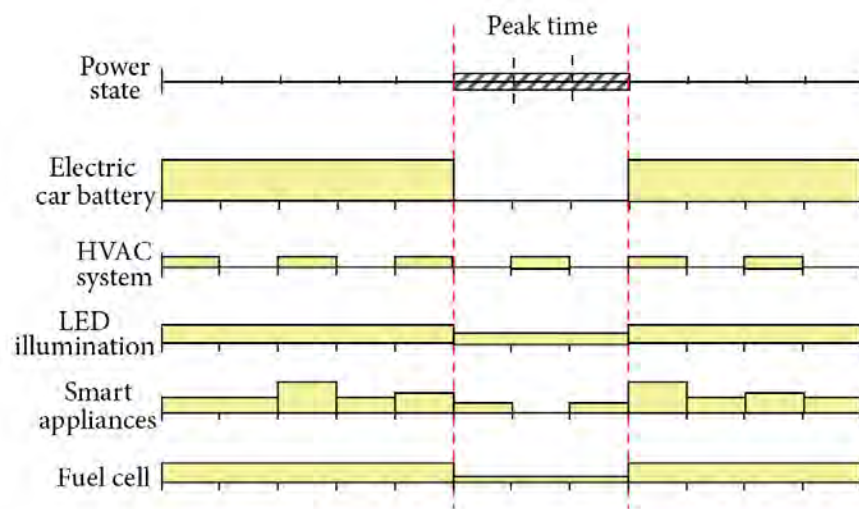


Рисунок 2.6 – Оптимальне планування споживання електроенергії [20]

2.3 Використання моделі ARIMA

У дослідженні [21] авторами була запропонована система, яка аналізує споживання електроенергії у СДА в режимі реального часу та буде прогнозувати майбутнє енергоспоживання з використанням моделі ARIMA. Цей аналіз та прогнозування допоможе у майбутньому знизити споживання електроенергії у СДА, що призведе до економії енергії, а також до впровадження ефективної системи аналізу енергоспоживання.

ARIMA – це одна з моделей для роботи з даними часових рядів, яка може бути використана для прогностичного аналізу. Її ідея полягає в тому, що деякий часовий ряд являє собою набір випадкових величин, які залежать

від часу, але зміни в усьому часовому ряді мають певні правила, які можуть бути представлені математичною моделлю та на основі якої можна прогнозувати майбутні значення [21].

Для проведення експерименту було використано вибірку, яка включає дані про споживання електроенергії з 2000 року по 2022 рік. Отримані результати показують, що прогнозовані значення дуже близькі до історичних та довірчий інтервал становить 95% (рис. 2.7).

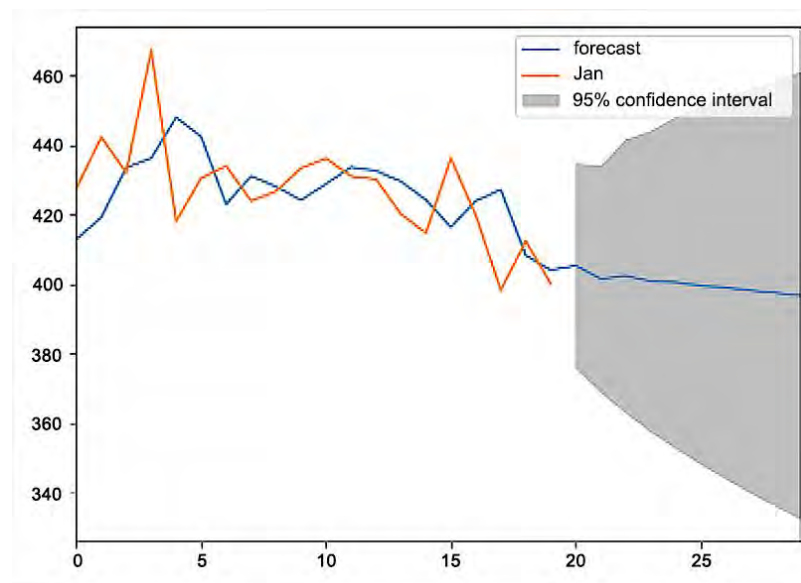


Рисунок 2.7 – Спрогнозовані дані споживання електроенергії [21]

2.4 Використання штучних нейронних мереж

У роботах [22]-[29] автори пропонують використовувати нейронні мережі для прогнозування споживання електроенергії та у майбутньому використовувати ці дані для зниження витрат на електроенергію та оптимізацію роботи побутових приладів.

У статті [22] було продемонстровано роботу штучної нейронної мережі довгої короткочасної пам'яті (ДКЧП). Вона являє собою особливий вид рекурентних нейронних мереж (РНМ) та характеризується тим, що на відміну від традиційних РНМ, така мережа добре підходить для навчання з метою класифікації, обробки або передбачення деяких значень. Вибірка була

розділена на навчальну і тренувальну та становила відповідно 80% та 20%. Вхідними даними виступав час, а вихідними – навантаження на електромережу. Отже, розроблена система дуже добре виконує прогнозування споживання електроенергії на основі історичних даних, про що свідчить низьке значення середньоквадратичної похибки, значення якої становить 0,43.

У статті [23] також було проведено дослідження ефективності нейронної мережі ДКЧП для прогнозування навантаження на електромережу. Для навчання було використано чотири вибірки, що були отримані з різних будинків. Результати свідчать, що такий вид нейронної мережі добре виконує прогнозування на 7 днів вперед та середня абсолютна похибка становить 14%, а для прогнозування на 24 години – 35%, що є набагато гіршим результатом.

Порівняння нейронних мереж, які частіше використовуються для прогнозування, а саме: нейронної мережі прямого поширення та нейронної мережі ДКЧП – було розглянуто у роботі [24]. За отриманими результатами можна зробити висновок про те, що другий тип нейронної мережі працює набагато краще першої та середня абсолютна похибка дорівнює 21,99%.

Оскільки енергоспоживання у будинку має високу кореляцію з погодою, у дослідженні [25] було запропоновано також використовувати нейронну мережу ДКЧП. Проведені експерименти показують різні результати щодо прогнозування (рис. 2.8). Наприклад, після агрегації даних зі всіх 114 квартир можна побачити, що прогноз мав найвищу точність та значення середньої абсолютної похибки становило 9,42%. Значення середньої абсолютної похибки з використанням погодних умов для окремої квартири становить 24,92%, в той час як без її використання – 32,08%. Отже, додавання погодних умов може значно поліпшити прогнозування навантаження як для окремого домогосподарства, так і для всієї будівлі.

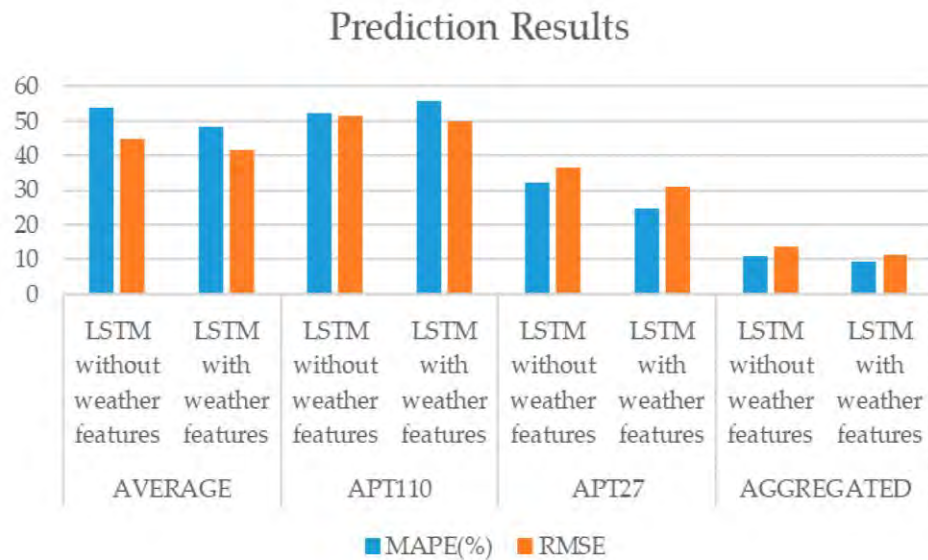


Рисунок 2.8 – Порівняння результатів прогнозування за допомогою штучної нейронної мережі ДКЧП [25]

Автори [26] запропонували використовувати РНМ для прогнозування енергоспоживання. Проведені експерименти дозволили порівняти дану нейронну мережу з популярним статистичним методом ARIMA, що широко використовується для прогнозування в різних областях. Згідно отриманих результатів, можна зробити висновок, що РНМ здатна робити прогнози з більш високим ступенем точності, ніж порівнювана модель. Для методу ARIMA значення кореляції становить 39%, в той час як для нейронної мережі значення кореляції варіюється від 55% до 95%, що є набагато кращим результатом, але не є достатнім для прогнозування.

Задача прогнозування генерації електроенергії за допомогою сонячної електростанції була розглянута у роботі [27]. Автори також запропонували використовувати РНМ, яка приймає на вхід метеорологічні дані (сезон, час, УФ-індекс, температура, відносна вологість, швидкість вітру, тиск повітря) та здатна прогнозувати потужність у короткостроковій перспективі. Після тестування навченої моделі можна побачити, що середньоквадратична помилка варіюється від 0,0248 МДж до 0,538 МДж, а значення кореляції

становить 97%. Це свідчить про те, що розроблена система дозволяє точно передбачати вихідну генерацію сонячної електростанції (рис. 2.9).

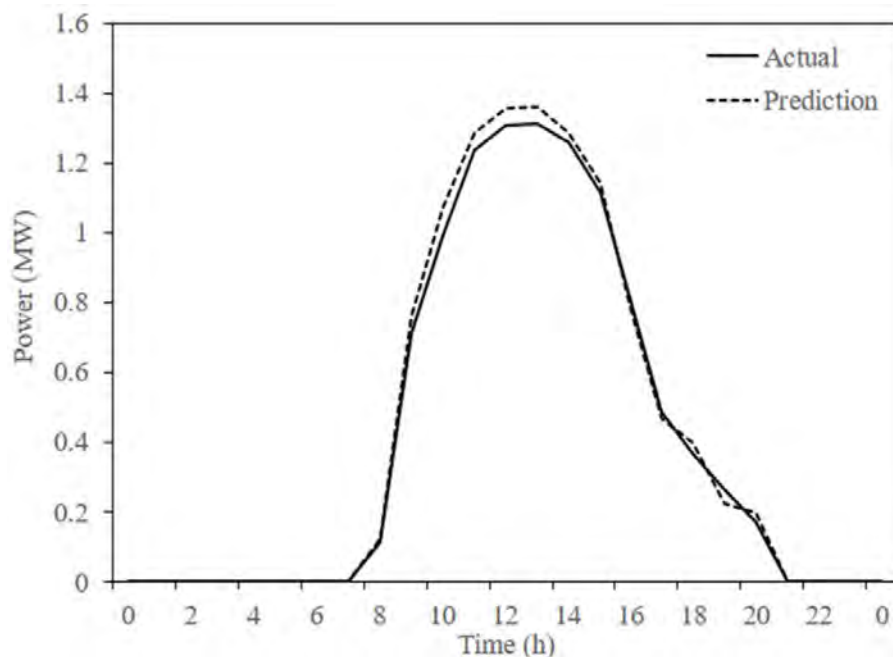


Рисунок 2.9 – Результати прогнозування генерації електроенергії за допомогою РНМ [27]

У роботі [28] було досліджено ефективність використання згорткової нейронної мережі для прогнозування енергоспоживання в одній будівлі. Використовуваний набір даних містить дані про споживання електроенергії за чотири роки (грудень 2006 – листопад 2010) з інтервалом в одну хвилину. Його було розділено на навчальну та тестову вибірку, де перші три роки використовувалися для навчання, а дані за останній рік – для тестування. Результати прогнозування навченої моделі показують, що середньоквадратична помилка становить 0,732 кВт (рис. 2.10). Це означає, що така система потребує більше тестування на різних наборах даних, а також модифікацій для поліпшення точності прогнозування.

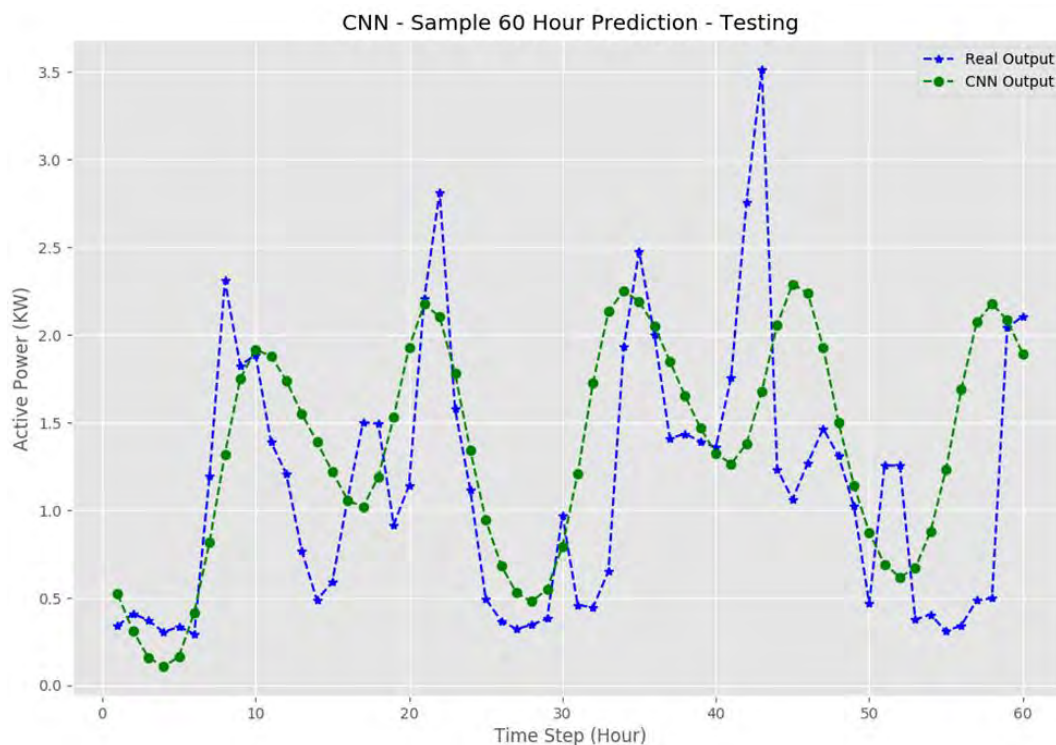


Рисунок 2.10 – Результати прогнозування, використовуючи згорткову нейронну мережу [28]

Автори [29] дослідили використання основних типів нейронних мереж для вирішення різних завдань у СДА. Оскільки кожна штучна нейронна мережа має певну перевагу та з різною точністю дозволяє обробляти різноманітні типи даних, а також прогнозувати деякі значення. Тому для вирішення цієї проблеми було запропоновано метод вибору оптимального типу штучної нейронної мережі для одного і того ж набору історичних даних.

Проведені експерименти показують, що використання нейронної мережі з прямим зв'язком найкраще підходить для задачі регулювання температури, бо помилка прогнозування становить 5% та є найменшою зі всіх протестованих нейронних мереж. Для забезпечення освітлення у СДА найкращі результати забезпечила РНМ, у якої помилка прогнозування становить менше 10%. Для забезпечення оптимального режиму безпеки найбільше підходить нейронна мережа ДКЧП, яка має похибку між прогнозованими та тестовими результатами менше 8%.

Аналізуючи результати, можна зробити висновок, що неможливо створити універсальну штучну нейронну мережу для завдань пов'язаних з СДА. Тому для вирішення поставленої задачі необхідно обрати таку нейронну мережу, яка буде забезпечувати найкращу продуктивність відповідно до специфіки завдання.

2.5 Висновки до розділу

В ході досліджень було детально проаналізовано наукові роботи, у яких автори пропонують різні інтелектуальні методи для ефективного контролю енергоспоживання. В результаті, всі розглянуті методи було поділено на дві основні групи: оптимізація розкладу роботи побутових приладів та прогнозування споживання або генерації електроенергії. Результати досліджень зведено в таблицю 2.1.

Таблиця 2.1 – Задачі щодо зниження енергоспоживання та методи їх вирішення

Задача	Метод вирішення
Оптимізація розкладу роботи побутових приладів	Метод рою часток [13]-[17]
	Генетичний алгоритм [18]-[20]
Прогнозування споживання або генерації електроенергії	ARIMA [21], [26]
	Нейронна мережа з довгою короткочасною пам'яттю [22]-[25], [29]
	Рекурентна нейронна мережа [26], [27], [29]
	Нейронна мережа прямого поширення [24], [29]
	Згорткова нейронна мережа [28]

Ідея методу оптимізації розкладу роботи побутових приладів полягає в автоматичному плануванні використання електроприладів, враховуючи

звички мешканців, в періоди доби, коли електроенергія має низьку ціну. Таким чином, цей підхід дозволяє вирішити проблему мінімізації пікового навантаження в мережі, що дозволяє знизити комунальні витрати на електроенергію. Отримані результати у дослідженнях [13]-[20] показали, що використання такого підходу дозволило знизити витрати на електроенергію до 55%, що є досить високим показником. Проте найбільшим недоліком оптимізації розкладу є те, що рівень комфорту мешканців може бути значно знижено, тому що використовувати більшу кількість електроприладів дозволено тільки в певний час, що не завжди буває зручно.

Аналіз історичних даних та прогнозування генерації електроенергії означає, що спрогнозовані дані допоможуть у майбутньому розробити план економного користування приладами та знизити споживання електроенергії у СДА, а також впровадити ефективну систему аналізу енергоспоживання. Дослідження у роботах [22]-[29] показали, що найвища точність прогнозування, яка була досягнута, становить 97%, що є дуже високим показником та означає, що така система може точно робити прогнозування.

Підсумовуючи отримані результати досліджень, можна сказати, що запропоновані методи для зниження енергоспоживання є доцільними у використанні при удосконаленні реальної СДА.

Задача оптимізації розкладу роботи побутових приладів яка базується на використанні плаваючого тарифу не є актуальною, тому що в нашій країні встановлено фіксований тариф на електроенергію. Оскільки досліджувана СДА використовує альтернативні джерела енергії, то постановка такої задачі може бути змінена на оптимальне планування користування приладами, в періоди, коли генерація електроенергії є найбільшою.

Дослідження показали, що вирішення задачі оптимізації розкладу роботи побутових приладів пропонується такими алгоритмами: метод рою часток та генетичний алгоритм. У роботах [16], [19] було проведено порівняння цих двох алгоритмів та результати показали, що метод рою часток зміг знизити ціну на електроенергію на 8% більше, ніж генетичний

алгоритм, використовуючи одні й ті ж самі дані. Автори [30] порівняли продуктивність цих же алгоритмів та результати показали, що алгоритм рою часток потребує менше ітерацій для обчислення оптимального рішення, а точність перевищує генетичний алгоритм. Таким чином, можна зробити висновок, що алгоритм рою часток використовується частіше та показав себе набагато краще, тому його використання є доцільним у роботі.

Задача прогнозування генерації електроенергії є також актуальною, оскільки СДА використовує сонячні електростанції та вітрогенератори. Так, спрогнозовані дані будуть використовуватись алгоритмом оптимізації для створення оптимального розкладу роботи побутових приладів, що дозволить користувачу спланувати розпорядок майбутнього дня.

Проведені дослідження показали, що для прогнозування генерації електроенергії пропонується ряд методів, а саме: ARIMA, нейронна мережа ДКЧП, РНМ, нейронна мережа прямого поширення та згорткова нейронна мережа. Дослідження у роботі [29] свідчать, що неможливо створити універсальну нейронну мережу, яка буде підходити під різні задачі, тому необхідно провести ряд комп'ютерних експериментів та обрати найкращу модель, яка буде забезпечувати найбільшу точність прогнозування для досліджуваної СДА. Таким чином, пропонується порівняти наступні нейромережеві моделі: нейронна мережа ДКЧП, РНМ та мережа прямого поширення. Використання методу ARIMA та згорткової нейронної мережі не є доцільним, оскільки у дослідженнях [26], [28] вони показали незадовільні результати у порівнянні з вище перерахованими нейронними мережами.

Отже, для досягнення поставленої мети пропонується розробка комплексного інтелектуального методу контролю енергоспоживання на основі прогнозування генерації електроенергії з максимальним використанням альтернативних джерел та оптимізації розкладу користування побутовими приладами, використовуючи штучні нейронні мережі та алгоритм рою часток.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз вимог

Першим етапом під час проєктування програмного забезпечення є аналіз вимог до нього. Виходячи з дослідження архітектури реальної системи домашньої автоматизації (розд. 1), а також проведеного огляду існуючих інтелектуальних методів в сфері оптимального споживання енергії (розд. 2) було складено функціональні та нефункціональні вимоги до розроблюваного програмного забезпечення (рис. 3.1), реалізація яких дозволить підвищити ефективність користування електроприладами в реальній СДА.



Рисунок 3.1 – Вимоги до програмного забезпечення

Для забезпечення надання рекомендацій щодо ефективного користування електроприладами, підсистема інтелектуальної підтримки повинна реалізовувати комплексний метод на основі нейронної мережі для прогнозування генерації електроенергії та алгоритму рою часток для створення оптимального погодинного розкладу користування приладами.

Для генерації оптимального розкладу необхідно мати список найпотужніших побутових приладів, що використовуються. Всі ці прилади повинні зберігатися у базі даних, яка входить до складу СДА. Кожен електроприлад включає наступні дані: назва, номінальна потужність (Вт), кількість годин для роботи.

Оскільки історичні дані, що використовуються нейронною мережею для навчання постійно поповнюються з метеосервісу, то підсистема інтелектуальної підтримки повинна надавати можливість навчання та перенавчання моделі нейронної мережі на нових даних для збільшення точності прогнозування.

Підсистема інтелектуальної підтримки повинна інтегруватися з СДА OpenHAB. Для цілодобового моніторингу показників енергоспоживання необхідно забезпечити наявність модулів вимірювання енергоспоживання, а саме PZEM-004T (розд. 1.4, рис. 1.10), які будуть підключені до платформи OpenHAB та зберігати вимірювані показники у базу даних, яка входить до складу СДА. У разі виникнення небезпечних змін у показниках, що моніторяться, користувач повинен отримати відповідне повідомлення.

Для захисту від несанкціонованого доступу до всіх функцій підсистеми інтелектуальної підтримки, користувач повинен авторизуватися, використовуючи логін та пароль.

Графічний інтерфейс повинен бути інтуїтивно зрозумілим під час використання, а також та не повинен бути перевантаженим графічними елементами. Доступ до графічного інтерфейсу має бути отримано за допомогою будь-якого сучасного браузера, що позбавить необхідності встановлювати додаткові застосунки та не залежати від операційної системи.

Це означає, що розроблювана підсистема має бути реалізована у вигляді вебзастосунку. Також інтерфейс має коректно відображатися на планшетах та мобільних пристроях. У разі виникнення нештатних ситуацій, користувач повинен отримувати повідомлення, які мають містити опис проблеми, а також код помилки. Всі повідомлення мають відображатися у вигляді спливаючих вікон графічного інтерфейсу.

Програмна документація повинна містити відомості для налаштування та експлуатації програмного виробу, а також описувати вимоги до апаратної частини.

Діаграму прецедентів для розроблюваної підсистеми інтелектуальної підтримки користувачів зображено на рисунку 3.2, яка демонструє інформацію про користувачів та їх взаємодію з підсистемою.

Основним актором виступає користувач, який взаємодіє з підсистемою інтелектуальної підтримки, а прецедентом – виконувана системою дія. Також є зацікавлені особи – це набір зовнішніх систем, які надають необхідну інформацію для роботи розроблюваної підсистеми: база даних для збереження необхідних даних, метеосервіс для надання прогнозу погоди та платформа OpenHAB для отримання даних з датчиків.

Використовуючи графічний інтерфейс, користувач взаємодіє з підсистемою та може виконувати наступні функції: авторизуватись, додати або видалити електроприлади, створити оптимальний розклад користування побутовими приладами, а також переглянути показники споживання та генерації електроенергії. В свою чергу, підсистема інтелектуальної підтримки отримує команди від користувача та виконує певні дії, щоб задовольнити його вимоги.

Таким чином, розроблювана підсистема інтелектуальної підтримки дозволить надавати рекомендації мешканцям будинку щодо ефективного користування побутовими приладами, використовуючи лише альтернативні джерела енергії та не залежати від зовнішньої електромережі.

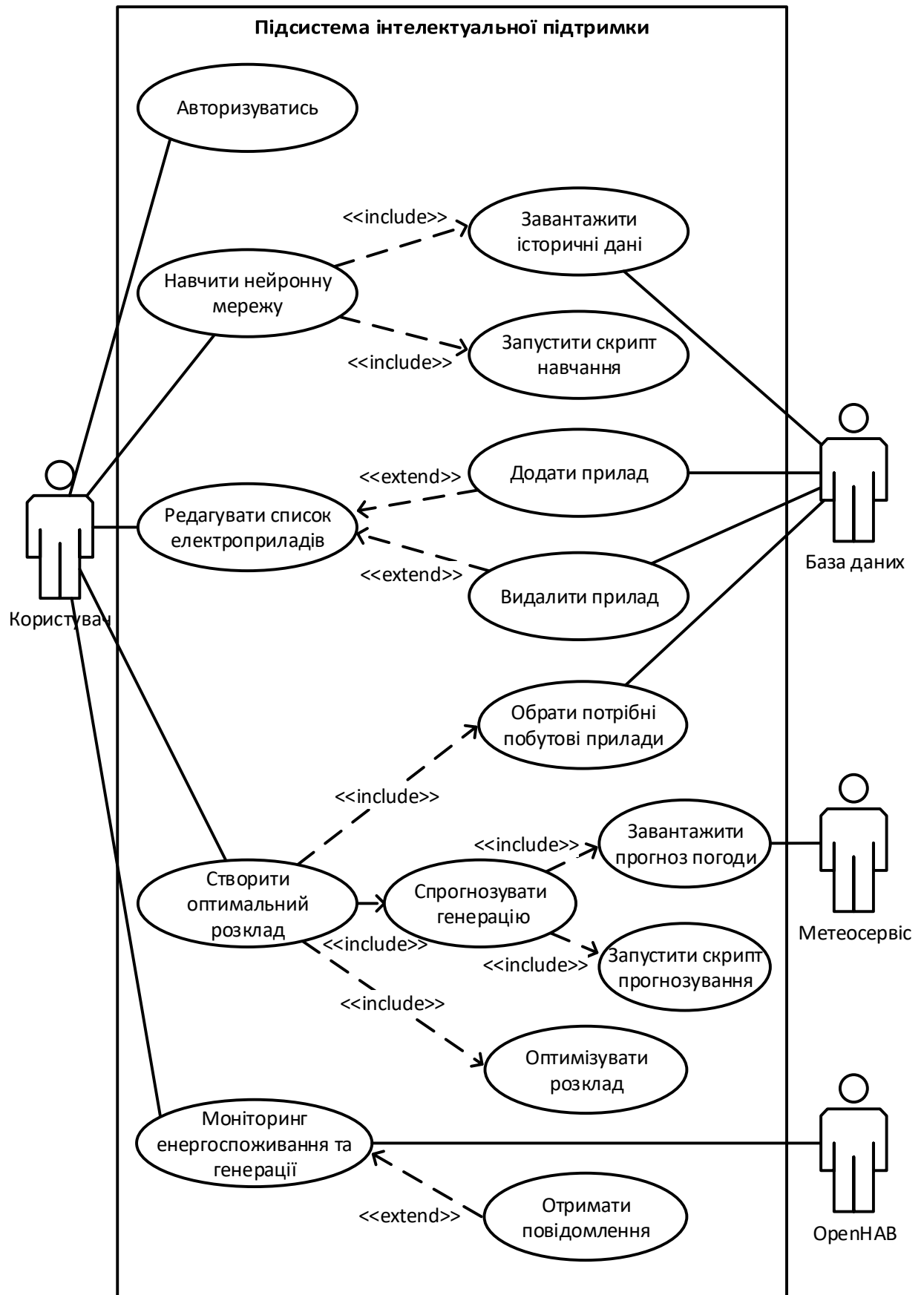


Рисунок 3.2 – Діаграма прецедентів підсистеми інтелектуальної підтримки

3.2 Вдосконалення архітектури системи домашньої автоматизації

Архітектуру СДА та її взаємодію з розроблюваною підсистемою інтелектуальної підтримки наведено на рисунку 3.3.

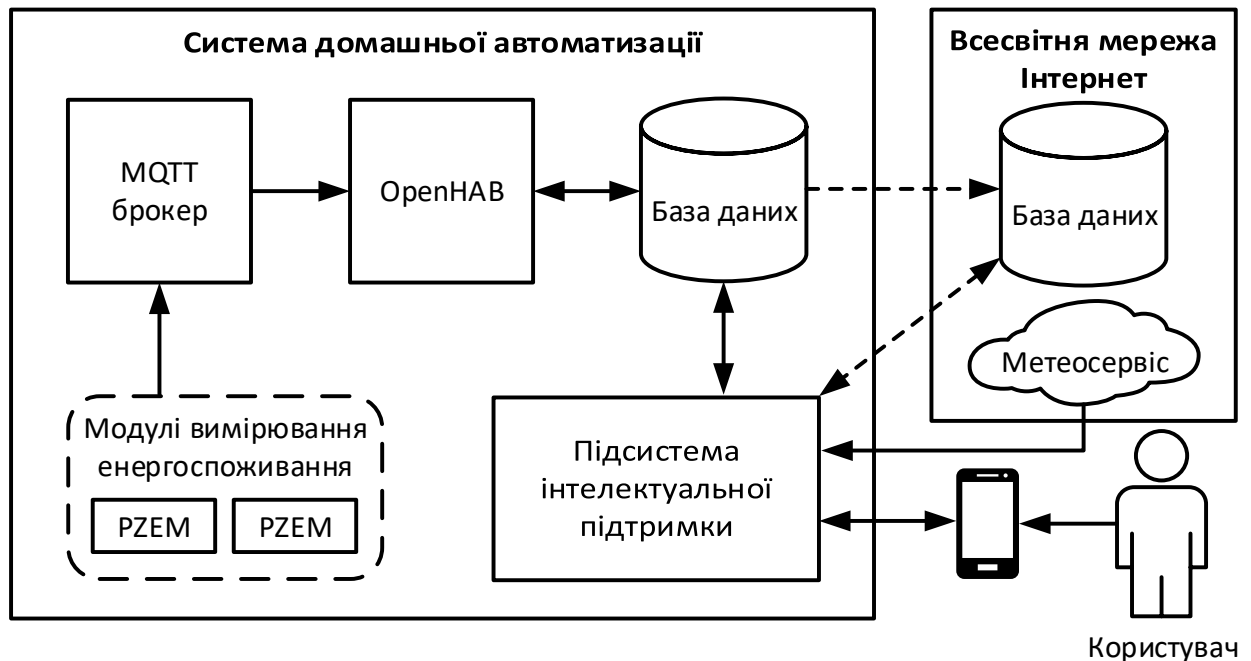


Рисунок 3.3 – Узагальнена архітектура СДА

Головним елементом досліджуваної СДА виступає платформа OpenHAB. Дані з модулів PZEM-004T приймаються MQTT брокером, який відправляє їх до OpenHAB та який в свою чергу записує їх у базу даних.

Розроблювана підсистема інтелектуальної підтримки взаємодіє з локальною базою даних, що входить до СДА, та отримує всі необхідні показники з модулів вимірювання електроенергії, аналізує їх у реальному часі, а також використовує для навчання нейронної мережі. Зберігання списку найпотужніших електроприладів, що використовуються, відбувається у цій же базі даних. Віддалена база даних використовує механізм реплікації та є необов'язковим елементом архітектури, хоча є необхідною для проведення тестувань роботи підсистеми.

За допомогою будь-якого пристрою користувач може підключитися до підсистеми інтелектуальної підтримки, використовуючи тільки браузер, для створення оптимального погодинного розкладу користування приладами, моніторингу енергоспоживання та генерації, редагування списку найпотужніших електроприладів, а також отримання важливих повідомлень.

Взаємодію між підсистемою інтелектуальної підтримки та іншими об'єктами СДА для створення оптимального розкладу користування електроприладами наведено на рисунку 3.4.

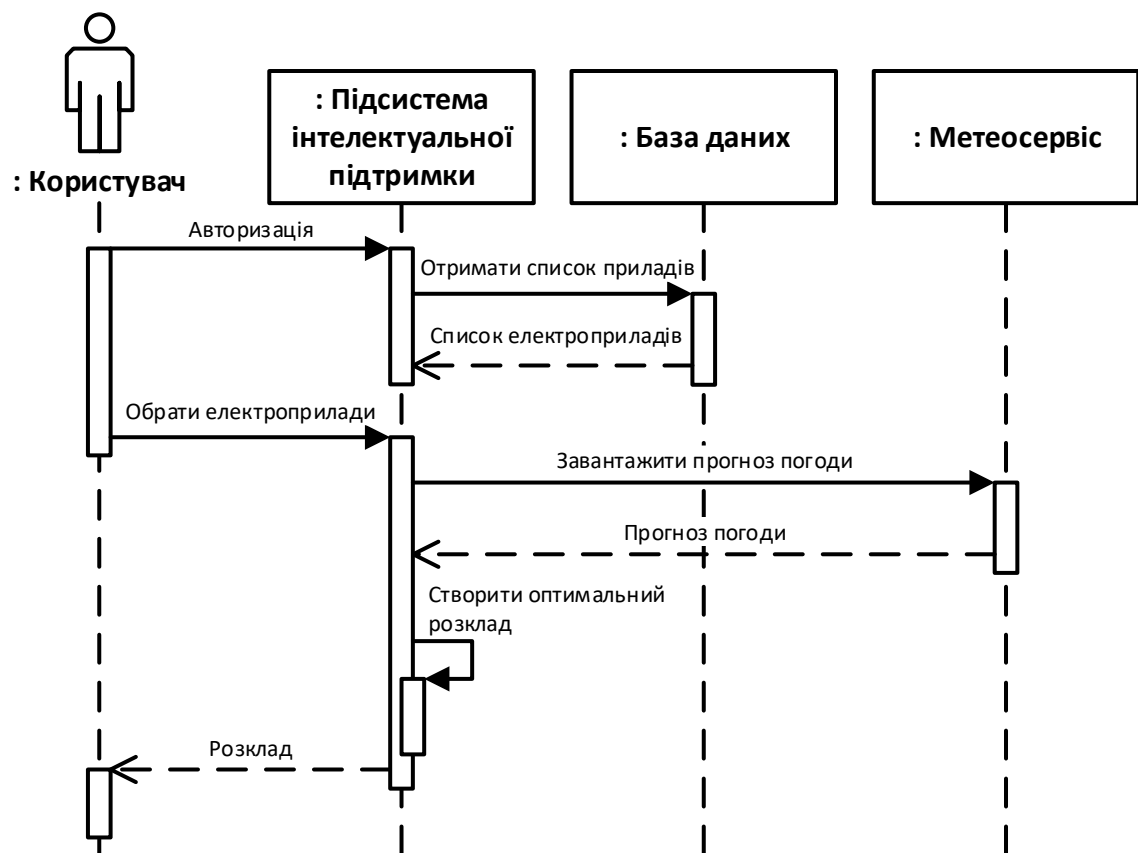


Рисунок 3.4 – Діаграма послідовності створення оптимального розкладу

Використовуючи графічний інтерфейс, користувач повинен авторизуватися. Спочатку він має обрати необхідні йому електроприлади для користування, які завантажуються з бази даних. Далі програмне забезпечення отримує прогноз погоди, використовуючи метеосервіс, після чого починається створення оптимального розкладу, використовуючи

комплексний інтелектуальний метод для підвищення ефективності користування електроприладами. Усі результати відображаються у браузері користувача.

3.3 Реалізація комплексного інтелектуального методу прогнозування та оптимізації

Виходячи з проведених досліджень щодо існуючих інтелектуальних методів в сфері оптимального споживання електроенергії (розд. 2), було запропоновано розробити комплексний інтелектуальний метод контролю енергоспоживання на основі прогнозування генерації електроенергії та оптимізації розкладу використання електроприладів, використовуючи штучні нейронні мережі та алгоритм рою часток.

Алгоритм реалізації комплексного інтелектуального методу для підвищення ефективності користування електроприладами представлено у вигляді діаграми діяльності (рис. 3.5).

Реалізація комплексного методу передбачає два основні етапи:

- прогнозування генерації електроенергії;
- створення погодинного розкладу користування електроприладами.

Для реалізації комплексного інтелектуального методу спочатку необхідно провести навчання нейронної мережі на історичних даних. Історичні дані мають включати дату та час, містити інформацію про генерацію електроенергії, а також дані про погоду в певний час.

Після успішного навчання, нейронна мережа буде здатна прогнозувати генерацію електроенергії. Вхідними даними для роботи нейронної мережі є погодинний прогноз погоди на добу, який можна отримати, використовуючи метеосервіс. Таким чином, спрогнозована інформація допоможе оцінити доступну енергію та розробити план ефективного користування приладами на наступний день, використовуючи тільки альтернативні джерела.

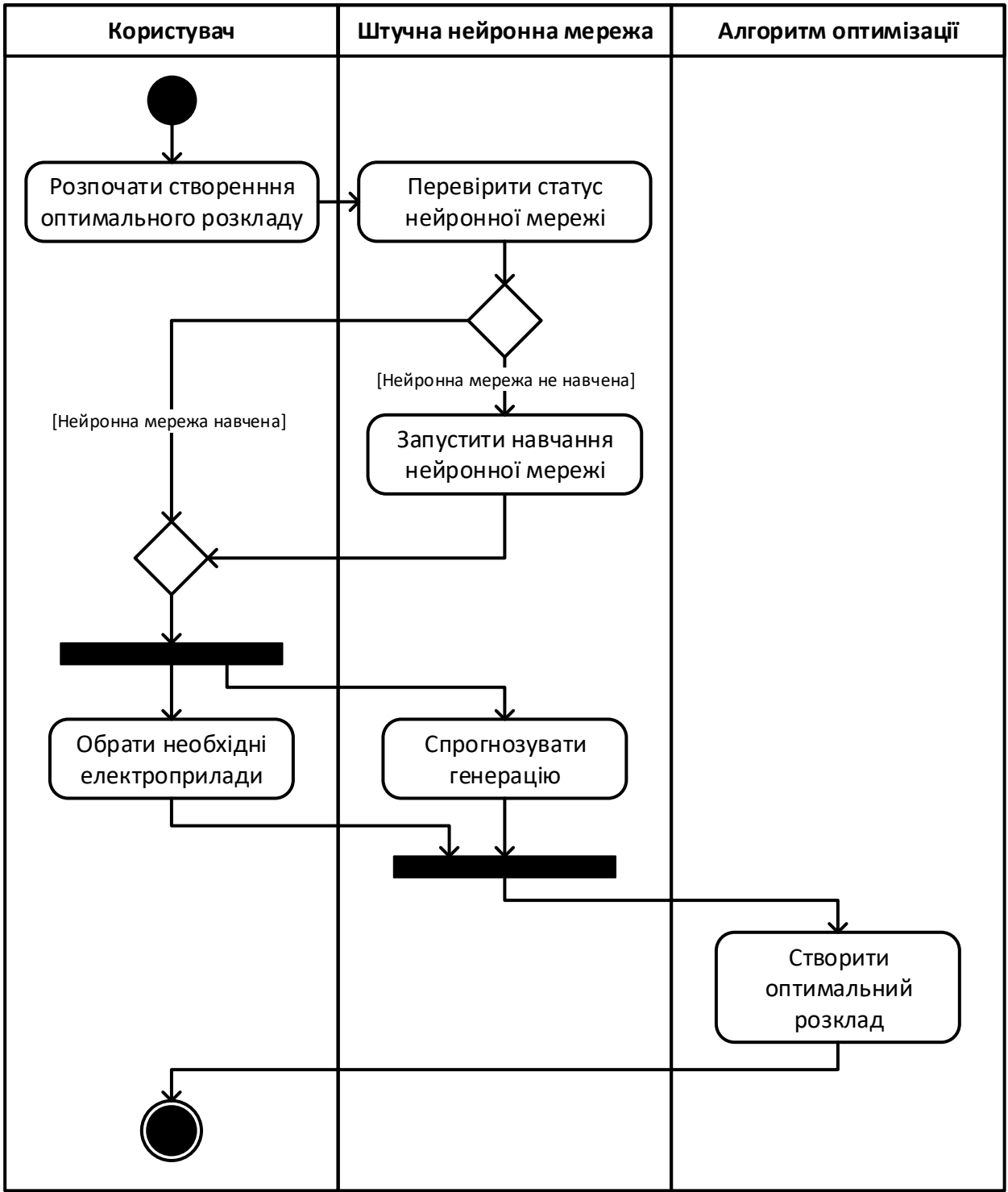


Рисунок 3.5 – Алгоритм запропонованого комплексного інтелектуального методу прогнозування та оптимізації

Наступним етапом комплексного методу є створення оптимального погодинного розкладу користування електроприладами, використовуючи алгоритм рою часток (розд. 2.5). Ідея полягає в автоматичному плануванні використання електроприладів протягом наступної доби, враховуючи звички

мешканців, в періоди, коли генерація електроенергії є найбільшою. Критерієм оптимізації є створення такого розкладу, у якому буде використано якнайбільше альтернативної електроенергії, щоб забезпечити роботу всіх запланованих побутових приладів. Таким чином, користувач зможе суттєво заощадити кошти та менше залежати від тарифів на комунальні послуги за користування електроенергією. Вхідними даними для роботи алгоритму оптимізації є спрогнозовані дані щодо генерації електроенергії, а також список побутових приладів, що будуть використані наступного дня. Кожен такий прилад має включати наступні дані: назва, номінальна потужність (Вт), кількість годин для роботи та проміжок часу для зручного користування протягом дня.

3.4 Вибір метеосервісу

Для реалізації комплексного інтелектуального методу необхідно забезпечити завантаження даних погодинного прогнозу погоди з метеосервісу, використовуючи Application Programming Interface (API).

API – це інтерфейс прикладного програмування, який має визначений набір функцій. Цей набір дозволить розробникам стороннього програмного забезпечення отримати доступ до певних функцій сервісу або завантажити необхідні дані [31].

На сьогоднішній день можна виділити декілька найпопулярніших метеосервісів, які надають обмежений безкоштовний доступ до прогнозу погоди: «OpenWeatherMap», «Weatherbit», «AccuWeather», «Visual Crossing».

«OpenWeatherMap» API надає безкоштовний доступ до невеликої кількості даних, а саме: до поточної інформації про погоду та прогноз погоди на п'ять днів. Додаткові функції доступні тільки у версії за кошти [32].

«Weatherbit» API є схожим за обмеженнями до попереднього метеосервісу та дозволяє отримати дані про поточну погоду, а також прогноз на тиждень [33].

«AccuWeather» API забезпечує надання інформації про погоду в даний момент часу, добовий прогноз на п'ять днів, історичний прогноз за добу, а також погодинний прогноз на 12 годин, що не є достатнім для реалізації комплексного інтелектуального методу [34].

«Visual Crossing» API дозволяє отримати багато різноманітної інформації. Так, безкоштовна версія надає доступ до погодинного прогнозу погоди на 15 діб та доступ до історичних даних аж до 50 років [35].

Порівняння розглянутих метеосервісів, що підтримують роботу з API зведено в таблицю 3.1.

Таблиця 3.1 – Порівняння популярних метеосервісів

Критерій порівняння	Open Weather Map	Weather bit	Accu Weather	Visual Crossing
Історичні дані погоди	–	–	±	+
Інформація про поточну погоду	+	+	+	–
Добовий прогноз погоди	+	+	+	+
Погодинний прогноз погоди	–	–	±	+
Кількість безкоштовних запитів	60 / хв	50 / день	50 / день	1000 / день

В результаті порівняльного аналізу можна зробити висновок, що «Visual Crossing» API задовольняє всім вимогам для реалізації комплексного інтелектуального методу, тому що він надає безкоштовний доступ до історичних даних, які будуть використовуватися для навчання нейронної мережі, а також до погодинного прогнозу погоди на декілька діб, що необхідно для прогнозування генерації електроенергії. Кількість безкоштовних запитів в день є достатньою для забезпечення розроблюваного комплексного інтелектуального методу необхідними даними.

3.5 Вибір системи керування базами даних

Досліджувана СДА отримує дані з модулів PZEM (розд. 1.4, рис. 1.10) кожні вісім секунд та зберігає інформацію щодо енергоспоживання у базу даних `rrd4j`. Оскільки використовувана база даних є циклічною, то вона має незмінний розмір. Це досягається за рахунок збереження фіксованої кількості даних та стиснення старих даних методом збереження середнього значення за деякий період [4]. Таким чином, для старих даних неможливо отримати детальну статистику.

Для можливості збереження всієї статистики енергоспоживання, яка необхідна для навчання нейронної мережі запропонованого комплексного інтелектуального методу, необхідно обрати нову систему керування базами даних (СКБД), що буде задовольняти цим вимогам.

Платформа OpenHAB підтримує зберігання даних до таких популярних СКБД: MySQL, MariaDB, PostgreSQL, MongoDB, SQLite [36].

MySQL – це одна з найпопулярніших реляційних систем СКБД з відкритим вихідним кодом. Клієнт-серверна архітектура дозволяє створювати хмарні бази даних та інтегрувати їх з розроблюваним програмним забезпеченням, що забезпечує стабільну роботу. Функції безпеки включають систему привілеїв доступу до бази даних, яка забезпечує автентифікацію користувача та систему управління обліковими записами користувачів. Основною перевагою MySQL є простота у використанні, що забезпечує гнучкі способи керування даними за допомогою Structured query language (SQL). Головним недоліком є проблема масштабованості. Це означає, що зі збільшенням розміру бази даних зростає потреба в ресурсах для доступу, аналізу та використання цих даних, що потребує немало коштів. Таким чином, база даних MySQL найчастіше використовується для вебзастосунків з невеликою кількістю даних [37], [38]. Ліцензія General Public License (GPL) надає користувачеві права на копіювання, зміну й розповсюдження програми [39].

MariaDB є форком MySQL з відкритим вихідним кодом, який має ті ж самі команди, API та бібліотеки. Основна ідея цієї СКБД – це оптимізація та покращення вже існуючої бази даних MySQL. Унікальний механізм зберігання даних, що було реалізовано, робить запис даних на 25% швидше, ніж в MySQL. Це дозволяє більш ефективно зберігати різні типи даних та доступ до них стає менш залежним від продуктивності сервера. Крім того, MariaDB підтримує використання плагінів, які можуть бути додані без необхідності перевстановлювати базу даних з самого початку. Так, плагіни можуть завантажуватися під час запуску або вже під час працюючої системи. Оскільки ця СКБД була створена не так давно, кількість фахівців відносно невелика та це означає, що отримати підтримку може бути складно [37], [38].

PostgreSQL – це об'єктно-реляційна СКБД, в якій об'єкти та табличні підходи об'єднуються для побудови більш складних структур даних. PostgreSQL дозволяє зберігати користувальницькі типи даних. Оскільки для розроблюваного програмного забезпечення можуть знадобитися різні типи даних, які будуть зберігатися у СКБД, це дозволяє забезпечити гнучкість та підвищити продуктивність системи. Крім того, PostgreSQL має підтримку додаткових інструментів, як безкоштовних, так і комерційних, які дозволяють розширювати функціонал бази даних. Таким чином, PostgreSQL надає більше функціоналу та можливостей на відміну від MySQL. Одним з недоліків є те, що PostgreSQL працює повільніше із-за широкого спектру додаткових функцій, ніж MySQL [37]. Ліцензія Free/Libre Open Source software (FLOSS) означає, що СКБД має відкритий вихідний код та кожен розробник може присвятити себе розробці цієї бази даних, а також запропонувати власні рішення помилок [40].

MongoDB – це документо-орієнтована СКБД з відкритим вихідним кодом. Головною відмінністю такої бази даних від реляційних полягає в тому, що дані зберігаються у форматі JavaScript Object Notation (JSON). Це означає, що MongoDB може працювати з напівструктурованими даними в документах та не обмежуватися суворими вимогами до таблиць. Такий підхід

дозволяє мати гнучку СКБД, яка може зберігати будь-які типи даних. Завдяки документо-орієнтованому підходу MongoDB може забезпечити більш високу швидкість роботи, ніж традиційна реляційна база даних. Великою перевагою є горизонтальна масштабованість, коли дані розподіляються по різних серверах. Це стає більш важливим для систем, які працюють з додатками для обробки великих об'ємів даних. На відміну від реляційних баз даних, MongoDB використовує власний механізм для доступу даних MongoDB Query API, який є значно простішим у використанні, але він не підтримує транзакції та об'єднання даних, що є основним недоліком. Також MongoDB потребує використання великої кількості оперативної пам'яті, що допомагає збільшити продуктивність системи. Зазвичай MongoDB використовується в тому випадку, коли обсяг інформації великий та немає необхідності зберігати складні структури даних [41]. Ліцензія Affero General Public License (AGPL) є дуже схожою на GPL та також дозволяє використовувати, розповсюджувати та модифікувати програмне забезпечення [42].

SQLite є файловою СКБД, яка не потребує встановлення та налаштування. Це означає, що така база даних міститься в одному файлі на диску та всі операції читання та запису виконуються безпосередньо в цьому файлі. Перевагою є те, що бібліотека SQLite є дуже компактною та її розмір становить менше 600 Кбайт, що робить її актуальною для Інтернету речей, де місце на диску є дуже обмеженим. Оскільки SQLite є файловою СКБД, це може спричинити проблеми з продуктивністю при роботі з великими наборами даних через обмеження файлової системи, що є головним недоліком [43]. Ліцензія public domain (PD) означає, що дана СКБД вільна для використання будь-ким в будь-яких цілях без обмежень [44].

Результати порівняльного аналізу підтримуваних СКБД платформою OpenHAB зведено в таблицю 3.2.

Таблиця 3.2 – Порівняння популярних СКБД

Критерій порівняння	MySQL	MariaDB	PostgreSQL	MongoDB	SQLite
Архітектура	клієнт-серверна	клієнт-серверна	клієнт-серверна	клієнт-серверна	файлова
Тип	реляційна	реляційна	об'єктно-реляційна	документо-орієнтована	реляційна
Масштабованість	вертикальна	вертикальна	вертикальна	горизонтальна	вертикальна
Схема	статична	статична	статична	динамічна	статична
Мова запитів	SQL	SQL	SQL	MongoDB Query API	SQL
Рівень безпеки	високий	високий	високий	середній	низький
Тип ліцензії	GPLv2	GPLv2	FLOSS	AGPLv3	PD
Швидкість читання	висока	висока	середня	низька	висока
Швидкість запису	середня	середня	низька	висока	середня
Розмір ком'юніті на «Stack Overflow»	653 тис.	13 тис.	162 тис.	168 тис.	92 тис.

Згідно порівняльного аналізу розглянутих СКБД можна зробити висновок, що використання нереляційної бази даних, а саме MongoDB є доцільним у досліджуваній СДА. Оскільки дані щодо енергоспоживання надходять до платформи OpenHAB кожні вісім секунд, саме ця СКБД забезпечить зберігання великих об'ємів даних без втрати продуктивності у порівнянні з традиційними реляційними базами даних. Крім того, для такої системи немає необхідності використовувати транзакції та складні об'єднання даних, тому що платформа OpenHAB зберігає історичні дані у вигляді пар «ключ – значення», які будуть використовуватись в такому ж вигляді. Можливість горизонтальної масштабованості дозволить у майбутньому за невеликі кошти розширити об'єм використовуваної СКБД.

Отже, для вдосконалення архітектури реальної СДА було прийнято рішення використовувати СКБД MongoDB замість rrd4j, що була встановлена за замовчуванням з платформою OpenHAB. Таким чином, MongoDB задовольняє всім вимогам щодо збереження великих об'ємів даних, які будуть використовуватися для реалізації запропонованого комплексного інтелектуального методу.

3.6 Вибір інструментарію розробки

3.6.1 Вибір вебфреймворку для розробки підсистеми

Проаналізувавши функціональні та нефункціональні вимоги до розроблюваного програмного забезпечення (рис. 3.1), було виявлено, що підсистема інтелектуальної підтримки має бути реалізована у вигляді вебзастосунку. Тому важливо обрати правильний стек технологій, перш ніж розпочати реалізацію програмного забезпечення.

Вебфреймворк – це набір модулів та бібліотек, які пропонують розробникам стандартний спосіб створення вебзастосунків. Зазвичай вони поділяються на дві групи: клієнтські та серверні, тому для розробки підсистеми інтелектуальної підтримки необхідно обрати серверний фреймворк, який буде відповідати за логіку роботи, взаємодіяти з базою даних та робити всі необхідні обчислення [45].

Так, за останні роки було створено багато різноманітних вебфреймворків, серед яких можна виділити найпопулярніші серверні фреймворки: Laravel, Django, Spring, Ruby on Rails [45].

Laravel – це фреймворк з відкритим вихідним кодом, написаний на мові програмування PHP. Такий інструмент дозволяє розробити невеличкий та повністю функціональний вебсайт із налаштованою логікою сервера. Laravel має безліч зручних для розробників функцій, наприклад, він надає готові пакети для створення авторизації користувачів. Також цей фреймворк популярний завдяки своєму вбудованому і потужному шаблонізатору, який легко використовувати для створення сучасного та динамічного інтерфейсу. Крім того, вебзастосунок, розроблений за допомогою Laravel, можна розмістити на спільному хостингу, що зробить утримання серверу дуже дешевим, коли вартість є ключовим параметром. Хоча мова PHP вважається не дуже безпечною, тому вона не рекомендується для використання там, де безпека є обов'язковою [46], [47]. Ліцензія MIT дозволяє розробникам використовувати цей фреймворк у закритому програмному забезпеченні [48].

Django – це вебфреймворк, написаний на Python, та який розроблений з урахуванням концепції «Batteries Included». Це означає, що все, що потрібно розробнику, вже включено в склад фреймворку. Таким чином, цей фреймворк не вимагає додаткових дій зі встановленням та налаштуванням сторонніх плагінів, оскільки все готово для використання. Django підходить для розробки великомасштабних інтерактивних вебзастосунків, які керуються базами даних. Також він має велике ком'юніті та документацію, що забезпечує одну з найкращих платформ для розробки вебзастосунків. Вагомим недоліком використання цього фреймворку є те, що він не підходить для створення невеликих проєктів, оскільки використовувана концепція зробить такий проєкт важким та великим, який буде потребувати великих обчислювальних потужностей [46], [49]. Ліцензія Berkeley Software Distribution (BSD) дозволяє пропрієтарне комерційне використання [50].

Spring є дуже популярним фреймворком, що написаний на мові програмування Java та використовується для створення вебзастосунків у корпоративному сегменті. Він має безліч корисних модулів, які легко підключаються та полегшують розробку складних додатків. Найчастіше використовуваними є Spring JDBC, що дозволяє організувати зручну взаємодію з популярними базами даних; Spring Transactions дозволяє створювати та управляти транзакціями; Spring Security дозволяє налаштовувати правила безпеки, які підтримують безліч протоколів та стандартів. Крім того, Spring можна використовувати разом з будь-якими іншими Java фреймворками, що дозволяє розширити можливості та функціональність розроблюваного проєкту. Вбудована перевірка працездатності постійно контролює, чи є застосунок працюючим та здатним виконувати свої функції, в іншому випадку він автоматично перезапускається, забезпечуючи роботу для користувача без збоїв. Таким чином, Spring не має обмежень та може задовольнити будь-які потреби під час розробки як для невеликих, так і для корпоративних вебзастосунків [51], [52].

Ruby on Rails виступає платформою для розробки вебзастосунків, що написана на мові програмування Ruby. Оскільки цей фреймворк постачається з широким спектром вбудованих плагінів та модулів, розробники можуть використовувати їх під час створення додатків, що дозволяє значно скоротити час, необхідний для написання шаблонного коду. Головним недоліком використання Ruby on Rails полягає в тому, що він повільніший за продуктивністю, ніж Spring. Також цей фреймворк не розрахований для використання вебсайтів з великим навантаженням, а також його складно масштабувати. Таким чином, Ruby on Rails найчастіше використовується для швидкого створення прототипу невеликих вебзастосунків [52], [53].

Таблиця 3.3 – Порівняння популярних вебфреймворків

Критерій порівняння	Laravel	Django	Spring	Ruby on Rails
Рік розробки	2011	2005	2002	2004
Мова програмування	PHP	Python	Java	Ruby
Технічна документація	+	+	+	+
Шаблонізатор	+	+	+	+
Відкритий вихідний код	+	+	+	+
Тип ліцензії	MIT	BSD	Apache 2.0	MIT
Інтеграція зі сторонніми бібліотеками	+	+	+	+
Підтримка додаткових мов програмування	–	–	Groovy, Kotlin	–
Висока масштабованість	–	+	+	–
Використання у високонавантажених додатках	–	+	+	–
Кількість знайдених програмних вразливостей за останні 5 років	56	82	30	46
Розмір ком'юніті на «Stack Overflow»	199 тис.	300 тис.	201 тис.	334 тис.
Кількість розробників на «LinkedIn»	472 тис.	395 тис.	941 тис.	308 тис.
Кількість вакансій на «Dou.ua»	42	26	110	51

В ході порівняльного аналізу (табл. 3.3) було віддано перевагу вебфреймворку Spring, тому що він має велику кількість модулів та бібліотек, які значно полегшують та пришвидшують написання коду. Також цей фреймворк є самим популярним серед його конкурентів та це означає, що існує чимало технічної документації та різних статей, які допоможуть легко та швидко освоїти цей інструмент. Ліцензія Apache 2.0 означає, що вихідний код можна модифікувати, а також Spring можна використовувати як в комерційних, так і в особистих проєктах. Головною особливістю є те, цей фреймворк здатний працювати у висконавантажених вебзастосунках, що задовольняє основним вимогам до розроблюваного програмного продукту. Крім того, Spring має найменшу статистику з виявлення програмних вразливостей та це означає, що цей фреймворк є найбезпечнішим серед порівняних.

Оскільки обраний фреймворк написано на мові програмування Java, тому саме ця мова буде використана під час розробки підсистеми інтелектуальної підтримки.

Розробка вебзастосунків з використанням фреймворку Spring передбачає використання шаблону Model-View-Controller (MVC), що реалізується за допомогою класу DispatcherServlet [54]. У контексті даного фреймворку цей клас є найголовнішим, бо він отримує запит та зіставляє його з потрібними ресурсами, такими як контролери, моделі та подання (рис. 3.6).

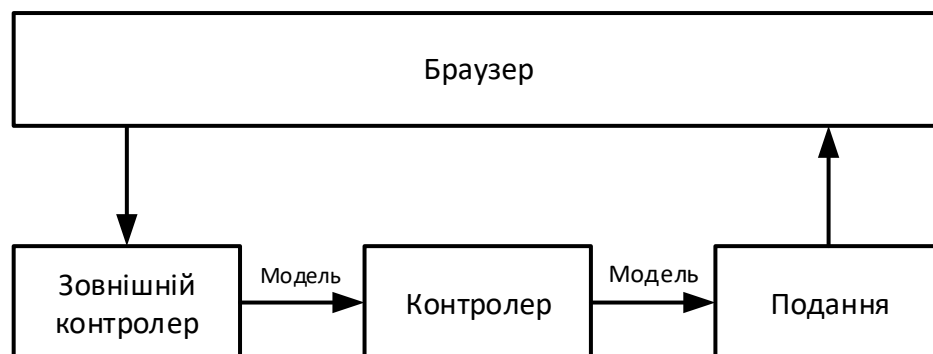


Рисунок 3.6 – Шаблон MVC для фреймворку Spring

Таким чином, шаблон MVC у контексті фреймворку Spring розділяє вебзастосунок на чотири основні компоненти:

- модель містить необхідні дані для відображення клієнтові;
- контролер реалізує бізнес-логіку додатку;
- подання відповідає за відображення потрібної інформації у браузері;
- зовнішнім контролером виступає клас `DispatcherServlet`, який перехоплює всі запити та розподіляє їх по контролерам, яким ці запити було адресовано [54].

Такий підхід допомагає зручно розділити код на три рівні та організувати логіку вебзастосунків. Використання шаблону MVC дозволяє легко та швидко модифікувати розроблюваний застосунок, оскільки кожний рівень не залежить один від одного [54].

Створення графічного інтерфейсу в Spring відбувається за допомогою Java Server Pages (JSP). Це технологія програмування на стороні сервера, яка дозволяє створювати динамічні вебсторінки, використовуючи мову розмітки HyperText Markup Language (HTML). Також JSP можна використовувати в поєднанні з контролерами, які обробляють бізнес-логіку, що робить створення вебзастосунку більш гнучким [55].

Серед існуючих інтегрованих середовищ для розробки програмного забезпечення на Java [56] було віддано перевагу IntelliJ IDEA. Це найпотужніше та найпопулярніше інтегроване середовище на сьогоднішній день, яке підтримується міжнародною компанією JetBrains та забезпечує швидку розробку, а інтелектуальні плагіни вміють аналізувати код та надавати рекомендації щодо покращення якості коду. Також IntelliJ IDEA включає інструменти для інтеграції зі всіма сучасними технологіями, що значно покращує розробку програмного забезпечення на Java [57].

Отже, для реалізації підсистеми інтелектуальної підтримки у вигляді вебзастосунку було обрано фреймворк Spring, який написано на мові програмування Java та інтегроване середовище розробки IntelliJ IDEA.

3.6.2 Вибір мови програмування для машинного навчання

Аналіз вакансій для спеціалістів з машинного навчання [58] показав які мови програмування найбільш затребувані в цій галузі. Так, в нашій країні роботодавці найбільшу перевагу віддають Python, в той час як за кордоном в окремих випадках необхідно також знати мови R, C++ та Java.

Python – це універсальна мова програмування, яка використовується для розробки програмного забезпечення в різних сферах. Спеціалісти з машинного навчання найчастіше віддають перевагу саме цій мові, бо її синтаксис менш складний у порівнянні з C++ та Java, що дозволяє швидко реалізовувати поставлені задачі. Завдяки великій кількості розроблених бібліотек, Python може виконувати всі види операцій, від попередньої обробки даних, візуалізації та статистичного аналізу до розгортання моделей машинного навчання. Тому основною перевагою є те, що Python має широкий спектр бібліотек та інструментів для вирішення задач з машинного навчання, а також велику кількість документації. Недоліком є те, що Python є інтерпретованою мовою та це уповільнює швидкість роботи розробленої програми у порівнянні з компільованими мовами [59], [60].

R – мова програмування, що була розроблена виключно для статистичної обробки та аналізу великих об'ємів даних. Перевагу цій мові зазвичай віддають професіонали, які не мають великого досвіду у програмуванні, тобто аналітики. Також R має велику кількість бібліотек, які дозволяють створювати звіти з використанням графіків та діаграм, що дають більш детальну інформацію про результати [60].

C++ є суворо типізованою мовою програмування високого рівня. Головною перевагою використання цієї мови є її продуктивність, що дозволяє забезпечити швидке виконання розробленої програми, яка має вимогу до часу. Основним недоліком є те, що C++ має складний синтаксис, що потребує великої кількості часу на вивчення цієї мови. Також необхідно враховувати той факт, що на сьогоднішній день відбувається зростання

обчислень, прискорених графічним процесором. Це призвело до створення таких бібліотек, як CUDA Python та це означає, що більшість обчислювальних навантажень для машинного навчання перекладається на графічні процесори та в результаті будь-яка перевага в продуктивності, яку може мати C++, стає все більш неактуальною [61].

Java – це об’єктно-орієнтована мова з суворою типізацією, яка найбільше використовується під час розробки вебзастосунків, хоча в останні роки набирає популярність у галузі машинного навчання. Java має відносно складний синтаксис, ніж Python, що є недоліком для новачків, але швидкість виконання програми значно швидша. Ця мова не є популярною серед розробників, оскільки вона має невелику кількість створених бібліотек та інструментів для машинного навчання, ніж Python, що є головним недоліком використання Java [62].

Таблиця 3.4 – Порівняння мов програмування для реалізації алгоритмів машинного навчання

Критерій порівняння	Python	R	C++	Java
Рік розробки	1991	1993	1985	1996
Поріг входження	низький	низький	високий	середній
Інтерпретованість	+	+	–	–
Динамічна типізація	+	+	–	–
Висока швидкість роботи	–	–	+	+
Автоматичне керування пам’яттю	+	+	–	+
Об’єктно-орієнтована парадигма	+	+	+	+
Використання в сфері машинного навчання	87%	31%	15%	15%
Велика кількість бібліотек для аналізу даних	+	+	–	–
Велика кількість бібліотек для машинного навчання	+	±	–	±
Технічна документація	+	±	±	±

Згідно результатів порівняльного аналізу (табл. 3.4) можна зробити висновок, що мова програмування Python найкраще підходить для реалізації алгоритмів машинного навчання. Оскільки ця мова є універсальною та має інтуїтивно зрозумілий синтаксис, це дозволяє швидко реалізовувати поставлені задачі. Велика кількість бібліотек для попередньої обробки даних та машинного навчання роблять Python найкращим інструментом серед його конкурентів для реалізації нейронних мереж. Хоча швидкість роботи Python не є високою та це не є вагомим недоліком, оскільки час відгуку нейронної мережі не є ключовою вимогою для розроблюваної підсистеми.

Серед популярних інтегрованих середовищ для розробки на Python [63] можна виділити PyCharm, яке вважається найкращим серед розробників. Це крос-платформне середовище розробки, яке було розроблено компанією JetBrains. PyCharm включає широкий набір функцій для відлагодження, тестування, інтелектуального завершення коду, швидкого виправлення помилок, а також має підтримку популярних фреймворків та бібліотек [64].

Отже, для реалізації нейронних мереж комплексного інтелектуального методу було прийнято рішення використовувати мову програмування Python та середовище розробки PyCharm.

3.6.3 Вибір бібліотеки для машинного навчання

Мова програмування Python пропонує широкий набір бібліотек, які полегшують реалізацію алгоритмів машинного навчання та дозволяють зосередитися на результаті, витративши якомога менше часу на розробку. Серед найчастіше використовуваних можна виділити наступні бібліотеки для машинного навчання: Scikit-learn, PyTorch, Keras та Tensorflow [65].

Scikit-learn – це бібліотека, яка надає базові алгоритми для машинного навчання, використовуваних для кластеризації, класифікації та регресії. Найчастіше ця бібліотека використовується новачками в даній сфері та у невеликих дослідницьких проєктах [66].

PyTorch – це відносно молода бібліотека, що розроблена компанією Facebook та відома своєю простотою використання та гнучкістю. Підходить для створення та навчання глибоких нейронних мереж з можливістю створення власних моделей та алгоритмів навчання. Також вона підтримує використання графічного процесора для прискорення навчання [66], [67].

TensorFlow – це бібліотека з відкритим вихідним кодом, що була розроблена корпорацією Google та надає інструменти для глибокого навчання. Вона добре відома своєю документацією, можливостями розгортання та масштабування моделей. Вона використовується, коли необхідна більша функціональність та гнучкість для побудови нейронних мереж, а також дозволяє використовувати можливості графічного процесора під час навчання [66], [67].

Keras – це бібліотека, що орієнтована на глибоке навчання та яка побудована на основі іншої бібліотеки TensorFlow. Так, Keras робить акцент на модульності, зручності використання та розширюваності. Це дозволяє зменшити складність коду, надаючи API високого рівня, незважаючи на те, що це платформа глибокого навчання, яка найкраще підходить для швидкого експериментування з глибокими нейронними мережами [66], [67].

Таблиця 3.5 – Порівняння бібліотек машинного навчання

Критерій порівняння	Scikit-learn	PyTorch	Keras	TensorFlow
Рівень API	високий	низький	високий	низький
Простота використання	легко	складно	середньо	складно
Підтримка глибинного навчання	–	+	+	+
Використання графічного процесору	–	+	+	+
Технічна документація	+	±	+	+
Розмір ком'юніті на «Stack Overflow»	26 тис.	19 тис.	41 тис.	79 тис.

В ході порівняльного аналізу бібліотек для реалізації алгоритмів машинного навчання (табл. 3.5) було відданого перевагу Keras, тому що ця бібліотека підтримує методи глибинного навчання, а використання графічного процесору дозволить значно пришвидшити навчання розроблюваних моделей. Оскільки Keras надає високий рівень API, це дозволить швидко створити та обрати найкращу модель для прогнозування. Великий об'єм документації та невеликий розмір ком'юніті дозволить швидко освоїти цю бібліотеку та вирішити усі виникаючі проблеми.

Отже, для реалізації штучних нейронних мереж запропонованого комплексного інтелектуального методу варто використовувати бібліотеку Keras.

3.7 Висновки до розділу

Відповідно до поставленої мети роботи, було розроблено функціональні та нефункціональні вимоги програмного забезпечення (рис. 3.1), вдосконалено архітектуру досліджуваної СДА (рис. 3.3) та було запропоновано розробити комплексний інтелектуальний метод контролю енергоспоживання на основі прогнозування генерації електроенергії з максимальним використанням альтернативних джерел та оптимізації розкладу використання електроприладів. Такий підхід дозволить підвищити ефективність користування електроприладами в реальній СДА. Алгоритм реалізації комплексного інтелектуального методу представлено у вигляді діаграми діяльності (рис. 3.5).

Для реалізації комплексного інтелектуального методу необхідно забезпечити завантаження даних прогнозу погоди. Порівняльний аналіз метеосервісів (табл. 3.1) дозволив обрати «Visual Crossing», оскільки він надає безкоштовний доступ до історичних даних, які будуть використовуватися для навчання нейронної мережі, а також до погодинного

прогнозу погоду на декілька діб, що необхідно для прогнозування генерації електроенергії.

Для можливості збереження історичних показників енергоспоживання у платформі OpenHAB, які необхідні для навчання нейронної мережі запропонованого комплексного інтелектуального методу, необхідно обрати нову СКБД. Згідно порівняльного аналізу СКБД (табл. 3.2), що підтримуються OpenHAB, було віддано перевагу MongoDB. Оскільки дані щодо енергоспоживання зберігаються кожні вісім секунд, саме ця СКБД забезпечить зберігання великих об'ємів даних без втрати продуктивності.

Згідно з визначеними функціональними та нефункціональними вимогами (рис. 3.1), було виявлено, що підсистема інтелектуальної підтримки має бути реалізована у вигляді вебзастосунку. Порівняння вебфреймворків (табл. 3.3) дозволило обрати фреймворк Spring, тому що він має велику кількість модулів та бібліотек, які значно полегшують та пришвидшують написання коду. Оскільки обраний фреймворк написано на мові програмування Java, тому саме ця мова буде використана під час розробки підсистеми інтелектуальної підтримки. Розробка вебзастосунків з використанням фреймворку Spring передбачає використання шаблону MVC, а створення графічного інтерфейсу за допомогою технології JSP. Серед існуючих інтегрованих середовищ було віддано перевагу IntelliJ IDEA.

Для реалізації алгоритмів машинного навчання запропонованого комплексного інтелектуального методу було виконано порівняння мов програмування, що використовуються в цій галузі (табл. 3.4). Таким чином, було віддано перевагу мові Python, оскільки вона є універсальною та має інтуїтивно зрозумілий синтаксис, що дозволяє швидко реалізовувати поставлені задачі. Як інтегроване середовище для розробки на Python було обрано PyCharm.

В ході порівняльного аналізу бібліотек для реалізації алгоритмів машинного навчання (табл. 3.5) було відданого перевагу Keras, тому що вона

надає високий рівень API та це дозволить швидко створити та обрати найкращу модель для прогнозування.

Обраний стек технологій дозволить розробити сучасне програмне забезпечення, що буде відповідати усім поставленим вимогам та дозволить надавати рекомендації мешканцям будинку щодо ефективного користування електроприладами, використовуючи альтернативні джерела енергії та якнайменше залежати від тарифів на електроенергію, що на сьогодні є актуальною проблемою особливо в умовах нестабільної економічної та політичної ситуації в країні.

4 КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Основні рішення щодо модульної побудови підсистеми

Розроблювана підсистема складається декількох модулів, які взаємодіють один з одним для задоволення певних потреб користувача. Модульна побудова дозволяє легко модифікувати підсистему у майбутньому, а також додавати нові функції, не переписуючи весь програмний код.

Структурно-функціональну схему підсистеми інтелектуальної підтримки зображено на рисунку 4.1.

Вебзастосунок надає користувачеві графічний інтерфейс, за допомогою якого він може отримати доступ до всіх функцій підсистеми, а також керувати ними та отримувати важливі повідомлення, використовуючи браузер.

До складу вебзастосунку входять наступні модулі:

- модуль моніторингу показників, який призначено для аналізу даних енергоспоживання в реальному часі та у разі перевищення споживання електроенергії із зовнішньої мережі надсилає повідомлення користувачу, використовуючи графічний інтерфейс;

- модуль редагування електроприладів дозволяє користувачу додавати, модифікувати, а також видаляти потужні електроприлади, що знаходяться в будинку;

- модуль оптимізації розкладу призначено для генерації оптимального погодинного розкладу роботи побутових приладів, використовуючи список електроприладів, якими користувач бажає користуватися та генерацію електроенергії, що була спрогнозована.

Модуль прогнозування є окремою програмою, що була написана на мові програмування Python з використанням бібліотеки Keras. Цей модуль призначено для передбачення генерації електроенергії на наступний день, використовуючи прогноз погоди, який завантажується з метеосервісу та історичні дані енергоспоживання, що зберігаються у базі даних. Модуль

прогнозування викликається вебзастосунком у разі необхідності та використовує результати прогнозування під час генерації оптимального розкладу.

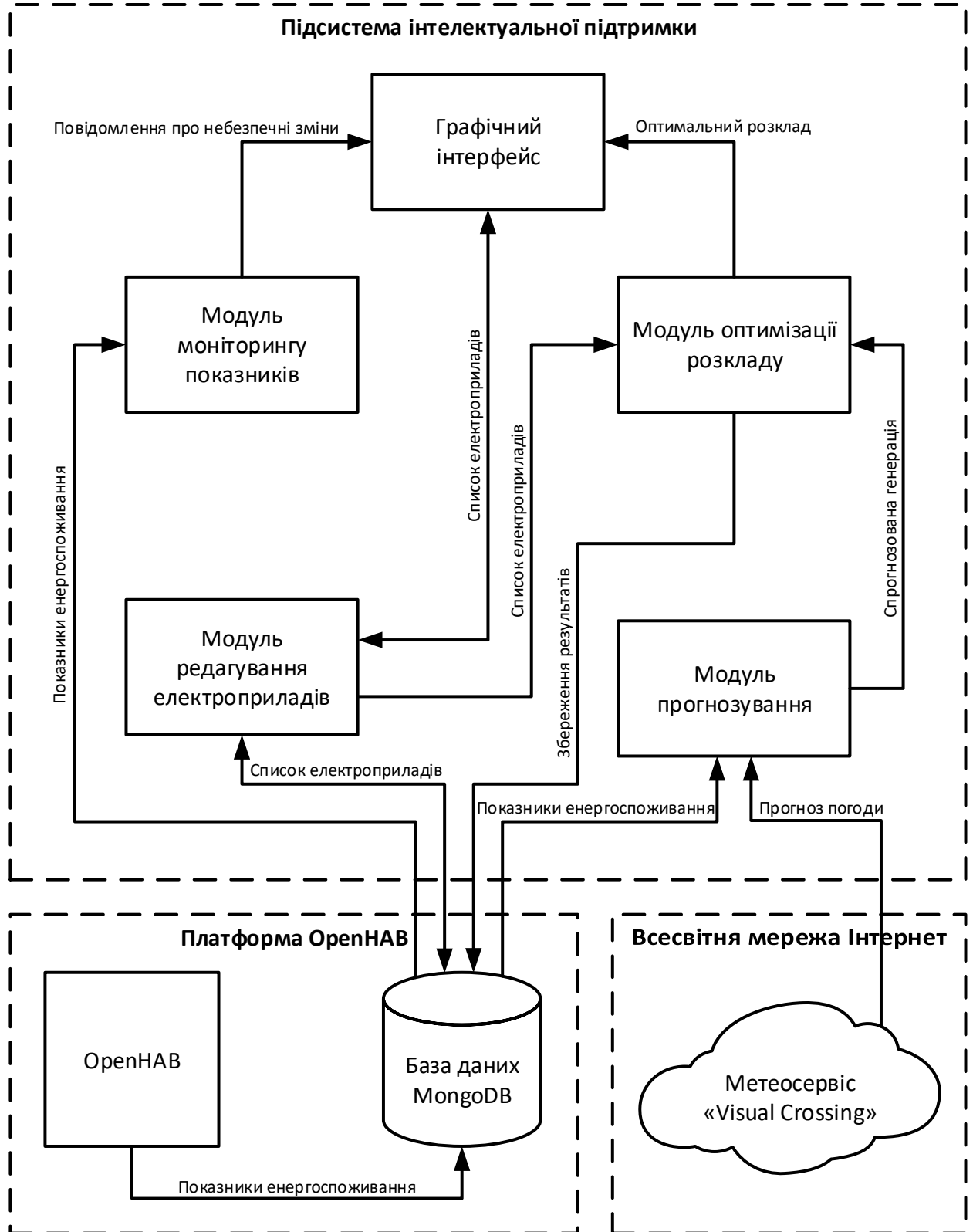


Рисунок 4.1 – Структурно-функціональна схема розроблюваної підсистеми

Розроблена підсистема інтелектуальної підтримки складається з двох проєктів, а саме: з вебзастосунку та модулю прогнозування.

Загальну структуру вебзастосунку, який розроблено з використанням фреймворку Spring наведено на рисунку 4.2.

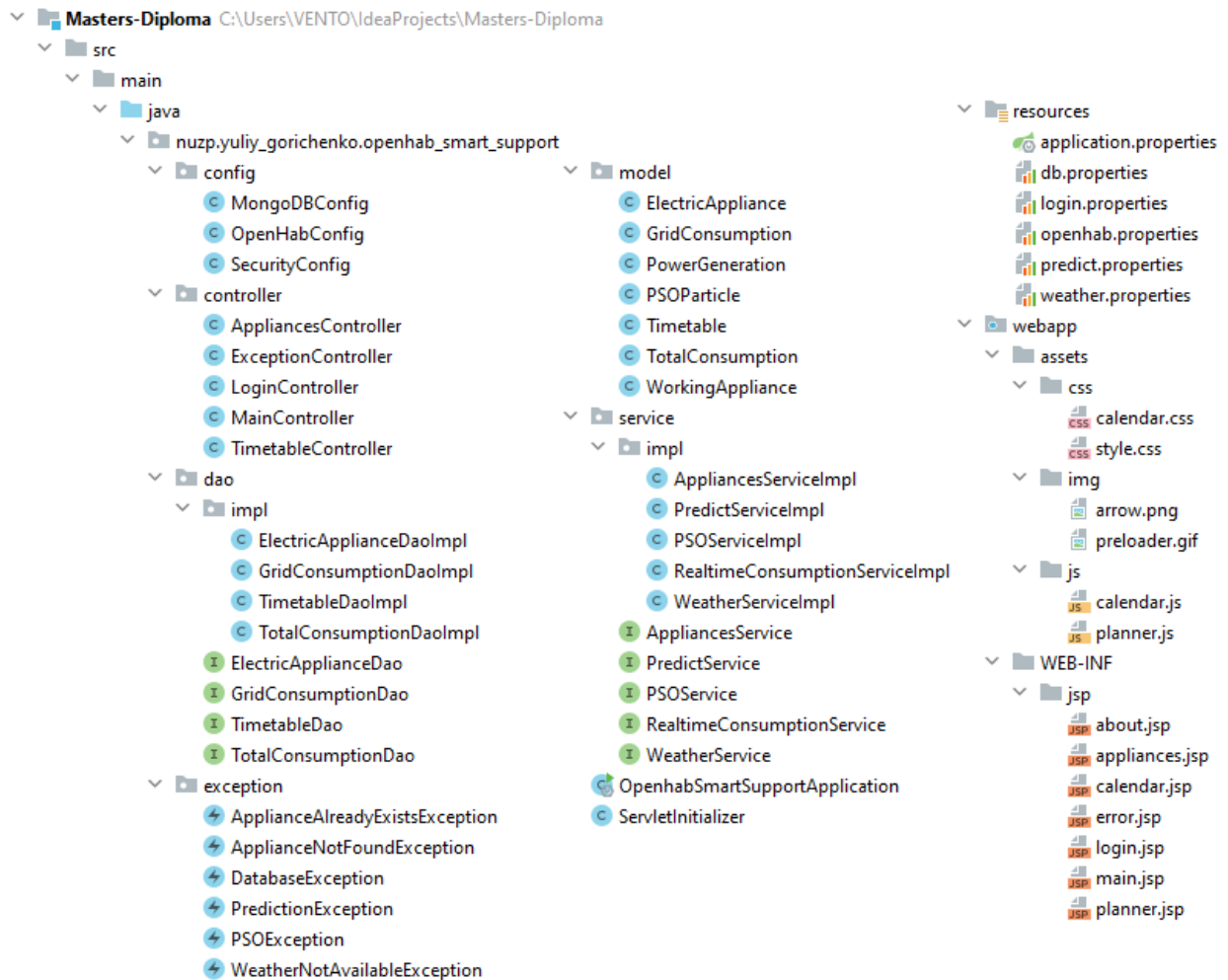


Рисунок 4.2 – Структура вебзастосунку підсистеми інтелектуальної підтримки

Кореневою директорією вебзастосунку є папка з назвою «src», яка містить увесь вихідний код, що написано на мові програмування Java.

Пакет «config» містить класи, які реалізують зчитування файлів конфігурації під час запуску вебзастосунку.

Пакет «controller» містить реалізацію контролерів, які відповідають за відображення кожної сторінки вебзастосунку та які оброблюють запити, що надходять з браузера від користувача.

Пакет «dao» містить інтерфейси та класи, що реалізують шаблон доступу до даних об'єкту [68]. Це дозволяє ізолювати прикладний рівень від рівня збереження інформації про об'єкт у базі даних.

Пакет «exception» включає класи, що використовуються для реалізації механізму виведення помилок, які можуть трапитися під час роботи.

Пакет «model» містить класи, що описують моделі об'єктів, які використовуються для реалізації шаблону MVC.

Пакет «service» містить інтерфейси та класи, що реалізують бізнес-логіку всіх модулів підсистеми інтелектуальної підтримки. Так, цей пакет містить класи для редагування списку електроприладів, прогнозування, оптимізації, моніторингу показників енергоспоживання та доступу до метеосервісу.

Класи «OpenhabSmartSupportApplication» та «ServletInitializer» є службовими файлами фреймворку Spring, які запускають вебзастосунок.

Директорія «resources» включає в себе файли, які містять налаштування вебзастосунку, а саме: підключення до бази даних, логін та пароль адміністратора, інтеграція з платформою OpenHAB та метеосервісом.

Директорія «webapp» містить файли, що використовуються для відображення необхідних сторінок користувачу. Так, в директорії «assets» знаходяться каскадні таблиці стилів, скрипти для вебсторінок на мові програмування JavaScript, а також деякі програмні зображення. Директорія «jsp» містить JSP вебсторінки, які реалізують їх структуру.

Другий проєкт підсистеми інтелектуальної підтримки виконує роль прогнозування генерації електроенергії, що написано на мові програмування Python, структуру якого зображено на рисунку 4.3.

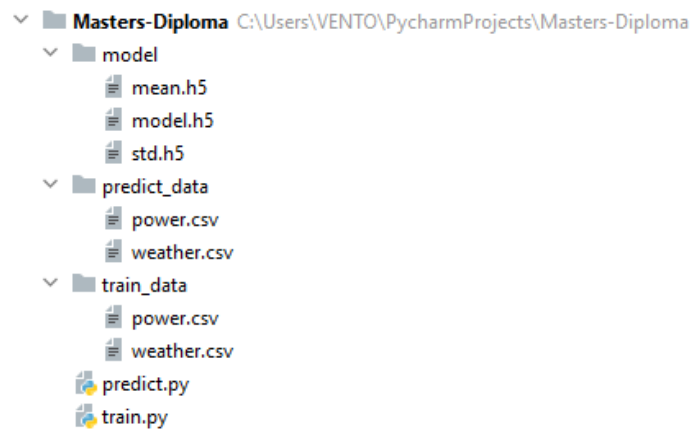


Рисунок 4.3 – Структура модулю прогнозування підсистеми інтелектуальної підтримки

Файл «train.py» є виконуваним файлом, який забезпечує навчання нейромережевої моделі.

Директорія «train_data» містить два файли, а саме: «weather.csv» та «power.csv», які містять історичні дані для навчання та тестування моделі. Ці файли формуються заздалегідь вебзастосунком, завантажуючи дані з метеосервісу та бази даних.

Директорія «model» зберігає статичні дані про навчену модель та файли з її вагами. Під час прогнозування, ваги моделі завантажуються саме з цієї директорії.

Файл «predict.py» є виконуваним файлом, який забезпечує прогнозування генерації електроенергії.

Директорія «predict_data» також містить два файли. Файл «weather.csv» формується заздалегідь вебзастосунком та містить прогноз погоди на наступний день. Файл «power.csv» є результатом роботи прогнозування, який містить інформацію про генерацію електроенергії.

Взаємодію між вебзастосунком та модулем прогнозування наведено на рисунку 4.4.

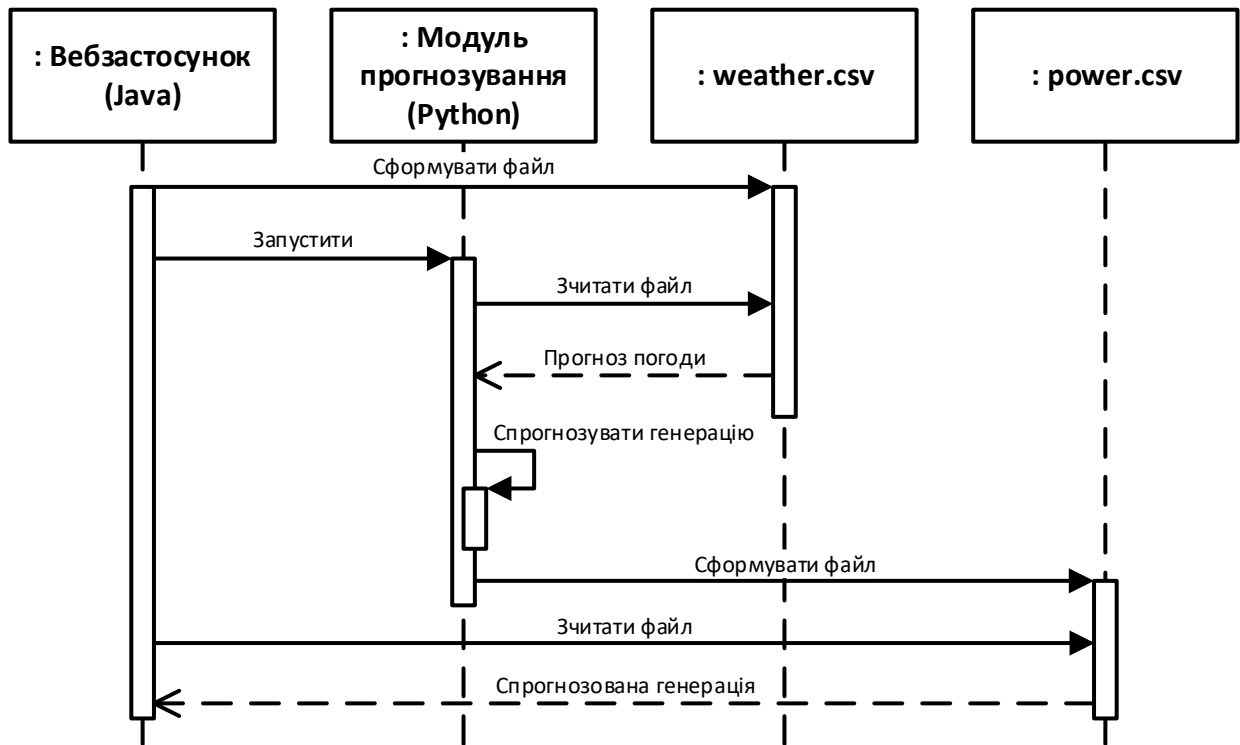


Рисунок 4.4 – Діаграма послідовності запуску модуля прогнозування

Перед початком прогнозування, вебзастосунок має підготувати вхідні дані для нейромережевої моделі. Використовуючи метеосервіс, створюється файл «weather.csv», який містить прогноз погоди на наступний день. Після цього вебзастосунок запускає модуль прогнозування, а саме файл «predict.py», який в свою чергу зчитує заздалегідь сформований файл з прогнозом погоди. Далі навчена модель виконує прогнозування генерації електроенергії та записує результати у файл «power.csv». Після завершення роботи модуля прогнозування, вебзастосунок зчитує файл, що містить спрогнозовані значення генерації та використовує їх під час створення оптимального розкладу користування побутовими приладами.

Аналогічним чином працює навчання та тестування нейромережевої моделі. Спочатку вебзастосунок створює файл «weather.csv», який містить історичні дані прогнозу погоди, а також файл «power.csv», який включає історичні дані генерації електроенергії, що отримуються з бази даних. Після цього вебзастосунок запускає виконуваний файл «train.py», який здійснює

навчання нейромережевої моделі, використовуючи заздалегідь сформовані історичні дані.

Оскільки історичні дані, що використовуються нейронною мережею для навчання постійно поповнюються, то для збільшення точності прогнозування, підсистема інтелектуальної підтримки здійснює перенавчання нейромережевої моделі на нових даних раз в тиждень в автоматичному режимі.

4.2 Основні рішення щодо представлення даних

Для роботи вебзастосунку необхідно забезпечити збереження усіх необхідних даних у СКБД, а саме: історичних даних енергоспоживання (загальне споживання та споживання з мережі), списку електроприладів, що використовуються у будинку та створений оптимальний розклад на кожен день. Так, база даних MongoDB налічує чотири колекції, які зберігають цю інформацію (рис. 4.5).

Колекція – це група документів у СКБД MongoDB. Вони зберігаються у форматі JSON та це означає, що документи в колекції можуть мати різні поля. Колекція є еквівалентом таблиці в системі реляційних баз даних [69].

Для збереження даних щодо енергоспоживання з мережі та загального споживання використовуються колекції «Item1» та «Item7» відповідно, які були автоматично створені платформою OpenHAB. Ці дані надходять з модулів PZEM, зберігаються у базі даних кожні вісім секунд та включають наступні поля:

- ідентифікатор, який є унікальним для кожного запису;
- дата та час;
- значення енергоспоживання (Вт).

Collections				
Create collection View Sort by Documents				
Item1				
Storage size: 20.73 MB	Documents: 922 K	Avg. document size: 51.00 B	Indexes: 1	Total index size: 10.36 MB
Item7				
Storage size: 19.96 MB	Documents: 893 K	Avg. document size: 51.00 B	Indexes: 1	Total index size: 9.60 MB
ElectricAppliances				
Storage size: 20.48 kB	Documents: 27	Avg. document size: 172.00 B	Indexes: 1	Total index size: 36.66 kB
Timetable				
Storage size: 20.48 kB	Documents: 3	Avg. document size: 2.42 kB	Indexes: 1	Total index size: 36.66 kB

Рисунок 4.5 – Список колекцій у базі даних MongoDB

Структуру документу споживання електроенергії, що зберігається у базі даних представлено у вигляді діаграми класів на рисунку 4.6.

Item
- id : String - time : Long - value : Double

Рисунок 4.6 – Структура документу для збереження даних енергоспоживання

Для збереження списку електроприладів, що знаходяться у будинку використовується колекція «ElectricAppliances», яка має наступну структуру (рис. 4.7):

- ідентифікатор, який є унікальним для кожного приладу;
- назва побутового приладу;
- номінальна потужність (Вт);
- кількість годин роботи електроприладу.

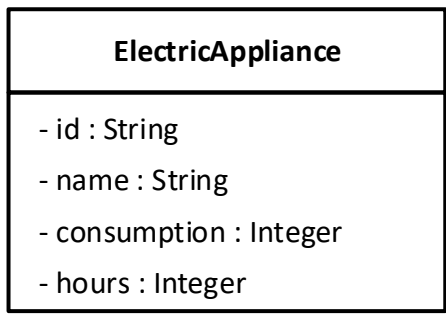


Рисунок 4.7 – Структура документа для збереження списку електроприладів

Збереження оптимального розкладу на кожен день реалізовано у колекції «Timetable», яка включає наступні поля (рис. 4.8):

- унікальний ідентифікатор;
- дата розкладу;
- список роботи запланованих електроприладів (ідентифікатор, назва, номінальна потужність, пріоритет, години роботи побутового приладу);
- спрогнозована генерація (ідентифікатор, дата та час, потужність генерації).

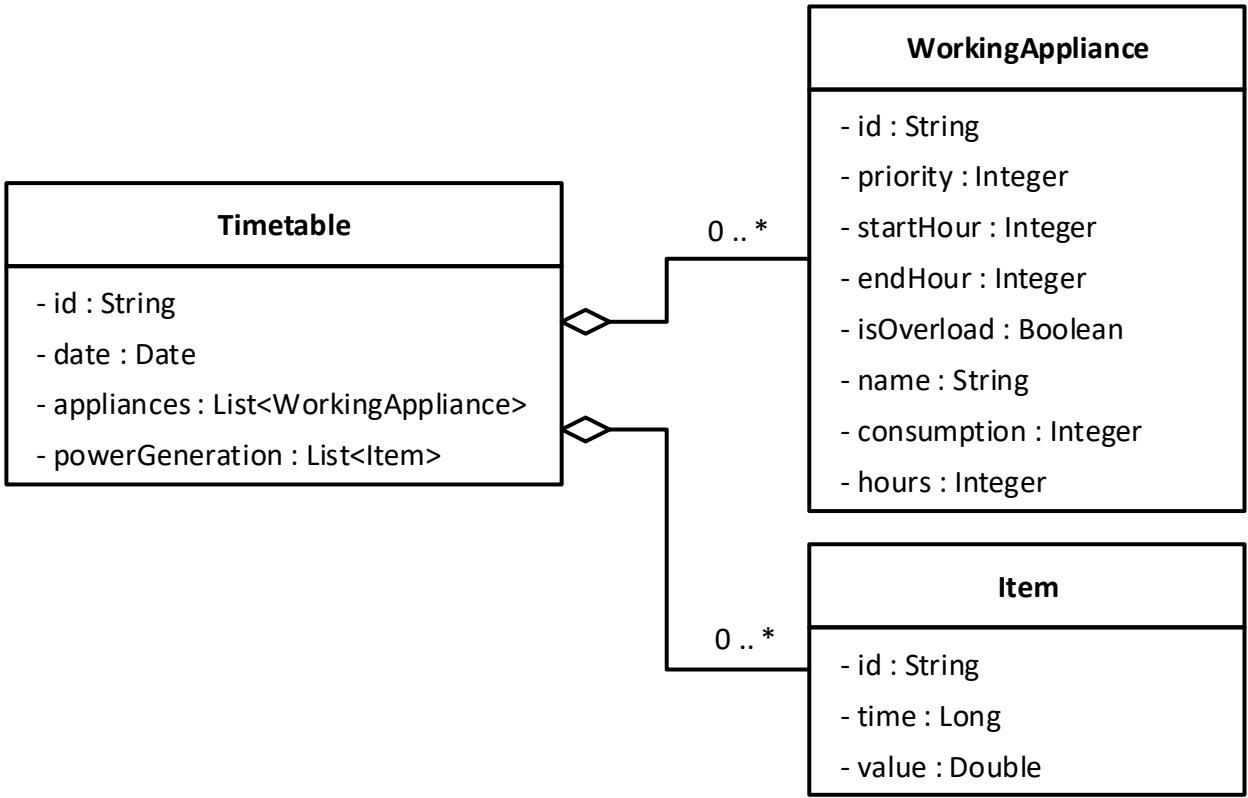


Рисунок 4.8 – Структура документа для збереження оптимального розкладу

Модуль прогнозування використовує файли формату Comma-separated values (CSV) для роботи з даними.

CSV – це текстовий файл, який зберігає табличні дані, що розділені комами [70]. Перевагою використання такого формату є те, що він підтримується всіма мовами програмування та це дозволяє легко реалізувати взаємодію між вебзастосунком, який написано на мові програмування Java та модулем прогнозування, що написано на Python.

Дані про прогноз погоди, які завантажуються з метеосервісу «Visual Crossing» зберігаються у файлі «weather.csv». Отриманий прогноз погоди включає різноманітні атрибути, які представлено у таблиці 4.1.

Таблиця 4.1 – Атрибути прогнозу погоди з метеосервісу [71]

Назва атрибуту	Опис	Одиниця вимірювання
tempmax	Максимальна температура	°C
tempmin	Мінімальна температура	°C
temp	Середня температура	°C
dew	Точка роси	°C
feelslike	Тепловий індекс	°C
precip	Кількість опадів	мм
precipprob	Ймовірність опадів	%
preciptype	Тип опадів	–
snow	Кількість снігу	см
snowdepth	Глибина снігу	см
windspeed	Швидкість вітру	км/год
windgust	Порив вітру	км/год
winddir	Напрямок вітру	°
visibility	Видимість	км
cloudcover	Хмарність	%
humidity	Відносна вологість	%
pressure	Тиск	мм рт. ст.
solarrradiation	Сонячна радіація	Вт/м ²
solarenergy	Потужність сонячного випромінювання	МДж/м ²
uvindex	Ультрафіолетовий індекс	–

Спрогнозована генерація електроенергії зберігається у файлі «power.csv», який включає наступні дані (рис. 4.9):

- дата та час;
- потужність генерації (Вт).

	datetime	power
0	2022-10-23T00:00:00	29.754204
1	2022-10-23T01:00:00	33.095028
2	2022-10-23T02:00:00	29.948603
3	2022-10-23T03:00:00	29.45277
4	2022-10-23T04:00:00	32.581596
5	2022-10-23T05:00:00	31.30991
6	2022-10-23T06:00:00	35.236927
7	2022-10-23T07:00:00	33.319096
8	2022-10-23T08:00:00	275.0755043
9	2022-10-23T09:00:00	409.5880872
10	2022-10-23T10:00:00	940.0503062
11	2022-10-23T11:00:00	1459.53043
12	2022-10-23T12:00:00	2018.943331
13	2022-10-23T13:00:00	2252.682387
14	2022-10-23T14:00:00	2876.194856
15	2022-10-23T15:00:00	2825.599184
16	2022-10-23T16:00:00	1207.965085
17	2022-10-23T17:00:00	323.1250682
18	2022-10-23T18:00:00	112.5072917
19	2022-10-23T19:00:00	60.109768
20	2022-10-23T20:00:00	53.0758
21	2022-10-23T21:00:00	8.815404
22	2022-10-23T22:00:00	13.368188
23	2022-10-23T23:00:00	11.534533

Рисунок 4.9 – Приклад вмісту файлу «power.csv»

4.3 Основні розроблені алгоритми

4.3.1 Алгоритм оптимізації розкладу роботи електроприладів

Для оптимізації розкладу роботи побутових приладів було прийнято рішення використовувати метод рою часток (розд. 2.5).

Метод рою часток – це метаевристичний алгоритм оптимізації, який було вперше запропоновано у 1995 році Джеймсом Кеннеді і Расселом Еберхартом [72]. Ідея алгоритму була запозичена з досліджень поведінки пташиних зграй, де кожне рішення задачі оптимізації в області пошуку вважається «птахом» та у контексті алгоритму називається «частинкою», а сукупність рішень називається «роєм», у якому зібрані усі «частинки». На початку вони розташовані випадковим чином по всій області пошуку та кожна така «частинка» має дві характеристики: її положення та випадковий

вектор швидкості. У кожній точці, де побувала «частинка», розраховується значення цільової функції. Водночас кожна «частинка» запам'ятовує найкраще значення цільової функції, яке вона знайшла особисто, а також найкращу позицію серед усього «рою». Так, на кожній ітерації «частинки» змінюють свою швидкість та позицію, щоб наблизитися до точки, яка в момент ітерації є найкращою. Таким чином, через деяку кількість ітерацій «частинки» повинні зібратися поблизу глобального екстремуму, який можна вважати рішенням проблеми [73].

Задачею створення оптимального розкладу користування побутовими приладами є генерація такого розкладу, у якому буде використано якнайбільше альтернативної електроенергії, тобто перевищення споживання електроенергії від згенерованої має бути мінімальним. До того ж оптимальний погодинний розклад повинен забезпечувати високий рівень комфорту, тому користувач має змогу обрати список електроприладів, які планує використовувати, а також вказати зручні для нього часові інтервали користування цим приладом. Таким чином, алгоритм оптимізації запропонує найкращий розклад, підлаштувавшись під вимоги та звички мешканця.

Оскільки сонячні електростанції та вітрогенератори надають обмежену кількість електроенергії, було запропоновано модифікувати алгоритм рою часток та ввести систему пріоритетів. Під пріоритетом розуміється число, яке може приймати значення від одного до п'яти, де п'ять означає найвищий пріоритет. Це дозволить користувачу обрати найважливіші для нього побутові прилади та у разі недостатньої генерації електроенергії, алгоритм запропонує не використовувати ті, що мають найменший пріоритет.

Таким чином, вхідними даними алгоритму рою часток є спрогнозована генерація на 24 години та список побутових приладів, кожен з яких має часові інтервали та кількість годин роботи, а також пріоритет.

Алгоритм оптимізації розкладу роботи побутових приладів на основі методу рою часток зображено на рисунку 4.10.

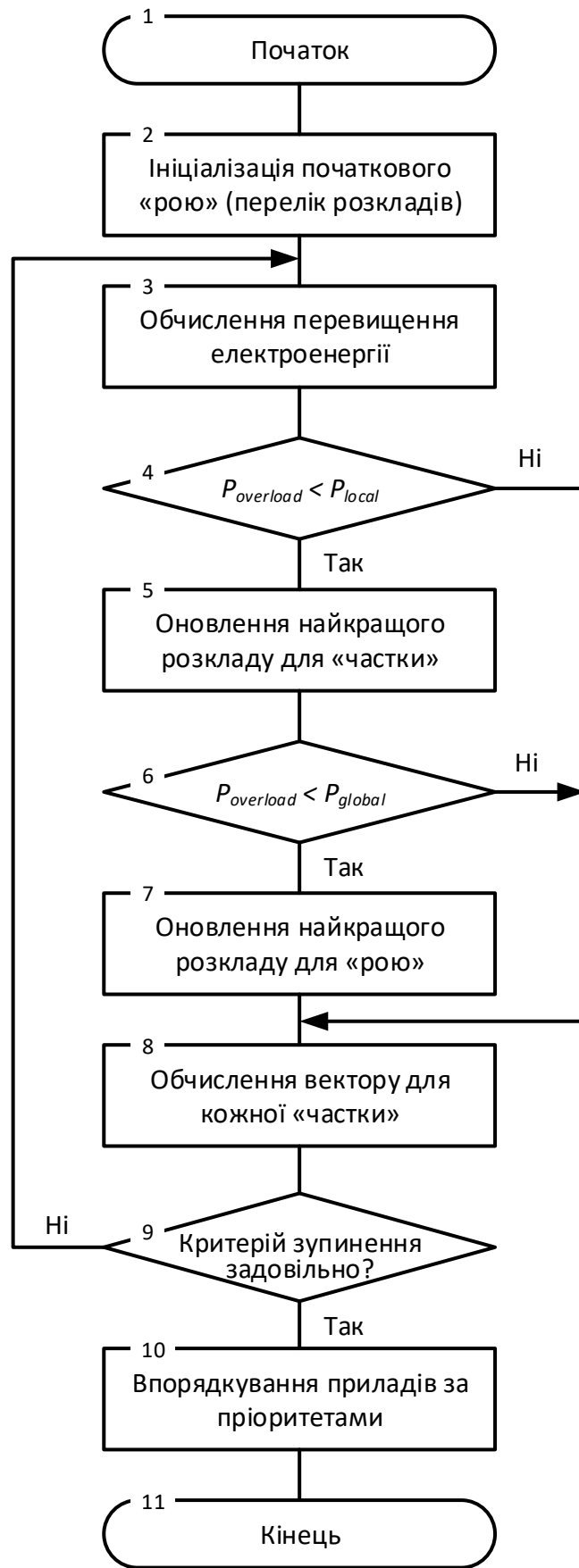


Рисунок 4.10 – Алгоритм оптимізації розкладу роботи електроприладів на основі методу рою часток

Блок 1. Початок.

Блок 2. Ініціалізація початкового «рою», тобто створення розкладу для кожної «частки» та встановлення випадкового значення початку роботи всіх електроприладів.

Блок 3. Обчислення перевищення електроенергії $P_{overload}$ від згенерованої для кожного розкладу виконується за формулою (4.1):

$$P_{overload} = \sum_{h=1}^{H=24} \left(\sum_{i=1}^D P_{ih} \right) - G_h \quad (4.1)$$

де P_{ih} – номінальна потужність i -го електроприладу, що працює в h годині;

G_h – спрогнозована генерація електроенергії в h годині;

D – кількість увімкнених електроприладів для поточної години;

H – кількість годин доби.

Блок 4. Якщо обчислене перевищення електроенергії $P_{overload}$ менше, ніж найкраще для «частки» P_{local} , то перехід до наступного блоку, в іншому випадку – до блоку 8.

Блок 5. Оновлення найкращого розкладу для «частки», тобто збереження розкладу, який має найменше обчислене перевищення електроенергії.

Блок 6. Якщо обчислене перевищення електроенергії $P_{overload}$ менше, ніж найкраще для «рою» P_{global} , то перехід до наступного блоку, в іншому випадку – до блоку 8.

Блок 7. Оновлення найкращого розкладу для всього «рою», тобто збереження розкладу, який має найменше обчислене перевищення електроенергії.

Блок 8. Обчислення вектору v_i для кожної «частки» відбувається за формулою (4.2):

$$v_i = w \times v_i + c_1 \times r_1 (P_{local} - x_i) + c_2 \times r_2 (P_{global} - x_i) \quad (4.2)$$

де v_i – значення вектору для i -ї «частки»;

w – коефіцієнт інерції;

c_1 – когнітивний коефіцієнт;

c_2 – соціальний коефіцієнт;

r_1 та r_2 – випадкові числа в інтервалі $[0, 1]$;

x_i – поточне рішення для i -ї «частки»;

P_{local} – найкраще рішення для «частки»;

P_{global} – найкраще рішення для «рою».

Блок 9. Якщо критерій зупинення було виконано, тобто кількість ітерацій дорівнює максимальному значенню, то перехід до наступного блоку, в іншому випадку – до блоку 3.

Блок 10. Впорядкування побутових приладів за пріоритетами та визначення тих приладів, які можна не використовувати у разі недостатньої генерації електроенергії.

Блок 11. Кінець.

4.3.2 Алгоритм моніторингу показників енергоспоживання

Алгоритм моніторингу показників енергоспоживання забезпечує цілодобовий контроль електромережі та у разі перевищення споживання електроенергії із зовнішньої мережі повідомляє про це власника, що дозволяє миттєво відреагувати на виникаючі події.

Алгоритм моніторингу показників енергоспоживання зображено на рисунку 4.11.

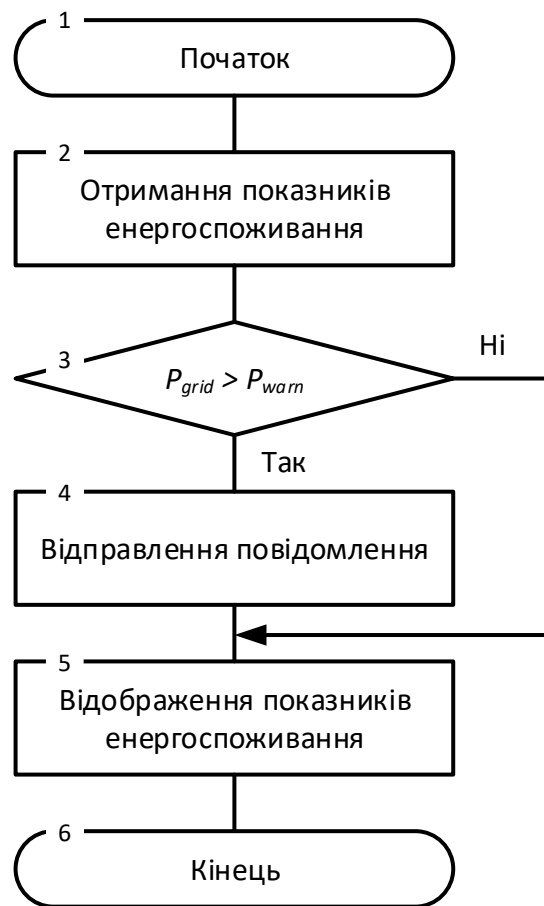


Рисунок 4.11 – Алгоритм моніторингу показників енергоспоживання

Блок 1. Початок.

Блок 2. Отримання показників енергоспоживання відбувається кожні вісім секунд, інтегруючись з платформою OpenHAB.

Блок 3. Якщо навантаження із зовнішньої електромережі P_{grid} перевищує ліміт навантаження P_{warn} , який було встановлено в налаштуваннях підсистеми, то перехід до наступного блоку, в іншому випадку – до блоку 5.

Блок 4. Відправлення повідомлення користувачу про високе споживання електроенергії із зовнішньої мережі, використовуючи графічний інтерфейс.

Блок 5. Відображення показників енергоспоживання, що були отримані з платформи OpenHAB, використовуючи графічний інтерфейс.

Блок 6. Кінець.

4.4 Опис особливостей програмної реалізації підсистеми

Розроблений вебзастосунок підсистеми інтелектуальної підтримки було реалізовано за допомогою фреймворку Spring, який поставляється з різноманітними модулями. Це робить даний фреймворк універсальним, бо розробник має можливість обрати тільки потрібні йому модулі для реалізації поставленої задачі [74].

Під час створення вебзастосунку було використано наступні модулі:

- Spring Core – базовий модуль, який реалізує основні функції фреймворку та є обов'язком для використання;
- Spring Data – модуль, який надає інструменти для взаємодії з даними в реляційних та нереляційних базах даних;
- Spring Web – модуль, який надає функції для створення сервлетів, які приймають запити від користувачів та формують відповідь у вигляді вебсторінки;
- Spring Security – модуль, який надає інструменти для автентифікації, що дозволяє захистити вебзастосунок від несанкціонованого доступу.

Для покращення якості коду було підключено бібліотеку Lombok, яка поширюється з відкритим вихідним кодом, а ліцензія MIT дозволяє використовувати її у будь-яких проєктах [75].

Lombok – це бібліотека для мови програмування Java, яка дозволяє скоротити шаблонний програмний код. Таким чином, ця бібліотека позбавить розробника від написання конструкторів, модифікаторів доступу та стандартних методів, лише додавши необхідну анотацію. Це дозволить підтримувати вихідний код компактним, чистим та зручним для сприйняття іншими розробниками [76].

В ході розробки вебзастосунку було розроблено інтерфейси та класи, які можна розділити на такі категорії:

- класи для зчитування файлів конфігурації: MongoDBConfig, OpenHabConfig, SecurityConfig;
- класи, що реалізують логіку контролерів: AppliancesController, TimetableController, ExceptionController, LoginController, MainController;
- інтерфейси для взаємодії з базою даних: ElectricApplianceDao, GridConsumptionDao, TimetableDao, TotalConsumptionDao;
- класи, що описують моделі даних для реалізації шаблону MVC: ElectricAppliance, WorkingAppliance, GridConsumption, PowerGeneration, PSOParticle, TotalConsumption, Timetable;
- інтерфейси для реалізації бізнес-логіки всіх модулів підсистеми: AppliancesService, PredictService, WeatherService, PSOService, RealtimeConsumptionService.

Клас MongoDBConfig реалізує зчитування файлу конфігурації «db.properties» та створення підключення до СКБД MongoDB. Поля та методи класу наведено в таблиці 4.2.

Таблиця 4.2 – Поля та методи класу MongoDBConfig

Назва	Опис
private:	
String mongoHost;	Адреса бази даних
long mongoPort;	Порт бази даних
String mongoUser;	Користувач бази даних
String password;	Пароль бази даних
String mongoName;	Назва бази даних
public:	
mongoTemplate();	Ініціалізація підключення до MongoDB

Клас OpenHabConfig реалізує зчитування файлу конфігурації «openhab.properties», який містить налаштування для інтеграції з платформою OpenHAB. Поля класу наведено в таблиці 4.3.

Таблиця 4.3 – Поля класу OpenHabConfig

Назва	Опис
private:	
String gridConsumptionItem;	Назва колекції в СКБД для збереження статистики споживання електроенергії із зовнішньої мережі
String totalConsumptionItem;	Назва колекції в СКБД для збереження статистики загального споживання електроенергії
long writeInterval;	Частота запису даних в СКБД
Integer overloadThreshold;	Поріг попередження про високе використання електроенергії із зовнішньої мережі

Клас SecurityConfig реалізує зчитування файлу конфігурації «login.properties» та ініціалізує налаштування автентифікації вебзастосунку. Поля та методи класу наведено в таблиці 4.4.

Таблиця 4.4 – Поля та методи класу SecurityConfig

Назва	Опис
private:	
String username;	Логін адміністратора
String password;	Пароль адміністратора
public:	
configureGlobal();	Додавання адміністратора
configure();	Ініціалізація налаштування автентифікації

Клас AppliancesController реалізує логіку контролера, який приймає запити для додавання, видалення та редагування списку побутових приладів. Методи класу наведено в таблиці 4.5.

Таблиця 4.5 – Методи класу AppliancesController

Назва	Опис
public:	
listAppliances();	Сторінка для перегляду списку всіх електроприладів
addAppliance();	Додавання нового електроприладу
editAppliance();	Редагування електроприладу
deleteAppliance();	Видалення електроприладу

Клас TimetableController реалізує логіку контролера, який приймає запити для перегляду розкладу, створення нового розкладу користування побутовими приладами, а також обробляє запити для моніторингу показників енергоспоживання в реальному часі. Методи класу наведено в таблиці 4.6.

Таблиця 4.6 – Методи класу TimetableController

Назва	Опис
public:	
timetableToday();	Сторінка з розкладом на сьогодні
timetableTomorrow();	Сторінка з розкладом на завтра
timetableAfterTomorrow();	Сторінка з розкладом на післязавтра
newTimetable();	Сторінка для створення нового розкладу
createTimetable();	Створення нового розкладу
realtimeData();	Моніторинг показників енергоспоживання в реальному часі

Клас ExceptionController реалізує контролер, який відправляє повідомлення користувачу у разі виникнення помилок. Методи класу наведено в таблиці 4.7.

Таблиця 4.7 – Методи класу ExceptionController

Назва	Опис
public:	
showError();	Виведення повідомлення про помилку

Клас LoginController реалізує контролер, який виконує функцію автентифікації. Методи класу наведено в таблиці 4.8.

Таблиця 4.8 – Методи класу LoginController

Назва	Опис
public:	
loginPage();	Сторінка автентифікації користувача

Клас MainController реалізує контролер для відображення головної сторінки вебзастосунку. Методи класу наведено в таблиці 4.9.

Таблиця 4.9 – Методи класу MainController

Назва	Опис
public:	
mainPage();	Головна сторінка вебзастосунку
aboutPage();	Сторінка з інформацією про розроблений вебзастосунок

Клас ElectricApplianceDaoImpl реалізує інтерфейс шаблону доступу до даних об'єкту про список побутових приладів, що зберігається у базі даних. Методи класу наведено в таблиці 4.10.

Таблиця 4.10 – Методи класу ElectricApplianceDaoImpl

Назва	Опис
public:	
findAll();	Отримання списку всіх приладів
add();	Додавання побутового приладу
delete();	Видалення побутового приладу
findByName();	Пошук побутового приладу за назвою

Клас GridConsumptionDaoImpl реалізує інтерфейс шаблону доступу до даних об'єкту про споживання електроенергії із зовнішньої мережі, що зберігається у базі даних. Методи класу наведено в таблиці 4.11.

Таблиця 4.11 – Методи класу GridConsumptionDaoImpl

Назва	Опис
public:	
findAll();	Отримання статистики споживання із зовнішньої мережі за весь час
findLast();	Отримання останніх показників споживання із зовнішньої мережі

Клас TimetableDaoImpl реалізує інтерфейс шаблону доступу до даних об'єкту про розклад користування побутовими приладами, що зберігається у базі даних. Методи класу наведено в таблиці 4.12.

Таблиця 4.12 – Методи класу TimetableDaoImpl

Назва	Опис
public:	
findByDate();	Отримання розкладу за датою
save();	Збереження розкладу
periodToDate();	Конвертація періоду в дату

Клас `TotalConsumptionDaoImpl` реалізує інтерфейс шаблону доступу до даних об'єкту про загальне споживання електроенергії, що зберігається у базі даних. Методи класу наведено в таблиці 4.13.

Таблиця 4.13 – Методи класу `TotalConsumptionDaoImpl`

Назва	Опис
public:	
<code>findAll();</code>	Отримання статистики загального споживання за весь час
<code>findLast();</code>	Отримання останніх показників загального споживання

Клас `ElectricAppliance` реалізує модель даних для збереження списку електроприладів у базі даних. Поля класу наведено в таблиці 4.14.

Таблиця 4.14 – Поля класу `ElectricAppliance`

Назва	Опис
private:	
<code>ObjectId id;</code>	Унікальний ідентифікатор в СКБД
protected:	
<code>String name;</code>	Назва електроприладу
<code>int consumption;</code>	Номінальна потужність електроприладу
<code>int hours;</code>	Кількість годин роботи електроприладу

Клас `WorkingAppliance` розширює клас `ElectricAppliance`, додаючи нові поля для збереження інформації про побутовий прилад, що використовується у розкладі. Поля та методи класу наведено в таблиці 4.15.

Таблиця 4.15 – Поля та методи класу WorkingAppliance

Назва	Опис
private:	
int priority;	Пріоритет електроприладу
int startHour;	Час початку роботи електроприладу
int endHour;	Час закінчення роботи електроприладу
boolean isOverload;	Перевитрата електроенергії
public:	
getStartMinValue();	Мінімальний час для початку роботи електроприладу
getStartMaxValue();	Максимальний час для початку роботи електроприладу
getPositionMaxValue();	Максимальний час для закінчення роботи електроприладу

Клас GridConsumption реалізує модель даних для збереження статистики споживання електроенергії із зовнішньої мережі, що зберігається у базі даних. Поля класу наведено в таблиці 4.16.

Таблиця 4.16 – Поля класу GridConsumption

Назва	Опис
private:	
ObjectId id;	Унікальний ідентифікатор в СКБД
Date Time;	Дата та час зняття показників
double Value;	Споживана кількість електроенергії

Клас PowerGeneration реалізує модель даних для збереження інформації про генерацію електроенергії. Поля класу наведено в таблиці 4.17.

Таблиця 4.17 – Поля класу PowerGeneration

Назва	Опис
private:	
Date Time;	Дата та час спрогнозованої генерації
double Value;	Потужність згенерованої електроенергії

Клас PSOParticle реалізує модель даних «частки» методу рою часток. Поля та методи класу наведено в таблиці 4.18.

Таблиця 4.18 – Поля та методи класу PSOParticle

Назва	Опис
private:	
double c1;	Когнітивний коефіцієнт
double c2;	Соціальний коефіцієнт
double w;	Коефіцієнт інерції
int[] currentSchedule;	Поточний розклад
int currentOverrun;	Оцінка поточного розкладу
int[] currentChanging;	Вектор змін
int[] bestSchedule;	Найкращий розклад «частки»
int bestOverrun;	Найкраща оцінка розкладу «частки»
public:	
updateChanging();	Обчислення нового вектору змін
updateCurrentPosition();	Обчислення нового розкладу
updateBest();	Збереження найкращого розкладу

Клас TotalConsumption реалізує модель даних для збереження статистики загального споживання електроенергії, що зберігається у базі даних. Поля класу наведено в таблиці 4.19.

Таблиця 4.19 – Поля класу TotalConsumption

Назва	Опис
private:	
ObjectId id;	Унікальний ідентифікатор в СКБД
Date Time;	Дата та час зняття показників
double Value;	Споживана кількість електроенергії

Клас Timetable реалізує модель даних для збереження розкладу користування побутовими приладами в базі даних. Поля класу наведено в таблиці 4.20.

Таблиця 4.20 – Поля класу Timetable

Назва	Опис
private:	
ObjectId id;	Унікальний ідентифікатор в СКБД
List<WorkingAppliance> appliances;	Список розкладу роботи побутових приладів
List<PowerGeneration> powerGeneration;	Спрогнозована генерація електроенергії

Клас AppliancesServiceImpl реалізує інтерфейс бізнес-логіки модулю редагування списку електроприладів. Методи класу наведено в таблиці 4.21.

Таблиця 4.21 – Методи класу AppliancesServiceImpl

Назва	Опис
public:	
loadAll();	Завантаження списку всіх приладів
addAppliance();	Додавання побутового приладу
deleteAppliance();	Видалення побутового приладу
loadByName();	Пошук побутового приладу за назвою

Клас `PredictServiceImpl` реалізує інтерфейс взаємодії з модулем прогнозування та зчитування файлу конфігурації «`predict.properties`». Поля та методи класу наведено в таблиці 4.22.

Таблиця 4.22 – Поля та методи класу `PredictServiceImpl`

Назва	Опис
private:	
<code>String path;</code>	Шлях до модулю прогнозування
<code>String trainPy;</code>	Виконуваний файл навчання моделі
<code>String predictPy;</code>	Виконуваний файл прогнозування
public:	
<code>predictPowerGeneration();</code>	Запуск прогнозування генерації електроенергії
<code>prepareHistoricalData();</code>	Підготовка навчальної вибірки
<code>trainModel();</code>	Запуск навчання моделі

Клас `WeatherServiceImpl` реалізує інтерфейс взаємодії з метеосервісом та зчитування файлу конфігурації «`weather.properties`». Поля та методи класу наведено в таблиці 4.23.

Таблиця 4.23 – Поля та методи класу `WeatherServiceImpl`

Назва	Опис
private:	
<code>String key;</code>	Ключ API доступу до метеосервісу
<code>String longitude;</code>	Довгота місцезнаходження США
<code>String latitude;</code>	Широта місцезнаходження США
public:	
<code>getWeather();</code>	Завантаження прогнозу погоди
<code>prepareData();</code>	Формування файлу з прогнозом погоди для модулю прогнозування
<code>saveHistoricalWeather();</code>	Збереження історичних даних про погоду для модулю прогнозування

Клас `PSOServiceImpl` реалізує інтерфейс бізнес-логіки модулю оптимізації розкладу побутових приладів на основі методу рою часток. Поля та методи класу наведено в таблиці 4.24.

Таблиця 4.24 – Поля та методи класу `PSOServiceImpl`

Назва	Опис
private:	
<code>int numParticles;</code>	Кількість часток в «рої»
<code>int maxIterations;</code>	Кількість ітерацій
<code>WorkingAppliance[] appliances;</code>	Перелік побутових приладів
<code>Integer[] powerGeneration;</code>	Спрогнозована генерація електроенергії
<code>PSOParticle[] swarm;</code>	Перелік елементів «рою»
public:	
<code>setPowerGeneration();</code>	Додавання списку спрогнозованої генерації електроенергії
<code>setElectricAppliances();</code>	Додавання списку побутових приладів, які обрав користувач
<code>optimizeTimetable();</code>	Створення оптимального розкладу
<code>init();</code>	Ініціалізація початкового «рою»
<code>optimize();</code>	Реалізація методу рою часток
<code>findBestSchedule();</code>	Пошук найкращого розкладу серед «рою»
<code>evaluateOverrun();</code>	Обчислення оцінки розкладу
<code>getTimetable();</code>	Формування розкладу користування побутовими приладами

Клас `RealtimeConsumptionServiceImpl` реалізує інтерфейс бізнес-логіки модулю моніторингу показників енергоспоживання. Методи класу наведено в таблиці 4.25.

Таблиця 4.25 – Методи класу RealtimeConsumptionServiceImpl

Назва	Опис
private:	
<code>init();</code>	Ініціалізація модулю
public:	
<code>getGridConsumption();</code>	Отримання поточного значення споживання електроенергії із зовнішньої мережі
<code>getTotalConsumption();</code>	Отримання поточного значення загального споживання електроенергії
<code>isOverload();</code>	Перевірка високого споживання електроенергії із зовнішньої мережі
<code>analyzeConsumption();</code>	Моніторинг показників енергоспоживання

Клас `OpenhabSmartSupportApplication` є найголовнішим файлом вебзастосунку, який запускає його як серверний додаток, сканує підключені модулі, розроблені сервіси, сервлети, а також ініціалізує фреймворк Spring. Методи класу наведено в таблиці 4.26.

Таблиця 4.26 – Методи класу OpenhabSmartSupportApplication

Назва	Опис
public:	
<code>main();</code>	Вхідна точка для запуску вебзастосунку, використовуючи фреймворк Spring

Взаємодію основних розроблених класів між собою зображено на рисунку 4.12.

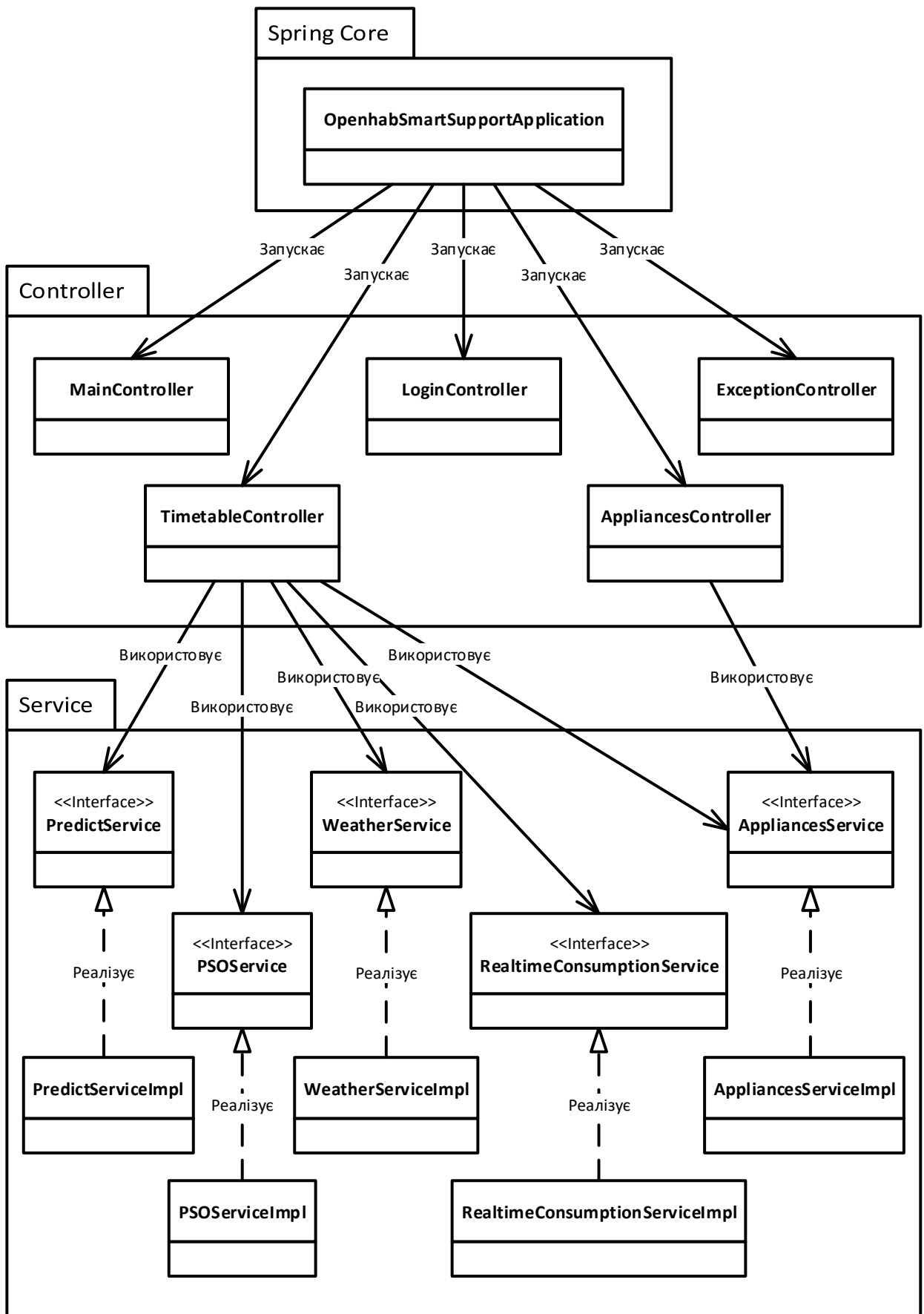


Рисунок 4.12 – Діаграма класів розробленого вебзастосунку

Модуль прогнозування було розроблено з використанням бібліотеки Keras, яка дозволяє швидко реалізовувати нейронні мережі. Також додатково було використано наступні бібліотеки, які поставляються безкоштовно в середовищі Python:

- бібліотека `pandas` дозволяє обробляти та аналізувати дані, що будуть використовуватись під час навчання моделі;
- бібліотека `sklearn` надає широкий набір інструментів для попередньої обробки даних;
- бібліотека `joblib` дозволяє зберігати дані про навчену модель та файли з її вагами, щоб використовувати їх під час прогнозування.

Кожен розроблений клас модулю прогнозування зберігається в окремому виконуваному файлі. Таким чином, було створено два класи: `TrainService` та `PredictService`.

Клас `TrainService` відповідає за навчання та тестування розробленої нейромережевої моделі. Методи класу наведено в таблиці 4.27.

Таблиця 4.27 – Методи класу `TrainService`

Назва	Опис
<code>__init__()</code>	Конструктор
<code>train()</code>	Навчання нейромережевої моделі

Клас `PredictService` відповідає за прогнозування генерації електроенергії. Методи класу наведено в таблиці 4.28.

Таблиця 4.28 – Методи класу `PredictService`

Назва	Опис
<code>__init__()</code>	Конструктор
<code>predict()</code>	Прогнозування генерації

4.5 Реалізація нейромережевої моделі

4.5.1 Набір навчальних даних та їх структура

Одна із найголовніших стадій під час реалізації нейромережевої моделі є перетворення зібраних «сирих» даних у форму, яку можна використовувати під час навчання моделі. Ця стадія складається з наступних кроків: завантаження даних, їх очищення, вибір ознак, нормалізація та створення навчальної вибірки [77], [78].

Етапи щодо підготовки даних для створення моделі прогнозування генерації електроенергії зображено на рисунку 4.13.



Рисунок 4.13 – Етапи попередньої обробки даних

Етап 1. Оскільки генерація електроенергії сонячними електростанціями та вітрогенераторами залежить від погодних умов, то вхідним набором даних має бути історичний прогноз погоди. Таким чином, завантаження даних буде відбуватися з двох різних джерел, де прогноз погоди буде отримано за допомогою метеосервісу «Visual Crossing» (розд. 3.4), а історичні дані споживання електроенергії із мережі та загального споживання із бази даних «MongoDB» (розд. 3.5), у яку платформа OpenHAB зберігає ці показники кожні вісім секунд.

Етап 2. Задача очищення даних полягає у виявленні та виправленні помилок, які містяться у даних. Вимірювання енергоспоживання модулем PZEM іноді супроводжується збоєм в його роботі, що тягне за собою запис невірних даних у базу даних (рис. 4.14). Оскільки діапазон вимірювання цього модулю становить від 0 до 22 кВт [12], то всі значення, що

перевищують цей діапазон мають бути замінені середнім значенням їх сусідніх показників.

datetime	Item1
02-10-22 14:29	289
02-10-22 14:29	250.9
02-10-22 14:29	228.4
02-10-22 14:29	0
02-10-22 14:30	421831040
02-10-22 14:31	0
02-10-22 14:31	5.4
02-10-22 14:31	1224.7
02-10-22 14:31	669.9

Рисунок 4.14 – Приклад аномальних значень енергоспоживання

Оскільки оптимізація розкладу користування побутовими приладами передбачає створення погодинного розкладу, то для прогнозування необхідна навчальна вибірка погодинних змін генерації електроенергії. Спочатку необхідно обчислити середнє значення споживання з мережі та загальне споживання електроенергії для кожної години (рис. 4.15).

datetime	Item1	Item7
2022-10-22T08:00:00	894.7211132	1069.590061
2022-10-22T09:00:00	614.9566219	1249.520487
2022-10-22T10:00:00	413.6865125	866.4380081
2022-10-22T11:00:00	1035.437023	1351.686992
2022-10-22T12:00:00	769.7129771	1331.048984
2022-10-22T13:00:00	1230.768893	1776.077688
2022-10-22T14:00:00	335.4435849	951.0853955
2022-10-22T15:00:00	345.0532443	705.0691207
2022-10-22T16:00:00	370.8289575	577.2464066

Рисунок 4.15 – Приклад погодинних змін енергоспоживання

Також можуть траплятися випадки, коли історичні дані відсутні для декількох годин, саме цьому показники для всієї доби мають бути вилучені, оскільки це може вплинути на точність моделі. Генерація електроенергії, що є прогнозованим значенням розраховується за допомогою формули 1.1.

Етап 3. Для визначення вхідних ознак, що впливають на цільову змінну, необхідно проаналізувати історичні дані (рис. 4.16).

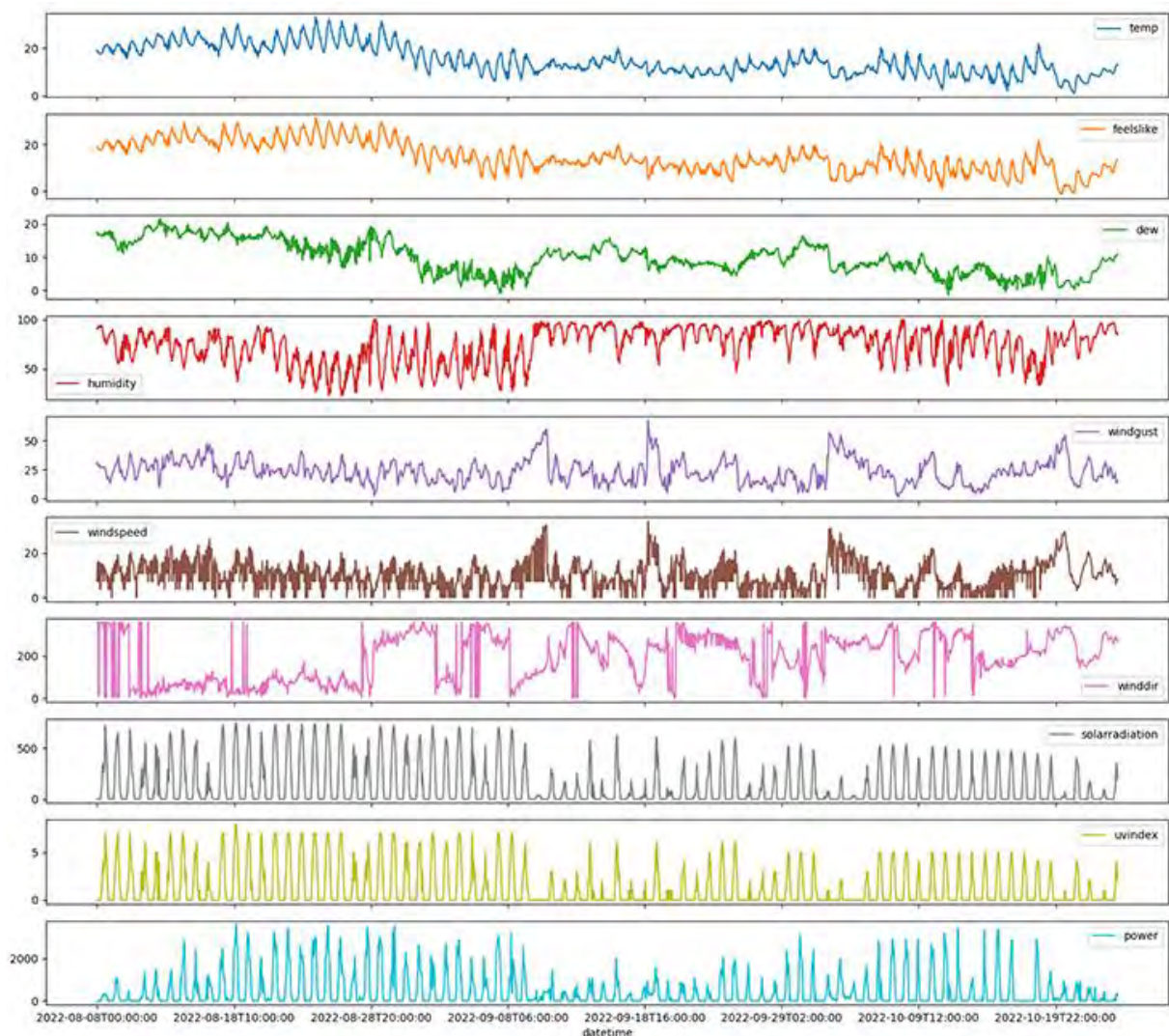


Рисунок 4.16 – Візуалізація вхідних даних

Для визначення залежності між змінними вибірки необхідно створити кореляційну матрицю (рис. 4.17). Таким чином, можна зробити висновок, що цільова змінна генерації електроенергії «power» залежить від таких параметрів:

- потужність сонячного випромінювання «solarradiation» (81%);
- ультрафіолетовий індекс «uvindex» (80%);
- середня температура «temp» (33%);
- тепловий індекс «feelslike» (32%);

- порив вітру «windgust» (18%);
- швидкість вітру «windspeed» (14%).

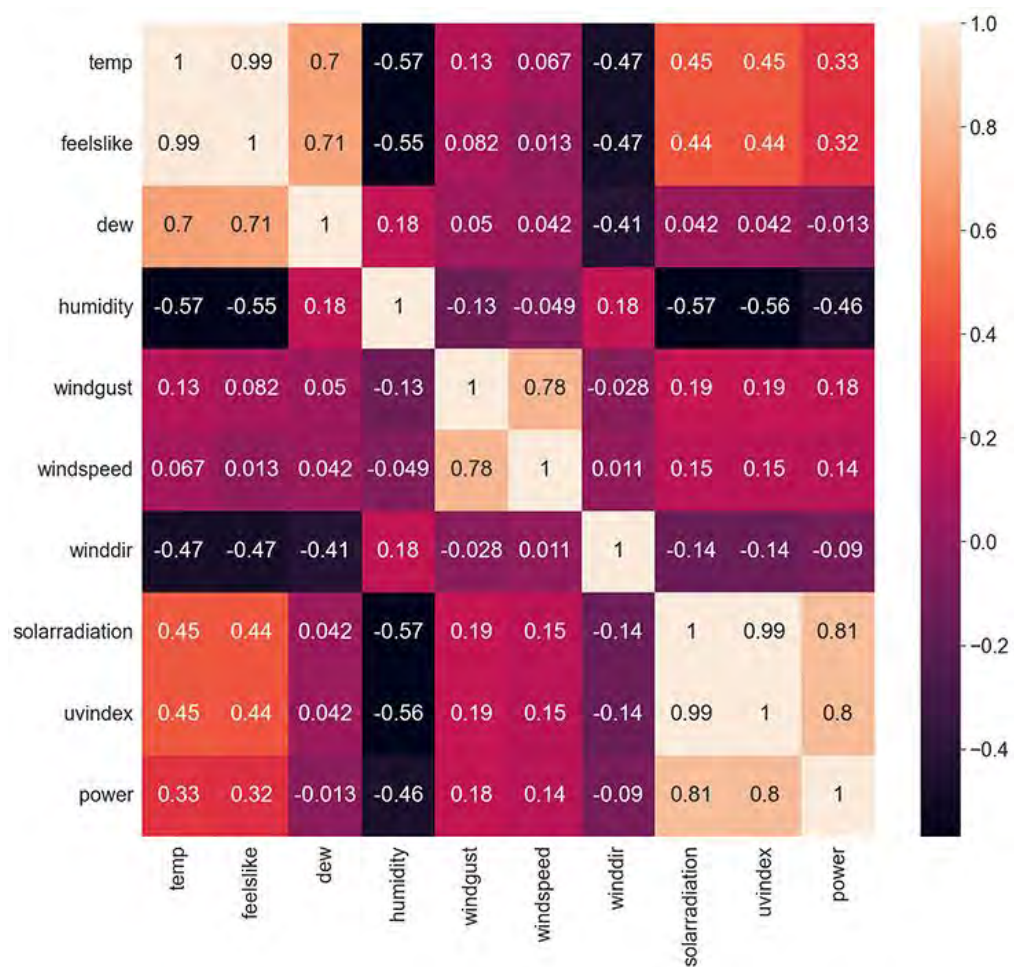


Рисунок 4.17 – Матриця кореляцій між прогнозом погоди та генерацією електроенергії

Етап 4. Нормалізація даних є одним з найважливіших етапів підготовки даних. Часто зустрічається так, що набір даних має різні ознаки, які відрізняються за типом розподілу та діапазоном. Тому необхідно привести дані до єдиного масштабу, щоб ні у якої ознаки не було переваг перед іншими. Цей процес може бути корисним для моделей машинного навчання, які вимагають, щоб функції були в одному діапазоні для отримання точних результатів [79].

Найпопулярнішими методами нормалізації даних є мінімальне-максимальне масштабування, а також нормалізація Z-оцінки [79].

Мета мінімальної-максимальної нормалізації полягає в тому, щоб гарантувати, що всі значення ознак знаходяться в однаковому масштабі, наприклад, в діапазоні $[0, 1]$ або $[-1, 1]$, що робить навчання алгоритму більш ефективним [79].

Нормалізоване значення X' для певного діапазону обчислюється за формулою (4.3):

$$X' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (4.3)$$

де x – початкове значення;

$x_{\min}$ – мінімальне значення всієї вибірки;

$x_{\max}$ – максимальне значення всієї вибірки.

Ідея методу нормалізації Z -оцінки полягає у зміні масштабу об'єктів таким чином, щоб вони мали середнє значення, що дорівнює нулю та стандартне відхилення, рівне одиниці [79].

Нормалізована оцінка Z обчислюється за формулою (4.4):

$$Z = \frac{x - \mu}{\sigma} \quad (4.4)$$

де x – початкове значення;

μ – середнє значення всієї вибірки;

σ – обчислене стандартне відхилення для вибірки.

Етап 5. Формування навчальної вибірки передбачає співставлення значень вхідних ознак, тобто прогнозу погоди зі значенням вихідного параметра – генерацією електроенергії, а також зберігання навчальної вибірки у файлі. Таким чином, результуючий набір включає 2160 записів та охоплює період з 8 серпня 2022 року по 13 листопада 2022 року.

4.5.2 Порівняння та вибір найкращої нейромережевої моделі

Одним із етапів реалізації інтелектуального комплексного методу є прогнозування генерації електроенергії (розд. 3.3), який має бути реалізовано за допомогою штучних нейронних мереж.

Проведені дослідження (розд. 2.5) показали, що неможливо створити універсальну нейромережеву модель, яка буде вирішувати задачі різного типу. Таким чином, необхідно провести ряд комп'ютерних експериментів та серед наявних моделей обрати ту, яка буде забезпечувати найбільшу точність прогнозування для досліджуваної СДА. Так, було запропоновано порівняти наступні моделі: нейронна мережа прямого поширення, РНМ та мережа ДКЧП.

Нейромережа прямого поширення – це така нейромережа, у якій нейрони розділено на групи – шари, а зв'язки між нейронами організовано таким чином, що вхідний шар сприймає зовнішній сигнал, обробляє його і передає значення з виходів своїх нейронів наступному шару, при цьому всередині кожного шару зв'язки між нейронами відсутні, а на входи нейронів схованих шарів, поступають значення з відповідних виходів нейронів попереднього шару [80].

Для реалізації нейронної мережі прямого поширення було використано послідовну модель `keras.models.Sequential`, у якій мережа представлена у вигляді шарів у послідовному порядку, а дані передаються з одного шару на інший у заданому порядку, поки вони не досягнуть вихідного шару [81].

Дана модель повинна приймати чотири параметри, а саме: набір значень вхідних ознак прогнозу погоди (потужність сонячного випромінювання, середня температура, порив вітру) та значення вихідної ознаки – потужність генерації електроенергії. Оскільки вихідна ознака представлена у вигляді безперервного значення, тому функція активації має бути лінійною та повертати значення в діапазоні $[0, +\infty)$.

Проведені дослідження показали, що найкращі результати прогнозування дає модель, яка має наступні додані шари:

- повнозв'язний шар Dense, що складається з 128 нейронів та має функцію активації ReLU;
- повнозв'язний шар Dense, що складається з 24 нейронів;
- повнозв'язний шар Dense, що складається з одного нейрону.

Нейронна мережа зі зворотними зв'язками – це така нейронна мережа, де окрім прямих зв'язків наявні і зворотні зв'язки, тобто зв'язки нейронів самих з собою або зв'язки наступного шару з попереднім [80].

Для реалізації РНМ було також використано послідовну модель `keras.models.Sequential`. На відміну від нейронної мережі прямого поширення, кількість параметрів було збільшено. Таким чином, набір значень вхідних ознак тепер включає час зняття показників, який було перетворено на ціле число, що починається з нуля.

Проведені дослідження реалізації даної моделі показали, що найкращі результати прогнозування дає РНМ, яка має наступні шари:

- рекурентний шар SimpleRNN, який включає 200 нейронів та має функцію активації ReLU;
- повнозв'язний шар Dense, що складається зі 100 нейронів;
- повнозв'язний шар Dense, що складається з одного нейрону.

Нейромережа ДКЧП – це архітектура, що відноситься до парадигми РНМ, але на відміну від традиційних РНМ, мережа ДКЧП добре підходить вирішення задач класифікації, а також для передбачення часових рядів. Мережа ДКЧП складається з шарів, серед яких містяться вузли ДКЧП, які в свою чергу виділяються запам'ятовуванням значень для довгих або коротких проміжків часу [80].

Для реалізації нейронної мережі ДКЧП було використано також послідовну модель `keras.models.Sequential` та аналогічний набір ознак, як і для РКН.

Проведені дослідження показали, що найкращі результати прогнозування дає нейромережа ДКЧП, яка включає наступні шари:

- шар LSTM, який включає 200 нейронів та має функцію активації ReLU;
- повнозв'язний шар Dense, що складається зі 100 нейронів та має функцію активації ReLU;
- повнозв'язний шар Dense, що складається з одного нейрону.

Також для прогнозування часових рядів було реалізовано ще одну мережу ДКЧП, яка має всього два параметри, а саме: дату та потужність генерації електроенергії. Така модель має наступні шари:

- шар LSTM, який включає 100 нейронів та має функцію активації ReLU;
- повнозв'язний шар Dense, що складається зі 100 нейронів та має функцію активації ReLU;
- повнозв'язний шар Dense, що складається з одного нейрону.

Навчання кожної нейромережевої моделі виконувалося впродовж 100 епох з використанням бібліотеки Keras. Набір вхідних даних було розділено на дві частини, де навчальна вибірка має частку у 80% від усього набору, а тренувальна – 20% (рис. 4.18).

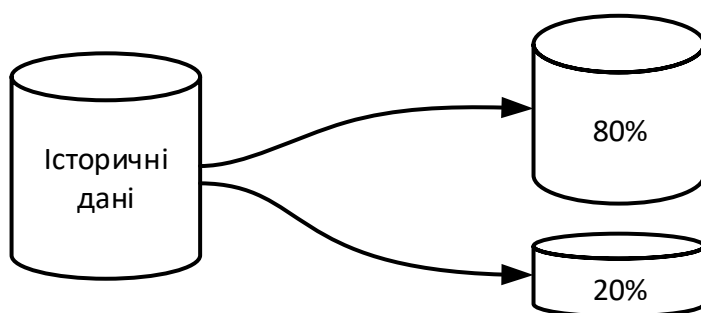


Рисунок 4.18 – Розбиття історичних даних на навчальну та тестову вибірку

В результаті навчання нейромережевих моделей, а також застосування до них тестової вибірки, були отримані графіки точності та втрат моделей.

Графіки зміни точності в залежності від епох для нейронних мереж прямого поширення зображено на рисунку 4.19.

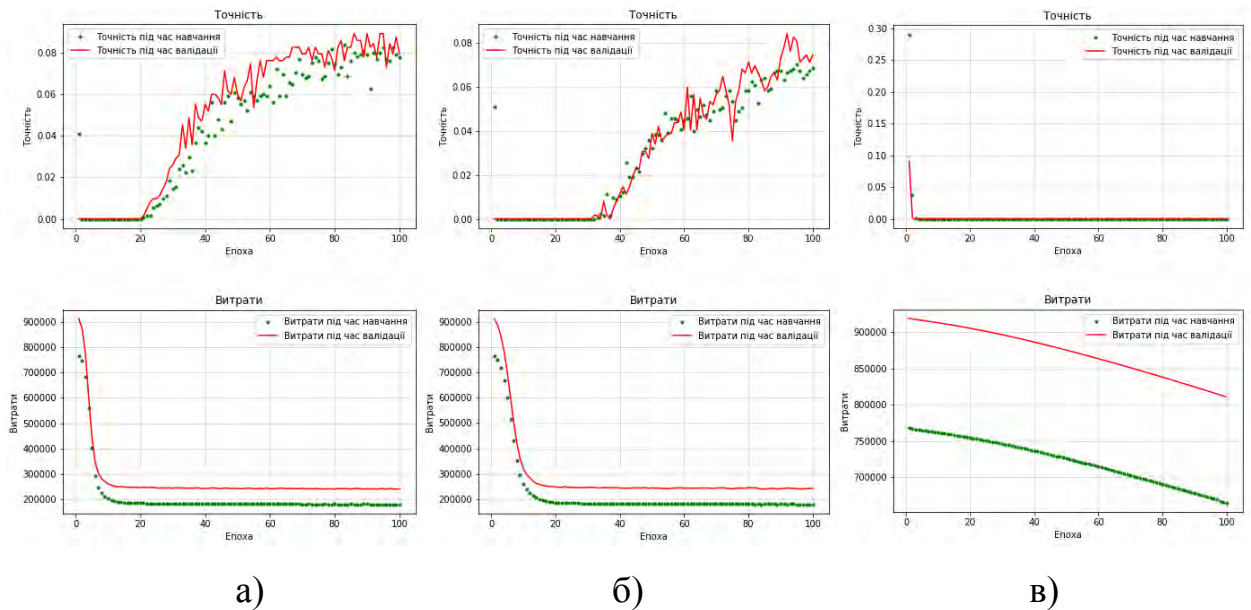


Рисунок 4.19 – Графіки зміни точності для моделей нейронних мереж прямого поширення з використанням оптимізатору ADAM (а), RMSprop (б) та AdaGrad (в)

На рисунку 4.20 наведено графік зміни точності навченої РНМ в залежності від епох.

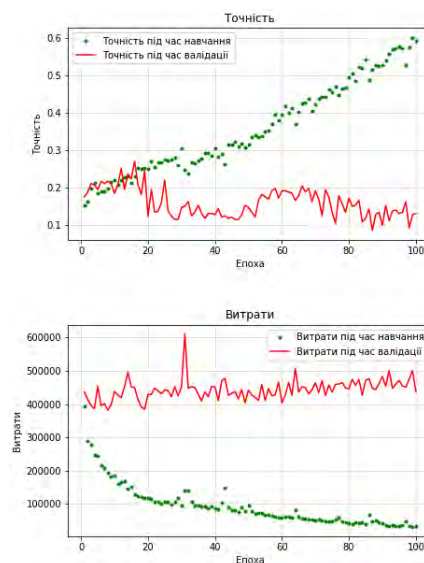


Рисунок 4.20 – Графік зміни точності для РНМ

Графіки зміни точності для навчених нейронних мереж ДКЧП в залежності від епох зображено на рисунку 4.21.

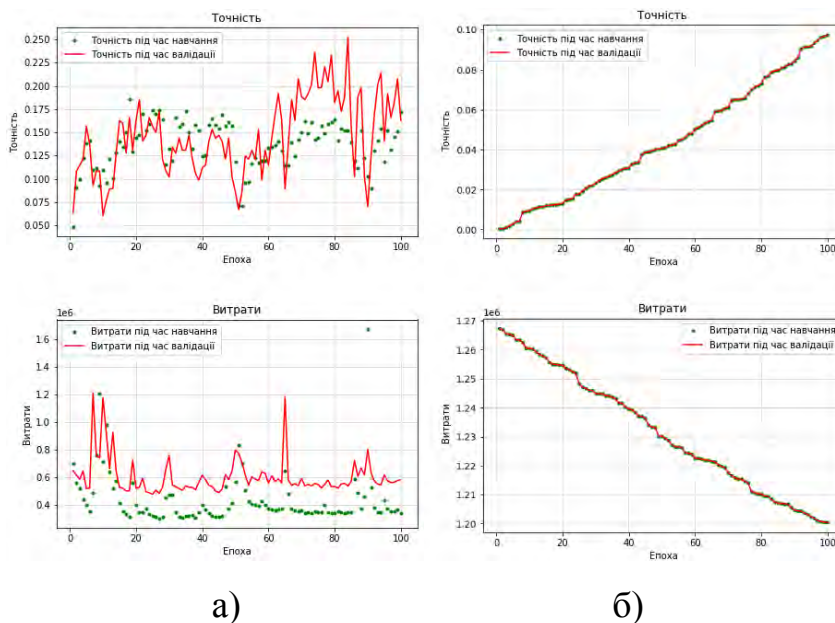


Рисунок 4.21 – Графіки зміни точності для нейронних мереж ДКЧП, які мають п'ять (а) та два (б) параметри

Серед досліджуваних моделей значно виділяється нейронна мережа прямого поширення, яка має найбільше значення точності, тому було проведено ряд додаткових експериментів для збільшення точності прогнозування. Одним із таких підходів є використання різних алгоритмів оптимізації.

Сутність оптимізаційних методів полягає в пошуку екстремуму цільового критерію, в якості якого зазвичай використовується середньоквадратична похибка [80].

Градентні методи є одними з найпопулярніших, які використовуються для оптимізації нейронних мереж. Використовувана бібліотека Keras містить в собі реалізації різних алгоритмів для оптимізації градієнтного спуску. Таким чином, можна виділити три найпопулярніших алгоритми: AdaGrad, RMSProp та ADAM, які реалізують різні підходи для того, щоб зробити градієнтний спуск більш ефективним при знаходженні

оптимальних ваг. Однак ці алгоритми часто використовуються як оптимізатори «чорної скриньки», оскільки важко знайти практичні пояснення їх сильних та слабких сторін [82].

Порівняння результатів навчання та роботи створених моделей прогнозування наведено у таблиці 4.29.

Таблиця 4.29 – Результати навчання реалізованих моделей

Критерій порівняння	Нейронна мережа прямого поширення			PHM	Мережа ДКЧП	
Загальна кількість шарів	4	4	4	4	4	4
Кількість параметрів	4	4	4	5	5	2
Оптимізатор	ADAM	RMSprop	AdaGrad	ADAM	ADAM	ADAM
Помилка на навчальному наборі (RMSE)	446,9633	425,1278	812,8156	183,9365	499,6775	1066,8383
Помилка на тестовому наборі (RMSE)	466,1486	492,0719	831,0850	672,0999	635,0425	1066,8383
Розмір навчальної вибірки	1728	1728	1728	1728	1728	2112
Розмір тренувальної вибірки	432	432	432	432	432	48
Розмір файлу навченої моделі	153,9 кБ	153,9 кБ	153,9 кБ	773,52 кБ	344,8 кБ	166,79 кБ
Час навчання	21 с	22 с	21 с	1 хв 20 с	4 хв 36 с	18 хв 22 с
Кількість батчів	24	24	24	24	24	24
Кількість епох	100	100	100	100	100	100

Для порівняння реалізованих моделей було використано метрику, яка відображає середньоквадратичне відхилення.

Середньоквадратична помилка (англ. Root Mean Square Error (RMSE)) – це середня квадратична різниця між розрахунковими та фактичними значеннями використовуваної моделі. RMSE є найчастіше застосовуваним показником точності для порівняння помилок прогнозування різних моделей та має ті ж самі одиниці вимірювання, що і спрогнозовані значення [83].

Значення середньоквадратичної помилки $RMSE$ для розробленої моделі обчислюється за формулою (4.5):

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (4.5)$$

де n – розмір вибірки;

y_i – очікуване значення для i -го елементу вибірки;

$\hat{y}_i$ – прогнозоване значення для i -го елементу вибірки.

Згідно з отриманими результатами можна побачити, що розроблені моделі РНМ та ДКЧП мають досить велику помилку прогнозування. Серед цих моделей можна виділити мережу ДКЧП з п'ятьма параметрами, оскільки значення RMSE для тестової вибірки становить 635 Вт. Це доводить той факт, що мережа ДКЧП найкраще підходить для прогнозування часових рядів, однак точність таких моделей не є задовільною. Це пояснюється тим, що цільовий параметр ґрунтується не тільки на залежних часових рядах, а й на незалежних від часу показниках, тобто погодних умовах. Тому для збільшення точності прогнозування необхідно мати окремий набір історичних даних для генерації електроенергії сонячними електростанціями та вітрогенераторами. Також іншим рішенням може бути наявність спостережуваних даних для декількох років, що дасть можливість робити прогнозування часових рядів в залежності від сезонності.

Реалізовані моделі на основі нейронної мережі прямого поширення показали більш кращі результати прогнозування. Таким чином, найкращою серед досліджених моделей виявилася така, що має чотири параметри та використовує оптимізатор ADAM. Для нормалізації навчальної вибірки було використано метод Z-оцінки, оскільки він більш точно зберігає значущі зміни показників. Значення RMSE для тестової вибірки дорівнює 466 Вт, що є досить непоганим показником. Тому саме цю модель доцільно використовувати для реалізації комплексного інтелектуального методу для підвищення ефективності користування електроприладами. Серед основних переваг також можна виділити швидкість навчання та невеликий розмір файлу навченої моделі, що дозволить використовувати таку модель навіть на слабких обчислювальних пристроях.

Спроектвану архітектуру для нейронної мережі прямого поширення зображено на рисунку 4.22.

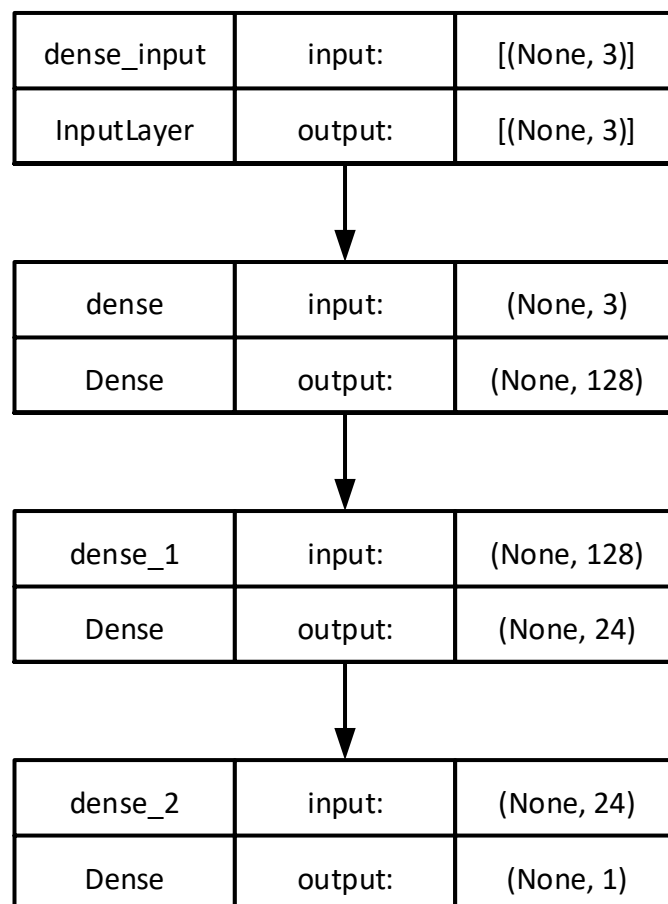


Рисунок 4.22 – Архітектура реалізованої мережі прямого поширення

Хоча точність даної моделі не є досить високою, тому для покращення реалізованої моделі необхідно мати більшу кількість історичних даних, хоча б від одного року. Оскільки підсистема інтелектуальної підтримки має механізм автоматичного перенавчання нейромережевої моделі, саме через деякий час точність прогнозування електроенергії буде збільшено.

Отже, проведені комп'ютерні експерименти, дозволили серед наявних моделей обрати нейронну мережу прямого поширення, яка має найбільшу точність прогнозування, а також основною перевагою є те, що таку модель можна використовувати на слабких обчислювальних пристроях. Тому для реалізації комплексного інтелектуального методу для підвищення ефективності користування електроприладами було прийнято рішення використовувати нейронну мережу прямого поширення.

4.6 Особливості реалізації графічного інтерфейсу

Реалізація графічного інтерфейсу користувача є не менш важливою задачею під час розробки програмного забезпечення, оскільки інтуїтивно-зрозумілий та зручний інтерфейс сприяє більш ефективній роботі користувача з розробленою системою.

Згідно до функціональних та нефункціональних вимог до розроблюваного програмного забезпечення (рис. 3.1), підсистема інтелектуальної підтримки має бути реалізована у вигляді вебзастосунку, що забезпечить широкий доступ до підсистеми, використовуючи будь-який сучасний браузер, а також не залежати від операційної системи [84].

Розробка графічного інтерфейсу підсистеми інтелектуальної підтримки відбувалася з дотриманням п'яти головних принципів, які позначаються аббревіатурою SAPCO (Simple – простий, Aesthetic – естетичний, Productive – продуктивний, Customizable – пристосовуваний, Other – інший) [84].

Простий. Об'єкти графічного інтерфейсу мають бути мінімально складними та багаторівневими. Це означає, що такий інтерфейс не потребує довідника або діалогової довідкової системи, яка буде описувати як почати виконувати прості задачі кінцевих користувачів [84].

Естетичний. Застосунок повинен мати естетичну та ергономічну привабливість, а також справляти гарне враження на користувача [84].

Продуктивний. Розроблений продукт повинен дозволяти виконувати роботу швидко та вимагати мінімальних зусиль для виконання задач кінцевого користувача. Тому необхідно уникати складну ієрархічність вікон, а також непотрібних дій з клавіатурою та мишею [84].

Пристосований. Інтерфейс повинен надавати вибір користувачу та мати властивість пристосовності до різних пристроїв, тобто правильно відображатися на різних розмірах екранів [84].

Інші принципи. Існує безліч інших критеріїв до графічного інтерфейсу, наприклад, легкість, ефективність, надійність тощо [84].

Розробка вебсторінок для обраного фреймворку Spring (розд. 3.6.1) відбувалася з використанням технології JSP. Ця технологія дозволяє реалізувати створення динамічних вебсторінок на стороні сервера. Особливістю використання JSP полягає в тому, що в розмітку HTML можна додавати спеціальні теги та інтегрувати вихідний код на мові Java для реалізації відображення необхідної інформації. Такий підхід значно спрощує та пришвидшує розробку вебзастосунків [55].

В процесі створення вебсторінок було використано бібліотеку jQuery, яка розроблена на мові програмування JavaScript. Її особливість полягає в тому, що це потужна та невелика бібліотека, яка має багато функцій для зручної розробки вебсторінок. Вона надає можливість маніпулювати елементами HTML сторінки, створювати анімації, обробляти події користувача, а також відправляти асинхронні запити до сервера [85]. Бібліотека jQuery розроблена під ліцензією MIT та це означає, що її можна використовувати у будь-яких проєктах безкоштовно [86].

Однією з нефункціональних вимог розроблюваної підсистеми є необхідність коректного відображення графічного інтерфейсу на планшетах та мобільних пристроях. Для реалізації такої вимоги було використано бібліотеку Bootstrap, яка надає можливість легко створювати адаптивний дизайн. Таким чином, сторінки будуть автоматично налаштовуватися так, щоб добре відображатися на всіх пристроях: від телефонів до настільних комп'ютерів [87]. Як і jQuery, бібліотека Bootstrap має ліцензію MIT [88].

Для відображення різних іконок та значків на вебсторінках було використано бібліотеку FontAwesome, що робить графічний інтерфейс більш зрозумілим та привабливим. Ліцензія Creative Commons надає право вільно використовувати цю бібліотеку в будь-яких проєктах [89].

4.7 Висновки до розділу

Підсистему інтелектуальної підтримки було реалізовано у вигляді вебзастосунку, використовуючи фреймворк Spring. Було розроблено структурно-функціональну схему підсистеми (рис. 4.1), яка має наступні модулі: модуль моніторингу показників; редагування електроприладів; оптимізації розкладу користування побутовими електроприладами; модуль прогнозування.

Модуль прогнозування генерації електроенергії було розроблено на мові програмування Python з використанням бібліотеки Keras. Взаємодію між вебзастосунком, що написано на мові Java та модулем прогнозування зображено на рисунку 4.4.

Зберігання усіх необхідних даних реалізовано у базі даних MongoDB, яка має чотири колекції (рис. 4.5). Структуру для кожного документу, що зберігається у базі даних зображено на рисунках 4.6 – 4.8. Зберігання історичних даних про прогноз погоди, а також дані спрогнозованої електроенергії реалізовано за допомогою CSV файлів.

Основні розроблені алгоритми наведено на рисунках 4.10 – 4.11, де продемонстровано алгоритм оптимізації розкладу роботи електроприладів на основі методу рою часток та алгоритм моніторингу показників енергоспоживання.

Розроблена підсистема складається з інтерфейсів та класів, які поділяються на такі групи: класи для зчитування файлів конфігурації; класи, що реалізують логіку контролерів; інтерфейси для взаємодії з базою даних; класи, що описують моделі даних; інтерфейси для реалізації бізнес-логіки всіх модулів. Опис всіх реалізованих класів подано у таблицях 4.2 – 4.28. Діаграму основних класів вебзастосунку зображено на рисунку 4.12.

Підготовчим етапом реалізації нейромережевої моделі було перетворення «сирих» даних у форму, яку можна буде використовувати під час навчання моделі. Цей етап складається з наступних кроків: завантаження даних, їх очищення, вибір ознак, нормалізація та створення навчальної вибірки. Аналіз історичних даних та матрицю кореляції зображено на рисунках 4.16 та 4.17 відповідно.

Проведені дослідження (розд. 2.5) показали, що неможливо створити універсальну нейромережеву модель для вирішення всіх типів задач. Тому було проведено ряд комп'ютерних експериментів, які дозволили порівняти різні типи нейронних мереж, а саме: нейронну мережу прямого поширення, РНМ та мережу ДКЧП. Графіки зміни точності в залежності від епох для всіх реалізованих моделей зображено на рисунках 4.19 – 4.21. Порівняння результатів навчання та роботи створених моделей прогнозування подано у таблиці 4.29. Згідно з отриманими результатами було виявлено, що найменшу помилку прогнозування має нейронна мережа прямого поширення, для якої значення RMSE становить 466 Вт, що є досить непоганим показником. Спроектовану архітектуру для цієї моделі зображено на рисунку 4.22.

Розробка графічного інтерфейсу підсистеми відбувалася з дотриманням п'яти головних принципів SAPCO. Розробка вебсторінок для

фреймворку Spring відбувалася з використанням технології JSP. В процесі створення вебсторінок було використано бібліотеку jQuery, яка надає широкий спектр функцій для зручної розробки. Для реалізації адаптивного дизайну, який буде добре відображатися на всіх пристроях було використано бібліотеку Bootstrap. Бібліотека FontAwesome дозволяє додавати різні іконки та значки, що робить графічний інтерфейс більш зрозумілим та привабливим.

Розроблена підсистема інтелектуальної підтримки інтегрується з платформою OpenHAB та призначена для надання рекомендацій мешканцям будинку щодо ефективного користування електроприладами з максимальним використанням альтернативних джерел енергії, а також з мінімальною залежністю від зовнішньої електромережі.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОЇ ПІДСИСТЕМИ

5.1 Функціональне тестування

Фінальним етапом розробки будь-якого програмного забезпечення виступає перевірка працездатності всіх функцій ресурсу. Для цього необхідно провести функціональне тестування розробленої підсистеми [90].

Функціональне тестування – це один із найбільш головних та масштабних видів робіт, основною метою якого є переконання в коректності роботи всіх основних функцій вебзастосунку [90].

Оскільки власник будинку досліджуваної СДА не має змогу надати віддалений доступ до локальної бази даних, що використовується платформою OpenHAB, то було прийнято рішення на етапі розробки придбати сервер та встановити на нього таку ж саму базу даних, тобто MongoDB. Для обох баз даних було налаштовано механізм реплікації, який дозволяє копіювати дані в автоматичному режимі з локальної СКБД до віддаленої (рис. 3.3). Такий підхід дозволить проводити тестування розроблюваної підсистеми у віддаленому режимі з реальним набором даних, що накопичується платформою OpenHAB.

Враховуючи вимоги до розроблюваної підсистеми інтелектуальної підтримки користувачів, було створено список тест-кейсів, які подано у таблицях 5.1 – 5.4.

Таблиця 5.1 – Тест-кейс успішної авторизації

Короткий опис	Перевірка можливості успішної авторизації			
Передумови	Користувач неавторизований та відкрито сторінку авторизації			
Кроки	№	Покрокові дії	Очікуваний результат	Статус
	1	Ввести логін у відповідне поле	Логін користувача введено	Пройдено
	2	Ввести пароль у відповідне поле	Пароль користувача введено	Пройдено
	3	Натиснути на кнопку «Вхід»	Відкрито головну сторінку	Пройдено

Таблиця 5.2 – Тест-кейс невдалої авторизації

Короткий опис	Перевірка можливості авторизації з неправильним логіном або паролем			
Передумови	Користувач неавторизований та відкрито сторінку авторизації			
Кроки	№	Покрокові дії	Очікуваний результат	Статус
	1	Ввести неправильний логін у відповідне поле	Логін користувача введено	Пройдено
	2	Ввести неправильний пароль у відповідне поле	Пароль користувача введено	Пройдено
	3	Натиснути на кнопку «Вхід»	Виведено повідомлення про помилку під час авторизації	Пройдено

Таблиця 5.3 – Тест-кейс перегляду списку всіх електроприладів

Короткий опис	Перевірка можливості перегляду списку електроприладів			
Передумови	Користувач авторизований та відкрито головну сторінку			
Кроки	№	Покрокові дії	Очікуваний результат	Статус
	1	Натиснути на кнопку «Побутові прилади»	Відкрито сторінку «Побутові прилади» зі списком всіх приладів	Пройдено
	2	Перемістити положення повзунка	Відбувається гортання списку електроприладів	Пройдено
	3	Натиснути на кнопку «На головну»	Відкрито головну сторінку	Пройдено

Таблиця 5.4 – Тест-кейс видалення побутового приладу зі списку всіх електроприладів

Короткий опис	Перевірка можливості видалення побутового приладу зі списку електроприладів			
Передумови	Користувач авторизований та відкрито головну сторінку			
Кроки	№	Покрокові дії	Очікуваний результат	Статус
	1	Натиснути на кнопку «Побутові прилади»	Відкрито сторінку «Побутові прилади» зі списком всіх приладів	Пройдено
	2	Біля необхідного приладу натиснути на кнопку «Видалити»	Оновлено список електроприладів, у якому видалено електроприлад	Пройдено

Список тест-кейсів для перевірки роботи основних функцій роботи підсистеми інтелектуальної підтримки подано у таблицях 5.5 – 5.8.

Таблиця 5.5 – Тест-кейс додавання нового побутового приладу до списку всіх електроприладів

Короткий опис	Перевірка можливості додавання нового побутового приладу до списку електроприладів			
Передумови	Користувач авторизований та відкрито головну сторінку			
Кроки	№	Покрокові дії	Очікуваний результат	Статус
	1	Натиснути на кнопку «Побутові прилади»	Відкрито сторінку «Побутові прилади» зі списком всіх приладів	Пройдено
	2	Ввести назву приладу у відповідне поле	Назву побутового приладу введено	Пройдено
	3	Ввести номінальну потужність приладу у відповідне поле	Номінальну потужність приладу введено	Пройдено
	4	Ввести кількість годин роботи приладу у відповідне поле	Кількість годин роботи приладу введено	Пройдено
	5	Натиснути на кнопку «Додати»	Оновлено список електроприладів, у якому відображено новий електроприлад	Пройдено

Таблиця 5.6 – Тест-кейс моніторингу показників енергоспоживання в реальному часі

Короткий опис	Перевірка можливості моніторингу показників енергоспоживання в реальному часі			
Передумови	Користувач авторизований та відкрито головну сторінку			
Кроки	№	Покрокові дії	Очікуваний результат	Статус
	1	Натиснути на кнопку «Переглянути розклад»	Відкрито сторінку «Календар» та обрано сьогоднішній день	Пройдено
	2	Перемістити положення повзунка до поточної години	Відображено інформацію про енергоспоживання для поточної години	Пройдено

Таблиця 5.7 – Тест-кейс оповіщення про небезпечні зміни у показниках, що моніторяться

Короткий опис	Перевірка можливості оповіщення користувача про небезпечні зміни у показниках, що моніторяться			
Передумови	Користувач авторизований та відкрито головну сторінку, а споживання електроенергії з мережі перевищує заданий поріг			
Кроки	№	Покрокові дії	Очікуваний результат	Статус
	1	Натиснути на кнопку «Переглянути розклад»	Відкрито сторінку «Календар» та обрано сьогоднішній день	Пройдено
	2	Зачекати оновлення даних	Виведено повідомлення про високе споживання електроенергії	Пройдено

Таблиця 5.8 – Тест-кейс створення оптимального розкладу

Короткий опис	Перевірка можливості створити оптимальний розклад користування побутовими приладами на наступний день			
Передумови	Користувач авторизований та відкрито головну сторінку			
Кроки	№	Покрокові дії	Очікуваний результат	Статус
	1	Натиснути на кнопку «Створити розклад»	Відкрито сторінку «Планувальник розкладу»	Пройдено
	2	Натиснути на кнопку «Завтра»	Відображено порожній список запланованих приладів	Пройдено
	3	Натиснути на кнопку «Додати прилад»	Відкрито модальне вікно зі списком всіх електроприладів	Пройдено
	4	Біля необхідного приладу натиснути на кнопку «Додати»	Прилад додано до списку запланованих приладів	Пройдено
	5	Натиснути на необхідні значення початку та кінця роботи приладу	Відображено часовий інтервал роботи електроприладу	Пройдено
	6	Натиснути на кнопки «+» або «-» для зміни пріоритету	Змінено пріоритет електроприладу	Пройдено
	7	Натиснути на кнопку «Створити розклад»	Виведено повідомлення про необхідність зачекати	Пройдено
	8	Зачекати декілька секунд	Відкрито сторінку «Календар» та обрано завтрашній день	Пройдено

Під час тестування розробленої підсистеми було також виконано різні перевірки, перелік яких подано у таблиці 5.9.

Таблиця 5.9 – Контрольний список тестування

Категорія перевірки	Вимога	Статус
1	2	3
Доступність сторінок	Доступність сторінки авторизації користувача	Пройдено
	Доступність головної сторінки	Пройдено
	Доступність сторінки «Календар»	Пройдено
	Доступність сторінки «Планувальник розкладу»	Пройдено
	Доступність сторінки «Побутові прилади»	Пройдено
	Доступність сторінки «Про підсистему»	Пройдено
Можливості сторінки авторизації	Перенаправлення на сторінку авторизації при відсутності активної сесії	Пройдено
	Перенаправлення на головну сторінку при успішній авторизації	Пройдено
	Виведення повідомлення при невдалій спробі авторизації	Пройдено
Можливості сторінки «Календар»	Відображення розкладу користування електроприладами	Пройдено
	Відображення спрогнозованої генерації електроенергії	Пройдено
	Можливість відобразити розклад на «сьогодні»	Пройдено
	Можливість відобразити розклад на «завтра»	Пройдено
	Можливість відобразити розклад на «післязавтра»	Пройдено
	Відображення показників електроспоживання в реальному часі	Пройдено
	Виведення повідомлення у разі високого використання електроенергії із зовнішньої мережі	Пройдено
	Відображення назви та номінальної напруги для кожного побутового приладу	Пройдено
	Відображення побутових приладів червоним кольором, що перевищують генерацію електроенергії	Пройдено
	Відображення повного імені електроприладу при наведенні	Пройдено
	Можливість перейти на головну сторінку	Пройдено
	Можливість завершити сесію	Пройдено
Можливості сторінки «Планувальник розкладу»	Можливість обрати день створення розкладу	Пройдено
	Відображення повного імені для кожного приладу	Пройдено
	Відображення номінальної потужності для кожного приладу	Пройдено
	Відображення пріоритету для кожного приладу	Пройдено

Продовження таблиці 5.9.

1	2	3
	Можливість додати новий прилад до списку запланованих приладів	Пройдено
	Можливість видалити прилад зі списку запланованих приладів	Пройдено
	Можливість додати декілька приладів	Пройдено
	Можливість змінити пріоритет доданого приладу	Пройдено
	Можливість встановити максимальний пріоритет, що дорівнює п'ять	Пройдено
	Можливість встановити мінімальний пріоритет, що дорівнює один	Пройдено
	Можливість обрати часовий інтервал роботи приладу	Пройдено
	Можливість розпочати створення оптимального розкладу	Пройдено
	Можливість завершити сесію	Пройдено
Можливості сторінки «Побутові прилади»	Відображення списку всіх електроприладів	Пройдено
	Відображення повного імені для кожного приладу	Пройдено
	Відображення номінальної потужності для кожного приладу	Пройдено
	Відображення кількості годин роботи для кожного приладу	Пройдено
	Можливість гортати список електроприладів	Пройдено
	Можливість модифікувати інформацію про прилад	Пройдено
	Можливість видалити електроприлад зі списку	Пройдено
	Попередження користувача під час видалення приладу	Пройдено
	Можливість додати новий електроприлад	Пройдено
	Перевірка правильності введення даних під час додавання нового електроприладу	Пройдено
	Перевірка на вже існуючий електроприлад під час додавання	Пройдено
	Можливість перейти на головну сторінку	Пройдено
	Можливість завершити сесію	Пройдено
Можливості сторінки «Про підсистему»	Виведення опису розробленої підсистеми	Пройдено
	Виведення версії розробленої підсистеми	Пройдено
	Виведення використаних під час розробки технологій	Пройдено
	Виведення інформації про розробника	Пройдено
	Можливість перейти на головну сторінку	Пройдено
	Можливість завершити сесію	Пройдено
Інше	Читабельний розмір шрифтів на сторінках	Пройдено
	Мова графічного інтерфейсу – українська	Пройдено
	Висока швидкість відклику підсистеми на дії користувача	Пройдено

5.2 Кросплатформенне тестування

Однією з розроблених вимог підсистеми інтелектуальної підтримки є те, що вебзастосунок повинен коректно відображатися на всіх платформах, а саме: на комп'ютерах, планшетах та мобільних пристроях. Тому необхідно перевірити роботу вебзастосунку на пристроях з різною роздільною здатністю, використовуючи для кожного з них розроблені тест кейси та контрольний список тестування (розд. 5.1).

Для проведення тестування було використано браузер «Google Chrome», який має вбудовані інструменти для імітації різних платформ та розміру дисплею [91]. Результати тестування вебзастосунку для різних платформ зведено в таблиці 5.10.

Таблиця 5.10 – Результати кросплатформенного тестування

Платформа	Ширина дисплею	Статус
Мобільний пристрій	320 px (розмір S)	Пройдено
	375 px (розмір M)	Пройдено
	425 px (розмір L)	Пройдено
Планшет	768 px	Пройдено
Ноутбук	1024 px	Пройдено
Комп'ютер	1440 px (розмір L)	Пройдено
	2560 px (розмір 4K)	Пройдено

На рисунках 5.1 – 5.3 наведено приклад відображення вебзастосунку на різних платформах.

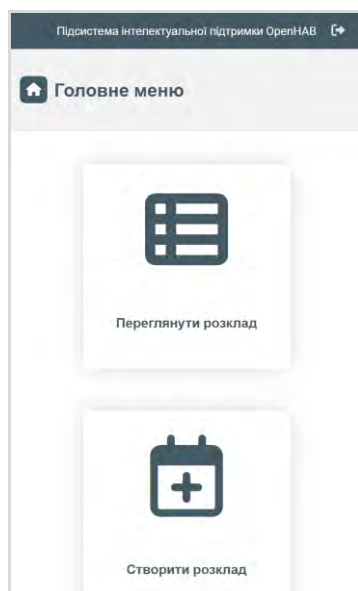


Рисунок 5.1 – Відображення вебзастосунку на мобільному пристрої

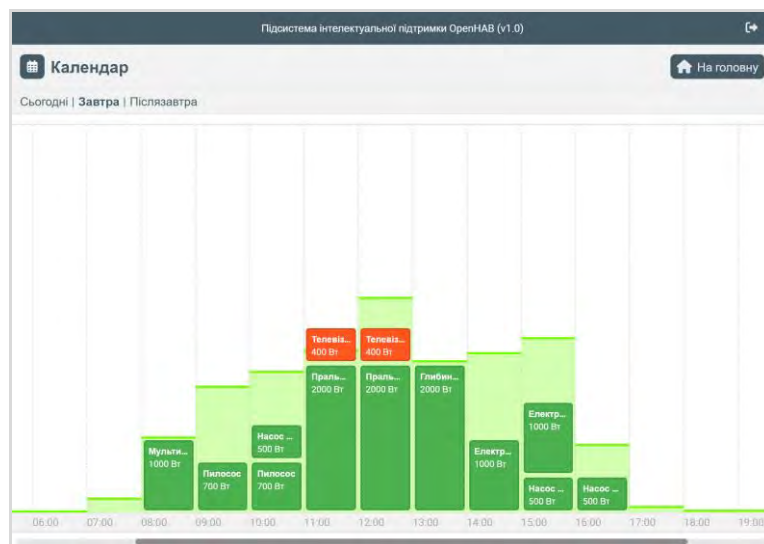


Рисунок 5.2 – Відображення вебзастосунку на планшеті

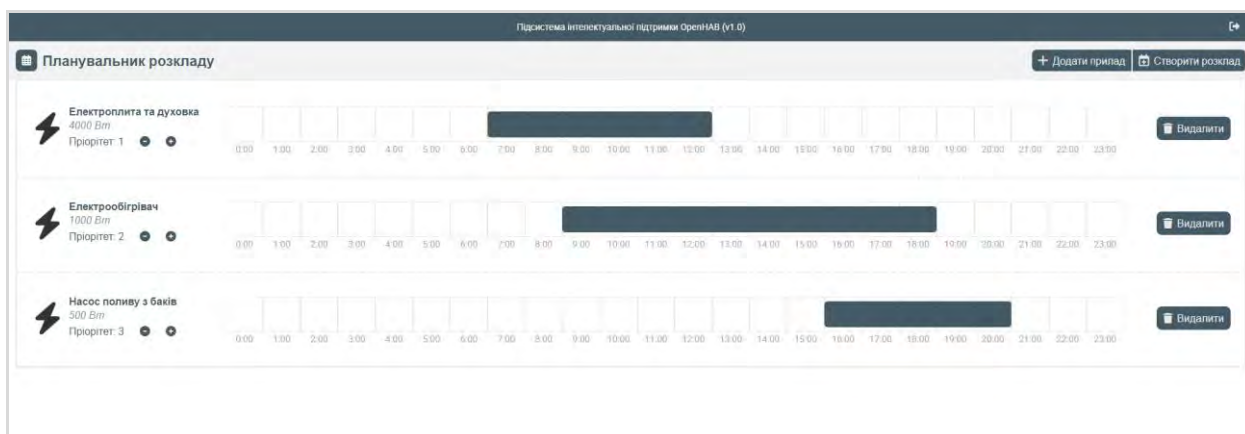


Рисунок 5.3 – Відображення вебзастосунку на комп'ютері

5.3 Висновки до розділу

Було виконано тестування розробленої підсистеми інтелектуальної підтримки користувачів. Для можливості перевірки роботи підсистеми на реальному наборі даних, що накопичується платформою OpenHAB, було створено віддалену базу даних MongoDB. Для неї було налаштовано механізм реплікації, який дозволяє в автоматичному режимі копіювати дані з локальної бази даних досліджуваної СДА до віддаленої.

Функціональне тестування було проведено за допомогою розроблених тест-кейсів (табл. 5.1 – 5.8), а також контрольного списку (табл. 5.9). Кросплатформне тестування дозволило перевірити коректність відображення вебзастосунку на різних платформах, а саме: мобільних пристроях, планшетах та комп'ютерах (табл. 5.10). Таким чином, згідно з результатами тестування можна зробити висновок, що підсистема інтелектуальної підтримки функціонує правильно та відповідає всім поставленим вимогам.

6 КЕРІВНИЦТВО ПРОГРАМІСТА

6.1 Призначення та умови застосування підсистеми

Розроблена підсистема інтелектуальної підтримки інтегрується з платформою OpenHAB та призначена надавати рекомендації мешканцям будинку щодо ефективного користування електроприладами з максимальним використанням альтернативних джерел енергії, а також з мінімальною залежністю від зовнішньої електромережі.

Підсистема інтелектуальної підтримки користувачів надає можливість створювати оптимальний розклад роботи побутових приладів, підлаштовуючись під вимоги та звички мешканців, використовуючи спрогнозовану генерацію електроенергії на наступний день. Крім того, підсистема виконує цілодобовий контроль показників енергоспоживання та у разі високого споживання електроенергії із зовнішньої мережі відправляє повідомлення користувачу, використовуючи графічний інтерфейс.

Як апаратно-технічні засоби для експлуатації підсистеми інтелектуальної підтримки повинен використовуватися IBM-сумісний комп'ютер з характеристиками не нижче:

- процесор Intel Pentium 2266 МГц;
- оперативна пам'ять 512 МБ;
- 50 МБ вільного місця на жорсткому диску для зберігання виконуваних файлів;
- маніпулятор типу «миша» та / або клавіатура.

Як програмні засоби для коректного функціонування підсистеми інтелектуальної підтримки мають бути встановлені:

- операційна система не нижче Microsoft Windows 7 / Microsoft Windows Server 2012 / OS X 10.9 / Ubuntu Linux 12.04;
- виконавче середовище Java Runtime Environment (JRE) версії 1.8;
- виконавче середовище Python версії 3.10.0;
- бібліотека для Python: Keras-Preprocessing версії 1.1.2;

- бібліотека для Python: pandas версії 1.5.2;
- бібліотека для Python: sklearn версії 1.1.3;
- бібліотека для Python: joblib версії 1.2.0.

6.2 Характеристики підсистеми

Підсистему інтелектуальної підтримки користувачів реалізовано у вигляді вебзастосунку, який розроблено на мові програмування Java з використанням фреймворку Spring, та модуль прогнозування, який написано на мові Python з використанням бібліотеки Keras. Підсистема інтегрується з платформою OpenHAB та повинна працювати цілодобово для моніторингу показників енергоспоживання.

Розроблена підсистема дозволить виконувати такі функції:

- виконувати постійний моніторинг показників споживання електроенергії з мережі;
- виконувати постійний моніторинг показників загального споживання електроенергії;
- відправлення повідомлення користувачу у разі високого споживання електроенергії із зовнішньої мережі;
- можливість редагування списку електроприладів;
- збереження списку електроприладів у базі даних;
- можливість створення оптимального розкладу користування побутовими приладами, враховуючи вимоги та звички мешканців;
- можливість збереження створеного розкладу у базі даних;
- прогнозування генерації електроенергії на наступний день, використовуючи метеосервіс;
- автоматичне перенавчання нейронної мережі, використовуючи історичні дані, що зберігаються у базі даних.

6.3 Інсталяція та налаштування підсистеми

Для розгортання вебзастосунку необхідно встановити безкоштовне програмне забезпечення Apache Tomcat версії 9, яке виконує роль контейнера сервлетів. Інструкцію щодо інсталяції можна знайти на офіційному вебсайті програмного забезпечення [92].

Далі необхідно перемістити скомпільований вебзастосунок «OpenhabSmartSupportApplication-1.0.0.war» у директорію «webapps», що знаходиться в папці встановленого Apache Tomcat.

Після цього необхідно розпакувати архів з виконуваними файлами модулю прогнозування «OpenhabSmartSupportApplication-1.0.0_predict.zip» у будь-яке зручне місце.

Для налаштування підсистеми інтелектуальної підтримки необхідно відкрити файл «OpenhabSmartSupportApplication-1.0.0.war» будь-яким архіватором та перейти у директорію WEB-INF/classes, в якій знаходяться файли конфігурації (рис. 6.1).

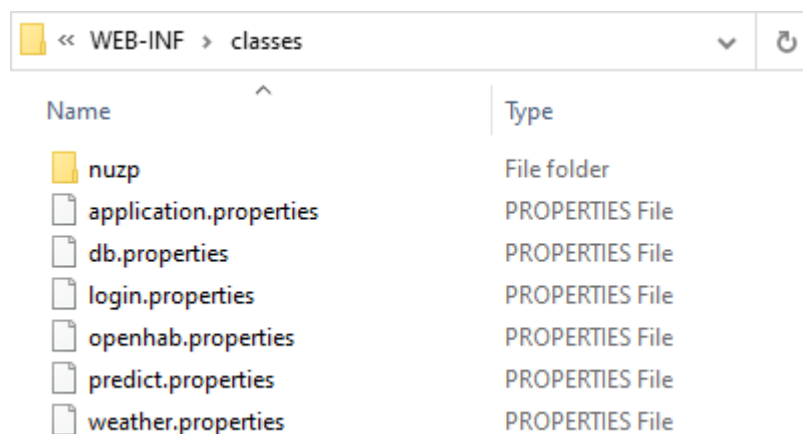


Рисунок 6.1 – Файли конфігурації вебзастосунку

Кожен файл включає певну групу налаштувань для відповідних модулів вебзастосунку. Редагування файлів конфігурації відбувається будь-яким текстовим редактором.

Опис всіх файлів конфігурації вебзастосунку, а також приклади налаштування подано у таблицях 6.1 – 6.6.

Таблиця 6.1 – Файл конфігурації «application.properties»

Параметр конфігурації	Опис	Приклад налаштування
server.port	Порт, на якому розгорнуто вебзастосунок	8080

Таблиця 6.2 – Файл конфігурації «db.properties»

Параметр конфігурації	Опис	Приклад налаштування
mongodb.host	Адреса бази даних	127.0.0.1
mongodb.port	Порт бази даних	27017
mongodb.user	Користувач бази даних	root
mongodb.password	Пароль бази даних	root
mongodb.database	Назва бази даних	openhav

Таблиця 6.3 – Файл конфігурації «login.properties»

Параметр конфігурації	Опис	Приклад налаштування
admin.login	Логін адміністратора	admin
admin.password	Пароль адміністратора	admin

Таблиця 6.4 – Файл конфігурації «predict.properties»

Параметр конфігурації	Опис	Приклад налаштування
predict.path	Шлях до модулю прогнозування	/opt/predictor/
predict.python.train	Виконуваний Python файл для навчання моделі	train.py
predict.python.predict	Виконуваний Python файл для прогнозування	predict.py

Таблиця 6.5 – Файл конфігурації «weather.properties»

Параметр конфігурації	Опис	Приклад налаштування
<code>weather.api.key</code>	Ключ API доступу до метеосервісу «Visual Crossing»	ABCDEF12345
<code>weather.api.long</code>	Довгота місцезнаходження СДА	50.123456
<code>weather.api.latt</code>	Широта місцезнаходження СДА	30.123456

Таблиця 6.6 – Файл конфігурації «openhhab.properties»

Параметр конфігурації	Опис	Приклад налаштування
<code>openhhab.consumption.grid</code>	Назва колекції в СКБД для збереження статистики споживання електроенергії із зовнішньої мережі	Item1
<code>openhhab.consumption.total</code>	Назва колекції в СКБД для збереження статистики загального споживання електроенергії	Item7
<code>openhhab.pzem.interval</code>	Частота запису даних в СКБД в мілісекундах	8000
<code>openhhab.overload.threshold</code>	Поріг попередження про високе використання електроенергії із зовнішньої мережі (Вт)	0

Останнім етапом інсталяції та налаштування підсистеми є запуск контейнера сервлетів Apache Tomcat. Після цього підсистема інтелектуальної підтримки почне свою роботу в фоновому режимі.

6.4 Звертання до підсистеми

Звертання до підсистеми інтелектуальної підтримки відбувається за допомогою будь-якого сучасного браузера. Для цього необхідно в адресний рядок ввести адресу та порт, на якому було розгорнуто вебзастосунок.

6.5 Вхідні та вихідні дані

Вхідними даними для роботи підсистеми є набір історичних даних споживання із зовнішньої мережі та загального споживання, що зберігається у базі даних та використовується для навчання нейронної мережі. Крім того, для навчання нейронної мережі та прогнозування генерації електроенергії використовується прогноз погоди, що завантажується з метеосервісу. Також вхідними даними виступають показники енергоспоживання, які постійно моніторяться підсистемою та список електроприладів, які користувач може додавати, видаляти та редагувати.

Вихідними даними виступає спрогнозована генерація електроенергії на наступний день, а також оптимальний розклад користування побутовими приладами, який було створено підсистемою, враховуючи вимоги та звички мешканців. Також вихідними даними є повідомлення, які відправляються користувачу у разі високого споживання електроенергії із зовнішньої мережі, використовуючи графічний інтерфейс.

6.6 Повідомлення

Розроблена підсистема забезпечує стабільну цілодобову роботу. Зазвичай помилки при роботі можуть виникати з причин, коли підсистема інтелектуальної підтримки була неправильно налаштована або стався збій через зовнішні обставини, наприклад, коли немає з'єднання з мережею Інтернет. В такому випадку користувач отримає повідомлення, що буде відображено на сторінці вебзастосунку. Код помилки, а також її повний опис буде виведено в консоль та збережено у логах вебзастосунку.

Під час роботи підсистеми можуть виникати такі виняткові ситуації:

- помилка під час роботи з базою даних (рис. 6.2);
- помилка під час отримання показників енергоспоживання (рис. 6.3);
- помилка під час завантаження прогнозу погоди (рис. 6.4).



Виникла помилка при роботі з базою даних!

Повну інформацію можна переглянути в консолі вебзастосунку.

Рисунок 6.2 – Повідомлення про помилку під час роботи з базою даних



Виникла помилка при отриманні показників енергоспоживання!

Повну інформацію можна переглянути в консолі вебзастосунку.

Рисунок 6.3 – Повідомлення про помилку під час отримання показників енергоспоживання



Виникла помилка при завантаженні прогнозу погоди!

Повну інформацію можна переглянути в консолі вебзастосунку.

Рисунок 6.4 – Повідомлення про помилку під час завантаження прогнозу погоди

Підсистема також підтримує обробку наступних виняткових ситуацій:

- помилка під час роботи модуля прогнозування (рис. 6.5);
- помилка під час створення оптимального розкладу користування побутовими приладами (рис. 6.6).



Виникла помилка при роботі модуля прогнозування!

Повну інформацію можна переглянути в консолі вебзастосунку.

Рисунок 6.5 – Повідомлення про помилку під час роботи модуля прогнозування



Виникла помилка при створенні оптимального розкладу!

Повну інформацію можна переглянути в консолі вебзастосунку.

Рисунок 6.6 – Повідомлення про помилку під час створення оптимального розкладу користування побутовими приладами

7 КЕРІВНИЦТВО КОРИСТУВАЧА

7.1 Призначення підсистеми

Підсистему інтелектуальної підтримки користувачів призначено для надавання рекомендацій мешканцям будинку щодо ефективного користування електроприладами з максимальним використанням альтернативних джерел енергії, а також з мінімальною залежністю від зовнішньої електромережі.

Підсистема надає користувачу можливість створювати оптимальний розклад роботи побутових приладів, підлаштувавшись під вимоги та звички мешканців, використовуючи спрогнозовану генерацію електроенергії на наступний день. Крім того, підсистема виконує цілодобовий контроль показників енергоспоживання та у разі високого споживання електроенергії із зовнішньої мережі відправляє повідомлення користувачу, використовуючи графічний інтерфейс.

7.2 Умови виконання підсистеми

Як апаратно-технічні засоби для експлуатації підсистеми інтелектуальної підтримки повинен використовуватися IBM-сумісний комп'ютер з характеристиками не нижче:

- процесор Intel Pentium 2266 МГц;
- оперативна пам'ять 512 МБ;
- 50 МБ вільного місця на жорсткому диску для зберігання виконуваних файлів;
- маніпулятор типу «миша» та / або клавіатура.

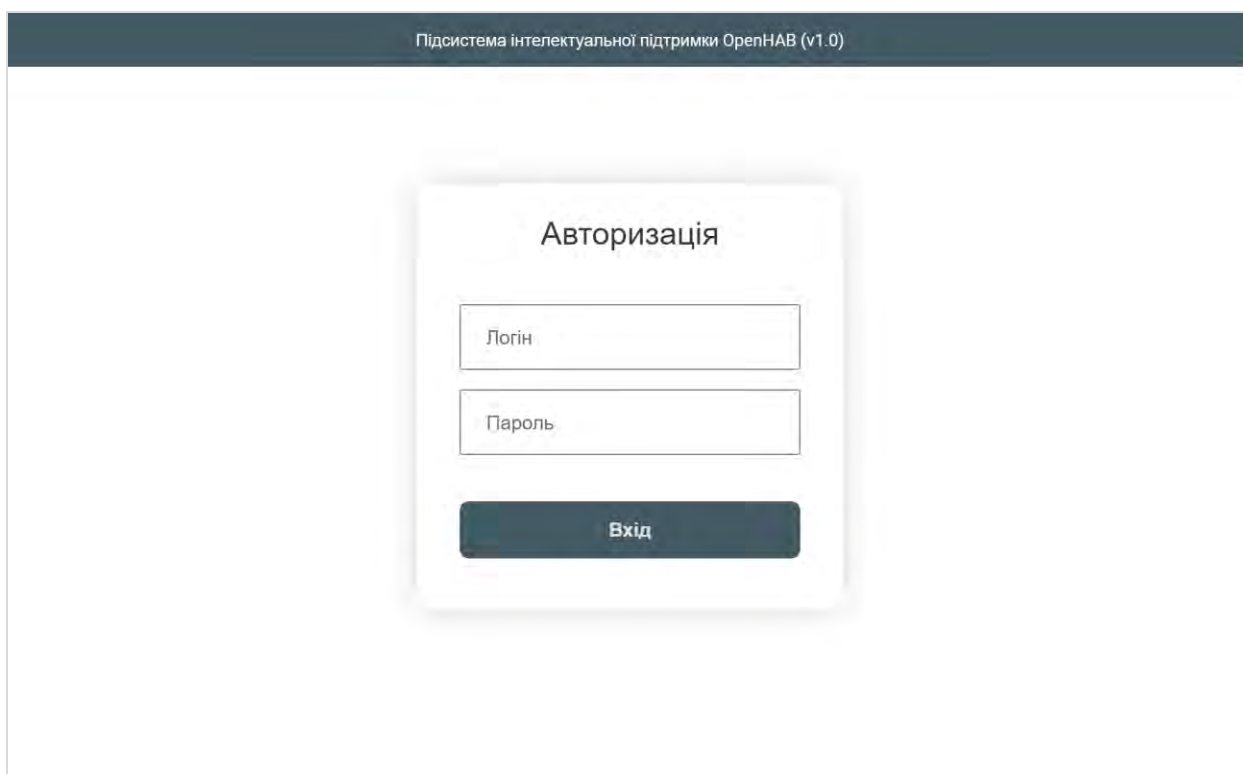
Як програмні засоби для коректного функціонування підсистеми інтелектуальної підтримки мають бути встановлені:

- операційна система не нижче Microsoft Windows 7 / Microsoft Windows Server 2012 / OS X 10.9 / Ubuntu Linux 12.04;

- виконавче середовище Java Runtime Environment (JRE) версії 1.8;
- виконавче середовище Python версії 3.10.0;
- бібліотека для Python: Keras-Preprocessing версії 1.1.2;
- бібліотека для Python: pandas версії 1.5.2;
- бібліотека для Python: sklearn версії 1.1.3;
- бібліотека для Python: joblib версії 1.2.0.

7.3 Виконання підсистеми

Робота з підсистемою інтелектуальної підтримки користувачів відбувається за допомогою будь-якого сучасного браузера. Для цього необхідно в адресний рядок ввести адресу та порт, на якому було розгорнуто вебзастосунок. Після цього відкриється сторінка авторизації (рис. 7.1), де необхідно ввести логін та пароль адміністратора, що було встановлено під час інсталяції підсистеми.



The image shows a web browser window displaying the authorization page of the 'Підсистема інтелектуальної підтримки OpenNAV (v1.0)'. The page has a dark blue header bar with the text 'Підсистема інтелектуальної підтримки OpenNAV (v1.0)'. The main content area is white and contains a central login form titled 'Авторизація'. The form consists of two input fields: 'Логін' (Login) and 'Пароль' (Password), followed by a dark blue button labeled 'Вхід' (Login).

Рисунок 7.1 – Сторінка авторизації

У разі введення правильних даних, після натискання на кнопку «Вхід» користувача буде переадресовано на головну сторінку вебзастосунку (рис. 7.2), де він зможе перейти у потрібний розділ для виконання певних задач.

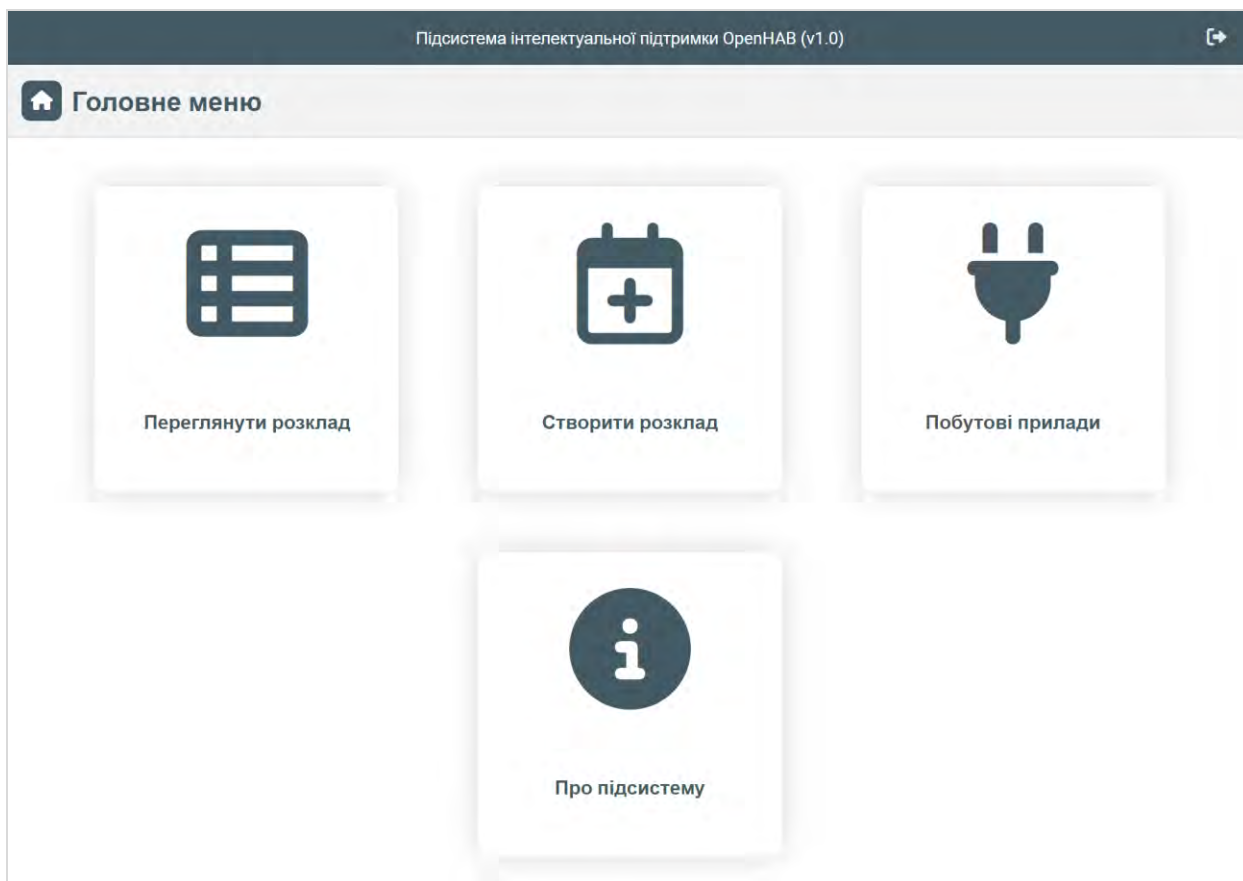


Рисунок 7.2 – Головне меню підсистеми інтелектуальної підтримки

На головній сторінці розташовано п'ять кнопок, які відповідають за наступні функції:

- кнопка «Переглянути розклад» відкриває календар зі створеним розкладом на сьогоднішній день;
- кнопка «Створити розклад» дозволяє користувачу створити оптимальний розклад користування побутовими приладами на наступні дні;
- кнопка «Побутові прилади» відображає список всіх електроприладів та також є можливість додавати нові прилади або видаляти непотрібні;

- кнопка «Про підсистему» дозволяє переглянути інформацію про розроблений вебзастосунок, його розробника та використані технології;
- кнопка виходу з облікового запису, що знаходиться у верхньому правому куті.

7.3.1 Сторінка «Календар»

Інтерфейс сторінки «Календар» зображено на рисунку 7.3, який відкривається натисканням кнопки «Переглянути розклад».

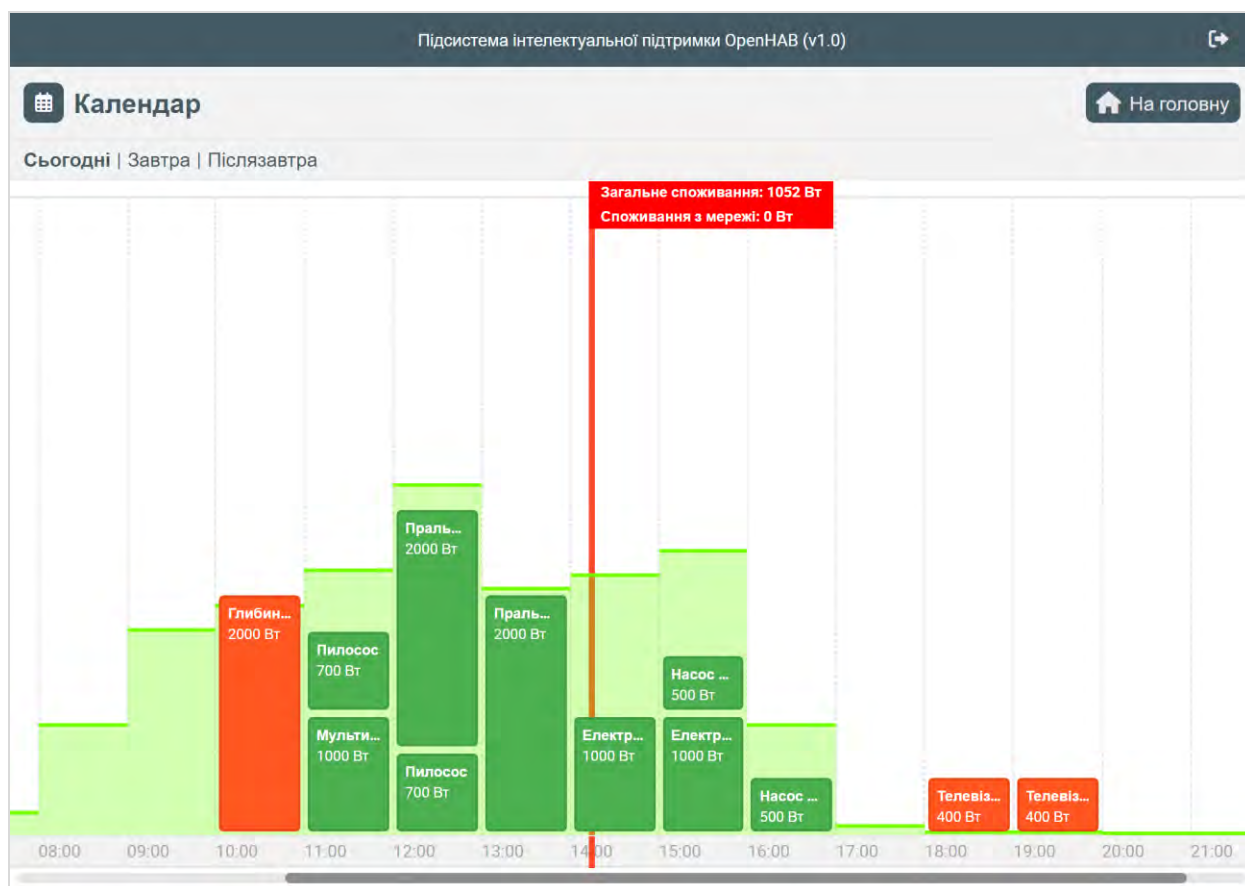


Рисунок 7.3 – Інтерфейс сторінки «Календар»

На цій сторінці відображено розклад користування побутовими приладами на всю добу. Кожен стовпчик означає одну годину та включає наступну інформацію:

- лаймовим кольором показано потужність генерації електроенергії;

- зеленими прямокутниками показано список електроприладів, які рекомендовано використовувати у відповідну годину;
- червоними прямокутниками показано побутові прилади, які не рекомендовано використовувати через недостатню генерації електроенергії.

Кожен електроприлад включає інформацію про його повну назву та номінальну потужність. Висота прямокутника для кожного електроприладу є пропорційною до його номінальної потужності, що дозволяє наочно оцінити споживання побутових приладів відносно генерації електроенергії. При натисканні або наведенні на прилад, буде виведено спливаючу підказку, яка відображає повну назву про електроприлад (рис. 7.4).

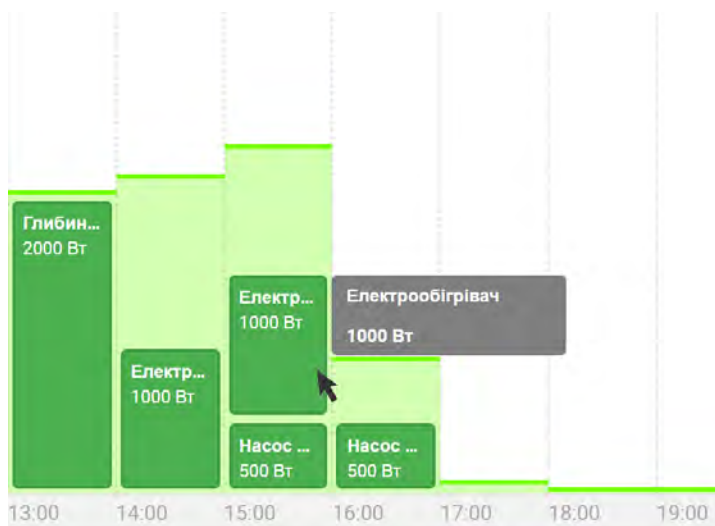


Рисунок 7.4 – Відображення спливаючої підказки з інформацією про прилад

У верхньому лівому куті є три кнопки: «Сьогодні», «Завтра» та «Післязавтра», які дозволяють переглядати розклад користування електроприладами на різні дні.

Якщо було обрано відображення розкладу на сьогоднішній день, то підсистема буде виконувати моніторинг показників електроспоживання у реальному часі та виводити у червоному прямокутнику споживання із зовнішньої мережі та загальне споживання.

Для випадків, коли перевищено споживання електроенергії із зовнішньої мережі, користувач отримає повідомлення, яке буде відображено

в помаранчевому прямокутнику справа (рис. 7.5). Це означає, що згенерованої електроенергії сонячними електростанціями та вітрогенераторами не вистачає для роботи увімкнених приладів, тому додатково використовується електроенергія із мережі та це збільшує витрати на комунальні послуги. Для запобігання використанню електроенергії із мережі необхідно вимкнути потужні побутові прилади, після чого повідомлення зникне.



Рисунок 7.5 – Повідомлення під час високого енергоспоживання із зовнішньої мережі

При натисканні на кнопку «На головну» користувач має змогу повернутися до головного меню вебзастосунку.

7.3.2 Сторінка «Планувальник розкладу»

Інтерфейс сторінки «Планувальник розкладу» відкривається натисканням кнопки «Створити розклад».

Спочатку користувачу буде запропоновано обрати день, для якого він бажає створити оптимальний розклад. Для цього буде відображено модальне вікно, яке буде містити три кнопки: «Сьогодні», «Завтра» та «Післязавтра» (рис. 7.6).

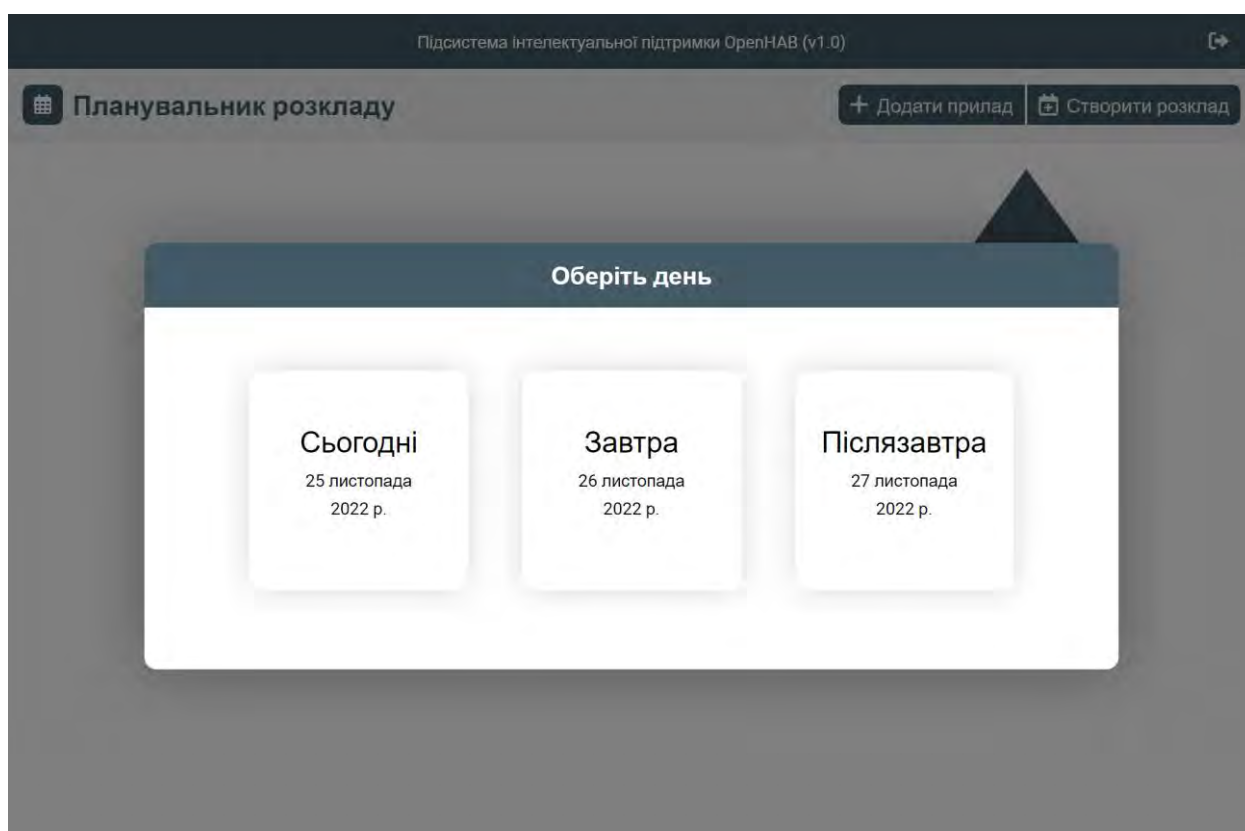


Рисунок 7.6 – Модальне вікно вибору дня для створення оптимального розкладу користування електроприладами

Після вибору потрібного дня, користувач перейде до інтерфейсу створення оптимального розкладу користування електроприладами (рис. 7.7).

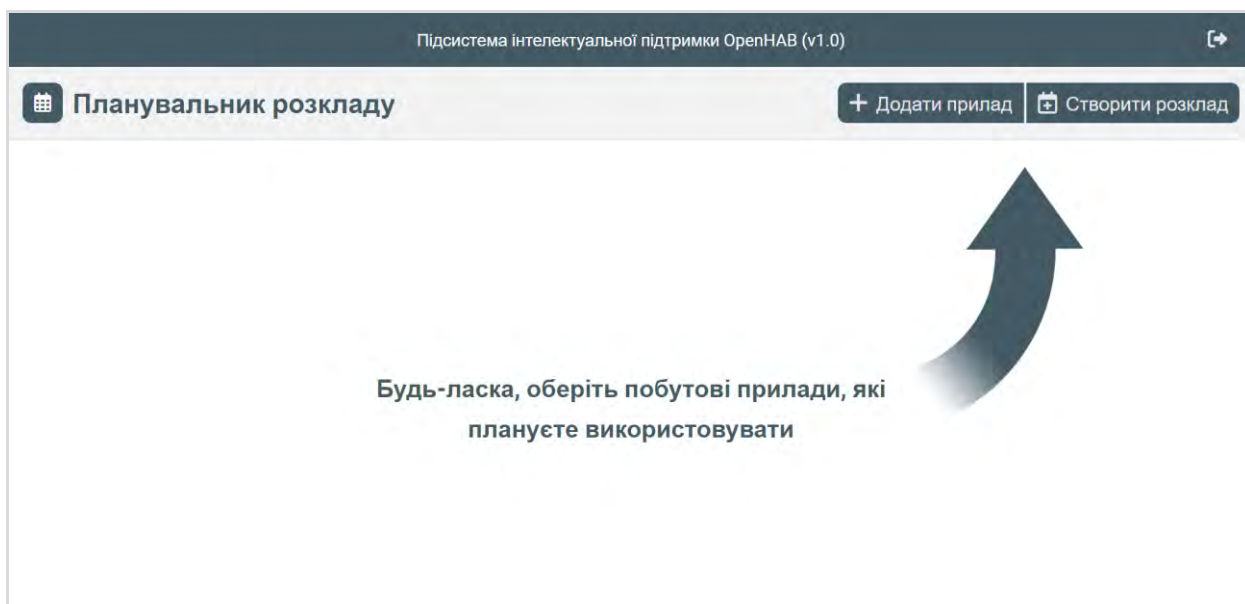


Рисунок 7.7 – Інтерфейс сторінки «Планувальник розкладу»

Далі необхідно натиснути на кнопку «Додати прилад», що знаходиться у верхньому правому куті, яка відкриє модальне вікно зі списком всіх електроприладів (рис. 7.8). З цього списку потрібно обрати прилади, які планується використовувати, натискаючи на кнопку «Додати».

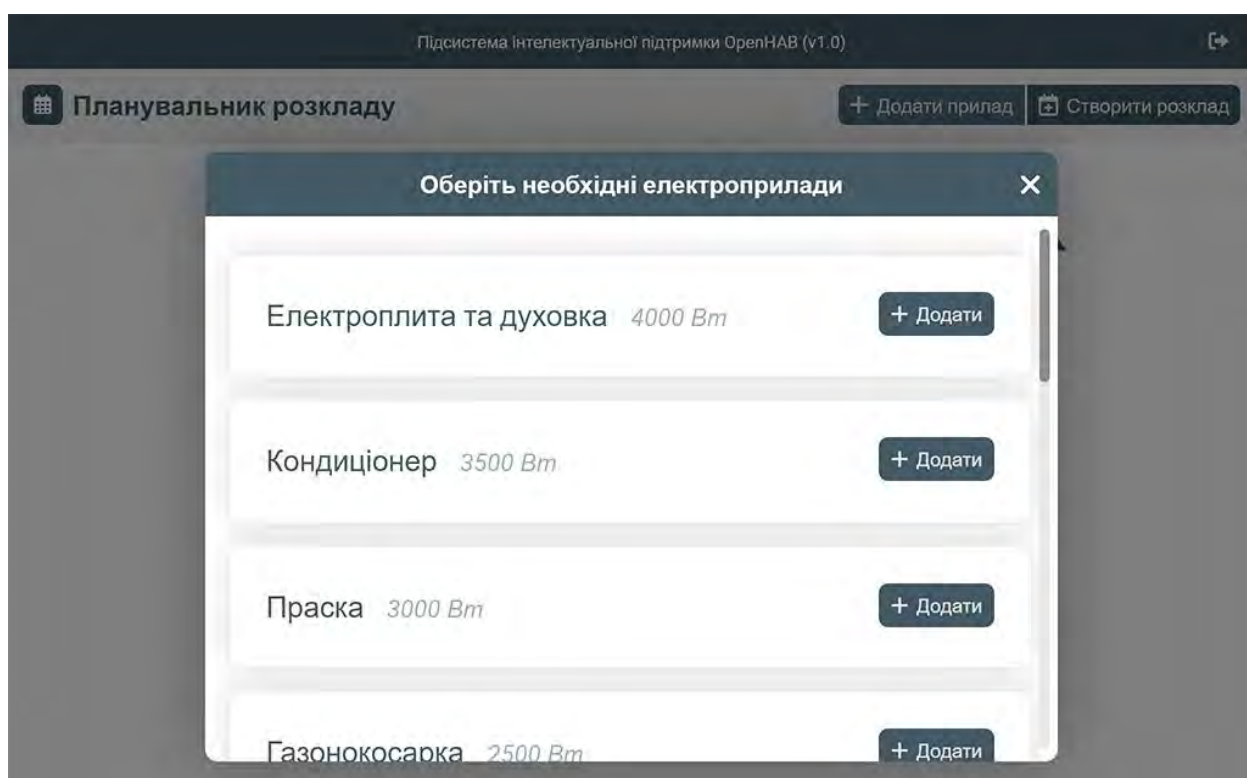


Рисунок 7.8 – Вибір необхідних електроприладів для створення розкладу

Після того, які усі необхідні побутові прилади було обрано, треба закрити модальне вікно.

Наступним кроком буде налаштування вимог користування побутовими приладами (рис. 7.9). Для кожного електроприладу необхідно обрати пріоритет та часовий інтервал користування.

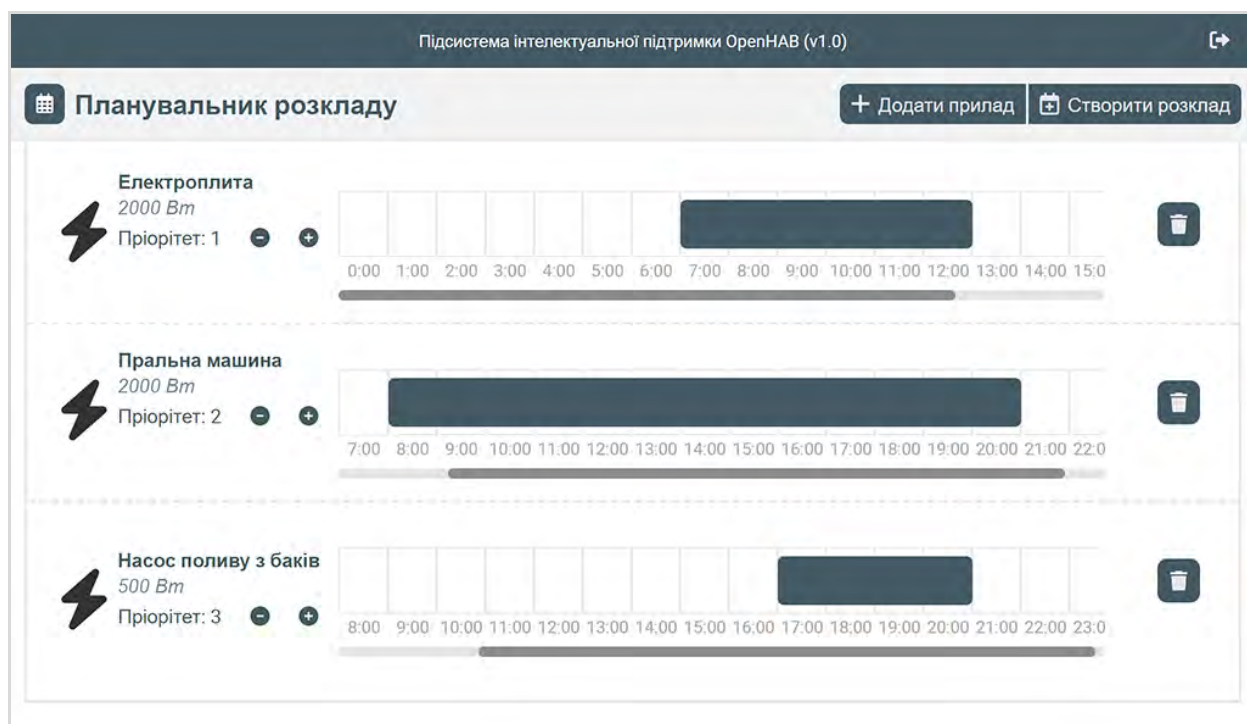


Рисунок 7.9 – Налаштування вимог користування електроприладами

Під пріоритетом розуміється число, яке може приймати значення від одного до п'яти, де п'ять означає найвищий пріоритет. Це дозволить обрати найважливіші побутові прилади та у разі недостатньої генерації електроенергії, підсистема запропонує не використовувати ті, що мають найменший пріоритет, тобто буде відображати їх червоним кольором.

Часовий інтервал дозволяє обрати зручний час користування певним електроприладом. Для цього необхідно натиснути на прямокутник з годиною, з якої починається часовий інтервал, після чого натиснути на прямокутник з годиною, яким завершується часовий інтервал.

Кнопка видалення дозволяє прибрати електроприлад, який не планується використовувати.

Натискання на кнопку «Створити розклад», що знаходиться у верхньому правому куті, запустить процес прогнозування генерації електроенергії та створення оптимального розкладу користування побутовими приладами, який буде тривати деякий час (рис. 7.10). Після цього користувача буде переадресовано на сторінку «Календар», де буде відображено створений розклад.



Рисунок 7.10 – Процес створення оптимального розкладу

7.3.3 Сторінка «Побутові прилади»

Інтерфейс сторінки «Побутові прилади» зображено на рисунку 7.11, який відкривається натисканням кнопки «Побутові прилади».

На цій сторінці відображено список всіх електроприладів, які використовуються під час створення оптимального розкладу.

Користувач має змогу додавати нові прилади та для цього потрібно у відповідні поля ввести повну назву побутового приладу, його номінальну потужність, що вимірюється у ватах, та час роботи у годинах. Після

натискання на кнопку «Додати», прилад буде додано до списку та відображено на сторінці.

Для того, щоб модифікувати вже існуючий побутовий прилад, необхідно натиснути на кнопку «Модифікувати», після чого у діалоговому вікні ввести нові значення та натиснути на кнопку «Зберегти».

Для видалення електроприладу, що більше не використовується потрібно натиснути на кнопку «Видалити».

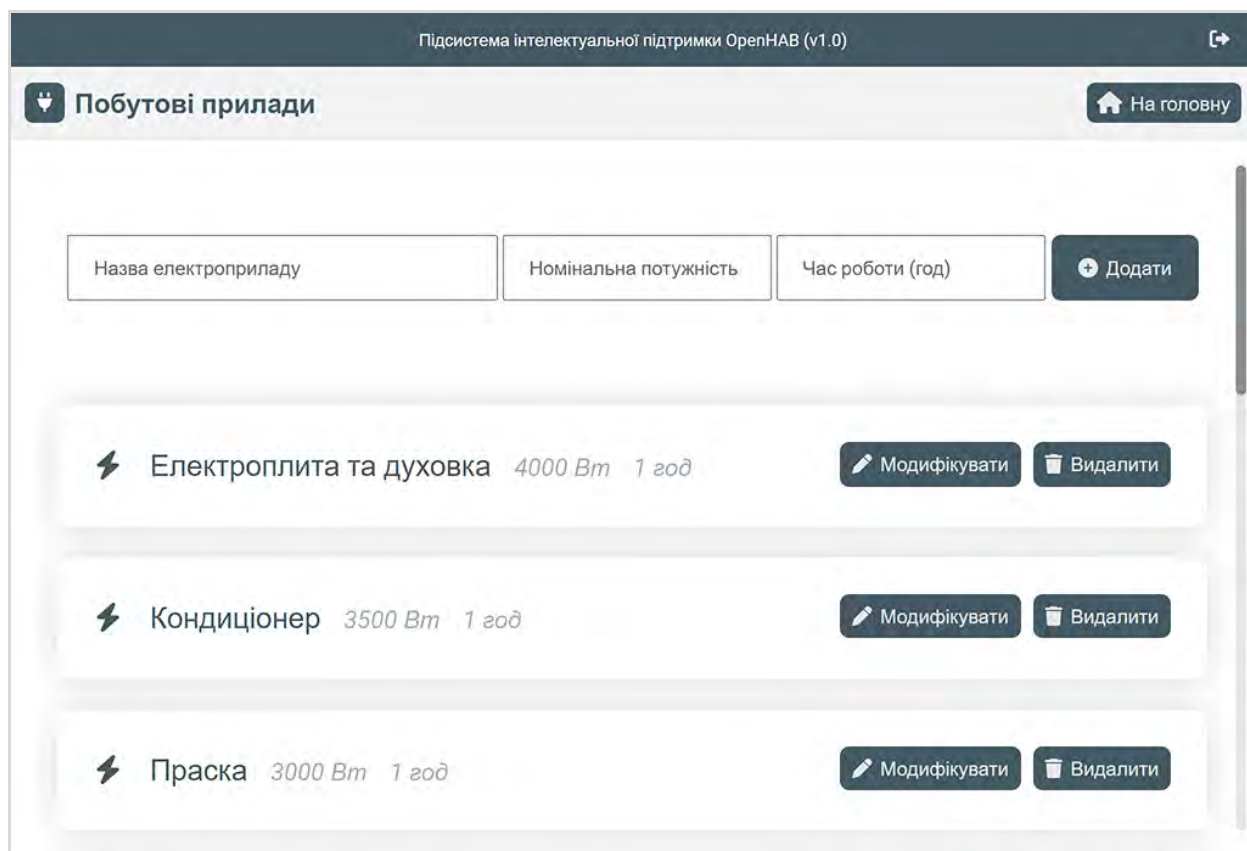


Рисунок 7.11 – Інтерфейс сторінки «Побутові прилади»

7.3.4 Сторінка «Про підсистему»

На сторінці «Про підсистему» користувач може ознайомитися з призначенням підсистеми інтелектуальної підтримки, її розробником та використаними технологіями (рис. 7.12).

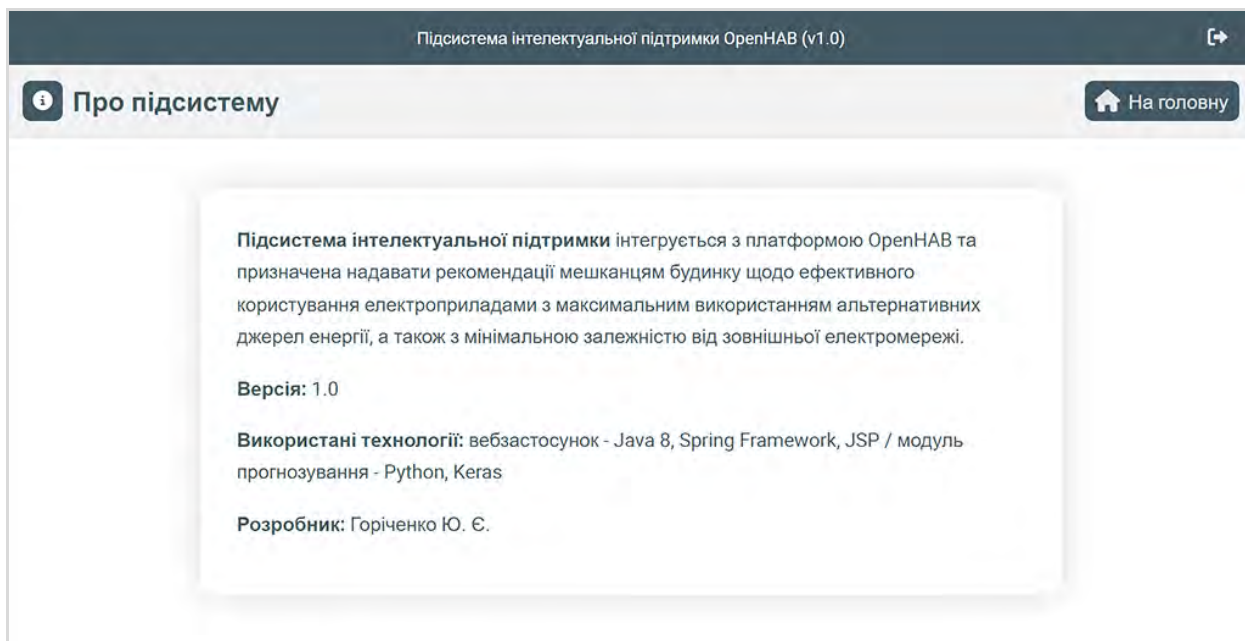


Рисунок 7.12 – Сторінка «Про підсистему»

7.4 Повідомлення

Під час роботи з підсистемою інтелектуальної підтримки передбачено виведення інформаційних повідомлень та діалогових повідомлень для підтвердження виконання важливих дій.

При невдалій спробі авторизуватися, користувач отримає відповідне інформаційне повідомлення (рис. 7.13).

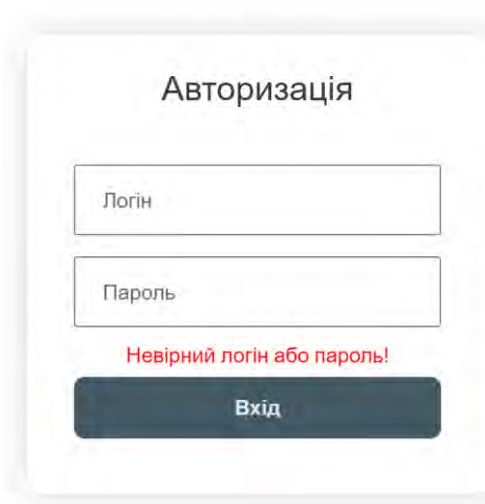


Рисунок 7.13 – Повідомлення про невдалу спробу авторизації

При спробі додати вже існуючий електроприлад на сторінці «Побутові прилади», буде виведено відповідне повідомлення (рис. 7.14).

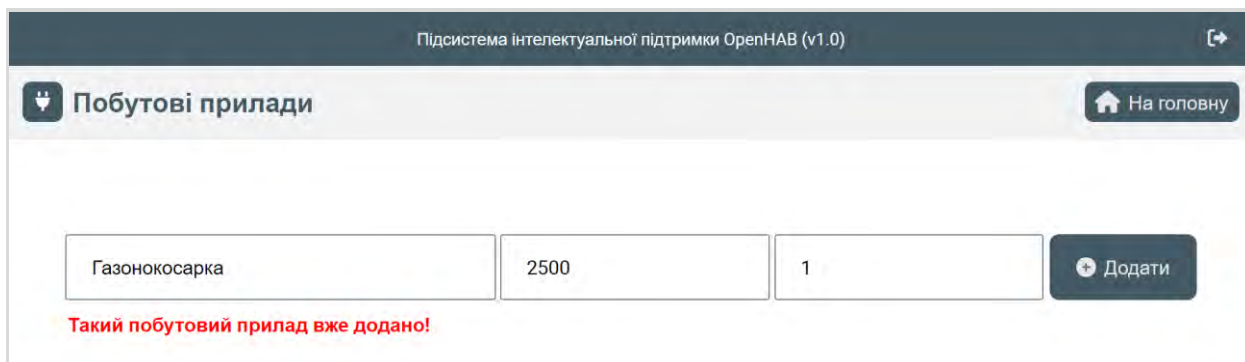


Рисунок 7.14 – Повідомлення при додаванні існуючого електроприладу

При спробі переглянути розклад на сторінці «Календар», який ще не створено, буде виведено інформаційне повідомлення (рис. 7.15).

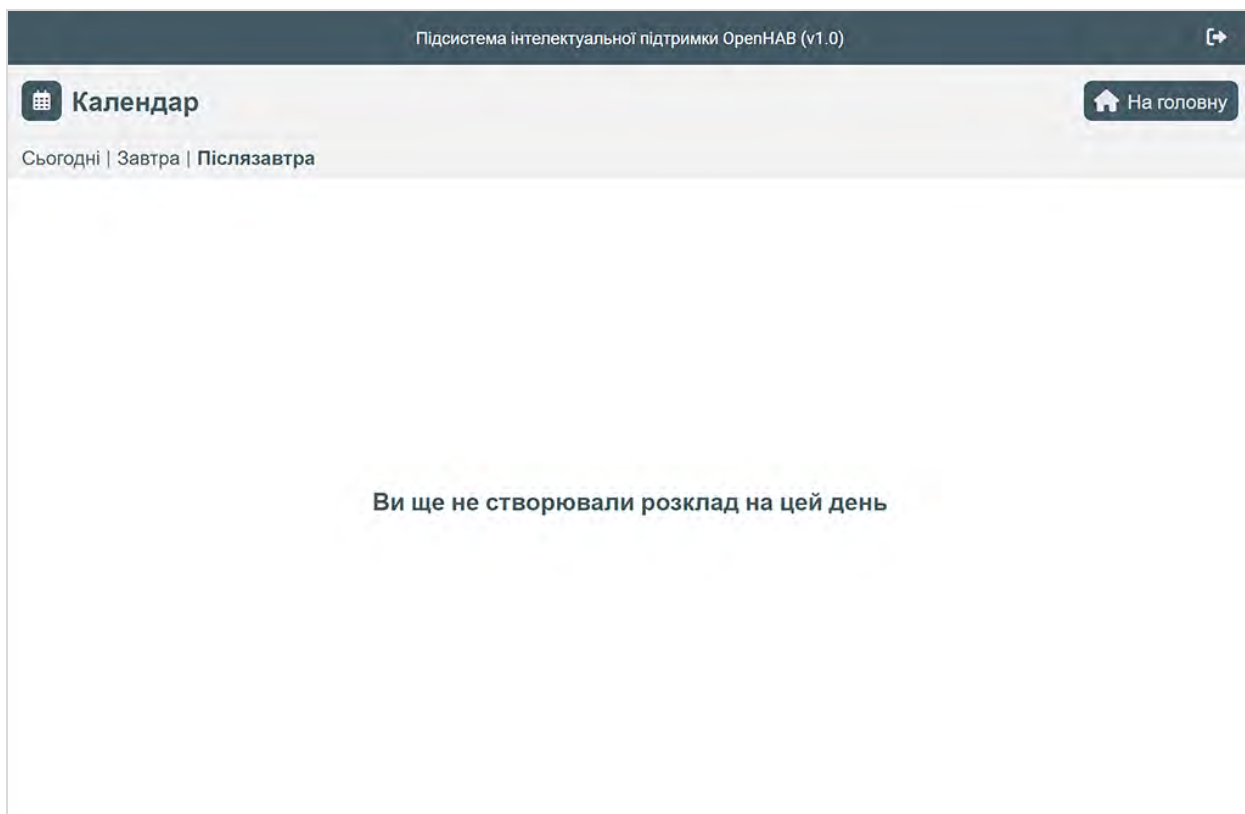


Рисунок 7.15 – Інформаційне повідомлення для ще не створеного розкладу

При натисканні на кнопку «Видалити» на сторінці «Побутові прилади», користувач отримає діалогове повідомлення, де необхідно підтвердити видалення бажаного приладу (рис. 7.16).

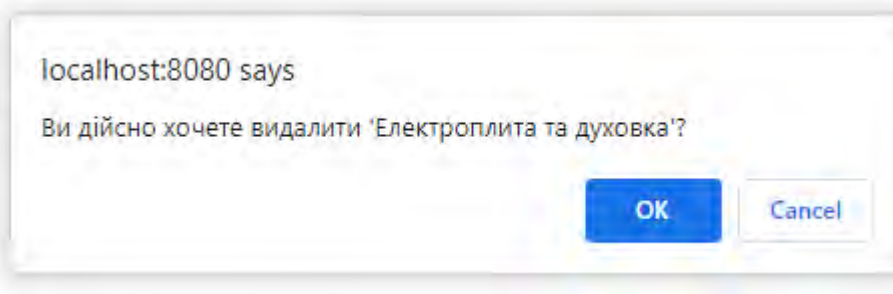


Рисунок 7.16 – Діалогове вікно підтвердження видалення електроприладу

При спробі створити оптимальний розклад на добу, для якої вже існує розклад, то користувач отримає діалогове повідомлення, де необхідно підтвердити видалення старого розкладу та створення нового (рис. 7.17).

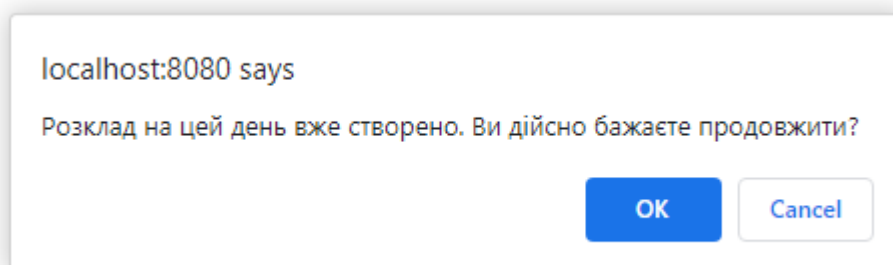


Рисунок 7.17 – Діалогове вікно підтвердження створення розкладу

Перед початком створення оптимального розкладу користування побутовими приладами, користувач отримає діалогове повідомлення про підтвердження запуску процесу оптимізації розкладу (рис. 7.18).

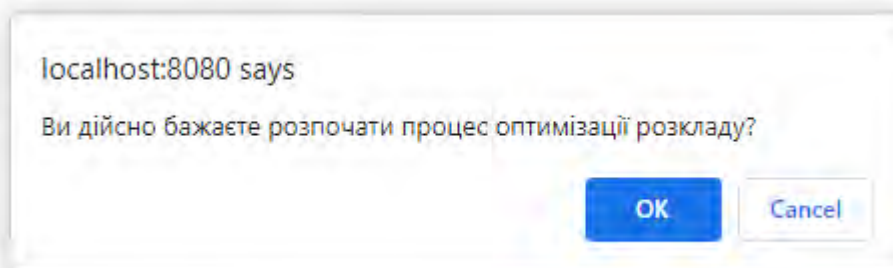


Рисунок 7.18 – Діалогове вікно підтвердження запуску процесу оптимізації розкладу

Оскільки підсистема інтелектуальної підтримки виконує цілодобовий моніторинг показників енергоспоживання, то у разі перевищення споживання електроенергії із зовнішньої мережі, користувач отримає відповідне повідомлення на сторінці «Календар» (рис. 7.19).

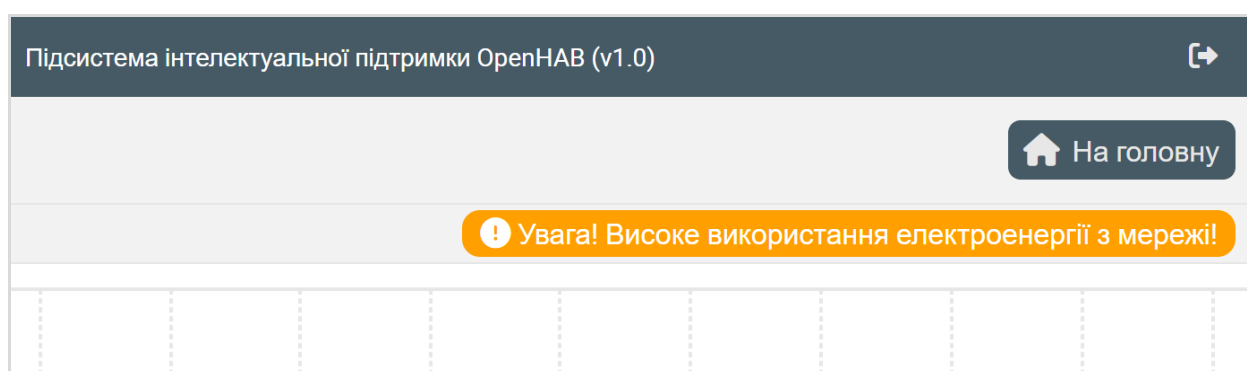


Рисунок 7.19 – Інформаційне повідомлення про високе споживання із зовнішньої електромережі

ВИСНОВКИ

В результаті виконання дипломної кваліфікаційної роботи магістра було проаналізовано предметну область та досліджено існуючі інтелектуальні методи в сфері ефективного споживання електроенергії.

В ході досліджень було розроблено комплексний інтелектуальний метод контролю енергоспоживання на основі прогнозування генерації електроенергії з максимальним використанням альтернативних джерел та оптимізації розкладу використання електроприладів, а також запропоновану власну модифікацію існуючого методу оптимізації.

Для розробки підсистеми інтелектуальної підтримки, яка має бути реалізована у вигляді вебзастосунку, було обрано вебфреймворк Spring, що написано на мові програмування Java. Для реалізації модулю прогнозування було прийнято рішення використовувати мову Python та бібліотеку Keras. Для зберігання всіх необхідних даних було використано базу даних MongoDB, яка найкраще підходить для зберігання великих об'ємів інформації. Завантаження прогнозу погоди відбувається з використанням метеосервісу «Visual Crossing».

Проведені комп'ютерні експерименти дозволили обрати найкращу нейромережеву модель для прогнозування генерації електроенергії, а також було висунуто ряд припущень щодо чинників, які впливали на точність прогнозування. Для навчання нейромережевих моделей було використано набір історичних даних споживання електроенергії, що було накопичено реальною СДА.

Розроблена підсистема інтелектуальної підтримки інтегрується з платформою OpenHAB та призначена для надання рекомендацій мешканцям будинку щодо ефективного користування електроприладами з максимальним використанням альтернативних джерел енергії, а також з мінімальною залежністю від зовнішньої електромережі.

Тестування розробленого програмного забезпечення показало, що підсистема інтелектуальної підтримки функціонує правильно та відповідає всім поставленим вимогам.

Наукова новизна роботи полягає в розробці інтелектуального методу контролю енергоспоживання, що відрізняється від існуючих комплексним підходом на основі прогнозування генерації електроенергії з максимальним використанням альтернативних джерел та оптимізації розкладу використання електроприладів на основі системи пріоритетів.

Практичне значення роботи полягає в тому, що інтеграція розробленої підсистеми інтелектуальної підтримки в СДА дозволить контролювати процеси генерації та споживання електроенергії, а також ефективно користуватися побутовими приладами та заощаджувати ресурси.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bruhathireddy, C. Design and Implementation of Home Automation system using Raspberry Pi [Text] / C. Bruhathireddy , G. N. Kodandaramaiah, M. Lakshmipathy // International Journal of Science, Technology & Management. – 2014. – Vol. 03, Issue 12. – P. 94-98.
2. Fanless Mini PC Intel Celeron J1900 [Electronic resource]. – Access mode: <https://aliexpress.com/item/32478994721.html>.
3. Smart Home Human Activity Simulation Tool for OpenHAB-based Research : Technical Report (final) : 10.1145 / A. Alotaibi, C. Perera / Cardiff University. – Cardiff, 2019. – 10 p.
4. rrd4j Persistence [Electronic resource]. – Access mode: <https://openhab.org/addons/persistence/rrd4j/>.
5. What is MQTT and how to use it with OpenHAB? [Electronic resource]. – Access mode: <https://smarthomeblog.net/mqtt-openhab/>.
6. Bernstein, C. MQTT (MQ Telemetry Transport) [Electronic resource] / C. Bernstein. – Access mode: <https://techtarget.com/iotagenda/definition/MQTT-MQ-Telemetry-Transport/>.
7. What is MQTT? Definition and Details [Electronic resource]. – Access mode: <https://paessler.com/it-explained/mqtt/>.
8. Modbus – Definition and Details [Electronic resource]. – Access mode: <https://paessler.com/it-explained/modbus/>.
9. Tanzilli, S. How to use the 1-WIRE bus [Electronic resource] / S. Tanzilli. – Access mode: <https://acmesystems.it/1wire/>.
10. Pankaj, P. Difference between Sensor and Actuator [Electronic resource] / P. Pankaj. – Access mode: <https://geeksforgeeks.org/difference-between-sensor-and-actuator/>.
11. What is a pure sine wave inverter? [Electronic resource]. – Access mode: <https://medium.com/@alphazee17/what-is-a-pure-sine-wave-inverter-116d01372e24/>.

12. Багатофункціональний вимірювач споживаної електроенергії PZEM-004T V3 [Електронний ресурс]. – Режим доступу: <https://arduino.ua/prod3619-mnogofynkcionalnii-izmeritel-potrebyaemoi-elektroe-nergii-pzem-004t-v3/>.

13. Scheduling optimization of smart homes based on demand response [Text] / [J. Zhu, F. Lauri, A. Koukam et al.] // Artificial Intelligence Applications and Innovations : 11th IFIP International Conference, Bayonne, 14–17 September 2015 : proceedings. – Bayonne: Springer, 2015. – P. 223-236.

14. Two-Stage Optimization Model for Smart House Daily Scheduling Considering User Perceived Benefits [Text] / [C. Deng, S. Zhang, W. Yang et al.] // Mathematics, Modelling, Simulation and Algorithms : 2018 International Conference, Chengdu, 25-26 March 2018 : proceedings. – Chengdu, Atlantis Press, 2018. – P. 64-68.

15. Multi-objective optimization model of electricity behavior considering the combination of household appliance correlation and comfort [Text] / [Z. Qu, N. Qu, Y. Liu et al.] // Journal of Electrical Engineering and Technology. – 2018. – Vol. 13, Issue 5. – P. 1821-1830.

16. Gupta, I. An hour wise device scheduling approach for demand side management in smart grid using particle swarm optimization [Text] / I. Gupta, G. N. Anandini, M. Gupta // NPSC : 19th National Power Systems Conference, Bhubaneswar, 19-21 December 2016 : proceedings. – Bhubaneswar: IEEE, 2016. – P. 1-6.

17. Home energy management in smart grid with renewable energy resources [Text] / [G. Zhou, G. Chen, G. Xu et al.] // Computer Modelling and Simulation : UKSim-AMSS 16th International Conference, Cambridge, 26-28 March 2014 : proceedings. – Cambridge : IEEE, 2014. – P. 351-356.

18. Iqbal, F. Optimal load scheduling using genetic algorithm to improve the load profile [Text] / F. Iqbal, K. Iqbal // ADRRI Journal of Engineering and Technology. – 2021. – Vol. 4, Issue 9. – P. 1-15.

19. Demand side management in smart grid using heuristic algorithm [Text] / [A. Thivy, K. Thamini, P. Narmadha et al.] // Emerging Enhancement in Engineering and Technology : Second International Conference, Tiruchirappalli, 18-19 March 2016 : proceedings. – Tiruchirappalli: IJARMATE, 2016. – P. 361-365.
20. Lee, E. Electricity usage scheduling in smart building environments using smart devices [Text] / E. Lee, H. Bahn // The Scientific World Journal. – 2013. – Vol. 2013. – P. 134–143.
21. Gopikrishna, P. B. Power Consumption Analysis and Prediction of a Smart Home Using ARIMA Model [Text] / P. B. Gopikrishna, J. A. Mathew. – Rochester: SSRN, 2021. – 9 p.
22. Taksir, M. Data-driven time series based prediction in smart home appliance energy consumption [Text] / M. Taksir, S. Aktar // International Journal of Computer Applications. – 2018. – Vol. 178, Issue 15. – P. 41-46.
23. LSTM forecasts for smart home electricity usage [Text] / [R. E. Alden, H. Gong, C. Ababei et al.] // Renewable Energy Research and Application : 9th International Conference, Glasgow, 27-30 September 2020 : proceedings. – Glasgow: IEEE, 2020. – P. 434-438.
24. Short-Term Residential Load Forecasting Based on Resident Behaviour Learning [Text] / [W. Kong, Z. Y. Dong, D. J. Hill et al.] // IEEE Transactions on Power Systems. – 2017. – Vol. 33, Issue 1. – P. 1087-1088.
25. Wang, Y. A Short-Term Residential Load Forecasting Model Based on LSTM Recurrent Neural Network Considering Weather Features [Text] / Y. Wang, N. Zhang, X. Chen // Energies. – 2021. – Vol. 14, Issue 10. – P. 1-13.
26. Prediction of electrical energy consumption through recurrent neural networks [Text] / [O. D. Diaz-Castillo, A. E. Puerto-Lara, J. A. Saenz-Leguizamon et al.] // Smart and Sustainable Cities : Fourth International Workshop, Buenos Aires, 28-30 October 2021 : proceedings. – Buenos Aires: CEUR, 2021. – P. 174-182.

27. Forecasting of Power Output of a PVPS Based on Meteorological Data Using RNN Approaches [Text] / [M. Beigi, H. Beigi, M. Torki et al.] // Sustainability. – 2022. – Vol. 14, Issue 5. – P. 1-12.
28. Amarasinghe, K. Deep neural networks for energy load forecasting [Text] / K. Amarasinghe, D. L. Marino, M. Manic // Industrial Electronics : IEEE 26th International Symposium, Edinburgh, 19-21 June 2017 : proceedings. – Edinburgh: IEEE, 2017. – P. 1483-1488.
29. Optimal Artificial Neural Network Type Selection Method for Usage in Smart House Systems [Text] / [V. Teslyuk, A. Kazarian, N. Kryvinska et al.] // Sensors. – 2021. – Vol. 21, Issue 1. – P. 1-14.
30. Wihartiko, F. D. Performance comparison of genetic algorithms and particle swarm optimization for model integer programming bus timetabling problem [Text] / F. D. Wihartiko, H. Wijayanti, F. Virgantari // IOP Conference Series: Materials Science and Engineering. – 2018. – Vol. 332, Issue 1. – P. 1-6.
31. Yambadwar, S. What is Web API and why we use it? [Electronic resource] / S. Yambadwar. – Access mode: <https://geeksforgeeks.org/what-is-web-api-and-why-we-use-it/>.
32. Pricing – OpenWeatherMap [Electronic resource]. – Access mode: <https://openweathermap.org/price/>.
33. Weatherbit API – Pricing [Electronic resource]. – Access mode: <https://weatherbit.io/pricing/>.
34. AccuWeather APIs | Packages [Electronic resource]. – Access mode: <https://developer.accuweather.com/packages/>.
35. Weather Data & Weather API Pricing | Visual Crossing [Electronic resource]. – Access mode: <https://visualcrossing.com/weather-data-editions/>.
36. JDBC Persistence [Electronic resource]. – Access mode: <https://openhab.org/addons/persistence/jdbc/>.
37. Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others [Electronic resource]. – Access mode: <https://altexsoft.com/blog/business/comparing-database-managem>

ent-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/.

38. Ivanovs, A. MongoDB vs MariaDB vs MySQL [Electronic resource] / A. Ivanovs. – Access mode: <https://geekflare.com/mongodb-vs-mariadb-vs-mysql/>.

39. GNU General Public License [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/GNU_General_Public_License/.

40. Free and open-source software [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Free_and_open-source_software/.

41. Understanding the key MongoDB pros and cons [Electronic resource]. – Access mode: <https://thinkautomation.com/our-two-cents/understanding-the-key-mongodb-pros-and-cons/>.

42. GNU Affero General Public License [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/GNU_Affero_General_Public_License/.

43. Yigal, A. SQLite vs. MySQL vs. PostgreSQL: A Comparison of Relational Databases [Electronic resource] / A. Yigal. – Access mode: <https://dzone.com/articles/sqlite-vs-mysql-vs-postgresql-a-comparison-of-rela/>.

44. Public domain [Electronic resource]. – Access mode: https://wiki.creativecommons.org/wiki/public_domain/.

45. Clark, J. Top 10 Web Development Frameworks [Electronic resource] / J. Clark. – Access mode: <https://blog.back4app.com/web-development-frameworks/>.

46. Django vs. Laravel | Important Difference between Laravel and Django [Electronic resource]. – Access mode: <https://javatpoint.com/django-vs-laravel>.

47. Lachhwani, P. How to choose the right backend technology? Node.js vs Spring boot vs Laravel vs Django [Electronic resource] / P. Lachhwani. – Access mode: <https://blog.iamprafull.com/how-to-choose-the-right-backend/>.

48. MIT License [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/MIT_License/.

49. Patel, J. List of 8 Best Backend Frameworks To Use in Web App Development [Electronic resource] / J. Patel. – Access mode: <https://monocubed.com/blog/best-backend-frameworks/>.

50. BSD licenses [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/BSD_licenses/.

51. Fadatare, R. Spring Framework Overview [Electronic resource] / R. Fadatare. – Access mode: <https://javaguides.net/2018/06/spring-framework-overview.html/>.

52. Glib, O. Most Popular Backend Frameworks in 2021-2022 (Pros and Cons, What to Choose) [Electronic resource] / O. Glib. – Access mode: <https://acropolium.com/blog/most-popular-backend-frameworks-in-2021-2022-pros-and-cons-what-to-choose/>.

53. Java Spring Framework Vs. Ruby on Rails [Electronic resource]. – Access mode: <https://siliconvalleycloudit.com/java-spring-framework-vs-ruby-on-rails/>.

54. Spring MVC Tutorial [Electronic resource]. – Access mode: <https://javatpoint.com/spring-mvc-tutorial/>.

55. JSP Tutorial [Electronic resource]. – Access mode: <https://tutorialspoint.com/jsp/index.htm/>.

56. Sahu, A. 7 Best Java IDEs and Editors in 2022 [Electronic resource] / A. Sahu. – Access mode: <https://turing.com/blog/best-java-ides-and-editors-in-2022/>.

57. IntelliJ IDEA – the Leading Java and Kotlin IDE [Electronic resource]. – Access mode: <https://jetbrains.com/idea/>.

58. Скороход, Д. Що має знати Data Scientist. Аналіз вакансій в Україні та Каліфорнії [Електронний ресурс] / Д. Скороход. – Режим доступу: <https://dou.ua/lenta/articles/knowledge-for-data-scientist/>.

59. AI Programming: 5 Most Popular AI Programming Languages [Electronic resource]. – Access mode: <https://existek.com/blog/ai-programming-and-ai-programming-languages/>.

60. Python vs. R: What's the Difference? [Electronic resource]. – Access mode: <https://ibm.com/cloud/blog/python-vs-r/>.

61. Rohner, T. Python vs. C++ for Machine Learning [Electronic resource] / T. Rohner. – Access mode: <https://netguru.com/blog/cpp-vs-python/>.
62. AI Programming: 5 Most Popular AI Programming Languages [Electronic resource]. – Access mode: <https://existek.com/blog/ai-programming-and-ai-programming-languages/>.
63. Top 10 Python IDE and Code Editors in 2022 [Electronic resource]. – Access mode: <https://geeksforgeeks.org/top-10-python-ide-and-code-editors-in-2020/>.
64. PyCharm – The Python IDE for Professional Developers [Electronic resource]. – Access mode: <https://jetbrains.com/pycharm/>.
65. Machine learning with Python: best libraries, tutorials, use case examples [Electronic resource]. – Access mode: <https://svitla.com/blog/machine-learning-with-python-best-libraries-tutorials-use-case-examples/>.
66. Scikit-learn vs. TensorFlow vs. PyTorch vs. Keras [Electronic resource]. – Access mode: <https://ritza.co/articles/scikit-learn-vs-tensorflow-vs-pytorch-vs-keras/>.
67. In-Depth Guide on Deep Learning Frameworks - Keras, TensorFlow, PyTorch [Electronic resource]. – Access mode: <https://turing.com/kb/guide-on-deep-learning-frameworks-keras-tensorflow-pytorch/>.
68. The DAO Pattern in Java [Electronic resource]. – Access mode: <https://baeldung.com/java-dao-pattern/>.
69. Collections – MongoDB Compass [Electronic resource]. – Access mode: <https://mongodb.com/docs/compass/current/collections/>.
70. Comma-separated values [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Comma-separated_values/.
71. Weather Data Documentation – Visual Crossing Weather [Electronic resource]. – Access mode: <https://visualcrossing.com/resources/documentation/weather-data/weather-data-documentation/>.
72. Kennedy, J. Particle swarm optimization [Text] / J. Kennedy, R. Eberhart // Neural Networks : International Conference, Perth, 27 November

1995 – 01 December 1995 : proceedings. – Perth: IEEE, 1995. – Vol. 4. – P. 1942-1948.

73. Li, C. Improved Particle Swarm Optimization Algorithms for Optimal Designs with Various Decision Criteria [Text] / C. Li, D. C. Coster // Mathematics. – 2022. – Vol. 10, Issue 13. – P. 1-16.

74. Modules – Introduction to Spring Framework [Electronic resource]. – Access mode: <https://docs.spring.io/spring-framework/docs/3.0.0.M4/reference/html/ch01s02.html/>.

75. Project Lombok – Order license info [Electronic resource]. – Access mode: <https://projectlombok.org/order-license-info/>.

76. Project Lombok [Electronic resource]. – Access mode: <https://github.com/projectlombok/lombok/>.

77. Ray, S. Building your First Neural Network on a Structured Dataset (using Keras) [Electronic resource] / S. Ray. – Access mode: <https://medium.com/analytics-vidhya/build-your-first-neural-network-model-on-a-structured-dataset-using-keras-d9e7de5c6724/>.

78. Brownlee, J. What Is Data Preparation in a Machine Learning Project [Electronic resource] / J. Brownlee. – Access mode: <https://machinelearningmastery.com/what-is-data-preparation-in-machine-learning/>.

79. Kumar, A. Feature Scaling in Machine Learning: Python Examples [Electronic resource] / A. Kumar. – Access mode: <https://vitalflux.com/python-improve-model-performance-using-feature-scaling/>.

80. Субботін, С. О. Нейронні мережі : теорія та практика: навч. посіб. [Текст] / С. О. Субботін. – Житомир : Вид. О. О. Євенок, 2020. – 184 с.

81. Keras – Models [Electronic resource]. – Access mode: https://tutorialspoint.com/keras/keras_models.htm/.

82. Oppermann, A. Optimization in Deep Learning: AdaGrad, RMSProp, ADAM [Electronic resource] / A. Oppermann. – Access mode: <https://artemoppermann.com/optimization-in-deep-learning-adagrad-rmsprop-adam/>.

83. Root-Mean-Square Error in R Programming [Electronic resource]. – Access mode: <https://geeksforgeeks.org/root-mean-square-error-in-r-programming/>.

84. Шевчук, Р. П. Сучасні інструментальні засоби розробки користувацького інтерфейсу : опорний конспект лекцій для студентів за спеціальностями 7.05010301 «Програмне забезпечення систем», 8.05010301 «Програмне забезпечення систем» [Текст] / Р. П. Шевчук. – Тернопіль: ТНЕУ, 2012. – 103 с.

85. jQuery API [Electronic resource]. – Access mode: <https://api.jquery.com/>.

86. License jQuery Foundation [Electronic resource]. – Access mode: <https://jquery.org/license/>.

87. Grid system – Bootstrap v5.2 [Electronic resource]. – Access mode: <https://getbootstrap.com/docs/5.2/layout/grid/>.

88. License FAQs – Bootstrap v5.2 [Electronic resource]. – Access mode: <https://getbootstrap.com/docs/5.2/about/license/>.

89. FontAwesome [Electronic resource]. – Access mode: <https://github.com/FortAwesome/Font-Awesome/>.

90. Види тестування сайтів: класифікація, етапи та типи тестів [Електронний ресурс]. – Режим доступу: <https://webtune.com.ua/statti/web-rozrobka/vydy-testuvannya-sajtiv/>.

91. Basques, K. Simulate mobile devices with Device Mode [Electronic resource] / K. Basques, S. Emelianova. – Access mode: <https://developer.chrome.com/docs/devtools/device-mode/>.

92. Apache Tomcat 9 – Tomcat Setup [Electronic resource]. – Access mode: <https://tomcat.apache.org/tomcat-9.0-doc/setup.html/>.

ДОДАТОК А
ТЕКСТ ПРОГРАМИ

A.1 Файл OpenhabSmartSupportApplication.java

```
package nuzp.yuliy_gorichenko.openhab_smart_support;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class OpenhabSmartSupportApplication {
    public static void main(String[] args) {
        SpringApplication.run(OpenhabSmartSupportApplication.class, args);
    }
}
```

A.2 Файл TimetableController.java

```
package nuzp.yuliy_gorichenko.openhab_smart_support.controller;
import com.google.gson.JsonArray;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;
import nuzp.yuliy_gorichenko.openhab_smart_support.dao.TimetableDao;
import nuzp.yuliy_gorichenko.openhab_smart_support.model.PowerGeneration;
import nuzp.yuliy_gorichenko.openhab_smart_support.model.Timetable;
import nuzp.yuliy_gorichenko.openhab_smart_support.model.WorkingAppliance;
import nuzp.yuliy_gorichenko.openhab_smart_support.service.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.view.RedirectView;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.*;

@Controller
public class TimetableController {
    /**
     * Сервіс для роботи з електроприладами
     */
    @Autowired
    private AppliancesService appliancesService;

    /**
     * Сервіс для завантаження прогнозу погоди
     */
    @Autowired
    private WeatherService weatherService;

    /**
     * Сервіс для прогнозування генерації енергії
     */
    @Autowired
    private PredictService predictService;
```

```

/**
 * Сервіс для створення оптимального розкладу
 */
@Autowired
private PSOService psoService;

/**
 * Сервіс для збереження розкладу
 */
@Autowired
private TimetableDao timetableDao;

/**
 * Сервіс для контролю споживання в реальному часі
 */
@Autowired
private RealtimeConsumptionService realtimeConsumptionService;

/**
 * Контролер розкладу на сьогодні
 */
@RequestMapping(value = "/timetable/today", method = RequestMethod.GET)
public ModelAndView timetableToday(@RequestParam(defaultValue = "today")
    String period) {
    ModelAndView model = new ModelAndView("calendar");
    model.addObject("period", period);

    Timetable timetable = timetableDao.findByDate(
        timetableDao.periodToDate(period));
    if (timetable != null) {
        model.addObject("timetableExists", true);
        model.addObject("timetable", timetable.getAppliances());
        model.addObject("generation", timetable.getPowerGeneration());
    } else {
        model.addObject("timetableExists", false);
    }

    return model;
}

/**
 * Контролер розкладу на завтра
 */
@RequestMapping(value = "/timetable/tomorrow", method =
    RequestMethod.GET)
public ModelAndView timetableTomorrow(@RequestParam(defaultValue =
    "tomorrow") String period) {
    ModelAndView model = new ModelAndView("calendar");
    model.addObject("period", period);

    Timetable timetable = timetableDao.findByDate(
        timetableDao.periodToDate(period));
    if (timetable != null) {
        model.addObject("timetableExists", true);
        model.addObject("timetable", timetable.getAppliances());
        model.addObject("generation", timetable.getPowerGeneration());
    } else {
        model.addObject("timetableExists", false);
    }

    return model;
}

```

```

/**
 * Контролер розкладу на післязавтра
 */
@RequestMapping(value = "/timetable/aftertomorrow", method =
    RequestMethod.GET)
public ModelAndView timetableAfterTomorrow(@RequestParam(defaultValue =
    "aftertomorrow") String period) {
    ModelAndView model = new ModelAndView("calendar");
    model.addObject("period", period);

    Timetable timetable = timetableDao.findByDate(
        timetableDao.periodToDate(period));
    if (timetable != null) {
        model.addObject("timetableExists", true);
        model.addObject("timetable", timetable.getAppliances());
        model.addObject("generation", timetable.getPowerGeneration());
    } else {
        model.addObject("timetableExists", false);
    }

    return model;
}

/**
 * Контролер вибіру побутових приладів
 */
@RequestMapping(value = "/timetable/new", method = RequestMethod.GET)
public ModelAndView newTimetable() throws Exception {
    ModelAndView model = new ModelAndView("planner");
    model.addObject("appliancesList", appliancesService.loadAll());

    DateTimeFormatter formatter = DateTimeFormatter
        .ofPattern("d MMMM y 'p'."), new Locale("uk", "UA"));
    LocalDateTime now = LocalDateTime.now();
    model.addObject("today_date", formatter.format(now));
    model.addObject("tomorrow_date", formatter.format(now.plusDays(1)));
    model.addObject("aftertomorrow_date",
        formatter.format(now.plusDays(2)));

    return model;
}

/**
 * Створити оптимальний розклад
 */
@RequestMapping(value = "/timetable/create", method = RequestMethod.POST)
public RedirectView realtimeData(@RequestParam("json") String json,
    @RequestParam("period") String period)
    throws Exception {
    // Отримати побутові прилади, обрані користувачем
    List<WorkingAppliance> appliances = new ArrayList<>();
    JsonElement jsonElement = JsonParser.parseString(json);
    JsonArray array = jsonElement.getAsJsonArray();
    array.forEach(element -> {
        JsonObject object = element.getAsJsonObject();
        String name = object.get("deviceName").getString();
        JsonArray hours = object.getAsJsonArray("timeRange");
        int startHour = hours.get(0).getAsInt();
        int endHour = hours.get(hours.size() - 1).getAsInt();
        int priority = object.get("priority").getAsInt();

        WorkingAppliance appliance = new WorkingAppliance(
            appliancesService.loadByName(name));
        appliance.setStartHour(startHour);
    });
}

```

```

        appliance.setEndHour(endHour);
        appliance.setPriority(priority);
        appliances.add(appliance);
    });

    // Завантажити прогноз погоди
    List<String> weather = weatherService.getWeather(
        timetableDao.periodToDate(period));
    weatherService.prepareData(weather, predictService.getPythonPath());

    // Спрогнозувати генерацію електроенергії
    List<PowerGeneration> powerGeneration = predictService
        .predictPowerGeneration();

    // Створити оптимальний розклад
    psoService.setElectricAppliances(appliances);
    psoService.setPowerGeneration(powerGeneration);
    psoService.optimizeTimetable();
    List<List<WorkingAppliance>> timetable = psoService.getTimetable();
    timetableDao.save(new Timetable(null, timetableDao
        .periodToDate(period), timetable, powerGeneration));

    return new RedirectView("/timetable/" + period);
}

@RequestMapping(value = "/timetable/realtime", method =
    RequestMethod.GET)
public ResponseEntity<Map<String, Object>> realtimeData() {
    Map<String, Object> realData = new HashMap<>();
    realData.put("current", realtimeConsumptionService
        .getGridConsumption().getValue());
    realData.put("total", realtimeConsumptionService
        .getTotalConsumption().getValue());
    realData.put("isOverload", realtimeConsumptionService.isOverload());

    return new ResponseEntity<>(realData, HttpStatus.OK);
}
}

```

A.3 Файл AppliancesService.java

```

package nuzp.yuliy_gorichenko.openhab_smart_support.service;

import nuzp.yuliy_gorichenko.openhab_smart_support.exception.*;
import nuzp.yuliy_gorichenko.openhab_smart_support.model.ElectricAppliance;
import java.util.List;

public interface AppliancesService {
    List<ElectricAppliance> loadAll() throws DatabaseException;

    void addAppliance(ElectricAppliance appliance) throws
        ApplianceAlreadyExistsException, DatabaseException;

    void deleteAppliance(String id) throws
        ApplianceNotFoundException, DatabaseException;

    ElectricAppliance loadByName(String name);
}

```

A.4 Файл PredictService.java

```
package nuzp.yuliy_gorichenko.openhab_smart_support.service;

import nuzp.yuliy_gorichenko.openhab_smart_support.model.PowerGeneration;
import org.springframework.scheduling.annotation.Scheduled;
import java.util.List;

public interface PredictService {
    List<PowerGeneration> prepareHistoricalData();

    List<PowerGeneration> predictPowerGeneration();

    @Scheduled
    void trainModel();

    String getPythonPath();
}
```

A.5 Файл PSOService.java

```
package nuzp.yuliy_gorichenko.openhab_smart_support.service;

import nuzp.yuliy_gorichenko.openhab_smart_support.model.PowerGeneration;
import nuzp.yuliy_gorichenko.openhab_smart_support.model.WorkingAppliance;
import java.util.List;

public interface PSOService {
    void optimizeTimetable();

    void setPowerGeneration(List<PowerGeneration> generation);

    void setElectricAppliances(List<WorkingAppliance> appliance);

    List<List<WorkingAppliance>> getTimetable();
}
```

A.6 Файл RealtimeConsumptionService.java

```
package nuzp.yuliy_gorichenko.openhab_smart_support.service;

import nuzp.yuliy_gorichenko.openhab_smart_support.model.GridConsumption;
import nuzp.yuliy_gorichenko.openhab_smart_support.model.TotalConsumption;
import org.springframework.scheduling.annotation.Scheduled;

public interface RealtimeConsumptionService {
    GridConsumption getGridConsumption();

    TotalConsumption getTotalConsumption();

    boolean isOverload();

    @Scheduled
    void analyzeConsumption();
}
```

A.7 Файл WeatherService.java

```
package nuzp.yuliy_gorichenko.openhab_smart_support.service;

import org.springframework.scheduling.annotation.Scheduled;
import java.util.List;

public interface WeatherService {
    List<String> getWeather(String date) throws Exception;

    void prepareData(List<String> data, String path);

    @Scheduled
    void saveHistoricalWeather();
}
```

A.8 Файл train.py

```
import math
import joblib
import pandas as pd
from keras.layers import Dense
from keras.models import Sequential
from sklearn.model_selection import train_test_split
from sklearn import preprocessing

class TrainService:
    def __init__(self):
        # Підготовка навчальних даних
        weather = pd.read_csv("train_data/weather.csv")
        power = pd.read_csv("train_data/power.csv")
        weather = weather[['temp', 'windgust', 'solarradiation']]
        self.x = weather.astype(float).to_numpy()
        self.y = power['power'].to_numpy()

    def train(self):
        # Створення навчальної та тестової вибірки
        x_train, x_test, y_train, y_test = train_test_split(self.x, self.y,
                                                            test_size=0.2, random_state=42)

        # Розрахунок середнього значення
        mean = x_train.mean(axis=0)
        # Розрахунок стандартного відхилення
        std = x_train.std(axis=0)
        # Збереження даних у файл
        joblib.dump(mean, "model/mean.h5")
        joblib.dump(std, "model/std.h5")
        # Стандартизація даних
        x_train -= mean
        x_train /= std
        x_test -= mean
        x_test /= std
        # Створення моделі прогнозування
        model = Sequential()
        model.add(Dense(128, activation='relu',
                        input_shape=(x_train.shape[1],)))
        model.add(Dense(24))
        model.add(Dense(1))
```

```

model.compile(optimizer='adam', loss='mse', metrics=['mae'])
model.fit(x_train, y_train, epochs=100, batch_size=24, verbose=2)
mse, mae = model.evaluate(x_test, y_test, verbose=2)
# Збереження моделі у файл
model.save("model/model.h5")
# Розрахунок функцій помилок
print('RMSE = ', math.sqrt(mse))
print('MAE = ', mae)
print('MSE = ', mse)

if __name__ == "__main__":
    service = TrainService()
    service.train()

```

A.9 Файл predict.py

```

import joblib
import pandas as pd
from keras.saving.save import load_model

class PredictService:
    def __init__(self):
        # Підготовка даних для прогнозування
        weather = pd.read_csv("predict_data/weather.csv")
        self.dataset = pd.DataFrame()
        self.dataset['datetime'] = weather['datetime']
        weather = weather[['temp', 'windgust', 'solarradiation']]
        self.x = weather.astype(float).to_numpy()

    def predict(self):
        # Завантаження середнього значення
        mean = joblib.load("model/mean.h5")
        # Завантаження стандартного відхилення
        std = joblib.load("model/std.h5")
        # Стандартизація даних
        self.x -= mean
        self.x /= std
        # Завантаження моделі прогнозування
        model = load_model("model/model.h5")
        # Прогнозування генерації електроенергії
        y = model.predict(self.x)
        # Збереження спрогнозованих даних у файл
        self.dataset['power'] = y
        self.dataset.to_csv("predict_data/power.csv")

if __name__ == "__main__":
    service = PredictService()
    service.predict()

```

ДОДАТОК Б
СЛАЙДИ ПРЕЗЕНТАЦІЇ

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота магістра

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНИХ МЕТОДІВ КОНТРОЛЮ ЕНЕРГОСПОЖИВАННЯ

Виконав:

ст. гр. КНТ-121м

Ю. Є. Горіченко

Керівник:

к.т.н., доцент

А. В. Пархоменко

Рисунок Б.1 – Слайд №1



Рисунок Б.2 – Слайд №2

МЕТА ТА ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

Метою роботи є дослідження та практична реалізація інтелектуальних методів контролю енергоспоживання для підвищення ефективності користування електроприладами та економії електроенергії в системі домашньої автоматизації (СДА).

Задачі роботи:

- огляд існуючих інтелектуальних методів в сфері ефективного споживання електроенергії;
- розробка комплексного інтелектуального методу контролю енергоспоживання на основі прогнозування генерації електроенергії з максимальним використанням альтернативних джерел та оптимізації розкладу використання електроприладів;
- проектування та конструювання підсистеми інтелектуальної підтримки;
- інтеграція з платформою OpenHAB та тестування розробленої підсистеми;
- розробка програмної документації.

3

Рисунок Б.3 – Слайд №3

ОГЛЯД ІСНУЮЧИХ ІНТЕЛЕКТУАЛЬНИХ МЕТОДІВ

Задача	Метод вирішення
Оптимізація розкладу роботи побутових приладів	Метод рою часток
	Генетичний алгоритм
Прогнозування споживання або генерації електроенергії	ARIMA
	Нейронна мережа з довгою короткочасною пам'яттю
	Рекурентна нейронна мережа
	Нейронна мережа прямого поширення
	Згортова нейронна мережа

4

Рисунок Б.4 – Слайд №4

ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

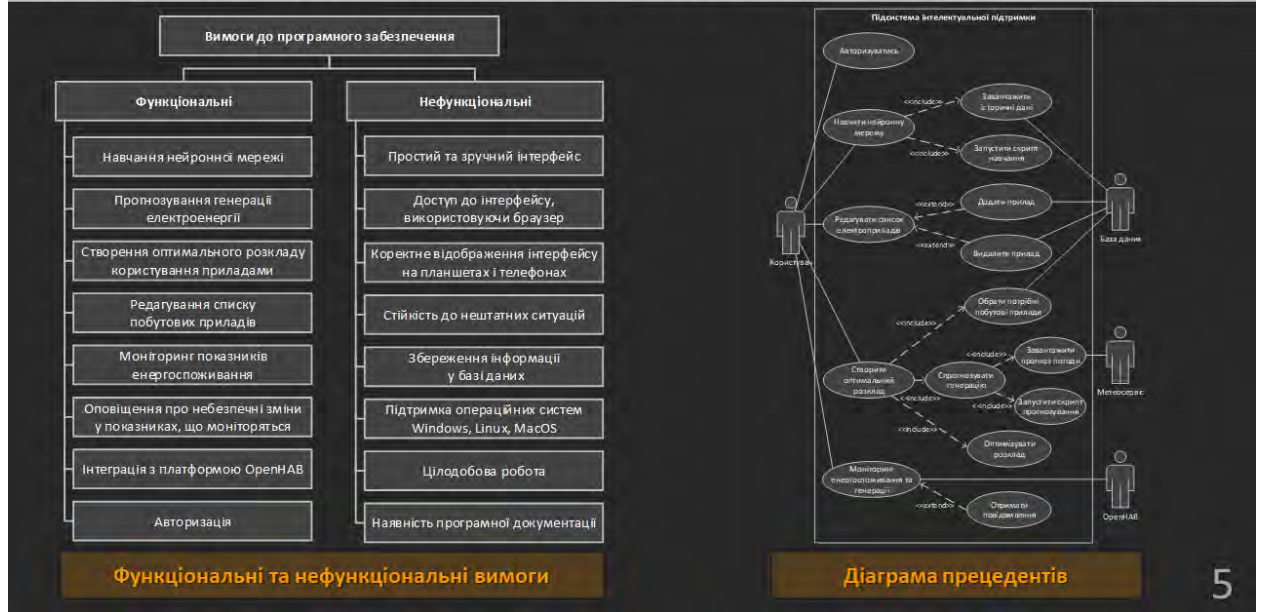


Рисунок Б.5 – Слайд №5

КОМПЛЕКСНИЙ ІНТЕЛЕКТУАЛЬНИЙ МЕТОД ПРОГНОЗУВАННЯ ТА ОПТИМІЗАЦІЇ

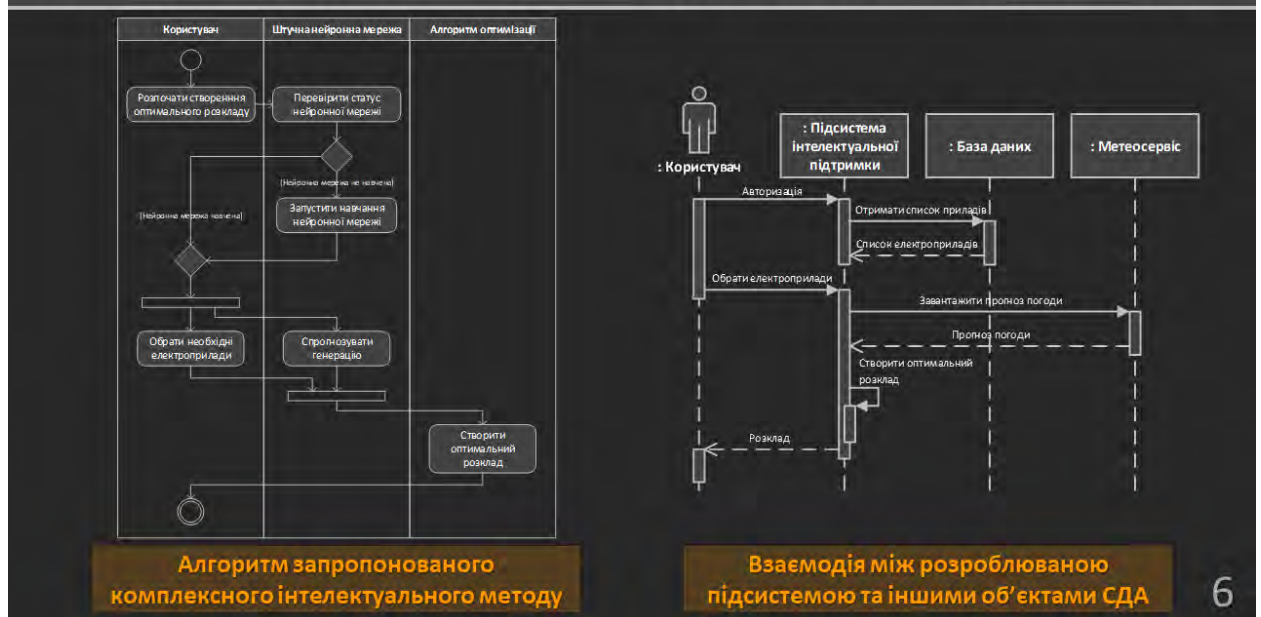


Рисунок Б.6 – Слайд №6

ВИБІР ІНСТРУМЕНТАРІЮ РОЗРОБКИ

Вибір стеку технологій для розробки програмного забезпечення:

- метеосервіс: **Visual Crossing**;
- система керування базами даних: **MongoDB**;
- вебфреймворк: **Spring (Java)**;
- мова програмування для машинного навчання: **Python**;
- бібліотека для машинного навчання: **Keras**.

Критерій порівняння	Python	R	C++	Java
Надбудова	1991	1993	1983	1996
Паралелізм	низький	низький	високий	середній
Інтерпретованість	+	+	-	-
Динамічна типізація	+	+	-	-
Висока швидкість роботи	-	-	+	+
Адаптивне керування швидкістю	-	-	-	-
Об'єктно-орієнтована парадигма	-	-	-	+
Використання в сфері машинного навчання	+	+	+	+
Велика кількість бібліотек для аналізу даних	+	+	+	+
Висока кількість бібліотек для машинного навчання	+	+	+	+
Тренувальна документація	+	+	+	+

Критерій порівняння	Python	R	C++	Java
Надбудова	1991	1993	1983	1996
Паралелізм	низький	низький	високий	середній
Інтерпретованість	+	+	-	-
Динамічна типізація	+	+	-	-
Висока швидкість роботи	-	-	+	+
Адаптивне керування швидкістю	-	-	-	-
Об'єктно-орієнтована парадигма	-	-	-	+
Використання в сфері машинного навчання	+	+	+	+
Велика кількість бібліотек для аналізу даних	+	+	+	+
Висока кількість бібліотек для машинного навчання	+	+	+	+
Тренувальна документація	+	+	+	+

Критерій порівняння	Python	R	C++	Java
Надбудова	1991	1993	1983	1996
Паралелізм	низький	низький	високий	середній
Інтерпретованість	+	+	-	-
Динамічна типізація	+	+	-	-
Висока швидкість роботи	-	-	+	+
Адаптивне керування швидкістю	-	-	-	-
Об'єктно-орієнтована парадигма	-	-	-	+
Використання в сфері машинного навчання	+	+	+	+
Велика кількість бібліотек для аналізу даних	+	+	+	+
Висока кількість бібліотек для машинного навчання	+	+	+	+
Тренувальна документація	+	+	+	+

Рисунок Б.7 – Слайд №7

КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

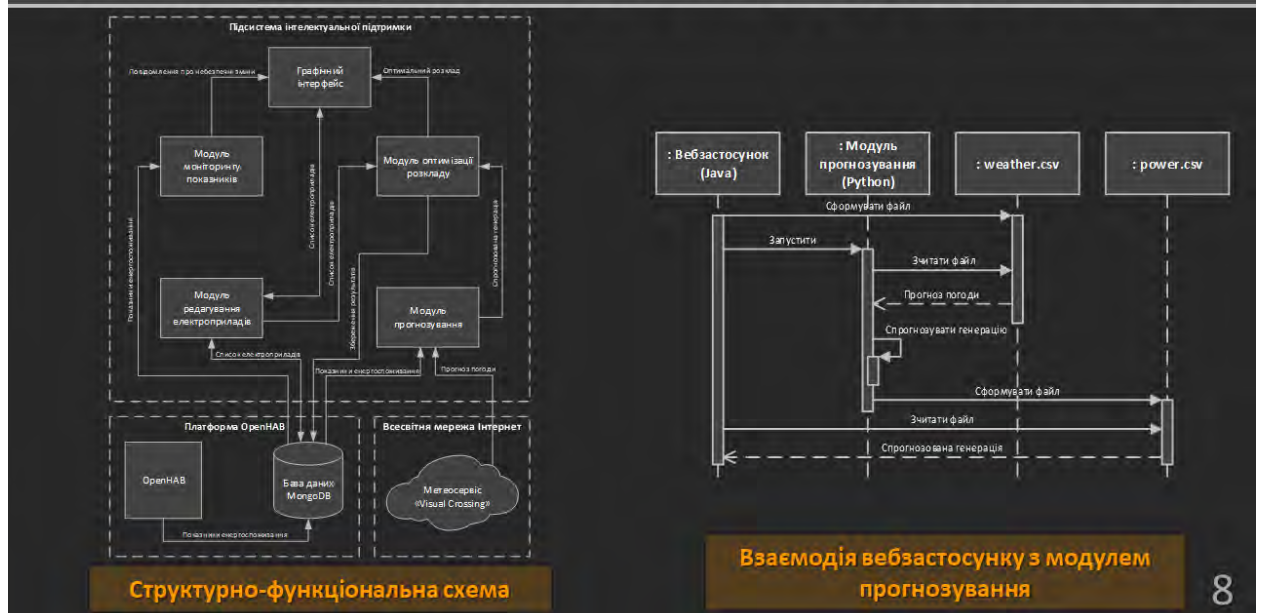
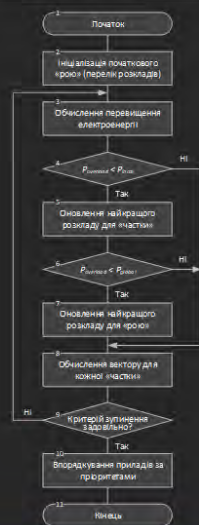
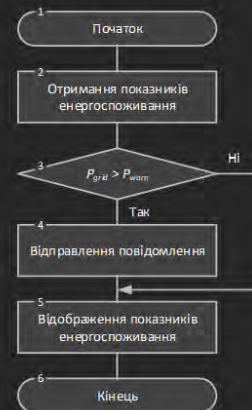


Рисунок Б.8 – Слайд №8

ОСНОВНІ РОЗРОБЛЕНІ АЛГОРИТМИ



Алгоритм оптимізації розкладу роботи електроприладів

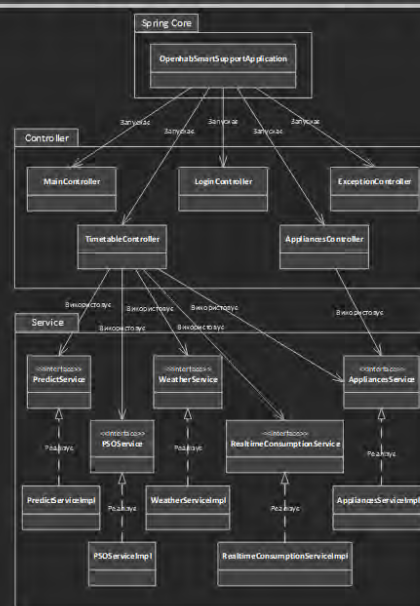


Алгоритм моніторингу показників енергоспоживання

9

Рисунок Б.9 – Слайд №9

ДІАГРАМА ОСНОВНИХ КЛАСІВ ВЕБЗАСТОСУНКУ



10

Рисунок Б.10 – Слайд №10

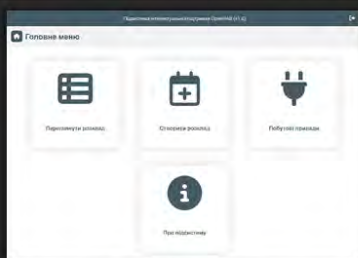
РЕАЛІЗАЦІЯ НЕЙРОМЕРЕЖЕВОЇ МОДЕЛІ

Критерій порівняння	Нейронна мережа прямого поширення			PHM	Мережа ДКЧП	
Загальна кількість шарів	4	4	4	4	4	4
Кількість параметрів	4	4	4	5	5	2
Оптимізатор	ADAM	RMSprop	AdaGrad	ADAM	ADAM	ADAM
Помилка на навчальному наборі (RMSE)	446,9633	425,1278	812,8156	183,9365	499,6775	1066,8383
Помилка на тестовому наборі (RMSE)	466,1486	492,0719	831,0850	672,0999	635,0425	1066,8383
Розмір навчальної вибірки	1728	1728	1728	1728	1728	2112
Розмір тренувальної вибірки	432	432	432	432	432	48
Розмір файлу навченої моделі	153,9 кБ	153,9 кБ	153,9 кБ	773,52 кБ	344,8 кБ	166,79 кБ
Час навчання	21 с	22 с	21 с	1 хв 20 с	4 хв 36 с	18 хв 22 с
Кількість батчів	24	24	24	24	24	24
Кількість епох	100	100	100	100	100	100

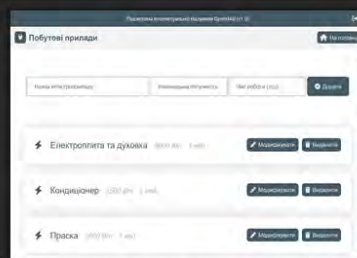
11

Рисунок Б.11 – Слайд №11

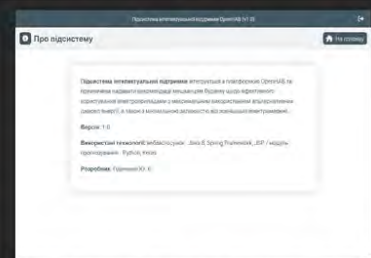
СКРІНШОТИ ПІДСИТЕМИ ІНТЕЛЕКТУАЛЬНОЇ ПІДТРИМКИ



Головне меню



Редагування побутових приладів



Інформація про підсистему



Створення оптимального розкладу



Перегляд створеного розкладу

12

Рисунок Б.12 – Слайд №12

ВИСНОВКИ

- ✓ **В результаті** роботи було проаналізовано предметну область та досліджено існуючі інтелектуальні методи в сфері ефективного споживання електроенергії.
- ✓ **Наукова новизна** роботи полягає в розробці інтелектуального методу контролю енергоспоживання, що відрізняється від існуючих комплексним підходом на основі прогнозування генерації електроенергії з максимальним використанням альтернативних джерел та оптимізації розкладу використання електроприладів на основі системи пріоритетів.
- ✓ **Практичне значення** роботи полягає в тому, що інтеграція розробленої підсистеми інтелектуальної підтримки в СДА дозволить контролювати процеси генерації та споживання електроенергії, а також ефективно користуватися побутовими приладами та заощаджувати ресурси.

13

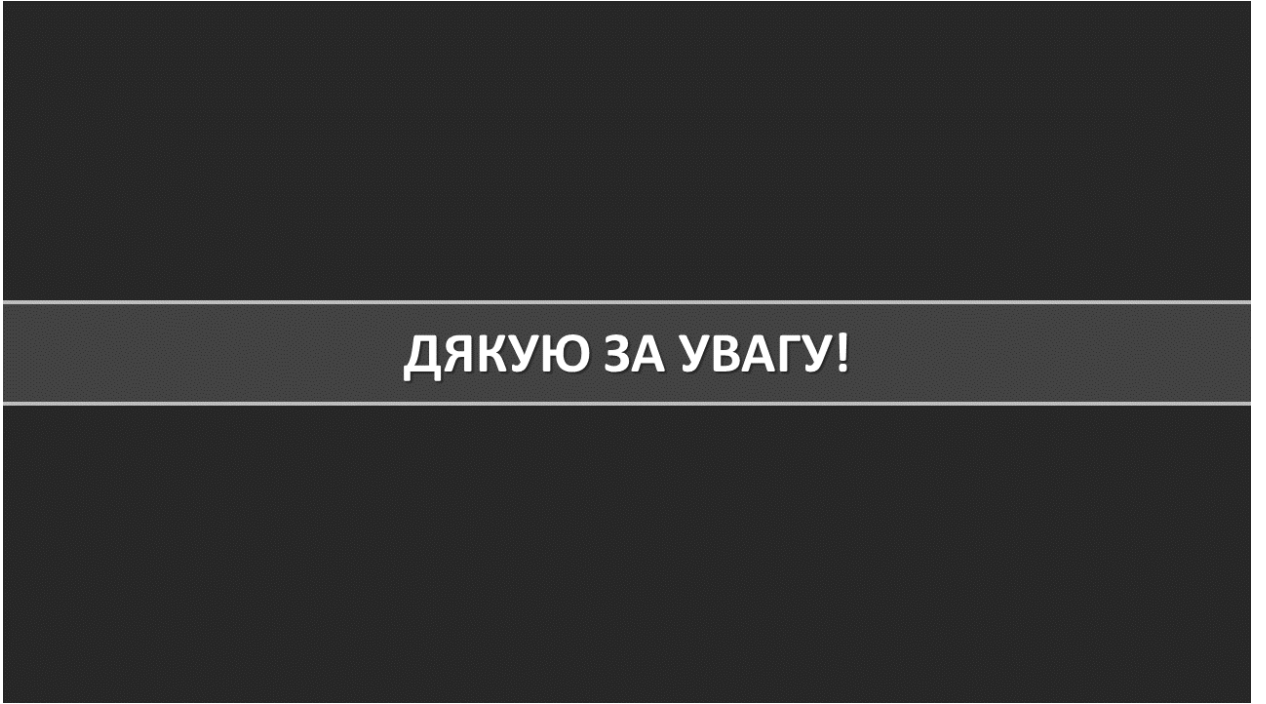
Рисунок Б.13 – Слайд №13

ПУБЛІКАЦІЇ

1. Data Mining to Achieve Quality of Life for Home Automation Users / [M. Zadoian, Y. Horichenko, A. Tulenkov et al.] // Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications : 11th IEEE International Conference, Cracow, 22-25 September 2021 : proceedings. – Cracow: IEEE, 2021.
2. Горіченко, Ю. Є. Дослідження та програмна реалізація інтелектуальних методів контролю енергоспоживання / Ю. Є. Горіченко, А. В. Пархоменко, О. А. Поздняков // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій : XI Міжнародна науково-практична конференція, Запоріжжя, 12–14 грудня 2022 : тези доповідей. – Запоріжжя : НУ «Запорізька політехніка», 2022.

14

Рисунок Б.14 – Слайд №14



ДЯКУЮ ЗА УВАГУ!

Рисунок Б.15 – Слайд №15