

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ДАНИХ
СИСТЕМИ ДОМАШНЬОЇ АВТОМАТИЗАЦІЇ
SOFTWARE FOR DATA ANALYSIS
OF A HOME AUTOMATION SYSTEM

Виконав: студент 4 курсу, групи КНТ-227

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

Горіченко Ю. Є.

(прізвище та ініціали)

Керівник Пархоменко А. В.

(прізвище та ініціали)

Рецензент Морщавка С. В.

(прізвище та ініціали)

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

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н., проф.
С. О. Субботін
« » 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Горіченка Юлія Євгенівна

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи): Програмне забезпечення для аналізу даних системи домашньої автоматизації. Software for Data Analysis of a Home Automation System

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 «30» березня 2021 року № 103

2. Строк подання студентом проєкту (роботи): 17 травня 2021 року

3. Вихідні дані до проєкту (роботи): згідно технічного завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): 1 Дослідження особливостей системи домашньої автоматизації. 2 Огляд особливостей інтелектуального аналізу даних. 3 Практична реалізація інтелектуального аналізу даних для системи домашньої автоматизації. 4 Програмна реалізація. 5 Керівництво програміста. 6 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	Дейнега Л. Ю., ст. викладач		

7. Дата видачі завдання «28» лютого 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Огляд особливостей інтелектуального аналізу даних	3 тиждень	Розділ 2
4	Проведення обчислювальних експериментів з програмним інструментом Orange	4 тиждень	Розділ 3
5	Розробка та тестування програми	5-6 тижні	Розділ 4
6	Розробка програмної документації	7 тиждень	Розділ 5, 6
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	ПЗ, Додатки
8	Захист роботи	9 тиждень	

Студент

Керівник проєкту (роботи)

(підпис)

(підпис)

Горіченко Ю. Є.

(прізвище та ініціали)

Пархоменко А. В.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 163 с., 82 рис., 24 табл., 3 дод., 70 джерел.

СИСТЕМА ДОМАШНЬОЇ АВТОМАТИЗАЦІЇ, ПЛАТФОРМА OPENHAB, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, АСОЦІАТИВНІ ПРАВИЛА, ЕНЕРГОСПОЖИВАННЯ, ГРАФІЧНИЙ ФРЕЙМВОРК.

Об'єкт дослідження – моніторинг даних системи домашньої автоматизації.

Предмет дослідження – методи та засоби аналізу даних щодо споживання електроенергії в системі домашньої автоматизації.

Мета роботи – розробка програмного застосунку для аналізу даних щодо споживання електроенергії для підвищення безпеки та ефективності системи домашньої автоматизації.

Матеріали, методи та технічні засоби: платформа OpenHAB, Orange, мова програмування Java, графічний фреймворк JavaFX, персональний комп'ютер, що має процесор з частотою 1 ГГц під управлінням операційної системи Microsoft Windows 7+ / RHEL 5.5+ / Ubuntu 12.04+ / macOS 10.9+.

Результати. Створено програмний застосунок, який виконує цілодобовий моніторинг та аналіз показників споживання електроенергії системи домашньої автоматизації, а також повідомляє власника про небезпечні зміни показників та надає рекомендації щодо безпечного користування електроприладами.

Висновки. Розроблено програмний застосунок для системи домашньої автоматизації, який дозволить забезпечити надійний захист електромережі від перегріву, що може спричинити пожежу, а також дозволить підвищити ефективність використання електроприладів.

Галузь використання – системи домашньої автоматизації на платформі OpenHAB.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 163 pages, 82 figures, 24 tables, 3 appendixes, 70 sources.

HOME AUTOMATION SYSTEM, OPENHAB PLATFORM, DATA MINING, ASSOCIATION RULES, ELECTRIC ENERGY CONSUMPTION, GRAPHICS FRAMEWORK.

Object of study is a home automation system data monitoring.

Subject of study are methods and tools for analyzing energy consumption data in a home automation system.

The goal of the work is to develop a software application for analyzing energy consumption data to improve safety and efficiency of a home automation system.

Materials, methods, and tools: OpenHAB platform, Orange, Java programming language, JavaFX graphics framework, an IBM-compatible personal computer with a 1 GHz processor running the Microsoft Windows 7+ / RHEL 5.5+ / Ubuntu 12.04+ / macOS 10.9+ operating system.

Results. A software application for round-the-clock monitoring and analyzing energy consumption data in a home automation system was developed. It also notifies the owner about dangerous changes and provides recommendations for the safe use of electrical appliances.

Conclusions. A software application for a home automation system was developed. It will provide a reliable protection from overheating of wires and cables that can lead to a fire and will also increase the efficiency of using electrical appliances.

The field of application is the home automation systems based on the OpenHAB platform.

ЗМІСТ

	С.
ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК	9
ВСТУП	10
1 ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ СИСТЕМИ ДОМАШНЬОЇ АВТОМАТИЗАЦІЇ.....	11
1.1 Система домашньої автоматизації	11
1.2 Досвід використання інтелектуального аналізу даних для систем домашньої автоматизації	13
1.3 Постановка задач дипломної роботи	19
2 ОГЛЯД ОСОБЛИВОСТЕЙ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ...	21
2.1 Інтелектуальний аналіз даних.....	21
2.2 Задачі інтелектуального аналізу даних.....	23
2.3 Методи інтелектуального аналізу даних	25
2.4 Етапи інтелектуального аналізу даних	29
2.5 Алгоритми пошуку асоціативних правил.....	32
2.5.1 Алгоритм Apriori	36
2.5.2 Алгоритм FP-Growth.....	38
2.5.3 Алгоритм Eclat.....	40
2.6 Огляд існуючих програмних засобів інтелектуального аналізу даних	42
2.7 Огляд функціональних особливостей Orange	46
2.8 Висновки до розділу	48
3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ ДЛЯ СИСТЕМИ ДОМАШНЬОЇ АВТОМАТИЗАЦІЇ	50
3.1 Опис архітектури системи домашньої автоматизації	50
3.2 Аналіз вхідних даних.....	56
3.3 Використання Orange для пошуку асоціативних правил та застосування алгоритмів машинного навчання	58
3.3.1 Обчислювальний експеримент 1	58

3.3.2 Обчислювальний експеримент 2	62
3.3.3 Обчислювальний експеримент 3	65
3.3.4 Обчислювальний експеримент 4	67
3.4 Висновки до розділу	70
4 ПРОГРАМНА РЕАЛІЗАЦІЯ.....	71
4.1 Проєктування програмного забезпечення	71
4.1.1 Аналіз вимог	71
4.1.2 Архітектура системи.....	73
4.1.3 Структурна схема програмного застосунку	75
4.1.4 Вибір інструментарію розробки	78
4.2 Конструювання програмного забезпечення.....	86
4.2.1 Основні рішення щодо представлення даних програми	86
4.2.2 Основні розроблені алгоритми	101
4.2.3 Основні рішення щодо модульної побудови програми	105
4.2.4 Особливості реалізації бази даних	109
4.2.5 Особливості реалізації графічного інтерфейсу.....	110
4.3 Висновки до розділу	113
5 КЕРІВНИЦТВО ПРОГРАМІСТА	115
5.1 Призначення та умови застосування програми	115
5.2 Характеристики програми.....	115
5.3 Звертання до програми	116
5.4 Вхідні та вихідні дані.....	120
5.5 Повідомлення	121
6 КЕРІВНИЦТВО КОРИСТУВАЧА	124
6.1 Призначення програми	124
6.2 Умови виконання програми	124
6.3 Виконання програми.....	125
6.3.1 Вкладка «Головна»	125
6.3.2 Вкладка «Прилади».....	127
6.3.3 Вкладка «Параметри».....	129

6.3.4 Вкладка «Про програму».....	130
6.4 Повідомлення	130
ВИСНОВКИ.....	133
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	135
ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ	142
ДОДАТОК Б ТЕКСТ ПРОГРАМИ.....	147
ДОДАТОК В СЛАЙДИ ПРЕЗЕНТАЦІЇ	154

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
BLE	– Bluetooth Low Energy;
CLR	– Common Language Runtime;
CSS	– Cascading Style Sheets;
HDMI	– High Definition Multimedia Interface;
HTML	– HyperText Markup Language;
JRE	– Java Runtime Environment;
JSON	– JavaScript Object Notation;
JVM	– Java Virtual Machine;
MQTT	– Message Queuing Telemetry Transport;
MVC	– Model-View-Controller;
OLAP	– Online Analytical Processing;
OpenHAB	– Open Home Automation Bus;
REST	– Representational State Transfer;
SQL	– Structured Query Language;
URI	– Uniform Resource Identifier;
Wi-Fi	– Wireless Fidelity;
XML	– eXtensible Markup Language;
ОС	– операційна система;
СДА	– система домашньої автоматизації.

ВСТУП

У зв'язку з розвитком технологій Інтернету речей, системи домашньої автоматизації (СДА) сьогодні набирають все більшої популярності. Головною метою таких систем є підвищення рівня комфорту та безпеки мешканців, а також зниження споживання ресурсів. Такі системи складаються з датчиків та пристроїв, якими мешканець будинку може керувати будь-яким зручним для нього способом. Всі ці пристрої виробляють велику кількість даних, які зазвичай не використовуються, хоча можуть містити дуже цінну інформацію. Отже, методи інтелектуального аналізу даних та машинного навчання можуть відігравати важливу роль, оскільки накопичені дані з СДА можна перетворювати на корисну інформацію для користувача у реальному часі. Таким чином, це дозволяє спрогнозувати певні ситуації, а також швидко відреагувати на поточні події, надати корисні поради, зробити СДА більш безпечною та підвищити рівень автоматизації процесів управління.

Існують різні дослідження, в яких автори намагалися накопичені дані з СДА за допомогою інтелектуального аналізу перетворити на корисні знання. Але в цих роботах не пропонується рішення для зниження споживання електроенергії та безпечного користування електроприладами, тому використані методи та реалізовані підходи потребують подальшого розвитку. Тому, тема дипломної роботи є актуальною.

Метою роботи є розробка програмного застосунку для аналізу даних щодо споживання електроенергії для підвищення безпеки та ефективності системи домашньої автоматизації.

1 ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ СИСТЕМИ ДОМАШНЬОЇ АВТОМАТИЗАЦІЇ

1.1 Система домашньої автоматизації

СДА сьогодні набирає все більшої популярності. Головною метою таких систем є підвищення комфорту людей, зниження енергоспоживання, підвищення безпеки, віддалений доступ та управління обладнанням [1]. Для цього потрібно з'єднати різні пристрої один з одним, що дозволить автоматизувати процеси всередині й зовні будинку. Великою перевагою такої системи є те, що користувач може керувати будинком з будь-якої точки планети, наприклад, коли він перебуває у відпустці та хоче перевірити, чи все світло вимкнено, а двері зачинені. Для управління таким будинком можна використовувати різні засоби:

- планшет;
- смартфон;
- персональний комп'ютер або ноутбук;
- камера (за допомогою жестів);
- голосове управління [2].

СДА також відома як автоматизований або інтелектуальний будинок та включає в себе загальні пристрої, які можуть керувати функціями будинку. Ця технологія не тільки використовується, наприклад, для управління опаленням, освітленням або вмиканням та вимиканням пристроїв, але може управляти пристроями за заздалегідь створеними сценаріями або на вимогу користувача. Багато з таких пристроїв використовуються в СДА для того, щоб створити середовище, в якому багато функцій будуть автоматизовані, а пристрої зможуть взаємодіяти один з одним [3].

Всі вони виробляють велику кількість даних, які майже не використовуються, хоча можуть містити дуже цінну інформацію (рис. 1.1). Методи інтелектуального аналізу даних та машинного навчання можуть відігравати важливу роль, тому що накопичені дані з СДА можна

перетворювати на корисну інформацію для користувача у реальному часі. Таким чином, це дозволяє спрогнозувати та швидко відреагувати на виникаючі ситуації, дати корисні поради та зробити систему більш автоматизованою. Наприклад, отримані знання можна застосувати при розпізнаванні діяльності для виявлення проблем зі здоров'ям мешканців, проаналізувати споживання електроенергії та спланувати енергозбереження, а також виконати прогнозування графіка технічного обслуговування обладнання, щоб уникнути коштовного ремонту та забезпечити ефективну роботу з точки зору споживання енергії [4].

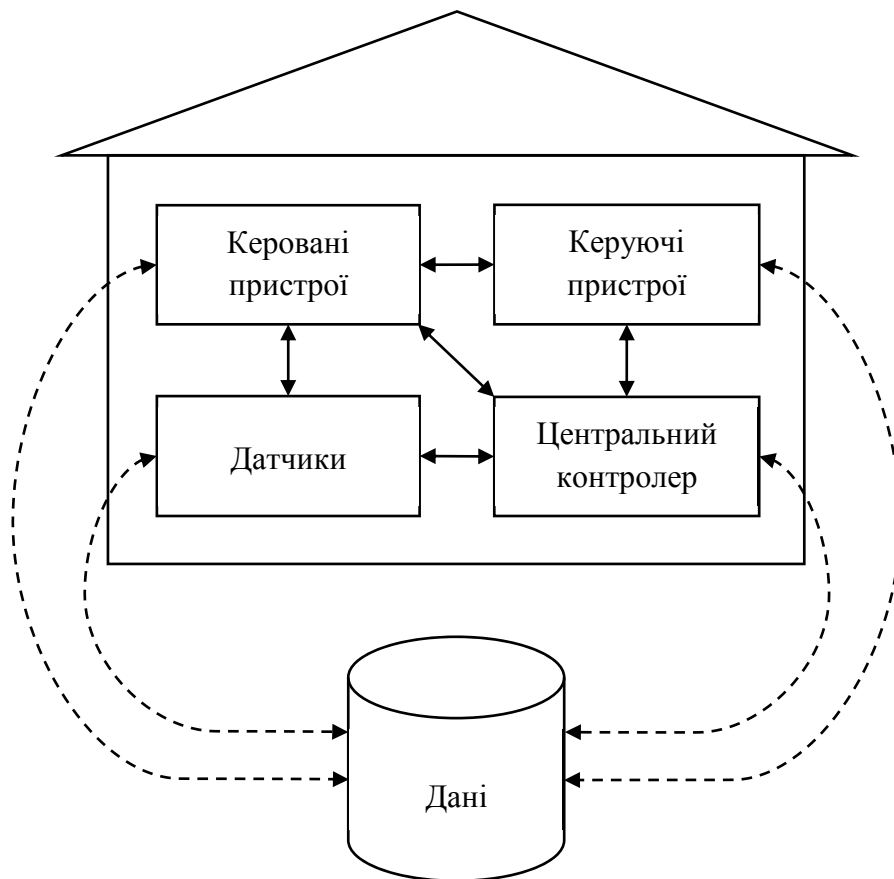


Рисунок 1.1 – Джерела та потоки даних у СДА

1.2 Досвід використання інтелектуального аналізу даних для систем домашньої автоматизації

Використання інтелектуального аналізу для отриманих даних з СДА було розглянуто в статті [5]. Автори [5] пропонують модель, яка використовує накопичені дані для вивчення та виявлення шаблонів людської діяльності, щоб оцінити стан здоров'я мешканців. Для цього було використано наступні методи: пошук асоціативних правил, кластерний аналіз та мережа Байєса для вимірювання і аналізу змін енергоспоживання, викликаних поведінкою мешканців. Оскільки звички людей в основному визначаються повсякденними діями, то виявлення цих дій дозволяє розпізнати аномальні дії, які можуть вказувати на погіршення здоров'я. Таким чином, ці знання можуть бути використані медичними програмами для виявлення потенційних проблем зі здоров'ям та швидкого вжиття заходів, таких як відправка попереджень пацієнтам або постачальникам медичних послуг. Також це забезпечить цілодобову службу моніторингу, яка автоматично ідентифікує нормальну і ненормальну поведінку, що буде актуальним як для звичайних людей, так і для літніх людей або пацієнтів з когнітивними порушеннями. У результаті чого була розроблена система, яка вміє аналізувати накопичені дані (рис. 1.2). Для дослідження використовувався набір даних по споживанню електроенергії на рівні побутових приладів для п'яти будинків у Великобританії протягом трьох років. Перевагою є те, що ця модель працює з високою точністю, а недоліком є те, що вона не працює у реальному часі та потрібно провести багато досліджень, щоб ефективно навчити систему і підвищити точність виявлення людської діяльності [5].

Задача зі зниження споживання електроенергії за допомогою інтелектуального аналізу була представлена у роботі [6]. Метою цієї статті було те, щоб перевірити, чи може метод пошуку асоціативних правил бути використаний для економії електроенергії.

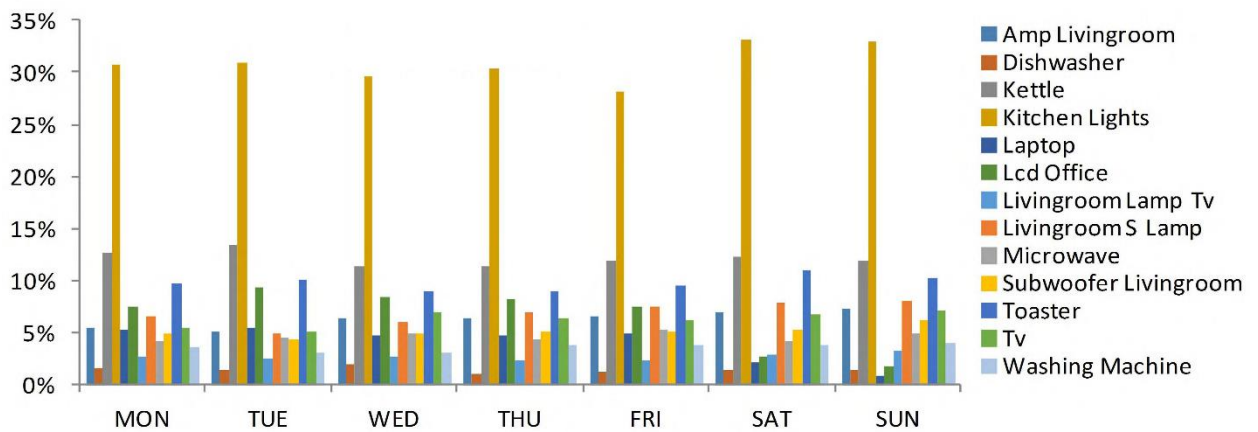


Рисунок 1.2 – Результати виявлення шаблонів людської діяльності [5]

Для цього використовували дані, які були зібрані з шести будинків впродовж декількох місяців у міській агломерації Великий Бостон, що знаходиться в Сполучених Штатах. В ході досліджень було з'ясовано, що цей метод може бути використаний для визначення того, які прилади використовуються разом (рис. 1.3). Отримані знання можна використовувати для внесення змін під час користування побутовими приладами, що може допомогти зберегти електроенергію. Наприклад, якщо буде виявлено, що зазвичай людина вмикає два або більше світла, коли вона знаходиться в кімнаті, то дуже ймовірно, що вона забуде вимкнути один з них, виходячи з кімнати. За допомогою централізованої керованої системи цю проблему можна вирішити, яка буде автоматично вимикати світло коли людина виходить з кімнати. Така система буде постійно навчатися з плином часу і адаптуватися до поведінки мешканців. Отже, це дає великий потенціал для зменшення постачання електроенергії. Перевагою такого методу є те, що він працює швидко та має високу точність. Це говорить про те, що алгоритм має високу ймовірність правильного передбачення правил. Недоліком такої системи є те, що потрібно провести ще безліч досліджень та створити рекомендаційну систему, яка зможе давати ефективні та правильні поради для економії електроенергії [6].

House Association	
1	refrigerator, bathroom gfi, kitchen outlet3 → lighting 1
2	stove, kitchen outlet 2, refrigerator → washer, dryer
3	dishwasher, furnace, microwave, smoke alarm, kitchen outlet1 → lighting 5
4	unknown outlet, stove, miscellaneous, bathroom gfi1 → bathroom gfi 2
5	microwave, unknown outlet1, subpanel 1, lighting1, bathroom gfi, refrigerator, electronics, lighting3, kitchen outlet → furnace
6	kitchen outlet1, refrigerator, kitchen outlet2 → stove

Рисунок 1.3 – Отримані асоціативні правила для кожного будинку [6]

Аналіз енергоспоживання в інтелектуальних будівлях було розглянуто в статті [7]. Для дослідження використовувався набір даних для одного будинку протягом одного року, який включає інформацію про навколишнє середовище та загальне споживання електроенергії інтервалом у 8 годин. Автори [7] проаналізували декілька методів машинного навчання: нейронні мережі, метод опорних векторів, мережа Байєса, гауссівський процес та алгоритм «випадковий ліс». Після того, як були отримані результати експерименту (рис. 1.4), можна побачити, що найкраще працював алгоритм «випадковий ліс» та мережа Байєса, бо вони мають найменший відсоток середнього квадратичного відхилення, а саме 12.09% та 12.86% відповідно. Таким чином, використання цієї моделі дозволяє передбачити споживання електроенергії, використовуючи дані про навколишнє середовище та загальне споживання енергії. Єдиним недоліком такої системи є те, що вона не працює в реальному житті та для цього на основі цієї моделі потрібно розробити стратегію управління для економії електроенергії в будівлях [7].

У роботі [8] було проведено дослідження, що дозволило дізнатися, які конфіденційні дані можна видобути, використовуючи інтелектуальний аналіз даних.

Moment	Technique	Best Parameter	RMSE (KWh)	CV RMSE (%)	R^2
1	Gauss	$\sigma = 0.1$	1.1	13.43	0.57
1	MLP	size = 34	1.1	13.46	0.55
1	SVM	cost = 4	1.09	13.26	0.58
1	BRNN	neurons = 3	1.1	13.47	0.55
1	RF	mtry = 4	1	12.18	0.65
2	Gauss	$\sigma = 0.1$	7.76	14.1	0.67
2	MLP	size = 37	1.56	15	0.68
2	SVM	cost = 1	4.26	13.4	0.71
2	BRNN	neurons = 2	7.08	12.86	0.75
2	RF	mtry = 2	7.48	13.59	0.72
3	Gauss	$\sigma = 0.1$	4.5	14.07	0.67
3	MLP	size = 37	4.81	15.03	0.69
3	SVM	cost = 1	4.20	13.14	0.73
3	BRNN	neurons = 5	4.31	13.45	0.73
3	RF	mtry = 2	3.87	12.09	0.76

Рисунок 1.4 – Порівняння результатів алгоритмів машинного навчання [7]

Для цього використовувався набір даних для одного будинку, що має обмежену інформацію, а саме який датчик було активовано в певний час. Алгоритм пошуку асоціативних правил намагається ідентифікувати дії мешканців, використовуючи набір даних датчиків, які розташовані по всьому будинку. Логіка цього методу заснована на двох припущеннях, а саме, що люди зазвичай залишаються в своїх ліжках між 2 та 6 годинами, а також вони зазвичай перебувають на кухні між 18 та 19 годинами. Це дозволяє вивести місця, де розташовані датчики. Експеримент показав, що цей метод здатен вивести глобальну топологію датчиків, що відповідає реальній для розумного будинку (рис. 1.5). Таким чином, це дозволяє отримати багато конфіденційної інформації, наприклад, щоденний розклад життя мешканців, як часто вони залишаються в своїх ліжках, коли вони виходять та повертаються додому тощо. Це може нести серйозну загрозу безпеці мешканців та будинку. Перевагою такого методу є те, що він працює з великою точністю та може використовувати дуже обмежений набір інформації [8].

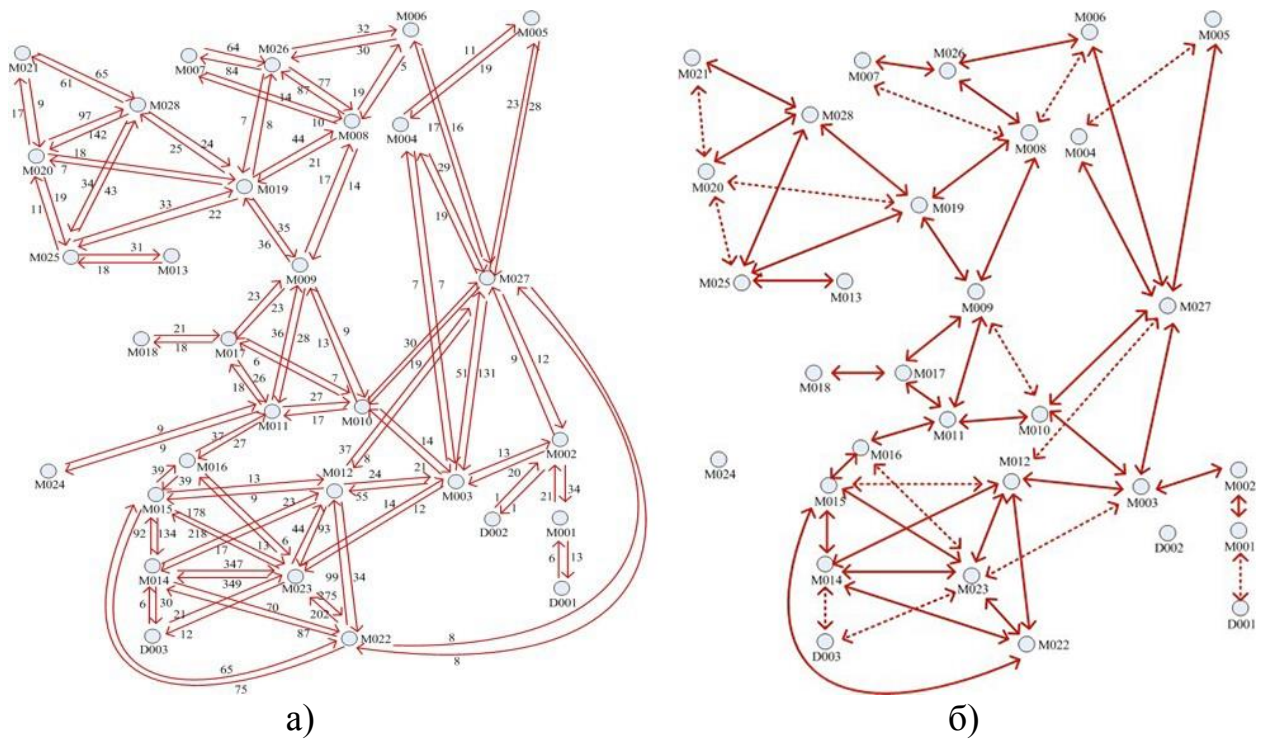


Рисунок 1.5 – Отримана (а) та реальна (б) топологія датчиків будинку [8]

У статті [9] було досліджено використання датчиків руху у СДА, що дозволило виявити кількість мешканців будинку. Використані дані були отримані з тридцяти будинків мешканців похилого віку, які приймали участь у цьому експерименті протягом шести тижнів. Також ці люди вели журнал присутності гостей в своєму будинку, який було використано для оцінки точності побудованої моделі. Автори [9] запропонували використовувати метод опорних векторів для оцінки діяльності та поведінки людей. Здатність визначати, коли відвідувачі знаходяться в будинку є важливою для оцінки здоров'я літніх людей. Соціально ізольовані люди зазвичай мають більш високий кров'яний тиск та піддаються ризику розвитку когнітивних порушень. З накопичених даних, які включають час перебування, кількість спрацьовувань датчиків та кількість переходів між кімнатами дозволило визначити присутність відвідувачів в будинку (рис. 1.6). Недоліком такої системи є те, що вона потребує покращення, тому що точність отриманих результатів варіюється від 67% до 90%, що не є задовільним результатом [9].

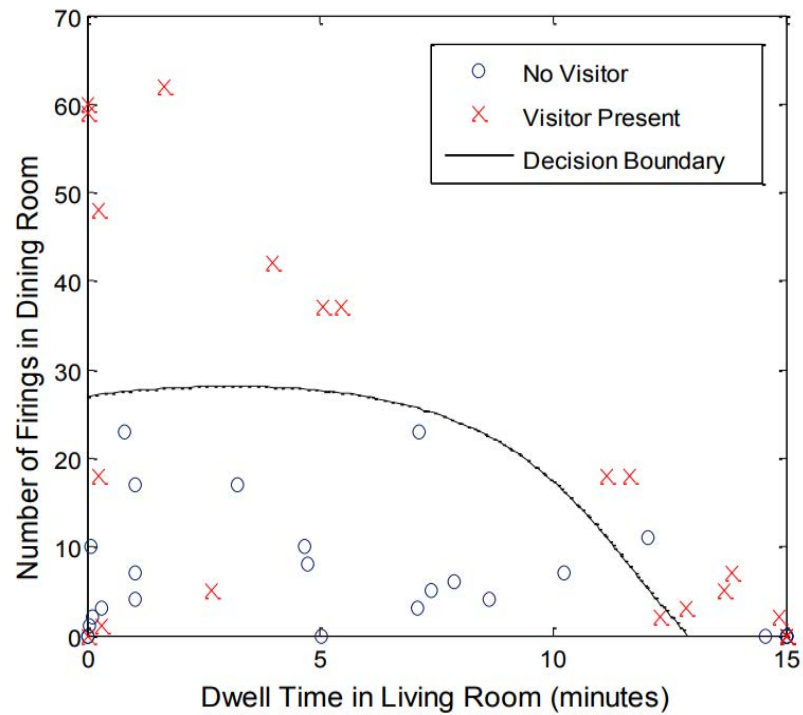


Рисунок 1.6 – Результати визначення присутності гостей в будинку [9]

У роботі [10] було проведено дослідження, щоб виявити, який тип штучної нейронної мережі краще використовувати в СДА. Однак кожна нейронна мережа має певну перевагу та з різною точністю дозволяє обробляти різні типи даних. Це робить неможливим створення однієї універсальної нейронної мережі для вирішення різних типів задач. Таким чином, було розроблено програмне забезпечення, яке дозволяє проводити навчання, тестування алгоритму та обирати найкращу нейронну мережу за результатами точності. Для цього були використані дані, які були зібрані з одного будинку протягом року. Отриманий набір даних містить інформацію про температуру в двох кімнатах, статус освітлення, статус сигналізації та інформацію про присутність людей в будинку. Перевагою такого дослідження є те, що розроблений застосунок допоможе зекономити багато часу та підібрати найкращий алгоритм для вирішення поставленої задачі, а також знизити кількість помилок у результатах [10].

1.3 Постановка задач дипломної роботи

Як показали проведені дослідження, СДА накопичує багато даних з різних датчиків та пристроїв, які практично не використовуються. Накопичені дані можна перетворювати на корисну інформацію для користувача у реальному часі, наприклад, для прогнозування та виконання швидких дій щодо виникаючих ситуацій, відправлення корисних рекомендацій та повідомлень. Все це допоможе зробити систему більш автоматизованою та безпечною у використанні.

У багатьох роботах автори показали досвід використання інтелектуального аналізу даних для СДА. Дослідження робіт [5]-[10] показали, які методи та алгоритми машинного навчання краще використовувати для вирішення певних задач. На рисунку 1.7 зображені використовувані методи інтелектуального аналізу для СДА.

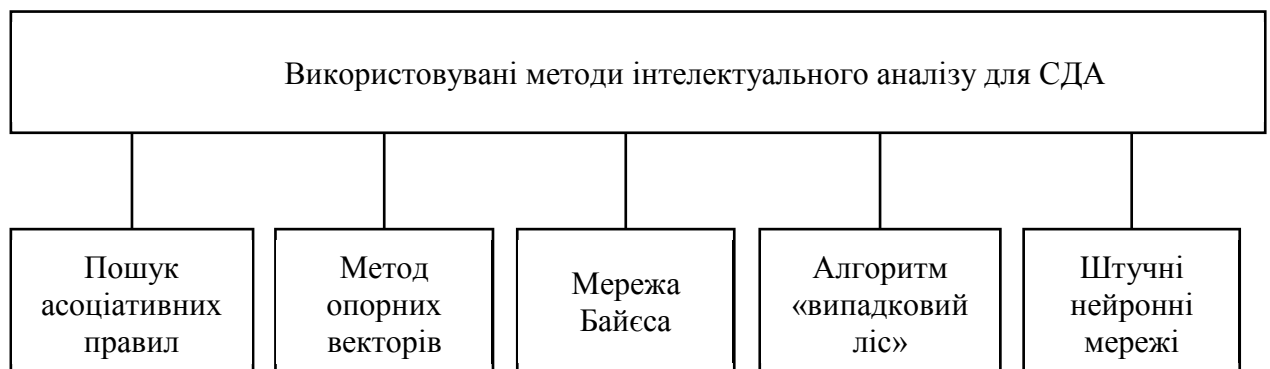


Рисунок 1.7 – Використовувані методи інтелектуального аналізу для СДА

Метод пошуку асоціативних правил найчастіше використовувався для аналізу великих обсягів даних та встановлення залежності між діями, які виконуються в один момент часу. Отримані результати дають можливість побачити звички мешканців, їх розпорядок дня, використання побутових приладів та багато іншої цікавої інформації. Таким чином, цей підхід можна використати для кращого розуміння поведінки людей або для прогнозування ситуацій у майбутньому.

Для аналізу діяльності та поведінки людей було запропоновано метод опорних векторів, який використовується для вирішення задач класифікації. За допомогою цього підходу вдалося виявити коли мешканець знаходиться один в будинку, а коли до нього прийшли гості. У результаті експерименту було виявлено, що цей метод потребує покращення, тому що його точність варіюється від 67% до 90%.

Для передбачення ситуацій у майбутньому, наприклад, використання електроенергії, було зроблено багато досліджень. В ході проведення експериментів було виявлено, що найкращі результати дає використання методів: мережа Байєса, алгоритм «випадковий ліс» та штучні нейронні мережі. Такий підхід дозволяє з високою точністю робити припущення.

Всі описані вище методи не дозволяють вирішити конкретних задач. Вони мають лише великий потенціал, бо можуть з «сирих» даних витягати багато корисної інформації. А для того, щоб мати змогу використовувати отримані знання, потрібно провести ще безліч досліджень та розробити стратегію для вирішення поставленої мети.

На жаль, в цих роботах не пропонується підхід для зниження споживання електроенергії та безпечного користування електроприладами, тому тема дипломної роботи є актуальною.

Метою роботи є розробка програмного застосунку для аналізу даних щодо споживання електроенергії для підвищення безпеки та ефективності СДА.

Для досягнення мети потрібно вирішити наступні задачі:

- виконати огляд особливостей методів, алгоритмів та програмних засобів інтелектуального аналізу даних;
- застосувати програмний інструмент з функціями інтелектуального аналізу для даних, що були накопичені з СДА;
- удосконалити архітектуру СДА;
- розробити програмний застосунок для аналізу даних з СДА;
- розробити програмну документацію.

2 ОГЛЯД ОСОБЛИВОСТЕЙ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

2.1 Інтелектуальний аналіз даних

У зв'язку з розвитком інформаційних технологій за останні десятиліття людство змогло накопичити дуже великі об'єми інформації. Це дані з різних областей, наприклад, промислові підприємства, органи державного управління, медична та наукова галузі, великі та малі фірми, соціальні мережі, інтернет-магазини. Отримані дані настільки значного об'єму, що людині не під силу самотійно їх проаналізувати. Зрозуміло, що у накопиченій інформації міститься прихований потенціал знань, володіючи яким можна значно підвищити ефективність та прибуток будь-якої діяльності. На сьогоднішній день діяльність кожного підприємства фіксується, записується та зберігається у цифровій формі. Однак, такі дані без необхідного аналізу та обробки утворюють нікому не потрібну купу сміття. Це призводить до того, що добування прихованих знань є дуже актуальною задачею, що дозволяє знайти серед терабайтів накопичених даних дуже корисну інформацію [11].

Для обробки накопичених даних мають виконуватися наступні вимоги:

- оброблювані дані можуть бути необмеженого обсягу;
- дані не є однорідними;
- отримані результати повинні бути зрозумілими та конкретними;
- інструменти для аналізу та обробки даних повинні бути прості у використанні [11].

Звідси виникає питання, як під час процесу постійного оновлення та додавання нової інформації можна виявити закономірності статистичних даних. На сьогодні метод асоціативних правил є одним із популярних для пошуку взаємозв'язків серед великого об'єму даних, який відноситься до такої області, як інтелектуальний аналіз даних або Data Mining.

Data Mining являє собою процес виявлення закономірностей та взаємозв'язків у великих наборах даних, які можна використовувати для прийняття більш обґрунтованих рішень та для прогнозування майбутніх тенденцій [12]. Рівні знань, які можна отримати з даних та застосовувані до них аналітичні інструменти наведено у табл. 2.1.

Таблиця 2.1 – Рівні знань, що отримуються з даних

Рівень знань	Аналітичні інструменти
Поверхневий	Мова простих запитів
Неглибокий	Оперативна аналітична обробка (OLAP)
Схований	Інтелектуальний аналіз даних (Data Mining)

Засоби формування простих запитів допомагають отримати потрібну інформацію з бази даних. Цей спосіб є корисним, коли потрібно дізнатися які дії виконувалися у певний час. Засоби Online Analytical Processing (OLAP) забезпечують отримання підсумків, порівнянь та прогнозів шляхом екстраполяції. Цей спосіб використовується при проведенні деталізації для отримання більш докладних даних. Data Mining йде далі та аналізує минуле для того, щоб передбачити майбутнє [13].

Головна особливість інтелектуального аналізу даних – це отримання нетривіальних, корисних, доступних для розуміння людиною та раніше невідомих знань.

По-перше, знання повинні бути нетривіальними. Тобто, результати повинні відображати неочевидні та несподівані закономірності в даних, які включають так звані сховані знання. По-друге, вони повинні бути корисними та можуть бути застосовані на нових даних з досить високим рівнем достовірності. Корисність полягає в тому, щоб вони могли принести певну вигоду при використанні. По-третє, знання мають бути представлені в зрозумілому для людини вигляді. Отримані закономірності повинні бути

логічними, інакше існує ймовірність, що вони є випадковими. Також знання мають бути новими та раніше невідомими [14].

2.2 Задачі інтелектуального аналізу даних

В основі технології інтелектуального аналізу даних лежить поняття шаблонів, яке описує собою закономірності. Таким чином, під час виявлення цих закономірностей вирішуються задачі Data Mining. Різним типам закономірностей, які виражаються у формі, що є зрозумілою для людини, відповідають певні завдання [15].

Виділяють 5 основних типів закономірностей, які ще називають задачами Data Mining: класифікація, кластеризація, регресія, асоціація та послідовність [15].

На рисунку 2.1 зображено дерево основних задач, які використовуються в інтелектуальному аналізі даних.

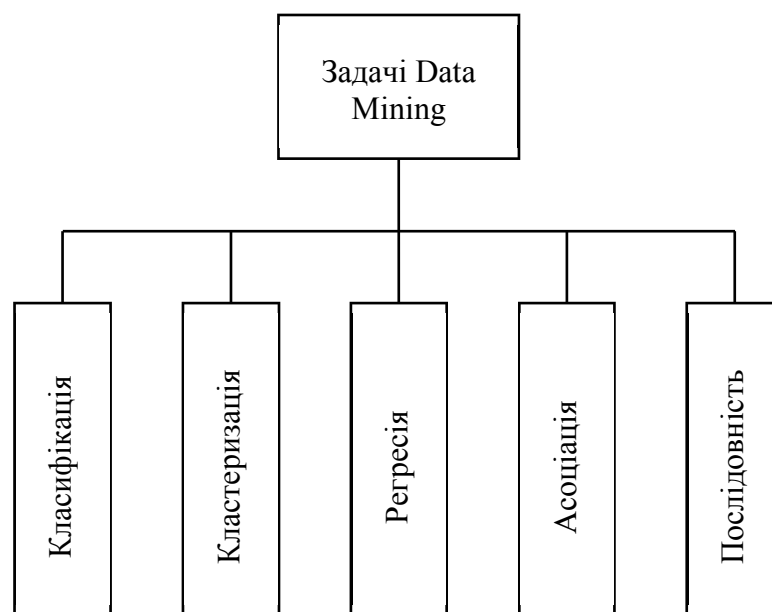


Рисунок 2.1 – Дерево основних задач інтелектуального аналізу даних

Класифікація є найбільш простою та поширеною задачею у Data Mining. Її метою є розподіл множини об'єктів на заздалегідь відомі класи

(групи). Кожен клас має певні ознаки та за цими ознаками будь-який об'єкт можна віднести до одного з визначених класів [16].

Кластеризація є логічним продовженням ідеї класифікації. Її особливість полягає в тому, що класи об'єктів спочатку не є визначеними. Таким чином, кластеризація виконує розподіл об'єктів на групи «схожих» об'єктів, які самі визначаються в ході вирішення завдання і називаються кластерами [16].

Регресія є завданням, в результаті якого на основі особливостей історичних даних виконується оцінка пропущених або майбутніх значень цільових чисельних показників [15]. Таким чином, регресія обчислює нове, раніше невідоме значення за історично накопиченими даними на основі деякої моделі залежності [17].

Асоціація вирішує задачу знаходження частих залежностей (або асоціацій) між об'єктами або подіями, які відбуваються одночасно. Знайдені залежності описуються у вигляді правил і можуть використовуватись для передбачення появи нових подій або для кращого розуміння природи даних, що аналізуються [14]. Цю модель можна представити у вигляді «Якщо – то» з деяким ступенем достовірності та її корисно використовувати в якості предикативної моделі, що дозволяє з набору A спрогнозувати набір B [17].

Послідовність дозволяє знаходити тимчасову закономірність між подіями. Це завдання є подібним до асоціації, але його метою є пошук закономірностей не між одночасними подіями, а тими, які пов'язані в часі [15]. Правило послідовності можна представити у вигляді «Після події A через певний час буде здійснена подія B » [17].

Перераховані задачі за призначенням діляться на описові та прогнозуючі.

Описові задачі спрямовані на удосконалення розуміння інформації. Головна особливість для таких моделей – легкість та прозорість результатів для сприйняття людиною [14]. Метою описових задач є краще розуміння даних, що аналізуються, виявлення наявних в них закономірностей, навіть

якщо в інших наборах даних вони зустрічатися не будуть [18]. До такого виду задач відноситься кластеризація та пошук асоціативних правил [14].

Прогнозуючі задачі використовуються на побудові моделі для передбачення нових значень на підставі набору даних з відомими результатами [18]. До такого виду задач відносяться задачі класифікації та регресії. Також до прогнозуючих задач можна віднести задачу пошуку асоціативних правил, якщо результати її вирішення можуть використовуватись для передбачення деяких подій [14].

За способом вирішення завдання поділяють на два типи: навчання з учителем та навчання без учителя.

Для типу навчання з учителем використовується навчальний набір даних, на основі якого створюється та навчається модель. Після цього вона тестується, тобто перевіряється якість роботи. Готову модель використовують для передбачення значень у нових наборах даних [18]. До цього типу задач відносяться завдання класифікації та регресії [14].

Метою для завдань з навчанням без учителя є виявлення закономірностей, наявних в існуючому наборі даних. Перевагою є те, що вирішення цих завдань можливе без будь-яких попередніх знань про дані, що аналізуються. До такого типу відносяться завдання кластеризації та пошуку асоціативних правил [14].

2.3 Методи інтелектуального аналізу даних

До методів інтелектуального аналізу даних відносяться наступні: штучні нейронні мережі, кластерний аналіз, лінійна регресія, дерева рішень, методи найближчого сусіда, метод опорних векторів, символні правила, методи пошуку асоціативних правил, мережі Байєса, кореляційно-регресійний аналіз, метод обмеженого перебору, еволюційне програмування та генетичні алгоритми, різноманітні методи візуалізації даних тощо [19]. Далі будуть описані основні з них.

Нейронні мережі – це клас моделей, які мають аналогію з мозком людини та призначені для вирішення різних типів завдань аналізу даних. Побудована мережа піддається процесу навчання на наявних даних, після чого вона готова до роботи та може використовуватися для прогнозування. Нейронна мережа – це свого роду «чорний ящик», де підхід зосереджується виключно на результаті, а не на тому, чому саме вона прийняла те чи інше рішення [14].

Кластерний аналіз є методом класифікаційного аналізу, який працює шляхом організації об'єктів в групи або кластери на основі того, наскільки тісно вони пов'язані між собою [20].

Лінійна регресія – це алгоритм, який дозволяє розрахувати лінійну залежність між вхідними і прогнозованими параметрами. Вхідні параметри повинні бути безперервними і мати відповідний тип даних. Це ж стосується і прогнозованих параметрів, яких може бути кілька [16].

Дерево рішень також є алгоритмом класифікації. Він будує модель аналізу даних шляхом розбиття даних за атрибутами. Ці розбиття представляються у вигляді вузлів, які нагадують гілки дерев [16].

Метод найближчого сусіда є найпростішим алгоритмом класифікації, який заснований на оцінці подібності об'єктів. Об'єкт відноситься до того класу, якому належать найближчі до нього об'єкти навчальної вибірки [21].

Метод опорних векторів є методом, за допомогою якого вирішується задача бінарної класифікації. Його мета полягає в побудові гіперплощини, які буде розділяти об'єкти вибірки оптимальним способом [19].

Методи пошуку асоціативних правил вирішують задачу знаходження частих залежностей між об'єктами або подіями, які відбуваються одночасно. Знайдені залежності описуються у вигляді правил і можуть використовуватись для передбачення появи нових подій [14].

Мережа Байєса – це граф, вершини якого з'єднані спрямованими ребрами, де кожен вузол має імовірнісну функцію. Вона являє собою спрямований ациклічний граф, тобто такий граф, в якому не існує

спрямованого маршруту, що починається і закінчується в одній і тій же вершині. Байєсівські мережі використовуються для моделювання предметних областей, які характеризуються невизначеністю. Ця невизначеність може бути обумовлена недостатнім розумінням предметної області, неповним знанням її стану в момент прийняття рішення, випадковим характером механізмів, що визначають поведінку або комбінацією цих факторів [22].

Кореляційний аналіз є методом обробки статистичних даних для визначення кількісної оцінки взаємозв'язку деяких наборів даних. Коефіцієнт кореляції використовується для визначення наявності взаємозв'язку між декількома властивостями [19].

Регресійний аналіз є методом вивчення статистичного взаємозв'язку між однією залежною кількісною змінною та однією або декількома незалежними кількісними змінними. Взаємозв'язок виражається у вигляді рівняння регресії. Регресійний та кореляційний аналіз дуже тісно пов'язані між собою. У кореляційному аналізі досліджується напрямок та тіснота зв'язку, а у регресійному – форма залежності між кількісними змінними. Таким чином обидва методи вивчають один і той же взаємозв'язок, але з різних сторін, і доповнюють один одного [23].

Методи обмеженого перебору використовуються для пошуку логічних закономірностей в даних. Ці алгоритми розраховують частоти комбінацій простих логічних подій в підгрупах даних. Обмеженням виступає довжина комбінації простих логічних подій. На основі аналізу розрахованих частот робиться висновок про корисність тієї чи іншої комбінації для встановлення асоціації в даних, класифікації, прогнозування тощо [24].

Всі методи інтелектуального аналізу даних можна поділити на дві основні групи за принципом роботи з вихідними даними. Це залежить від того, чи зберігаються дані після використання Data Mining або вони дистиллюються для подальшого використання [19].

На рисунку 2.2 зображено дерево основних методів, які використовуються в інтелектуальному аналізі даних.

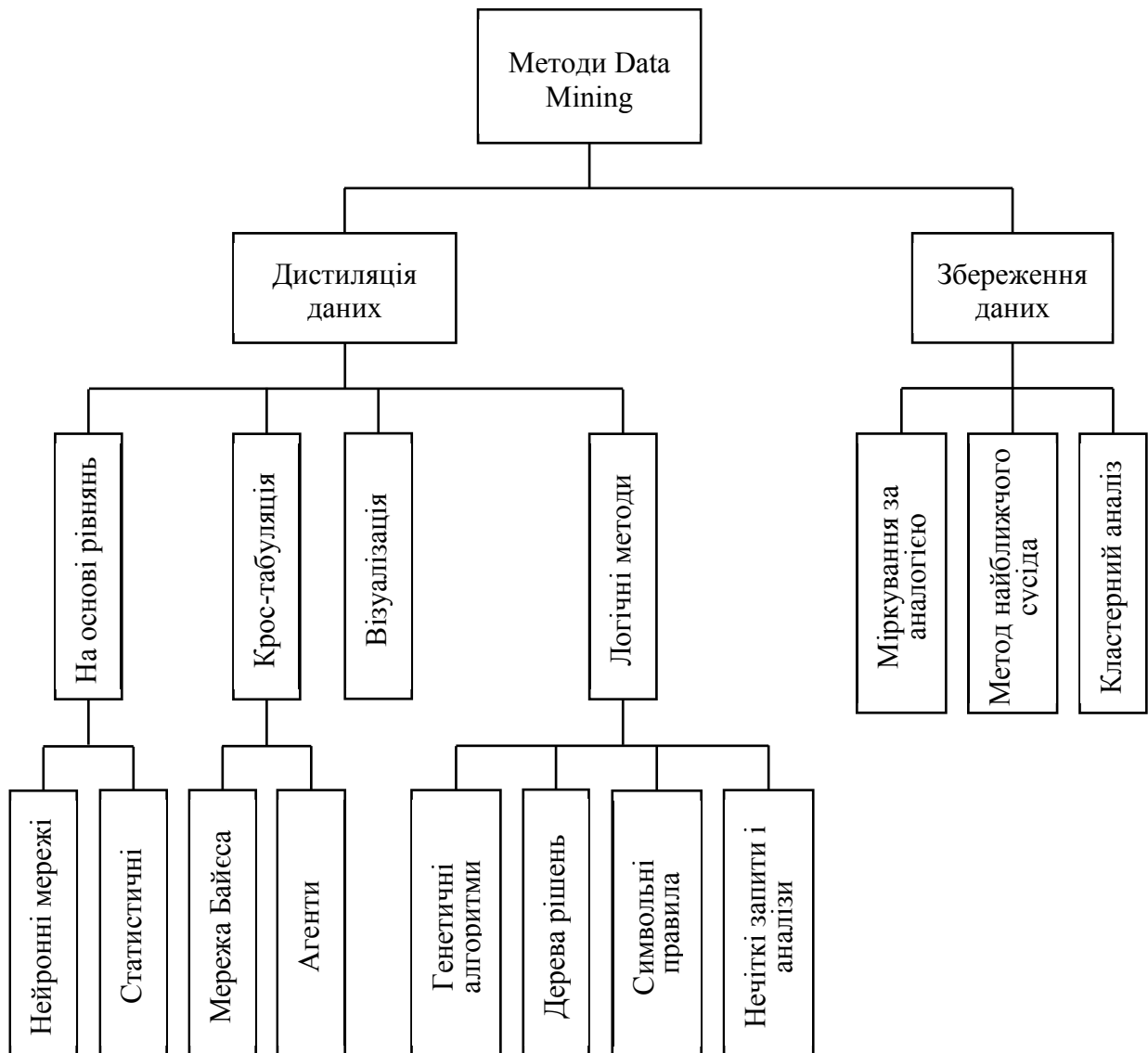


Рисунок 2.2 – Дерево основних методів інтелектуального аналізу даних

Для методів першої групи всі дані зберігаються в явному деталізованому виді та використовуються на стадіях прогностичного моделювання. Недоліком цієї категорії є те, що використання надвеликих баз даних може викликати деякі складності. До цієї групи належать наступні методи: кластерний аналіз, метод найближчого сусіда та міркування за аналогією [19].

У методах другої групи відбувається дистиляція даних, тобто на стадії вільного пошуку витягуються шаблони інформації, які у майбутньому можна використовувати на стадіях прогностичного моделювання. Такі шаблони займають менше пам'яті, тому що вони є значно компактнішими за самі бази

даних. До цієї групи належать: логічні методи; методи візуалізації; методи крос-табуляції; методи на основі рівнянь [19].

Логічні методи в свою чергу поділяються на нечіткі запити і аналізи, символічні правила, дерева рішень та генетичні алгоритми. Результати цих методів можна інтерпретувати у досить прозорому та зрозумілому виді для користувача [19].

Методи крос-табуляції включають агентів, мережу Байєса та крос-табличну візуалізацію [19].

Методи на основі рівнянь виявляють закономірності у вигляді математичних виразів, тобто рівнянь. Це означає, що вони можуть працювати тільки з числовими даними. Такі методи використовуються для вирішення різних задач, особливо для прогнозування. Вони поділяються на статистичні методи та нейронні мережі [19].

2.4 Етапи інтелектуального аналізу даних

Для добування знань з накопичених даних недостатньо тільки застосування методів Data Mining, хоча цей етап є ключовим у процесі інтелектуального аналізу [14]. Весь процес складається з 6 етапів, в який входять завдання від формування питань щодо даних до створення та розгортання моделі в робочому середовищі [15]. Отже, цей процес можна представити як послідовність наступних етапів:

- формування завдання;
- вивчення даних;
- підготовка даних;
- побудова моделей;
- перевірка моделей;
- розгортання моделей [25].

На рисунку 2.3 зображено процес інтелектуального аналізу даних, який є ітеративним. Він продовжує тривати навіть після розгортання моделі. Отримані знання в ході цього процесу можуть формувати нові завдання [25]. Також може виникнути ситуація, коли після побудови та оцінки моделі виявиться, що вона не дає адекватну відповідь на поставлене питання, тому необхідно поставити завдання іншими чином та почати весь процес спочатку. Може виникнути необхідність в оновленні вже розгорнутих моделей за рахунок надходження нових даних. Таким чином, для створення адекватної моделі може знадобитися багаторазово повторити кожен крок процесу [26].

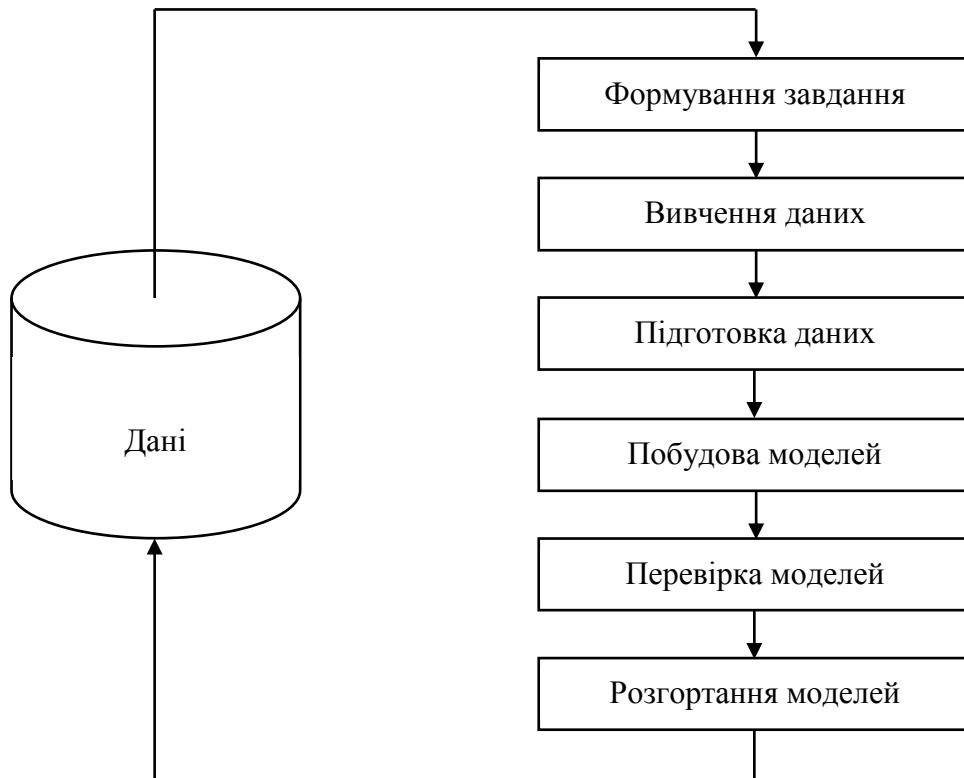


Рисунок 2.3 – Етапи інтелектуального аналізу даних

Етап 1. Спочатку виконується розуміння цілей та вимог проєкту, які будуть досягнуті методами Data Mining. Цей етап є важливим, тому що потрібно правильно сформулювати завдання і обрати необхідні методи для досягнення мети, бо від цього залежить подальша ефективність всього процесу [14].

Етап 2. Після того, коли сформоване завдання, потрібно почати збір даних. Після ознайомлення з ними, важливо зрозуміти які дані є доступними, як вони виглядають та як їх можна використовувати. Дані можуть бути різних форматів: звичайний текст, зображення, відео, звук тощо. Вивчення даних дозволяє виявити проблеми з якістю даних та сформувані гіпотези про приховану інформацію, щоб надалі знайти рішення поставленої задачі [25].

Етап 3. Підготовка даних зазвичай займає близько 90% від часу проєкту. Результатом підготовки даних є перетворення вихідних необроблених даних у кінцевий набір даних, який буде використаний під час побудови моделі. На цьому етапі вибираються тільки необхідні дані, відбувається очищення та перетворення їх у потрібну форму [25].

Етап 4. Побудова моделей – це етап, де застосовуються методи інтелектуального аналізу даних. На цьому етапі створюється модель, де знаходяться закономірності даних, яка буде використовуватись для рішення поставлених завдань. Сценарії застосування можуть бути різноманітними та можуть включати складну комбінацію різних методів Data Mining [14].

Етап 5. Перед розгортанням моделі потрібно перевірити ефективність роботи моделі. Дуже часто використовується спосіб, коли всі дані розділяються на дві групи. Як правило, одна з них більшого розміру, інша меншого. На більшій групі застосовуються методи Data Mining для побудови моделі, а на меншій – перевіряють її. За результатами точності між тестовою та навчальною вибірками можна оцінити адекватність побудованої моделі [14]. Також на етапі побудови моделей зазвичай створюється кілька моделей з різними параметрами, а потім обирається найкраща, яка задовольняє рішення поставленого завдання. Якщо жодна з моделей не володіє потрібною ефективністю, може виникнути необхідність повернутися до попереднього кроку [26].

Етап 6. Розгортання моделей – це організація та представлення результатів інтелектуального аналізу даних. На сьогодні існує два популярних

способи розгортання моделі: вбудовування створеної моделі в існуючу програму або спілкування з нею через Application Programming Interface (API). Знання повинні бути представлені таким чином, щоб зацікавлені особи змогли використовувати їх у будь-який момент. Цей етап може бути таким же простим як створення звіту, або таким же складним як повторюваний процес Data Mining [25].

2.5 Алгоритми пошуку асоціативних правил

Алгоритм пошуку асоціативних правил дозволяє знаходити часті поєднання елементів даних, які відбуваються одночасно та використовувати виявлені закономірності для побудови прогнозу [18].

Отримані результати зазвичай представляються у вигляді асоціативних правил. Вирішення цього завдання складається з двох етапів:

- пошук наборів об'єктів, які часто зустрічаються;
- пошук асоціативних правил по знайденим наборам об'єктів [14].

Далі буде описана задача пошуку частих наборів об'єктів в загальному виді. Об'єкти, які утворюють досліджувані набори можна представити як множину (2.1):

$$I = \{i_1, i_2, \dots, i_j, \dots, i_n\}, \quad (2.1)$$

де i_j – об'єкти, які входять у досліджувані набори;

n – загальна кількість об'єктів.

Транзакція – це набір об'єктів, які входять у множину I та будуть аналізуватися програмою. Її можна представити як множину (2.2):

$$T = \{i_j \mid i_j \in I\}. \quad (2.2)$$

Набір транзакцій, який доступний для аналізу можна представити у вигляді наступної множини (2.3):

$$D = \{T_1, T_2, \dots, T_r, \dots, T_m\}, \quad (2.3)$$

де m – кількість доступних для аналізу транзакцій.

Множина транзакцій, до якої входить об'єкт i_j можна позначити як множину (2.4):

$$D_{i_j} = \{T_r \mid i_j \in T_r; j = 1..n; r = 1..m\} \subseteq D. \quad (2.4)$$

Довільний набір об'єктів можна позначити як множину (2.5):

$$F = \{i_j \mid i_j \in I; j = 1..n\}. \quad (2.5)$$

Набір транзакцій, у яку входить набір F можна позначити наступною множиною (2.6):

$$D_F = \{T_r \mid F \subseteq T_r; r = 1..m\} \subseteq D. \quad (2.6)$$

Асоціативні правила можна представити у вигляді «Якщо A , то B ». В даному випадку A – це умова, тобто набір будь-яких об'єктів з множини I , яка пов'язана з B , тобто результатом, який також включає об'єкти з множини I [14].

Головна перевага для асоціативних правил – це їх наочність, а також те, що вони легко сприймаються людиною та інтерпретуються різними мовами програмування [14]. Розділяють наступні види асоціативних правил:

- корисні правила – це такі правила, що містять в собі інформацію, яка зможе принести значну користь. Вони раніше не були відомі та були

отримані в процесі інтелектуального аналізу даних. Також ці правила повинні бути зрозумілими та логічними;

- тривіальні правила – це такі правила, що містять в собі інформацію, яка є істинною, але вже є відомою та очевидною. Ці правила легко пояснити точки зору логіки, але їх використання не принесе багато користі, тому що більшість з них відображають вже відому інформацію. Іноді такі правила можна застосовувати для перевірки правильності вже прийнятих рішень;

- незрозумілі правила – це такі правила, що містять інформацію, яку неможливо пояснити. Вони можуть бути отримані в результаті аналізу глибоко прихованих знань або в результаті аналізу аномальних та нестандартних значень. Такі правила не можна використовувати в чистому вигляді, тому що через неможливість пояснення їх з точки зору логіки, застосування може призвести до невідомих та непередбачуваних результатів [14].

Для пошуку асоціативних правил потрібно знайти набори об'єктів, які часто зустрічаються. Внаслідок великої кількості таких наборів при великому обсязі даних, що аналізуються можна отримати занадто велику кількість правил, які важко зрозуміти людині. До того ж, не всі правила можуть бути корисними. Для того, щоб скоротити кількість непотрібних правил та оцінити їх корисність використовуються наступні величини [14].

Підтримка (support) – це величина, яка характеризує якість асоціативного правила. Вона показує який відсоток транзакцій підтримує дане правило, тобто скільки раз воно зустрічається у всьому наборі [14]. Обчислюється як відношення числа транзакцій, в який входить набір F , тобто у якому з'являється умова A і результат B правила, до загального набору транзакцій D за формулою (2.7):

$$Supp_{A \Rightarrow B} = Supp_F = \frac{D_{F=A \cup B}}{D}. \quad (2.7)$$

Підтримка може приймати значення від 0 (коли умова та результат не зустрічаються ні в одній транзакції) до 1 (коли умова та результат з'являються у всіх транзакціях). У загальному випадку підтримка є мірою надійності, яка виражає асоціативний зв'язок між умовою та результатом. Якщо значення підтримки $Supp > 80\%$, то зв'язок є сильним та це правило заслуговує довіри. Якщо $50\% < Supp < 80\%$, то асоціативний зв'язок середній, а правило слід використовувати з обережністю. При $Supp < 50\%$, зв'язок є слабким, а правило вважається сумнівним [27].

Достовірність (confidence) – це величина, яка показує ймовірність того, що поява події A спричиняє за собою появу події B , тобто наскільки часто спрацьовує правило [14]. Обчислюється як відношення числа транзакцій, що містять набори A і B , до числа транзакцій, що містять набір A за формулою (2.8):

$$Conf_{A \Rightarrow B} = \frac{D_{F=A \cup B}}{D_A} = \frac{Supp_{A \cup B}}{Supp_A}. \quad (2.8)$$

Достовірність дозволяє відсіяти частину слабких правил, але вона не показує, наскільки правило є корисним. Для визначення того, наскільки воно є корисним, використовується міра, як ліфт правила (також зустрічається термін «поліпшення» або «improvement») [14].

Ліфт (lift) – це міра, яка показує, чи є правило випадковим або ж воно має логічне обґрунтування, тобто наскільки підвищується ймовірність появи події B в наборі, якщо в ньому вже є подія A [14]. Обчислюється як відношення числа транзакцій, що містять набори A та B , до добутку кількості транзакцій, що містять набір A та транзакцій, що містять набір B за формулою (2.9):

$$Lift_{A \Rightarrow B} = \frac{D_{F=A \cup B}}{D_A \times D_B} = \frac{Supp_{A \cup B}}{Supp_A \times Supp_B}. \quad (2.9)$$

Ліфт може приймати значення від 0 до нескінченності. Якщо значення ліфта $Lift > 1$, це вказує на те, що умова та результат частіше зустрічаються разом у транзакції, ніж окремо, тобто поява умови з великою ймовірністю спричиняє появу результату. Таке правило можна вважати значущим. Значення ліфта $Lift < 1$ вказує на те, що умова та результат зустрічаються у транзакціях частіше окремо, ніж разом. Якщо значення ліфта є близьким до 1, то умова та результат зустрічаються спільно приблизно в стільки ж транзакціях, в скількох і окремо, тобто умова та результат не впливають на появу один одного [28].

Існує багато алгоритмів для пошуку асоціативних правил, зокрема: Apriori, FP-Growth, Eclat, dEclat, AprioriDP, AprioriTID, Apriori Hybrid, CBPNARM, FIN, PrePost, PrePost+, PPV, ASSOC, OPUS, Relim, H-mine, LCMFreq [29], [30].

2.5.1 Алгоритм Apriori

Алгоритм Apriori було запропоновано у 1994 році та на сьогодні він є одним з найбільш популярних алгоритмів пошуку асоціативних правил. Він дозволяє скоротити область пошуку завдяки використанню властивості антимонотонності, яка стверджує, що підмножина набору з частих елементів робить цей набір частим. Аналогічно, що підмножина набору з нечастих елементів робить цей набір нечастим і така підмножина відсікається. Це дозволяє виконувати пошук по ширині. Хоча властивість антимонотонності не дозволяє розглядати всі можливі комбінації елементів, то навіть таке сканування з відсіканням неефективно використовувати для великих наборів даних [30].

На основі дослідження джерел [30] та [31] було складено схему алгоритму Apriori (рис. 2.4).

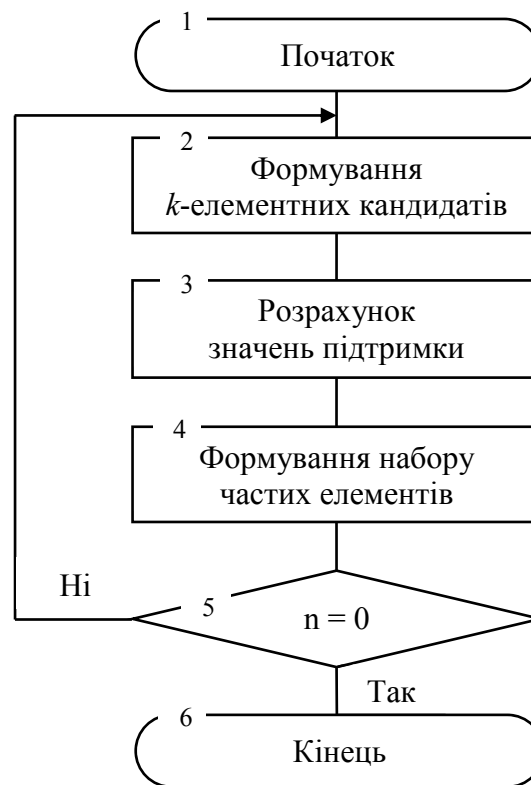


Рисунок 2.4 – Схема алгоритму Apriori

Блок 1. Початок.

Блок 2. Формування множини k -елементних кандидатів, де k – це номер ітерації.

Блок 3. Розрахунок значення підтримки для кожного k -елементу з множини сформованих кандидатів.

Блок 4. Формування набору з n частих елементів, тобто здійснюється відсікання таких кандидатів, підтримка яких менша мінімальної.

Блок 5. Якщо подальша генерація кандидатів неможлива, тобто відсутній набір елементів, що часто зустрічається, то перехід до блоку 6. В іншому випадку $k + 1$ та перехід до блоку 2.

Блок 6. Кінець.

Можна виділити наступні переваги алгоритму Apriori:

- зрозумілість та простота в реалізації;
- швидке зменшення числа згенерованих кандидатів для високого значення мінімальної підтримки [32].

Недоліки алгоритму Apriori:

- багаторазове сканування вхідного набору, що призводить до великого використання обчислювальних ресурсів;
- генерація великої кількості кандидатів при занадто великому наборі даних або при занадто низькому значенні підтримки, що призводить до великого використання оперативної пам'яті [32].

Таким чином алгоритм Apriori ефективний тільки для невеликих наборів даних або при високому значенні мінімальної підтримки [32].

2.5.2 Алгоритм FP-Growth

На основі дослідження джерел [30]-[33] було складено схему алгоритму FP-Growth (рис. 2.5).

FP-Growth є одним з найефективніших алгоритмів, що дозволяє уникнути не тільки кошовної процедури формування кандидатів, але й багаторазового сканування вхідного набору [32]. Метод виконує сканування вхідного набору лише два рази. Під час першого, він знаходить список всіх частих елементів та сортує їх в порядку убутання. Під час другого сканування весь вхідний набір стискається в FP-дерево, після чого відбувається пошук популярних наборів [33].

Блок 1. Початок.

Блок 2. Сканування вхідного набору та обчислення значення підтримки для кожного елемента.

Блок 3. Сортування всіх елементів за значенням підтримки в порядку убутання.

Блок 4. Видалення елементів, підтримка яких менша за мінімальну.

Блок 5. Побудова префіксного FP-дерева з елементів, що залишилися.

Блок 6. Формування набору частих елементів з побудованого FP-дерева.

Блок 7. Кінець.



Рисунок 2.5 – Схема алгоритму FP-Growth

Можна виділити наступні переваги алгоритму FP-Growth:

- уникнення коштовної процедури генерації кандидатів;
- стиснення вхідного набору в компактну структуру, що забезпечує більш ефективне використання оперативної пам'яті;
- кількість сканувань вхідного набору скорочено до двох разів [32].

Недоліки алгоритму FP-Growth:

- побудова FP-дерева є витратною за часом операцією [32].

Таким чином алгоритм FP-Growth більше підходить для швидкого та точного отримання асоціацій з дуже великих наборів даних [33].

2.5.3 Алгоритм Eclat

Eclat є алгоритмом, який здійснює пошук в глибину, тобто використовує вертикальний формат даних та відрізняється від Apriori і FP-Growth тим, що вони використовують горизонтальний формат даних [34].

Горизонтальні дані представляють унікальну транзакцію $T_1, T_2, \dots, T_m$ в наборі даних та включають набір елементів $I_1, I_2, \dots, I_n$. Вертикальні дані представляють унікальний елемент $I_1, I_2, \dots, I_n$ та включають набір транзакцій $T_1, T_2, \dots, T_m$ в яких знаходиться відповідний елемент. Основною перевагою вертикального формату є можливість швидкого підрахунку значення підтримки елементу, тому що не потрібно сканувати весь набір, як при використанні горизонтальних даних [34].

На основі дослідження джерел [31], [34] - [35] було складено схему алгоритму Eclat (рис. 2.6).

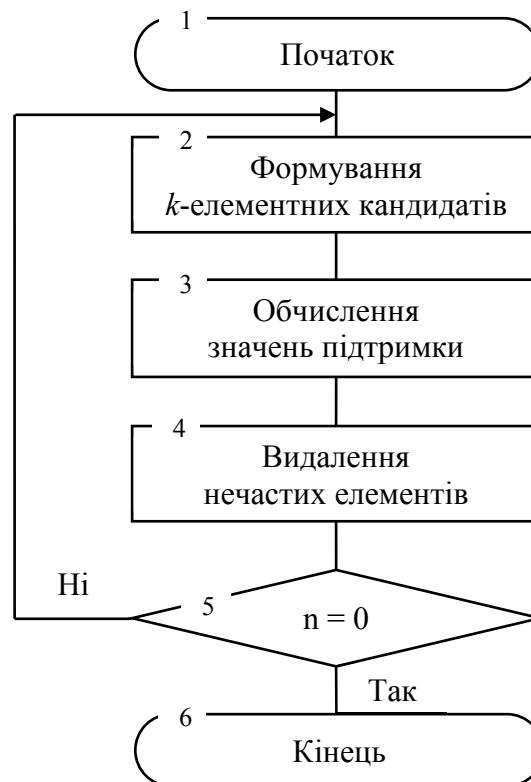


Рисунок 2.6 – Схема алгоритму Eclat

Блок 1. Початок.

Блок 2. Формування множини k -елементних кандидатів, де k – це номер ітерації.

Блок 3. Обчислення значення підтримки для кожного k -елементу з множини сформованих кандидатів.

Блок 4. Видалення тих елементів, підтримка яких менша за мінімальну, тобто відбувається формування набору з n частих елементів.

Блок 5. Якщо подальша генерація кандидатів неможлива, тобто відсутній набір елементів, що часто зустрічається, то перехід до блоку 6. В іншому випадку $k + 1$ та перехід до блоку 2.

Блок 6. Кінець.

Можна виділити наступні переваги алгоритму Eclat:

- зрозумілість та простота в реалізації;
- використання підходу пошуку по глибині забезпечує менше споживання оперативної пам'яті, ніж алгоритм Apriori;
- значення підтримки для будь-якого елемента розраховується без сканування вхідного набору;
- відбувається сканування лише поточного згенерованого набору даних на відміну від Apriori, який сканує вихідний набір даних;
- число сканувань базового набору скорочено до одного разу [35].

Недоліки алгоритму Eclat:

- вертикальне представлення даних іноді може виявитися занадто великим, тому операції з ними можуть займати тривалий час [32].

Таким чином алгоритм Eclat краще підходить для невеликих та середніх наборів даних, з якими він працює швидше у порівнянні з алгоритмом Apriori [35].

2.6 Огляд існуючих програмних засобів інтелектуального аналізу даних

Існує багато програмних засобів та бібліотек, які дозволяють аналізувати великі набори даних, а також застосовувати до них різні алгоритми інтелектуального аналізу та машинного навчання.

На основі дослідження джерел [36] - [38] було розглянуто наступні програмні засоби, що дозволяють вирішувати аналогічні завдання: Rapid Miner, Orange, Weka, R, scikit-learn.

Rapid Miner – це інтегроване середовище, яке надає можливості для обробки та підготовки вхідних даних, а також включає в себе різноманітні алгоритми машинного навчання та інтелектуального аналізу даних. Програма повністю написана на мові програмування Java та її остання версія – 9.6, яка була опублікована 2 березня 2020 року. Rapid Miner є дуже простим у використанні та не потребує знань в області програмування, що є головною перевагою. Недоліком є те, що використання такого програмного забезпечення є платним [37]-[39].

Приклад інтерфейсу Rapid Miner зображено на рисунку 2.7.

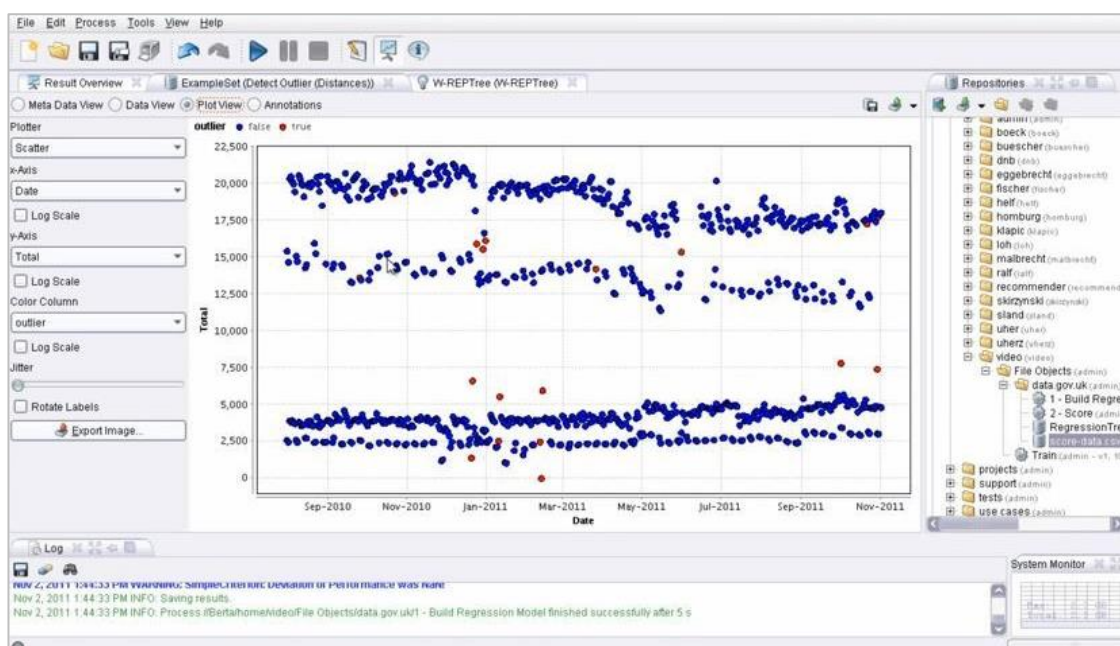


Рисунок 2.7 – Інтерфейс Rapid Miner [39]

Orange – це програмне забезпечення з відкритим вихідним кодом для візуалізації даних, машинного навчання та інтелектуального аналізу даних. Головною перевагою використання Orange є те, що користувачу доступний інтерфейс візуального програмування, в якому робочі процеси створюються шляхом зв'язування вбудованих віджетів. Також є можливість створення власних віджетів для розширення функціональності. Програма написана на мові програмування Python та є повністю безкоштовною. Остання версія Orange – 3.28, яка була опублікована 14 березня 2021 року [37], [40].

Приклад інтерфейсу Orange зображено на рисунку 2.8.

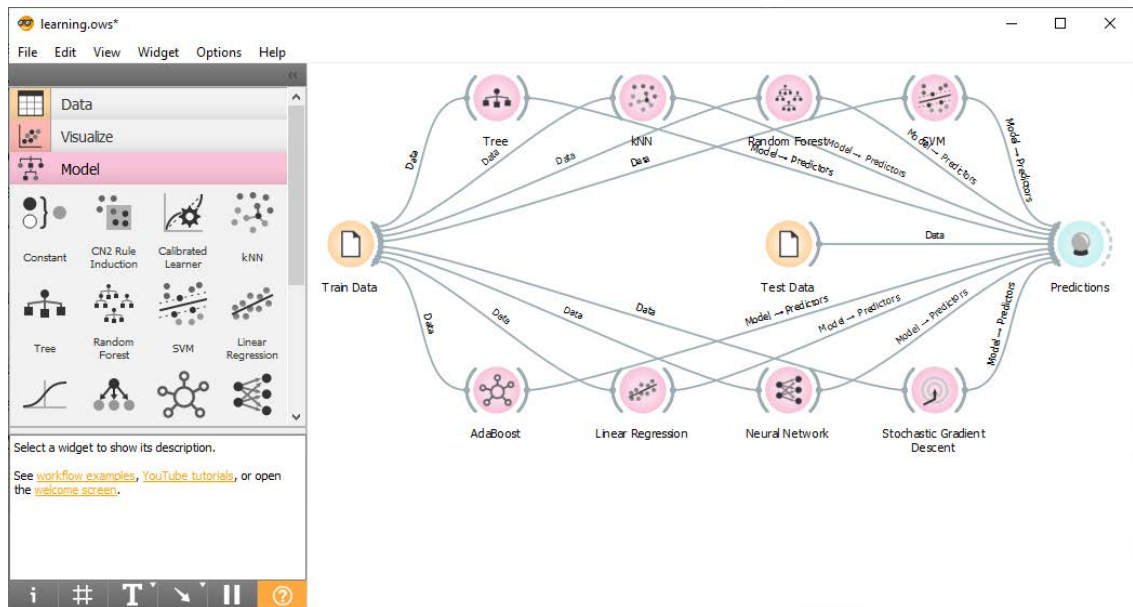


Рисунок 2.8 – Інтерфейс Orange

Weka – це програмне забезпечення з відкритим вихідним кодом, яке написано на мові програмування Java. Воно надає повну колекцію алгоритмів машинного навчання та інструментів попередньої обробки даних, що дозволяє користувачам швидко випробовувати та порівнювати різні методи машинного навчання на нових наборах даних. Простий графічний інтерфейс Weka надає можливість створювати складні процеси інтелектуального аналізу даних на основі широкого набору базових алгоритмів навчання та інструментів, що є великою перевагою. Використання API дозволяє розширювати інструментарій

програми та створювати різні плагіни. Остання версія Weka – 3.6.13, яка була опублікована 11 вересня 2015 року [41].

Приклад інтерфейсу Weka зображено на рисунку 2.9.

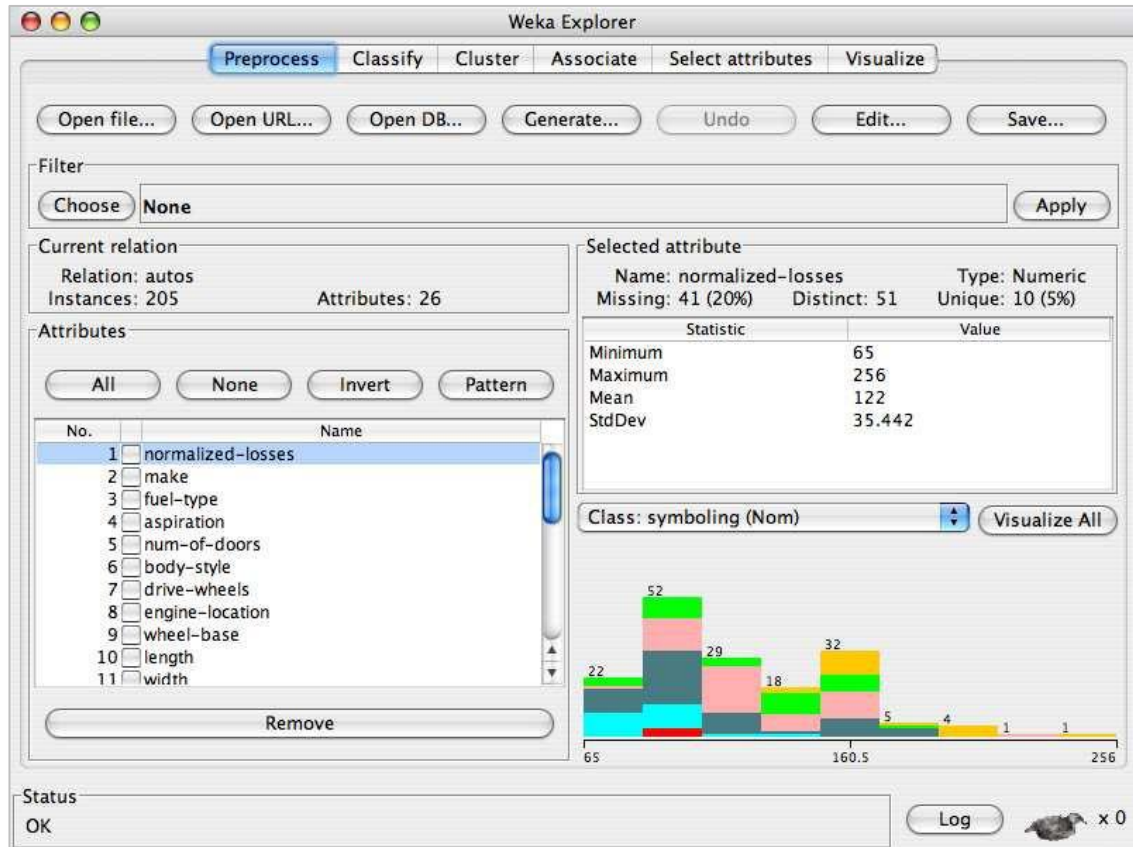


Рисунок 2.9 – Інтерфейс Weka [41]

R – це інтерпретована мова програмування та інструмент з відкритим вихідним кодом, який включає в себе реалізацію багатьох алгоритмів машинного навчання, а також широкий набір методів візуалізації статистичних даних. Інструмент пропонує середовище розробки, яке може виявитися не дуже зручним для багатьох користувачів, тому що всі команди повинні бути введені на мові R. Це є головним недоліком, тому що для роботи з таким інструментом потрібно володіти навичками з програмування, а на ретельне вивчення цієї мови може знадобитися деякий час. Остання версія мови R – 4.0.5, яка була опублікована 31 березня 2021 року [42].

scikit-learn – це бібліотека для мови програмування Python, яка надає широкий спектр алгоритмів машинного навчання, інструментів оцінки, методів попередньої обробки та інтелектуального аналізу даних. Бібліотека є безкоштовною, яку легко можна інтегрувати в різні програми та використовувати її як в академічних, так і в комерційних цілях. Також до переваг можна віднести добре написану документацію по всім реалізованим алгоритмам, що робить цю бібліотеку простою у використанні. Незважаючи на всі переваги, scikit-learn вимагає базових знань у програмуванні на мові Python, що може не підійти до користувачів, які не є програмістами. Остання версія бібліотеки – 0.24.1, яка була опублікована 19 січня 2021 року [43].

Результати порівняльного аналізу розглянутих програмних засобів зведено в табл. 2.2.

Таблиця 2.2 – Порівняння розглянутих програмних засобів

Характеристика	Rapid Miner	Orange	Weka	R	scikit-learn
Безкоштовний тип ліцензії	–	+	+	+	+
Останнє оновлення	Березень 2020 р.	Березень 2021 р.	Вересень 2015 р.	Березень 2021 р.	Січень 2021 р.
Мова програмування	Java	Python	Java	R	Python
Графічний інтерфейс	+	+	+	–	–
Створення скриптів	–	+	+	+	+
Рейтинг користувачів	9.5 / 10.0	9.5 / 10.0	9.1 / 10.0	9.1 / 10.0	7.6 / 10.0

Як висновок до даного порівняння, можна зазначити той факт, що програмний засіб Orange є найкращим, тому що він має найбільше переваг серед розглянутих аналогів, а саме: він є повністю безкоштовним, регулярно

оновлюється розробниками, має простий та зручний у використанні графічний інтерфейс, а також не потребує навичок у програмуванні та має дуже високий рейтинг серед користувачів.

2.7 Огляд функціональних особливостей Orange

В Orange всі компоненти називаються віджетами, які можна зв'язувати один з одним для отримання робочих процесів. Всі віджети можна поділити на такі групи: робота з даними та їх попередня обробка, візуалізація даних, алгоритми машинного навчання та оцінка їх ефективності [37].

Дерево всіх віджетів Orange зображено на рисунку 2.10.

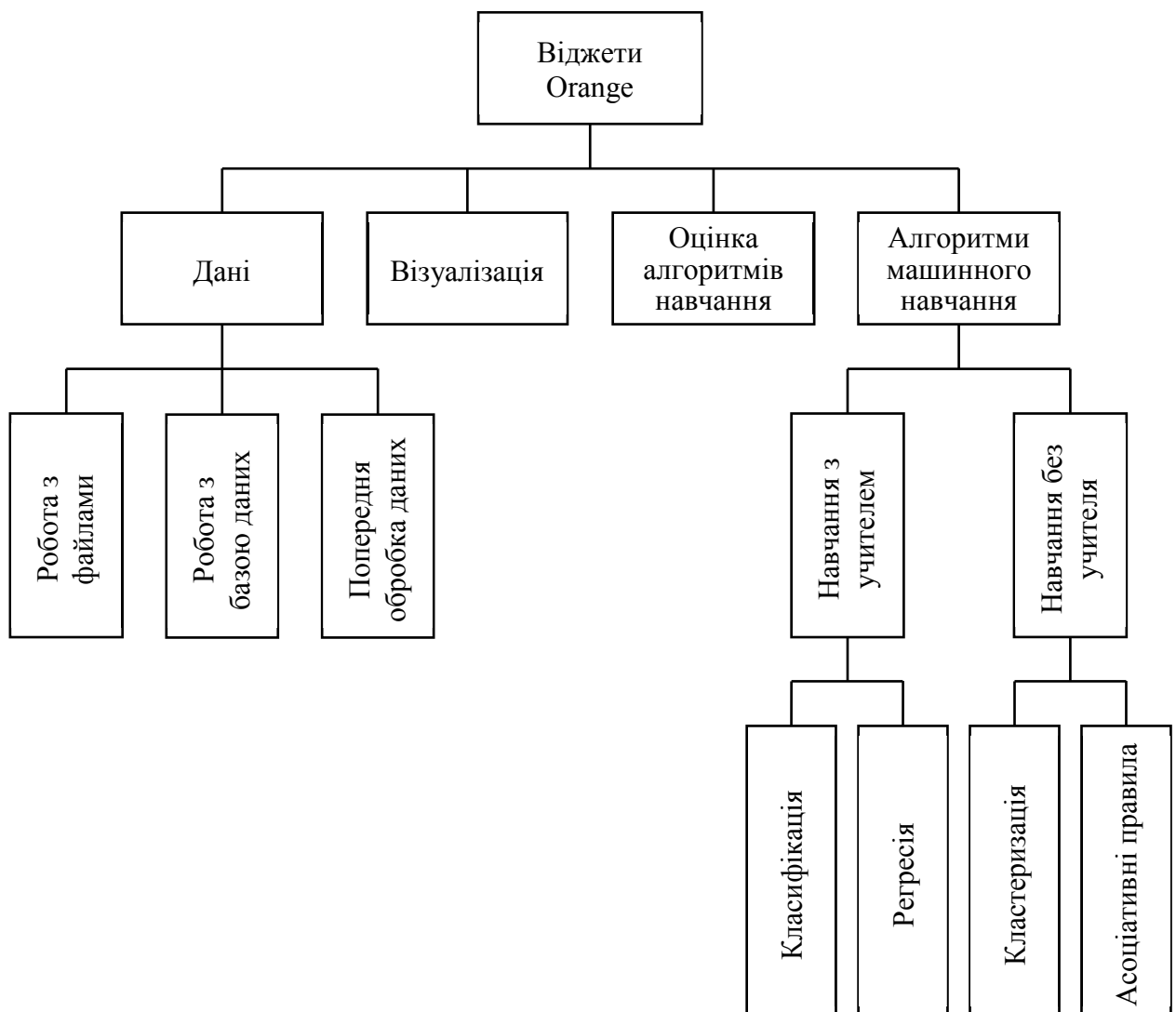


Рисунок 2.10 – Дерево всіх віджетів програмного застосунку Orange

Група віджетів для роботи з даними дозволяє працювати з файлами різних форматів: файли Excel (.xlsx), текстові дані з роздільниками у вигляді табуляції (.txt), CSV-файли (.csv), посилання на файли з мережі Інтернет. Також є можливість для роботи з популярними базами даних. Крім того, ця група надає різні інструменти для попередньої обробки даних, наприклад: фільтрація даних за умовою; створення навчальної та тестової вибірки; заповнення пропущених значень; видалення невикористовуваних значень та атрибутів; дискретизація безперервних атрибутів тощо. Ще в Orange є можливість створення власних скриптів на мові програмування Python для реалізації власних функцій під час роботи з даними [44].

Група віджетів для візуалізації даних дозволяє відобразити дані у вигляді різних графіків та діаграм, що може допомогти проаналізувати вхідні дані або результати роботи алгоритмів [44].

Група віджетів для роботи з алгоритмами класифікації та регресії дозволяють застосувати такі методи та алгоритми: алгоритм k найближчих сусідів, «випадковий ліс», метод опорних векторів, лінійної регресії, класифікатор Байєса, алгоритм AdaBoost, алгоритм багат шарового перцептрона, метод стохастичного градієнтного спуску [44].

Група віджетів для оцінки ефективності алгоритмів навчання дозволяє провести тестування використаних алгоритмів на тестових даних та побачити які конкретні випадки були класифіковані неправильно та яким чином [44].

Група віджетів для роботи з алгоритмами кластеризації дозволяють застосувати такі методи та алгоритми: алгоритм ієрархічної кластеризації, кластеризації k-середніх, метод Лувена, алгоритм DBSCAN, метод багатовимірного шкалювання [44].

Група віджетів для пошуку асоціативних правил дає можливість застосувати алгоритм пошуку асоціативних правил FP-Growth, який генерує асоціативні правила для вхідного набору даних [44].

2.8 Висновки до розділу

Були розглянуті основні методи інтелектуального аналізу даних. Для СДА можна виділити шість основних напрямків використання інтелектуальних технологій: розпізнавання активності, голосу і зображень; обробка даних; прийняття рішень і прогнозування. На сьогоднішній день технології розпізнавання голосу і зображень є найбільш розвиненими в сучасних системах. У той же час, існуючі рішення для розпізнавання активності, обробки даних, підтримки прийняття рішень та прогнозування не дають повного функціоналу для проведення інтелектуального моніторингу енергоспоживання, а застосовувані методи інтелектуального аналізу даних для СДА вимагають подальшого розвитку [45]. На рисунку 2.11 зображені основні напрямки використання інтелектуальних технологій для СДА.



Рисунок 2.11 – Основні напрямки використання інтелектуальних технологій для СДА

Також були проаналізовані існуючі програмні засоби та бібліотеки, що вирішують аналогічні завдання: Rapid Miner, Orange, Weka, R та scikit-learn. Їх порівняльний аналіз зведено в табл. 2.2. Після вивчення всіх особливостей,

можна зробити висновок, що програмне забезпечення Orange має найбільше переваг серед розглянутих аналогів та є найкращим у використанні. По-перше, воно є повністю безкоштовним та з відкритим вихідним кодом. По-друге, останнє оновлення було опубліковано у березні 2021 року та це означає, що цей програмний продукт підтримується та регулярно оновлюється розробниками. По-третє, Orange має простий та зручний у використанні графічний інтерфейс, що дає змогу швидко навчитися користуватися всіма його функціями. Найголовніша перевага полягає в тому, що цей програмний засіб не потребує навичок у програмуванні. Таким чином, за допомогою інтерфейсу візуального програмування користувач з легкістю може використовувати всі алгоритми та інструменти, шляхом зв'язування вбудованих віджетів один з одним. Також важливим критерієм є те, що Orange має дуже високий рейтинг серед користувачів у порівнянні з його аналогами.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ ДЛЯ СИСТЕМИ ДОМАШНЬОЇ АВТОМАТИЗАЦІЇ

3.1 Опис архітектури системи домашньої автоматизації

Архітектуру СДА для реальної однокімнатної квартири наведено на рисунку 3.1.

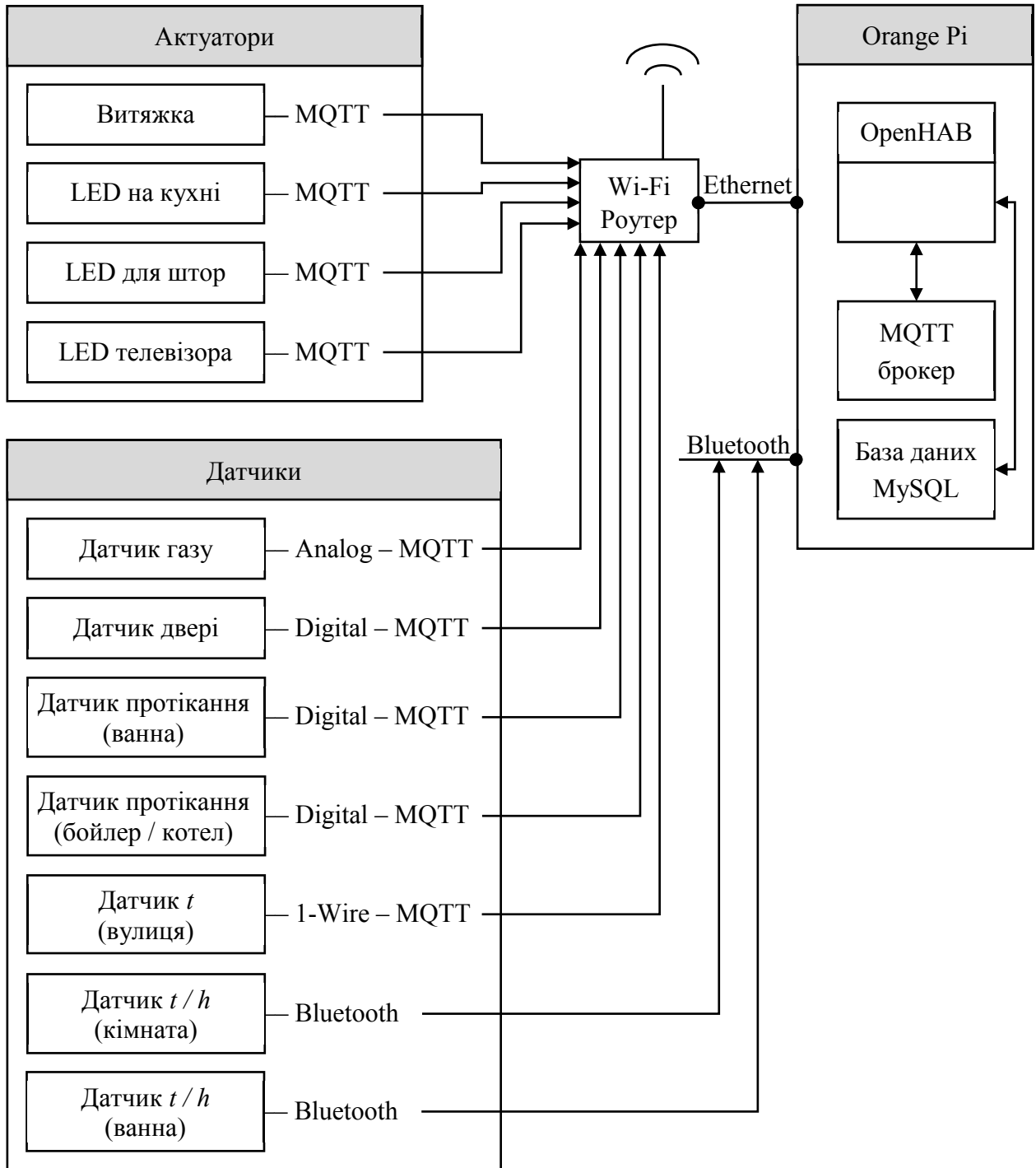


Рисунок 3.1 – Архітектура досліджуваної СДА

Для накопичення даних з СДА та їх подальшого аналізу була використана справжня однокімнатна квартира в якій проживає дві людини.

В ролі програмної частини використовується платформа Open Home Automation Bus (OpenHAB). Це програмне забезпечення з відкритим вихідним кодом, яке написано на мові програмування Java. OpenHAB допомагає об'єднати багато різних систем домашньої автоматизації, пристроїв та технологій в єдине рішення. Воно забезпечує єдиний користувацький інтерфейс (рис. 3.2) та загальний підхід до правил автоматизації у всій системі, незалежно від кількості виробників, що використовуються [2].

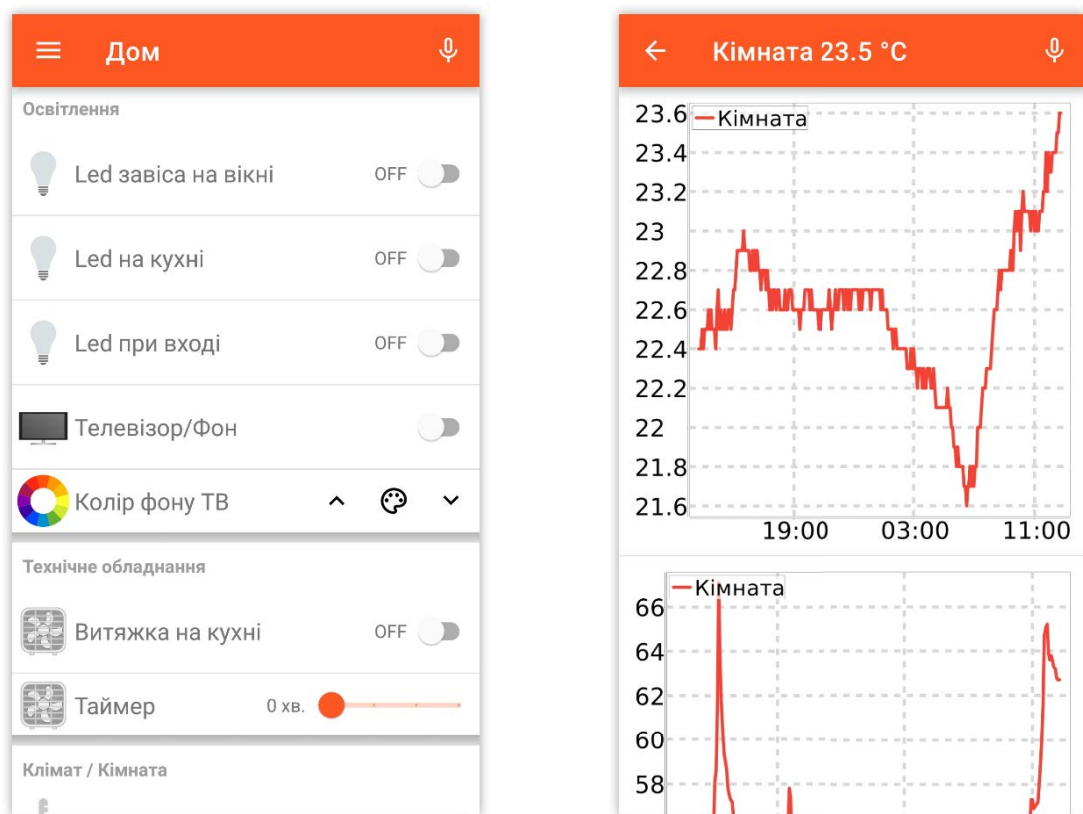


Рисунок 3.2 – Інтерфейс управління OpenHAB

Використання OpenHAB надає безліч переваг:

- повністю з відкритим вихідним кодом;
- можливість встановлення на будь-яку операційну систему (ОС):

Windows, Linux або Mac;

- інтеграція з багатьма різними технологіями домашньої автоматизації незалежно від виробників;
- регулярне оновлення програмного забезпечення;
- можливість керування за допомогою веб-додатку, а також за допомогою мобільних застосунків для Android та iOS;
- створення власних правил та сценаріїв для автоматизації пристроїв;
- використання API для взаємодії з іншими системами [46].

Wireless Fidelity (Wi-Fi) – це мережева технологія, яка використовує радіохвилі для високошвидкісної передачі даних на короткі відстані з клієнтського пристрою на точку доступу, що включає в себе маршрутизатор, який виконує з'єднання з іншими пристроями в локальній мережі або мережі Інтернет [47].

Bluetooth – це технологія бездротового зв'язку, яка дозволяє таким пристроям, як мобільні телефони, комп'ютери та периферійні пристрої передавати дані по бездротовій мережі на коротку відстань. Всі з'єднання є зашифрованими, що дозволяє безпечно передавати дані між пристроями [48].

Bluetooth Low Energy (BLE) – це різновид бездротового стандарту Bluetooth, який призначений для низького енергоспоживання. Він використовується у пристроях, які не потребують великої кількості передачі даних, що дозволяє працювати пристрою від однієї батареї протягом кількох років [49].

Мікрокомп'ютер Orange Pi – це пристрій, що складається всього з однієї плати, але має характеристики, які можна порівняти з повноцінним ПК. Цей прилад оснащений чотирьохядерним процесором AllWinner H5 та має 1 або 2 ГБ оперативної пам'яті. У платі є різні роз'єми для підключення периферії: High Definition Multimedia Interface (HDMI), аудіовихід, місце для установки карти пам'яті, роз'єм для камери та спеціальний вивід з 40 пінами для інших приладів [50].

Message Queuing Telemetry Transport (MQTT) – це мережевий протокол, який використовується для спілкування між розумними пристроями, шляхом передачі повідомлень про свій стан. Для цього використовується Wi-Fi мережа та центральний пристрій під назвою брокер, який отримує повідомлення від різних пристроїв та зберігає їх [51].

1-Wire – це інтерфейс, який дозволяє створювати мережу з централізованим керуванням та шинною топологією, тобто один ведучий та багато ведених. Для здійснення зв'язку використовується два дроти: один для живлення і даних, а другий для заземлення. Такий інтерфейс зазвичай використовується для передачі даних у простих пристроях, наприклад: цифрові термометри та вимірювачі параметрів зовнішнього середовища [52].

MySQL – це система управління реляційними базами даних, заснована на мові запитів Structured Query Language (SQL). Вона використовується для різноманітних задач, включаючи зберігання даних, електронної комерції та ведення журналів [53].

У системі присутні два типи пристроїв: датчики та актуатори. Для підключення до мікрокомп'ютера Orange Pi датчики використовують дві технології: Wi-Fi та BLE. Пристрої, що використовують Wi-Fi працюють за технологією MQTT та використовують Wi-Fi мікроконтролер ESP32 для передачі даних. Пристрої BLE працюють у пасивному режимі, що дозволяє використовувати батареї для живлення. У системі присутні датчики двох типів: аналогові та цифрові.

Використовувана система містить наступні цифрові датчики: датчик відкриття дверей, датчик температури та вологості, датчик протікання води. Датчик зовнішньої температури використовує цифровий протокол передачі даних 1-Wire. Результатом роботи датчика зовнішньої температури є цифрові дані – число. Аналоговим датчиком є детектор газу та результатом його роботи є аналоговий сигнал (0-3.3В). Перелік всіх датчиків та актуаторів, їх типи (аналогові або цифрові) і протокол передачі даних наведено у табл. 3.1.

Таблиця 3.1 – Опис всіх пристроїв використовуваної системи

Пристрій	Тип сигналу	Технологія передачі даних	Протокол передачі даних
Датчик газу	Аналоговий	Wi-Fi	MQTT
Датчик двері	Цифровий	Wi-Fi	MQTT
Датчик протікання	Цифровий	Wi-Fi	MQTT
Датчик зовнішньої температури	Цифровий	Wi-Fi	1-Wire
Датчик температури та вологості	Цифровий	Bluetooth	–
Актuator витяжки	Цифровий	Wi-Fi	MQTT
Актuator LED	Цифровий	Wi-Fi	MQTT

Дані з усіх датчиків поступають у брокер MQTT. Програмне забезпечення OpenHAB є клієнтом, що зчитує дані у брокері MQTT та виконує зберігання даних до бази даних MySQL. Актuatorами (виконавчими механізмами) є реле, що керуються командами MQTT (ON/OFF). Всі датчики та актуатори працюють у ізольованій Wi-Fi мережі.

Архітектура пристрою моніторингу датчиків обліку енергоспоживання зображена на рисунку 3.3.

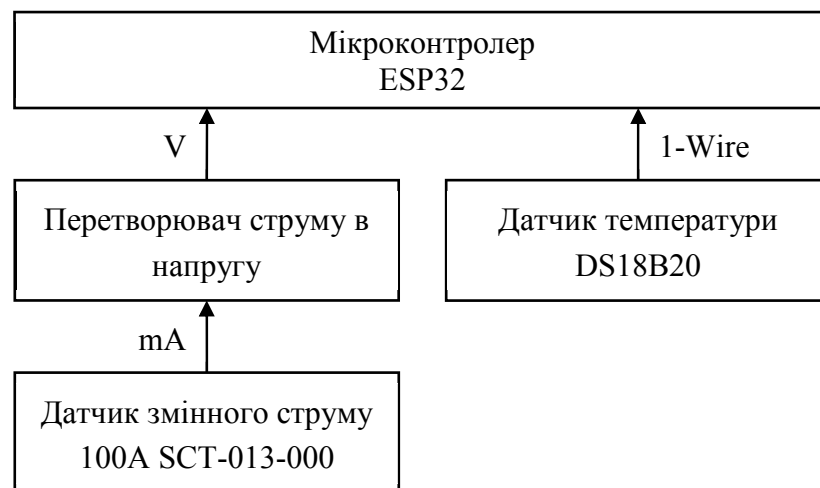


Рисунок 3.3 – Архітектура пристрою моніторингу енергоспоживання

Пристрій моніторингу складається з Wi-Fi мікроконтролера ESP32, який має два ядра для обчислень та підтримує наступні протоколи для передачі даних: Wi-Fi, Bluetooth та BLE. До мікроконтролера підключено високоточний датчик температури ввідного автомата DS18B20 та неінвазивний датчик змінного струму 100A через плату-перетворювач струму в пропорційну напругу. DS18B20 – це повноцінний цифровий термометр, який здатний вимірювати температуру в діапазоні від -55 °C до +125 °C з програмованою точністю 9-12 біт.

Пристрій моніторингу передає отримані дані з датчиків в сервер-брокер, де дані відправляються в індивідуальний топік, використовуючи протокол MQTT. Під топіком розуміється символічна ієрархічна структура типу «Level/Sensor/Data», що визначає формат повідомлень для пристроїв. Платформа OpenHAB в синхронному режимі отримує дані з брокера MQTT для відображення в інтерфейсі користувача та збереження в базу даних.

Система моніторингу енергоспоживання була підключена до реального розподільного щита. Розподільний щит містить наступні пристрої: ввідний автоматичний вимикач, пристрій захисного відключення, реле контролю напруги, автоматичні вимикачі ліній житлового приміщення.

На лініях, що відходять та приєднані стаціонарно можуть бути підключені в певні моменти часу наступні типи приладів: лампи освітлення, стаціонарна побутова техніка (холодильник, бойлер або котел, пральна машина), портативна побутова техніка (фен, пилосос, ноутбук). Такі прилади можуть мати статичну періодичність автоматичного вмикання і вимикання (наприклад, холодильник), а так само тимчасову періодичність ручного вмикання (наприклад, щоденне вмикання світла в темний час доби).

Використовуючи пристрій моніторингу датчиків обліку енергоспоживання, були отримані наступні дані: температура ввідного автомата, споживаний струм та сумарна потужність споживання приладами.

3.2 Аналіз вхідних даних

Вхідною інформацією є використання даних з датчиків і актуаторів, що були накопичені впродовж деякого часу платформою OpenHAB та збережені у базі даних MySQL. Система OpenHAB може працювати з наступними видами даних: рядки, числові значення, GPS координати, зображення, дата, час, колір, статус (вимкнено або увімкнено). Опис всіх накопичених даних наведено у табл. 3.2.

Таблиця 3.2 – Опис збережених даних у досліджуваній системі

№ датчика	Вид даних	Опис	Період накопичення
Item1	Число	Температура в кімнаті, °C	28.08.20 – 30.12.20
Item2	Число	Температура у ванній кімнаті, °C	28.08.20 – 30.12.20
Item3	Число	Вологість у ванній кімнаті, %	28.08.20 – 30.12.20
Item4	Число	Вологість в кімнаті, %	28.08.20 – 30.12.20
Item5	Час	Будильник	28.08.20 – 30.12.20
Item12	Статус	Кухонна витяжка	29.08.20 – 30.12.20
Item13	Статус	Підсвітка стільниці на кухні	29.08.20 – 30.12.20
Item14	Статус	Підсвітка телевізора	29.08.20 – 30.12.20
Item15	Статус	Підсвітка штор	29.08.20 – 30.12.20
Item16	Статус	Вхідні двері	13.09.20 – 30.12.20
Item17	Число	Температура ввідного автомата в електричному щиті, °C	16.11.20 – 30.12.20
Item19	Число	Сила струму, A	25.12.20 – 30.12.20
Item20	Число	Потужність струму, W	25.12.20 – 30.12.20

Система зберігає дані з датчиків температури та вологості кожні п'ять хвилин. Дані з датчиків обліку енергоспоживання (температура ввідного

автомата в електричному щиті, споживаний струм та сумарна потужність струму) зберігаються кожну хвилину. Дані статусів використання приладів і актуаторів записуються в базу даних в момент їх вмикання або вимикання. Всі збережені дані мають часову мітку та певний тип даних в залежності від опитуваного обладнання (рис. 3.4).

Date	Value
30-12-20 09:40	0.42
30-12-20 09:45	1.06
30-12-20 09:50	8.74
30-12-20 09:55	1.1
30-12-20 10:00	9.3
30-12-20 10:05	9.25
30-12-20 10:10	1.37
30-12-20 10:15	1.23
30-12-20 10:20	1.19
30-12-20 10:25	1.19
30-12-20 10:40	1.64
30-12-20 10:50	1.35

Рисунок 3.4 – Приклад зберігання даних про стан елемента з часовою міткою

Таким чином отримуючи та зберігаючи дані, користувач може переглянути графіки (рис. 3.5) та проаналізувати результати вимірювань різних датчиків, вмикання і вимикання певних приладів. Також накопичені дані можна перетворювати на корисну інформацію у реальному часі, що допоможе спрогнозувати та швидко відреагувати на майбутні ситуації, а також дати корисні поради користувачу.

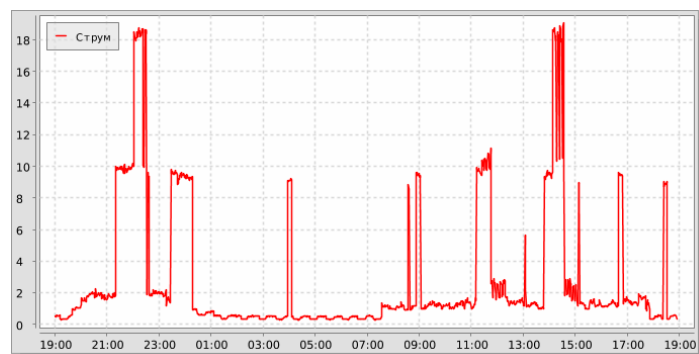


Рисунок 3.5 – Графік щоденного споживання електроенергії

3.3 Використання Orange для пошуку асоціативних правил та застосування алгоритмів машинного навчання

3.3.1 Обчислювальний експеримент 1

У першому експерименті застосуємо алгоритм пошуку асоціативних правил. Вибірка має 8 атрибутів та включає близько 30 тис. записів, що охоплює період від 29 серпня до 25 грудня 2020 року (майже 4 місяці). Опис атрибутів для цього набору даних подано у табл. 3.3.

Таблиця 3.3 – Вибірка для першого експерименту

Назва атрибуту	Вид даних	Опис
Room t	Число	Температура в кімнаті, °C
Room %	Число	Вологість в кімнаті, %
Bathroom t	Число	Температура у ванній кімнаті, °C
Bathroom %	Число	Вологість у ванній кімнаті, %
Kitchen Fan	Статус (ON / OFF)	Кухонна витяжка
LED Kitchen	Статус (ON / OFF)	Підсвітка стільниці на кухні
LED TV	Статус (ON / OFF)	Підсвітка телевізора
LED Curtains	Статус (ON / OFF)	Підсвітка штор

Для проведення експерименту в Orange була побудована схема (рис. 3.6), яка зчитує дані з файлу, дискретизує їх, тобто числові значення розбиває на інтервали та застосовує алгоритм пошуку асоціативних правил.



Рисунок 3.6 – Схема в Orange для пошуку асоціативних правил

Результат завантаження вибірки у програмний застосунок Orange зображено на рисунку 3.7.

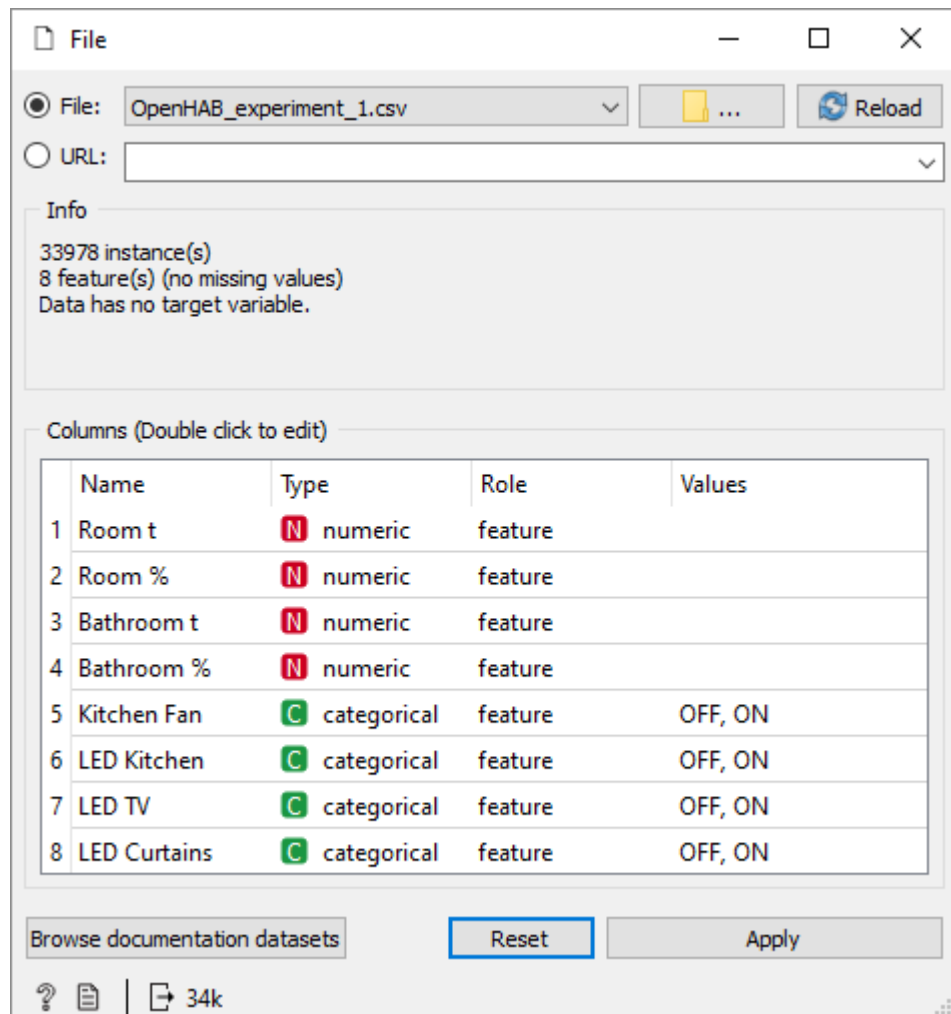


Рисунок 3.7 – Завантаження вибірки для першого експерименту в Orange

Після того, як алгоритм закінчив роботу, відкрилося вікно з налаштуваннями та знайденими асоціативними правилами (рис. 3.8).

Функція «Minimal support» може приймати значення від 0% до 100%. Ця величина характеризує якість асоціативного правила, тобто скільки раз воно зустрічається у всьому наборі.

Функція «Minimal confidence» може приймати значення від 0% до 100%. Ця величина характеризує ймовірність того, що поява одної події спричиняє за собою появу другої події, тобто наскільки часто спрацьовує це правило.

Функція «Max. number of rules» дозволяє обмежити максимальну кількість пошуку асоціативних правил.

У розділі «Filter rules» можливо встановити мінімальне та максимальне значення, яке буде знаходити тільки ті правила, які мають встановлену кількість умов та/або результатів.

*** Association Rules

Info
Number of rules: 567
Filtered rules: 567
Selected rules: 0
Selected examples: 0

Find association rules
Minimal support: 10%
Minimal confidence: 70%
Max. number of rules: 10000
☐ Induce classification (itemset → class) rules

Filter rules
Antecedent
Contains:
Min. items: 2 Max. items: 999
Consequent
Contains:
Min. items: 1 Max. items: 999
☒ Apply these filters in search
☒ Send Selection Automatically

Supp	Conf	Covr	Strg	Lift	Levr	Antecedent	Consequent
0.667	0.922	0.724	1.172	1.086	0.053	Kitchen Fan=OFF, Kitchen Light=OFF	→ Curtains Light=OFF
0.667	0.998	0.668	1.481	1.008	0.005	Kitchen Light=OFF, Curtains Light=OFF	→ Kitchen Fan=OFF
0.667	0.792	0.841	0.862	1.092	0.056	Kitchen Fan=OFF, Curtains Light=OFF	→ Kitchen Light=OFF
0.662	0.926	0.715	1.186	1.092	0.056	Kitchen Fan=OFF, TV Light=OFF	→ Curtains Light=OFF
0.662	0.992	0.668	1.483	1.002	0.002	TV Light=OFF, Curtains Light=OFF	→ Kitchen Fan=OFF
0.662	0.787	0.841	0.859	1.089	0.054	Kitchen Fan=OFF, Curtains Light=OFF	→ TV Light=OFF
0.604	0.998	0.606	1.634	1.008	0.005	Kitchen Light=OFF, TV Light=OFF	→ Kitchen Fan=OFF
0.604	0.845	0.715	1.015	1.165	0.086	Kitchen Fan=OFF, TV Light=OFF	→ Kitchen Light=OFF
0.604	0.835	0.724	0.999	1.156	0.082	Kitchen Fan=OFF, Kitchen Light=OFF	→ TV Light=OFF
0.579	0.956	0.606	1.400	1.128	0.066	Kitchen Light=OFF, TV Light=OFF	→ Curtains Light=OFF
0.579	0.868	0.668	1.087	1.196	0.095	TV Light=OFF, Curtains Light=OFF	→ Kitchen Light=OFF
0.579	0.867	0.668	1.081	1.200	0.096	Kitchen Light=OFF, Curtains Light=OFF	→ TV Light=OFF
0.578	0.956	0.604	1.403	1.127	0.065	Kitchen Fan=OFF, Kitchen Light=OFF, TV Light=OFF	→ Curtains Light=OFF
0.578	0.954	0.606	1.389	1.134	0.068	Kitchen Light=OFF, TV Light=OFF	→ Kitchen Fan=OFF, Curtains Light=OFF
0.578	0.808	0.715	0.935	1.210	0.100	Kitchen Fan=OFF, TV Light=OFF	→ Kitchen Light=OFF, Curtains Light=OFF
0.578	0.799	0.724	0.923	1.197	0.095	Kitchen Fan=OFF, Kitchen Light=OFF	→ TV Light=OFF, Curtains Light=OFF
0.578	0.998	0.579	1.709	1.008	0.004	Kitchen Light=OFF, TV Light=OFF, Curtains Light=OFF	→ Kitchen Fan=OFF
0.578	0.873	0.662	1.095	1.203	0.097	Kitchen Fan=OFF, TV Light=OFF, Curtains Light=OFF	→ Kitchen Light=OFF
0.578	0.866	0.668	1.084	1.197	0.095	TV Light=OFF, Curtains Light=OFF	→ Kitchen Fan=OFF, Kitchen Light=OFF
0.578	0.867	0.667	1.084	1.200	0.096	Kitchen Fan=OFF, Kitchen Light=OFF, Curtains Light=OFF	→ TV Light=OFF
0.578	0.865	0.668	1.070	1.210	0.100	Kitchen Light=OFF, Curtains Light=OFF	→ Kitchen Fan=OFF, TV Light=OFF
0.229	0.807	0.284	2.989	0.951	-0.012	Bath-t=21 - 22, Kitchen Fan=OFF	→ Curtains Light=OFF
0.229	0.996	0.230	4.305	1.006	0.001	Bath-t=21 - 22, Curtains Light=OFF	→ Kitchen Fan=OFF
0.207	0.878	0.236	3.599	1.035	0.007	Room-t=21 - 22, Kitchen Fan=OFF	→ Curtains Light=OFF
0.207	0.996	0.208	4.765	1.006	0.001	Room-t=21 - 22, Curtains Light=OFF	→ Kitchen Fan=OFF

Рисунок 3.8 – Вікно налаштувань асоціативних правил

В налаштуваннях встановимо наступні значення: мінімальна підтримка = 50%; мінімальна достовірність = 75%; максимальна кількість правил = 10000; мінімальна кількість умов = 2. Отримані асоціативні правила зображені на рисунку 3.9.

Можна побачити, що більшість правил має дуже велику достовірність. Це означає, що ці правила дуже часто зустрічаються у вибірці. Проаналізувавши результати, можна побачити, що коли вимкнено два або більше прилади, то будуть вимкнені все інші прилади. Можна припустити, що мешканець будинку рідко користується приладами або рідко буває вдома.

Supp	Conf	Covr	Strg	Lift	Levr	Antecedent	Consequent
0.667	0.922	0.724	1.172	1.086	0.053	Kitchen Fan=OFF, Kitchen Light=OFF	→ Curtains Light=OFF
0.667	0.998	0.668	1.481	1.008	0.005	Kitchen Light=OFF, Curtains Light=OFF	→ Kitchen Fan=OFF
0.667	0.792	0.841	0.862	1.092	0.056	Kitchen Fan=OFF, Curtains Light=OFF	→ Kitchen Light=OFF
0.662	0.926	0.715	1.186	1.092	0.056	Kitchen Fan=OFF, TV Light=OFF	→ Curtains Light=OFF
0.662	0.992	0.668	1.483	1.002	0.002	TV Light=OFF, Curtains Light=OFF	→ Kitchen Fan=OFF
0.662	0.787	0.841	0.859	1.089	0.054	Kitchen Fan=OFF, Curtains Light=OFF	→ TV Light=OFF
0.604	0.998	0.606	1.634	1.008	0.005	Kitchen Light=OFF, TV Light=OFF	→ Kitchen Fan=OFF
0.604	0.845	0.715	1.015	1.165	0.086	Kitchen Fan=OFF, TV Light=OFF	→ Kitchen Light=OFF
0.604	0.835	0.724	0.999	1.156	0.082	Kitchen Fan=OFF, Kitchen Light=OFF	→ TV Light=OFF
0.579	0.956	0.606	1.400	1.128	0.066	Kitchen Light=OFF, TV Light=OFF	→ Curtains Light=OFF
0.579	0.868	0.668	1.087	1.196	0.095	TV Light=OFF, Curtains Light=OFF	→ Kitchen Light=OFF
0.579	0.867	0.668	1.081	1.200	0.096	Kitchen Light=OFF, Curtains Light=OFF	→ TV Light=OFF
0.578	0.956	0.604	1.403	1.127	0.065	Kitchen Fan=OFF, Kitchen Light=OFF, TV Light=OFF	→ Curtains Light=OFF
0.578	0.954	0.606	1.389	1.134	0.068	Kitchen Light=OFF, TV Light=OFF	→ Kitchen Fan=OFF, Curtains Light=OFF
0.578	0.808	0.715	0.935	1.210	0.100	Kitchen Fan=OFF, TV Light=OFF	→ Kitchen Light=OFF, Curtains Light=OFF
0.578	0.799	0.724	0.923	1.197	0.095	Kitchen Fan=OFF, Kitchen Light=OFF	→ TV Light=OFF, Curtains Light=OFF
0.578	0.998	0.579	1.709	1.008	0.004	Kitchen Light=OFF, TV Light=OFF, Curtains Light=OFF	→ Kitchen Fan=OFF
0.578	0.873	0.662	1.095	1.203	0.097	Kitchen Fan=OFF, TV Light=OFF, Curtains Light=OFF	→ Kitchen Light=OFF
0.578	0.866	0.668	1.084	1.197	0.095	TV Light=OFF, Curtains Light=OFF	→ Kitchen Fan=OFF, Kitchen Light=OFF
0.578	0.867	0.667	1.084	1.200	0.096	Kitchen Fan=OFF, Kitchen Light=OFF, Curtains Light=OFF	→ TV Light=OFF
0.578	0.865	0.668	1.070	1.210	0.100	Kitchen Light=OFF, Curtains Light=OFF	→ Kitchen Fan=OFF, TV Light=OFF

Рисунок 3.9 – Результати пошуку асоціативних правил

Для отримання нових правил, у налаштуваннях встановимо наступні значення: мінімальна підтримка = 0.5%; мінімальна достовірність = 70%; максимальна кількість правил = 10000; мінімальна кількість умов = 2.

Розглянемо правило «Kitchen Fan=OFF, LED Kitchen=ON, LED Curtains=ON → LED TV=ON». Воно має дуже низьке значення підтримки (0.6%) та це означає, що асоціативний зв'язок низький. Але це правило має велику достовірність (71%). Можна припустити, що мешканець любить дивитися телевізор та вечеряти.

В налаштуваннях встановимо наступні значення: мінімальна підтримка = 9%; мінімальна достовірність = 50%; максимальна кількість правил = 10000; мінімальна кількість умов = 3.

Розглянемо правило «Kitchen Fan=OFF, LED Kitchen=ON, LED Curtains=OFF → LED TV=ON». Це правило має низьке значення підтримки (9%) та має середню достовірність (51%). Можна зробити припущення, що мешканець також любить дивитися телевізор вранці або в обід під час прийому їжі.

3.3.2 Обчислювальний експеримент 2

У другому експерименті також застосуємо алгоритм пошуку асоціативних правил, але вже для нового набору даних. Вибірка має 10 атрибутів та включає близько 1,5 тис. записів, що охоплює період від 25 грудня до 30 грудня 2020 року (5 днів). Опис атрибутів для цього набору даних подано у табл. 3.4.

Таблиця 3.4 – Вибірка для другого експерименту

Назва атрибуту	Вид даних	Опис
Room t	Число	Температура в кімнаті, °C
Room %	Число	Вологість в кімнаті, %
Bathroom t	Число	Температура у ванній кімнаті, °C
Bathroom %	Число	Вологість у ванній кімнаті, %
Kitchen Fan	Статус (ON / OFF)	Кухонна витяжка
LED Kitchen	Статус (ON / OFF)	Підсвітка стільниці на кухні
LED TV	Статус (ON / OFF)	Підсвітка телевізора
LED Curtains	Статус (ON / OFF)	Підсвітка штор
Electric t	Число	Температура ввідного автомата в електричному щиті, °C
Electric A	Число	Сила струму, A

Для цього експерименту була використана схема, що і у попередньому експерименті, яка завантажує дані з файлу, дискретизує їх та застосовує алгоритм пошуку асоціативних правил.

Результат завантаження вибірки для другого експерименту у програмний застосунок Orange зображено на рисунку 3.10.

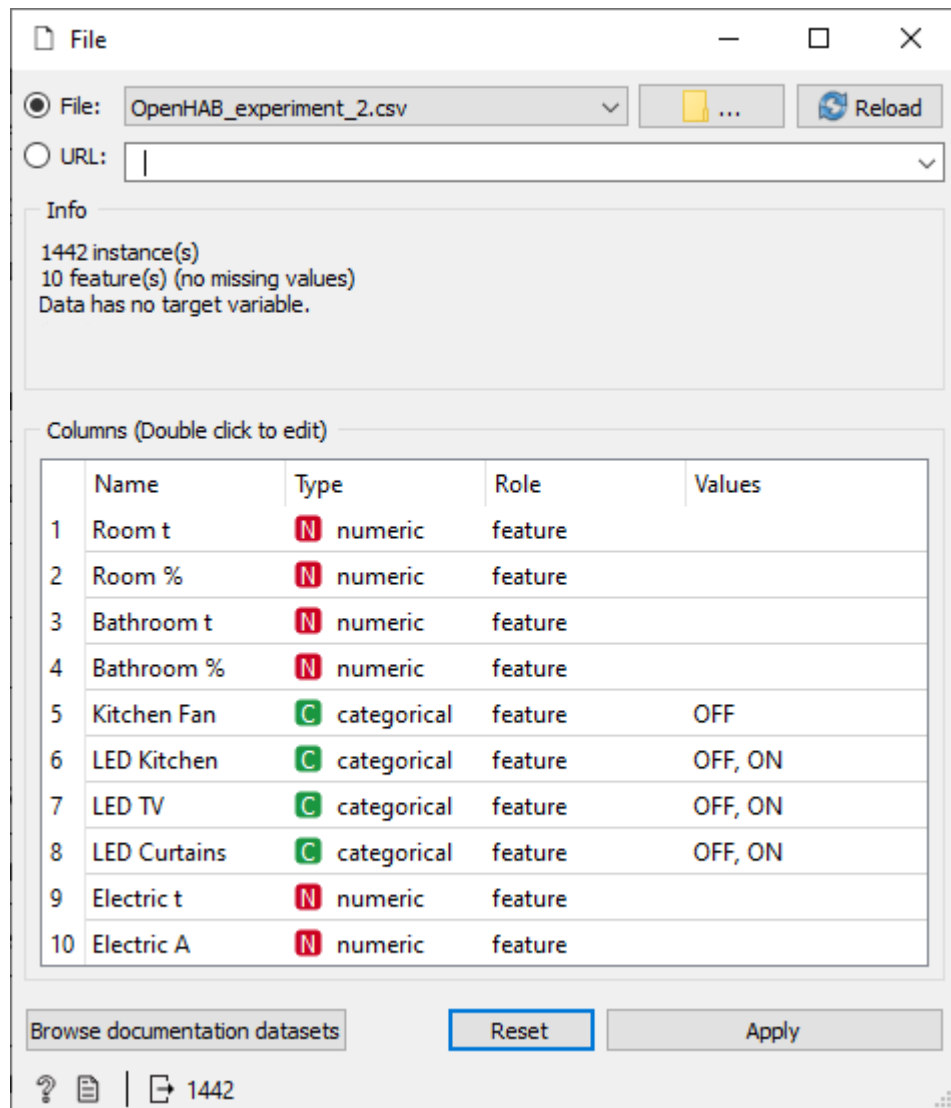


Рисунок 3.10 – Завантаження вибірки для другого експерименту в Orange

Для отримання асоціативних правил, у налаштуваннях встановимо наступні значення: мінімальна підтримка = 10%; мінімальна достовірність = 80%; максимальна кількість правил = 10000; мінімальна кількість умов = 3. Отримані асоціативні правила зображені на рисунку 3.11.

Можна побачити, що всі вони мають низьку підтримку (10%) та це означає, що вони є сумнівними та краще не використовувати. Але вони мають досить велику достовірність (приблизно 90%) та це означає, що ці правила дуже часто зустрічаються у вибірці. Якщо проаналізувати ці правила, то можна зрозуміти, що вони не мають ніякого сенсу.

Supp	Conf	Covr	Strg	Lift	Levr	Antecedent	Consequent
0.117	0.824	0.141	6.525	0.892	-0.014	Bath-t=21 - 22, Kitchen Light=ON, Electric-A=1 - 2	TV Light=OFF
0.117	0.949	0.123	6.480	1.193	0.019	Kitchen Light=ON, TV Light=OFF, Electric-A=1 - 2	Bath-t=21 - 22
0.115	0.922	0.125	2.039	3.624	0.083	Bath-t=21 - 22, TV Light=OFF, Curtains Light=ON	Kitchen Light=ON
0.115	0.892	0.129	7.156	0.967	-0.004	Bath-t=21 - 22, Kitchen Light=ON, Curtains Light=ON	TV Light=OFF
0.115	0.912	0.126	6.302	1.147	0.015	Kitchen Light=ON, TV Light=OFF, Curtains Light=ON	Bath-t=21 - 22

Рисунок 3.11 – Результати пошуку асоціативних правил

Для отримання нових правил, у налаштуваннях встановимо наступні значення: мінімальна підтримка = 5%; мінімальна достовірність = 70%; максимальна кількість правил = 10000; мінімальна кількість умов = 2.

Розглянемо правило «Room t=22-23, Bathroom t=21-22, LED Kitchen=ON → Electric t=30-32». Воно має низьке значення підтримки (0.8%) та має низьку достовірність (0.7%). Це означає, що це правило є сумнівним та рідко зустрічається. Проаналізувавши його, можна помітити, що температура ввідного автомата в щиті є великою тоді, коли температура в кімнаті в межах 22-23 °C, а температура в ванній кімнаті в межах 21-22 °C та увімкнена підсвітка стільниці на кухні. Можливо температура автомата в щиті збільшується, тому що мешканець щось готує на кухні або вмикає прилади, які створюють велике навантаження на електромережу. Але це є припущенням, бо недостатньо вхідних даних, щоб робити висновки.

В налаштуваннях встановимо наступні значення: мінімальна підтримка = 50%; мінімальна достовірність = 70%; максимальна кількість правил = 10000; мінімальна кількість умов = 3.

Розглянемо два правила: «LED Kitchen=OFF, LED TV=OFF, LED Curtains=OFF → Electric t=22-24» та «LED Kitchen=OFF, LED TV=OFF, LED Curtains=OFF → Electric A=0-1». Обидва правила мають середню підтримку (50%) та велику достовірність (70%). Це означає, що правило добре виражає асоціативний зв'язок між умовою і наслідком та часто зустрічається у вибірці. Проаналізувавши отримані правила, можна зрозуміти, що коли більшість приладів вимкнено, то температура автомата в щиті є нормальною, а

навантаження на мережу є мінімальним. Але це правило не дало нових знань, тому ще це зрозумілий для всіх факт.

3.3.3 Обчислювальний експеримент 3

Для третього експерименту вибірка була розбита на частини доби, а саме: сніданок (6:00 – 10:00), денний час (10:00 – 17:00), вечір (17:00 – 23:00) та ніч (23:00 – 6:00). Для температурних показників, показників вологості та струму було обчислене середнє значення за кожен період. Значення, які зберігають статус роботи електроприлада було обчислено загальний час користування. Вибірка має 11 атрибутів та охоплює період від 25 грудня до 30 грудня 2020 року (5 днів). Опис атрибутів для цього набору даних подано у табл. 3.5.

Таблиця 3.5 – Вибірка для другого експерименту

Назва атрибуту	Вид даних	Опис
DayPart	Рядом	Частина доби
Room t	Число	Температура в кімнаті, °C
Room %	Число	Вологість в кімнаті, %
Bathroom t	Число	Температура у ванній кімнаті, °C
Bathroom %	Число	Вологість у ванній кімнаті, %
Kitchen Fan	Статус (ON / OFF)	Кухонна витяжка
LED Kitchen	Статус (ON / OFF)	Підсвітка стільниці на кухні
LED TV	Статус (ON / OFF)	Підсвітка телевізора
LED Curtains	Статус (ON / OFF)	Підсвітка штор
Electric t	Число	Температура ввідного автомата в електричному щиті, °C
Electric A	Число	Сила струму, A

Результат завантаження вибірки для третього експерименту у програмний застосунок Orange зображено на рисунку 3.12.

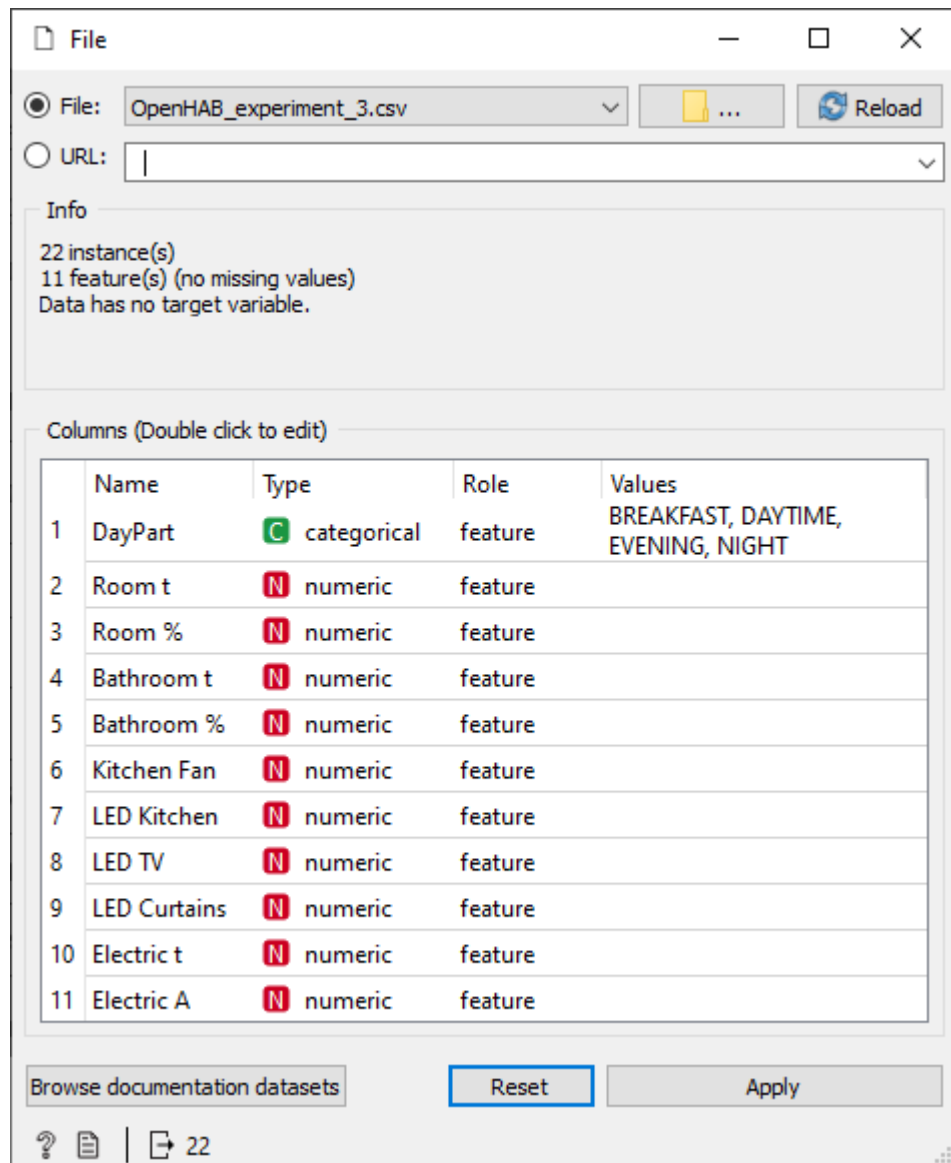


Рисунок 3.12 – Завантаження вибірки для третього експерименту в Orange

Для отримання асоціативних правил, у налаштуваннях встановимо наступні значення: мінімальна підтримка = 20%; мінімальна достовірність = 80%; максимальна кількість правил = 10000; мінімальна кількість умов = 2.

Розглянемо правило «DayPart=BREAKFAST, Room t=21-22 → LED TV=0-1». Це правило має велику достовірність (83%). Проаналізувавши це правило, можна зробити висновок, що мешканець дуже рідко дивиться телевізор вранці.

Для отримання нових правил, у налаштуваннях встановимо наступні значення: мінімальна підтримка = 4%; мінімальна достовірність = 70%; максимальна кількість правил = 10000; мінімальна кількість умов = 2.

Розглянемо правило «DayPart=EVENING, Bathroom t=21-22, LED TV=60-90, LED Curtains=60-90 → Bathroom %=65-70». Це правило має 100% достовірність і це означає, що воно завжди виконується. Проаналізувавши це правило, можна побачити, що ввечері, якщо мешканець дивиться телевизор та увімкнена підсвітка штор, то у ванній кімнаті підвищується вологість. Можна припустити, що мешканець в цей час приймає ванну або душ.

В налаштуваннях встановимо наступні значення: мінімальна підтримка = 4%; мінімальна достовірність = 75%; максимальна кількість правил = 10000; мінімальна кількість умов = 3.

Розглянемо правило «DayPart=DAYTIME, LED Kitchen=30-60, Electric A=1-2 → LED Curtains=30-60». Це правило також має 100% достовірність. Проаналізувавши це правило, можна побачити, що вдень мешканець проводить на кухні близько однієї години, використовує якісь прилади, можливо комп'ютер та вмикає підсвітку штор, яка вдень є тільки зайвою.

3.3.4 Обчислювальний експеримент 4

Четвертий експеримент полягає у використанні алгоритмів машинного навчання, які зможуть у майбутньому прогнозувати певні значення. Для цього експерименту була використана та ж сама вибірка, що і у другому експерименті (табл. 3.4). Значення струму було обрано як залежну змінну, тобто це значення навчена модель зможе прогнозувати.

10% записів було вилучено з набору даних, які не брали участь у навчальній вибірці, але використовувалися в процесі тестування результатів процесу навчання різних алгоритмів.

Для проведення експерименту за допомогою Orange була побудована схема (рис. 3.13), де спочатку завантажувється вибірка для навчання моделей, після чого всі моделі перевіряються на тестовій вибірці, для оцінки точності прогнозування.

Було обрано всі доступні алгоритми машинного навчання в Orange, а саме: Tree, kNN, Random Forest, SVM, AdaBoost, Linear Regression, Neural Network, Stochastic Gradient Descent.

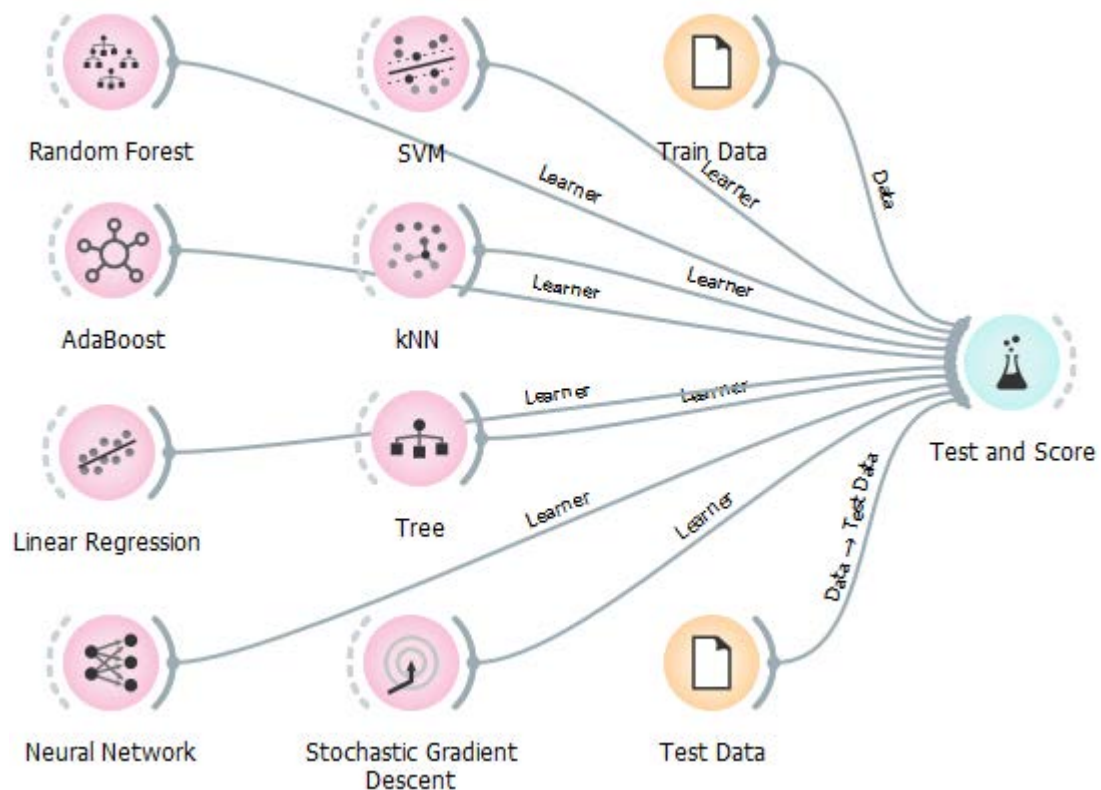


Рисунок 3.13 – Схема в Orange для машинного навчання

Результат завантаження навчальної вибірки для четвертого експерименту у програмний застосунок Orange зображено на рисунку 3.14.

Після закінчення навчання всіх алгоритмів, було перевірено їх точність, тобто розрахована середня квадратична помилка, середньоквадратична помилка, середня абсолютна помилка та коефіцієнт детермінації. За допомогою цих показників можна оцінити якість кожної моделі. Чим менше значення середньої абсолютної помилки, тим краще

навчена модель, і навпаки, чим ближче значення коефіцієнта детермінації до 1, то продуктивність такої моделі є кращою.

На рисунку 3.15 зображено точність прогнозування для кожної моделі. Таким чином, можна побачити, що найкращий результат дає алгоритм AdaBoost, бо він має найменше значення середньої абсолютної помилки (59%), що не є гарним результатом для прогнозування. Наступними йдуть алгоритми Random Forest та kNN, які працюють набагато гірше. Отже, всі ці алгоритми мають незадовільні результати та не підходять для прогнозування.

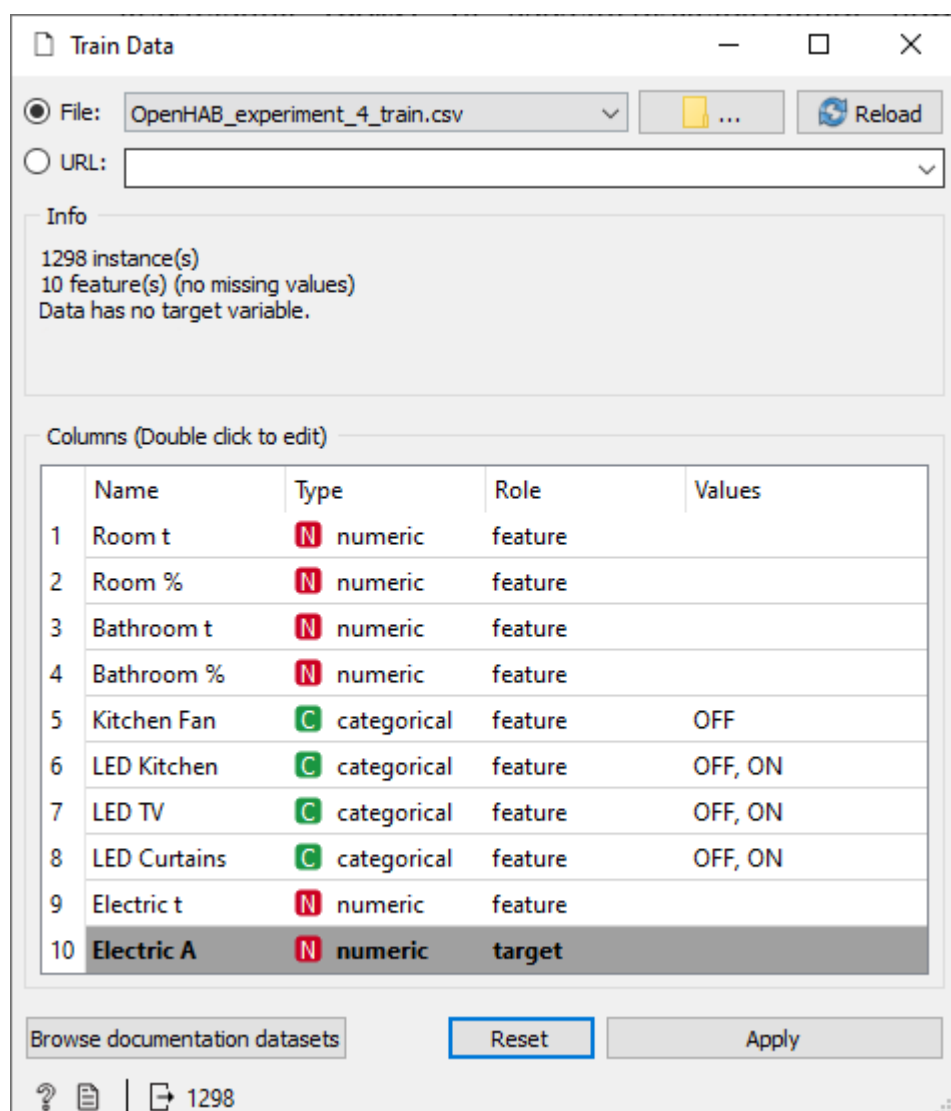


Рисунок 3.14 – Завантаження вибірки для четвертого експерименту в Orange

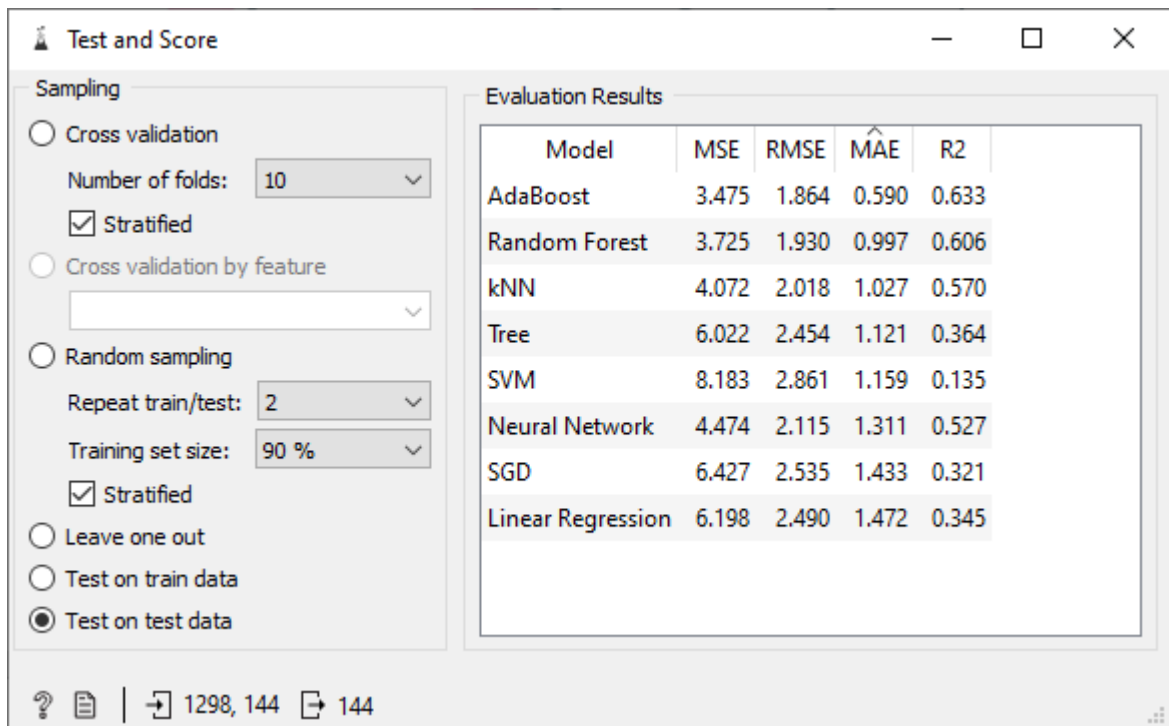


Рисунок 3.15 – Порівняння точності прогнозування кожної навченої моделі

3.4 Висновки до розділу

Отже, після проведення експериментів з використанням алгоритму пошуку асоціативних правил можна зробити висновок, що більшість правил не мають сенсу або вже є відомими, тобто вони є тривіальними. В даному випадку асоціативні правила можуть дати інформацію лише про те, які звички та розпорядок дня має мешканець будинку. Також важливо відзначити, що OpenNAV накопичує недостатньо даних, які можна перетворити на корисні знання. Для цього потрібно зберігати дані про всі електроприлади, що використовуються у будинку, а не тільки деякі з них, як у досліджуваній системі. Тому використання пошуку асоціативних правил не дозволить сформувати рекомендації щодо економії електроенергії або безпечного користування електроприладами. Підводячи підсумки експерименту з алгоритмами машинного навчання, було перевірено багато алгоритмів та найкращим з них виявився AdaBoost. Але всі ці алгоритми мають незадовільні результати та не підходять для прогнозування.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Проєктування програмного забезпечення

4.1.1 Аналіз вимог

Згідно технічного завдання (додаток А) були проаналізовані функціональні та нефункціональні вимоги розроблюваного програмного забезпечення, яке має бути реалізовано у вигляді десктопного застосунку.

Для моніторингу показників роботи електрощитового обладнання необхідно забезпечити наявність датчиків, які будуть підключені до платформи OpenHAB, а саме: датчик напруги, струму та температури ввідного автомата в електричному щиті (розд. 3.1, рис. 3.1). Для отримання інформації з датчиків потрібно реалізувати взаємодію з центральним контролером OpenHAB, використовуючи REST API.

Для виведення списку увімкнених електроприладів та їхнього споживання необхідно мати список усіх приладів, що використовуються, та за допомогою алгоритму повного перебору потрібно знайти таку комбінацію, де сума їхнього навантаження буде дорівнювати загальному навантаженню в мережі. Так як до платформи OpenHAB не було підключено всі прилади (розд. 3.1, рис. 3.1), саме такий алгоритм дозволить з високою вірогідністю розпізнати всі увімкнені електроприлади у будинку.

Для збереження списку електроприладів, що знаходяться у будинку, буде використана база даних MySQL (розд. 3.1, рис. 3.1), яка входить до складу СДА. Використання іншої бази даних не є доцільним, тому що набагато зручніше зберігати всі дані, що стосуються СДА в єдиному сховищі. Кожен електроприлад включає наступні дані: назва, навантаження (А), назва актуатора в системі OpenHAB (якщо є).

Розроблюване програмне забезпечення повинно мати гнучкі налаштування всіх функцій. Таким чином користувач має змогу вказати параметри для наступних полів: хост, порт та пароль для з'єднання з центральним контролером та базою даних MySQL; інтервал оновлення

моніторингу датчиків; назви всіх датчиків та актуаторів в OpenHAB; значення мінімального та номінального струму; поріг попередження.

Всі налаштування програми повинні зберігатися у реєстрі Windows (для ОС сімейства Windows) та у файлі конфігурації системи, що знаходиться в директорії користувача (для ОС сімейства Linux та macOS).

У разі виникнення помилок під час виконання програми, користувач повинен отримувати повідомлення у вигляді діалогових вікон, які мають містити опис проблеми, що виникла, а також код помилки.

Оповіщення власника будинку про небезпечні зміни у показниках, що відслідковуються, повинні бути реалізовані трьома статусами:

- «Все добре», коли електромережа в порядку та немає ніяких загроз;
- «Увага», коли навантаження в електромережі перевищує поріг попередження, але ще не перевищує номінальне значення струму;
- «Небезпека», коли навантаження в електромережі перевищує номінальне значення струму та існує загроза виникнення пожежі.

Також оповіщення повинно включати інформацію про хибне або неможливе відключення ввідного автомата, що залежить від його температури, бо саме через зміну температури відбувається зміщення його параметрів (номінальний струм). Робоча температура ввідного автомата дорівнює +25 °C та може коливатися від +15 °C до +35 °C. Захист електромережі від перевантаження відбувається методом теплового контролю спеціального елемента – біметалічної пластини. При протіканні струму ця пластина починає нагріватися та при підвищенні температури відбувається її вигин, яка штовхає спусковий механізм ввідного автомата. Таким чином при збільшенні навколишньої температури існує ризик хибного відключення всієї електромережі, а при зниженні – ризик відмови захисту від перевантаження кабелю, що може призвести до виникненню пожежі [54].

Всі оповіщення повинні відправлятися на мобільний застосунок платформи OpenHAB у вигляді повідомлень, використовуючи REST API.

Рекомендації щодо безпечного користування електроприладами повинні надаватися разом з оповіщеннями про небезпечні зміни та включати інформацію про подальші дії користувача з електроприладами, які не слід вмикати або потрібно вимкнути для запобігання виникненню пожежі.

Графічний інтерфейс програми повинен бути простим і зручним у використанні та не повинен бути перевантажений графічними елементами. Також інтерфейс повинен відповідати сучасним ергономічним вимогам та забезпечувати зручний доступ до всіх функцій програми. Таким чином на формі повинно бути чотири вкладки:

- «Головна», де відображаються всі показники з датчиків СДА, графік навантаження мережі, а також загальний статус електромережі («Все добре», «Увага», «Небезпека»);
- «Електроприлади», де відображається список всіх увімкнених електроприладів та їхнє споживання, а також є форма для редагування всіх приладів;
- «Параметри», де знаходяться налаштування всіх функцій програмного застосунку;
- «Про програму», де вказана інформація про призначення програми та її розробника.

Розроблений застосунок повинен підтримувати українську, англійську та російську локалізацію.

4.1.2 Архітектура системи

Архітектуру платформи OpenHAB та її взаємодія з розроблюваним застосунком «OpenHAB Helper» наведено на рисунку 4.1.

OpenHAB виступає у ролі центрального контролера у СДА, який збирає дані з усіх датчиків та зберігає їх у базі даних MySQL. Управління побутовими приладами відбувається за допомогою актуаторів, які також підключені до центрального контролера.

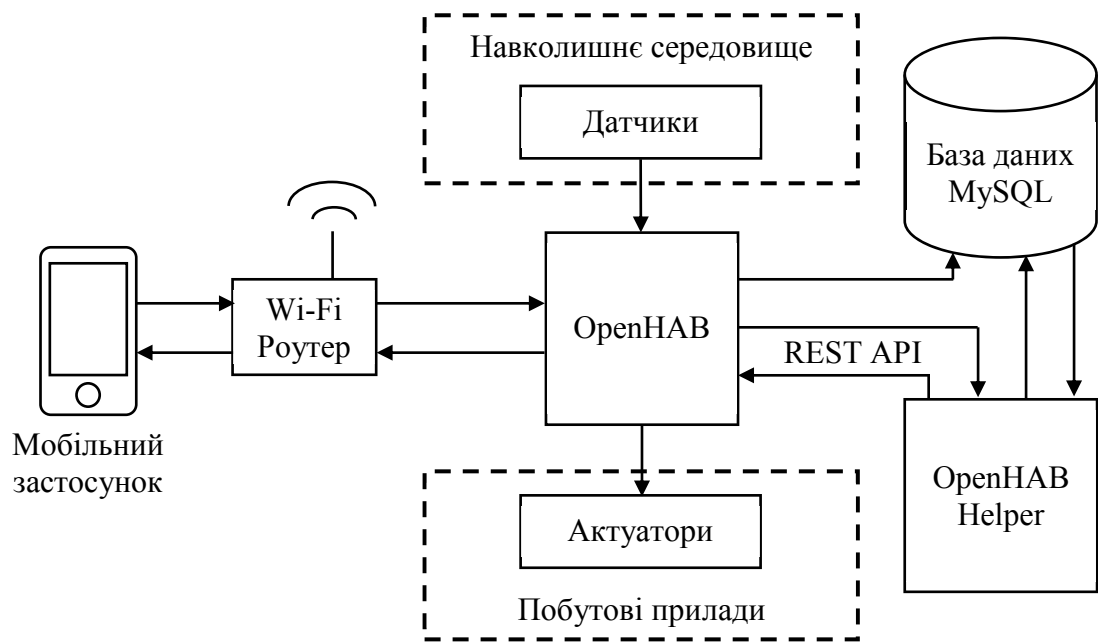


Рисунок 4.1 – Взаємодія «OpenHAB Helper» з СДА

API – це інтерфейс прикладного програмування, який включає в себе набір методів та процедур, що дозволяють отримати доступ до даних і функцій конкретного додатку, служби або операційної системи [55].

Representational State Transfer (REST) – це архітектурний стиль проєктування децентралізованих систем, який поєднує в собі клієнт-серверну архітектуру з додатковими обмеженнями, в результаті чого це призводить до підвищення продуктивності та спрощення архітектури. REST API дозволяє отримати доступ до потрібного ресурсу або змінити його стан, використовуючи Uniform Resource Identifier (URI). Під URI розуміється символічний рядок, який дозволяє ідентифікувати будь-який ресурс. Також REST може обробляти різні типи даних, наприклад: HyperText Markup Language (HTML), JavaScript Object Notation (JSON), eXtensible Markup Language (XML) або будь-який інший формат залежно від того, що запитує клієнт [56], [57].

Використання REST API дає можливість отримати доступ до більшості функцій програмного забезпечення OpenHAB. Таким чином можна переглядати всі дані, які пов'язані з датчиками або управляти поведінкою пристроїв, які підключені до платформи OpenHAB [58].

OpenHAB REST API надає такі можливості:

- отримати інформацію про підключені пристрої та дані з них;
- виконати команди для запуску подій пристроїв, що підключені (наприклад, детектори руху або камери спостереження);
- отримати інформацію про датчики, їх поточні стани, параметри або проблеми, що виникли;
- взаємодіяти з платформою OpenHAB, використовуючи інші програми написані на будь-яких мовах програмування;
- використання стороннього програмного забезпечення на мобільних телефонах для автоматизації деяких задач (наприклад, мобільний застосунок «Tasker» для відкривання дверей гаража) [58].

Розроблюване програмне забезпечення «OpenHAB Helper» взаємодіє з центральним контролером OpenHAB за допомогою REST API та отримує всі потрібні дані з датчиків, аналізує їх і у разі небезпечних змін у показниках відправляє оповіщення власнику будинку. Також «OpenHAB Helper» використовує базу даних MySQL, що входить до складу СДА, для збереження списку електроприладів, які знаходяться в будинку.

За допомогою офіційного мобільного застосунку від OpenHAB користувач може підключитися до СДА через Wi-Fi роутер для управління всіма побутовими приладами, перегляду інформації з датчиків, а також для отримання повідомлень та рекомендацій від розроблюваного програмного застосунку «OpenHAB Helper».

4.1.3 Структурна схема програмного застосунку

Розроблюване програмне забезпечення «OpenHAB Helper» складається з декількох модулів, які взаємодіють один з одним. Це дозволяє у майбутньому з легкістю модифікувати програмний застосунок та додати до нього нові функції.

Структурна схема програмного застосунку зображена на рисунку 4.2.

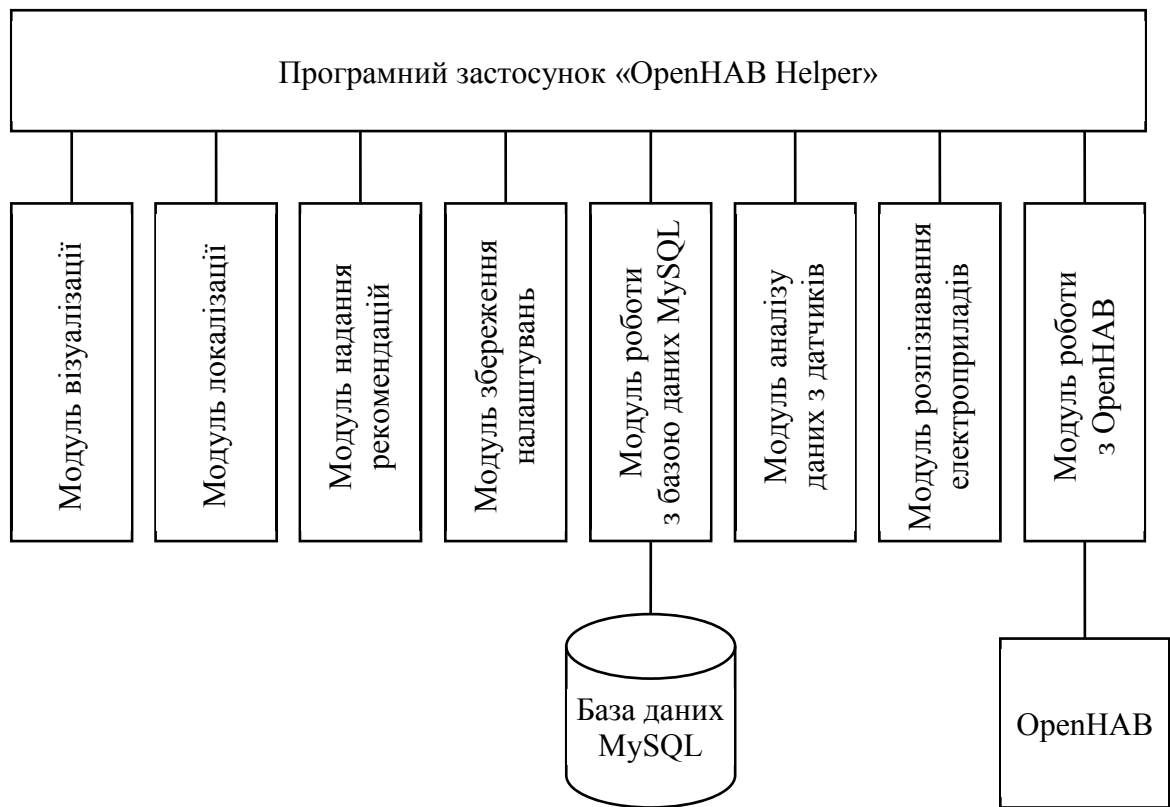


Рисунок 4.2 – Структурна схема програмного застосунку

Програмний застосунок складається з наступних модулів:

- модуль візуалізації, який призначено для реалізації діалогу між користувачем та програмою за допомогою графічного інтерфейсу, а також виведення всієї потрібної інформації;
- модуль локалізації, який надає користувачу можливість обирати зручну для нього мову графічного інтерфейсу;
- модуль надання рекомендацій, який призначено для формування рекомендацій щодо безпечного користування електроприладами для тих випадків, коли електромережа має велике навантаження та може виникнути загроза виникнення пожежі;
- модуль збереження налаштувань, який дозволяє зберігати всі параметри програми у реєстрі Windows або у файлі конфігурації системи та завантажувати їх під час запуску програмного застосунку;
- модуль роботи з OpenHAB, який призначено для взаємодії з платформою OpenHAB для отримання інформації про підключені пристрої та

дані з них, а також для відправлення повідомлень користувачу, використовуючи REST API;

- модуль аналізу даних з датчиків, який отримує інформацію з датчиків, що підключені до центрального контролера та перевіряє їх показники на те, щоб вони не перевищували допустимі значення, а також виводить загальний статус електромережі («Все добре», «Увага», «Небезпека»);

- модуль розпізнавання електроприладів, дозволяє розпізнати увімкнені електроприлади у будинку, використовуючи лише загальне навантаження в електромережі, та обчислити їхнє споживання;

- модуль роботи з базою даних MySQL, який призначено для читання та зберігання списку електроприладів у базі даних, яка входить до складу СДА.

Алгоритм роботи розроблюваного програмного застосунку «OpenHAB Helper» представлено у вигляді діаграми послідовності (рис. 4.3).

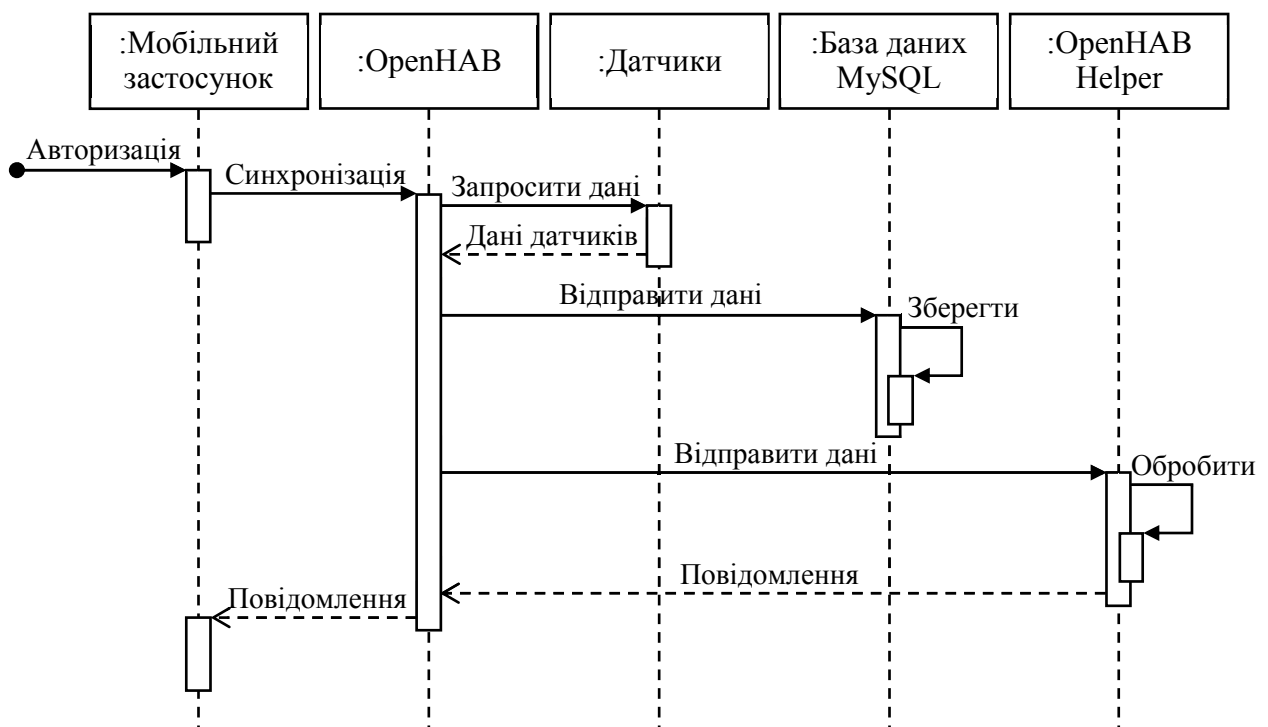


Рисунок 4.3 – Діаграма послідовності дій

За допомогою офіційного мобільного застосунку від OpenHAB користувач може авторизуватися в ньому та підключитися до СДА. Центральний контролер у свою чергу отримує дані з датчиків та зберігає їх у базі даних MySQL. Використовуючи REST API, програмний застосунок «OpenHAB Helper» отримує ці дані та обробляє їх, після чого користувачу на мобільний застосунок через OpenHAB відправляється повідомлення та рекомендації щодо безпечного користування електроприладами для запобігання виникненню небезпеки.

4.1.4 Вибір інструментарію розробки

На сьогоднішній день існує багато мов програмування, які використовуються для вирішення певних завдань, мають свої переваги та недоліки. Найбільш затребувані ті, які задовольняють потреби корпорацій, бізнесу, мобільних та інтернет-платформ.

На рисунку 4.4 зображена гістограма найпопулярніших мов програмування станом на 2021 рік.

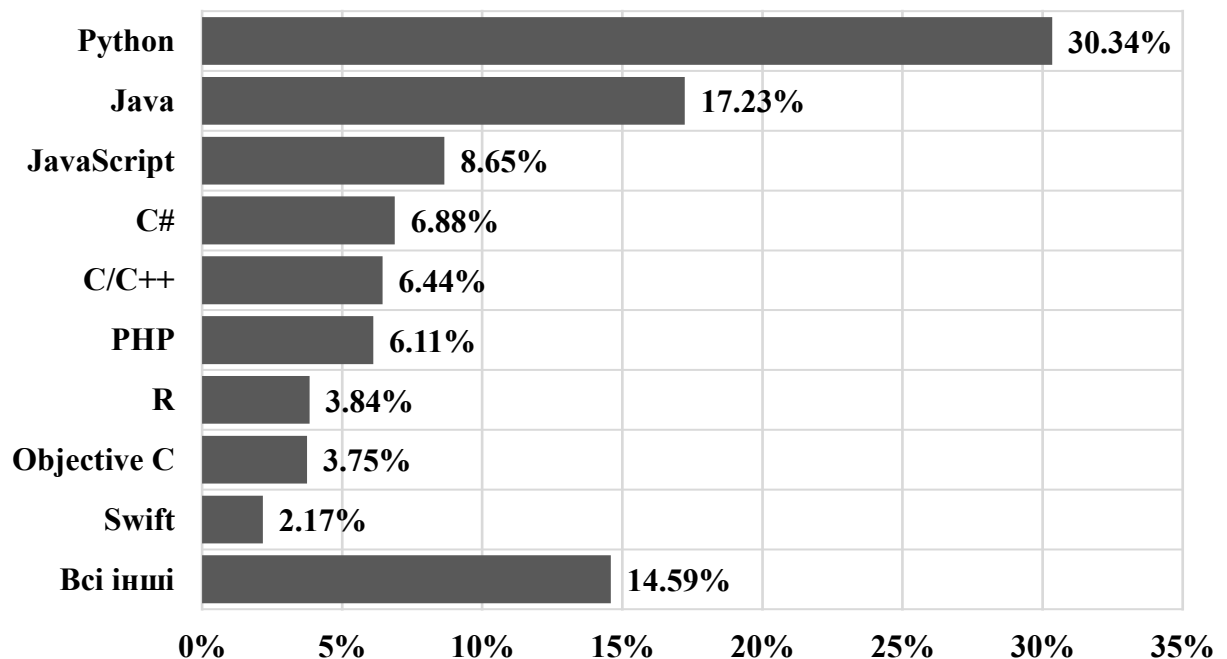


Рисунок 4.4 – Рейтинг мов програмування у 2021 році [59]

На основі дослідження джерела [59] було розглянуто чотири найпопулярніші мови програмування у 2021 році, а саме: Python, Java, JavaScript та C#.

Python – це одна з найбільш часто використовуваних на сьогоднішній день мов програмування, яку легко освоїти новачкам через її простоту та читабельність у порівнянні з іншими мовами, такими як Java, C, C++ тощо. Python є інтерпретованою і динамічно типізованою мовою та це означає, що програмісту не потрібно визначати тип даних змінних та немає необхідності в компіляції, що робить налагодження коду простіше, ніж на всіх інших мовах. Але ця особливість впливає на швидкість роботи програми. Таким чином, Python працює повільніше, ніж Java. Також Python пропонує кілька парадигм програмування, наприклад: об'єктно-орієнтоване програмування, імперативне та функціональне програмування. Крім того, ця мова є кросплатформною, тобто одна і та ж сама програма, написана на Python, може бути виконана на різних ОС, таких як Windows, Linux та macOS. Python використовується в різних областях, включаючи десктопні та веб застосунки, штучний інтелект, ігрову індустрію, фінансові послуги тощо [60].

Java – це одна з найбільш поширених і затребуваних мов програмування, використовуваних сьогодні. Вона є більш складною мовою для вивчення, але досвідчені програмісти з навичками програмування на Java користуються великим попитом. Java виступає як скомпільована мова програмування та є статично типізованою, тобто всі змінні повинні бути оголошені перед присвоєнням значень. Таким чином, програми, написані на Java, працюють швидше, ніж на Python, але повільніше у порівнянні з C++. Головним недоліком є те, що Java споживає більше оперативної пам'яті у порівнянні з C++. Процес компіляції перетворює програму в незалежний від платформи байтовий код, який потім виконується за допомогою Java Virtual Machine (JVM). Використання JVM робить мову Java незалежною від ОС. Ця мова широко використовується в розробці десктопних та веб застосунків, мобільних застосунків для Android, а також в індустрії Big Data [60].

JavaScript – це інтерпретована мова програмування з об'єктно-орієнтованими можливостями, яка забезпечує динамічну інтерактивність на веб-сторінках. Ця мова активно набирає популярність, тому що вона не потребує глибоких знань у програмуванні та сфера веб розробки користується великим попитом. Також JavaScript підтримується популярними ОС: Windows, Linux та macOS. Вона використовується зазвичай у веб застосунках та браузерях. Крім цього, розробники можуть використовувати JavaScript для створення простих веб-серверів та внутрішньої інфраструктури з використанням платформи Node.js. Головний недолік цієї мови – це те, що вона не має жодних можливостей для роботи з багатопотоковими процесами [61].

C# – це багатопарадигмальна мова програмування, що була розроблена корпорацією Microsoft та має багато спільних особливостей з мовою Java. Наприклад, мова C# дуже схожа за синтаксисом на Java, вона також має функції для збирання сміття, що допомагає ефективно використовувати оперативну пам'ять. Але на відміну від Java, програма на C# виконується у середовищі Common Language Runtime (CLR). CLR надає можливість об'єднувати в єдину програму код на різних мовах програмування: C#, C++ та Visual Basic, що є зручною функцією для програмістів. Для розробки програмного забезпечення використовується .NET Framework та така програма може бути запущена та скомпільована лише на ОС Windows, що є великим недоліком цієї мови. Для запуску такої програми, наприклад, на Linux або macOS потрібно ще додатково встановлювати фреймворк Mono, що є не дуже зручним рішенням під час розробки програмних застосунків на C#. Найчастіше ця мова використовується для створення програм для Windows, мобільних додатків, а також ігор з використанням рушія Unity [62], [63].

Результати порівняльного аналізу розглянутих мов програмування зведено в табл. 4.1.

Таблиця 4.1 – Порівняння розглянутих мов програмування

Характеристика	Python	Java	JavaScript	C#
Поріг входження	низький	середній	низький	середній
Платформа розробки	десктопна, веб	десктопна, мобільна, веб	веб	десктопна, веб
Імперативна парадигма	+	+	+	+
Декларативна парадигма	+	–	+	+
Типізація даних	динамічна	статична	динамічна	статична
Компільована мова	–	+	–	+
Багатопотоковість	+	+	–	+
Створення графічного інтерфейсу без бібліотек	–	+	+	+
Автоматичне очищення сміття	+	+	+	+
Підтримка Windows, Linux, macOS	+	+	+	±

В ході порівняльного аналізу було віддано перевагу мові Java, тому що вона є універсальною та сучасною мовою програмування високого рівня, а також задовольняє основним вимогам до розроблюваного програмного продукту, який має бути реалізовано у вигляді десктопного застосунку та працювати на популярних ОС, наприклад: Windows, Linux та macOS. Також Java є компільованою мовою програмування та має статичну типізацію даних, таким чином це дозволяє уникнути багато помилок вже на стадії компіляції та така програма працює швидше, ніж написана на мові з динамічною типізацією. Ваговою характеристикою є наявність інструментів для роботи у багатопотоковому режимі, що дозволяє одночасно виконувати декілька дій у програмі, наприклад, працювати з графічним інтерфейсом, виконувати обчислення та отримувати дані з датчиків, використовуючи REST API. Одним із важливих критеріїв є те, що Java має вбудовані можливості для створення

графічного інтерфейсу, чого немає у мові Python. У порівнянні з іншими мовами, Java не підтримує декларативну парадигму, яка дозволяє зменшити кількість написаного коду та оптимізувати його, що є єдиним недоліком для цієї мови.

Отже, беручи до уваги аргументи, що були наведені вище та власний досвід автора у програмуванні, можна вважати доцільним обрати мову Java для програмної реалізації.

До складу Java входять фреймворки, які дозволяють створювати графічний інтерфейс: Swing та JavaFX [64].

Фреймворк на Java – це попередньо розроблена платформа, що використовується програмістами для створення Java застосунків. Іншими словами – це набір визначених класів та функцій, які використовуються для обробки вхідних даних, управління апаратними пристроями та взаємодії з системним програмним забезпеченням. Фреймворк можна представити у вигляді скелета, який допомагає програмісту розробляти додаток, написавши свій власний код [65].

Swing – це графічний фреймворк, який було створено у 1997 році. Він дозволяє створювати Java застосунки з простим графічним інтерфейсом та надає широкий набір графічних елементів, наприклад: вікна, текстові поля, кнопки, прапорці тощо [66].

JavaFX – це графічний фреймворк, який було розроблено у 2008 році. Таким чином він замінив фреймворк Swing, тому що JavaFX підтримує всі сучасні технології та дозволяє створювати графічний інтерфейс будь-якої складності. Він має багато переваг, які роблять цей фреймворк кращим серед його аналогів для розробки застосунків:

- JavaFX написаний на мові програмування Java та це дозволяє використовувати всі функції цієї мови;
- JavaFX надає функції для застосування ефектів та анімацій, які є важливими для розробки ігрових додатків та роблять програмний застосунок більш сучасним та привабливим;

- JavaFX має підтримку мультимедіа, тобто можливе відтворення аудіо та відео;
- JavaFX дозволяє відображати веб-контент, а також підтримує 3D графіку;
- зовнішній вигляд графічних елементів у JavaFX можливо змінити за допомогою Cascading Style Sheets (CSS);
- розробка програмних застосунків з використанням JavaFX націлена на роботу з шаблоном (Model-View-Controller) MVC;
- JavaFX пропонує два способи створення графічного інтерфейсу: за допомогою коду та за допомогою безкоштовного візуального конструктора «Scene Builder» (рис. 4.5), у якому можна, перетягуючи елементи, швидко та просто створити інтерфейс програми [66].

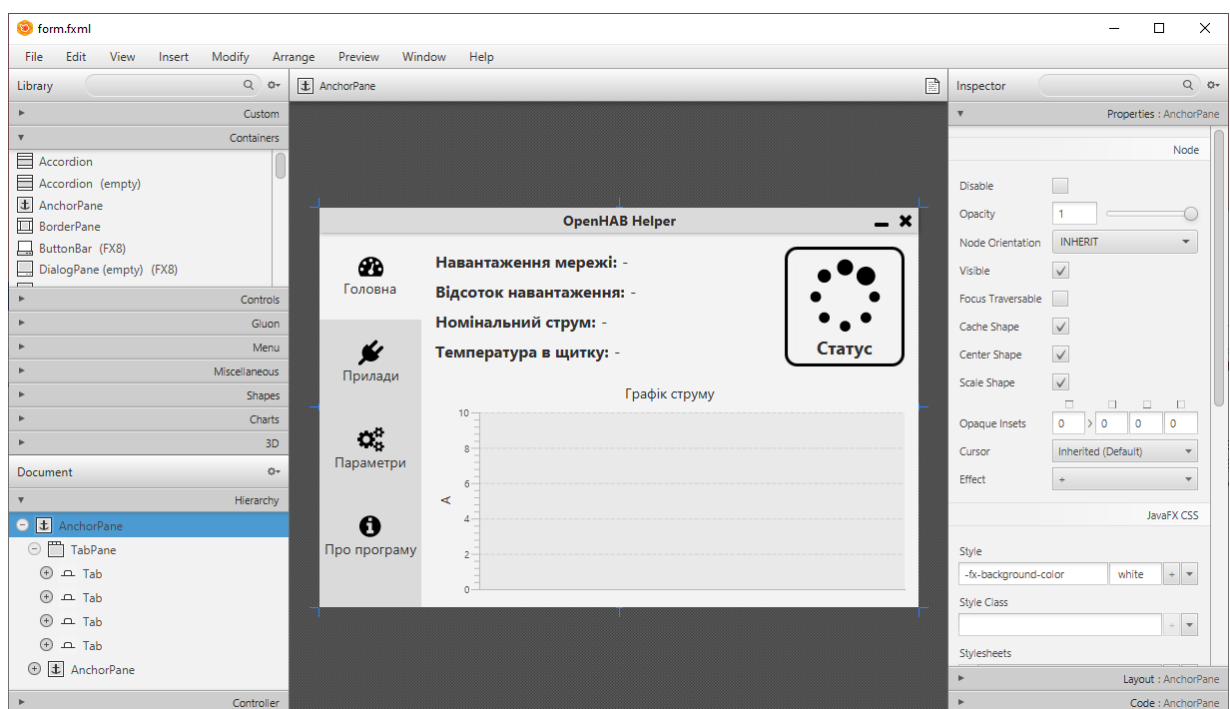


Рисунок 4.5 – Візуальний конструктор «Scene Builder»

Результати порівняльного аналізу розглянутих графічних фреймворків для мови програмування Java зведено в табл. 4.2.

Таблиця 4.2 – Порівняння розглянутих графічних фреймворків

Характеристика	Swing	JavaFX
Рік створення	1997	2008
Візуальний конструктор для створення графічного інтерфейсу	–	+
Підтримка CSS	–	+
Підтримка анімацій	–	+
Підтримка 3D графіки	–	+
Підтримка мультимедіа	–	+
Підтримка веб-контенту	–	+
Підтримка шаблону MVC	±	+

Порівнюючи графічні фреймворки, можна зробити висновок, що JavaFX має набагато більше переваг та є кращим, ніж Swing. Головна відмінність в тому, що він є відносно новим, тому що підтримує всі сучасні технології для створення графічних інтерфейсів. Таким чином JavaFX дозволяє використовувати CSS, створювати різні анімації та працювати з 3D графікою. Все це дає широкий спектр для роботи з дизайном та створенням сучасного, зручного та інтуїтивно-зрозумілого інтерфейсу у програмі. Одним із важливих критеріїв є те, що графічний інтерфейс для JavaFX можна створювати за допомогою візуального конструктора, що може допомогти зекономити багато часу та не потребує написання великої кількості коду у порівнянні зі Swing.

Для створення застосунків з графічним інтерфейсом дуже зручно використовувати шаблони проєктування, головною метою яких є відділення логіки інтерфейсу користувача від логіки програмування. Існують різні шаблони, наприклад: Model-View-Controller, Model-View-Presenter, Presentation Model та Model-View-ViewModel [67].

Для програмної реалізації з графічним інтерфейсом було обрано шаблон MVC, який розділяє програму на три окремих компоненти:

- модель – це частина, що містить в собі функціональну бізнес-логіку, а також дані, які можна завантажувати, обробляти та зберігати;
- представлення є відповідальним за відображення графічного інтерфейсу, також воно може спостерігати за моделлю і відображати дані з неї, але представлення не може безпосередньо впливати на модель;
- контролер обробляє дії користувача та оновлює модель або представлення, тобто користувач може вплинути тільки на контролер, який в свою чергу може вплинути на модель та представлення [67].

Основна ідея MVC в тому, що контролер та уявлення залежать від моделі, але модель ніяк не залежить від цих двох компонентів. Таким чином контролер реагує на дії користувача та змінює модель, викликаючи потрібний метод. Після того, як модель змінилася, представлення реагує на цю зміну та звертається до моделі за оновленими даними, після чого відображає їх [68].

Для ефективного та зручного створення програмних застосунків програмісти використовують інтегроване середовище розробки. Зазвичай воно включає в себе редактор вихідного коду, засоби автоматизації збірки проєктів та відлагоджувач. Все це є дуже корисними інструментами, які значно можуть підвищити продуктивність. На сьогодні існує багато таких середовищ та найпопулярнішими є такі: Eclipse, NetBeans, IntelliJ IDEA [69].

Для розробки програмного застосунку було обрано інтегроване середовище IntelliJ IDEA, яке є одним із кращих для розробки на мові програмування Java. IntelliJ IDEA – це функціональне та ергономічне середовище розробки з функціями для підвищення продуктивності, а також з приємним інтерфейсом. Воно має багато переваг серед його аналогів, бо включає в себе набір інструментів, що полегшують програмування, а саме: інтелектуальне завершення коду, рефакторинг коду, аналіз потоків даних тощо. Також це середовище підтримує різні мови програмування, має вбудовану систему контролю версій та великий набір корисних плагінів. IntelliJ

IDEA має дві версії: безкоштовна та комерційна. Безкоштовна версія має багато обмежень, що є єдиним недоліком такого програмного забезпечення [69].

4.2 Конструювання програмного забезпечення

4.2.1 Основні рішення щодо представлення даних програми

В ході роботи над програмним застосунком для платформи OpenHAB було розроблено наступні класи: Main, Settings, AnalyzeManager, ConnectionManager, DataBaseManager, DeviceManager, LocalizationManager, NotifyManager, OpenHabManager, SettingsManager, DeviceRecognizer, DialogUtils, FormatUtils, Controller, MainTab, DevicesTab, SettingsTab, OnlineDevice, EditableDevice, Device, OpenHabConnectionException, OpenHabException, OpenHabItemException, DataBaseConnectionException, DataBaseException. Далі будуть описані основні з них.

Клас AnalyzeManager містить в собі методи для аналізу показників з датчиків та роботи з ними. Дані та методи класу наведені в табл. 4.3.

Таблиця 4.3 – Дані та методи класу AnalyzeManager

Поля та методи класу	Опис
1	2
private:	
SettingsManager settingsManager;	Менеджер налаштувань
LocalizationManager localizationManager;	Менеджер локалізації
OpenHabManager openHabManager;	Менеджер для роботи з OpenHAB
DeviceManager deviceManager;	Менеджер для роботи з приладами
ScheduledExecutorService scheduledExecutorService;	Асинхронне виконання задачі в окремому потоці

Продовження таблиці 4.3

1	2
List<StatusListener> statusListeners;	Список спостерігачів зміни статусу
List<ElectricListener> electricListeners;	Список спостерігачів зміни показників електроспоживання
List<TemperatureListener> temperatureListeners;	Список спостерігачів зміни температурних показників
List<RecognizedDevicesListener> recognizedDevicesListeners;	Список спостерігачів зміни увімкнених електроприладів
double currentAmperage;	Поточний струм
double currentVoltage;	Поточна напруга
double currentTemperature;	Поточна температура
HashMap<Device, Boolean> enabledDevices;	Список увімкнених електроприладів в OpenHAB
LinkedHashMap<Device, Boolean> recognizedDevices;	Список розпізнаних увімкнених електроприладів
void initAnalyzer();	Ініціалізація аналізатора для роботи в окремому потоці
double getAmperage();	Отримання струму з датчика
double getVoltage();	Отримання напруги з датчика
double getTemperature();	Отримання температури з датчика
HashMap<Device, Boolean> getEnabledDevices();	Отримання списку поточних увімкнених електроприладів
LinkedHashMap<Device, Boolean> getRecognizedDevices();	Отримання списку розпізнаних увімкнених електроприладів
public:	
AnalyzeManager(SettingsManager settingsManager, LocalizationManager localizationManager, OpenHabManager openHabManager, DeviceManager deviceManager);	Конструктор
ScheduledExecutorService getAnalyzerThread();	Отримання задачі в окремому потоці
double getCurrentAmperage();	Отримання поточного струму
double getCurrentVoltage();	Отримання поточної напруги
double getCurrentTemperature();	Отримання поточної температури

Продовження таблиці 4.3

1	2
<code>LinkedHashMap<Device, Boolean> getCurrentRecognizedDevices();</code>	Отримання списку розпізнаних увімкнених електроприладів
<code>void addStatusListener (StatusListener listener);</code>	Підписатися на спостерігач зміни статусу
<code>void addElectricListener (ElectricListener listener);</code>	Підписатися на спостерігач зміни показників електроспоживання
<code>void addTemperatureListener (TemperatureListener listener);</code>	Підписатися на спостерігач зміни температурних показників
<code>void addRecognizedDevicesListener (RecognizedDevicesListener listener);</code>	Підписатися на спостерігач зміни увімкнених електроприладів

Клас `DataBaseManager` містить в собі методи для роботи з базою даних. Дані та методи класу наведені в табл. 4.4.

Таблиця 4.4 – Дані та методи класу `DataBaseManager`

Поля та методи класу	Опис
private:	
<code>SettingsManager settingsManager;</code>	Менеджер налаштувань
<code>Connection getConnection();</code>	Підключення до бази даних
<code>void createTableIfNotExists (Connection connection);</code>	Створення таблиці, якщо її немає
<code>void truncate(Connection connection);</code>	Очищення таблиці
public:	
<code>DataBaseManager (SettingsManager settingsManager);</code>	Конструктор
<code>List<Device> readData();</code>	Читання списку електроприладів
<code>void writeData(List<Device> devices);</code>	Збереження списку електроприладів

Клас `DeviceManager` містить в собі методи для читання та зберігання списку електроприладів, що знаходяться у будинку. Дані та методи класу наведені в табл. 4.5.

Таблиця 4.5 – Дані та методи класу DeviceManager

Поля та методи класу	Опис
private:	
DataBaseManager dataBaseManager;	Менеджер для роботи з базою даних
LocalizationManager localizationManager;	Менеджер локалізації
List<Device> allDevicesList;	Список електроприладів у будинку
List<Device> loadAllDevices();	Завантаження списку електроприладів
public:	
DeviceManager(DataBaseManager dataBaseManager, LocalizationManager localizationManager);	Конструктор
void saveAllDevices();	Збереження списку електроприладів
List<Device> getAllDevices();	Отримання списку електроприладів
void addDevice(Device device);	Додавання електроприладу у список
void removeDevice(Device device);	Видалення електроприладу зі списку

Клас LocalizationManager містить в собі методи для роботи з локалізацією. Дані та методи класу наведені в табл. 4.6.

Таблиця 4.6 – Дані та методи класу LocalizationManager

Поля та методи класу	Опис
1	2
private:	
SettingsManager settingsManager;	Менеджер налаштувань
Pane layout;	Батьківський контейнер інтерфейсу
ResourceBundle resources;	Клас для роботи з файлами локалізації
Locale locale;	Поточна мова локалізації
Locale loadLocale();	Завантаження поточної локалізації
void localize(Pane parent);	Переклад всіх елементів інтерфейсу
void localizeLabel(Node node);	Переклад текстового поля
void localizeButton(Node node);	Переклад кнопки

Продовження таблиці 4.6

1	2
<code>void localizeTextField(Node node);</code>	Переклад поля вводу
public:	
<code>LocalizationManager(SettingsManager settingsManager, AnchorPane layout, ResourceBundle resources);</code>	Конструктор
<code>List<Locale> getSupportedLocales();</code>	Отримання всіх файлів локалізації
<code>int getLocale();</code>	Отримання поточної локалізації
<code>void setLocale(int index);</code>	Встановлення поточної локалізації
<code>String translate(String key);</code>	Отримання локалізованої строки

Клас `NotifyManager` містить в собі методи для формування та відправлення повідомлень користувачу на мобільний застосунок. Дані та методи класу наведені в табл. 4.7.

Таблиця 4.7 – Дані та методи класу `NotifyManager`

Поля та методи класу	Опис
1	2
private:	
<code>SettingsManager settingsManager;</code>	Менеджер налаштувань
<code>LocalizationManager localizationManager;</code>	Менеджер локалізації
<code>OpenHabManager openHabManager;</code>	Менеджер для роботи з OpenHAB
<code>AnalyzeManager analyzeManager;</code>	Менеджер аналізу показників
<code>void initListener();</code>	Ініціалізація спостерігача зміни статусу
<code>String createMessage (StatusType statusType);</code>	Формування локалізованого повідомлення для користувача
<code>StringJoiner getDisallowedDevices();</code>	Отримання списку електроприладів, які не можна вмикати
<code>String getTemperatureMessage();</code>	Перевірка відхилення від температури та формування локалізованого повідомлення

Продовження таблиці 4.7

1	2
public:	
NotifyManager(SettingsManager settingsManager, LocalizationManager localizationManager, OpenHabManager openHabManager, AnalyzeManager analyzeManager);	Конструктор

Клас OpenHabManager містить в собі методи для роботи з OpenHAB REST API. Дані та методи класу наведені в табл. 4.8.

Таблиця 4.8 – Дані та методи класу OpenHabManager

Поля та методи класу	Опис
private:	
SettingsManager settingsManager;	Менеджер налаштувань
ConnectionManager connectionManager;	Менеджер для створення запитів
String createOpenHabURL (String itemName);	Формування посилання REST API
public:	
OpenHabManager (SettingsManager settingsManager)	Конструктор
void sendData (String itemName, String message);	Відправлення повідомлення на OpenHAB
String getData(String itemName);	Отримання даних з датчика підключеного до OpenHAB

Клас SettingsManager містить в собі методи для завантаження та збереження налаштувань програми. Дані та методи класу наведені в табл. 4.9.

Таблиця 4.9 – Дані та методи класу SettingsManager

Поля та методи класу	Опис
1	2
private:	
Preferences data;	Клас для роботи з налаштуваннями
int language;	Мова інтерфейсу
String openHabIp;	Хост OpenHAB
int openHabPort;	Порт OpenHAB
long updateInterval;	Інтервал оновлення датчиків
String itemAmperage;	Назва датчика струму
String itemVoltage;	Назва датчика напруги
String itemTemperature;	Назва датчика температури
double minAmperage;	Мінімальне значення струму
int maxAmperage;	Номінальне значення струму
int warningPercentage;	Поріг попередження
String itemMessage;	Відправлення повідомлень
String mysqlIp;	Хост бази даних
int mysqlPort;	Порт бази даних
String mysqlUser;	Користувач бази даних
String mysqlPassword;	Пароль бази даних
String mysqlDatabase;	Назва бази даних
void load();	Завантаження налаштувань
public:	
SettingsManager();	Конструктор
void save();	Збереження налаштувань
boolean isSettingsDone();	Перевірка на правильність налаштувань
int getLanguage();	Отримання мови
void setLanguage(int language);	Встановлення мови
String getOpenHabIp();	Отримання хосту OpenHAB
void setOpenHabIp(String openHabIp);	Встановлення хосту OpenHAB
int getOpenHabPort();	Отримання порту OpenHAB
void setOpenHabPort (int openHabPort);	Встановлення порту OpenHAB
long getUpdateInterval();	Отримання інтервалу оновлення
void setUpdateInterval (long updateInterval);	Встановлення інтервалу оновлення

Продовження таблиці 4.9

1	2
String getItemAmperage();	Отримання назви датчика струму
void setItemAmperage (String itemAmperage);	Встановлення назви датчика струму
String getItemVoltage();	Отримання назви датчика струму
void setItemVoltage (String itemVoltage);	Встановлення назви датчика струму
String getItemTemperature();	Отримання назви датчика температури
void setItemTemperature (String itemTemperature);	Встановлення назви датчика температури
double getMinAmperage();	Отримання мінімального значення струму
void setMinAmperage (double minAmperage);	Встановлення мінімального значення струму
int getMaxAmperage();	Отримання номінального значення струму
void setMaxAmperage (int maxAmperage);	Встановлення номінального значення струму
int getWarningPercentage();	Отримання значення порогу попередження
void setWarningPercentage (int warningPercentage);	Встановлення значення порогу попередження
String getItemMessage();	Отримання назви відправлення повідомлень
void setItemMessage (String itemMessage);	Встановлення назви відправлення повідомлень
String getMysqlIp();	Отримання хосту бази даних
void setMysqlIp(String mysqlIp);	Встановлення хосту бази даних
int getMysqlPort();	Отримання порту бази даних
void setMysqlPort(int mysqlPort);	Встановлення порту бази даних
String getMysqlUser();	Отримання користувача бази даних
void setMysqlUser(String mysqlUser);	Встановлення користувача бази даних
String getMysqlPassword();	Отримання паролю бази даних

Продовження таблиці 4.9

1	2
<code>void setMysqlPassword (String mysqlPassword);</code>	Встановлення паролю бази даних
<code>String getMysqlDatabase();</code>	Отримання назви бази даних
<code>void setMysqlDatabase (String mysqlDatabase);</code>	Встановлення назви бази даних

Клас `DeviceRecognizer` містить в собі методи для розпізнавання увімкнених електроприладів, використовуючи значення загального навантаження в електромережі. Дані та методи класу наведені в табл. 4.10.

Таблиця 4.10 – Дані та методи класу `DeviceRecognizer`

Поля та методи класу	Опис
1	2
<code>private:</code>	
<code>List<Device> allDevices;</code>	Список електроприладів у будинку
<code>double resultAmperage;</code>	Загальне навантаження
<code>HashMap<Device, Boolean> enabledDevices;</code>	Список поточних увімкнених електроприладів
<code>double minAmperage;</code>	Мінімальне навантаження
<code>List<Device> startDevices;</code>	Список увімкнених електроприладів
<code>double startAmperage;</code>	Поточне навантаження
<code>List<Device> finalDevices;</code>	Список розпізнаних увімкнених електроприладів
<code>double finalPrecision;</code>	Точність навантаження
<code>double calcStartAmperage();</code>	Розрахунок навантаження увімкнених електроприладів
<code>void subsetSum(List<Device> devices, List<Device> partial);</code>	Генерація множини електроприладів

Продовження таблиці 4.10

1	2
public:	
DeviceRecognizer(List<Device> allDevices, double resultAmperage, HashMap<Device, Boolean> enabledDevices, double minAmperage);	Конструктор
void recognize();	Розпізнавання увімкнених електроприладів
List<Device> getFinalDevices();	Отримання списку розпізнаних увімкнених електроприладів

Клас MainTab містить в собі реалізацію інтерфейсу вкладки «Головна». Дані та методи класу наведені в табл. 4.13.

Таблиця 4.11 – Дані та методи класу MainTab

Поля та методи класу	Опис
1	2
private:	
SettingsManager settingsManager;	Менеджер налаштувань
LocalizationManager localizationManager;	Менеджер локалізації
OpenHabManager openHabManager;	Менеджер для роботи з OpenHAB
AnalyzeManager analyzeManager;	Менеджер аналізу показників
Label currentVoltage;	Текстове поле напруги
Label currentAmperage;	Текстове поле струму
Label currentPercent;	Текстове поле відсотку струму
Label nominalAmperage;	Текстове поле номінального струму
Label currentTemperature;	Текстове поле температури
Rectangle statusBorder;	Фон загального статусу
FontAwesomeIconView statusIcon;	Іконка загального статусу
Label statusText;	Текстове поле загального статусу
AreaChart<String, Number> chart;	Графік струму
CategoryAxis xAxis;	Горизонтальна вісь графіку

Продовження таблиці 4.11

1	2
NumberAxis yAxis;	Вертикальна вісь графіку
StackPane mainBlock;	Контейнер повідомлень
XYChart.Series<String, Number> series;	Список всіх значень графіку
void initListener();	Ініціалізація спостерігача статусу, температурних показників та показників електроспоживання
void renderElectric (double amperage, double voltage);	Відображення показників електроспоживання
void renderTemperature (double temperature);	Відображення температурних показників
void setStatus(StatusType status);	Встановлення статусу
void renderChart(double amperage);	Відображення графіку струму
public:	
MainTab(SettingsManager settingsManager, LocalizationManager localizationManager, OpenHabManager openHabManager, AnalyzeManager analyzeManager);	Конструктор

Клас DevicesTab містить в собі реалізацію інтерфейсу вкладки «Прилади». Дані та методи класу наведені в табл. 4.12.

Таблиця 4.12 – Дані та методи класу DevicesTab

Поля та методи класу	Опис
1	2
private:	
SettingsManager settingsManager;	Менеджер налаштувань
LocalizationManager localizationManager;	Менеджер локалізації
DeviceManager deviceManager;	Менеджер для роботи з приладами
AnalyzeManager analyzeManager;	Менеджер аналізу показників

Продовження таблиці 4.12

1	2
AnchorPane showDevices;	Форма відображення увімкнених електроприладів
VBox currentDevices;	Контейнер увімкнених приладів
Button editDevices;	Кнопка редагування списку приладів
AnchorPane editorDevices;	Форма редагування списку приладів
VBox allDevices;	Контейнер списку приладів
TextField deviceName;	Поле вводу назви приладу
TextField deviceAmperage;	Поле вводу навантаження приладу
TextField deviceItem;	Поле вводу назви датчика приладу
Button addDevice;	Кнопка додавання приладу
Button saveDevices;	Кнопка збереження списку приладів
StackPane devicesBlock;	Контейнер повідомлень
void initButtonEvents();	Ініціалізація подій натискання кнопок
void initListener();	Ініціалізація спостерігача зміни увімкнених електроприладів
void renderRecognizedDevices (LinkedHashMap<Device, Boolean> recognizedDevices);	Відображення списку увімкнених електроприладів
void renderAllDevicesList();	Відображення списку всіх приладів
void addDevice(Device device);	Додавання електроприладу
void initTextFieldRegex();	Ініціалізація регулярних виразів для полів вводу
public:	
DevicesTab(SettingsManager settingsManager, LocalizationManager localizationManager, DeviceManager deviceManager, AnalyzeManager analyzeManager);	Конструктор
void deleteDevice(Device device);	Видалення електроприладу

Клас SettingsTab містить в собі реалізацію інтерфейсу вкладки «Параметри». Дані та методи класу наведені в табл. 4.13.

Таблиця 4.13 – Дані та методи класу SettingsTab

Поля та методи класу	Опис
private:	
SettingsManager settingsManager;	Менеджер налаштувань
LocalizationManager localizationManager;	Менеджер локалізації
ComboBox language;	Випадний список доступних мов
TextField openHabIp;	Поле вводу хосту OpenHAB
TextField openHabPort;	Поле вводу порту OpenHAB
TextField updateInterval;	Поле вводу інтервалу оновлення
TextField itemAmperage;	Поле вводу назви датчика струму
TextField itemVoltage;	Поле вводу назви датчика напруги
TextField itemTemperature;	Поле вводу назви датчика температури
TextField minAmperage;	Поле вводу мінімального струму
TextField maxAmperage;	Поле вводу номінального струму
TextField warningPercentage;	Поле вводу порогу попередження
TextField itemMessage;	Поле вводу відправлення повідомлень
TextField mysqlIp;	Поле вводу хосту бази даних
TextField mysqlPort;	Поле вводу порту бази даних
TextField mysqlUser;	Поле вводу користувача бази даних
TextField mysqlPassword;	Поле вводу паролю бази даних
TextField mysqlDatabase;	Поле вводу назви бази даних
Button saveSettings;	Кнопка збереження налаштувань
void render();	Відображення всіх налаштувань
void save(MouseEvent e);	Збереження всіх налаштувань
void onLanguageChange (Observable observable);	Подія під час зміни мови у випадному списку
void initTextFieldRegex();	Ініціалізація регулярних виразів для полів вводу
public:	
SettingsTab(SettingsManager settingsManager, LocalizationManager localizationManager);	Конструктор

Клас Device містить у собі загальну інформацію про електроприлад. Дані та методи класу наведені в табл. 4.14.

Таблиця 4.14 – Дані та методи класу Device

Поля та методи класу	Опис
private:	
String name;	Назва електроприладу
double amperage;	Навантаження електроприладу
String item;	Назва датчику електроприладу
public:	
Device(String name, double amperage, String item);	Конструктор
String getName();	Отримання назви електроприладу
double getAmperage();	Отримання навантаження електроприладу
String getItem();	Отримання назви датчику електроприладу
boolean equals(Object object);	Порівняння електроприладу з іншим

Клас Main наслідує клас Application та використовується для запуску програмного застосунку з використанням графічного фреймворку JavaFX. Методи класу наведені в табл. 4.15.

Таблиця 4.15 – Методи класу Main

Методи класу	Опис
public:	
void main(String[] args);	Вхідна точка для запуску програмного застосунку
void start(Stage stage);	Вхідна точка для запуску графічного фреймворку JavaFX

Діаграму основних класів програмного застосунку зображено на рисунку 4.6.

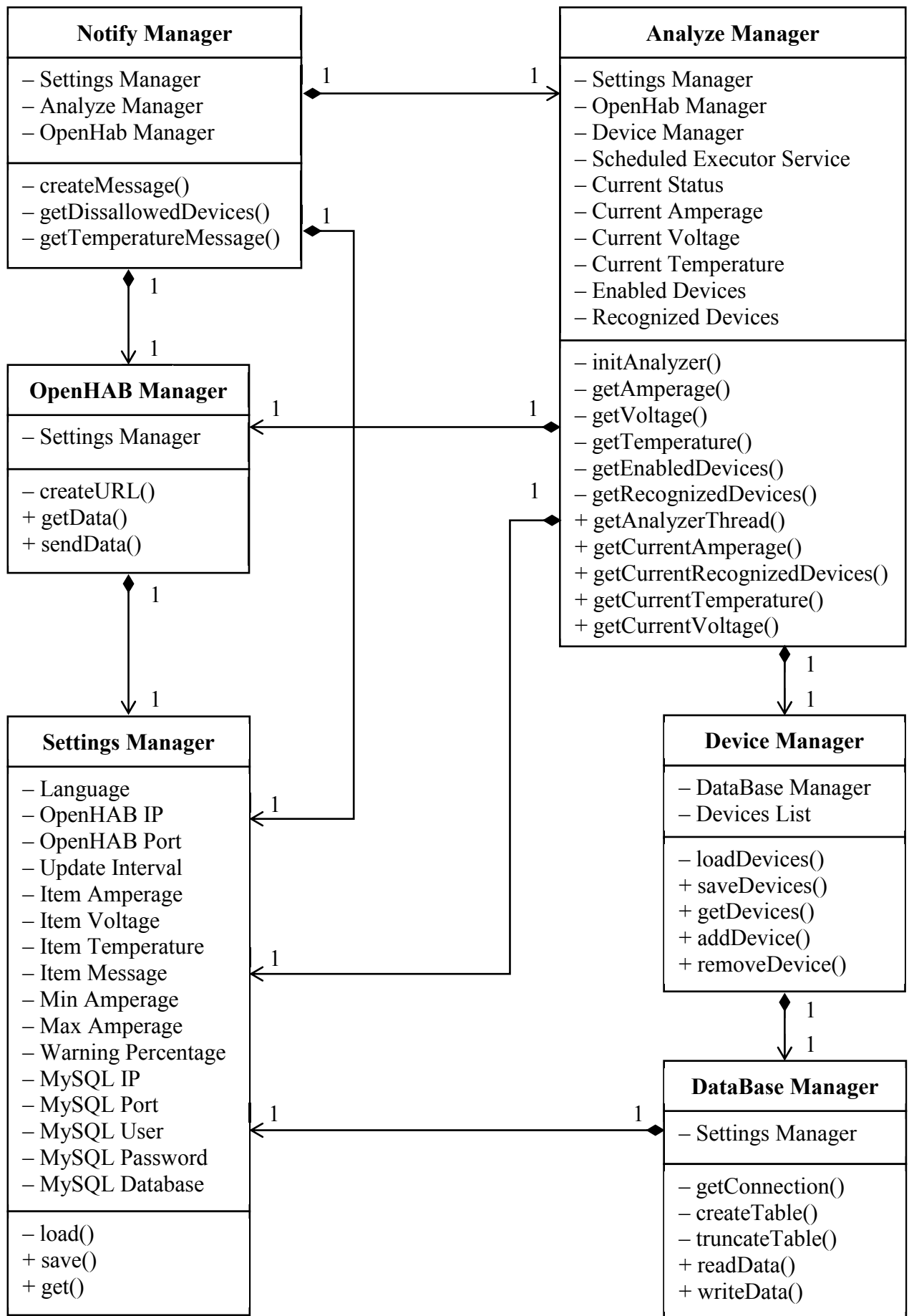


Рисунок 4.6 – Діаграма класів

4.2.2 Основні розроблені алгоритми

Узагальнений алгоритм роботи програмного застосунку зображено на рисунку 4.7.



Рисунок 4.7 – Узагальнений алгоритм роботи програмного застосунку

Блок 1. Початок.

Блок 2. Підключення до OpenHAB та отримання показників електрощитового обладнання, використовуючи REST API.

Блок 3. Аналіз показників, тобто перевірка їх на те, щоб вони не перевищували допустимі значення, а також формування загального статусу електромережі.

Блок 4. Розпізнавання увімкнених електроприладів у будинку, використовуючи лише загальне навантаження в електромережі, а також обчислення їхнього споживання.

Блок 5. Візуалізація результатів за допомогою графічного інтерфейсу.

Блок 6. Якщо електромережа має велике навантаження, то перехід до блоку 7, в іншому випадку – до блоку 8.

Блок 7. Відправлення повідомлення користувачу на мобільний застосунок, а також рекомендації щодо безпечного користування електроприладами.

Блок 8. Кінець.

Алгоритм аналізу показників електроспоживання зображено на рисунку 4.8.

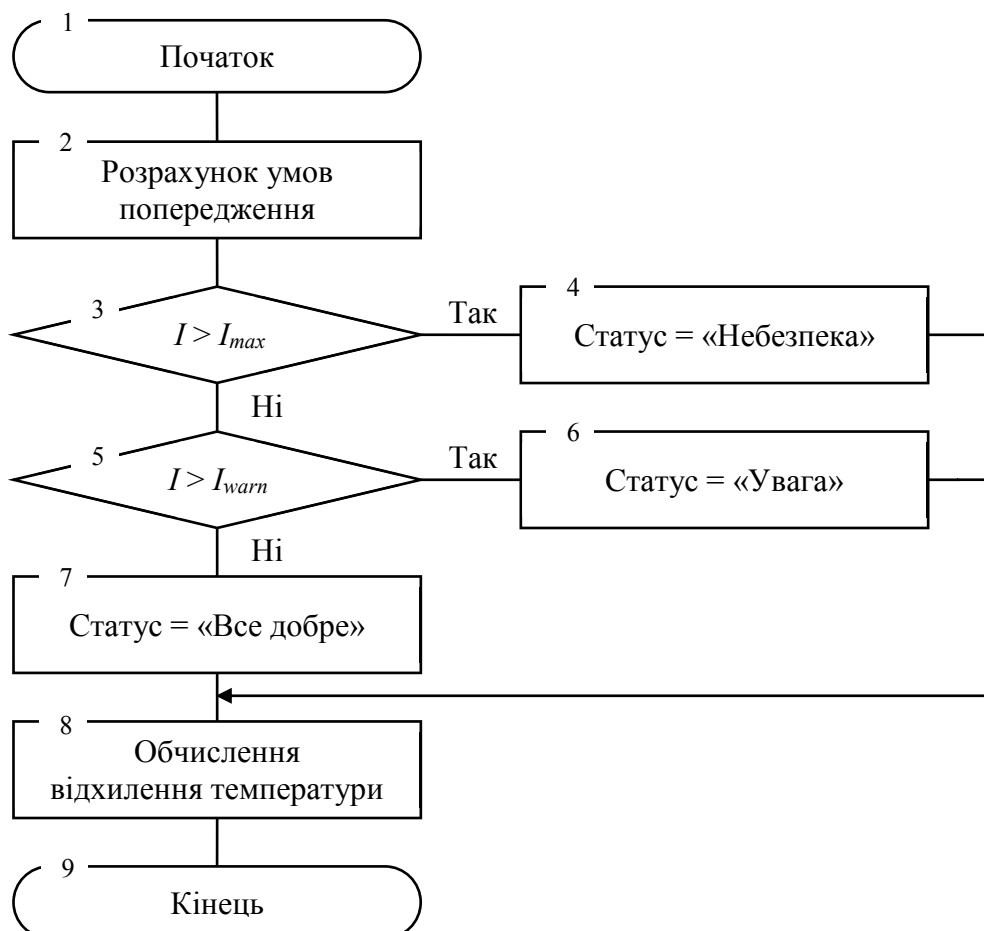


Рисунок 4.8 – Алгоритм аналізу показників електроспоживання

Блок 1. Початок.

Блок 2. Розрахунок умов попередження I_{warn} виконується за формулою (4.1):

$$I_{warn} = \frac{I_{max} \times P}{100\%}, \quad (4.1)$$

де I_{max} – номінальний струм для електромережі;

P – поріг попередження.

Блок 3. Якщо поточне навантаження перевищує номінальний струм I_{max} , то перехід до наступного блоку, в іншому випадку – до блоку 5.

Блок 4. Загальний статус електромережі «Небезпека» та перехід до блоку 8.

Блок 5. Якщо поточне навантаження перевищує поріг попередження I_{warn} , то перехід до наступного блоку, в іншому випадку – до блоку 7.

Блок 6. Загальний статус електромережі «Увага» і перехід до блоку 8.

Блок 7. Загальний статус електромережі «Все добре».

Блок 8. Обчислення відхилення температури t_{Δ} для ввідного автомата виконується за формулою (4.2):

$$t_{\Delta} = \frac{t - t_{norm}}{t_{norm}} \times 100\%, \quad (4.2)$$

де t – поточна температура ввідного автомата;

t_{norm} – робоча температура ввідного автомата.

Блок 9. Кінець.

Таким чином алгоритм аналізу показників електроспоживання перевіряє, щоб струм в електромережі не перевищував допустимих значень для запобігання загрози виникненню пожежі, обчислює відхилення температурних показників, а також формує загальний статус електромережі.

Алгоритм розпізнавання увімкнених приладів у будинку зображено на рисунку 4.9.



Рисунок 4.9 – Алгоритм розпізнавання увімкнених електроприладів

Блок 1. Початок.

Блок 2. Генерація множини увімкнених приладів, використовуючи повний перебір всіх комбінацій.

Блок 3. Розрахунок суми струму Σ для увімкнених приладів виконується за формулою (4.3):

$$\sum_{i=1}^n I_i = I_1 + I_2 + \dots + I_n, \quad (4.3)$$

де I_n – сила струму для кожного увімкненого електроприладу;

n – кількість увімкнених електроприладів.

Блок 4. Якщо сума струму Σ увімкнених приладів дорівнює загальному струму в електромережі I , то перехід до наступного блоку, в іншому випадку перехід до блоку 2.

Блок 5. Розрахунок навантаження P приладів виконується за формулою (4.4):

$$P = U \times I, \quad (4.4)$$

де U – напруга в електромережі;

I – сила струму.

Блок 6. Кінець.

Таким чином цей алгоритм дозволяє розпізнати увімкнені електроприлади у будинку, використовуючи лише загальне значення струму в електромережі, а також обчислює їхнє навантаження.

4.2.3 Основні рішення щодо модульної побудови програми

Програмний застосунок складається з файлів та класів, які розподілені між собою за різними функціями та взаємодіють один з одним для виконання поставленої задачі.

Структура програмного застосунку «OpenHAB Helper» зображена на рисунку 4.10.

Пакет lang:

- lang_en.properties – файл, який містить переклад елементів графічного інтерфейсу на англійську мову;
- lang_ru.properties – файл, який містить переклад елементів графічного інтерфейсу на російську мову;
- lang_uk.properties – файл, який містить переклад елементів графічного інтерфейсу на українську мову.

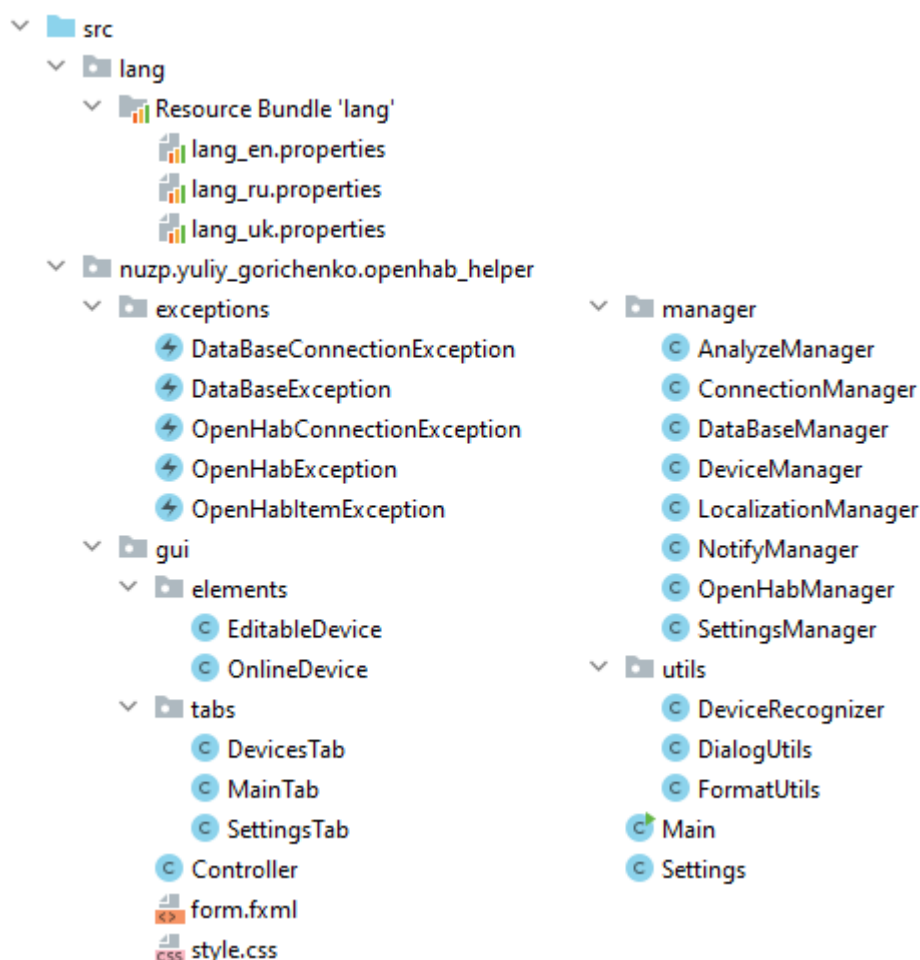


Рисунок 4.10 – Структура програмного застосунку

Пакет `nuzp.yuliy_gorichenko.openhab_helper`:

- `Main.java` – файл, у якому описана функція `main(String[] args)` – основна точка входу в програму;
- `Settings.java` – файл, що містить оголошення змінних, які виступають у ролі налаштувань різних функцій програми.

Пакет `nuzp.yuliy_gorichenko.openhab_helper.exceptions`:

- `DataBaseConnectionException.java` – файл, що містить оголошення класу `DataBaseConnectionException`, який призначено для обробки виняткових ситуацій коли відсутнє з'єднання з базою даних;
- `DataBaseException.java` – файл, що містить оголошення класу `DataBaseException`, який призначено для обробки виняткових ситуацій коли виникла помилка під час SQL-запиту у базі даних;

- `OpenHabConnectionException.java` – файл, що містить оголошення класу `OpenHabConnectionException`, який призначено для обробки виняткових ситуацій коли відсутнє з'єднання з платформою OpenHAB;
- `OpenHabException.java` – файл, що містить оголошення класу `OpenHabException`, який призначено для обробки виняткових ситуацій коли виникла помилка під час роботи з OpenHAB REST API;
- `OpenHabItemException.java` – файл, що містить оголошення класу `OpenHabItemException`, який призначено для обробки виняткових ситуацій коли неможливо отримати дані з датчика.

Пакет `nuzp.yuliy_gorichenko.openhab_helper.gui`:

- `Controller.java` – файл, що містить оголошення класу `Controller` та виступає в ролі контролера для графічного інтерфейсу;
- `form.fxml` – файл, що містить розмітку графічного інтерфейсу на мові FXML, який створений на основі XML;
- `style.css` – файл, в якому міститься перелік правил CSS, які визначають стиль та зовнішній вигляд графічних елементів.

Пакет `nuzp.yuliy_gorichenko.openhab_helper.gui.elements`:

- `EditableDevice.java` – файл, що містить оголошення класу `EditableDevice`, який призначений для реалізації списку всіх приладів у графічному інтерфейсі;
- `OnlineDevice.java` – файл, що містить оголошення класу `OnlineDevice`, який призначений для реалізації списку увімкнених електроприладів у графічному інтерфейсі.

Пакет `nuzp.yuliy_gorichenko.openhab_helper.gui.tabs`:

- `DevicesTab.java` – файл, що містить оголошення класу `DevicesTab`, який містить методи для роботи з графічними елементами вкладки «Пристрої»;
- `MainTab.java` – файл, що містить оголошення класу `MainTab`, який містить методи для роботи з графічними елементами вкладки «Головна»;

- `SettingsTab.java` – файл, що містить оголошення класу `SettingsTab`, який містить методи для роботи з графічними елементами вкладки «Налаштування».

Пакет `nuzp.yuliy_gorichenko.openhab_helper.manager`:

- `AnalyzeManager.java` – файл, що містить оголошення класу `AnalyzeManager`, який містить методи для аналізу даних з датчиків та роботи з ними;

- `ConnectionManager.java` – файл, що містить оголошення класу `ConnectionManager`, який містить методи для підключення до OpenHAB;

- `DataBaseManager.java` – файл, що містить оголошення класу `DataBaseManager`, який містить методи для роботи з базою даних;

- `DeviceManager.java` – файл, що містить оголошення класу `DeviceManager`, який містить методи для роботи з електроприладами та реалізацію класу `Device`;

- `LocalizationManager.java` – файл, що містить оголошення класу `LocalizationManager`, який містить методи для роботи з локалізацією;

- `NotifyManager.java` – файл, що містить оголошення класу `NotifyManager`, який містить методи для формування та відправлення повідомлень, а також рекомендацій користувачу щодо безпечного користування електроприладами;

- `OpenHabManager.java` – файл, що містить оголошення класу `OpenHabManager`, який містить методи для роботи з OpenHAB REST API;

- `SettingsManager.java` – файл, що містить оголошення класу `SettingsManager`, який містить методи для зчитування та запису налаштувань програми.

Пакет `nuzp.yuliy_gorichenko.openhab_helper.utils`:

- `DeviceRecognizer.java` – файл, що містить оголошення класу `DeviceRecognizer`, який містить методи для розпізнавання увімкнених приладів, використовуючи лише загальне навантаження в електромережі, а також обчислення їхнього споживання;

- DialogUtils.java – файл, що містить оголошення класу DialogUtils, який містить методи для виведення інформативних помилок у вигляді діалогових вікон;
- FormatUtils.java – файл, що містить оголошення класу FormatUtils, який містить методи для форматування показників електроспоживання згідно потрібної локалізації.

4.2.4 Особливості реалізації бази даних

Для зберігання списку всіх електроприладів, що знаходяться у будинку, використовується база даних MySQL, що входить до складу СДА. Кожен електроприлад включає наступні дані: назва, навантаження, назва актуатора в системі OpenHAB (якщо є). Для зберігання цього списку у базі даних була створена таблиця «openhab_helper». Характеристику полів цієї таблиці подано у табл. 4.16.

Таблиця 4.16 – Характеристика полів таблиці «openhab_helper»

Ідентифікатор	Тип	Розмір	Ключ	Коментар
id	Число	Довге ціле	Первинний	Ключове поле
name	Короткий текст	30 символів	–	Назва приладу
amperage	Число	Дійсне число	–	Навантаження
item	Короткий текст	30 символів	–	Назва датчику

Усі тексти SQL-запитів, що використовувались у програмному застосунку для роботи з базою даних, подано у табл. 4.17.

Таблиця 4.17 – Тексти SQL-запитів

Текст SQL-запиту	Коментар
<pre>CREATE TABLE IF NOT EXISTS `openhab_helper`(`id` INT NOT NULL AUTO_INCREMENT, `name` VARCHAR(30) NOT NULL, `amperage` DOUBLE NOT NULL, `item` VARCHAR(30) NOT NULL, PRIMARY KEY (`id`));</pre>	Створення таблиці
<pre>TRUNCATE TABLE `openhab_helper`;</pre>	Очищення таблиці
<pre>SELECT * FROM `openhab_helper`;</pre>	Перегляд списку всіх приладів
<pre>INSERT INTO `openhab_helper` (`name`, `amperage`, `item`) VALUES (?, ?, ?);</pre>	Поповнення таблиці

4.2.5 Особливості реалізації графічного інтерфейсу

Для реалізації графічного інтерфейсу програмного застосунку було використано безкоштовний візуальний конструктор «Scene Builder», який дозволяє легко та швидко створювати інтерфейс за допомогою перетягування потрібних елементів, які вбудовані у графічний фреймворк JavaFX.

Для відображення різних значків та іконок у графічному інтерфейсі була використана бібліотека FontAwesomeFX, що включає великий набір векторних іконок, які роблять інтерфейс більш привабливим та зрозумілим. Ця бібліотека поширюється під ліцензією Apache 2.0 та це означає, що користувач має право вільно використовувати таке програмне забезпечення для будь-яких цілей [70].

Далі будуть описані основні елементи, що використовувались під час створення графічного інтерфейсу.

Вкладка «Головна» містить наступні елементи (рис. 4.11):

- елемент інтерфейсу TabPane;
- елемент інтерфейсу Label;
- елемент інтерфейсу Button;
- елемент інтерфейсу FontAwesomeIconView;
- елемент інтерфейсу AreaChart.

Вкладка «Прилади» містить наступні графічні елементи (рис. 4.12):

- елемент інтерфейсу Label;
- елемент інтерфейсу Button;
- елемент інтерфейсу ScrollPane;
- власний елемент інтерфейсу OnlineDevice.

Вкладка «Параметри» містить наступні графічні елементи (рис. 4.13):

- елемент інтерфейсу Label;
- елемент інтерфейсу ComboBox;
- елемент інтерфейсу TextField.

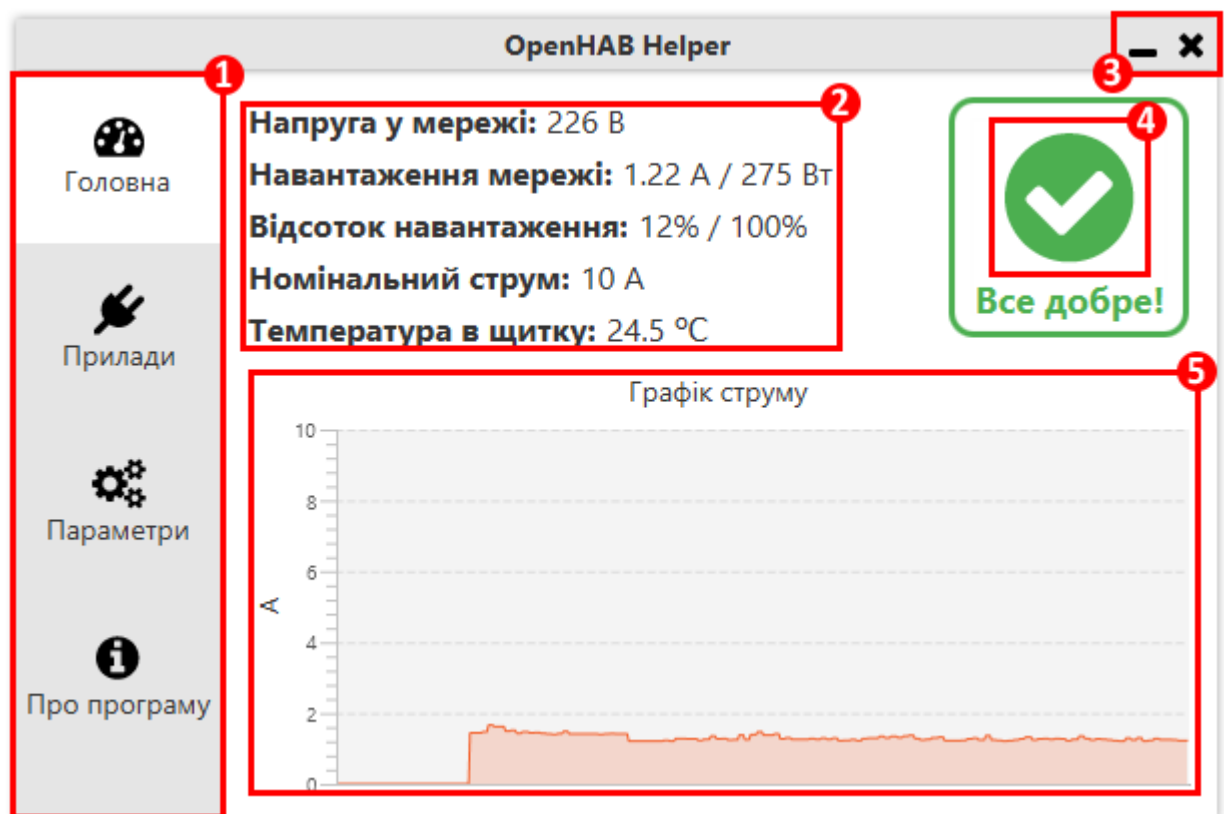


Рисунок 4.11 – Елементи вкладки «Головна»

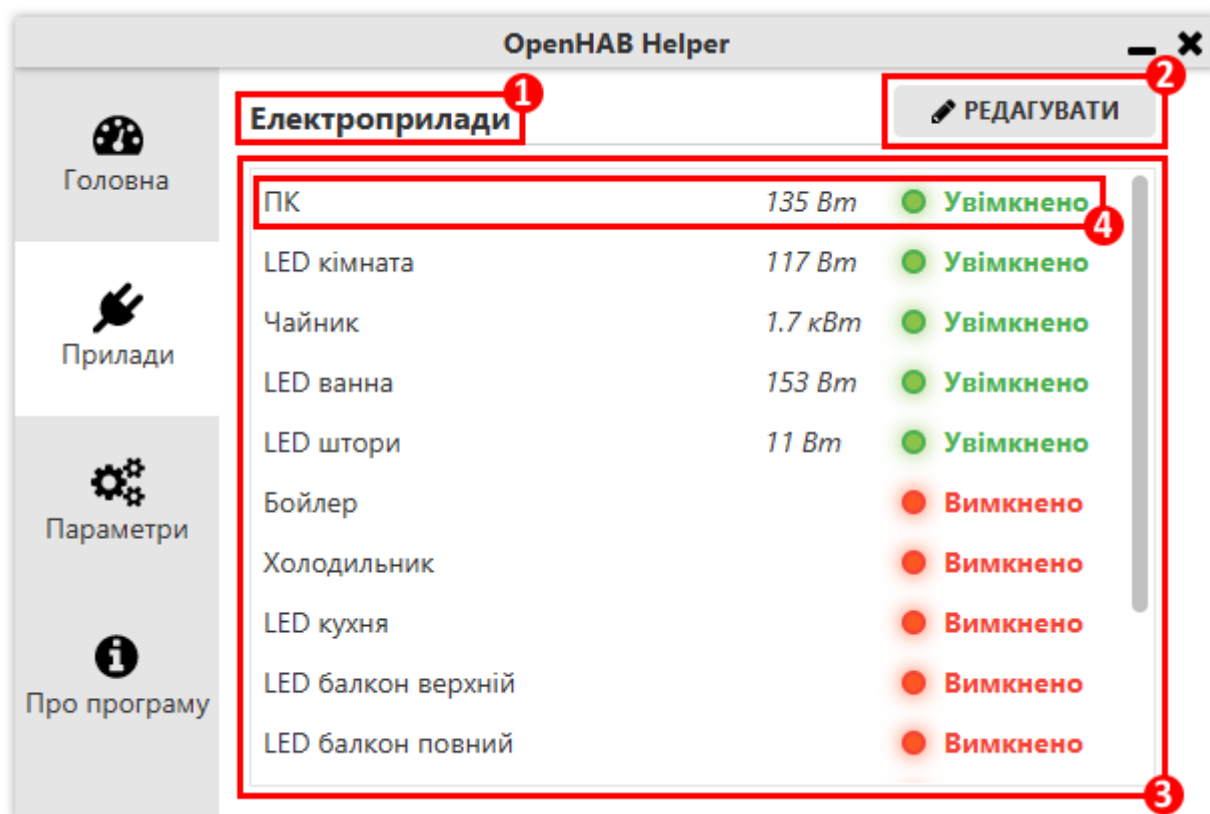


Рисунок 4.12 – Елементи вкладки «Прилади»

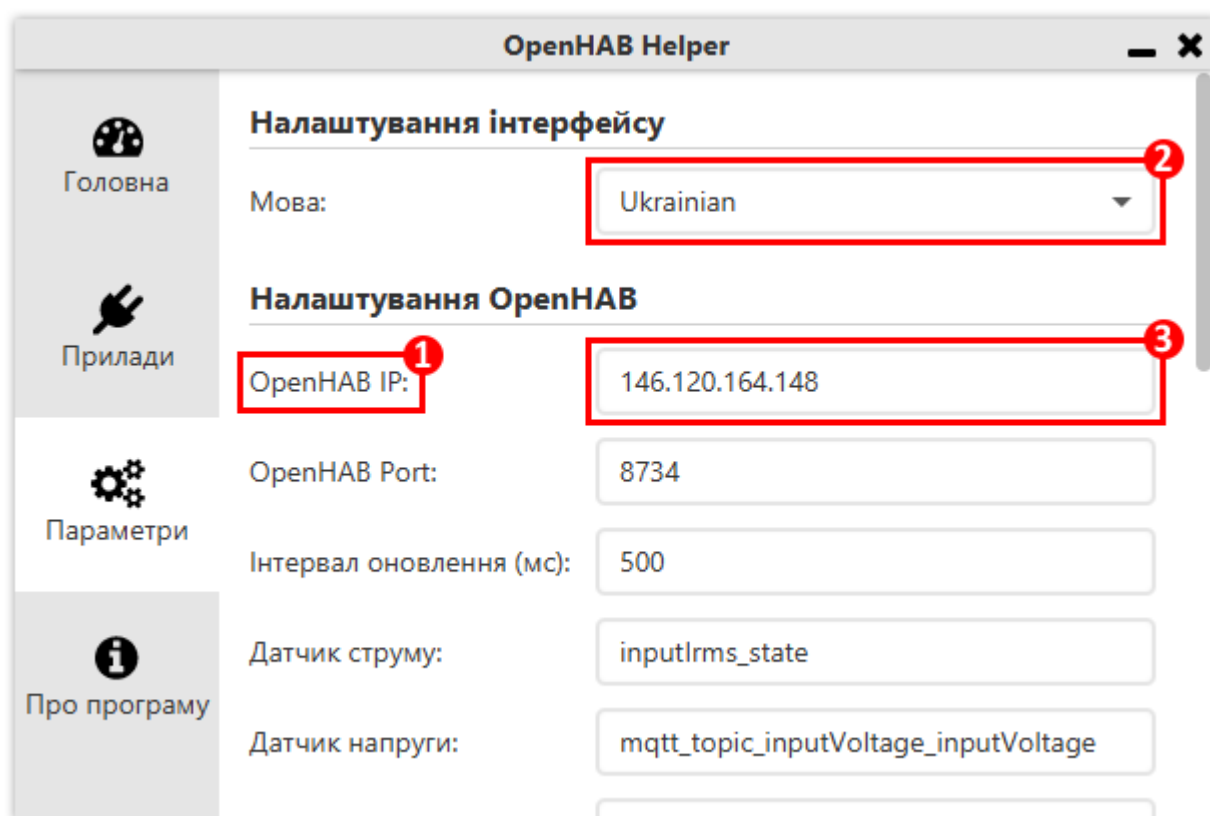


Рисунок 4.13 – Елементи вкладки «Параметри»

4.3 Висновки до розділу

Було виконано огляд технічного завдання (додаток А), а також були розроблені функціональні та нефункціональні вимоги до програмного застосунку. Було розроблено структурну схему програми, яка має наступні модулі: модуль візуалізації; локалізації; надання рекомендацій; збереження налаштувань; аналізу даних з датчиків; аналізу електроприладів; модуль роботи з базою даних MySQL та модуль для роботи з OpenHAB. Алгоритм роботи розроблюваного програмного застосунку представлено у вигляді діаграми послідовності (рис. 4.3).

Для програмної реалізації було віддано перевагу мові програмування Java, тому що вона відповідає основним вимогам до розроблюваного програмного продукту, підтримується популярними ОС (Windows, Linux, macOS) та має інструменти для роботи у багатопотоковому режимі, що дозволяє одночасно виконувати декілька дій. Порівнюючи з іншими мовами, Java працює набагато швидше та має вбудовані можливості для створення графічного інтерфейсу.

Для розробки інтерфейсу було обрано вбудований графічний фреймворк JavaFX, який підтримує усі нові технології та має багато переваг серед його аналогів, наприклад: візуальний конструктор для створення інтерфейсу, підтримка CSS, анімацій тощо. Все це дає можливість створювати сучасний, зручний та інтуїтивно-зрозумілий інтерфейс у програмі.

Для програмної реалізації з графічним інтерфейсом було обрано шаблон проектування MVC та середовище розробки IntelliJ IDEA, бо воно має багато переваг, тому що включає в себе набір інструментів, які полегшують програмування та підвищують ефективність.

Розроблений програмний застосунок складається з різних класів, які розподілені між собою за різними функціями та взаємодіють один з одним для виконання поставленої задачі. Опис основних створених класів подано у

таблицях 4.3-4.15. Діаграму основних класів програмного застосунку зображено на рисунку 4.6.

Основні розроблені алгоритми наведено на рисунках 4.7-4.9, де продемонстровано узагальнений алгоритм роботи програмного застосунку, алгоритм аналізу показників електроспоживання та алгоритм розпізнавання увімкнених електроприладів.

Структуру програмного застосунку наведено на рисунку 4.10, що складається з різних файлів та класів, які призначені для виконання різних функцій.

Для зберігання даних було реалізовано взаємодію з базою даних MySQL, що входить до складу СДА. Було створено таблицю «openhav_helper» та характеристику її полів подано у табл. 4.16.

Для реалізації графічного інтерфейсу було використано безкоштовний візуальний конструктор «Scene Builder» (рис. 4.5), який дозволяє легко та швидко створювати інтерфейс за допомогою перетягування потрібних елементів. Також під час розробки інтерфейсу було використано бібліотеку FontAwesomeFX, що включає в себе великий набір векторних значків та іконок, які зроблять інтерфейс більш привабливим та зрозумілим.

У результаті проведення тестувань не було виявлено дефектів, які могли б викликати помилки в роботі програми або її аварійне завершення. Розроблений програмний застосунок повністю відповідає вимогам технічного завдання.

5 КЕРІВНИЦТВО ПРОГРАМІСТА

5.1 Призначення та умови застосування програми

Розроблений програмний застосунок інтегрується у платформу OpenHAB та його призначення – це аналіз даних щодо споживання електроенергії для підвищення безпеки та ефективності СДА. Це дозволяє забезпечити надійний захист дротів та кабелів від перегріву, що може спричинити пожежу, а також самого будинку та електроприладів в ньому.

Програмний застосунок виконує цілодобовий контроль показників СДА та відправляє оповіщення власнику будинку про небезпечні зміни у показниках, що відслідковуються, а також надає рекомендації у реальному часі щодо безпечного користування електроприладами.

Для роботи програми необхідно мати встановлене виконавче середовище Java Runtime Environment (JRE) версії 1.8 або вище, а також мати наступні апаратно-технічні засоби:

- процесор з частотою 1 ГГц та вище;
- операційна система Microsoft Windows 7 та вище / Red Hat Enterprise Linux 5.5 та вище / Ubuntu 12.04 та вище / macOS 10.9 та вище;
- оперативна пам'ять 1 ГБ та вище;
- 3 МБ вільного місця на жорсткому диску для зберігання виконуваного файлу (*.jar);
- відеокарта з підтримкою VGA (True Color) та екран роздільною здатністю 1024 x 768 пікселів та вище;
- клавіатура;
- маніпулятор «миша».

5.2 Характеристики програми

Програмний застосунок під назвою «OpenHAB Helper» розроблено на мові програмування високого рівня Java з використанням графічного

фреймворку JavaFX та призначено для роботи на популярних операційних системах: Windows, Linux та macOS.

Розроблений програмний застосунок дозволить виконувати такі функції:

- виконувати моніторинг напруги та струму в електромережі;
- виконувати моніторинг температури ввідного автомата;
- виводити список увімкнених електроприладів та їхнє споживання в електромережі;
- можливість редагування списку електроприладів;
- збереження списку приладів у базі даних;
- можливість зміни налаштувань програми;
- збереження налаштувань програми у реєстрі Windows або у файлі конфігурації системи;
- оповіщення власника будинку про небезпечні зміни у показниках, що відслідковуються;
- надання рекомендацій щодо безпечного користування електроприладами.

5.3 Звертання до програми

Для початку роботи з програмою «OpenHAB Helper» потрібно запустити виконуваний файл `openhav_helper.jar`.

В момент першого запуску буде запропоновано перейти до налаштувань програми (рис. 5.1). Для цього потрібно натиснути на вкладку «Параметри», що знаходиться у лівій частині інтерфейсу програми.

Інтерфейс налаштування програмного застосунку зображено на рисунках 5.2-5.4.

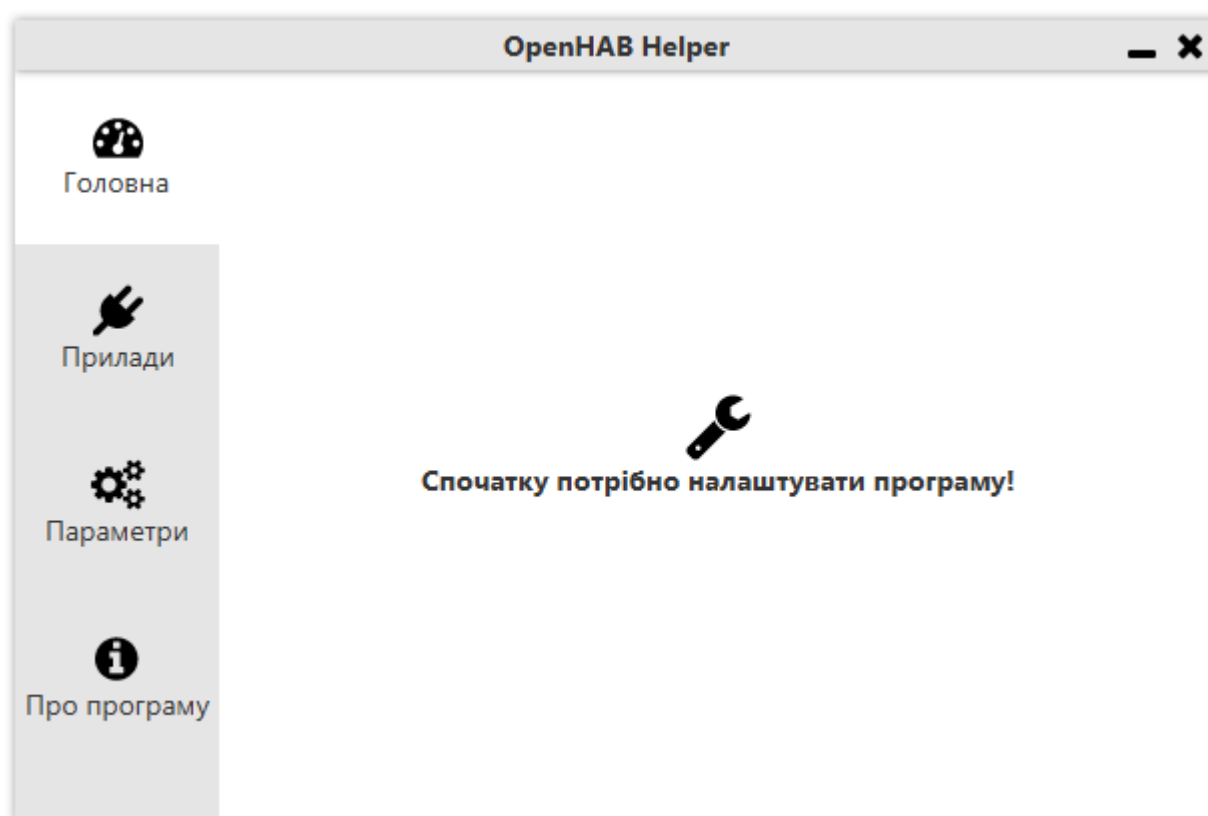


Рисунок 5.1 – Перший запуск програмного застосунку

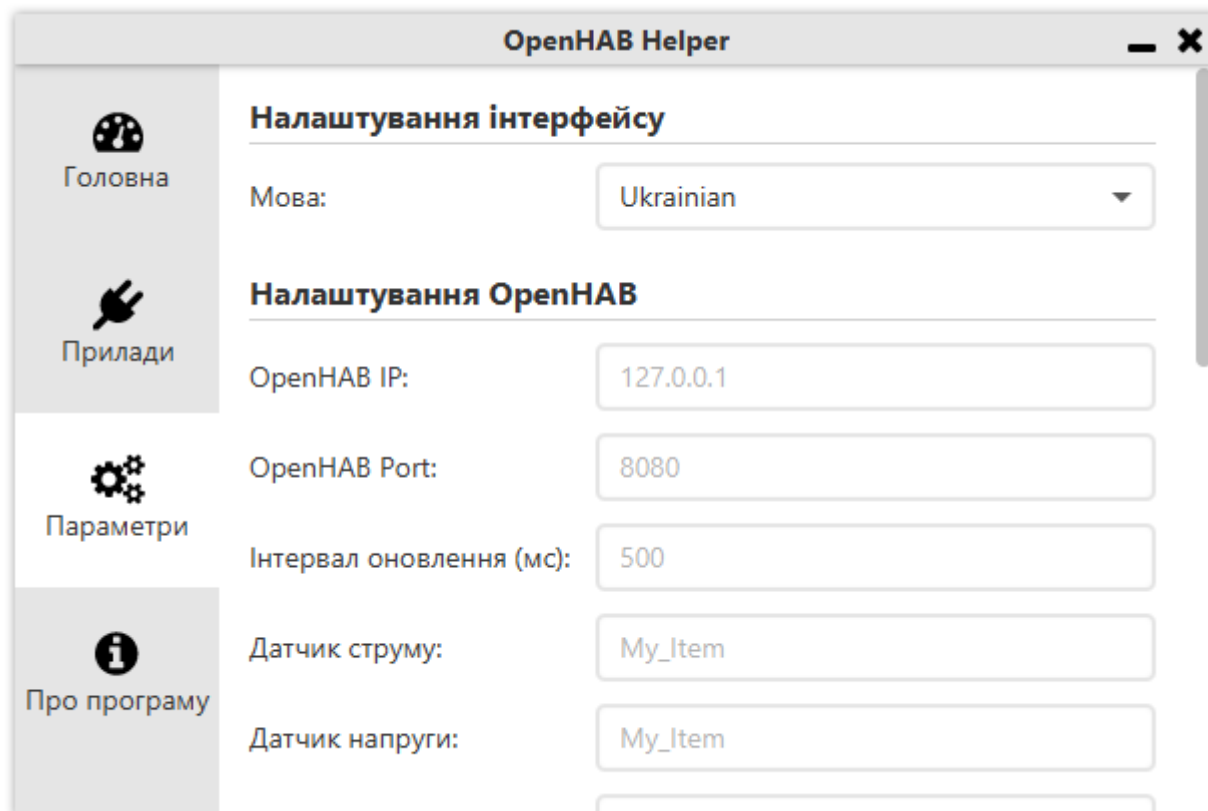


Рисунок 5.2 – Налаштування інтерфейсу

The screenshot shows the 'OpenHAB Helper' window with a sidebar on the left containing four icons and labels: 'Головна' (Home), 'Прилади' (Devices), 'Параметри' (Parameters), and 'Про програму' (About). The main area displays several configuration fields:

Інтервал оновлення (мс):	500
Датчик струму:	My_Item
Датчик напруги:	My_Item
Датчик t в щитку:	My_Item
Мінімальний струм (A):	0.1
Номінальний струм (A):	40
Поріг попередження (%):	80
Відправка повідомлень:	My_Item

Рисунок 5.3 – Налаштування OpenHAB

The screenshot shows the 'OpenHAB Helper' window with the same sidebar as Figure 5.3. The main area displays the 'Відправка повідомлень:' field set to 'My_Item', followed by a section titled 'Налаштування MySQL' (MySQL Settings) with the following fields:

MySQL IP:	127.0.0.1
MySQL Port:	3306
Логін:	db_user
Пароль:	db_password
База даних:	db_name

At the bottom of the main area, there is a button labeled '✓ ЗБЕРЕГТИ' (Save).

Рисунок 5.4 – Налаштування MySQL

Обов'язковими для вводу є всі поля, що розташовані у цій вкладці.

Розділ «Налаштування інтерфейсу» має випадний список «Мова», який дозволяє обрати мову інтерфейсу. Програмний застосунок підтримує англійську, українську та російську локалізації.

Розділ «Налаштування OpenHAB» дозволяє налаштувати параметри для роботи з платформою OpenHAB та має наступні поля:

- OpenHAB IP – це поле, у яке потрібно ввести IP-адресу для підключення до OpenHAB (якщо СДА встановлено на локальному сервері, то потрібно ввести адресу 127.0.0.1);
- OpenHAB Port – це поле, у яке потрібно ввести порт для підключення до OpenHAB (зазвичай порт має значення 8080);
- Інтервал оновлення – це поле, у яке потрібно ввести числове значення у мілісекундах, що буде визначати частоту моніторингу датчиків (рекомендується встановити 500 мс);
- Датчик струму – це поле, у яке потрібно ввести назву датчика, який вимірює струм в електромережі;
- Датчик напруги – це поле, у яке потрібно ввести назву датчика, який вимірює напругу в електромережі;
- Датчик t в щиті – це поле, у яке потрібно ввести назву датчика, який вимірює температуру ввідного автомата;
- Мінімальний струм – це поле, у яке потрібно ввести значення струму коли вимкнено всі електроприлади окрім платформи OpenHAB;
- Номінальний струм – це поле, у яке потрібно ввести значення номінального струму для ввідного автомата;
- Поріг попередження – це поле, у яке потрібно ввести числове значення, яке буде встановлювати поріг попередження для оповіщення власника будинку про можливе перевантаження електромережі;
- Відправка повідомлень – це поле, у яке потрібно ввести назву модулю в OpenHAB, який відправляє повідомлення користувачу на мобільний застосунок.

Розділ «Налаштування MySQL» дозволяє налаштувати параметри для роботи з базою даних MySQL та має наступні поля:

- MySQL IP – це поле, у яке потрібно ввести IP-адресу для підключення до бази даних (якщо базу даних встановлено на локальному сервері, то потрібно ввести адресу 127.0.0.1);
- MySQL Port – це поле, у яке потрібно ввести порт для підключення до бази даних (зазвичай порт має значення 3306);
- Логін – це поле, у яке потрібно ввести логін користувача для бази даних;
- Пароль – це поле, у яке потрібно ввести пароль користувача для бази даних;
- База даних – це поле, у яке потрібно ввести назву бази даних.

Після того, як всі поля були заповнені, потрібно натиснути на кнопку «Зберегти». Якщо все було налаштовано правильно, то програма почне свою роботу, про що буде свідчити відсутність помилок.

5.4 Вхідні та вихідні дані

Вхідними даними виступають показники з датчиків, які відслідковуються програмним застосунком (навантаження та струм в електромережі, температура ввідного автомата) та статус роботи електроприладів, які підключені до платформи OpenHAB. Також вхідними даними є список електроприладів, які додаються користувачем та зберігаються у базі даних.

Вихідними даними виступають повідомлення та рекомендації щодо безпечного користування електроприладами, які відправляються користувачу на мобільний застосунок, а також виведення на дисплей всіх показників з датчиків та увімкнених електроприладів.

5.5 Повідомлення

Розроблений програмний застосунок забезпечує стабільну роботу. Виняткові ситуації можуть виникати, якщо користувач неправильно налаштував програму або стався збій на стороні СДА. В такому випадку користувач отримає повідомлення у вигляді діалогового вікна, яке буде містити опис проблеми, що виникла, а також код помилки.

Виняткові ситуації можуть виникати під час взаємодії з платформою OpenHAB:

- відсутнє з'єднання з OpenHAB (рис. 5.5), коли неможливо підключитися до OpenHAB через неправильно введені адресу хосту та порту;
- помилка під час роботи з OpenHAB (рис. 5.6);
- помилка під час отримання даних з датчика (рис. 5.7), коли неправильно вказана назва датчику.

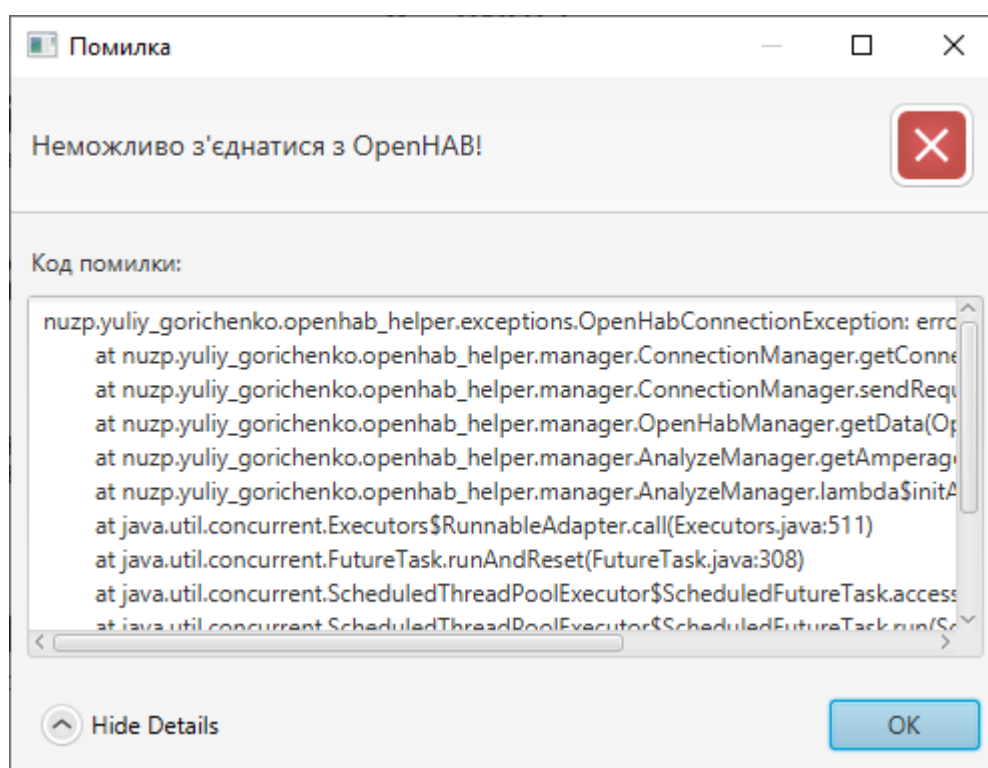


Рисунок 5.5 – Повідомлення, якщо неможливо з'єднатися з OpenHAB

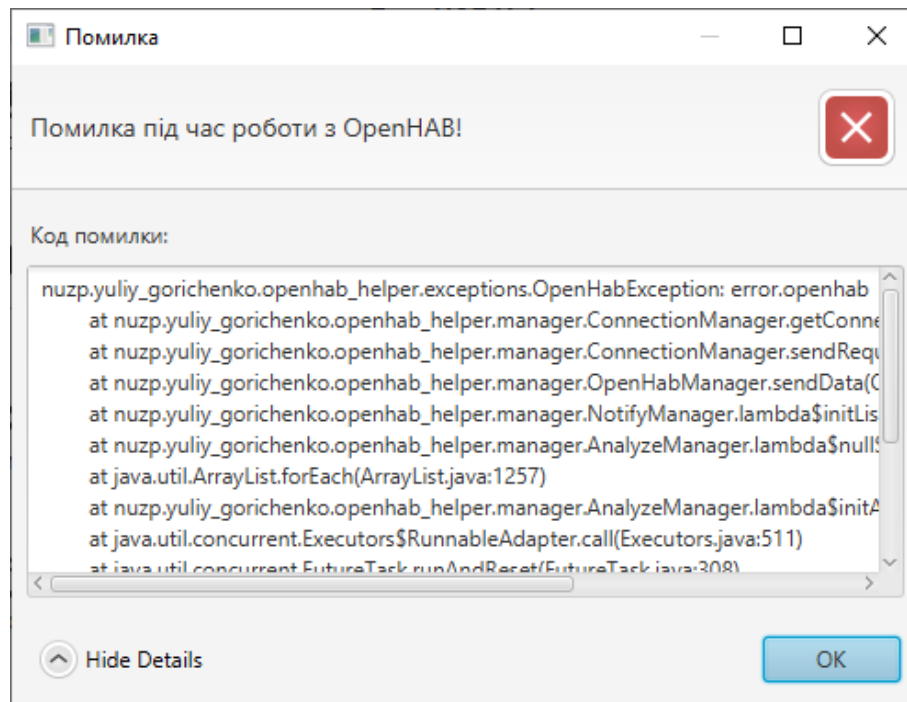


Рисунок 5.6 – Помилка під час роботи з OpenHAB

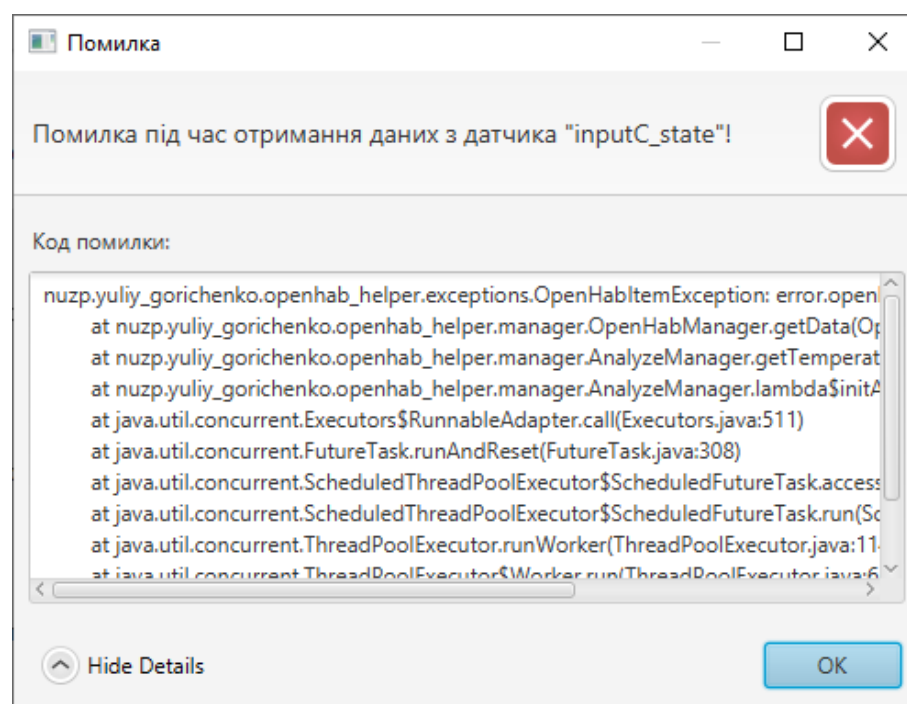


Рисунок 5.7 – Помилка під час отримання даних з датчика

Під час роботи з базою даних можуть виникати такі помилки:

- відсутнє з'єднання з базою даних (рис. 5.8), коли неможливо підключитися до бази даних через неправильно введені дані авторизації;
- помилка під час роботи з базою даних (рис. 5.9).

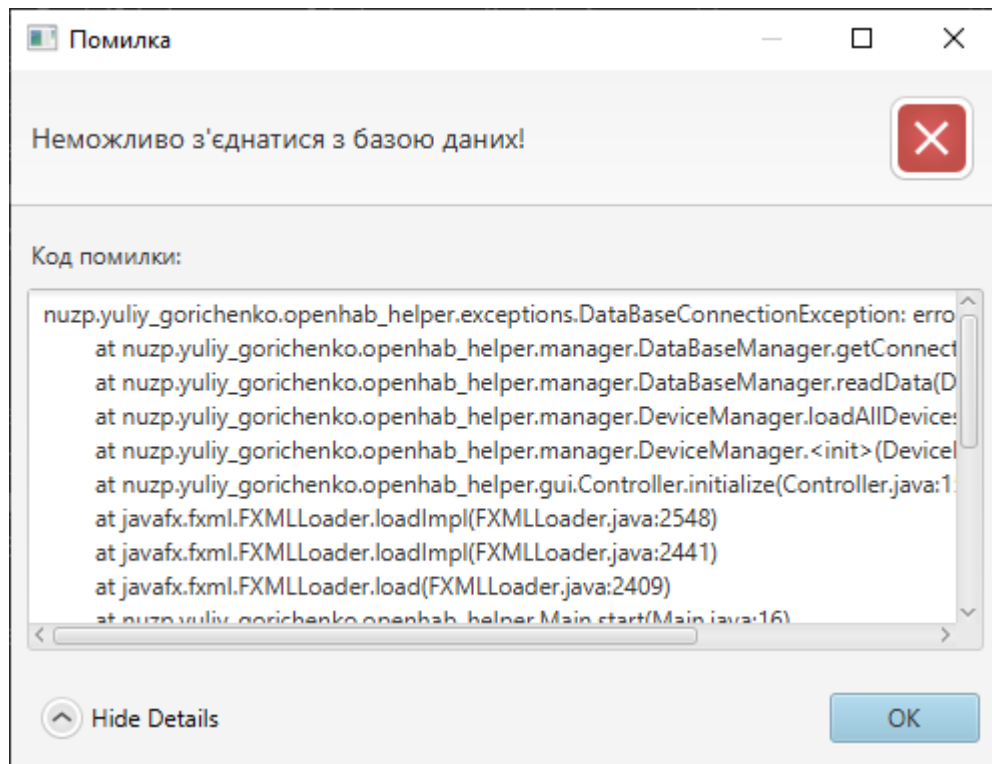


Рисунок 5.8 – Повідомлення, якщо неможливо з'єднатися з базою даних

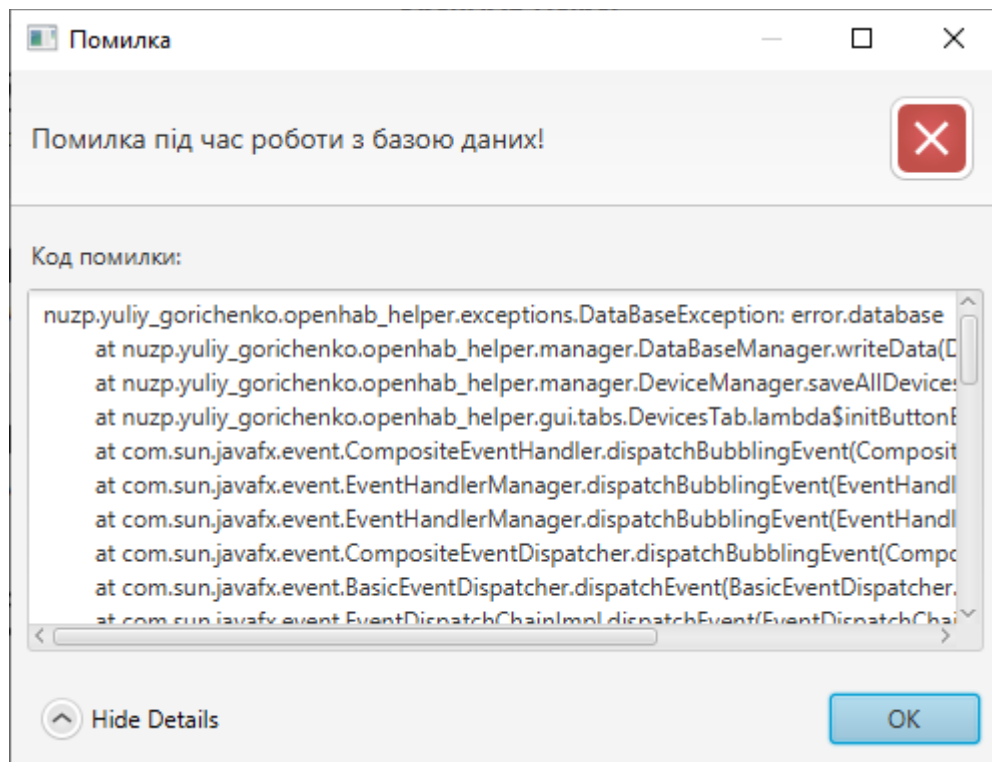


Рисунок 5.9 – Повідомлення у разі виникнення помилки під час роботи з базою даних

6 КЕРІВНИЦТВО КОРИСТУВАЧА

6.1 Призначення програми

Призначенням програми є аналіз даних щодо споживання електроенергії для підвищення безпеки та ефективності СДА. Це дозволить забезпечити надійний захист дротів та кабелів від перегріву, що може спричинити пожежу. Захист здійснюється методом контролю показників СДА, та у разі можливості виникнення небезпеки – власник будинку отримує повідомлення на мобільний застосунок у реальному часі з рекомендаціями щодо безпечного користування електроприладами.

6.2 Умови виконання програми

Умови застосування для коректної роботи програми:

- СДА на платформі OpenHAB, підключену до бази даних MySQL;
- використання останньої версії програмного застосунку;
- правильне налаштування програмного застосунку.

Для роботи програми необхідно мати встановлене виконавче середовище JRE версії 1.8 або вище, а також мати наступні апаратно-технічні засоби:

- процесор з частотою 1 ГГц та вище;
- операційна система Microsoft Windows 7 та вище / Red Hat Enterprise Linux 5.5 та вище / Ubuntu 12.04 та вище / macOS 10.9 та вище;
- оперативна пам'ять 1 ГБ та вище;
- 3 МБ вільного місця на жорсткому диску для зберігання виконуваного файлу (*.jar);
- відеокарта з підтримкою VGA (True Color) та екран роздільною здатністю 1024 x 768 пікселів та вище;
- клавіатура;
- маніпулятор «миша».

6.3 Виконання програми

Для початку роботи з програмою «OpenHAB Helper» потрібно запуснути виконуваний файл `openhav_helper.jar`. Після запуску з'явиться головне вікно, яке має чотири вкладки: «Головна», «Прилади», «Параметри», «Про програму». Користувач може перейти у будь-який розділ, натиснувши на потрібну вкладку.

6.3.1 Вкладка «Головна»

Інтерфейс вкладки «Головна» зображено на рисунку 6.1.

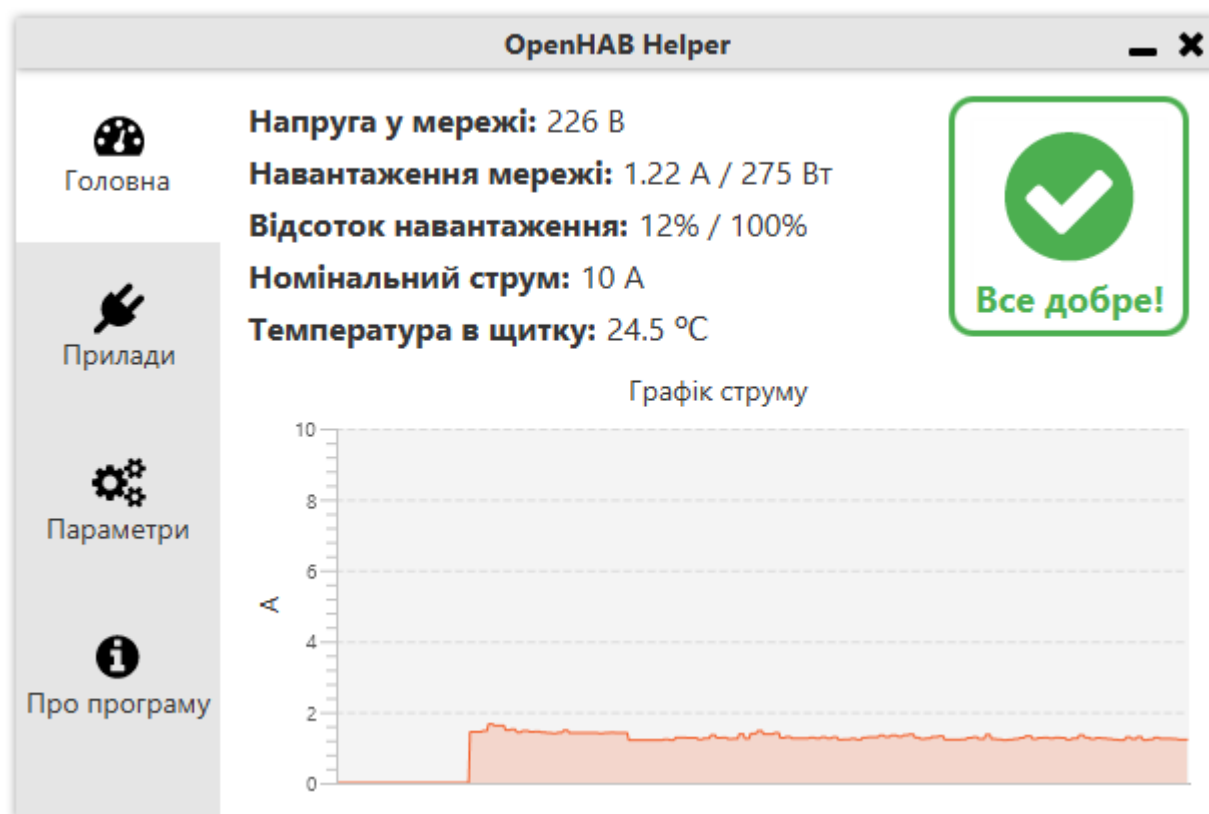


Рисунок 6.1 – Вкладка «Головна»

У цьому розділі користувач має змогу побачити всі показники електрощитового обладнання у реальному часі, а саме: напругу в електромережі, навантаження електромережі, відсоток навантаження,

номінальний струм та температуру ввідного автомата в електричному щиті. За допомогою графіку струму можна візуально побачити в який момент часу було навантаження в електромережі. Головний показник – це загальний статус мережі, який позначено прямокутником у правій верхній області.

Загальний статус може приймати три значення:

- «Все добре» позначається зеленим кольором та відображається у випадку, коли електромережа в порядку та немає ніяких загроз та небезпек (рис. 6.1);

- «Увага» позначається помаранчевим кольором та відображається у випадку, коли навантаження в електромережі перевищує поріг попередження, але ще не перевищує номінальне значення струму (рис. 6.2);

- «Небезпека» позначається червоним кольором та відображається у випадку, коли навантаження в електромережі перевищує номінальне значення струму та існує загроза виникнення пожежі (рис. 6.3).

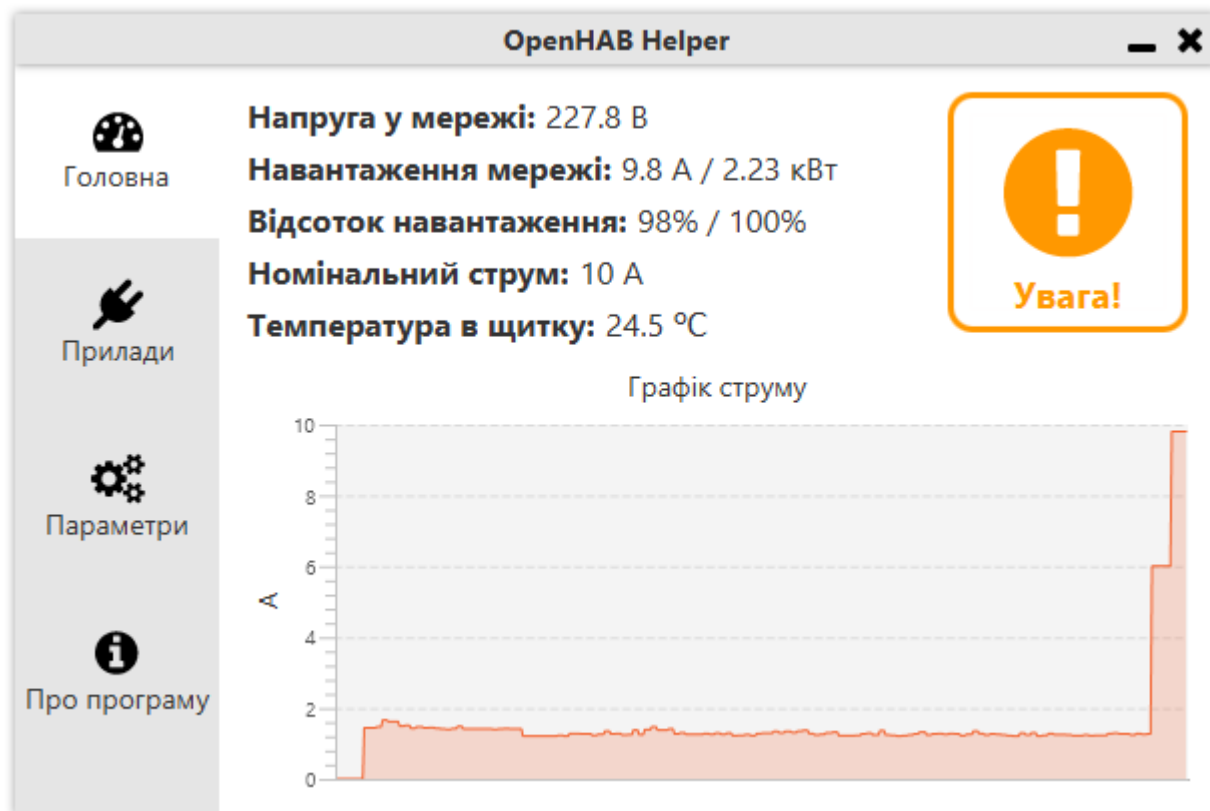


Рисунок 6.2 – Загальний статус електромережі «Увага»

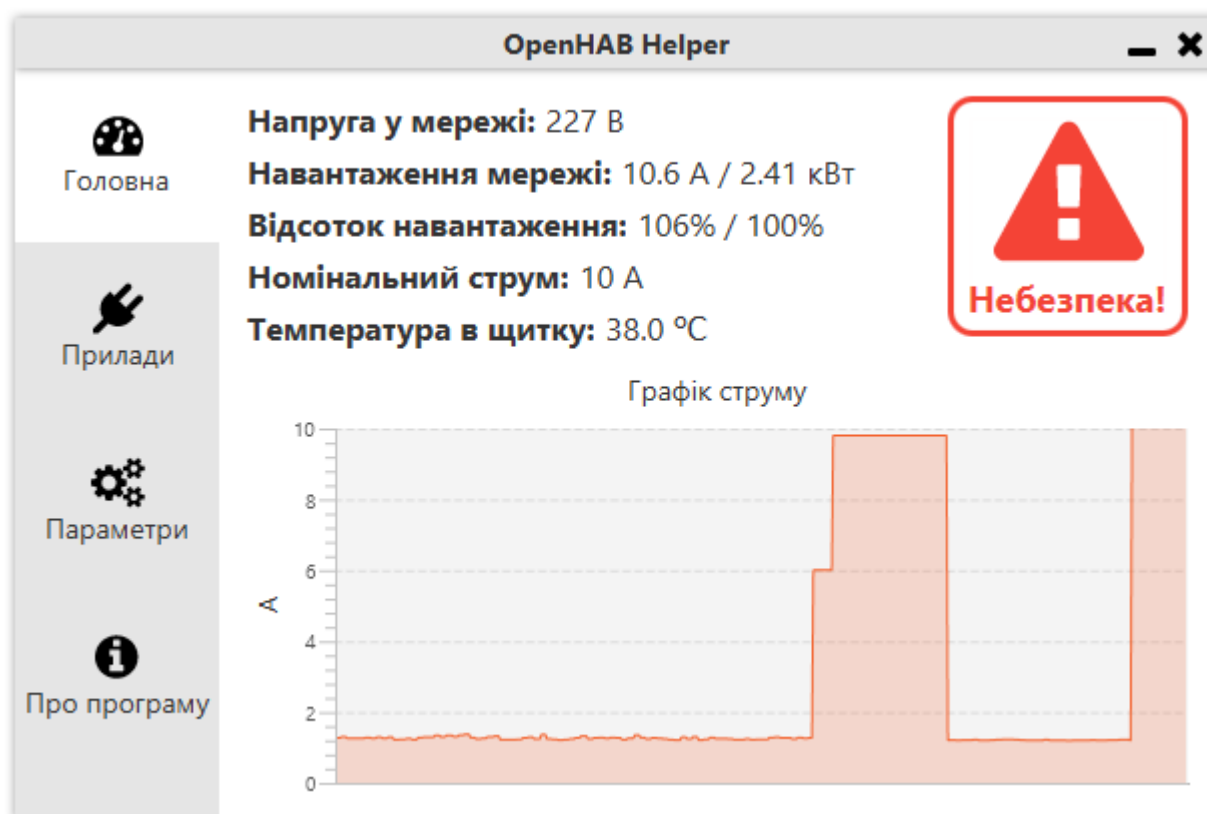


Рисунок 6.3 – Загальний статус електромережі «Небезпека»

6.3.2 Вкладка «Прилади»

Інтерфейс вкладки «Прилади» зображено на рисунку 6.4.

У цьому розділі користувач може побачити список увімкнених електроприладів у реальному часі, а також навантаження кожного з них.

Для того, щоб відредагувати список електроприладів, потрібно натиснути на кнопку «Редагувати», яка знаходиться у правій верхній частині інтерфейсу. Після чого відкриється редактор електроприладів (рис. 6.5).

Для того, щоб додати новий електроприлад, потрібно у текстові поля ввести необхідні дані, а саме: назва приладу, споживання струму (це значення можна знайти у паспорті самого приладу), назва актуатора (якщо цей прилад підключено до OpenHAB) та натиснути на кнопку зі знаком «+».

Для того, щоб видалити прилад зі списку, потрібно натиснути на червоний хрестик напроти його назви.

Для збереження списку, потрібно натиснути на кнопку «Зберегти».



Рисунок 6.4 – Вкладка «Прилади»

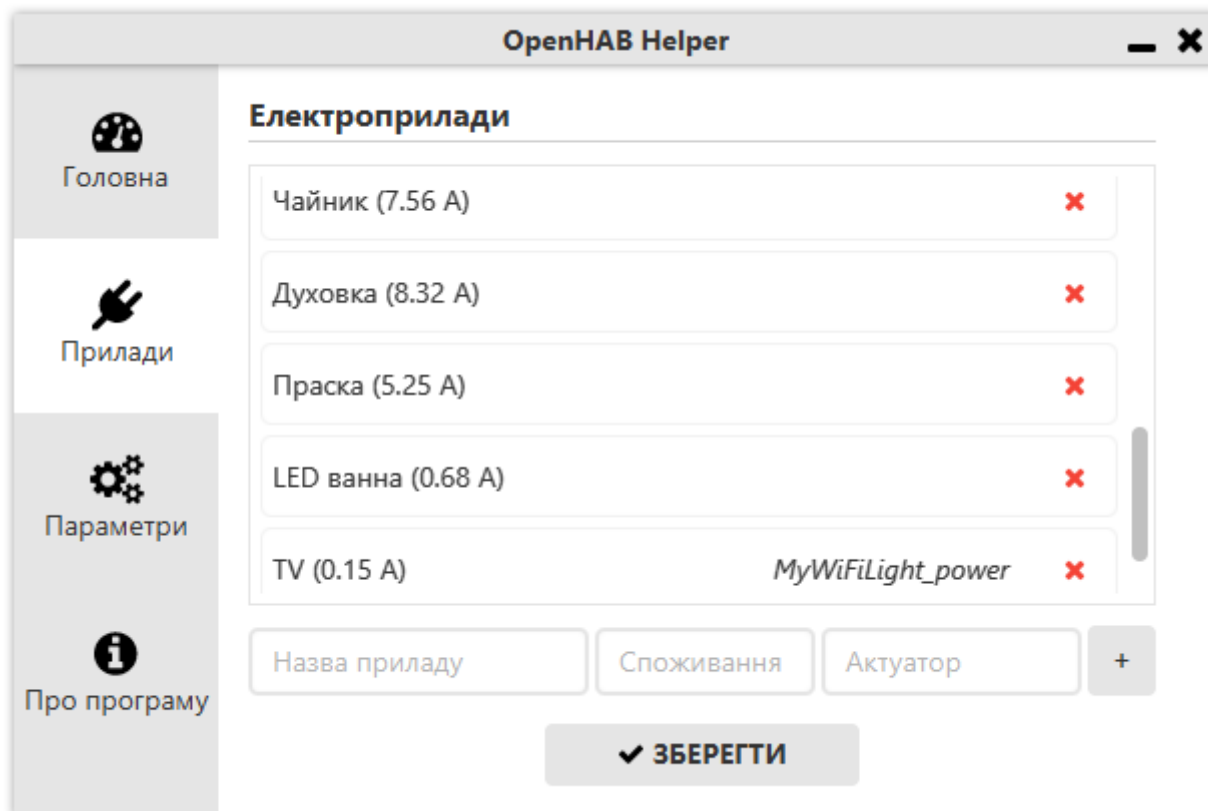


Рисунок 6.5 – Редактор списку електроприладів

6.3.3 Вкладка «Параметри»

Інтерфейс вкладки «Параметри» зображено на рисунку 6.6.

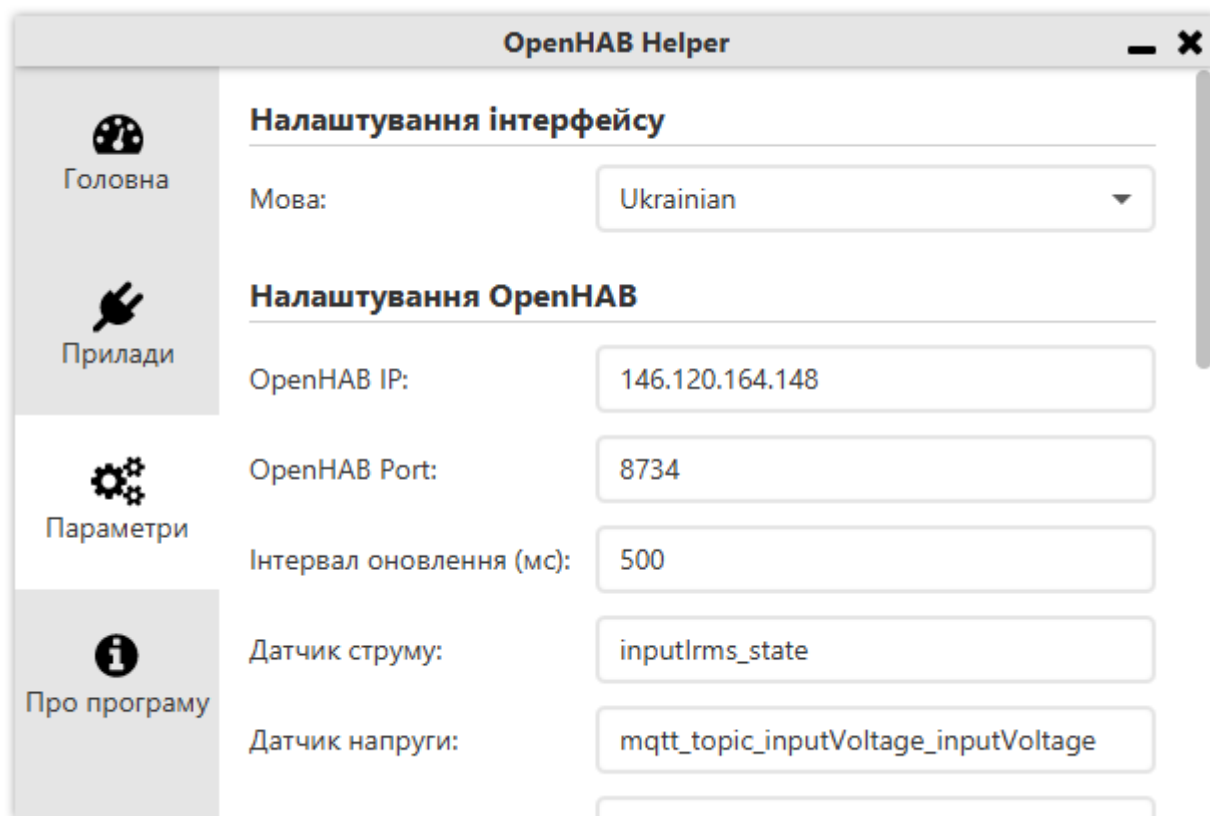


Рисунок 6.6 – Вкладка «Параметри»

У цій вкладці користувач може змінити будь-які налаштування програми, які містять три розділи: «Налаштування інтерфейсу», «Налаштування OpenHAB» та «Налаштування MySQL».

Розділ «Налаштування інтерфейсу» має випадний список «Мова», який дозволяє обрати мову інтерфейсу. Програмний застосунок підтримує англійську, українську та російську локалізації.

Розділ «Налаштування OpenHAB» дозволяє налаштувати параметри для роботи з платформою OpenHAB.

Розділ «Налаштування MySQL» дозволяє налаштувати параметри для роботи з базою даних MySQL.

6.3.4 Вкладка «Про програму»

Інтерфейс вкладки «Про програму» зображено на рисунку 6.7.

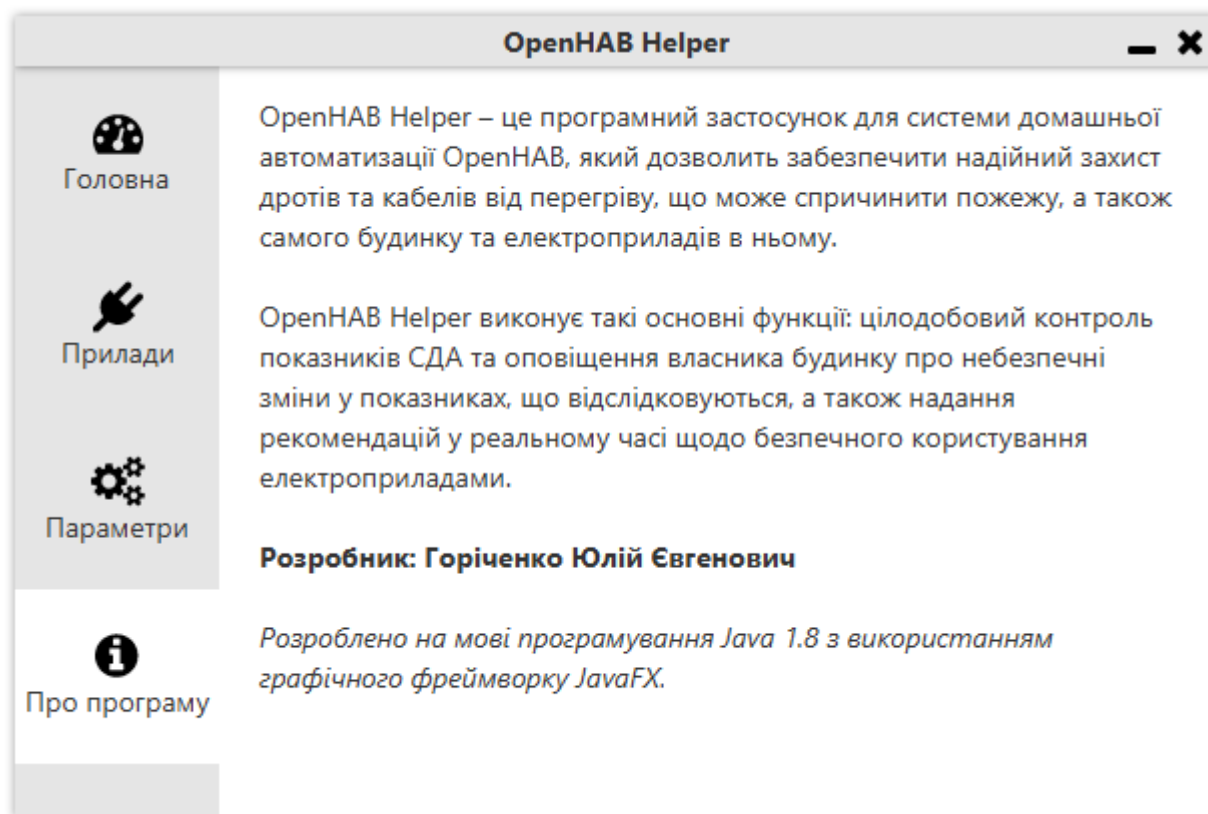


Рисунок 6.7 – Вкладка «Про програму»

У цьому розділі користувач може ознайомитися з призначенням програмного застосунку «OpenHAB Helper», його характеристиками та розробником.

6.4 Повідомлення

У разі можливості виникнення небезпеки або великого навантаження електромережі, власник будинку отримає повідомлення на мобільний застосунок у реальному часі з рекомендаціями щодо безпечного користування електроприладами.

Всі повідомлення поділяються на три групи та залежать від загального статусу електромережі:

- повідомлення для статусу «Увага» відправляється тоді, коли навантаження в електромережі перевищує поріг попередження, але ще не перевищує номінальне значення струму, також це повідомлення містить рекомендації про те, які електроприлади не можна вмикати, щоб не перевищити номінальний струм, що може призвести до перегріву кабелів і дротів та виникненню пожежі (рис. 6.8);

- повідомлення для статусу «Небезпека» відправляється тоді, коли навантаження в електромережі перевищує номінальне значення струму та існує загроза виникнення пожежі (рис. 6.9);

- повідомлення для статусу «Все добре» відправляється тоді, коли користувач виконав усі рекомендації щодо безпечного користування електроприладами та більше немає ніяких загроз та небезпек (рис. 6.10).

Також повідомлення може включати попередження про хибне або неможливе відключення електромережі, якщо температура ввідного автомата має велике відхилення від робочої температури (рис. 6.9).

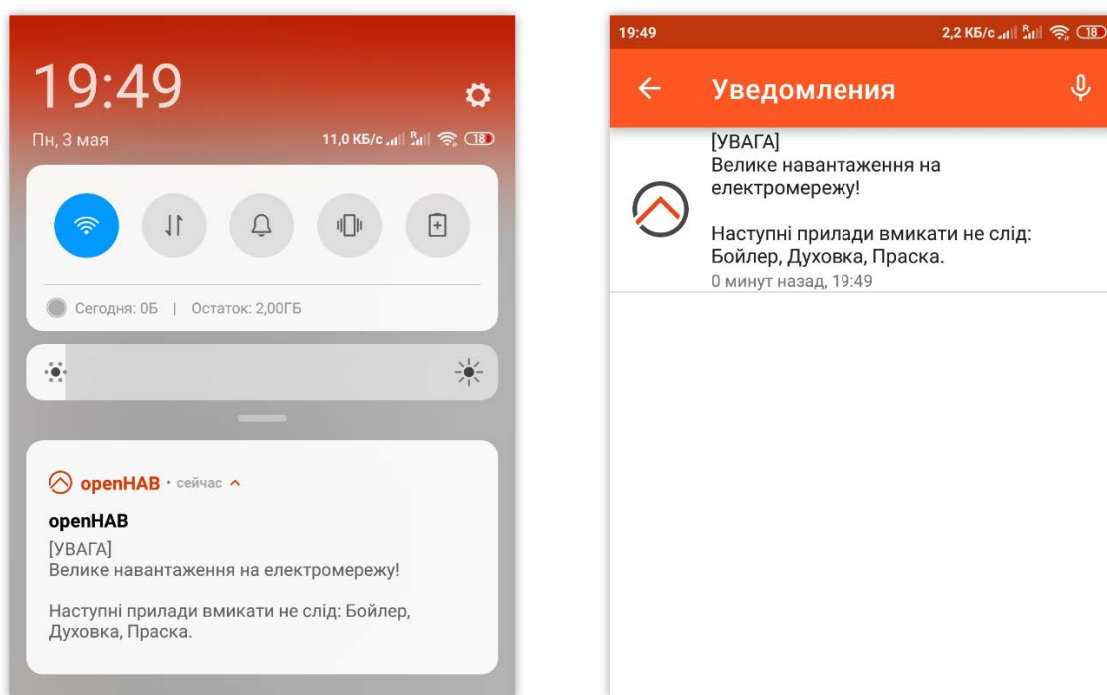


Рисунок 6.8 – Повідомлення у мобільному застосунку для статусу «Увага»

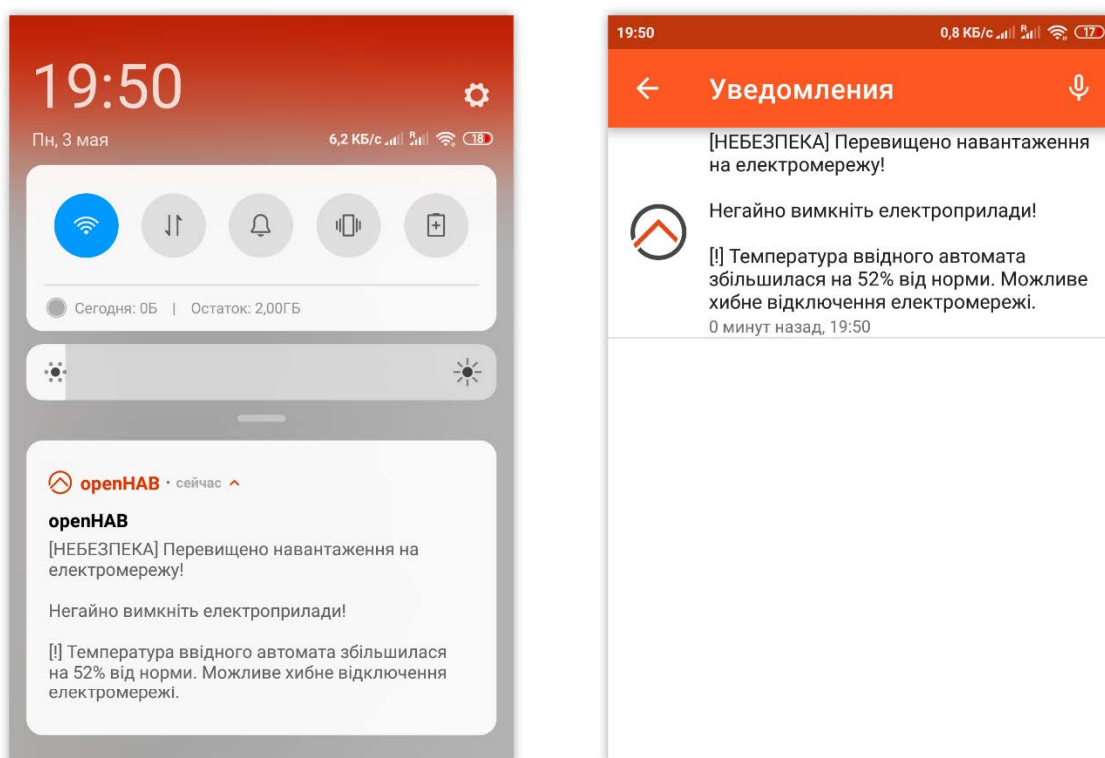


Рисунок 6.9 – Повідомлення у мобільному застосунку для статусу «Небезпека» з попередженням про хибне відключення електромережі

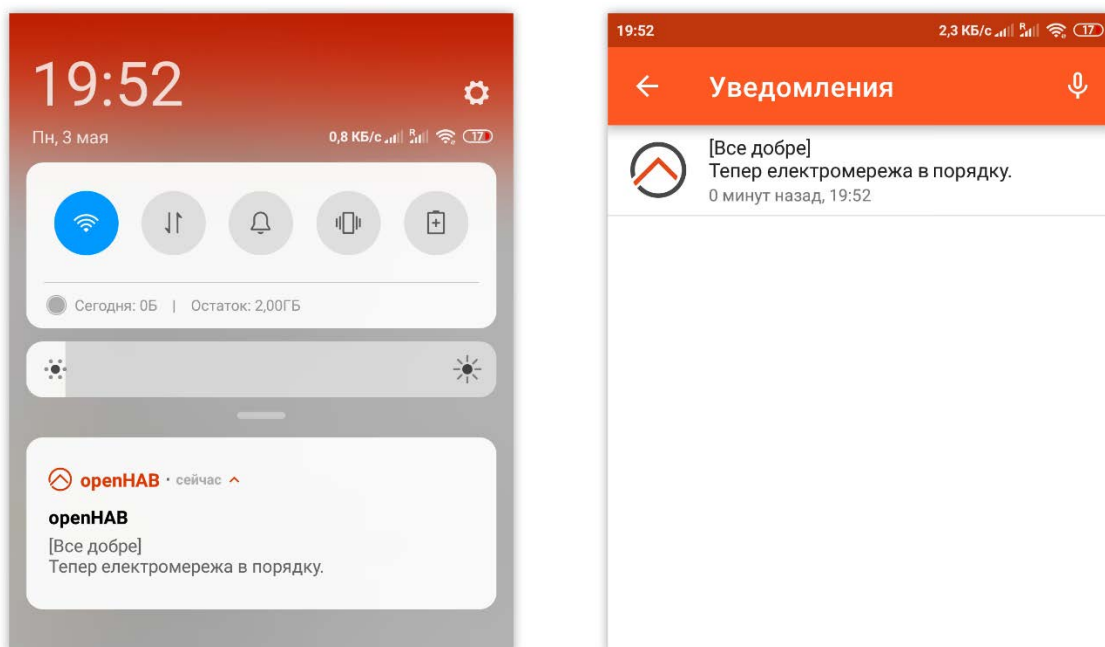


Рисунок 6.10 – Повідомлення у мобільному застосунку для статусу «Все добре»

ВИСНОВКИ

В результаті виконання дипломної роботи були вирішені наступні задачі:

- виконано огляд особливостей методів, алгоритмів та програмних засобів інтелектуального аналізу даних;
- досліджено особливості існуючих СДА на прикладі реальної однокімнатної квартири;
- досліджені методи та засоби організації таким систем, зокрема платформи OpenHAB;
- застосовано існуючий програмний засіб Orange з функціями інтелектуального аналізу для даних, що були накопичені з реальної СДА;
- розроблено архітектуру СДА;
- розроблено програмний застосунок для аналізу даних з СДА;
- розроблено програмну документацію.

Платформа OpenHAB допомагає об'єднати багато різних систем домашньої автоматизації, пристроїв та технологій в єдине рішення, а також надає можливості для моніторингу показників з датчиків та приладів, що використовуються. Але ця платформа не дозволяє контролювати навантаження в електромережі та не надає рекомендації щодо безпечного користування електроприладами, тому інтеграція застосунку «OpenHAB Helper» з платформою OpenHAB є доцільною.

Розроблене програмне забезпечення «OpenHAB Helper» дозволяє забезпечити надійний захист дротів та кабелів від перегріву, що може спричинити пожежу. Захист здійснюється методом контролю показників електрощитового обладнання, та у разі можливості виникнення небезпеки – власник будинку отримує повідомлення на мобільний застосунок у реальному часі з рекомендаціями щодо безпечного користування електроприладами.

У результаті проведення тестувань не було виявлено дефектів, які могли б викликати помилки в роботі програми або її аварійне завершення.

Розроблений програмний застосунок повністю відповідає вимогам технічного завдання.

У подальшій роботі планується впровадити алгоритми інтелектуального аналізу для надання рекомендацій щодо економії електроенергії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Smart Home [Electronic resource]. – Access mode: <https://www.investopedia.com/terms/s/smart-home.asp>.
2. Pranz, B. Smart Home with openHAB [Electronic resource] / B. Pranz, M. Schiller. – Access mode: <https://www.researchgate.net/publication/342171578>.
3. The smart home concept: our immediate future [Text] / [V. Riquebourg, D. Menga, D. Durand et al.] // E-Learning in Industrial Electronics: 1st IEEE International Conference, Hammamet, 18-20 December 2006: proceedings. – Hammamet: IEEE, 2006. – P. 23-28.
4. Machine learning and Data mining in Home Automation [Electronic resource]. – Access mode: <https://medium.com/@meetkumar/machine-learning-and-data-mining-in-home-automation-b06c136550a9>.
5. Yassine, A. Mining human activity patterns from smart home big data for health care applications [Text] / A. Yassine, S. Singh, A. Alamri // IEEE Access. – 2017. – Vol. 5. – P. 13131-13141.
6. Bhagwandin, D. Association Rule Mining of Household Electrical Energy Usage [Text] / D. Bhagwandin, V. Tripathi, P. Hosein // Data Mining, Communications and Information Technology : 1st International Conference, Phuket, 25-27 May 2017 : proceedings. – Phuket: DMCIT, 2017. – P. 1-5.
7. Towards energy efficiency smart buildings models based on intelligent data analytics [Text] / [A. González-Vidal, V. Moreno-Cano, F. Terroso-Sáenz et al.] // Procedia Computer Science. – 2016. – Vol. 83. – P. 994-999.
8. Lee, M. C. Privacy mining from IoT-Based smart homes [Text] / M. C. Lee, J. C. Lin, O. Owe // Broadband and Wireless Computing, Communication and Applications : 13th International Conference, Taichung, 27-29 October 2018 : proceedings. – Taichung: Springer, 2018. – P. 304-315.

9. SVM to detect the presence of visitors in a smart home environment [Text] / [J. Petersen, N. Larimer, J. A. Kaye et al.] // Medicine and Biology Society : 34th Annual International Conference of the IEEE Engineering, San Diego, 28 August-1 September 2012 : proceedings. – San Diego: IEEE, 2012. – P. 5850-5853.
10. 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.
11. Дюк, В. Data Mining : учебный курс [Текст] / В. Дюк, А. Самойленко. – СПб.: Питер, 2001. – 368 с.
12. What is a Data Mining? [Electronic resource]. – Access mode: <https://corporatefinanceinstitute.com/resources/knowledge/other/data-mining/>.
13. Data Mining от Oracle: настоящее и будущее [Электрон. ресурс]. – Режим доступа: http://citforum.ru/database/oracle/data_mining_solutions/.
14. Анализ данных и процессов : учеб. пособие [Текст] / [А. А. Барсегян, М. С. Куприянов, И. И. Холод и др.]. – 3-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2009. – 512 с.
15. Холюшкин, Ю. П. Web-система создания, использования и применения стратегий решения задач археологии : монография в журнале [Текст] / Ю. П. Холюшкин, Е. Е. Витяев, В. В. Мартынович. – Новосибирск: РИЦ НГУ, 2014. – 117 с.
16. Рафалович, В. Data mining, или интеллектуальный анализ данных для занятых: практический курс [Текст] / В. Рафалович. – М.: СмартБук, 2014. – 110 с.
17. Афанасьева, Т. В. Введение в проектирование систем интеллектуального анализа данных: учебное пособие [Текст] / Т. В. Афанасьева, А. Н. Афанасьев. – Ульяновск: УлГТУ, 2017. – 64 с.
18. Нестеров, С. А. Базы данных. Интеллектуальный анализ данных: учебное пособие [Текст] / С. А. Нестеров – СПб.: Изд-во Политехн. ун-та, 2011. – 272 с.

19. Чубукова, И. А. Data Mining : учебное пособие [Текст] / И. А. Чубукова. – 2-е изд., испр. – М.: ИНТУИТ; БИНОМ. Лаборатория знаний, 2008. – 382 с.
20. Cluster Analysis: Definition and Methods [Electronic resource]. – Access mode: <https://www.qualtrics.com/experience-management/research/cluster-analysis/>.
21. Метод ближайших соседей [Электрон. ресурс]. – Режим доступа: http://machinelearning.ru/wiki/index.php?title=Метод_ближайшего_соседа.
22. Шехтер, Д. Б. Байесовские методы в задаче оценки релевантности при поиске работы в Internet [Текст] / Д. Б. Шехтер, А. В. Чадюк, А. Л. Червинский-Ивашура // Проблемы програмування. – 2006. – № 2-3. – С. 519-525.
23. Линейный регрессионный анализ [Электрон. ресурс]. – Режим доступа: <https://www.statmethods.ru/statistics-metody/regressionnyj-analiz/>.
24. Data Mining – интеллектуальный анализ данных [Электрон. ресурс]. – Режим доступа: <https://blog.iteam.ru/data-mining-intellektualnyj-analiz-dannyh/>.
25. Data Mining Process [Electronic resource]. – Access mode: https://docs.oracle.com/cd/B19306_01/datamine.102/b14339/5dmtasks.htm.
26. Data Mining Concepts [Electronic resource]. – Access mode: <https://docs.microsoft.com/en/analysis-services/data-mining/data-mining-concepts>.
27. Поддержка ассоциативного правила [Электрон. ресурс]. – Режим доступа: <https://wiki.loginom.ru/articles/association-rule-support.html>.
28. Лифт ассоциативного правила [Электрон. ресурс]. – Режим доступа: <https://wiki.loginom.ru/articles/lift-of-association-rule.html>.
29. Обучение ассоциативным правилам [Электрон. ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Обучение_ассоциативным_правилам.
30. Busarov, V. A Comparative Analysis of Algorithms for Mining Frequent Itemsets [Text] / V. Busarov, N. Grafeeva, E. G. Mikhailova // Databases

and Information Systems: 12th International Baltic Conference, Riga, 4-6 July 2016: proceedings. – Riga: Springer, 2016. – P. 136-150.

31. Heaton, J. Comparing dataset characteristics that favor the Apriori, Eclat or FP-Growth frequent itemset mining algorithms [Text] / J. Heaton // Big Data and Warehouse Architecture: IEEE SoutheastCon 2016, Norfolk, 30 March-3 April 2016: proceedings. – Norfolk: IEEE, 2016. – P. 1-7.

32. Кириченко, Д. О. Оптимизация входных данных в задаче поиска шаблонов и ассоциативных правил [Текст] / Д. О. Кириченко, М. А. Артемов // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2014. – № 4. – С. 63-70.

33. Prithiviraj, P. A comparative analysis of association rule mining algorithms in data mining: a study [Text] / P. Prithiviraj, R. Porkodi // American Journal of Computer Science and Engineering Survey. – 2015. – Vol. 3, Issue 1. – P. 98-119.

34. Jia, L. An Improved Eclat Algorithm Based on Tissue-Like P System with Active Membranes [Text] / L. Jia, L. Xiang, X. Liu // Processes. – 2019. – Vol. 7, Issue 9. – P. 1-15.

35. Association Rule Mining using ECLAT Algorithm [Electronic resource]. – Access mode: <https://medium.com/machine-learning-and-artificial-intelligence/3-4-association-rule-mining-using-eclat-algorithm-b6e50aab2147>.

36. Data analysis [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Data_analysis.

37. The Top 10 Data Mining Tools of 2018 [Electronic resource]. – Access mode: <https://www.analyticsinsight.net/the-top-10-data-mining-tools-of-2018/>.

38. 10 Best Data Analysis Tools for Perfect Data Management [Electronic resource]. – Access mode: <https://www.softwaretestinghelp.com/data-analysis-tools/>.

39. RapidMiner Studio in 2021 [Electronic resource]. – Access mode: <https://www.predictiveanalyticstoday.com/rapidminer/>.

40. Orange Data Mining [Electronic resource]. – Access mode: <https://github.com/biolab/orange3>.
41. The WEKA data mining software: an update [Text] / [M. Hall, E. Frank, G. Holmes et al.] // ACM SIGKDD explorations newsletter. – 2009. – Vol. 11, Issue 1. – P. 10-18.
42. Krotov, V. A Quick Introduction to R and RStudio : tutorial [Text] / V. Krotov. – Murray: Murray State University, 2017. – 16 p.
43. API design for machine learning software: experiences from the scikit-learn project [Text] / [L. Buitinck, G. Louppe, M. Blondel et al.] // Machine Learning and Principles and Practices of Knowledge Discovery in Databases: European Conference, Prague, 23-27 September, 2013: proceedings. – Prague: ECML/PKDD, 2013. – P. 1-15.
44. Orange Data Mining – Widget catalog [Electronic resource]. – Access mode: <https://orangedatamining.com/widget-catalog/>.
45. Review on the application of artificial intelligence in smart homes [Text] / [X. Guo, Z. Shen, Y. Zhang et al.] // Smart Cities. – 2019. – Vol. 2, Issue 3. – P. 402-420.
46. openHAB: What Advantages Does This Home Automation Platform Offer? [Electronic resource]. – Access mode: <https://teks.co.in/site/blog/openhab-what-advantages-does-this-home-automation-platform-offer/>.
47. What is Wi-Fi and why is it so important? [Electronic resource]. – Access mode: <https://www.networkworld.com/article/3560993/what-is-wi-fi-and-why-is-it-so-important.html>.
48. What Is Bluetooth? [Electronic resource]. – Access mode: <https://www.lifewire.com/what-is-bluetooth-2377412>.
49. BLE (Bluetooth Low Energy) Definition [Electronic resource]. – Access mode: <https://techterms.com/definition/ble>.
50. Orange Pi – железо, распиновка, инструкции как установить ОС [Электрон. ресурс]. – Режим доступа: <https://arduinomaster.ru/raspberry-pi/orange-pi-konkurent-ili-podrazhatel-dlya-raspberry-pi/>.

51. MQTT, что это такое? Немного теории [Электрон. ресурс]. – Режим доступа: <https://sprut.ai/client/article/234>.
52. Новацький, А. О. Проектування та програмування мікропроцесорних систем і мереж: Проектування мережі 1-WIRE : навчальний посібник для студентів спеціальностей 7.05020101, 8.05020101 «Комп'ютеризовані системи управління та автоматика» кафедри автоматики та управління в технічних системах [Текст] / А. О. Новацький. – К.: НТУУ «КПІ», 2014. – 141 с.
53. What Is MySQL [Electronic resource]. – Access mode: <https://www.mysqltutorial.org/what-is-mysql/>.
54. Почему в жару отключаются автоматы? [Электрон. ресурс]. – Режим доступа: <https://electricvdome.ru/avtomaticheskie-vikluchатели/v-zharu-srabatyvaet-avtomat.html>.
55. What is an API? [Electronic resource]. – Access mode: <https://apifriends.com/api-management/what-is-an-api/>.
56. What is a REST API? [Electronic resource]. – Access mode: <https://apifriends.com/api-management/what-is-rest-api/>.
57. Уніфікований ідентифікатор ресурсів – Вікіпедія [Електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/URI>.
58. openHAB REST API [Electronic resource]. – Access mode: <https://www.openhab.org/docs/configuration/restdocs.html>.
59. The most popular programming languages in 2021 [Electronic resource]. – Access mode: <https://daryl.solutions/the-most-popular-programming-languages-in-2021/>.
60. Comparative Analysis of Python and Java for Beginners [Text] / [M. S. Khoirom, M. Sonia, B. Laikhuram et al.] // International Research Journal of Engineering and Technology. – 2020. – Vol. 7, Issue 8. – P. 4384-4407.
61. What is JavaScript Used for? [Electronic resource]. – Access mode: <https://www.hackreactor.com/blog/what-is-javascript-used-for>.

62. C# vs Java: Differences you should Know [Electronic resource]. – Access mode: <https://hackr.io/blog/c-sharp-vs-java>.
63. Alkadi, I. A Comparative Analysis of C# and Java as an Introductory Programming Language for Information Systems Students [Text] / I. Alkadi, G. Alkadi, H. Etheridge // Review of Business Information Systems (RBIS). – 2006. – Vol. 10, Issue 4. – P. 37-40.
64. JavaFX vs Swing | Java GUI Libraries (2020) [Electronic resource]. – Access mode: <https://coderslegacy.com/javafx-vs-swing/>.
65. What is Framework in Java [Electronic resource]. – Access mode: <https://www.javatpoint.com/what-is-framework-in-java>.
66. Sharan, K. Java APIs, Extensions and Libraries: With JavaFX, JDBC, jmod, jlink, Networking, and the Process API [Text] / K. Sharan. – 2nd edition. – New York: Apress, 2018. – 813 p.
67. Различия между MVVM и остальными MV*-паттернами [Электрон. ресурс]. – Режим доступа: <https://habr.com/ru/post/313538/>.
68. Паттерны для новичков: MVC vs MVP vs MVVM [Электрон. ресурс]. – Режим доступа: <https://habr.com/ru/post/215605/>.
69. Top 10 Best Java IDEs for 2021 [Electronic resource]. – Access mode: <https://www.educative.io/blog/best-java-ides-2021>.
70. Deters, J. Jerady / fontawesomefx – Bitbucket [Electronic resource] / J. Deters. – Access mode: <https://bitbucket.org/Jerady/fontawesomefx/>.

ДОДАТОК А
ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Метою дипломної роботи є розробка програмного застосунку, який виконує аналіз даних щодо споживання електроенергії для підвищення безпеки та ефективності СДА. Це дозволить забезпечити надійний захист дротів та кабелів від перегріву, що може спричинити пожежу, а також самого будинку та електроприладів в ньому.

Розроблене програмне забезпечення виконує такі основні функції: цілодобовий контроль показників СДА та оповіщення власника будинку про небезпечні зміни у показниках, що відслідковуються, а також надання рекомендацій у реальному часі щодо безпечного користування електроприладами.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну роботу бакалавра за темою «Програмне забезпечення для аналізу даних системи домашньої автоматизації», затверджену наказом № 103 від 30 березня 2021 року по Національному університету «Запорізька політехніка».

A.2 Призначення розробки

Призначенням програми є аналіз даних щодо споживання електроенергії для підвищення безпеки та ефективності СДА. Це дозволить забезпечити надійний захист дротів та кабелів від перегріву, що може спричинити пожежу, виконуючи цілодобовий моніторинг показників платформи OpenHAB та негайного оповіщення власника про небезпечні зміни у системі з наданням рекомендацій щодо безпечного користування електроприладами.

A.3 Вимоги до програмного застосунку

A.3.1 Вимоги до функціональних характеристик

Програмний застосунок повинен забезпечувати виконання наступних функцій:

- виконувати моніторинг показників роботи електрощитового обладнання;
- виводити список увімкнених приладів та їхнє споживання в електромережі;
- можливість редагування списку електроприладів у будинку;
- збереження списку електроприладів у базі даних;
- можливість зміни налаштувань програми;
- збереження налаштувань програми;
- оповіщення власника будинку про небезпечні зміни у показниках, що відслідковуються;
- надання рекомендацій у реальному часі щодо безпечного користування електроприладами.

A.3.2 Вимоги до надійності

Програма повинна розпізнавати ситуації нерозуміння користувача з програмою при введенні некоректних даних, виводити відповідні повідомлення при знаходженні помилок.

Програмний застосунок повинен забезпечувати цілодобову роботу.

A.3.3 Умови експлуатації

Програмний застосунок може бути записаний як на компакт-диску, так і на флеш-накопичувачі.

Експлуатація програмного застосунку здійснюється відповідно до експлуатаційної документації для даного програмного застосунку, що відповідає стандартам і містить інформацію, необхідну для освоєння та експлуатації програмного продукту.

А.3.4 Вимоги до складу та параметрів технічних засобів

Для надійного функціонування програми необхідно мати наступні апаратно-технічні засоби:

- процесор з частотою 1 ГГц та вище;
- операційна система Microsoft Windows 7 та вище / Red Hat Enterprise Linux 5.5 та вище / Ubuntu 12.04 та вище / macOS 10.9 та вище;
- оперативна пам'ять 1 ГБ та вище;
- 3 МБ вільного місця на жорсткому диску для зберігання виконуваного файлу (*.jar);
- відеокарта з підтримкою VGA (True Color) та екран роздільною здатністю 1024 x 768 пікселів та вище;
- клавіатура;
- маніпулятор «миша».

А.3.5 Вимоги до інформаційної та програмної сумісності

Розроблене програмне забезпечення повинно взаємодіяти з платформою OpenHAB, використовуючи REST API. Для взаємодії з базою даних, яка входить до складу СДА, повинен використовуватися протокол відповідної системи управління базами даних.

А.3.6 Вимоги до маркування й упакування

Програмний застосунок повинен зберігатися на жорсткому диску або поставлятися на цифрових носіях інформації.

А.3.7 Вимоги до транспортування та збереження

Вимоги до транспортування та збереження аналогічні вимогам, пропонованим до відповідних носіїв інформації.

А.3.8 Вимоги до програмної документації

Програмний застосунок повинен поставлятися з «Керівництвом користувача» і «Керівництвом програміста».

А.3.9 Спеціальні вимоги

Взаємодія користувача з програмним застосунком має бути реалізована за допомогою графічного інтерфейсу. Інтерфейс програми повинен бути простим і зручним у використанні, а також не повинен бути перевантажений графічними елементами. Інтерфейс повинен відповідати сучасним ергономічним вимогам та забезпечувати зручний доступ до всіх функцій програми. Графічний інтерфейс повинен бути розрахований на переважне використання маніпулятора типу «миша». Клавіатурний режим введення повинен використовуватися тільки при заповненні або редагуванні текстових і числових полів.

Інтерфейс програми повинен мати українську, англійську та російську локалізацію.

На головній формі повинні відображатися всі показники СДА, а також загальний статус електромережі («Все добре», «Увага», «Небезпека»).

ДОДАТОК Б
ТЕКСТ ПРОГРАМИ

Б.1 Файл AnalyzeManager.java

```
package nuzp.yuliy_gorichenko.openhab_helper.manager;

import javafx.application.Platform;
import nuzp.yuliy_gorichenko.openhab_helper.exceptions.*;
import nuzp.yuliy_gorichenko.openhab_helper.utils.*;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;

import static nuzp.yuliy_gorichenko.openhab_helper.manager.
    DeviceManager.Device;

public class AnalyzeManager {
    private final SettingsManager settingsManager;
    private final LocalizationManager localizationManager;
    private final OpenHabManager openHabManager;
    private final DeviceManager deviceManager;
    private final ScheduledExecutorService
        scheduledExecutorService;

    private StatusType lastStatus;
    private final List<StatusListener> statusListeners;
    private final List<ElectricListener> electricListeners;
    private final List<TemperatureListener>
        temperatureListeners;
    private final List<RecognizedDevicesListener>
        recognizedDevicesListeners;

    private double currentAmperage;
    private double currentVoltage;
    private double currentTemperature;
    private HashMap<Device, Boolean> enabledDevices;
    private LinkedHashMap<Device, Boolean> recognizedDevices;

    /**
     * Status Type
     */
    public enum StatusType {
        GOOD,
        WARNING,
        DANGER,
        CONNECTING,
        ERROR
    }
}
```

```

/**
 * Constructor
 */
public AnalyzeManager(SettingsManager settingsManager,
                      LocalizationManager localizationManager,
                      OpenHabManager openHabManager, DeviceManager
                      deviceManager) {
    this.settingsManager = settingsManager;
    this.localizationManager = localizationManager;
    this.openHabManager = openHabManager;
    this.deviceManager = deviceManager;
    this.scheduledExecutorService =
        Executors.newSingleThreadScheduledExecutor();
    this.lastStatus = StatusType.CONNECTING;
    this.statusListeners = new ArrayList<>();
    this.electricListeners = new ArrayList<>();
    this.temperatureListeners = new ArrayList<>();
    this.recognizedDevicesListeners = new ArrayList<>();
    this.currentAmperage = 0;
    this.currentTemperature = 0;
    this.enabledDevices = new HashMap<>();
    this.recognizedDevices = new LinkedHashMap<>();
    this.initAnalyzer();
}

/**
 * Init OpenHAB data analyzer
 */
private void initAnalyzer() {
    if (!settingsManager.isSettingsDone()) {
        return;
    }

    // run in new thread
    scheduledExecutorService.scheduleAtFixedRate(() -> {
        final double warning =
            settingsManager.getMaxAmperage() *
            settingsManager.getWarningPercentage() / 100.0;
        StatusType currentStatus = null;

        try {
            // retrieve data from OpenHAB
            currentAmperage = getAmperage();
            currentVoltage = getVoltage();
            currentTemperature = getTemperature();
            enabledDevices = getEnabledDevices();
            recognizedDevices = getRecognizedDevices();

            // get current status
            if (currentAmperage >= 0 && currentAmperage <
                warning) {
                currentStatus = StatusType.GOOD;
            } else if (currentAmperage >= warning &&

```

```

        currentAmperage <
            settingsManager.getMaxAmperage()) {
            currentStatus = StatusType.WARNING;
        } else if (currentAmperage >=
            settingsManager.getMaxAmperage()) {
            currentStatus = StatusType.DANGER;
        }

        // notify all listeners
        electricListeners.forEach(listener ->
            listener.onUpdate(currentAmperage,
                             currentVoltage));
        temperatureListeners.forEach(listener ->
            listener.onUpdate(currentTemperature));
        recognizedDevicesListeners.forEach(listener ->
            listener.onUpdate(recognizedDevices));
    } catch (OpenHabItemException |
            OpenHabConnectionException e) {
        currentStatus = StatusType.ERROR;
        Platform.runLater(() ->
            DialogUtils.showErrorMessage(e,
                                         localizationManager));
    }

    // notify status listener on change
    if (lastStatus != currentStatus) {
        lastStatus = currentStatus;
        statusListeners.forEach(listener ->
            listener.onUpdate(lastStatus));
    }
}, settingsManager.getUpdateInterval(),
    settingsManager.getUpdateInterval(),
    TimeUnit.MILLISECONDS);
}

/**
 * Get analyzer thread
 */
public ScheduledExecutorService getAnalyzerThread() {
    return scheduledExecutorService;
}

/**
 * Get current amperage
 */
private double getAmperage() throws OpenHabItemException,
    OpenHabConnectionException {
    return Double.parseDouble(openHabManager.getData(
        settingsManager.getItemAmperage()));
}

```

```

/**
 * Get current voltage
 */
private double getVoltage() throws OpenHabItemException,
                                OpenHabConnectionException {
    return Double.parseDouble(openHabManager.getData(
        settingsManager.getItemVoltage()));
}

/**
 * Get current temperature
 */
private double getTemperature() throws OpenHabItemException,
                                       OpenHabConnectionException {
    return Double.parseDouble(openHabManager.getData(
        settingsManager.getItemTemperature()));
}

/**
 * Get enabled devices
 */
private HashMap<Device, Boolean> getEnabledDevices() throws
    OpenHabItemException, OpenHabConnectionException {
    HashMap<Device, Boolean> enabledDevices =
        new HashMap<>();

    for (Device device : deviceManager.getAllDevices()) {
        if (!device.getItem().isEmpty()) {
            String state = openHabManager.getData(
                device.getItem());
            enabledDevices.put(device,
                state.equalsIgnoreCase("ON"));
        }
    }

    return enabledDevices;
}

/**
 * Get recognized devices
 */
private LinkedHashMap<Device, Boolean>
    getRecognizedDevices() {
    LinkedHashMap<Device, Boolean> recognizedDeices =
        new LinkedHashMap<>();

    DeviceRecognizer recognizer = new DeviceRecognizer(
        deviceManager.getAllDevices(), currentAmperage,
        enabledDevices, settingsManager.getMinAmperage());
    recognizer.recognize();
    for (Device device : recognizer.getFinalDevices()) {
        recognizedDeices.put(device, true);
    }
    for (Device device : deviceManager.getAllDevices()) {

```

```

        if (!recognizer.getFinalDevices().contains(device)){
            recognizedDeices.put(device, false);
        }
    }

    return recognizedDeices;
}

/**
 * Get current amperage value
 */
public double getCurrentAmperage() {
    return currentAmperage;
}

/**
 * Get current voltage value
 */
public double getCurrentVoltage() {
    return currentVoltage;
}

/**
 * Get current temperature value
 */
public double getCurrentTemperature() {
    return currentTemperature;
}

/**
 * Get current recognized devices
 */
public LinkedHashMap<Device, Boolean>
    getCurrentRecognizedDevices() {
    return recognizedDevices;
}

/**
 * Current status listener
 */
public interface StatusListener {
    void onUpdate(StatusType statusType);
}

/**
 * Add current status listener
 */
public void addStatusListener(StatusListener listener) {
    this.statusListeners.add(listener);
}

```

```

/**
 * Current electric listener
 */
public interface ElectricListener {
    void onUpdate(double amperage, double voltage);
}

/**
 * Add current electric listener
 */
public void addElectricListener(ElectricListener listener) {
    this.electricListeners.add(listener);
}

/**
 * Current temperature listener
 */
public interface TemperatureListener {
    void onUpdate(double temperature);
}

/**
 * Add current temperature listener
 */
public void addTemperatureListener(TemperatureListener
                                   listener) {
    this.temperatureListeners.add(listener);
}

/**
 * Current recognized devices
 */
public interface RecognizedDevicesListener {
    void onUpdate(LinkedHashMap<Device, Boolean> devices);
}

/**
 * Add current enabled devices listener
 */
public void addRecognizedDevicesListener(
    RecognizedDevicesListener listener) {
    this.recognizedDevicesListeners.add(listener);
}
}

```

ДОДАТОК В
СЛАЙДИ ПРЕЗЕНТАЦІЇ

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота бакалавра

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ДАНИХ СИСТЕМИ ДОМАШНЬОЇ АВТОМАТИЗАЦІЇ

Виконав:

ст. гр. КНТ-227

Ю. Є. Горіченко

Керівник:

к.т.н., доцент

А. В. Пархоменко

Рисунок В.1 – Слайд №1



Рисунок В.2 – Слайд №2

МЕТА ТА ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

Метою роботи є розробка програмного застосунку для аналізу даних щодо споживання електроенергії для підвищення безпеки та ефективності системи домашньої автоматизації (СДА).

Задачі дипломної роботи:

- виконати огляд особливостей методів, алгоритмів та програмних засобів інтелектуального аналізу даних;
- застосувати програмний інструмент з функціями інтелектуального аналізу для даних, що були накопичені з СДА;
- удосконалити архітектуру СДА;
- розробити програмний застосунок для аналізу даних з СДА;
- розробити програмну документацію.

3

Рисунок В.3 – Слайд №3

ОГЛЯД ОСОБЛИВОСТЕЙ ІАД ДЛЯ СДА



4

Рисунок В.4 – Слайд №4

АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ІАД

Характеристика	Rapid Miner	Orange	Weka	R	scikit-learn
Безкоштовний тип ліцензії	—	+	+	+	+
Останнє оновлення	Березень 2020 р.	Березень 2021 р.	Вересень 2015 р.	Березень 2021 р.	Січень 2021 р.
Мова програмування	Java	Python	Java	R	Python
Графічний інтерфейс	+	+	+	—	—
Створення скриптів	—	+	+	+	+
Рейтинг користувачів	9.5 / 10.0	9.5 / 10.0	9.1 / 10.0	9.1 / 10.0	7.6 / 10.0

5

Рисунок В.5 – Слайд №5

ОБЧИСЛЮВАЛЬНІ ЕКСПЕРИМЕНТИ У ПРОГРАМІ «ORANGE»

Кількість проведених експериментів:

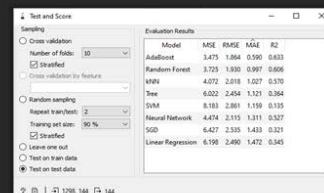
- 4 обчислювальні експерименти.

Типи експериментів:

- пошук асоціативних правил;
- прогнозування.

Використані вибірки:

- вибірка за весь період (4 місяці);
- вибірка за обмежений період (5 днів);
- вибірка з розбиттям на частини доби (сніданок, день, вечір, ніч).



Порівняння точності прогнозування

Antecedent	Consequent
Bath-to-21 - 22, Kitchen Lights= ON, Electric-A=1 - 2	TV Lights= OFF
Kitchen Lights= ON, TV Lights= OFF, Electric-A=1 - 2	Bath-to-21 - 22
Bath-to-21 - 22, TV Lights= OFF, Curtains Lights= ON	Kitchen Lights= ON
Bath-to-21 - 22, Kitchen Lights= ON, Curtains Lights= ON	TV Lights= OFF
Kitchen Lights= ON, TV Lights= OFF, Curtains Lights= ON	Bath-to-21 - 22

Результати пошуку асоціативних правил

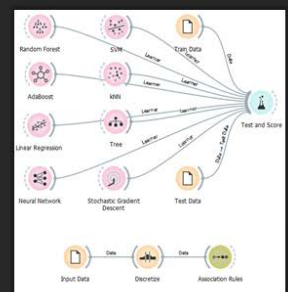


Схема в Orange для проведення експериментів

6

Рисунок В.6 – Слайд №6



Рисунок В.7 – Слайд №7



Рисунок В.8 – Слайд №8

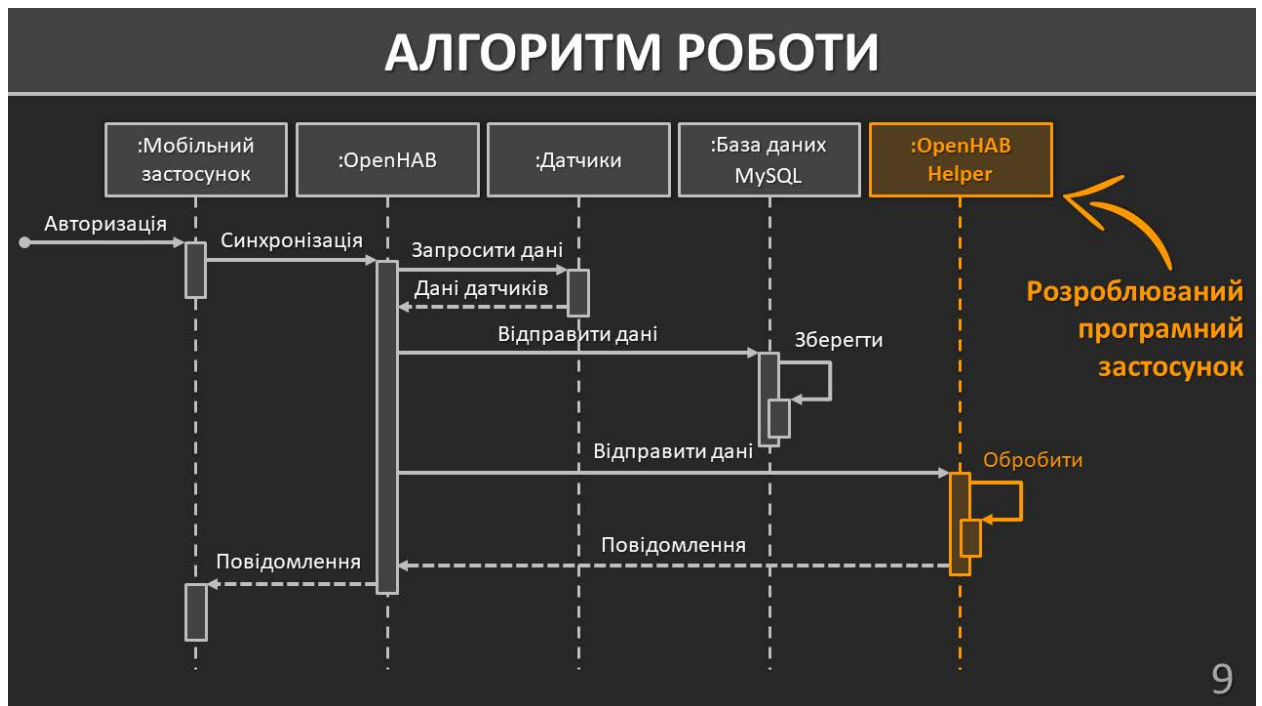


Рисунок В.9 – Слайд №9



Рисунок В.10 – Слайд №10

ВИБІР ІНСТРУМЕНТАРІЮ РОЗРОБКИ

ПОРІВНЯННЯ ГРАФІЧНИХ ФРЕЙМВОРКІВ

Характеристика	Swing	JavaFX
Рік створення	1997	2008
Візуальний конструктор для створення графічного інтерфейсу	—	+
Підтримка CSS	—	+
Підтримка анімацій	—	+
Підтримка 3D графіки	—	+
Підтримка мультимедіа	—	+
Підтримка веб-контенту	—	+
Підтримка шаблону MVC	±	+

Обрані технології розробки програмного застосунку:

Мова програмування:
Java

Графічний фреймворк:
JavaFX

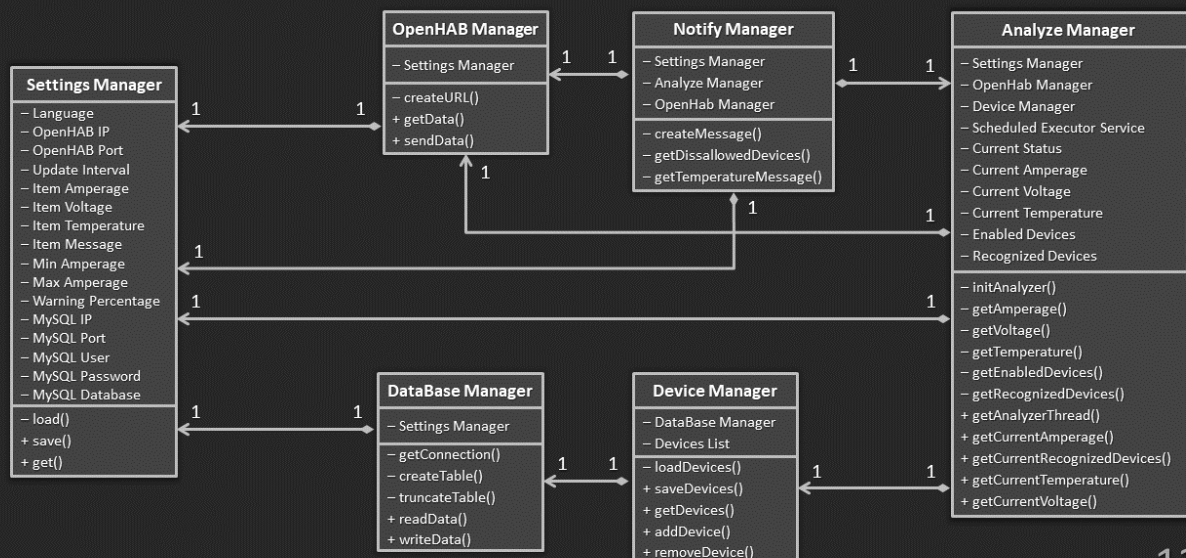
Шаблон проектування:
MVC

Інтегроване середовище розробки: **IntelliJ IDEA**

11

Рисунок В.11 – Слайд №11

ДІАГРАМА ОСНОВНИХ КЛАСІВ



12

Рисунок В.12 – Слайд №12

ОСНОВНІ РОЗРОБЛЕНІ АЛГОРИТМИ



Рисунок В.13 – Слайд №13

СКРІНШОТИ ПРОГРАМНОГО ЗАСТОСУНКУ

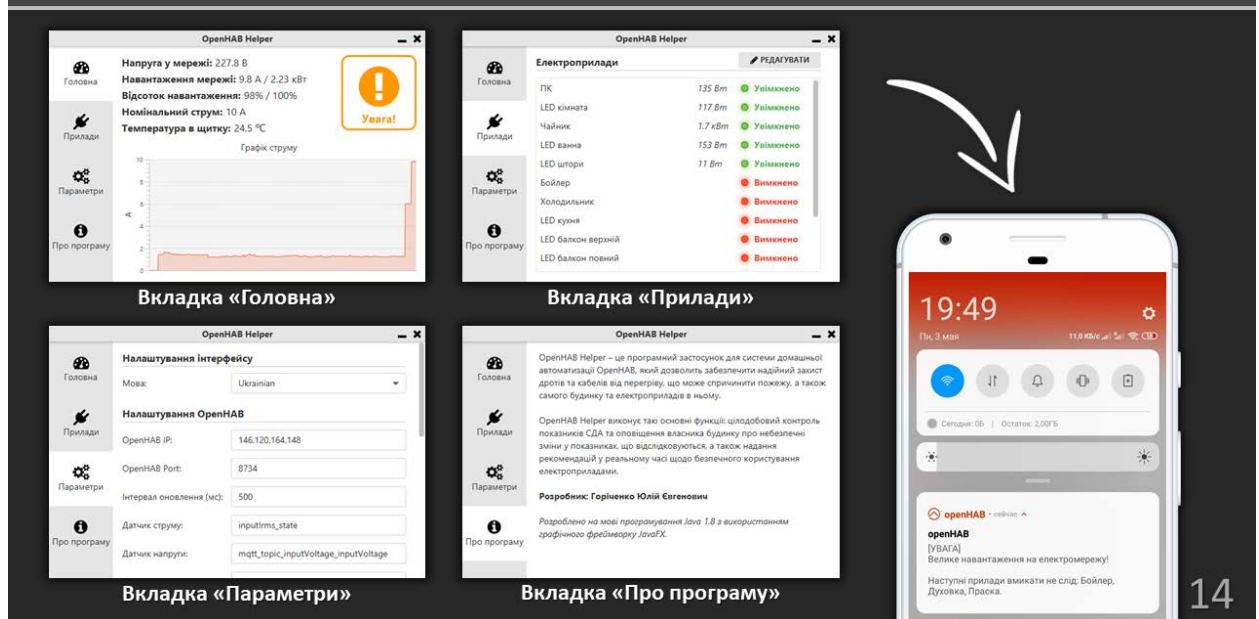


Рисунок В.14 – Слайд №14

ВИСНОВКИ

- ✓ Впровадження СДА є актуальним завданням, оскільки дозволяє підвищити рівень комфорту та безпеки, а також допомагає економити ресурси.
- ✓ Платформа OpenHAB є сучасним рішенням для СДА, що дозволяє виконувати моніторинг показників датчиків та приладів, але не надає засобів контролю використання електроенергії.
- ✓ Розроблений застосунок «OpenHAB Helper» надає можливість контролю показників електрощитового обладнання, тому його використання підвищує ефективність СДА та безпеку користувачів.
- ✓ У подальшій роботі планується впровадити алгоритми інтелектуального аналізу для надання рекомендацій щодо економії електроенергії.

15

Рисунок В.15 – Слайд №15

ПУБЛІКАЦІЇ

1. Горіченко, Ю. Є. Програмне забезпечення для аналізу даних системи домашньої автоматизації / Ю. Є. Горіченко, А. В. Туленков, А. В. Пархоменко // Тиждень науки : Науково-практична конференція, Запоріжжя, квітень 2021 р.: тези доповідей / Редкол. : В. В. Наумик (відпов. ред.) Електрон. дані. – Запоріжжя : НУ «Запорізька політехніка», 2021.
2. 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. (прийнято до розгляду).

16

Рисунок В.16 – Слайд №16

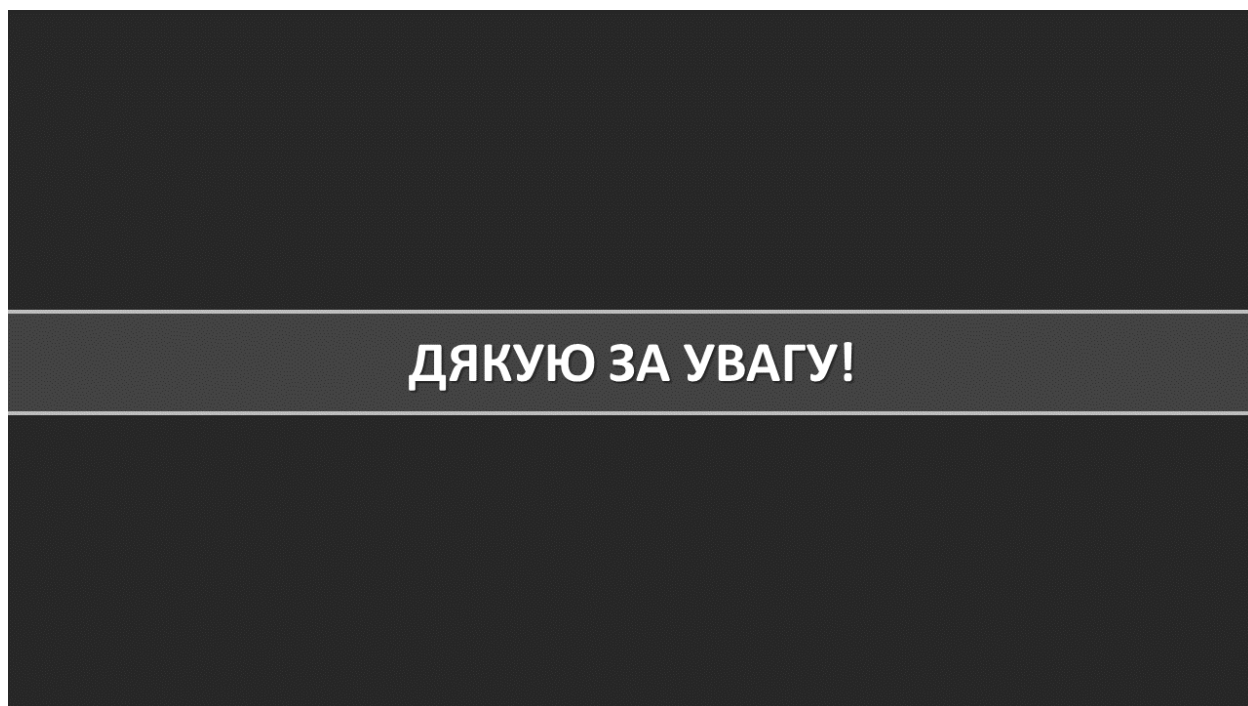


Рисунок В.17 – Слайд №17