

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 проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ ЕНЕРГОСПОЖИВАННЯ.
RESEARCH AND SOFTWARE IMPLEMENTATION OF THE METHODS FOR
THE INTELLIGENT DATA ANALYSIS OF ENERGY CONSUMPTION.

Виконав: студент(ка) 2 курсу, групи КНТ-120м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Задоян М.К.

(прізвище та ініціали)

Керівник Пархоменко А.В.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

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

Інститут, факультет ІІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Задояна Мирослава Костянтиновича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів інтелектуального аналізу даних енергоспоживання. Research and Software Implementation of the Methods for the Intelligent Data Analysis of Energy Consumption

керівник проєкту (роботи) Пархоменко А. В., к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “ 5 ” листопада 2021 року № 435

2. Строк подання студентом проєкту (роботи) 10 грудня 2021 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Дослідження методів інтелектуального аналізу даних. 3. Проєктування програмного забезпечення. 4. Вибір інструментарію розробки. 5. Конструювання програмного забезпечення. 6. Керівництво з використання програмного забезпечення.

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 “ 2 ” вересня 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	2-5 тижні	Розділ 1 – 2
3	Розробка архітектури програмної системи	6-8 тиждень	Розділ 3
4	Розробка програмної системи	9-12тижні	Розділ 4-5
5	Розробка керівництва з використання програмної системи	12-14 тиждень	Розділ 6
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	15 тиждень	ПЗ, Додатки
7	Захист роботи	16 тиждень	

Студент(ка)

(підпис) Задоян М.К.
(прізвище та ініціали)

Керівник проєкту (роботи)

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

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
155 с., 75 рис., 16 табл., 2 дод., 86 джерел.

ЕНЕРГОСПОЖИВАННЯ, ОФІСНА АВТОМАТИЗАЦІЯ,
ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, ПРОГНОЗУВАННЯ ДАНИХ, AZURE.

Об’єкт дослідження – процес моніторингу показників енергоспоживання в системі офісної автоматизації.

Предмет дослідження – методи та алгоритми інтелектуального аналізу даних (ІА) енергоспоживання.

Мета роботи – дослідження та практична реалізація методів і засобів інтелектуального аналізу даних для підвищення енергоефективності системи офісної автоматизації.

Матеріали, методи та технічні засоби: структурне та об’єктно-орієнтоване програмування, інтегроване середовище розробки Visual Studio Code, мова програмування Python та JavaScript, бібліотеки FastAPI, Vue, СКБД MSSQL в хмарній платформі Azure SQL.

Результати. Розроблено систему для ІА даних енергоспоживання, що забезпечує можливість обробки інформації з системи офісної автоматизації для прогнозування даних енергоспоживання.

Висновки. Інтелектуальний аналіз даних системи офісної автоматизації з можливістю прогнозування даних енергоспоживання дозволяє користувачу прийняти рішення щодо планованих витрат на комунальні послуги з бюджету компанії, а також попереджати потенційно небезпечні ситуації, пов’язані зі споживанням електроенергії.

Галузь використання – системи офісної та домашньої автоматизації.

ABSTRACT

Explanatory note to the diploma qualifying work of the magister: 155 pages, 75 figures, 16 tables, 2 appendixes, 86 sources.

POWER CONSUMPTION, OFFICE AUTOMATION, INTELLECTUAL DATA ANALYSIS, DATA FORECASTING, WEB-APPLICATION, AZURE.

The object of the study is the process of monitoring energy consumption indicators in the office automation system.

The subject of the study are methods and algorithms of intelligent energy consumption data analysis.

The purpose of the work is research and practical implementation of methods and tools of intelligent data analysis to improve the energy efficiency of office automation.

Materials, methods, and tools: structural and object-oriented programming, Visual Studio Code integrated development environment, Python and JavaScript programming language, FastAPI, Vue libraries, MSSQL database in Azure SQL resource.

Results. A system for IA energy consumption data has been developed, which provides the possibility of processing information from the office automation system for forecasting energy consumption data.

Conclusions. Intelligent data analysis of office automation systems with the functionality to predict energy consumption data allows the user to decide on the planned cost of utilities from the company's budget, as well as to predict potentially dangerous situations related to electricity consumption.

The fields of application are office and home automation systems.

ЗМІСТ

Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз предметної області	12
1.1 Обґрунтування необхідності застосування ІА для даних, отриманих з систем домашньої та офісної автоматизації	12
1.2 Порівняння існуючих рішень для моніторингу даних енергоспоживання	14
1.3 Порівняння існуючих платформ автоматизації	17
1.4 Аналіз функціональних та структурних особливостей існуючої соа	22
1.5 Засоби аналізу даних соа	27
1.6 Постановка задач магістерської роботи	29
2 Дослідження методів та засобів інтелектуального аналізу даних	31
2.1 Дослідження типів ІА даних СОА	31
2.2 Огляд алгоритмів ІА	36
2.3 Огляд програмних засобів для реалізації алгоритмів ІА	39
2.4 Аналіз вибірки даних для порівняння алгоритмів	41
2.5 Порівняння алгоритмів ІА	45
2.6 Висновки до розділу 2	48
3 Проєктування системи	50
3.1 Аналіз вимог до системи	50
3.1.1 Аналіз серверних систем	52
3.1.2 Порівняння інтерфейсів користувача	55
3.2 Проєктування архітектури системи	56
3.3 Розробка прототипу користувацького інтерфейсу	59
3.4 Проєктування схеми БД	62
3.5 Висновки до розділу 3	64
4 Вибір інструментарію розробки	65
4.1 Порівняння та вибір засобів розгортання продукту в хмарній платформі	65
4.2 Аналіз та вибір СКБД	66

4.3	Огляд та вибір мови програмування.....	70
4.4	Аналіз та вибір середовища розробки.....	73
4.5	Порівняння та вибір фреймворків та технологій для розробки.....	74
4.4.1	Вибір фреймворку або бібліотеки для серверної частини	75
4.4.2	Вибір фреймворку або бібліотеки для клієнтської частини.....	76
4.6	Висновки до розділу 4.....	78
5	Конструювання системи	80
5.1	Конструювання інфраструктури.....	81
5.2	Конструювання серверної частини.....	87
5.3	Конструювання скрипту для зберігання показників з сенсорів СОА	91
5.4	Конструювання скрипту для створення вибірок для модулю машинного навчання	93
5.5	Конструювання клієнтської частини.....	95
5.6	Конструювання модулю машинного навчання	97
5.7	Висновки до розділу 5.....	99
6	Керівництво з використання системи.....	101
6.1	Призначення та умови використання системи	101
6.2	Функції системи.....	101
6.3	Вхідні та вихідні дані	101
6.4	Налаштування системи	102
6.5	Початок роботи з системою.....	106
6.6	Взаємодія із системою	107
6.7	Висновки до розділу 6.....	111
	Висновки.....	112
	Перелік джерел посилання	114
	Додаток А Текст програми	123
	Додаток Б Слайди презентації.....	147

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
ASGI	– Asynchronous Server Gateway Interface;
HTML	– HyperText Markup Language;
HTTP	– HyperText Transfer Protocol;
IaC	– Infrastructure as a Code;
MVP	– Minimal Viable Product;
ORM	– Object-Relational Mapping;
PEP	– Python Enhancement Proposals;
TCP	– Transmission Control Protocol;
WSGI	– Web Server Gateway Interface;
БД	– база даних;
БІ	– бізнес інтелект;
Датасет	– вибірка даних;
ООП	– об’єктно-орієнтоване програмування;
ОС	– операційна система;
СКБД	– система контролю баз даних;
тне	– тон нафтового еквіваленту енергії.

ВСТУП

Україна має одну з найбільш енергоємних економік в світі, оскільки щорічно споживає близько 92 млн тон нафтового еквіваленту енергії (тне). Якщо українська економіка матиме енергоємність на середньосвітовому рівні, то це б дозволило скоротити споживання енергії приблизно на 50 млн тне [1].

Майже половина (44%) виробленої та імпортованої енергії втрачається під час її перетворення та транспортування до фінального споживача, що на 12% вище ніж в країнах ЄС. Так з 92 млн тне постачання первинної енергії, споживається близько 52 млн тне. Потенціал скорочення витрат енергії при споживанні може досягати 60%. Тільки за оцінкою в трьох секторах – житловий, бюджетний та постачання енергії (разом близько 63% енергетичного балансу України) потенціал енергоефективності дорівнює заощадженню близько 19 млн тне або 8,0 млрд євро щорічно [1].

Розвиток енергетики має вирішальний вплив на стан економіки в державі та рівень життя населення. Метою соціальної держави, якою відповідно до Конституції є Україна, має бути забезпечення умов для зростання добробуту громадян. Однією з найважливіших складових добробуту в цивілізованих державах є забезпечення громадян і компаній необхідними енергоресурсами. Запорукою реалізації цієї мети має стати надійне, економічно обґрунтоване й екологічно безпечне задоволення потреб населення й економіки в енергетичних продуктах [2].

Як наслідок, сьогодні набирають оберти системи домашньої, офісної та промислової автоматизації. Системи домашньої та офісної автоматизації – це сукупність апаратного та програмного забезпечення, що дозволяють легко керувати приладами такими як освітлення, опалення та звукові системи [3]. Промислові системи надають можливості автоматичного керування та

дозволяють людині контролювати промислові процеси, такі як розподіл, обробка та виробництво продукції.

Промислові системи управління використовуються на підприємствах з видобутку нафти, газу та вугілля, на фабриках, електростанціях, на підприємствах з очищення води/стічних вод, а також у целюлозно-паперовій та транспортній промисловості [4].

Але однією з найголовніших функцій, які повинні виконувати системи автоматизації – це моніторинг енергоспоживання. Якщо говорити про системи промислової автоматизації, то глобальна конкуренція в промисловому виробництві, за підвищення продуктивності, якості продукції, безпеки процесів на додаток до економічної та екологічної стійкості – все це призвело до розвитку сучасних систем моніторингу процесів та даних. Сучасні розробки, що стосуються технологій промислового Інтернету речей, алгоритмів машинного навчання та доступності великих даних, забезпечують платформи можливостями для реалізації складної аналітики технологічних даних. Існує кілька платформ хмарних обчислень, доступних для використання в промисловому Інтернеті речей та аналітиці великих даних. Основними гравцями є постачальники хмарних послуг (наприклад, Microsoft Azure, Amazon Web Services, IBM, Intel тощо), постачальники корпоративних рішень (зокрема, PTC та Oracle), мережеві компанії (такі як Cisco та AT&T) та промислові інженерні компанії (а саме General Electric, Siemens та Bosch). Деякі з цих платформ доступні за власними ліцензіями, а деякі інші доступні як проєкти з відкритим кодом [5].

Проте, коли мова йде про аналіз даних енергоспоживання, то зазвичай стандартні платформи автоматизації наділені простим функціоналом для аналізу даних енергоспоживання. Прикладами найвідоміших платформ автоматизацій, які можна використати для роботи з системами домашньої автоматизації є: OpenHAB, MajorDoMo, ioBroker, Domoticz, Home Assistant. Дані платформи

мають простий функціонал для надсилання повідомлень, або побудови простих статистик на основі даних [6]-[7].

Як результат – системи офісної та домашньої автоматизації генерують доволі великі об'єми даних щодо енергоспоживання. Проте, не проводиться повноцінний інтелектуальний аналіз цих самих даних. Тому, тема роботи є актуальною.

Метою роботи є дослідження та практична реалізація методів і засобів інтелектуального аналізу даних для підвищення енергоефективності системи офісної автоматизації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

В Україні дані енергоспоживання є найбільш доступними для користувачів різного рівня, їх подальша обробка здійснюється енергетичними компаніями які надсилають користувачам квитанції з вже розрахованою кількістю спожитої енергії. В наслідок цього споживачам необхідно зберігати безліч паперових квитанцій, що ускладнює самостійний аналіз цих самих даних [8]-[9].

1.1 Обґрунтування необхідності застосування ІА для даних, отриманих з систем домашньої та офісної автоматизації

Системи автоматизації можна встановити як в квартири, будинки, офіси та інші приміщення. Проте вимоги до функціоналу систем домашньої автоматизації (СДА) та для систем офісної автоматизації (СОА) відрізняються. Основний функціонал наведено на рисунку 1.1.



Рисунок 1.1 –Базові функції СДА та СОА

СДА здатна функціонально об'єднати між собою усі електроприлади в будинку, якими можна керувати дистанційно за допомогою дисплею або пульта.

Для віддаленого доступу до приладів через Інтернет, вони підключаються до єдиної мережі, що дозволить керувати ними за допомогою ПК або мобільного застосунку. До функцій розумного будинку можна віднести [10]:

- автоматичний контроль освітленості в різних частинах будинку у залежності від часу доби та пересування людей;
- побутові турботи, такі як, наприклад, полив саду, відкриття або закриття жалюзі, тощо;
- контроль за протіканням газу чи води.

Це тільки деякі приклади функціоналу, що може містити в собі СДА, кожна система автоматизації, має свої основні, додаткові та унікальні функції, що можуть полегшити повсякденне життя користувачів.

Все частіше великі підприємства, офіси та адміністративні будівлі в усьому світі успішно впроваджують та використовують інтелектуальні системи моніторингу електроспоживання (HIEMS) [11]. Такі дії дозволять, у майбутньому, використати політику електроспоживання і змінити поведінку використання ресурсів на локальному та глобальному рівні.

Бренд Intelligent Energy System Services (IE2S) [12] надає спеціалізовані компетенції для розробки індивідуальних консалтингових рішень для учасників ринку енергетики і мобільності. Даний бренд включає як консультативні можливості та програмне забезпечення, так і дослідницькі функції на енергетичному ринку. Інтелектуальний моніторинг енергоресурсів охоплює глобальні корпорації на рівні континентального розподілу енергії та кінцевих користувачів [9].

Згідно з рисунком 1.1 можна зробити висновок, що кожна з систем автоматизацій потребує майже однаковий функціонал, а саме:

- контролювати електроспоживання, щоб аналізувати та попереджувати потенційні перепади напруги в мережі, або щоб планувати потенційні витрати на електроенергію;

- контролювання системами освітлення, щоб була наявна можливість віддалено вмикати або вимикати освітлення для всієї площі системи, або для певних частин (кімнат);
- автоматичний виклик служб безпеки у непередбачуваних ситуаціях, щоб своєчасно загасити полум'я або спіймати грабіжників.

Кожна з систем має певні особливості, наприклад для СДА це:

- автоматичне вимикання та вимикання споживчих систем, таких як бойлери, або підігрів підлог;
- автоматичне вимикання та вимикання розважальних систем, таких як телевізори або ігрові приставки.

А для СОА важливий функціонал для контролю безпеки по всій площі, створення додаткових засобів безпеки для входних дверей, щоб забезпечити безпеку у приміщенні за рахунок спеціальних розумних ключів, або сканів відбитків пальців та мати можливість вести облік усіх персон, що відвідують офіс. Це все необхідно тому що, в офісі зберігається обладнання та конфіденційна інформація про компанію та працівників.

Таким чином, на сьогодні існує велика кількість СДА та інформації про них, у той час як про СОА інформації значно менше, хоча потреба в цих системах постійно зростає.

1.2 Порівняння існуючих рішень для моніторингу даних енергоспоживання

Домашні інтелектуальні системи моніторингу енергоспоживання призначені для отримання даних про напругу в мережі і величину споживання струму користувачами в поточний момент з можливістю подальшої передачі, зберігання і аналізу даних. Для моніторингу витоків і звітності, системи

моніторингу встановлюються на всіх етапах передачі електроенергії від станцій генерації енергії до кінцевого споживача.

Компанії з усього світу постійно створюють комплексні рішення для вимірювання електроенергії, що використовуються в будинках і квартирах. Так наприклад, компанія «Buckler»[13] створила розумний-лічильник електроенергії, він встановлюється на квартирному розподільному щиті. Пристрій дає можливість обліку і відображення інформації про енергоспоживання, з можливістю навчання лічильника визначати 10 побутових приладів і передачею інформації по Wi-Fi в хмарний сервіс [14]. Компанія «Aotec» [15] представляє розумний лічильник реального часу з безконтактним вимірюванням струму, однак дане рішення призначене для використання разом зі застосуванням сторонніх передавачів z-wave, що робить його не самостійним пристроєм. Компанія «smart-MAC» [16] презентувала серію лічильників електроенергії з сервісом візуалізації і можливістю використання датчиків різного типу. «Новатек-Електро» [17] представляє Wi-Fi лічильник електроенергії з функцією захисту і управління, а також наданням програмного інтерфейсу (API) для локальної взаємодії з ним.

Проаналізувавши пристрої для моніторингу енергоспоживання, стало зрозумілим, що головною характеристикою більшості з них є висока для користувачів цінна та відсутність можливості їх інтеграції в сторонні системи для передачі даних.

Наявність інтелектуального моніторингу в системах автоматизації дозволяє вирішувати ряд таких задач, як контроль витрат електроенергії та ефективність використання обладнання. Автори [18] запропонували інтелектуальний вимір споживання енергії з одночасним визначенням патерну використання енергоресурсів кожного підключеного пристрою, однак на запропонованому пристрої або на використовуваному хмарному сервісі не реалізовані приклади аналізу даних з подальшим визначенням патернів. Також

автори [19] стверджують, що система обліку енергоспоживання не має прямого і швидкого впливу на енергозбереження. Зазначено, що дані дають уявлення про звички і поведінку жителів. Наведено приклад аналізу даних інтелектуального лічильника та інтелектуальної системи обліку енергоспоживання. Автори [20] провели експеримент в домоволодіннях на різних групах сімей з різним типом поведінки і енергоспоживання з виявленням залежностей поведінки від температури.

Побудувати інтелектуальний моніторинг можна зі застосуванням датчиків та контролерів без серійного виробництва. Наприклад, автори [21] пропонують інтелектуальний лічильник на базі веб-інтерфейсу для систем розумного будинку загального призначення з використанням двох відомих контролерів ESP8266 та Arduino Nano. Недолік даного підходу є використання двох видів контролера, що є надлишковим і недоцільним в запропонованій системі.

Автори [22] застосували контролер NodeMCU разом з NRF24L01 передавачем, щоб знизити споживання енергії і питомої вартість енергії. Також у даній роботі для відображення інформації використовувався відкритий сервіс Blynk.

Отже, на основі даних отриманих в ході проведеного аналізу, можна зробити висновок, що існуючі на разі системи є жорсткими самостійними системами без можливості сторонньої інтеграції. Окрім того, аналіз також показав, що запропоновані пристрої є системами обліку даних без подальшого інтелектуального аналізу даних.

Таким чином на сьогодні існують системи для обліку та моніторингу споживання електроенергії, однак наявні в них недоліки не дозволяють віднести їх до інтелектуальних систем моніторингу.

Основними вимогами до інтелектуальної системи моніторингу є:

- динамічна інтеграція в сторонні системи;

- спосіб отримання даних без залучення стороннього фахівця для установки системи моніторингу;
- інтелектуальний аналіз та надання користувачеві рекомендацій.

1.3 Порівняння існуючих платформ автоматизації

Прикладами найвідоміших платформ автоматизацій, які можна використати для створення систем типу СДА та СОА є: OpenHAB, MajorDoMo, ioBroker, Domoticz, Home Assistant та інші, опис яких наведено в таблиці 1.1 [6]- [7].

Таблиця 1.1 – Опис платформи автоматизації

Платформа	Реалізація платформи	Розгортається	Особливості та функції платформи
1	2	3	4
OpenHAB	Розроблена на Java.	На системах з підтримкою Java (Windows, MacOS, Linux, Raspberry Pi, PINE і ін.).	Має 3 інтерфейси (Paper UI, Basic UI as well as HABmin). Керування за допомогою голосу. Є мобільні застосунки для iOS і Android. Інтегрується з Apple HomeKit. Можна зв'язати різні події, що відбуваються у будинку та в залежності від них налаштувати поведінку пристроїв.
MajorDoMo	Створена на програмній архітектурі Apache + PHP + MySQL.	На різних платформах (Windows, Linux, Raspberry Pi).	Підтримує багато протоколів зв'язку. Інтегрується з Apple HomeKit. Може виконувати функції з управління пристроями, реагувати на події та розсилати повідомлення. В системах з централізованою архітектурою роль головного процесора бере на себе сервер з системою MajorDoMo.
Domoticz	Розроблена на C/C++ з достойним графічним інтерфейсом.	Windows, MacOS, Linux, Raspberry Pi і ін.	Недоліком є обмежений набір драйверів або модулів для підтримки різного устаткування.

Продовження таблиці 1.1

1	2	3	4
Home Assistant (HASS)	Реалізована на Python з ідеології.	Windows, Linux, MacOS, Raspberry Pi і ін.	За допомогою плагінів можливості системи розширюються. YAML використовується для конфігурації. Node-RED використовується для управління правилами автоматизації.

OpenHAB, MajorDoMo, ioBroker, Domoticz, Home Assistant – платформи автоматизації з відкритим вихідним кодом. Вони здатні працювати практично на будь-якій ОС (Windows, MacOS, Linux), Raspberry Pi або на будь-якому локальному сервері.

Domoticz, ioBroker та MajorDoMo є системами з простим та зрозумілим користувацьким інтерфейсом для додавання та управління пристроями в будинку. Domoticz може відправляти користувачу push-сповіщення на Android або iPhone, а також пропонує менеджер подій на основі Blockly, що спрощує керування подіями та створення автоматизованих своєчасних подій. ioBroker потужна система автоматизації, яка написана на NodeJS та пропонує загалом близько 300 інтеграцій. Її програмне забезпечення абсолютно безкоштовне. Система підтримує додавання, налаштування та управління пристроями. MajorDoMo є новою платформою для домашньої автоматизації, яка підтримує безліч протоколів. Веб-інтерфейс користувача простий та легкий у використанні особливо для нетехнічних користувачів, що дозволяє їм додавати кімнати та пристрої, налаштовувати автоматизовані події, а також інтегрувати веб-програми та хмарні послуги [23] - [24].

Системи OpenHAB та Home Assistant забезпечують інтеграцію з Google Assistant та Amazon Alexa. До цього переліку OpenHAB додає підтримку IFTTT та Apple HomeKit, а Home Assistant підтримку голосового помічника з відкритим вихідним кодом – Mycroft.io [23] - [24].

Home Assistant – платформа автоматизації, яка працює на Python 3 під ліцензією Apache 2.0. Має багату бібліотеку підключених пристроїв, підтримуваних послуг, датчиків та камер, а також послугу розумного автомобілю. Система поєднує понад 1400 підтримуваних служб і пристроїв, які користувач може переглядати у бібліотеці інтеграцій, оскільки вони класифіковані за категоріями: VoiceUI, сигналізація, автоматика, автомобіль, енергія, датчики, здоров'я, обробка зображень, мультимедіа, тощо [23] - [24].

OpenHAB є системою “важкої ваги”, яка здатна працювати на багатьох операційних системах. Система має підтримку безлічі плагінів, що допомагає розробникам додавати нові пристрої або інтегрувати нові послуги. OpenHAB має зручний користувацький дизайн з тригерами, орієнтованими на час та події, службу сповіщень та керування за допомогою VoiceUI. А також пропонує мобільні застосунки для пристроїв на базі Android та застосунок iOS для iPhone та iPad [23] - [24].

Платформа автоматизації MajorDoMo намагається реалізувати ті ж самі функціональні можливості, що й OpenHAB, тільки за допомогою іншої програмної архітектури, яка хоч і є більш гнучкою, однак вимагає специфічного програмування від користувача. Також система MajorDoMo більше орієнтується на російськомовну спільноту. Обидві системи можуть інтегруватись з Apple HomeKit, що дозволяє їм керувати розумним будинком за допомогою пристроїв Apple без додаткового встановлення спеціальних розширень [6]-[7].

У порівняння з OpenHAB або MajorDoMo платформа автоматизації Domoticz забезпечує виконання основного функціоналу та не потребує при цьому значних ресурсів в процесі роботи. Однак через це система є обмеженою за кількістю модулів та драйверів, що необхідні для підтримки різного устаткування [6] - [7].

Home Assistant – потужна платформа автоматизації, що пропонує систему, яка задовольняє перш за все місцевий контроль та конфіденційність даних

клієнтів. Система має широкий функціонал та здатна розширювати його завдяки плагінам. Home Assistant та OpenHAB схожі у тому сенсі, що обидві програми поділяють схоже бачення та намагаються об'єднати якомога більше пристроїв в одну безкоштовну платформу. Також серед усіх платформ Home Assistant є найбільш гнучкою з точки боку розробників [24].

На рисунках 1.2 – 1.6 наведено приклади інтерфейсів платформ автоматизацій.

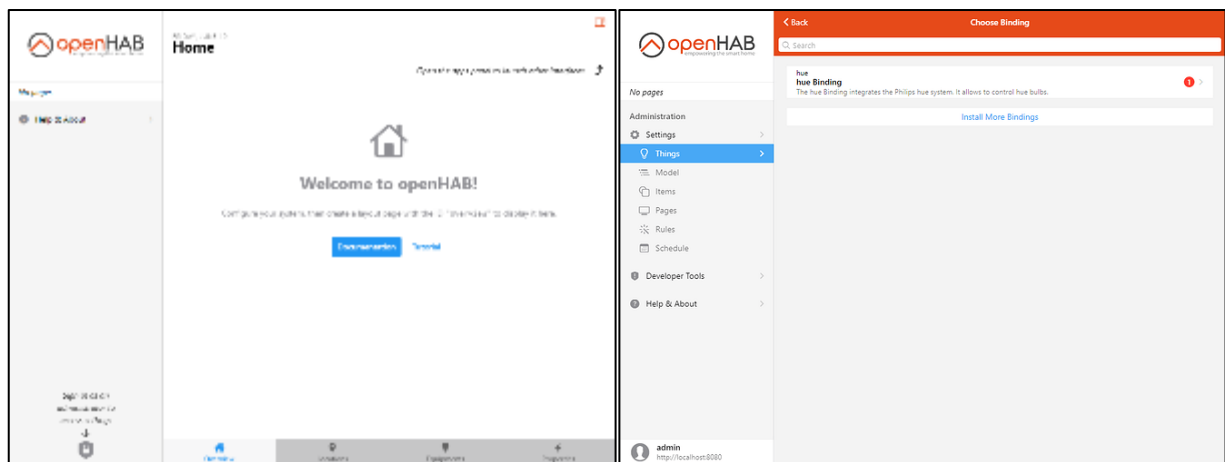


Рисунок 1.2 – Приклади інтерфейсу платформи автоматизації OpenHAB

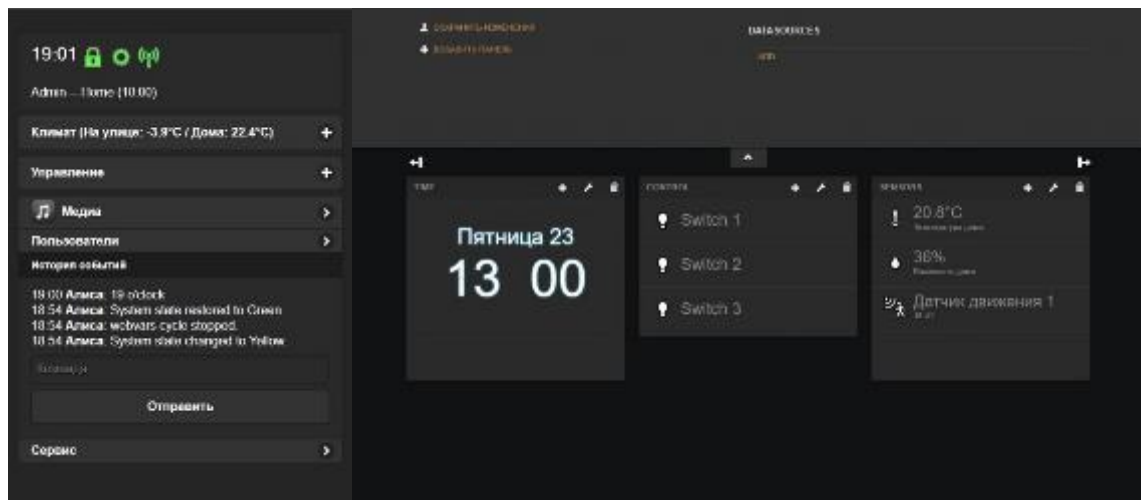


Рисунок 1.3 – Приклади інтерфейсу платформи автоматизації MajorDoMo

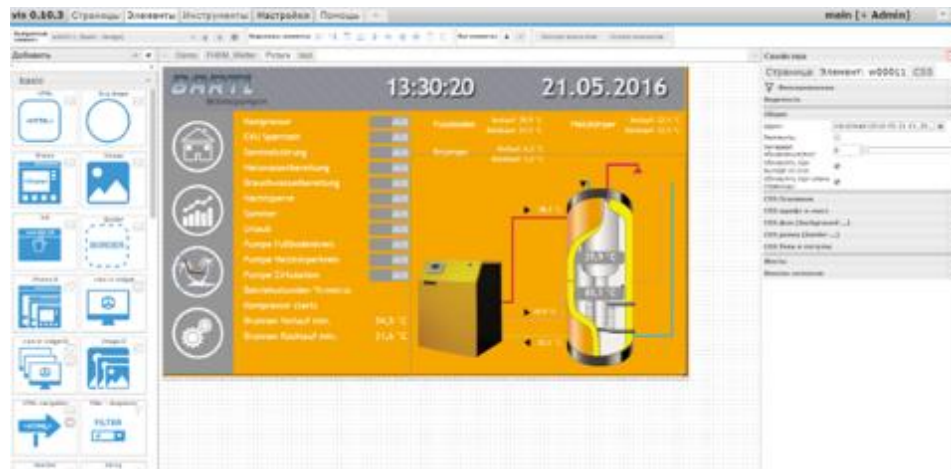


Рисунок 1.4 – Приклади інтерфейсу платформи автоматизації ioBroker

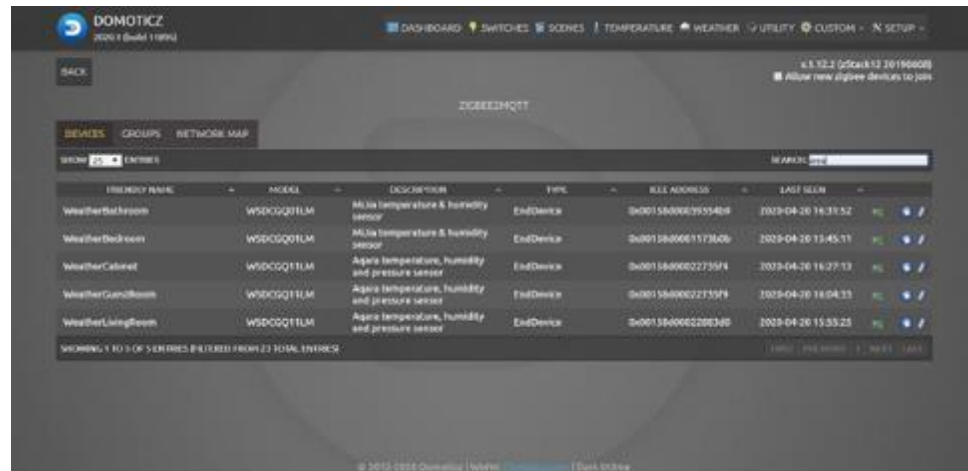


Рисунок 1.5 – Приклади інтерфейсу платформи автоматизації Domoticz

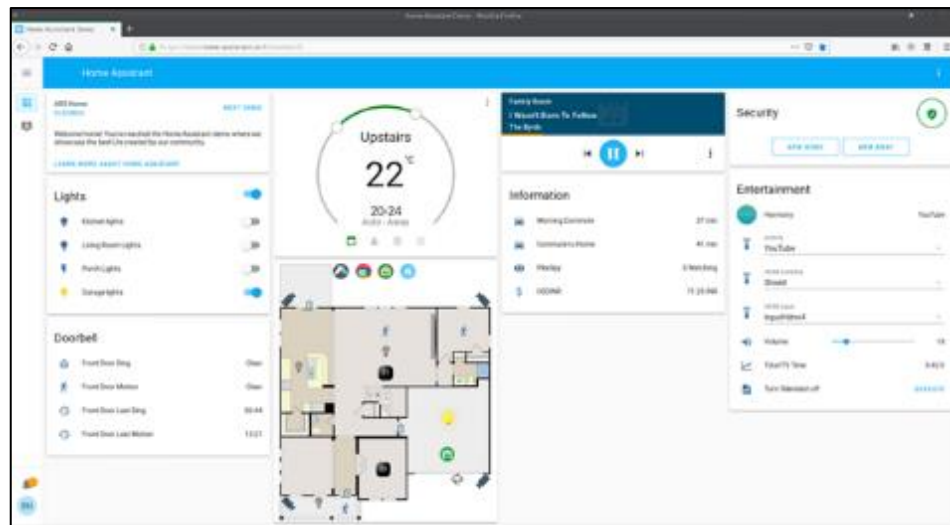


Рисунок 1.6 – Приклади інтерфейсу платформи автоматизації Home Assistant

В ході порівняння платформ автоматизацій для розробки може бути рекомендована Home Assistant, оскільки дана система є гнучкою та надійною системою з потужним функціоналом, до якого входить функція домашнього керування, що відповідає за збір інформації та управління пристроями. Також у порівнянні з іншими системами Home Assistant є набагато легшою, оскільки не використовує великого обсягу пам'яті і працює досить швидко. Програмне забезпечення не буде завантажувати оперативну пам'ять або процесор, що підвищує швидкість та продуктивність роботи системи [25]-[26].

Home Assistant є готовою для повноцінного використання системою, що постійно розвивається та оновлюється. Оновлення системи відбувається кожні два тижні, що дозволяє забезпечити безпеку користувача від нових загроз, а також підтримувати синхронізацію підключених систем із їх останніми оновленнями [25]-[26].

1.4 Аналіз функціональних та структурних особливостей існуючої СОА

Щоб довести необхідність ІА даних СОА є сенс розглянути реальну та налаштовану систему автоматизації, що побудована із використанням платформи автоматизації Home Assistant. На рисунку 1.7 зображено структура системи СОА.

На даному рисунку наведено опис датчиків та актуаторів у системі офісної автоматизації.

Дана система складається з наступних кімнат:

- майстерня перша – відбуваються процеси роботи з хімічними елементами, електронікою, 3D друк;
- майстерня друга (верстатна) – встановлені верстати, токарний, фрейзерний, зварювання;
- туалети – встановлені сололіфти;

- зона відпочинку – зона з проектором та диваном;
- кухня – холодильник, посудомийка, насос для відкачування води, подрібнювач відходів;
- кімната директора – комп’ютери;
- офісна кімната – комп’ютери;
- склад із обладнанням – пиломатеріали, просто матеріали, інструменти;
- вітальня – маленький хол і три двері в різні боки;
- коридор – зі стелажми та матеріалами, спільним світлом та дверима;
- загальні модулі, що відповідають за живлення усієї системи.

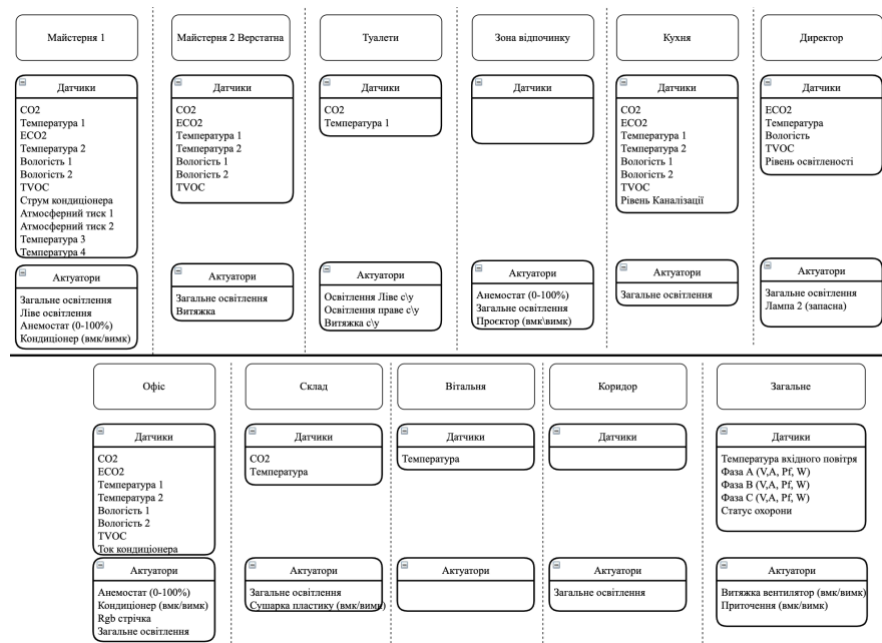


Рисунок 1.7 – Структура COA

Усі ці кімнати включають в себе моніторинг електропостачання на три фази, реле, датчики вологості та температури, датчики повітря, та датчики освітлення.

До фази А належать: світло та розетки у першій майстерні, електроніка у кімнаті «Директор», офіс, світло в туалеті, та світло в другій майстерні.

До фази В – кухня, розетки в туалеті, бойлер, серверна кімната, та світло в коридорі.

До фаза С – верстати з другої майстерні, розетки першої майстерні, світло на кухні, та розетки на складі.

В представленій СОА моніторинг електричного струму заснований на датчику PZEM-004-100A. На ньому зав'язані усі налаштовані датчики в системі, він являтиме собою першу причину потенційної загрози для користувача.

В офісі встановлено щит забезпечення харчування. До офісу приходить 3 фази струму – А,В,С. На кожен фазу встановлено розроблений пристрій обліку на базі вимірювача мір енергії PZEM-100. Три вимірювачі передають дані, за допомогою Wi-Fi модуля esp32 локальний MQTT Брокер. Систему живлення із використанням цього датчику наведено на рисунку 1.8.

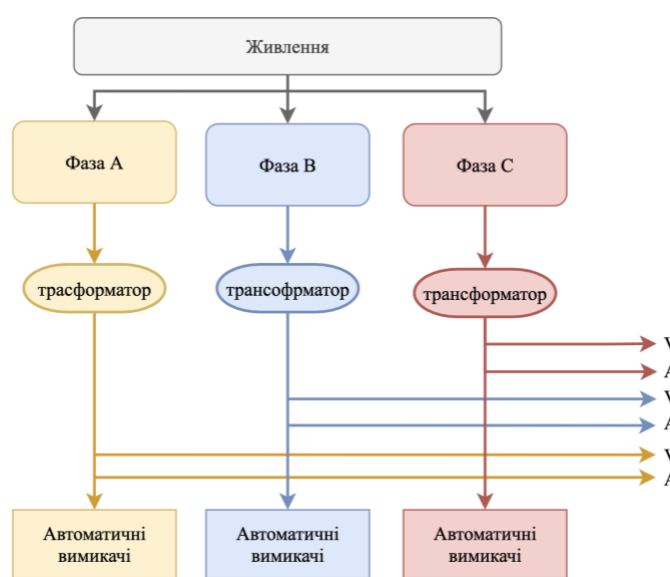


Рисунок 1.8 – Схема живлення СОА

Кожен вимірювач енергії кожні 2 секунди надсилає до брокера наступну інформацію: напруга, струм, поточна потужність, коефіцієнт потужності. Дані MQTT брокера відображаються в системі автоматизації Home Assistant у вигляді графіків. Приклад таких графіків наведено на рисунках 1.9 – 1.11.

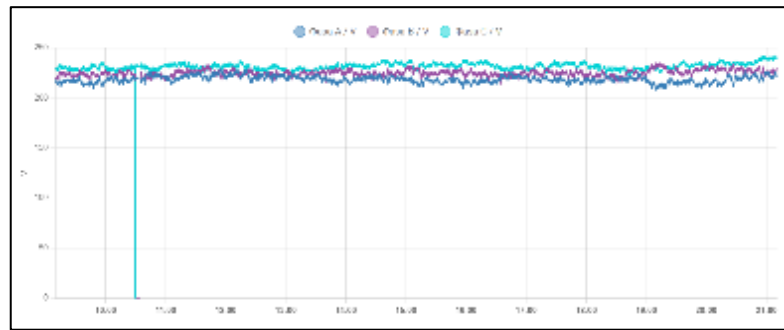


Рисунок 1.9 – Приклад графіку напруги/струму



Рисунок 1.10 – Приклад графіку сили струму

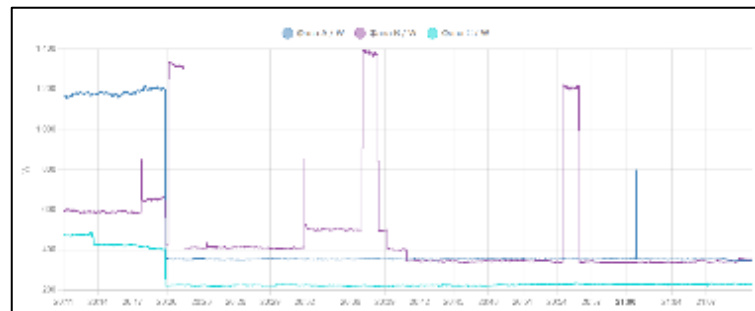


Рисунок 1.11 – Приклад графіку зміни напруги в системі

Також у даній системі наявні оповіщення у месенджер Telegram. Вони засновані на правилах, що прописуються у самій системі Home Assistant. Нижче наведено приклади таких правил:

- якщо в першій майстерні CO_2 перевищив 600ppm протягом чотирьох хвилин, то відкрити анємостат на 90%;

- якщо в першій майстерні Co2 нижче 500ppm протягом 4х хвилин то закрити анемоустат на 10%;
- оповіщення коли офіс знято з охорони і поставлено на охорону;
- оповіщення коли потоп на кухні, наприклад якщо насос перестав відкачувати воду;
- оповіщення про дні народження;
- включення вентиляції о 6 ранку, якщо день робочий, та вимкнення після ставлення офісу на охорону;
- увімкнення проєктора із задачами о дев'ятій ранку, якщо день робочий;
- вимкнути вентиляцію о дванадцятій годині, якщо ніхто не прийшов на роботу;
- пропало живлення на якійсь фазі датчика струму;
- перевищено номінальне споживання струму.

Приклади даних сповіщень наведено на рисунку 1.12.



Рисунок 1.12 – Приклади сповіщень згідно з описаними правилами

Дана система зберігає дані у власну СКБД. В офісі зберігається дорога техніка та обладнання, що у випадку непередбачуваних ситуацій може призвести до значних витрат. А оскільки для офісу важливим є розробка фінансового плану на кілька років, то аналіз та контроль даних енергоспоживання є важливим критерієм для його створення.

1.5 Засоби аналізу даних COA

Три найпоширеніші засоби для аналізу даних COA. із них наведено на рисунку 1.13.



Рисунок 1.13 – Найпоширеніші засоби аналізу даних COA

Порівняння цих засобів наведено в таблиці 1.2.

Таблиця 1.2 – Порівняння засобів аналізу даних

Назва	Переваги	Недоліки
1	2	3
Розроблення правил для Home Assistant	Використання вже існуючого функціоналу, легко інтегрувати в систему.	Правила системи мають величезну кількість обмежень в плані взаємодії із даними, та інформування користувача.

Продовження таблиці 1.2

1	2	3
Розроблення власних скриптів для Home Assistant	Використання вже існуючого функціоналу системи, уся система буде написана однією мовою програмування, надає більшу гнучкість для взаємодії із даними	Дані скрипти потребують додаткових ресурсів для запуску них самих. Вони будуть дуже обмежені та їх важко буде масштабувати. Також дані скрипти потребують використання стандартних для системи бібліотек, що спричиняє важкість у підтримці системи у майбутньому.
Інтелектуальний аналіз даних СОА у сторонній системі	Використання сторонніх систем дозволить розробити легко масштабовану та підтримувану систему. Для даної системи є можливість розробити оптимальну архітектуру, що дозволить знизити витрати на дистанції. До того ж розробка не займатиме сильно більше часу ніж розробка скриптів.	Більше зусиль на розробку архітектури та налаштування застосунку.

Таким чином використання стороннього додаткового програмного забезпечення для аналізу даних дозволить розробити гнучку та легко підтримувану систему. Тому такий засіб може бути рекомендований до використання.

1.6 Постановка задач магістерської роботи

Системи автоматизації об'єднують всі пристрої в приміщенні в єдину систему, яка має спеціалізований програмний інтерфейс у вигляді програми для ПК або мобільного застосунку, що дозволяє користувачеві керувати приладами у власному будинку, кліматом, освітленням, тощо.

Однак, обладнання системи домашньої автоматизації споживають досить високу кількість електроенергії. Тому моніторинг енергоспоживання, на сьогодні, є надзвичайно важливою процедурою, адже вона дозволяє виявити та усунути проблеми споживання, зменшити витрати і створити комфортні умови для споживачів.

Як показали дослідження, СОА та СДА дуже схожі за функціоналом, проте у кожній з систем наявні особливості. Наприклад в СДА:

- автоматичне вимикання та вимикання споживчих систем, таких як бойлери, або підігрів підлог;
- автоматичне вимикання та вимикання розважальних систем, таких як телевізори або ігрові приставки.

А для СОА важливий функціонал для контролю безпеки по всій площі, створення додаткових засобів безпеки для входних дверей, щоб забезпечити безпеку у приміщенні за рахунок спеціальних розумних ключів, або сканів відбитків пальців та мати можливість вести облік усіх персон, що відвідують офіс. Це все необхідно тому що, в офісі зберігається обладнання та конфіденційна інформація про компанію та працівників.

Існує велика кількість платформ автоматизації, найпопулярніші з них це – OpenHAB, MajorDoMo, ioBroker, Domoticz, Home Assistant. Зокрема рекомендована платформа Home Assistant, є готовою для повноцінного використання системою, що постійно розвивається та оновлюється. Оновлення

системи відбувається кожні два тижні, що дозволяє забезпечити безпеку користувача від нових загроз, а також підтримувати синхронізацію підключених систем із їх останніми оновленнями.

Однак аналіз показав, що в системі Home Assistant не передбачено повноцінний функціонал для ІА даних енергоспоживання, саме тому є сенс обрати використання сторонньої системи для аналізу даних, що надасть можливість розробити гнучку та легко підтримувану систему. Саме тому тема роботи є актуальною.

Для ефективного використання СОА користувачі потребують прогнозування даних електроспоживання та потенційних загроз внаслідок неправильного використання техніки та, як наслідок, покращення досвіду взаємодії із системою та можливість раціонально використовувати ресурси СОА.

Мета роботи є дослідження та практична реалізація методів і засобів інтелектуального аналізу даних для підвищення енергоефективності системи офісної автоматизації.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

- дослідити методи та засоби ІА даних енергоспоживання;
- проаналізувати вимоги до розроблюваної системи;
- розробити архітектуру системи;
- порівняти та обрати СКБД;
- обрати інструментарій розробки;
- розробити систему;
- описати керівництво з використання системи.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ТА ЗАСОБІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

2.1 Дослідження типів ІА даних СОА

Аналітика даних – це збирання та аналіз даних з метою оптимізації прийняття рішень. Вона переглядає великий обсяг необроблених і неорганізованих даних для виявлення шаблонів, які допомагають організаціям або особам, приймати правильні рішення. Це корисний інструмент для отримання розуміння того, як інформація, конфігурації, а також значення були згенеровані. Окрім того, аналіз виявляє і визначає домінуючі тенденції і проблеми. В наші дні організації завжди використовують аналітику даних, щоб робити висновки про бізнес інтелект (БІ) з контенту соціальних мереж для прийняття рішень і формулювання різних бізнес-стратегій для осіб, які приймають рішення. Аналітика даних – це допомога людям в прийнятті рішень, вони повинні обробляти свій аналіз на основі своїх навичок аналітичного мислення після отримання результату аналізу даних [27].

Використання аналітики даних в системах домашньої автоматизації надасть бізнес-підрозділу можливість пропонувати більш якісні послуги, а отже дасть можливість отримати конкурентну перевагу на ринку [28].

Подібна обробка інформації показує, чому Big Data стала такою важливою:

- велика частина зібраних зараз даних неструктурована і вимагає іншого сховища і обробки, ніж в традиційних реляційних базах даних;
- доступна обчислювальна потужність стрімко зростає, а це означає, що з'являється більше можливостей для обробки великих даних;
- інтернет демократизував дані, постійно збільшуючи обсяг доступних даних, а також виробляючи все більше і більше необроблених даних.

Дані в необробленому вигляді не мають цінності. Дані необхідно обробляти, щоб вони були цінними [29]. Є кілька типів аналітики даних [30], їх зображено на рисунку 2.1.



Рисунок 2.1 – Типи аналітики даних

Аналітика в реальному часі зазвичай виконується на основі даних, зібраних з датчиків. У цій ситуації дані постійно змінюються, і для отримання аналітичного результату в короткі терміни потрібні методи швидкого аналізу даних. Для аналізу в режимі реального часу було запропоновано дві існуючі архітектури: кластери паралельної обробки з використанням традиційних реляційних баз даних і обчислювальні платформи на основі пам'яті. Greenplum [31] і Hana [32] є прикладами архітектури аналітики в режимі реального часу.

Офлайн-аналітика використовується, коли не потрібно швидкої відповіді. Наприклад, багато інтернет-підприємств використовують архітектуру автономної аналітики на основі Hadoop для зниження вартості перетворення формату даних. Така аналітика підвищує ефективність збору даних. SCRIBE [33], Kafka [34], Time-Tunnel [35] і Chukwa [36] є прикладами архітектур, які проводять автономну аналітику і можуть задовольнити потреби в зборі даних.

Аналітика на рівні пам'яті застосовується, коли розмір даних менше пам'яті кластера. До теперішнього часу обсяг пам'яті кластерів досяг рівня терабайт (ТБ). Отже, для підвищення аналітичної ефективності потрібно кілька

технологій внутрішньої бази даних. Аналітика на рівні пам'яті підходить для проведення аналізу в реальному часі. MongoDB є прикладом цієї архітектури.

Аналітика БІ застосовується, коли розмір даних перевищує рівень пам'яті, але в цьому випадку дані можуть бути імпортовані в межі аналізу БІ. БІ-аналітика в даний час підтримує дані на рівні ТБ. Більш того, бізнес-аналітика може допомогти виявити стратегічні бізнес-можливості з потоку даних. Крім того, БІ-аналітика дозволяє легко інтерпретувати обсяги даних. Виявлення нових можливостей і реалізація ефективної стратегії забезпечують конкурентну ринкову перевагу і довгострокову стабільність.

Масова аналітика застосовується, коли розмір даних перевищує всю ємність продукту БІ-аналізу і традиційних баз даних. У масовій аналітиці використовується розподілена файлова система Hadoop для зберігання даних. Масова аналітика допомагає створити основу для бізнесу і підвищує конкурентоспроможність на ринку за рахунок вилучення значимих значень з даних. Більш того, масивна аналітика дозволяє отримувати точні дані, які дозволяють використовувати ризики, пов'язані з прийняттям будь-якого бізнес-рішення [30].

Для СОА є сенс вибрати аналітику в режимі реального часу. Для неї представлені методи аналізу великих даних в категоріях класифікації, кластеризації, аналізу асоціативних правил і прогнозування. На рисунку 2.2 представлена і узагальнена кожна з цих категорій. Кожна категорія являє собою функцію інтелектуального аналізу даних і включає безліч методів і алгоритмів для виконання вимог вилучення та аналізу інформації. Наприклад, баєсова мережа, машина опорних векторів (SVM) і k-nearest neighbor (KNN) пропонують методи класифікації. Так само в кластеризації дуже поширено поділ, ієрархічна кластеризація і спільне виникнення [30].

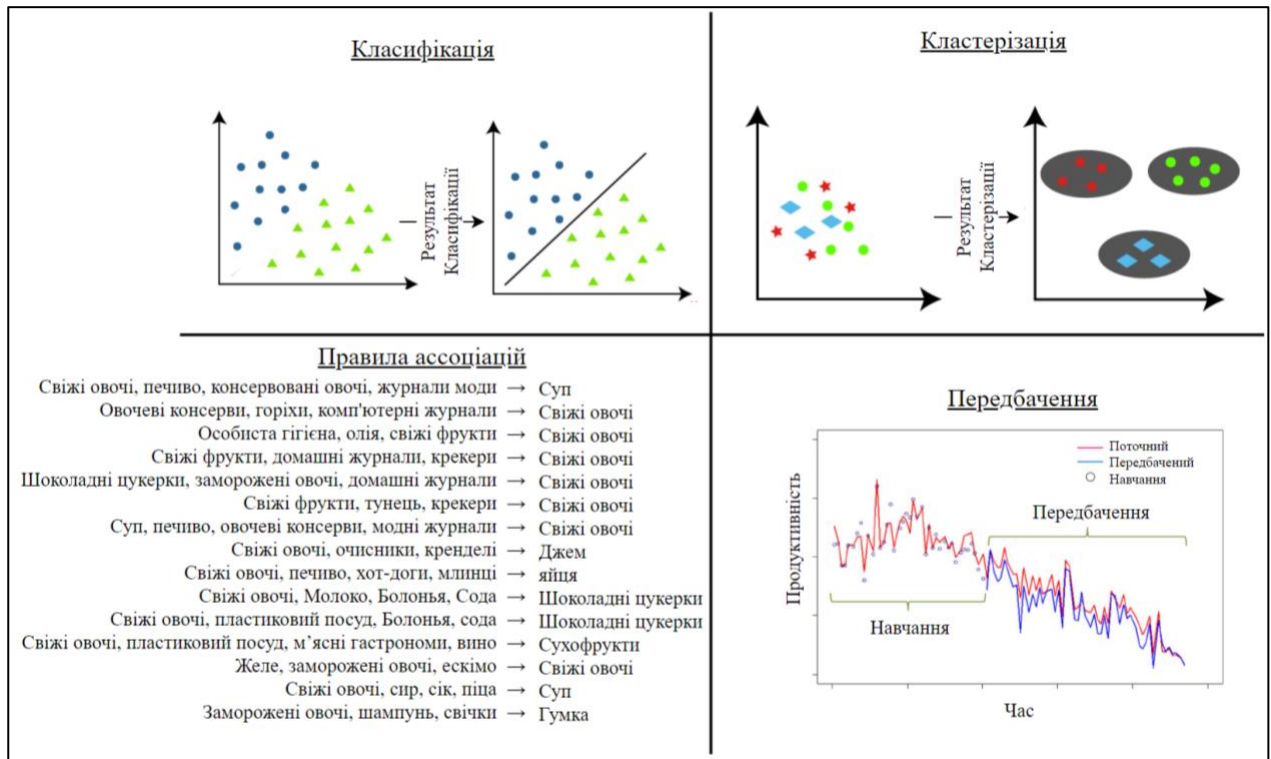


Рисунок 2.2 – Ілюстрації типів аналітик у режимі реального часу ІА [30]

Класифікація – це підхід до навчання з учителем, який використовує попередні знання в якості навчальних даних для класифікації об'єктів даних по групам. Зумовлена категорія присвоюється об'єкту, і, таким чином, досягається мета прогнозування групи або класу для об'єкта (див. рис. 2.2). Пошук невідомих або прихованих закономірностей є більш складним завданням для аналізу даних ІА. Крім того, критично важливим завданням є отримання цінної інформації з великих наборів даних для покращення системи прийняття рішень.

Аналіз шаблонів даних і створення груп ефективно виконуються за допомогою SVM, який також є підходом класифікації для аналізу великих даних. Класифікація має інші розширення, в той же час застосовуючи велику кількість методів штучного інтелекту та інтелектуального аналізу даних. Як наслідок, класифікація є одним з широко поширених методів інтелектуального аналізу даних для аналізу Big Data [30].

Кластеризація – це ще один метод інтелектуального аналізу даних, який використовується в якості методу аналізу даних. На відміну від класифікації, кластеризації використовує підхід навчання без вчителя і створює групи для заданих об'єктів на основі їх відмінних значущих характеристик. Як зображено на рисунку 2.2, угруповання великої кількості об'єктів у вигляді кластерів спрощує маніпулювання даними. Добре відомими методами, використовуваними для кластеризації, є ієрархічна кластеризація і поділ. Підхід з ієрархічною кластеризацією продовжує об'єднувати невеликі кластери об'єктів даних для формування ієрархічного дерева і створення агломеративних кластерів. Розділові кластери створюються протилежним чином шляхом ділення одного кластера, що містить всі об'єкти даних, на більш дрібні відповідні кластери [30].

Аналіз ринку та прийняття бізнес-рішень – найбільш важливі програми аналітики великих даних. Процес інтелектуального аналізу правил асоціації включає виявлення цікавих для бізнесу відносин між різними об'єктами, подіями або іншими об'єктами для аналізу ринкових тенденцій, купівельної поведінки споживачів і прогнозів попиту на продукцію (див. рис. 2.2). Аналіз асоціативних правил фокусується на ідентифікації та створення правил на основі частоти появи числових і нечислових даних [30].

Прогнозна аналітика використовує історичні дані, відомі як дані навчання, для визначення результатів у вигляді тенденцій або поведінки даних. SVM і алгоритми нечіткої логіки використовуються для визначення відносин між незалежними і залежними змінними і для отримання кривих регресії для прогнозів, або для стихійних лих. Крім того, прогнози покупців і тенденції в соціальних мережах аналізуються за допомогою прогновної. У разі аналітики великих даних вимоги до обробки змінюються в залежності від характеру і обсягу даних. Швидкий доступ до даних і методи інтелектуального аналізу структурованих і неструктурованих даних – основні проблеми, пов'язані з аналітикою великих даних. Крім того, подання даних є важливою вимогою в

аналітиці великих даних. Аналіз часових рядів знижує високу розмірність, пов'язану з великими даними, і пропонує уявлення для поліпшеного прийняття рішень [30].

Для ефективного використання СОА користувачі потребують прогнозування даних енергоспоживання та попередження потенційних загроз внаслідок нераціонального користування обладнання та як наслідок – покращення досвіду взаємодії із системою, та можливість раціонально використовувати ресурси СОА.

Тому ІА прогнозованої аналітики може бути рекомендовано до використання у СОА.

2.2 Огляд алгоритмів ІА

Прогнозування відбувається за рахунок алгоритмів прогнозованої аналітики. На сьогодні найпоширенішими алгоритмами є: Tree, kNN, Random Forest, SVM, AdaBoost, Linear Regression, Neural Network та Stochastic Gradient Descent. На рисунку 2.3 зображено принцип роботи кожного з алгоритмів.

Принцип роботи Tree – це графічне представлення всіх можливих рішень рішення на основі певних умов. На кожному кроці або вузлі дерева рішень, що використовується для класифікації, формується умова щодо ознак, щоб розділити всі мітки або класи, що містяться в наборі даних, з максимальною чистотою [37]. Вони нестабільні, а це означає, що невелика зміна даних може призвести до значної зміни структури оптимального дерева рішень. Часто вони відносно неточні. Багато інших предикторів працюють краще з подібними даними.

Принцип kNN – знаходження відстані між запитом і всіма прикладами в даних, вибираючи вказане число прикладів (K), найближче до запиту, потім голосує за найчастішу мітку (у разі класифікації) або усереднює мітки (у випадок регресії) Алгоритм k-найближчих сусідів (KNN) — це простий, контрольований

алгоритм машинного навчання, який можна використовувати для вирішення як задач класифікації, так і регресії. Його легко реалізувати та зрозуміти, але він має серйозний недолік, оскільки він значно сповільнюється зі збільшенням розміру використовуваних даних [38].

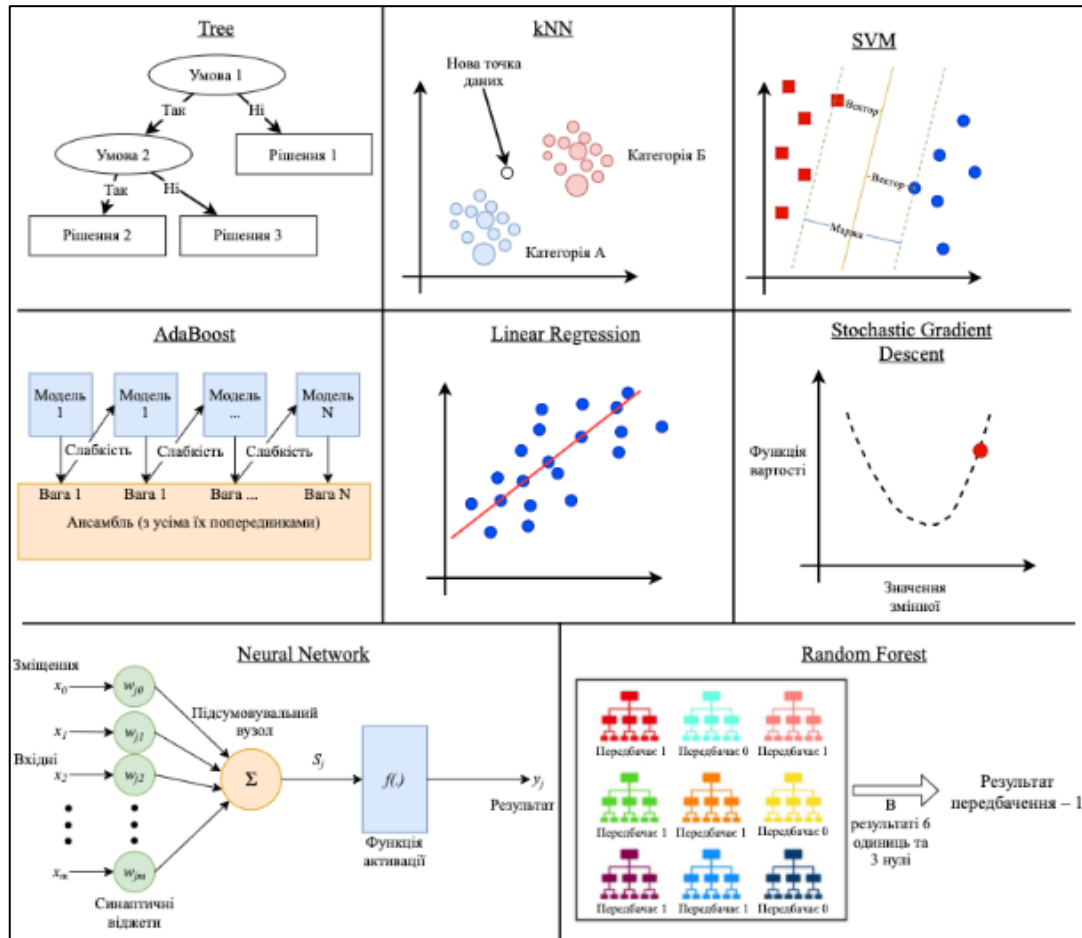


Рисунок 2.3 – Візуалізація принципів роботи алгоритмів прогнозованої аналітики [37]-[44]

Random Forest – це комбінація предикторів дерев, де кожне дерево залежить від значень випадкового вектора, відібраного незалежно з однаковим розподілом для всіх дерев у лісі. Основний принцип полягає в тому, що група «слабких учнів» може об'єднатися, щоб сформувати «сильного учня». Random Forest, як випливає з його назви, складається з великої кількості окремих дерев

рішень, які діють як ансамбль. Кожне окреме дерево у випадковому лісі видає прогноз класу, і клас з найбільшою кількістю голосів стає прогнозуванням моделі [39].

Ідея SVM проста: алгоритм створює лінію або гіперплощину, яка розділяє дані на класи. Відповідно до алгоритму SVM треба знайти точки, найближчі до прямої з обох класів. Ці точки називають опорними векторами. Наступний крок – це обчислення відстані між прямою та опорними векторами. Ця відстань називається маржею. Мета алгоритму — максимізувати маржу. Гіперплощина, для якої запас є максимальним, є оптимальною гіперплощиною. Таким чином, SVM намагається прийняти межу рішення таким чином, щоб поділ між двома класами (цією вулицею) був якомога ширшим [40].

Алгоритм AdaBoost, скорочення від Adaptive Boosting, є технікою Boosting, яка використовується як метод ансамблю в машинному навчанні. Це називається адаптивним підвищенням, оскільки ваги повторно призначаються кожному екземпляру, а вищі ваги призначаються неправильно класифікованим екземплярам. Підвищення використовується для зменшення упередженості, а також дисперсії для навчання під керівництвом. Він працює за принципом послідовного зростання учнів. За винятком першого, кожен наступний учень виростає з раніше дорослих учнів. Простіше кажучи, слабкі учні перетворюються на сильних. Алгоритм AdaBoost працює за тим же принципом, що і прискорення з невеликою різницею [41].

Linear Regression – це спосіб моделювання зв'язку між однією або кількома змінними. З точки зору машинного навчання, це робиться для забезпечення узагальнення, що дає моделі можливість прогнозувати вихідні дані для вхідних даних, яких вона ніколи не бачила. Суть Linear Regression полягає в тому, щоб знайти лінію, яка найкраще відповідає точкам даних на графіку, щоб ми могли більш-менш точно знати, де, ймовірно, впаде ціна акцій у 2021 році тощо [42].

Neural network – це паралельні системи обробки інформації, що складаються з ряду простих нейронів (також званих вузлами або блоками), які організовані в шари і з'єднані за допомогою зв'язків. Тобто це серія алгоритмів, які намагаються розпізнати основні зв'язки в наборі даних за допомогою процесу, який імітує роботу мозку людини. У цьому сенсі нейронні мережі відносяться до систем нейронів, як органічних, так і штучних за своєю природою [43]. Проте коли Neural Network дає пробне рішення, воно не дає уявлення про те, чому і як, що знижує довіру до мережі. Тривалість мережі невідома – мережа зменшена до певного значення помилки на вибірці, це означає, що навчання завершено. Це значення не дає нам оптимальних результатів.

Stochastic Gradient Descent (SGD) – це ітераційний алгоритм, який починається з випадкової точки функції і рухається по її схилу кроками, поки не досягне найнижчої точки цієї функції. Основна концепція SGD полягає в тому, що вам не потрібно аналізувати всі дані відразу, щоб рухатися вперед. Вам просто потрібна підмножина даних, щоб рухатися вперед, а потім продовжувати повторювати той самий процес, поки не досягнете наміченого пункту призначення або принаймні близько до нього [44].

За результатами дослідження можна рекомендувати усі алгоритми, оскільки усі вони можуть бути використані для вирішення поставленої задачі. Для обирання конкретного алгоритму слід виконати їх порівняння на вибірці даних. В якості вибірки даних було прийнято рішення використати вибірку даних існуючої СОА.

2.3 Огляд програмних засобів для реалізації алгоритмів ІА

Наступним кроком є вибір програмних засобів або бібліотек для ІА даних, що використовують описані алгоритми. Дослідження показали, що існує ряд програмних засобів (RapidMiner, Orange, Weka, KNIME), які дозволяють

аналізувати великі набори даних та використовувати для них різні алгоритми обробки даних та машинного навчання [45] - [46].

Порівняння цих засобів та бібліотек зображено в таблиці 2.1.

Таблиця 2.1 – Порівняння програмних засобів та бібліотек

Назва	Підтримка операційних систем	Підтримка алгоритмів	Важкість конфігурації	Можливість порівнювати результати
RapidMiner	Кросплатформна, працює за платну підписку.	Підтримка усіх сучасних алгоритмів.	Усе налаштовується перетягуванням необхідних блоків на схему.	Є можливість для порівняння результатів, отриманих окремими обчисленнями.
Orange	Кросплатформна, може бути встановлена як бібліотека.	Підтримка усіх сучасних алгоритмів	Усе конфігурується перетягуванням значків на головний екран.	Є можливість для порівняння результатів.
Weka	Кросплатформна, але потребує конкретну версію Java встановлену на ПК.	Підтримка усіх сучасних алгоритмів	Застарілий інтерфейс та незручне знаходження файлів, які виконують роль правил для алгоритмів.	Є можливість для поверхневого та базового порівнянь.
KNIME	Кросплатформна, але працює за платну підписку.	Підтримка усіх сучасних алгоритмів	Усе конфігурується на графічному інтерфейсі шляхом перетяганням необхідних блоків програм.	Є можливість для порівняння результатів, отриманих окремими обчисленнями.

На основі порівняльного аналізу може бути рекомендована програма Orange. Вона надає ряд віджетів для створення асоціативних правил, алгоритмів машинного навчання, впровадження та оцінки результатів [47]. Також ця утиліта є безкоштовною. Повноцінний графічний інтерфейс може бути встановленим в операційну систему як бібліотека для Python, що робить встановлення дуже швидким та зручним. Також наявна можливість встановити додаткові модулі, використовуючи вкладку «Options» а в ній кнопку «Add-ons» у самій програмі, або лише встановивши додаткові бібліотеки для Python, використовуючи пакетний менеджер цієї мови програмування, а програма Orange автоматично збере їх усі та додасть у список компонентів на головній сторінці.

2.4 Аналіз вибірки даних для порівняння алгоритмів

Для коректного порівняння алгоритмів необхідно сформувати вибірку даних з датчиків. Головним датчиком представленої СОА є сенсор електричного струму. Він заснований на датчику PZEM-004-100A. На ньому зав'язані усі налаштовані датчики в системі, він являтиме собою першу причину потенційної загрози для користувача. На основі даних цього датчика можна проводити аналіз використання ресурсів СОА. Тому саме на даних цього датчика буде проводитись порівняння.

Дані цього датчика надходять з серверу СОА під час роботи системи та записуються у базі даних. Дана БД складається з двох сутностей: «states» та «events». Структуру даної БД наведено на рисунку 2.4.

На рисунку 2.4 блакитним позначено назву сутності, помаранчевим унікальний ідентифікатор кожної сутності.

Сутність «states» зберігає дані, зібрані з датчиків, а сутність «events» зберігає унікальні ідентифікатори подій.

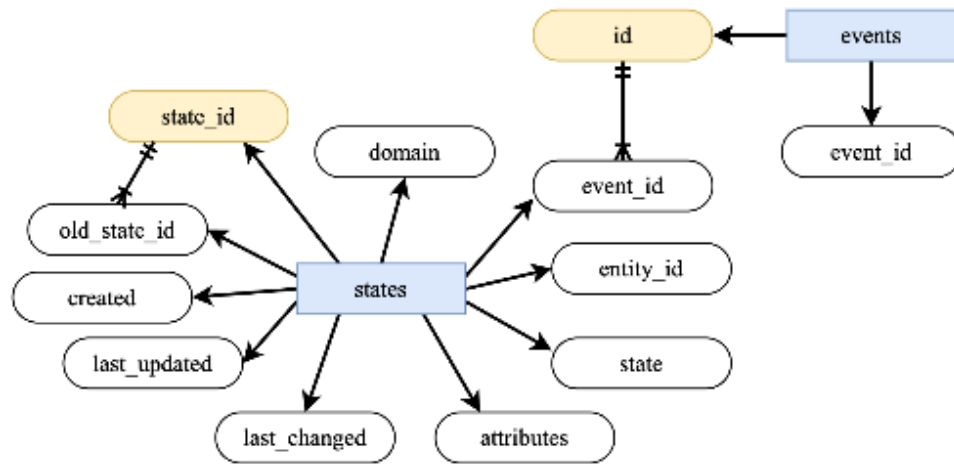


Рисунок 2.4 – Структура БД на серверах СОА

Для тестів було зроблено вибірку цієї бази даних, а саме таблиці «states». Обрана таблиця містить дані, які є сенс проаналізувати. Структура даних цієї таблиці включає колонки, що описано в таблиці 2.2.

Таблиця 2.2 – Опис структури даних таблиці «states»

Назва	Опис	Тип даних
state_id	Ідентифікаційний номер запису в БД	INT
domain	Тип даних, може бути sensor або switch.	VARCHAR(64)
entity_id	Тип та назва датчика, що зробив запис	VARCHAR(255)
state	Стан датчиків на момент створення запису	VARCHAR(255)
attributes	Додаткова інформація з датчику	LONGTEXT
event_id	Посилання на допоміжну таблицю з подіями	INT
last_changed	Час, коли було зроблено останні зміни	DATETIME
last_updated	Час, коли було виконано останнє оновлення	DATETIME
created	Час коли було створено запис в БД	DATETIME
old_state_id	Посилання на попередній запис певного датчика	INT

Розмір даної таблиці складає 3 944 712 записів. На рисунку 2.5 наведено приклад повної вибірки даних.

Homeassistant states (3,046,712 rows total (approximately), limited to 1,000)

state_id	domain	entity_id	state	attributes	event_id	last_changed	last_updated	created	old_state_id	...
1.764.432	sensor	sensor_time	04:12	{friendly_name: "Time", icon: "mdi:clock"}	1.767.360	2021-08-17 12:00:00.000297	2021-08-17 12:00:00.000297	2021-08-17 12:00:00.000297		...
1.764.433	sensor	sensor_date_time	2021-08-17 04:12	{friendly_name: "Date & Time", icon: "mdi:calendar"}	1.767.361	2021-08-17 12:00:00.000609	2021-08-17 12:00:00.000609	2021-08-17 12:00:00.000609		...
1.764.434	sensor	sensor_time_delta	04:12, 2021-08-17	{friendly_name: "Time & Date", icon: "mdi:calendar"}	1.767.362	2021-08-17 12:00:00.001193	2021-08-17 12:00:00.001193	2021-08-17 12:00:00.001193		...
1.764.435	sensor	sensor_faza_a	230.4	{unit_of_measurement: "V", friendly_name: "faza_a"}	1.767.363	2021-08-17 12:00:00.007906	2021-08-17 12:00:00.007906	2021-08-17 12:00:00.007906		...
1.764.436	sensor	sensor_faza_b	209.9	{unit_of_measurement: "V", friendly_name: "faza_b"}	1.767.364	2021-08-17 12:00:00.008221	2021-08-17 12:00:00.008221	2021-08-17 12:00:00.008221		...
1.764.437	sensor	sensor_faza_c	237.4	{unit_of_measurement: "V", friendly_name: "faza_c"}	1.767.365	2021-08-17 12:00:00.021878	2021-08-17 12:00:00.021878	2021-08-17 12:00:00.021878		...
1.764.438	sensor	sensor_faza_a_w	85.9	{unit_of_measurement: "W", friendly_name: "faza_a_w"}	1.767.366	2021-08-17 12:00:00.029112	2021-08-17 12:00:00.029112	2021-08-17 12:00:00.029112		...
1.764.439	sensor	sensor_faza_b_w	238.4	{unit_of_measurement: "W", friendly_name: "faza_b_w"}	1.767.367	2021-08-17 12:00:00.030080	2021-08-17 12:00:00.030080	2021-08-17 12:00:00.030080		...
1.764.440	sensor	sensor_faza_c_w	21.7	{unit_of_measurement: "W", friendly_name: "faza_c_w"}	1.767.368	2021-08-17 12:00:00.031683	2021-08-17 12:00:00.031683	2021-08-17 12:00:00.031683		...
1.764.441	sensor	sensor_faza_a_v	51.5	{unit_of_measurement: "V", friendly_name: "faza_a_v"}	1.767.369	2021-08-17 12:00:00.032224	2021-08-17 12:00:00.032224	2021-08-17 12:00:00.032224		...
1.764.442	sensor	sensor_faza_b_v	63.8	{unit_of_measurement: "V", friendly_name: "faza_b_v"}	1.767.370	2021-08-17 12:00:00.033811	2021-08-17 12:00:00.033811	2021-08-17 12:00:00.033811		...
1.764.443	sensor	sensor_faza_c_v	239.3	{unit_of_measurement: "V", friendly_name: "faza_c_v"}	1.767.371	2021-08-17 12:00:00.037224	2021-08-17 12:00:00.037224	2021-08-17 12:00:00.037224		...
1.764.444	sensor	sensor_faza_a_h	1.7	{unit_of_measurement: "h", friendly_name: "faza_a_h"}	1.767.372	2021-08-17 12:00:00.038110	2021-08-17 12:00:00.038110	2021-08-17 12:00:00.038110		...
1.764.445	sensor	sensor_faza_b_h	309.3	{unit_of_measurement: "h", friendly_name: "faza_b_h"}	1.767.373	2021-08-17 12:00:00.038111	2021-08-17 12:00:00.038111	2021-08-17 12:00:00.038111		...
1.764.446	sensor	sensor_faza_c_h	274.463	{unit_of_measurement: "h", friendly_name: "faza_c_h"}	1.767.374	2021-08-17 12:00:00.038112	2021-08-17 12:00:00.038112	2021-08-17 12:00:00.038112		...
1.764.447	sensor	sensor_temperature_c_kitchen	20.76	{unit_of_measurement: "C", friendly_name: "kitchen"}	1.767.375	2021-08-17 12:00:00.039167	2021-08-17 12:00:00.039167	2021-08-17 12:00:00.039167		...
1.764.448	sensor	sensor_humidity_kitchen	50.46	{unit_of_measurement: "%", friendly_name: "kitchen"}	1.767.376	2021-08-17 12:00:00.039168	2021-08-17 12:00:00.039168	2021-08-17 12:00:00.039168		...
1.764.449	sensor	sensor_humidity_kitchen	50	{unit_of_measurement: "%", friendly_name: "kitchen"}	1.767.377	2021-08-17 12:00:00.039169	2021-08-17 12:00:00.039169	2021-08-17 12:00:00.039169		...
1.764.450	sensor	sensor_cool_c_kitchen	407	{unit_of_measurement: "ppm", friendly_name: "c_kitchen"}	1.767.378	2021-08-17 12:00:00.039170	2021-08-17 12:00:00.039170	2021-08-17 12:00:00.039170		...
1.764.451	sensor	sensor_faza_a	237.7	{unit_of_measurement: "V", friendly_name: "faza_a"}	1.767.379	2021-08-17 12:00:00.039171	2021-08-17 12:00:00.039171	2021-08-17 12:00:00.039171		...
1.764.452	sensor	sensor_faza_b	6.87	{unit_of_measurement: "h", friendly_name: "faza_b"}	1.767.380	2021-08-17 12:00:00.039172	2021-08-17 12:00:00.039172	2021-08-17 12:00:00.039172		...
1.764.453	sensor	sensor_faza_c	86.8	{unit_of_measurement: "h", friendly_name: "faza_c"}	1.767.381	2021-08-17 12:00:00.039173	2021-08-17 12:00:00.039173	2021-08-17 12:00:00.039173		...
1.764.454	sensor	sensor_faza_d	238	{unit_of_measurement: "V", friendly_name: "faza_d"}	1.767.382	2021-08-17 12:00:00.039174	2021-08-17 12:00:00.039174	2021-08-17 12:00:00.039174		...
1.764.455	sensor	sensor_faza_e	234	{unit_of_measurement: "V", friendly_name: "faza_e"}	1.767.383	2021-08-17 12:00:00.039175	2021-08-17 12:00:00.039175	2021-08-17 12:00:00.039175		...
1.764.456	sensor	sensor_faza_f	53.1	{unit_of_measurement: "h", friendly_name: "faza_f"}	1.767.384	2021-08-17 12:00:00.039176	2021-08-17 12:00:00.039176	2021-08-17 12:00:00.039176		...
1.764.457	sensor	sensor_faza_g	607.487	{unit_of_measurement: "h", friendly_name: "faza_g"}	1.767.385	2021-08-17 12:00:00.039177	2021-08-17 12:00:00.039177	2021-08-17 12:00:00.039177		...
1.764.458	sensor	sensor_faza_h	6.88	{unit_of_measurement: "h", friendly_name: "faza_h"}	1.767.386	2021-08-17 12:00:00.039178	2021-08-17 12:00:00.039178	2021-08-17 12:00:00.039178		...
1.764.459	sensor	sensor_temperature_c_kitchen	20.80	{unit_of_measurement: "C", friendly_name: "kitchen"}	1.767.387	2021-08-17 12:00:00.039179	2021-08-17 12:00:00.039179	2021-08-17 12:00:00.039179		...
1.764.460	sensor	sensor_humidity_kitchen	50.55	{unit_of_measurement: "%", friendly_name: "kitchen"}	1.767.388	2021-08-17 12:00:00.039180	2021-08-17 12:00:00.039180	2021-08-17 12:00:00.039180		...
1.764.461	sensor	sensor_cool_c_kitchen	5	{unit_of_measurement: "ppm", friendly_name: "c_kitchen"}	1.767.389	2021-08-17 12:00:00.039181	2021-08-17 12:00:00.039181	2021-08-17 12:00:00.039181		...
1.764.462	sensor	sensor_cool_c_kitchen	404	{unit_of_measurement: "ppm", friendly_name: "c_kitchen"}	1.767.390	2021-08-17 12:00:00.039182	2021-08-17 12:00:00.039182	2021-08-17 12:00:00.039182		...
1.764.463	sensor	sensor_faza_a	235.1	{unit_of_measurement: "V", friendly_name: "faza_a"}	1.767.391	2021-08-17 12:00:00.039183	2021-08-17 12:00:00.039183	2021-08-17 12:00:00.039183		...
1.764.464	sensor	sensor_faza_b	1.7	{unit_of_measurement: "h", friendly_name: "faza_b"}	1.767.392	2021-08-17 12:00:00.039184	2021-08-17 12:00:00.039184	2021-08-17 12:00:00.039184		...
1.764.465	sensor	sensor_faza_c	309.6	{unit_of_measurement: "h", friendly_name: "faza_c"}	1.767.393	2021-08-17 12:00:00.039185	2021-08-17 12:00:00.039185	2021-08-17 12:00:00.039185		...
1.764.466	sensor	sensor_cool_c_kitchen	31.0207	{friendly_name: "CO2_kitchen"}	1.767.394	2021-08-17 12:00:00.039186	2021-08-17 12:00:00.039186	2021-08-17 12:00:00.039186		...
1.764.467	sensor	sensor_cool_c_kitchen	50.0208	{friendly_name: "CO2_kitchen"}	1.767.395	2021-08-17 12:00:00.039187	2021-08-17 12:00:00.039187	2021-08-17 12:00:00.039187		...
1.764.468	sensor	sensor_cool_c_kitchen	394	{friendly_name: "CO2_kitchen"}	1.767.396	2021-08-17 12:00:00.039188	2021-08-17 12:00:00.039188	2021-08-17 12:00:00.039188		...
1.764.469	sensor	sensor_faza_a	237.8	{unit_of_measurement: "V", friendly_name: "faza_a"}	1.767.397	2021-08-17 12:00:00.039189	2021-08-17 12:00:00.039189	2021-08-17 12:00:00.039189		...
1.764.470	sensor	sensor_faza_b	6.86	{unit_of_measurement: "h", friendly_name: "faza_b"}	1.767.398	2021-08-17 12:00:00.039190	2021-08-17 12:00:00.039190	2021-08-17 12:00:00.039190		...
1.764.471	sensor	sensor_faza_c	85.9	{unit_of_measurement: "h", friendly_name: "faza_c"}	1.767.399	2021-08-17 12:00:00.039191	2021-08-17 12:00:00.039191	2021-08-17 12:00:00.039191		...
1.764.472	sensor	sensor_faza_d	237.8	{unit_of_measurement: "V", friendly_name: "faza_d"}	1.767.400	2021-08-17 12:00:00.039192	2021-08-17 12:00:00.039192	2021-08-17 12:00:00.039192		...

Рисунок 2.5 – Приклад вибірки показань датчику електричного струму

Оскільки дана БД містить великий об'єм даних, що в необробленому вигляді не несуть цінності, тому для аналізу слід використати технології та алгоритми Big Data, оскільки вони вирішують подібні задачі [48].

Для більш швидкого та зручного використання у алгоритмах ІА було прийнято рішення прибрати непотрібні колонки у таблиці, відфільтрувати дані, щоб у вибірці лишились тільки показники датчику електроенергії та згенерувати дані у форматі CSV. На рисунку 2.6 зображено список датчиків, що відображають список таких датчиків у СДА, з якої було взято вибірку бази даних.

```

1  USE homeassistant;
2  SELECT DISTINCT entity_id FROM states
3  WHERE entity_id LIKE "%faza_%";

```

states (18x 1c)

entity_id
sensor.faza_a_a
sensor.faza_a_hz
sensor.faza_a_kwh
sensor.faza_a_pf
sensor.faza_a_v
sensor.faza_a_w
sensor.faza_b_a
sensor.faza_b_hz
sensor.faza_b_kwh
sensor.faza_b_pf
sensor.faza_b_v
sensor.faza_b_w
sensor.faza_s_a
sensor.faza_s_hz
sensor.faza_s_kwh
sensor.faza_s_pf
sensor.faza_s_v
sensor.faza_s_w

Рисунок 2.6 – Список датчиків електроживлення СДА

У цьому списку показники з датчика, що складається з трьох фаз, що пронумеровані відповідно «faza_a», «faza_b», «faza_s». Цей датчик записує наступні показники:

- «a» – сила струму;
- «hz» – частота струму;
- «kwh» – кількість Вольт на годину;
- «pf» – електрична ємність струму;
- «v» – напруга струму;
- «w» – показник вольтажу в системі.

Скорочені назви кожного показника зберігаються у суфіксі назви, наприклад сила струму фази «a» у вибірці матиме назву «sensor.faza_a_a», тощо.

Попередження потенційних небезпек є важливим для користувача пунктом в автоматизації. Саме тому найважливішим та найвагомішими показником для користувача є показник напруги в мережі. Адже через стрибки напруги, або не раціональне використання ресурсів відбуваються короткі замикання та як наслідок загоряння електричних дротів, потенційні пожежі у будинках та квартирах. Саме тому, щоб запобігти таким ситуаціям користувачеві буде корисно знати потенційне значення напруги в мережі, як на поточний момент часу, так і на майбутні конкретні дні, на майбутній тиждень та місяць.

Для цього прийнято рішення розробити окремий запит, що дозволить зробити вибірку даних значення напруги, згрупованих за днем та сенсором, що створив запис. Так як дані для «last_updated» зберігаються у форматі DATETIME, а для алгоритмів простіше і швидше оперувати із числовими значеннями, тому було прийнято рішення конвертувати цей формат у число, що складає суму з року, місяця та числа. Приклад такої вибірки наведено на рисунку 2.7.

1	USE homeassistant;
2	
3	SELECT
4	YEAR(last_changed) * 10000 + MONTH(last_changed) * 100 + DAY(last_changed) AS updated,
5	ROUND(state, 3) AS avg_state
6	
7	FROM states
8	WHERE
9	entity_id LIKE "%faza_%v"
10	LIMIT 500000
11	--

states (500,000r × 2c)	
updated	avg_state
20,210,817	230.400
20,210,817	237.400
20,210,817	238.400
20,210,817	230.300
20,210,817	237.700
20,210,817	238.000
20,210,817	230.100
20,210,817	237.800
20,210,817	237.800
20,210,817	230.400
20,210,817	237.200
20,210,817	238.100

Рисунок 2.7 – Приклад вибірки

Дану вибірку даних було збережено у CSV файли для подальшого зручного використання в системі Orange. Використання даного формату файлу дасть можливість швидше обробити та налаштувати систему Orange на використання створеної вибірки.

2.5 Порівняння алгоритмів ІА

Для вибору найкращого алгоритму, треба порівняти їх середню величину похибки на існуючій вибірці, що було обрано в підрозділі 2.4. Для порівняння величини похибки необхідно використати програму Orange, яку було обрано в підрозділі 2.3. Схему системи для тестування наведено на рисунку. 2.8.

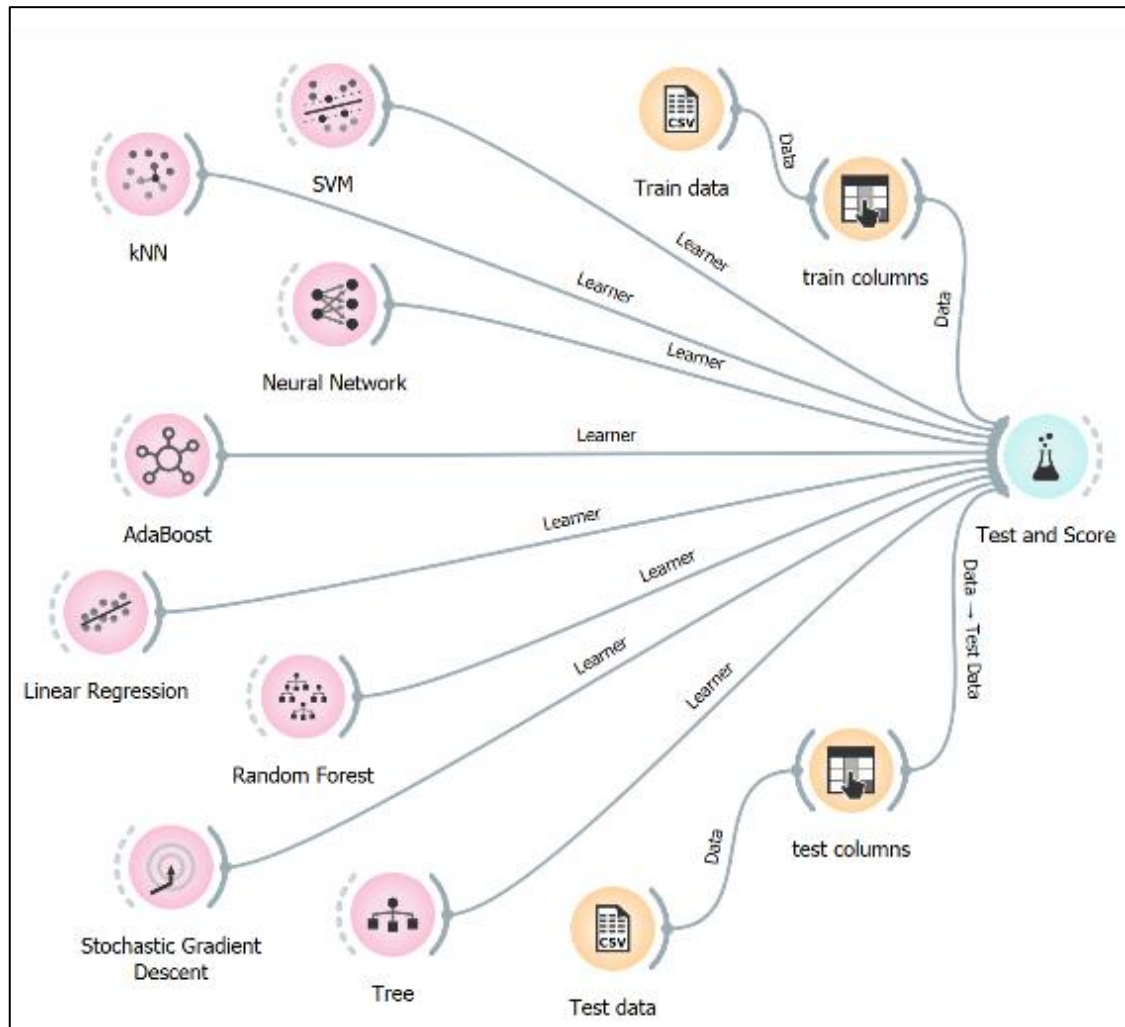


Рисунок 2.8 – Схема системи для тестування

В елемент «Train data» треба завантажити файл csv, що було згенеровано раніше та налаштувати колонки, обрати «target» та «features». Після цього автоматично запуститься навчання та тестування алгоритмів. В результаті отримаємо результати, що наведено на рисунку 2.9. Де а – тренування на вибірці розміром 5 000 записів, б – 10 000 записів, в – 100 000 записів, г – 500 000 записів. Згідно з результатів можна зробити висновок, що зі збільшенням величини вибірки зменшується величина похибки (MAE). Відображення похибок для кожного алгоритму та для кожної вибірки наведено на рисунку 2.10.

Test and Score				
Evaluation Results				
Model	MSE	RMSE	MAE	R2
kNN	3.719	1.928	1.720	-3.898
SVM	4.295	2.072	1.880	-4.656
AdaBoost	6.623	2.573	2.421	-7.722
Tree	6.990	2.644	2.496	-8.205
Linear Regression	6.990	2.644	2.496	-8.205
Random Forest	7.128	2.670	2.524	-8.387
SGD	10.167	3.188	3.067	-12.389
Neural Network	20.112	4.485	3.756	-25.487

a)

Test and Score				
Evaluation Results				
Model	MSE	RMSE	MAE	R2
SVM	1.368	1.170	0.898	-0.802
kNN	3.719	1.928	1.720	-3.898
AdaBoost	4.100	2.025	1.828	-4.399
Tree	5.677	2.383	2.217	-6.476
Linear Regression	5.677	2.383	2.217	-6.476
SGD	5.707	2.389	2.224	-6.515
Random Forest	5.747	2.397	2.233	-6.568
Neural Network	20.988	4.581	3.834	-26.641

б)

Test and Score				
Evaluation Results				
Model	MSE	RMSE	MAE	R2
AdaBoost	0.366	0.605	0.487	0.518
kNN	2.892	1.701	1.420	-2.809
Tree	5.189	2.278	2.107	-5.834
Random Forest	5.247	2.291	2.120	-5.910
Neural Network	429.930	20.735	17.008	-565.193
Linear Regression	655.759	25.608	21.176	-862.598
SGD	699.088	26.440	21.911	-919.660
SVM	12292.836	110.873	110.870	-16187.973

в)

Test and Score				
Evaluation Results				
Model	MSE	RMSE	MAE	R2
Tree	0.268	0.518	0.352	0.647
Random Forest	0.343	0.586	0.460	0.548
AdaBoost	0.423	0.651	0.546	0.442
Neural Network	0.922	0.960	0.745	-0.215
Linear Regression	2.013	1.419	1.139	-1.650
SGD	3.989	1.997	1.817	-4.253
SVM	8.156	2.856	2.720	-9.741
kNN	17.437	4.176	4.048	-21.964

г)

Рисунок 2.9 – Результати порівняння

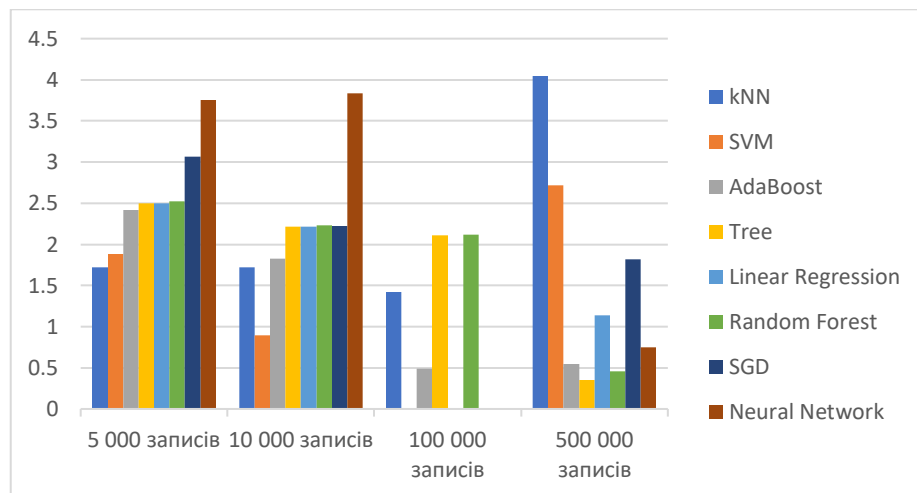


Рисунок 2.10 – Значення похибки алгоритмів на вибірках

Згідно порівняння відносно похибки є сенс обрати алгоритм Tree, бо він має найнижче значення – 35%. Проте даний алгоритм нестабільний, а це означає, що невелика зміна даних може призвести до значної зміни структури оптимального дерева рішень. Часто вони відносно неточні. Багато інших предикторів працюють краще з подібними даними. Та сама ситуація і з Random Forest. Тому є сенс обрати алгоритм AdaBoost, тому що він має відносно стабільне значення похибки та за рахунок збільшення кількості даних даний алгоритм надаватиме кращі та більш детальні результати, проте він має зовелику величину похибки.

Сьогодні сучасні платформи мають можливість використати технології даних алгоритмів та поєднати. Оскільки для наших цілей може бути рекомендовано два алгоритми – це AdaBoost та Tree, тому є сенс використати для розроблюваного програмного продукту поєднання даних алгоритмів – Boosted Decision Tree [49].

Дане поєднання дасть можливість використати сильні сторони обох алгоритмів, а саме використовуючи техніку Boosting з алгоритму AdaBoost та використовуючи представлення всіх можливих рішень рішення на основі певних умов.

2.6 Висновки до розділу 2

В даному розділі було проведено аналіз типів ІА даних для систем типу СОА. В результаті аналізу може бути рекомендовано алгоритми ІА прогнозованої аналітики.

Аналіз існуючих алгоритмів показав що, є їх досить багато, найпоширенішими алгоритмами є: Tree, kNN, Random Forest, SVM, AdaBoost, Linear Regression, Neural Network та Stochastic Gradient Descent. Для вибору

найбільш відповідних для поставлених задач було проведено дослідження на вибірці з існуючої системи типу СОА за допомогою програми Orange.

Згідно з порівняння відносно похибки є сенс обрати гібридний алгоритм Boosted Decision Tree, бо він має найкраще з двох алгоритмів AdaBoost та Tree, що показали себе найкраще у аналізи обраної вибірки.

Але цей алгоритм потребує подальшого розвитку оскільки для вирішення поставлених задач, а саме для використання технологій та алгоритмів Big Data, необхідно удосконалити етапи підготовки та виведення даних для покращення точності прогнозування даних енергоспоживання.

3 ПРОЄКТУВАННЯ СИСТЕМИ

3.1 Аналіз вимог до системи

Система призначена для організації спеціального інтерфейсу для ІА даних з датчиків енергоспоживання СДА.

Система повинна відповідати наступним функціональним вимогам:

- надати засоби перегляду історичних даних напруги у мережі за тиждень та місяць;
- надати засоби для прогнозування напруги на обраний користувачем інтервал;
- отримати повні показники з датчиків енергоспоживання за конкретний день.

Система повинна відповідати наступним нефункціональним вимогам:

- працювати в серверній системі;
- надавати доступ тільки для авторизованих користувачів;
- надавати інтуїтивно зрозумілий та функціональний інтерфейс;
- вартість обслуговування повинна бути мінімальною.

Початкові дані – це дати, що користувач надсилатиме для отримання потенційних даних електроспоживання, та опції, що будуть обрані для взаємодії із системою. Користувач повинен бути авторизованим ви системі, щоб мати можливість взаємодіяти із системою.

У якості вихідних даних користувач отримуватиме можливість взаємодіяти із системою використовуючи функціонал графічного інтерфейсу. Користувач отримає дані прогнозування та історичні дані (фактичне значення енергоспоживання за обраний часовий інтервал).

Аналіз взаємодії із розроблюваною системою наведено на рисунку 3.1 у вигляді UML-діаграми.

На UML-діаграмі зображено, що користувач повинен авторизуватися перш ніж взаємодіяти із розроблюваною системою. Це обумовлено тим, щоб забезпечити безпеку системи, щоб тільки реальні користувачі мали доступ до даних енергоспоживання системи.

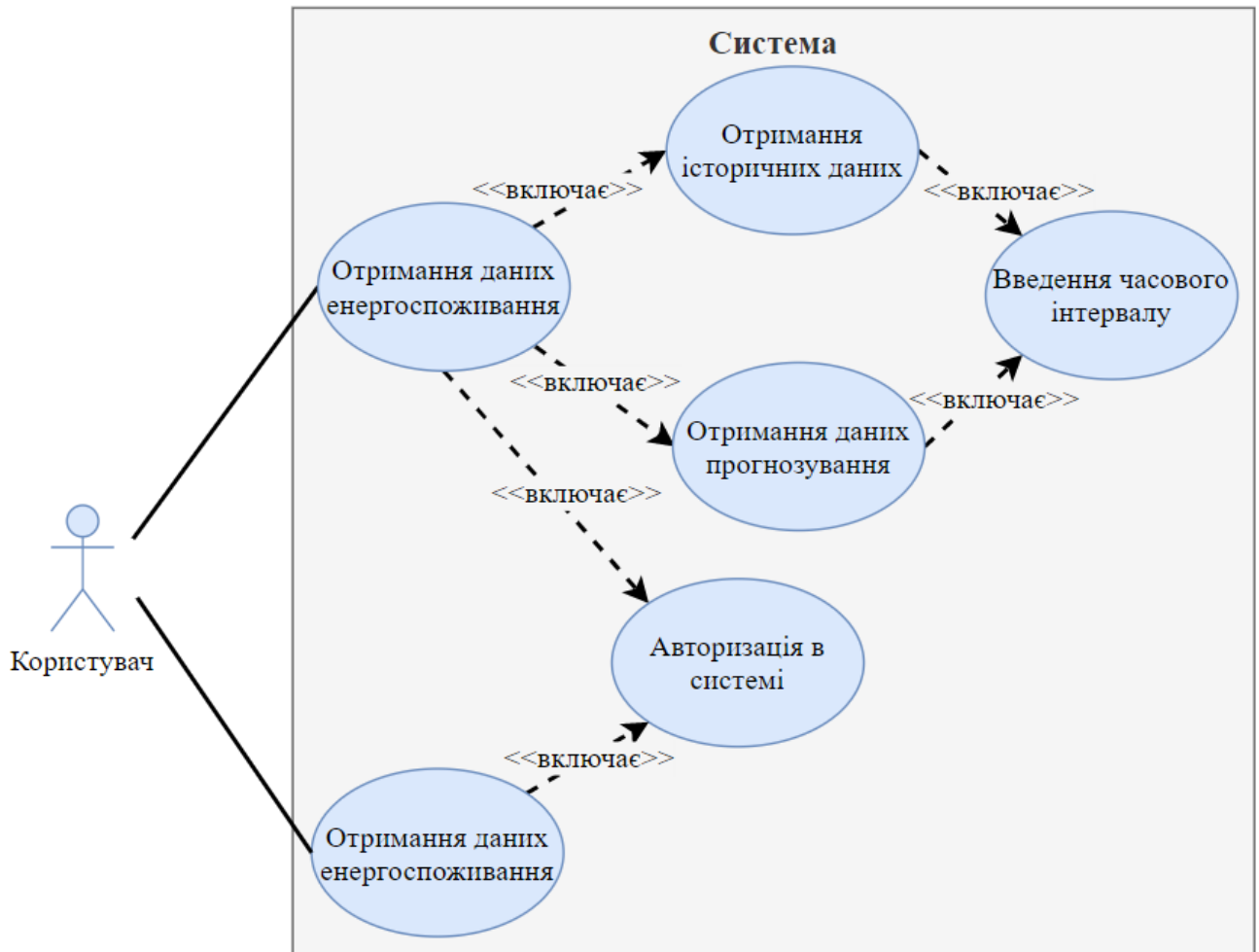


Рисунок 3.1 – Діаграма прецедентів використання

Наступним кроком є взаємодія із системою шляхом отримання даних історичних та даних прогнозування. Це відбуватиметься шляхом обирання часового проміжку, за який стоїть задача отримати дані. Щоб завершити роботу із системою треба виконати вихід з неї.

3.1.1 Аналіз серверних систем

Оскільки відсутня можливість запускати сховище даних в системі Home Assistant, тому є сенс запускати БД на сторонньому сервері, або хостингу. Так як будуть використані кошти для підтримки хостингу, тому є сенс запустити розроблювану систему на тому ж хостингу, щоб мати цілісність екосистеми.

Є 3 основні категорії серверних систем та платформ, які наведено в таблиці 3.1.

Таблиця 3.1 – Порівняння категорій серверних систем

Назва групи	Переваги	Недоліки
Власний сервер.	Дешевий у експлуатації, оскільки треба платити тільки за електроенергію.	Платити за всі частини сервера цілком, що дорого. Потрібно постійно слідкувати за сервером, оскільки якщо, наприклад, вимкнуть світло у будинку, то користувачі не зможуть скористатися програмним застосунком.
Простий хостинг, переважно система із віртуальними машинами.	Можна налаштувати все під потреби системи. Придбавши певну підписку можна самостійно конфігурувати ресурси системи	Дорожче в експлуатації ніж власний сервер. Уся конфігурація проводиться розробником, оскільки подібні хостинги надають майже повністю порожні системи.
Сучасний хостинг, повноцінні хмарні платформи.	Можна швидко і зручно налаштувати усю хмарну платформу під потреби програмного застосунку. Якщо правильно спроектувати систему, можна зекономити велику суму.	Дорожче ніж експлуатувати власний сервер. Щоб правильно проєктувати систему треба вивчати документацію конкретної хмарної платформи.

З даної таблиці можна зробити висновок, що можна запускати на простих серверах, по типу віртуальних машин (Linode тощо), але кращим рішенням буде запускати систему на хмарних платформах. Це сучасне, гарно спроектоване

рішення, під час використання якого не треба перейматися за роботу серверів та можна зекономити кошти за рахунок вартості обслуговування, якщо правильно спроектувати систему.

Кількість людей, що мають смартфони або інші мобільні пристрої, неухильно зростає. У найближчі роки більшість населення буде майже завжди мати постійний доступ до Інтернету. Тому поєднання хмарних обчислень та мобільних пристроїв, таких як смартфони та планшети, забезпечать прості та зручні способи дистанційного керування системами домашньої автоматизації [50].

Хмарні обчислення – це в основному надання обчислень як послуг, а не обладнання або продуктів, де послуги, спільні ресурси, що доставляються мережею за допомогою різних Інтернет протоколів [51]. Метою переміщення домашньої автоматизації до хмари є збільшення комфорту користувачів під час використання системи. Такий підхід забезпечує безпечний моніторинг та управління різними системами в будинку через Інтернет в режимі реального часу, з будь-якого місця без потреби у великій кількості пам'яті або додаткового обладнання [50].

До функцій, які хмарні платформи здатні надати засобам СОА можна віднести [50]:

- доступність даних у всьому світі: хмарні провайдери зазвичай мають вищу надійність та доступність, ніж приватні веб-сервери. Для того, щоб клієнт мав змогу оцінити свої ризики постачальники можуть мати певні SLA, де вказуються можливості та обмеження послуг;

- зменшення витрат для клієнта: постачальники хмарних послуг виставляють клієнтам рахунки за обчислення часу та пропускну здатності, однак досить часто ці витрати значно менші в порівнянні з потенційними ризиками самостійної роботи системи;

- низький ризик вимкнення серверу та втрати даних: хмарні обчислення широко розповсюджені та дані копіюються у хмару;
- платформи автоматизації підтримують декілька протоколів: більшість хмарних платформ підтримують усі фірмові протоколи та формати для спілкування з пристроями та їх автоматизації, що дозволяє користувачеві не дотримуватися певного постачальника.

В ідеальному варіанті для роботи з хмарним застосунком користувачу необхідно тільки встановити систему автоматизації, а також за потреби зареєструвати її у хмарі. Окрім того, виробник, що випустив пристрій має бути не важливим. Великою перевагою використання хмарних обчислень для домашньої автоматизації полягає в тому, що поки з пристроєм можна зв'язатися через Інтернет, користувачеві не потрібно дбати про те, який конкретний протокол він використовує, а проміжне програмне забезпечення має виконувати обмін даними з пристроєм замість нього та надавати йому інформацію у стандартизованому вигляді. Основні варіанти використання хмарної служби включають [50]:

- реєстрація та обробка даних у хмарі;
- моніторинг та управління пристроями через сервіс;
- автоматичне регулювання на основі певного набору правил, які були визначені та обробляються у хмарній програмі.

На сьогодні є наступні популярні хмарні платформи: Microsoft Azure [52]; Google Cloud Platform [53]; Amazon Web Services [54].

Усі ці хмарні платформи можуть виконувати однакові функції, але документація для AWS описана найгірше з представлених платформ, а функціонал GCP значно поступається Azure. Також значною перевагою саме платформи Azure для розроблюваної системи є те, що Home Assistant має вбудовану інтеграцію засобів обраної хмарної платформи [55].

3.1.2 Порівняння інтерфейсів користувача

Користувач потребує у взаємодії із системою. На сьогодні є наступні основні засоби для такої взаємодії:

- встановлення застосунків на персональні пристрої;
- взаємодія через веб-інтерфейси.

Встановлення застосунків на персональному пристрої потребує розробки для тих самих машин, і, зазвичай це потребує окремих розробок під кожний тип пристрою, тобто настільний ПК або смартфон. З іншої сторони, розробивши веб-інтерфейс, не буде потреби у розробці під окремий тип пристрою, не треба завантажувати програмний засіб на локальну машину. Все, що треба для взаємодії із подібною системою – це лише веб-браузер та інтернет. Саме тому є сенс розробити саме веб-інтерфейс.

Оскільки було обрано хмарну платформу Azure, тому є сенс розглянути у порівнянні ресурс даної платформи для роботи із даними – PowerBI. Порівняння користувацького веб-інтерфейсу та PowerBI наведено в таблиці 3.2 [56]-[58].

Проаналізувавши таблицю 3.2, можна рекомендувати користувацький веб-інтерфейс.

Таблиця 3.2 – Порівняння PowerBI та власного користувацького інтерфейсу

Назва	Користувацький веб-інтерфейс	PowerBI
1	2	3
Важкість розробки	Потребує знання мови програмування JavaScript або TypeScript та знання HTML та CSS.	Потребує знання тільки мови програмування Power Query, що була розроблена специфічно для платформи Azure.
Витрати на хмарній платформі	10 центів за годину	1 долар за годину

Продовження таблиці 3.2

1	2	3
Гнучкість системи	Дозволяє легко взаємодіяти із сторонніми програмними інтерфейсами, але для того, щоб отримувати дані з БД, потребує розробки додаткового сервісу, що буде отримувати та агрегувати дані з БД та надсилати на клієнтську частину.	Є невеликі складнощі із використання сторонніх програмних інтерфейсів, але є нативна можливість звертатися до сервісів у середині Azure, є можливість отримувати та агрегувати дані напряму з БД або з певних вибірок даних та перетворювати їх у графіки.

Він швидкий у розробці, не потребує додаткових сервісів для агрегування даних, відсутня складність у налаштуванні взаємодії із сторонніми програмними інтерфейсами. Він коштуватиме дешевше на дистанції, ніж PowerBI.

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

3.2 Проєктування архітектури системи

Для взаємодії користувача із системою було обрано розробку власного користувацького інтерфейсу. Функціональна схема системи наведена на рисунку 3.2.

Оскільки система повинна бути наділена функціоналом для прогнозування даних енергоспоживання, тому є сенс розробити модуль машинного навчання. Даний модуль потребує даних для навчання, для цього рекомендується створювати певні вибірки даних, щоб бути певними, що дані не було замінено, та що система не буде перенавантажувати БД. Тому є сенс розробити окремий скрипт, що підготовлює необхідні вибірки

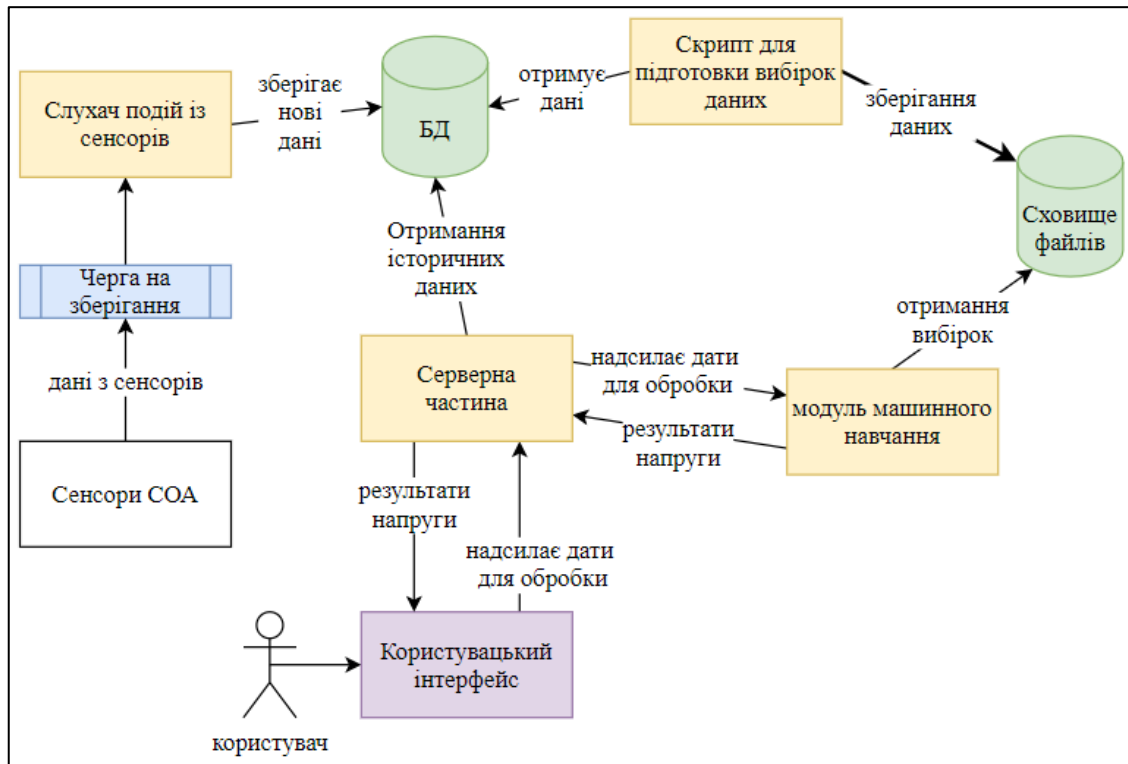


Рисунок 3.2 –Функціональна схема системи

Система потребує у серверній частині, оскільки система повинна надавати повну аналітику по енергоспоживанню, а це включає і історичні дані. Для зручного отримання та агрегування даних користувацький інтерфейс потребує серверну частину, де буде реалізована дана логіка. Система має потребу у черзі, щоб зберігати записи у БД послідовно та без перебоїв. Це дозволить додавати нові датчики у систему, не переймаючись через перенавантаження БД та не хвилюючись, що у непередбачуваних ситуаціях дані загубляться.

Система повинна правильно зберігати дані в БД та правильно керувати взаємодією даних у черзі, а саме видалення їх з черги тільки у випадку, коли дані було успішно записано в БД.

Для коректної та стабільної роботи необхідно спроектувати систему за принципом «Розділяй та керуй». Тобто необхідно розбити функціонал на окремі компоненти, які мають бути згруповані за певною логікою [59]. Це є корисним у випадках, якщо необхідно змінити чи оновити тільки частину коду та не вносить

при цьому зміни до інших частини, що їх не потребують, оскільки уся логіка у частинах інкапсульована. Спираючись на цей принцип було спроектовано діаграму компонентів системи, яку зображено на рисунку 3.3.

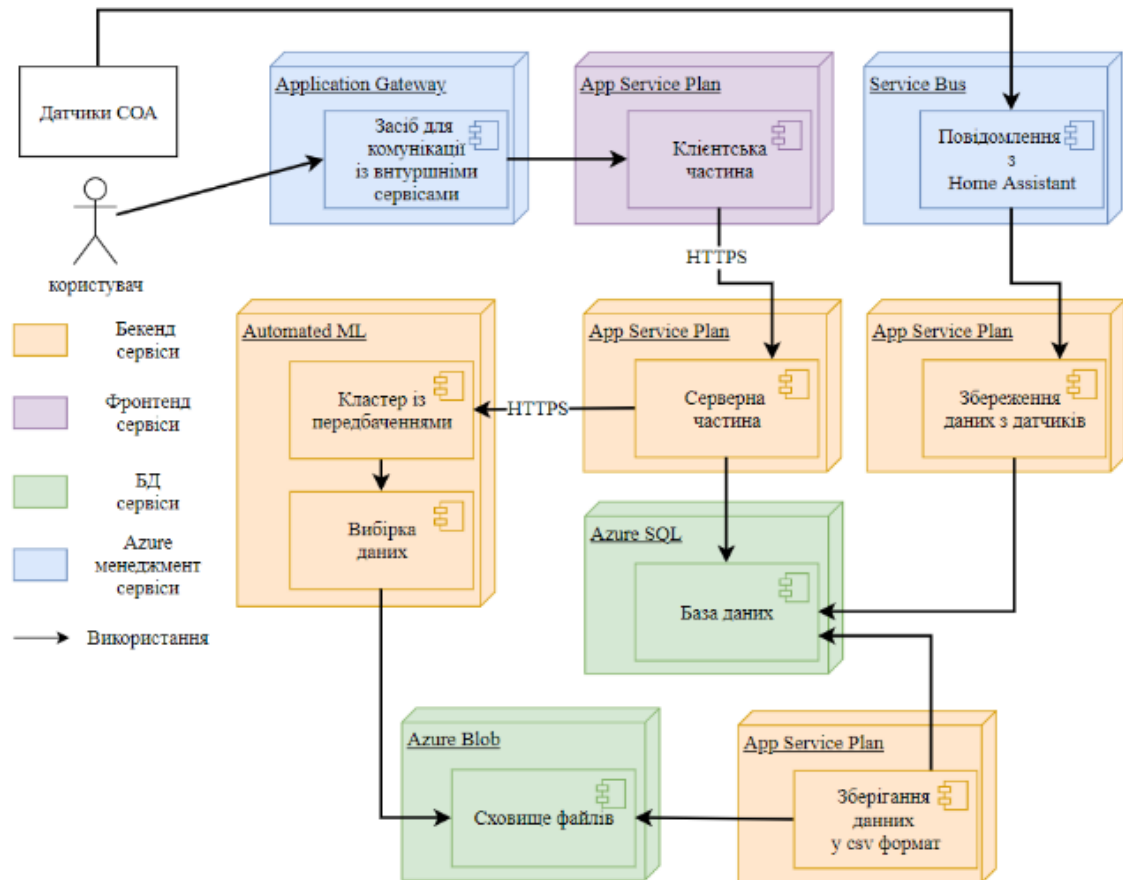


Рисунок 3.3 –Архітектура системи

Оскільки було обрано хмарну платформу Azure, тому було розроблено архітектуру системи із використанням засобів та технологій обраної платформи. Опис компонентів наведено в таблиці 3.3.

Було обрано Azure ML тому що це дозволить швидко та без встановлення додаткових бібліотек налаштувати модуль машинного навчання. Також в Automated ML присутній функціонал для конфігурації системи використовуючи графічний інтерфейс шляхом «перетягання» необхідних блоків програм на загальну схему.

Таблиця 3.3 – Опис компонентів системи

Ресурс	Компонент	Назва компоненту
Application Gateway		Засіб для комунікації із внутрішніми сервісами
App Service Plan	Azure WebApp	Клієнтська частина
	Azure WebApp	Серверна частина
	Azure Function	Збереження даних з датчиків
	Azure Function	Агрегація та збереження даних у csv форматі
Azure Machine Learning	Compute Cluster	Кластер із прогнозуваннями
	Dataset	Вибірка даних
	Pipeline	Процес навчання алгоритму та аналізу даних
Service Bus	Topic	Повідомлення з Home Assistant
Azure Blob	Container	Сховище файлів, що використовуватимуться в системі
Azure SQL		База даних

В результаті прийнятих рішень щодо розробки архітектури – вдалось спроектувати систему, що може витримувати високе навантаження, її легко модифікувати та інтегрувати в неї новий функціонал. Дана система завжди надаватиме актуальну інформацію про дані енергоспоживання у СОА.

3.3 Розробка прототипу користувацького інтерфейсу

Клієнтська частина – це все, що виконується та відображається на екрані будь-якого пристрою користувача, розробка якої полягає в інтуїтивно зрозумілому розміщенні необхідних елементів на сторінці та забезпеченні виконання ними всіх функціональних властивостей. Процедура розробки даної частини системи для ІА даних енергоспоживання складається з трьох основних етапів: створення прототипу зовнішнього вигляду сторінок, його верстання та програмування відповідно до розробленого прототипу.

Прототип сайту – базовий макет, що наглядно відображає розташування основних елементів на сторінці та їх функціонал. Прототип дозволяє заздалегідь визначити та узгодити всі нюанси між замовником та розробником, щодо кінцевого результату. Його розробка зазвичай відбувається на початкових етапах створення проєкту, що забезпечує економію часу, оскільки дозволяє частково або повністю виключити ситуації з переробки проєкту. Окрім того, прототип дозволяє більш точно оцінити розміри та терміни розробки проєкту, визначити його бюджет [60], [61].

Розробка прототипу графічного інтерфейсу користувача розроблюваної системи складається з трьох основних етапів. На першому етапі необхідно визначити кількість сторінок в застосунку та продумати шлях, яким користувач буде переходити між ними. На другому етапі необхідно розробити модульні сітки. Вони дозволяють візуально розділити сторінки на зони та структурувати зміст сайту. На третьому етапі визначаються основні компоненти їх функціональні можливості та розташування та модульній сітці.

Розроблювана системи скрадатиметься з однієї сторінки, модульна сітка якої наведена на рисунку 3.4. Для наочності області на модульній сітці зафарбовані кожна в свій колір, які повторюються на рисунку 3.5, де представлено розміщення компонентів на сторінці.

В таблиці 3.4 наведено опис та розташування на модульній сітці основних компонентів та функції.

Таблиця 3.4 – Опис основних компонентів

Назва	Функція
Логотип	Відображає назву сервісу.
Кнопка «Дата початку»	Надає можливість вибрати дату початку пошуку інтервалу пошуку.
Кнопка «Дата закінчення»	Надає можливість вибрати дату закінчення інтервалу пошуку.
Кнопка початку пошуку	Призначене для початку пошуку.



Рисунок 3.4 – Модульна сітка сторінки

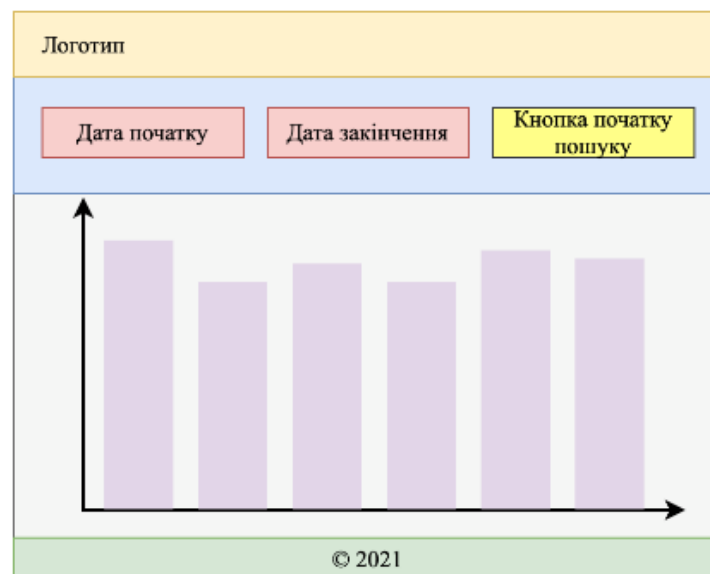


Рисунок 3.5 – Розміщення елементів на сторінці

В системі повинен бути наявний функціонал для перегляду повних деталей енергоспоживання за конкретний день. Для цього було розроблено прототип модального вікна, що наведено на рисунку 3.6.

Деталі за день	
Напруга:	123 V
Сила струму:	123 A
Частота:	123 hz
Споживання:	123 kW/h
Ємність струму:	123 pf
Потужність:	123 W
Закрити	

Рисунок 3.6 – Деталі електроспоживання за конкретний день

Створення прототипу допомагає визначити необхідну кількість сторінок у застосунку, структурувати їх зміст, визначити основні компоненти та їх розміщення на модульній сітці, а також функції, які вони повинні виконувати. Таким чином прототип є скелетом майбутньої системи, що дозволяє на ранніх етапах візуалізувати структуру інтерфейсу користувача та узгодити функції кожного з елементів. Такий підхід дозволяє оцінити правильність стратегії розробки застосунку, а у разі потреби можна внести правки в розроблюваний інтерфейс з мінімальними зусиллями та витратами ресурсів.

3.4 Проєктування схеми БД

Для розроблюваної системи є необхідність зберігати дані енергоспоживання для подальшого використання. На рисунку 3.7 зображено концептуальну модель бази даних, яка складається з двох сутностей: користувач та експеримент. Сутностями називають множину понять реальних або абстрактних об'єктів реального світу, процесу або явищ. Атрибутами зветься інформаційне відображення властивостей об'єктів.

На рисунку 3.7 блакитним позначено назву сутності, помаранчевим унікальний ідентифікатор сутності.

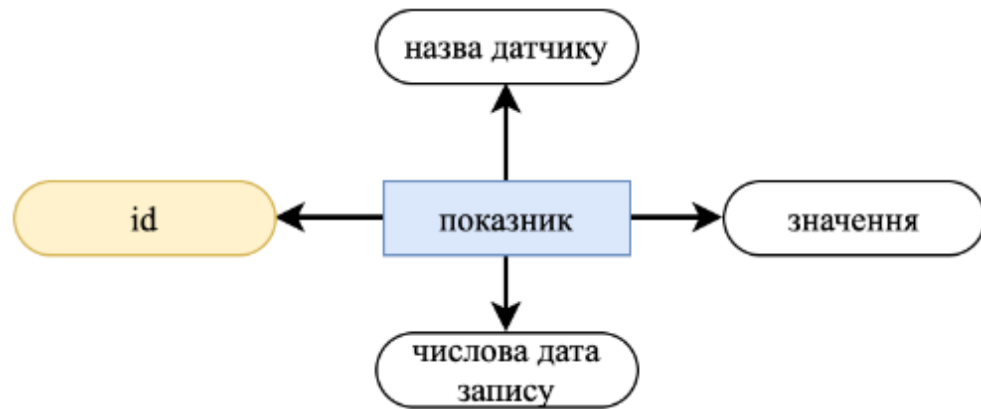


Рисунок 3.7 – Концептуальна модель БД

Опис атрибутів і сутностей даної системи зображено в таблиці 3.5.

Таблиця 3.5 – Опис сутностей і атрибутів

Сутність	Атрибут	Опис сутності
Показник	id	Містить інформацію про значення з сенсорів та час отримання цієї інформації
	Числова дата запису	
	Назва приладу	
	Значення сенсорів	

Атрибут id відповідає за персональний ключ для кожної сутності, що автоматично генерується СКБД. Числова дата запису відповідає за дату, коли було записано показники із сенсорів, також проте є потреба у числовій також, тому що із числовими показниками буде швидше працювати будь-який ІА алгоритм, буде швидше відбуватись отримання даних з БД. Дата оновлення відповідає за дату оновлення даних показника в розроблюваному продукту.

Спроектowana БД надає можливість правильно зберігати дані та швидко та зручно їх витягувати та оброблювати.

3.5 Висновки до розділу 3

В розділі було визначено основні функціональні та нефункціональні вимоги до системи, її вхідні та вихідні дані. Такий аналіз дозволив визначити масштаби та складність функціоналу розроблюваної системи.

Було прийнято рішення запустити розроблювану систему на хмарній платформі Microsoft Azure, оскільки дана платформа має інтеграцію із системою Home Assistant, щоб мінімізувати місячні фінансові витрати на обслуговування системи та зберігання даних.

Було проведено аналіз користувацьких інтерфейсів для взаємодії із системою. Згідно з результатів є сенс розробити власну клієнтську частину, а не PowerBI. Так як PowerBI коштуватиме дорожче на дистанції, ніж користувацький інтерфейс, та потребує створення повноцінної організації в мережі Azure, що коштуватиме зайвих грошей для MVP.

Було спроектовано архітектуру системи. Вона складається з десяти компонентів, які поділяються на групи за основними компонентами, а саме за клієнтською частиною, серверною частиною та модулем машинного навчання.

Спроектовано структуру СКБД для зберігання інформації, необхідної для роботи. Дане рішення дозволить легко та зручно взаємодіяти із даними та підтримувати їх актуальність.

Було спроектовано базовий макет системи. Дане рішення надасть можливість візуалізувати потенційну структуру інтерфейсу системи, показати розташування на сторінці основних елементів, а також описати їх функції.

4 ВИБІР ІНСТРУМЕНТАРІЮ РОЗРОБКИ

4.1 Порівняння та вибір засобів розгортання продукту в хмарній платформі

Для розгортання та налаштування інфраструктури системи необхідно розглянути способи розгортання та налаштування інфраструктури, що наведено на рисунку 4.1.

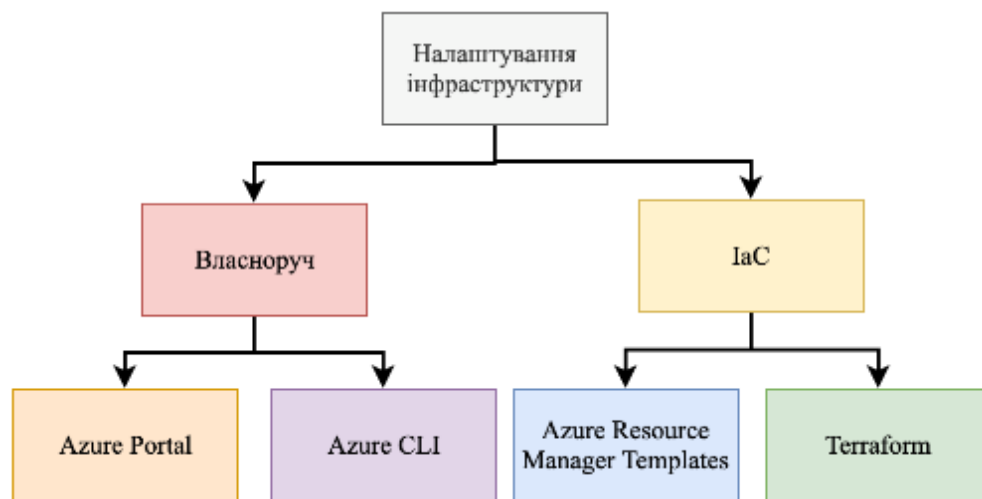


Рисунок 4.1 – Наявні способи розгортання та налаштування інфраструктури

Дану інфраструктуру є можливість розроблювати по черзі створюючи кожний із ресурсів даної системи. Для цього можна використати графічний інтерфейс самої платформи Azure – Azure Portal. Він надасть можливість обрати та створити систему із великою кількістю підказок та зручного пошуку необхідного.

Наявна можливість використати консольний інтерфейс – Azure CLI. Це значно пришвидшить налаштування системи, оскільки все відбуватиметься у власному терміналі. Не буде необхідності переключати різні «віконця» та чекати коли вони відкриються.

Проте наявний функціонал для автоматизації даного процесу, що в майбутньому полегшить задачу для контролю ресурсів, створення параметрів конфігурації, створення додаткових інфраструктур (оточень) тощо. Даний функціонал реалізується завдяки процесу автоматизації, що зветься інфраструктура як код (Infrastructure as a Code – IaC).

IaC – це процес керування та надання комп'ютерних центрів обробки даних за допомогою машиночитаних файлів визначення, а не фізичної конфігурації обладнання чи інтерактивних інструментів конфігурації. Вона надає можливість керування вашою IT-інфраструктурою за допомогою файлів конфігурації [62].

Для хмарної платформи Azure є можливість використати функціонал даної платформи – Resource Manger Template, або вже існуючий та доволі поширений функціонал Terraform.

Azure Resource Manager Templates має значно більше обмежень у порівнянні із Terraform. До того ж в Terraform присутній функціонал для використання системи Terraform Enterprise, що дозволить безпечно зберігати файли із конфігураціями на сервері, та процес налаштування інфраструктури стає ще зручнішим та швидшим. Також у Terraform дуже велика популярність у суспільства та підтримка розробників даної системи. Тому є сенс обрати дану IaC для розроблюваної системи.

4.2 Аналіз та вибір СКБД

Бази даних використовуються для структурованого зберігання та швидкого доступу до даних. Окрім самих даних кожна база даних повинна мати певну модель роботи, на основі якої виконується обробка даних. Для керування

базами даних розроблені системи керування базами даних або СКБД. Це програмне забезпечення, що взаємодіє з базою даних, програмами та інтерфейсами користувача для отримання та аналізу даних [63].

Існує два типи СКБД: реляційні та нереляційні, які також називаються SQL та NoSQL відповідно, різниця між ними полягає в тому, як здійснюється пошук, розповсюдження та обробка даних.

Реляційні СКБД використовують структуровану мову запитів (Structured Query Language, SQL) для визначення і обробки даних [64]. У таких СКБД дані відображаються у вигляді таблиць з рядками та стовпцями, які мають чітку структурою та залежності [63].

Інтегрована структура та система зберігання даних, бази даних SQL дозволяють розробити добре захищену базу даних не вимагаючи при цьому особливих інженерних зусиль. Реляційні СКБД є гарним вибором для створення та підтримки складних програмних рішень. Однією з основ SQL є їх відповідність вимогам ACID (атомність, узгодженість, ізоляція, довговічність), що дозволяє забезпечити цілісність розроблюваної бази даних. До таких реляційних СКБД можна віднести: MySQL, Oracle, PostgreSQL, MSSQL [63].

Нереляційні СКБД або NoSQL не обмежуються структурою таблиць та є системами для управління базами даних, що вважаються орієнтованими на документи. В таких СКБД неструктуровані дані (фото, відео, статті та інше) зберігаються в одному документі, їх легко можна запитати, проте вони не завжди класифікуються за рядками та стовпцями, як у реляційній базі даних [63].

Нереляційні СКБД дозволяють резервувати різні типи даних разом та масштабувати їх, а також не вимагають підготовки до розгортання, що полегшує швидке оновлення структури даних без затримок. До таких систем відноситься популярна СКБД – MongoDB [63].

У більшості випадків для SQL баз даних притаманне вертикальне масштабування, що означає що для збільшення потужності центральних

процесорів та обсягів ОЗУ або системи зберігання даних, необхідно збільшити навантаження на окремо взятий сервер. А для NoSQL баз даних притаманне горизонтальне масштабування, що говорить про те, що збільшити трафік можна, розподіляючи його або додаючи більше серверів до СКБД [64].

На рисунку 4.2 наведено статистику популярності СКБД, яка базується на методі, який збирає та усереднює наступні дані [65]:

- кількість згадок про систему на веб-сайтах;
- загальний інтерес до системи;
- частота технічних дискусій щодо системи;
- кількість пропозицій про роботу, в яких згадується система;
- кількість профілів у професійних мережах, в яких згадується система;
- актуальність у соціальних мережах.

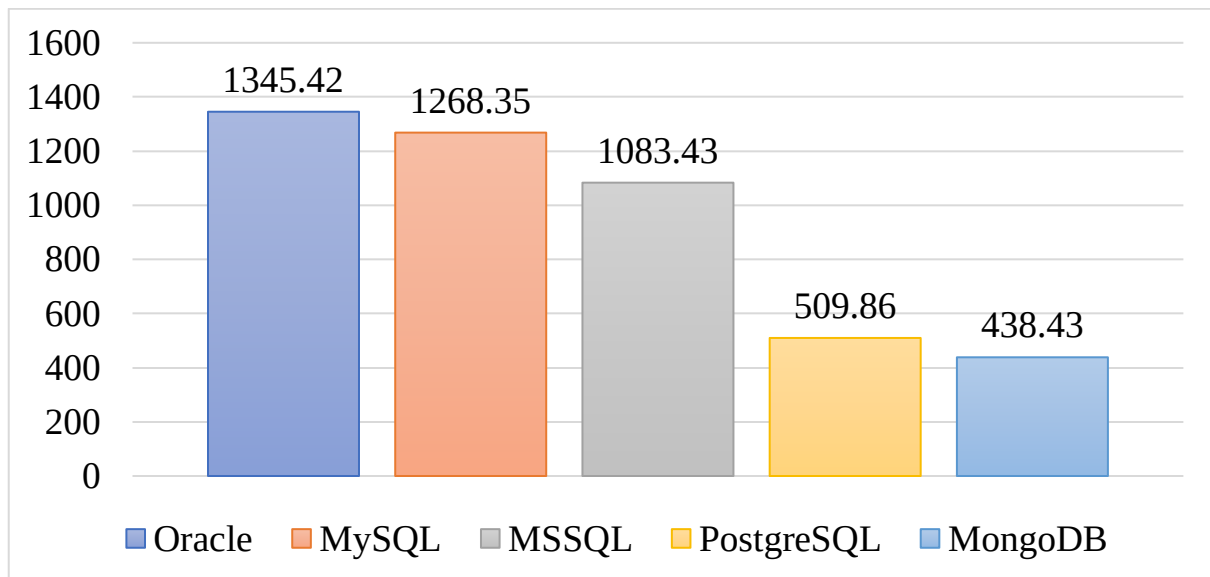


Рисунок 4.2 – Статистика популярності СКБД [65]

В таблиці 4.1 наведено порівняння найпопулярніших СКБД: Oracle, MySQL, MSSQL, PostgreSQL, MongoDB [63], [66], [67].

Таблиця 4.1 – Порівняння найпопулярніших СКБД

Назва СКБД	Тип БД	Структура даних	Ліцензія	Масштабування	Особливості
1	2	3	4	5	6
MySQL	Реляційна	SQL	Безкоштовна	Вертикальний комплекс	Підтримує ACID. Безпека та швидкість. Легкість використання. Підтримка багатьох ОС та Novell Cluster.
Oracle	Реляційна	Багато-модельна, SQL	Платна	Вертикальне	Обробляє великі дані. Підтримує ACID. Розширені багато модельні БД, що підтримують структуровані дані (SQL), напівструктуровані дані (JSON, XML), просторові дані та сховище RDF.
MSSQL Server	Реляційна	T-SQL	Платна	Вертикальний комплекс	Підтримує ACID. Висока продуктивність. Підтримує сценарії на стороні сервера через T-SQL, мови .NET, R, Python та Java. Генерація скриптів для переміщення даних. Підтримка інструментів як локально, так і в хмарі. Також Azure пропонує інструменти для переміщення локального SQL Server у хмару. Багато модельна БД, що підтримує SQL, JSON та просторові дані. Можливість встановити різні версії на одному комп'ютері.
PostgreSQL	Реляційна	Об'єктно-реляційна, SQL	Безкоштовна	Вертикальне	Підтримує ACID. Багатофункціональна багато модельна БД, що підтримує SQL, JSON, XML дані, ключ-значення, просторові дані. Асинхронна реплікація.

Продовження таблиці 4.1

1	2	3	4	5	6
MongoDB	Не-реляційна	NOSQL, орієнтована на документи	Безкоштовна	Горизонтальне	Висока продуктивність. Робота на декількох серверах. Дані зберігаються в формі документів JSON. Можливість індексувати всі поля в документі. Підтримка пошуку за регулярними виразами.

У реляційних СКБД дані представлені в вигляді таблиць, в той час як в нереляційних – у вигляді документів, пар «ключ-значення», графів або wide-column сховищ. В результаті чого SQL бази даних є кращим вибором для застосунків, які передбачають транзакції з декількома записами [64].

Згідно з проведеним аналізом було обрано MSSQL в якості СКБД. Дане рішення обумовлено тим, що дана СКБД може бути налаштована в обраному Azure SQL. Також наявна безкоштовна версія для роботи та обрана СКБД здатна забезпечити потужним функціоналом розроблюваний продукт. На додаток СКБД MSSQL надає надійний захист даних та доступ до них. В даній СКБД наявна можливість швидко оброблювати дані в БД, завдяки засобам оптимізації запитів та структур таблиць, що є головним функціоналом для технологій Big Data.

4.3 Огляд та вибір мови програмування

Існує велика кількість мов програмування, які можуть виконувати майже всі поставлені задачі, та безліч рейтингів, що визначають їх популярність, наприклад сервіс PYPL [68], в якому наявна доступна та наочна статистика. На рисунку 4.3 зображено статистику популярності мов програмування на жовтень 2021 року.

Всесвіт, Жовтень 2021 у порівнянні із попереднім роком:				
Ранг	Зміна	Мова програмування	Частина	Тренд
1		Python	29.66 %	-2.1 %
2		Java	17.18 %	+0.8 %
3		JavaScript	8.81 %	+0.4 %
4		C#	7.3 %	+1.1 %
5	↑	C/C++	6.48 %	+0.7 %
6	↓	PHP	5.92 %	+0.1 %
7		R	4.09 %	+0.2 %
8		Objective-C	2.24 %	-1.2 %
9	↑	TypeScript	1.91 %	+0.1 %

Рисунок 4.3 – Статистика частоти використання мов програмування [68]

За результатом статистики Python [69] є найпопулярнішою мовою програмування, на другому місці Java [70], на третьому місці JavaScript [71]. Тому порівняння саме цих мов програмування є актуальним для даної розроблюваної системи. Дане порівняння наведено в таблиці 4.2

Таблиця 4.2 – Порівняння мов програмування

Мова	Python	Java	JavaScript
Рік першого релізу	1991	1991	1995
Поріг входження в мову	низький	середній	низький
Підтримка суспільства (від 1 до 10)	9	7	9
Підтримка розробників (від 1 до 10)	8	7	8
Документація (від 1 до 10)	9	7	8
ООП	модифікований	повноцінний	мінімальний
Стандарти написання коду	PEP	-	ECMA

Для розроблюваної серверної частини мова програмування повинна:

- взаємодіяти зі сторонніми програмними інтерфейсами (API);
- бути зрозумілою та читабельною для підтримки системи;
- мати можливість розроблювати серверні застосунки;
- мати якісну документацію та підтримку суспільства та розробників

для пришвидшення розробки.

Тож Python є найкращою мовою для розробки серверної частини, оскільки мова має дуже гнучку та зрозумілу структуру завдяки стандартам написання коду PEP. Він має велику базу бібліотек для розробки серверних застосунків. Дана мова програмування має бібліотеки для взаємодії із усіма сучасними базами даних та надає можливість взаємодіяти із API сторонніх веб-застосунків та має одне з найбільших суспільств на жовтень 2021 року, завдяки чому є можливість швидко знайти необхідну інформацію.

Для розроблюваної клієнтської частини мова програмування повинна:

- взаємодіяти із сторонніми API;
- мати якісну документацію та підтримку суспільства;
- мати зрозумілий та читабельний код;
- мати можливість розроблювати клієнтську частину.

JavaScript є однією з найпопулярніших мов програмування, яка є невід’ємною частиною веб-застосунків та веб-сторінок. Це мова сценаріїв, що допомагає створити динамічні та інтерактивні веб-сторінки. JavaScript є мовою високого рівня з компілятором JIT (Just-in-Time) і динамічним типом. Довгий час JavaScript був реалізацією тільки на стороні клієнта, що дозволяло йому працювати у веб-браузері користувача без потреби в будь-яких ресурсах веб-сервера, але на сьогодні завдяки новим двигунам мова має також реалізації на стороні сервера [72]-[73].

JavaScript є гнучкою, динамічною та кросплатформною мовою, тому її можна рекомендувати для розробки клієнтської частини [72].

4.4 Аналіз та вибір середовища розробки

Існує багато популярних засобів розробки програмних застосунків. Кожен із них наділений величезним функціоналом, який допомагає швидко та якісно розроблювати програмні продукти. Найпопулярніші з них наведено в таблиці 4.3 [74] - [76].

Таблиця 4.3 – Порівняльний аналіз популярних середовищ розробки

Назва	IntelliJ IDEA	Visual Studio Code	Vim
1	2	3	4
Наявність плагінів	Багато плагінів, але частина з них платна.	Багато безкоштовних плагінів.	Велика кількість, не дивлячись, що це консольний інтерфейс.
Швидкість налаштування	Треба встановити, обрати тему, базові плагіни та розпочати роботу.	Треба встановити та розпочати роботу.	Треба встановити та розпочати роботу.
Інтуїтивність інтерфейсу	Дружній інтерфейс, проте деякі функції необхідно шукати в налаштуваннях.	Дружній інтерфейс, все знаходиться в швидкому та зручному доступі.	Інтерфейс складний для новачків, потрібен час, щоб освоїти взаємодію із системою.
Контроль версій	Є частиною стандартного функціоналу	Наявний після встановлення спеціального розширення.	Наявний, оскільки є швидкий доступ до командного рядка.
Перевірка синтаксису	Є частиною стандартного функціоналу	Наявна після встановлення спеціального розширення.	Наявна після встановлення спеціального розширення
Кастомізація інтерфейсу	Наявна, є вбудовані теми оформлення, та є можливість їх налаштування.	Наявна, є вбудовані теми оформлення, та є можливість встановити додаткові.	Наявна після встановлення спеціальних пакетів
Гарячі клавіші	Наявні.	Наявні.	Наявні. В даному середовищі розробці їх найбільше число.

Продовження таблиці 4.3

1	2	3	4
Авто- підстановка коду	Наявна.	Наявна після встановлення додаткового плагіну.	Наявна із використанням спеціальних пакетів.
Швидкість запуску	Довгий запуск, спочатку йде завантаження функціоналу середовища розробки, потім аналіз проєкту і тільки після цього надається доступ для модифікації коду, та у фоні оновлює список встановлених пакетів.	Швидкий запуск, бо увесь аналіз проєкту, що був завантажений відбувається після надання можливості модифікації коду.	Найшвидший запуск системи, оскільки увесь інтерфейс вбудований у командний рядок.

Проаналізувавши таблицю 4.3, можна зробити висновок, що Visual Studio Code є найкращим рішенням для розроблюваної системи, оскільки в ній функціонал залежить від спеціальних пакетів розширень, яких в системі величезна кількість. Це надає даному середовищу розробки гнучкість під будь-які поставлені задачі.

4.5 Порівняння та вибір фреймворків та технологій для розробки

Використання фреймворків під час створення будь-якого проєкту забезпечує розробників великою кількістю готових інструментів, що дозволяють полегшити та пришвидшити роботу. Вони пропонують початкову структуру для майбутнього проєкту, а також забезпечують автоматизацію реалізації загальних рішень тим самим скорочуючи час розробки та надає можливість розробникам зосередитись на створенні логіки проєкту, а не на рутинних елементах [77].

4.4.1 Вибір фреймворку або бібліотеки для серверної частини

Оскільки користувач потребує отримувати дані енергоспоживання як історичні, так і дані прогнозування. Для отримання даних прогнозування серверна частина потребує у взаємодії із модулем машинного навчання. Тому дана система являтиме собою серверний застосунок та має надавати API, для подальшої взаємодії клієнтської частини із ним. В Python наявна можливість розроблювати серверні застосунки завдяки вбудованим фреймворкам WSGI та ASGI.

WSGI (Web Service Graphical Interface) це інтерфейс шлюзу веб-сервера. Це специфікація, яка описує, як веб-сервер спілкується з веб-застосунками та як веб-застосунки можуть бути пов'язані разом для обробки одного запиту. WSGI - стандарт Python, докладно описаний у PEP 3333 [78].

ASGI (Asynchronous Server Gateway Interface) – духовний спадкоємець WSGI, призначений забезпечити стандартний інтерфейс між підтримуваними асинхронними веб-серверами, фреймворками та програмами Python.

Там, де WSGI надав стандарт для синхронних застосунків Python, ASGI надає і для асинхронних, і для синхронних застосунків, з реалізацією зворотної сумісності WSGI та декількома серверами та рамками застосунків [79].

Але ці фреймворки низького рівня, тобто більшу частину треба розроблювати з нуля. Існує багато готових бібліотек та функціоналу для реалізації серверних частин на базі цих фреймворків. Порівняння найпопулярніших із них [80]-[82] описано в таблиці 4.4, де підтримка суспільством та розробником оцінюється від 1 до 10.

Оскільки функціонал серверної частини буде доволі простий, тому для цього нема потреби у великому функціоналі, адже частина, що оброблює повідомлення пов'язана із серверною тільки вхідними даними.

Таблиця 4.4 – Порівняння рішень для розробки серверних застосунків

Назва	FastAPI	Flask	Django
Опис	Мінімалістичний фреймворк на базі starlette та ASGI.	Мінімалістичний фреймворк на базі WSGI.	Фреймворк високого рівня, на базі WSGI та ASGI в останніх версіях.
Версія	0.70.0	1.1.2	3.0.5
Підтримка суспільства	8	9	9
Підтримка розробника	9	7	8
Головні особливості	Не займає багато місця на диску, має добре прописану документацію. Існує велика кількість розширень, більшість яких описано в документації самої бібліотеки.	Не займає багато місця на диску. Має велике суспільство. Існує велика кількість розширень.	Має найбільший серед існуючий веб фреймворків функціонал. Наявна вбудована панель адміністратора та функціонал для роботи з БД. Має велике суспільство.

Саме тому найкращим рішенням для розроблюваного продукту є FastAPI, оскільки являє собою мінімалістичну бібліотеку та має найкращі показники швидкості та кількості оброблюваних водночас запитів. До того ж дана бібліотека використовує асинхронний код, що значно пришвидшуватиме роботу програми.

4.4.2 Вибір фреймворку або бібліотеки для клієнтської частини

Мова JavaScript має довгий список фреймворків та технологій, дозволяє скоротити час та витрати на розробку, оскільки зазвичай вони надають безліч вбудованих компонентів та функцій керування користувачем, які не потрібно писати з нуля. Також фреймворки та технології забезпечують додаткові заходи

безпеки, мають добре організовану інфраструктуру застосунків, тобто мають певний організований скелет проєкту, а також відокремлюють файли інтерфейсу від бізнес-логіки. Трійкою найпопулярніших технологій мови JavaScript – Vue , Angular.js та React, їх порівняння наведено в таблиці 4.5 [83].

Таблиця 4.5 – Порівняння технологій Vue , Angular та React

Параметри	Angular.js	React	Vue
Тип технологій	Фреймворк	Бібліотека	Бібліотека
Розмір	500+ KB	100 KB	80 KB
Популярність	Широко популярний серед розробників	За рік понад 27 000 запитів	Більше 40 000 запитів на GitHub протягом року
Використання	Для швидкої розробки застосунків, оскільки велика кількість функціоналу вже реалізована, оскільки це фреймворк	Для більш масштабною та цілеспрямованою розробки конкретного функціоналу. Дозволяє проєктувати свою гнучку та зрозумілу файлову архітектуру.	Для більш масштабною та цілеспрямованою розробки конкретного функціоналу. Дозволяє проєктувати свою гнучку та зрозумілу файлову архітектуру.

Angular – це повноцінний фреймворк, який пропонує всі необхідні інструменти від розробки до тестування під час створення веб-застосунків, тоді як React – це гнучка бібліотека, яка потребує встановлення інших бібліотек для розробки. Vue – це наймолодша бібліотека, що поєднує у собі Angular та React, який додає швидкості, ефективності та простоти в розробці веб-застосунків [84].

Велика перевага Vue та React – це те, що вони не містять великої кількості функціоналу як Angular, тому що у 85% відсотках випадків він просто не буде використовуватись.

Також Vue надає користувачу детальну документацію, що допомагає розробникам швидко освоїти технології, або знайти відповіді. Створені застосунки легко інтегрується в існуючу інфраструктуру. Завдяки невеликим розмірам, його швидкість та гнучкість є оптимальними, що дозволяє покращити продуктивність в порівнянні з іншими технологіями [85].

На основі аналізу було обрано бібліотеку Vue, яка є оптимальним рішенням для розробки веб-застосунків, оскільки достатньо знань HTML та JavaScript для розробки застосунків.

4.6 Висновки до розділу 4

В даному розділі було проведено порівняння та аналіз необхідного інструментарію для розробки системи.

В якості IaC системи було обрано Terraform, оскільки Azure Resource Manager Templates має значно більше обмежень у порівнянні із Terraform. До того ж Terraform наділен можливістю використати систему Terraform Enterprise, що дозволить безпечно зберігати файли із конфігураціями на сервері, та процес налаштування інфраструктури стає ще зручнішим та швидшим. Також у Terraform дуже велика популярність у суспільства та підтримка розробників даної системи. Тому є сенс обрати дану IaC для розроблюваної системи.

Згідно з проведеним аналізом було обрано MSSQL в якості СКБД. Дане рішення обумовлено тим, що дана СКБД може бути налаштована в обраному Azure SQL, надає потужний функціонал для розроблюваної системи та може надати необхідну безпеку даних.

На основі аналізу Visual Studio Code було обрано в якості інтегрованого середовища для розробки. Дане рішення обумовлено зручним та інтуїтивно зрозумілим інтерфейсом та здатністю надати розробнику базовий функціонал

розробки та можливість його розширення за рахунок встановлення додаткових плагінів.

Проаналізовано мови програмування та для розробки серверної частини обрано Python, оскільки він має гнучку та зрозумілу структуру завдяки стандартам написання коду PEP. Дана мова програмування має велику базу бібліотек для розробки серверних програмних застосунків та бібліотеки для взаємодії із усіма сучасними базами даних, має одне з найбільших суспільств на жовтень 2021 року, завдяки чому з'являється можливість швидко знайти відповіді на поставлені питання. В якості бібліотеки для серверної частини обрано FastAPI. Таке рішення обумовлено тим, що для розроблюваного продукту є FastAPI, оскільки являє собою мінімалістичну бібліотеку та має найкращі показники швидкості та кількості оброблюваних водночас запитів. До того ж дана бібліотека використовує асинхронний код, що значно пришвидшуватиме роботу програми.

На основі аналізу було обрано бібліотеку Vue, яка є оптимальним рішенням для розробки веб-застосунків, оскільки достатньо знань HTML та JavaScript для розробки застосунків.

5 КОНСТРУЮВАННЯ СИСТЕМИ

Розроблювана система складається з чотирьох основних частин: клієнтської та серверної частин, налаштувань інфраструктури та функцій. Реалізація обмежена єдиним репозиторій, що міститиме код проєкту, щоб пришвидшити розробку. Тому було створено чотири директорії, як зображено на рисунку 5.1, відповідно до кожної із частин.

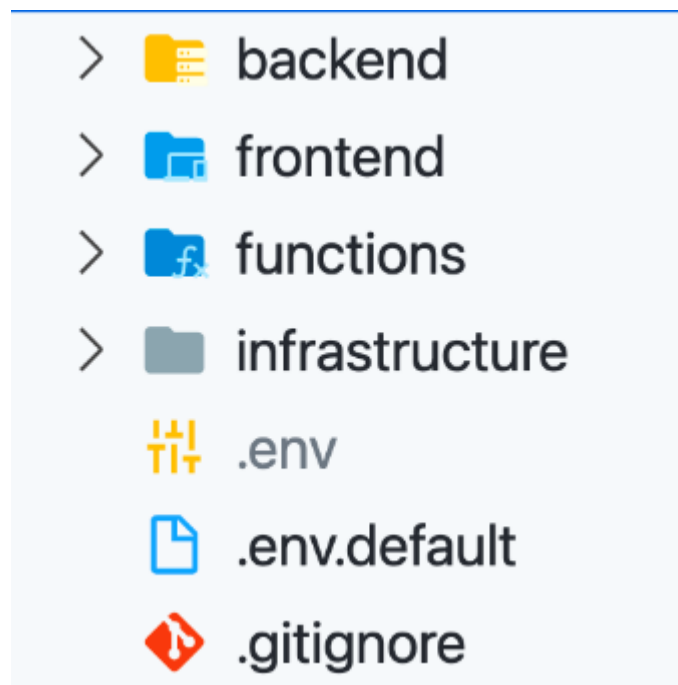


Рисунок 5.1 – Структура файлової системи

Базова файлова структура також включає в собі файл для конфігурацій проєкту «.env» та приклад, що зберігається в репозиторії – «.env.default». Також було розроблено файл «.gitignore», щоб не зберігати у репозиторії непотрібну, або приватну інформацію.

5.1 Конструювання інфраструктури

Інфраструктура – це основа або структура, яка підтримує систему або організацію. У обчислювальній сфері інфраструктура інформаційних технологій складається з фізичних і віртуальних ресурсів, які підтримують потік, зберігання, обробку та ІА даних. Інфраструктура може бути централізована в центрі обробки даних, або вона може бути децентралізована і розподілена між кількома центрами обробки даних, які контролюються або організацією, або третьою стороною, наприклад, центром розміщення або постачальником хмар [85].

Розроблювана система потребує інфраструктури, оскільки вона є основою. Згідно зі спроектованою архітектурою системи в результаті була розроблена та налаштована інфраструктура, що зображена на рисунку 5.2.

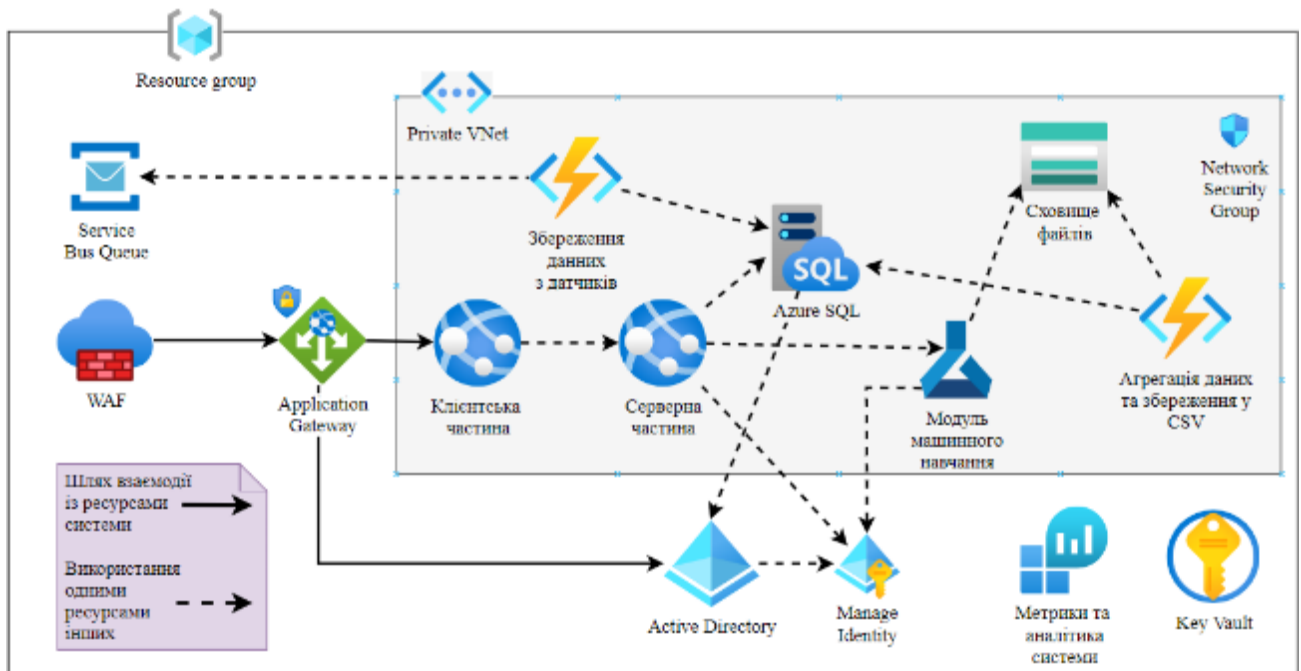


Рисунок 5.2 – Схема інфраструктури системи

На даному рисунку цільними стрілками показано шляхи взаємодії із ресурсами системи, а пунктирними – використання одними ресурсами інших.

Уся приватна інформація зберігається у Azure Key Vault, щоб до неї мали доступ тільки певна група осіб, та тільки необхідні ресурси системи. Завдяки цьому було налаштовано безпечну взаємодію ресурсів, які звертаються наприклад до БД використовуючи ключі, що засекречені в Key Vault.

Також в системі наявні метрики та аналітика, що зберігається в Azure Application Insights. Даний ресурс може відстежувати роботу усіх ресурсів, зберігати їх логування та в майбутньому проводити аналіз даних. Також даний ресурс зберігає логування власноруч створено спеціально для серверної частини.

Уся авторизація користувачів у системі відбувається за рахунок Active Directory. Даний ресурс створює щось на кшталт переадресування усіх запитів через свої системи, та перевіряє авторизовану особу у власній БД. Також через Active Directory налаштовано і Managed Identity, що контролює взаємодію ресурсів у системі.

Ресурси, такі як: серверна частина, клієнтська частина, функції для зберігання та агрегації даних, СКБД та сховище даних знаходяться в спеціально розробленій віртуальній мережі (Azure VNet). Це було зроблено для безпеки ресурсів системи із використанням Network Security Group (NSG).

Було розроблено Web Application Firewall (WAF) та Application Gateway для того щоб вказати які саме ресурси повинні бути доступні «зовні» тобто за публічними адресами, щоб забезпечити безпеку від проникнення в середину таких ресурсів як Azure ML або СКБД.

Усі правила для налаштування ресурсів для Terraform зберігаються у спеціальних файлах із розширенням «*.tf». Для більшої зручності та читабельності, що дасть можливість майбутнім розробникам легше підтримувати систему, слід розробити правильну файлову систему. Розроблена файлова система наведена на рисунку 5.3.

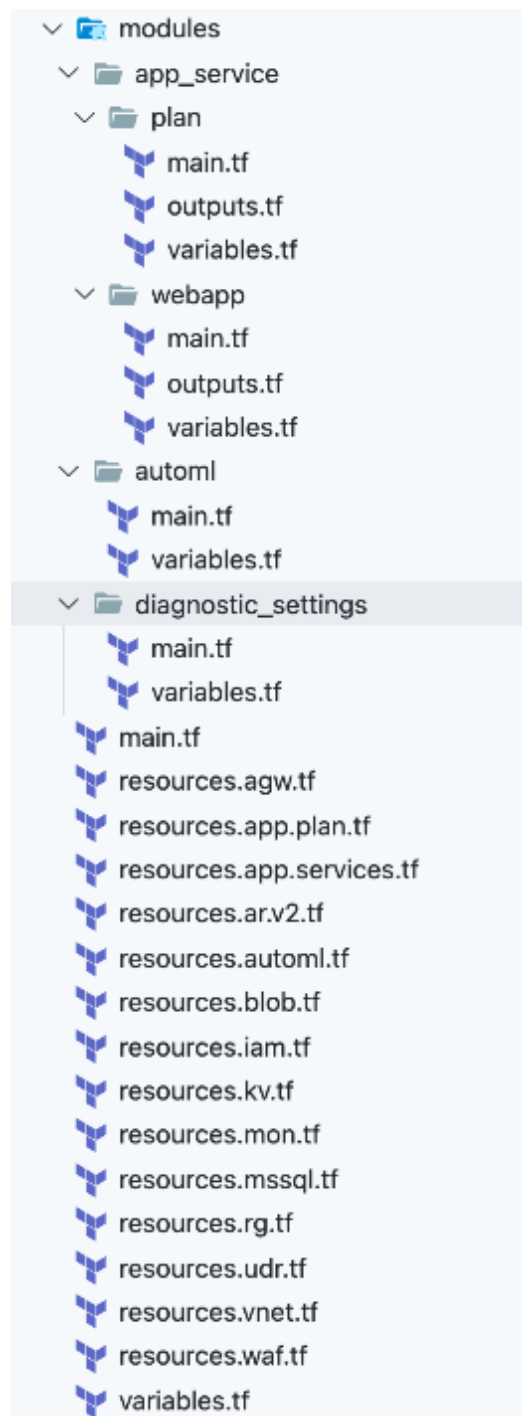


Рисунок 5.3 – Файлова система скриптів та правил для IaC Terraform

На даному рисунку зображено усі модулі системи. «main.tf» – найголовніший модуль, він є точкою, що стягує усі файли. У модулі «variables.tf» зберігаються головні параметри системи. Файли, що містять префікс «resources» містять конфігурації кожного із необхідних ресурсів, які наведено в таблиці 5.1.

Таблиця 5.1 – Опис основних файлів конфігурацій

Назва	Повна назва	Опис
1	2	3
agw	App Gateway	Було прописано правила для налаштування правил маршрутизації для клієнтської та серверної частини, було налаштовано «probes» та публічні ір адреси.
app.plan	App Service Plan	Було прописано плани, що будуть використані для App Service, та їх налаштування, до якої ресурс групи вони відносяться. Також для зручності було винесено частину правил в окремий модуль «app_service/plan»
app.service	WebApp	Було прописано правила налаштування WebApp для клієнтської та серверної частин та функцій. Було налаштовано використання підмережі у основній віртуальній мережі. Було описано правила для авторизації із використанням Azure Active Directory. Також для зручності було винесено частину правил в окремий модуль «app_service/webapp».
ar.v2	App Roles	Було налаштовану просту конфігурацію, щоб дозволити користуватися системою тільки необхідній групі людей.
automl	AutoML	Було прописано конфігурації для створення кластерів та скриптів. Було зроблено налаштування віртуальної мережі та приватних ір адрес. Також для зручності було винесено частину правил в окремий модуль «automl»
iam	Azure IAM	Було сконфігуровано, щоб розрізняти права для користувачів та для розробників, та щоб налаштувати правильну взаємодію між WebApp та AutoML та БД.

Продовження таблиці 5.1

1	2	3
kv	Azure KeyVault	Було прописано усі змінні в системі, усі ключі для приватної інформації, та доступність окремих ключів для окремих ресурсів системи.
mon	Application Insights, Log Analytics	Було налаштовано систему на збирання метрик та показників та зберігання логування самих ресурсів, та створених власноруч логів. Також для зручності було винесено частину правил в окремий модуль «diagnostic_settings»
mssql	Azure SQL (SQL Server)	Було налаштовано СКБД на генерацію випадкових доступів до самої системи та зберігання їх у KeyVault. Було прописано ресурси СКБД та її масштабованість та доступ у приватній мережі.
rg	Resource Group	Було налаштовано ресурс групу, до якої було створено та налаштовано усі прописані в системі ресурси та скрипти.
udr	—	Було прописано усі базові правила для налаштування маршрутизації в середині віртуальної та приватної мережі
vnet	Azure VNet	Було розроблено налаштування для віртуальної мережі системи, «в середині» якої знаходяться усі ресурси системи
waf	Azure Web Application Firewall	Було розроблено налаштування правил захисту системи від зовнішніх загроз.
blob	Azure Blob Storage	Було розроблено правила налаштування сховища файлів, налаштування для VNet та налаштування для створення контейнера, та зберігання його назви у параметрах.

Приклади файлів конфігурації наведено в додатку А.

В результаті отримаємо ресурс групу із налаштованими ресурсами як на рисунку 5.4.

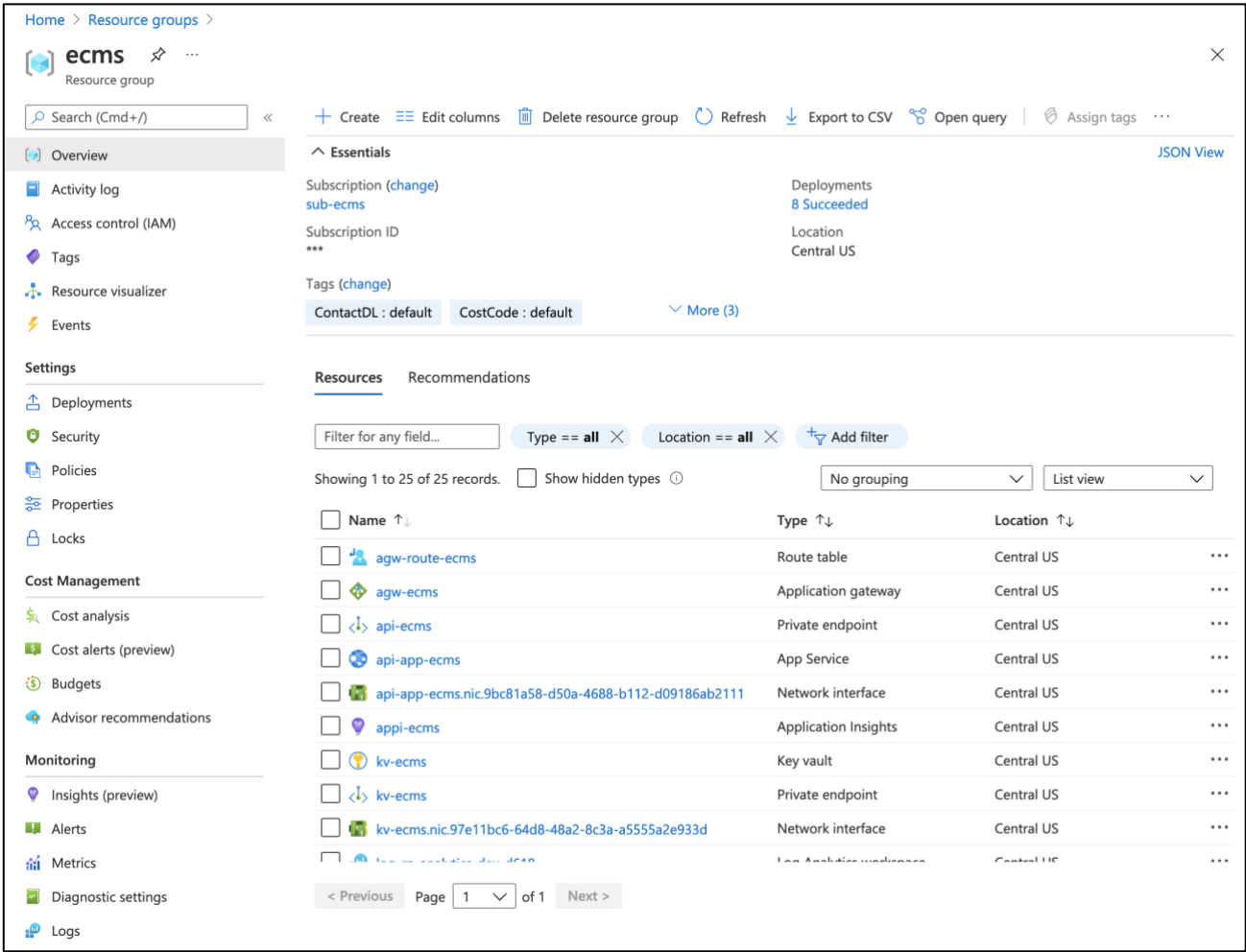


Рисунок 5.4 – Приклад налаштованої ресурс групи і компонентів

В результаті було розроблено повністю налаштовану та готову до експлуатації інфраструктуру. Було розроблено налаштування безпеки, що надає можливість безпечного використання та зберігання даних в системі. Приклади програмного коду наведено в додатку А.

5.2 Конструювання серверної частини

Серверна частина являє собою програмний застосунок, що надає API для взаємодії із ним. Було обрано фреймворк FastAPI, адже він є мінімалістичним та надає можливість швидко оброблювати запити.

Оскільки дана частина написана мовою програмування Python, тому кожна директорія міститиме файл «__init__.py». Цей файл позначає директорію, де він знаходиться, як пакетний модуль. Завдяки цьому з'являється можливість імпортувати файли із даного модулю в інші та використовувати їх за значенням.

Модульна структура серверної частини наведена на рисунку 5.5.

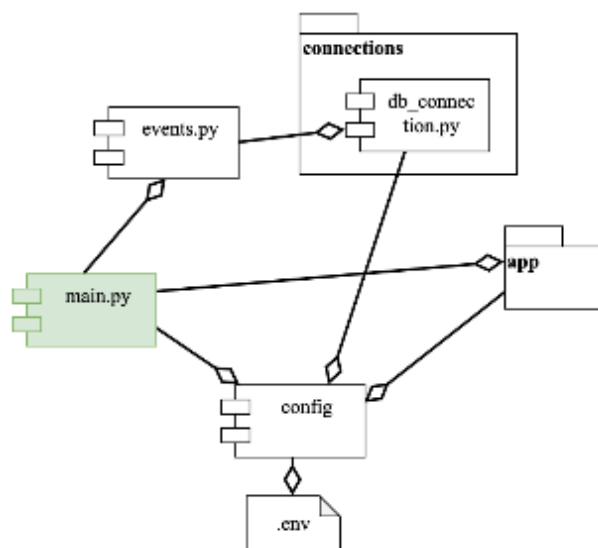


Рисунок 5.5 – Модульна структура системи

В даній системі наявен основний модуль «main.py», який являє собою точкою запуску серверної частини. В даному модулі було розроблено підключення усіх основних модулів системи та зроблено налаштування самої системи.

Модуль «main.py» використовує модуль «config.py», для використання конфігурацій для системи. Даний модуль налаштован таким чином, що може

оброблювати конфігурації, що знаходяться в змінних оточення та у спеціальному файлі, що повинен мати назву «.env». Таким чином в системі наявна можливість для отримання усієї необхідної конфіденційної інформації без ризику збереження її в репозиторії та як наслідок втрата її, або використання хакерами або іншими шахраями.

В даній системі наявен модуль «events.py». Даний модуль містить інформацію про конфігурацію подій, що відбуваються кожного запуску системи. В даному випадку це підключення до БД на запуск системи та її відключення на припинення роботи системи. Також об'єкт підключення до БД було налаштовано в системі із використанням принципів *dependency injection*, що надало можливість зручного використання у самій системі, а саме у кожному з розроблюваних програмних інтерфейсів.

Увесь основний функціонал було розроблено у пакеті `app`. Структуру даного пакету наведено на рисунку 5.6.

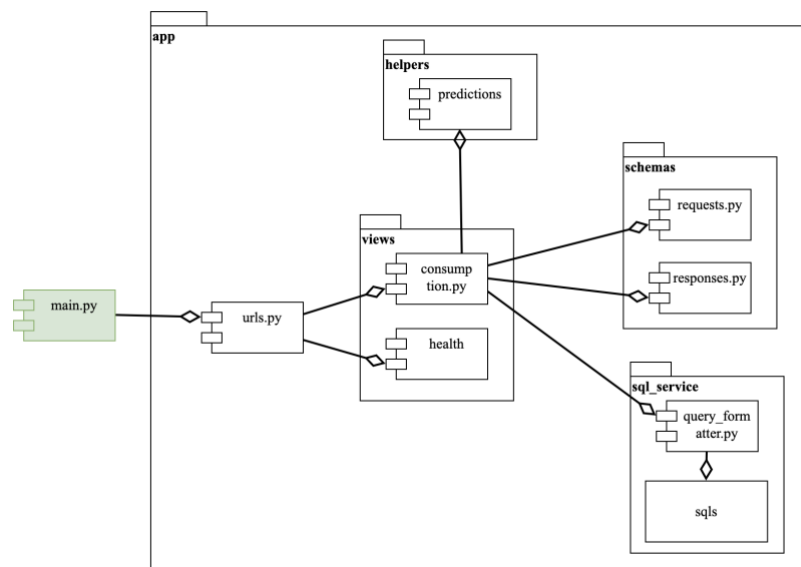


Рисунок 5.6 – Структура пакету «app»

В даному пакеті було розроблено основний модуль `urls.py`. Даний модуль було розроблено із використанням дизайн шаблону програмування – Фасад.

Даний модуль імпортує спеціалізовані маршрутизатори із пакету «views». Завдяки цьому для конфігурування програмних інтерфейсів розроблюваної системи треба імпортувати лише один об'єкт у модулі `urls.py`.

Пакет «views» містить у собі функціонал програмних інтерфейсів, а саме:

- «consumption.py» – даний модуль містить функціонал програмного інтерфейсу для обробки даних енергоспоживання та надсилання їх до клієнтської частини;

- «health.py» – даний модуль зберігає у собі функціонал програмного інтерфейсу для перевірки доступності серверу на хмарній платформі.

Модуль «consumption.py» використовує пакет «schemas» для опису та затвердження структури вхідних запитів та структур відповідей на них. Модулі даного пакету зберігають спеціальні класи-валідатори, що перевіряють та інформують користувача про помилки, або допомагають привести відповідь до очікуваної структури.

Було розроблено модуль «sql_service» щоб зберігати та формувати запити до Azure SQL. Було використано гібридну форму взаємодії серверної частини із БД, а саме використання чистого SQL-синтаксису із використання бібліотеки ORM для створення підключення та SQL-ін'єкції. Було розроблено таким чином, тому що агрегація даних виходила доволі складною, що стало причиною використання саме такого, гібридного підходу для взаємодії із БД. Приклад SQL запиту наведено в додатку А.

Пакет «helpers» містить у собі функціонал для взаємодії із програмним інтерфейсом сервісу Azure Machine Learning (Azure ML). Він працює за принципом примітивного спілкування використовуючи TCP/IP протоколи. Оскільки Azure ML надає спеціальний програмний інтерфейс для взаємодії із ним, тому отримання даних прогнозувань енергопостачання відбуватиметься шляхом надсилання запитів із використанням TCP протоколів.

Дану логіку було розроблено саме на серверній частині, щоб клієнтська частина мала лише одне джерело даних. До того ж було розроблено на серверній частині логіку по агрегації та комбінуванню даних історичних та даних прогнозування.

Алгоритм отримання даних енергоспоживання за обраний період дат наведено на рисунку 5.7.

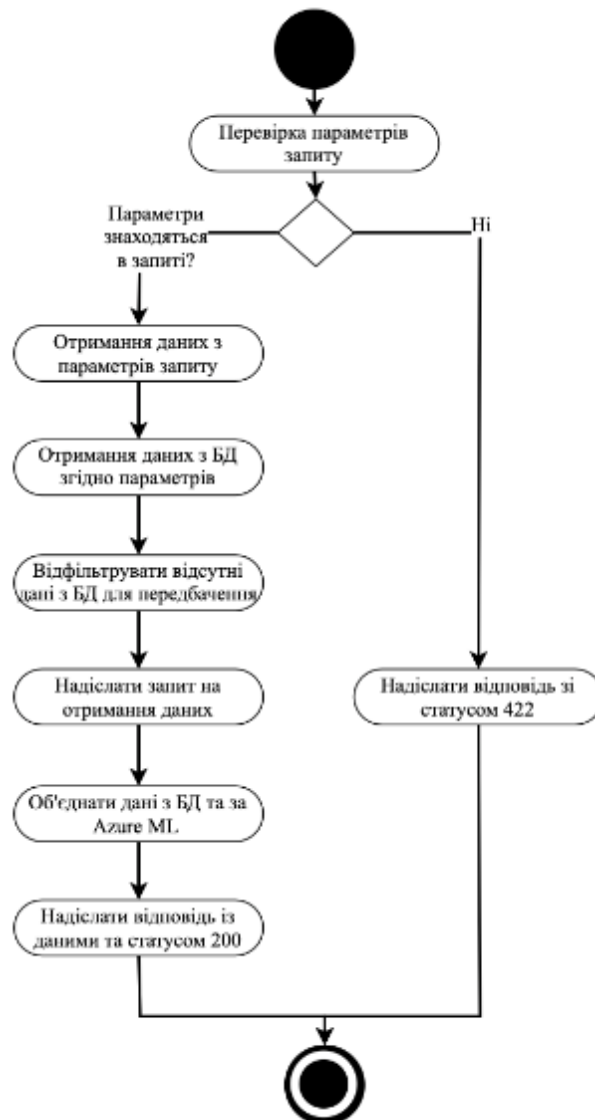


Рисунок 5.7 – Діаграма діяльності отримання даних енергоспоживання

Для отримання даних енергоспоживання треба надіслати до серверної частини запит, що міститиме дату початку інтервалу та дату закінчення інтервалу, за який користувач хоче отримати дані енергоспоживання.

Далі відбувається отримання історичних даних з БД. Наступний крок це ІА даних, що надійшли з БД, а саме отримання дат, на які не було збережено даних енергоспоживання, щоб отримати для них прогнозування.

Після отримання даних прогнозування відбувається процес поєднання історичних та даних прогнозування, та їх сортування за датами.

Після сортування список даних енергоспоживання надсилається у відповіді зі статусом кодом 200 (OK).

Якщо параметри у запиті були відсутні то надсилається повідомлення до користувача зі статусом 422 та інформацією яких даних не вистачає для запиту для отримання даних енергоспоживання.

Таким чином було розроблено та налаштовано серверну частину, що надало можливість для коректної взаємодії із клієнтською частиною, ресурсом Azure Machine Learning. Також був розроблений функціонал для хмарної платформи для перевірки доступності серверної частини для взаємодії із нею. Приклади коду системи наведено в додатку А.

5.3 Конструювання скрипту для зберігання показників з сенсорів COA

Розроблювана система потребує у функціоналі (скрипту) для зберігання даних показників з сенсорів COA, тому що є необхідність у подальшій обробці, аналізу та візуалізації даних. Діаграму діяльності даного скрипту наведено на рисунку 5.8.

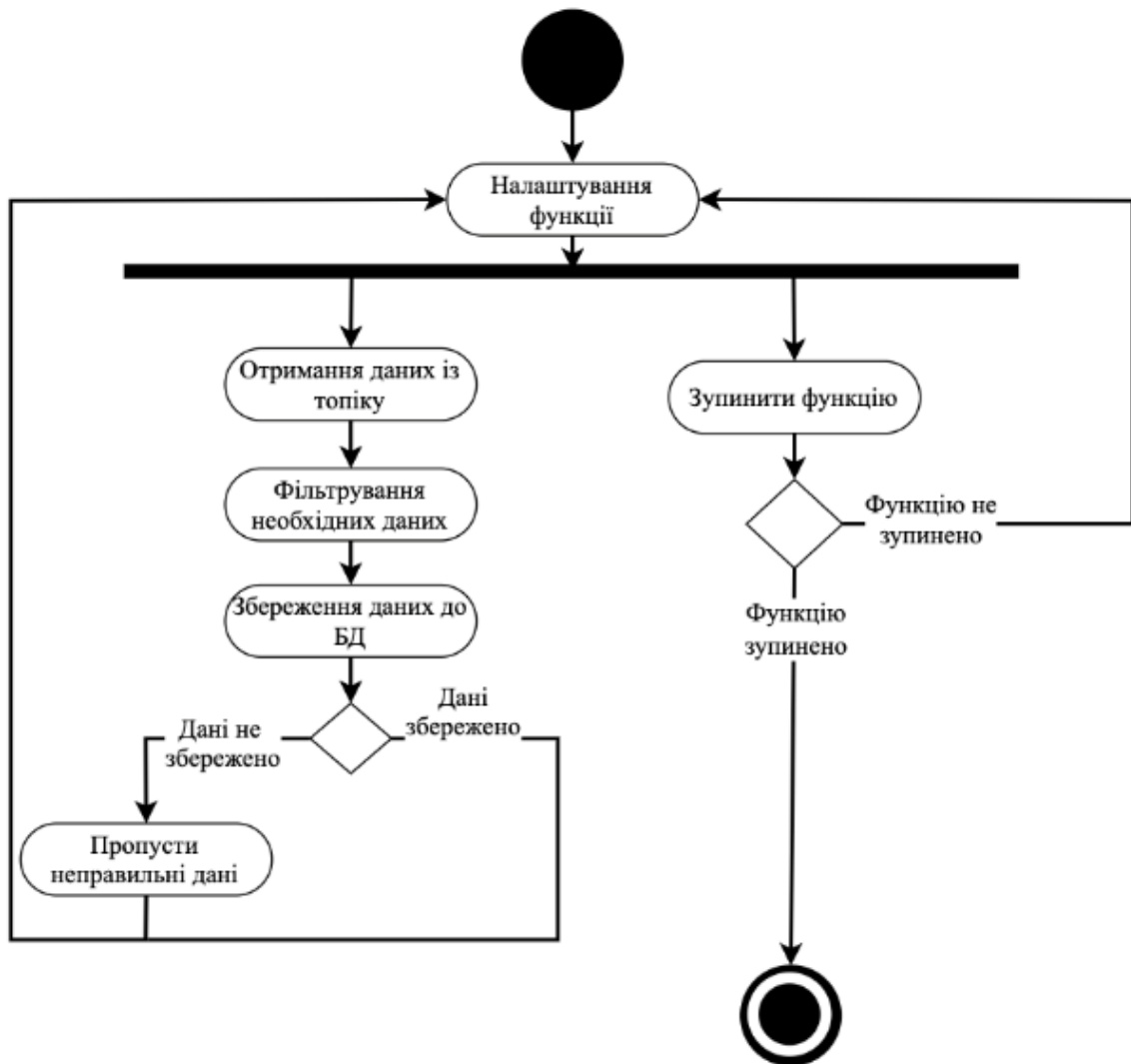


Рисунок 5.8 – Діаграма діяльності роботи скрипту

Дана функція була розроблена із використанням стандартного App Service плану. В ресурсі Service Bus Queue було налаштовано топіки, до яких буде можливість зробити підписку. Даний функціонал дозволить набагато швидше отримувати повідомлення з СОА та почати їх обробку.

В платформі Azure наявна можливість створювати функції, що підписуються на конкретний Service Bus Queue Topic, що було використано для реалізації даної функції.

На кожне повідомлення з СОА дана функція оброблює дані, видаляючи зайві, та підводячи у необхідні формати для того, щоб зберегти їх у БД. Зайвими

даними є ті, що не наділені інформацією про енергоспоживання, або просто містять неправильну інформацію, наприклад замість значення з показнику напруги надходить якийсь випадковий рядок, що не є числом, тощо.

Оскільки з системи СОА наявен дуже великий потік даних, тому, у випадку неправильних даних, ми просто пропускаємо цей запис, без сповіщення розробника. Якщо буде надсилатися повідомлення на кожен неправильний запис то буде надсилатися величезна кількість сповіщень, і будуть витрачені зайві кошти на системи сповіщення у системі Azure.

В результаті було розроблено функціонал, що дозволить приводити до правильного вигляду, зберігати їх у БД та відфільтровувати неправильні записи, що дозволить робити більш чіткий та правильний ІА даних. Приклад коду наведено в додатку А.

5.4 Конструювання скрипту для створення вибірок для модулю машинного навчання

Розроблювана система потребує у функціоналі для створення вибірок для Azure ML тому що даний ресурс потребує у тренуванні алгоритмів для надання більш точних результатів показань енергоспоживання.

Проте навчання алгоритмів відбувається за рахунок певних вибірок даних, що зберігаються наприклад у форматі CSV. Це дозволить прискорити навчання за рахунок розпаралелювання даних процесів. Діаграму діяльності даної функції наведено на рисунку 5.9.

Даний функціонал повинен відпрацьовувати кілька разів на день, щоб створювати нові вибірки, щоб навчати алгоритм. Для цього було використано Azure Cron Function, що є частиною плану App Service Standard. Дану функцію було налаштовано на виклик кожні пів години.

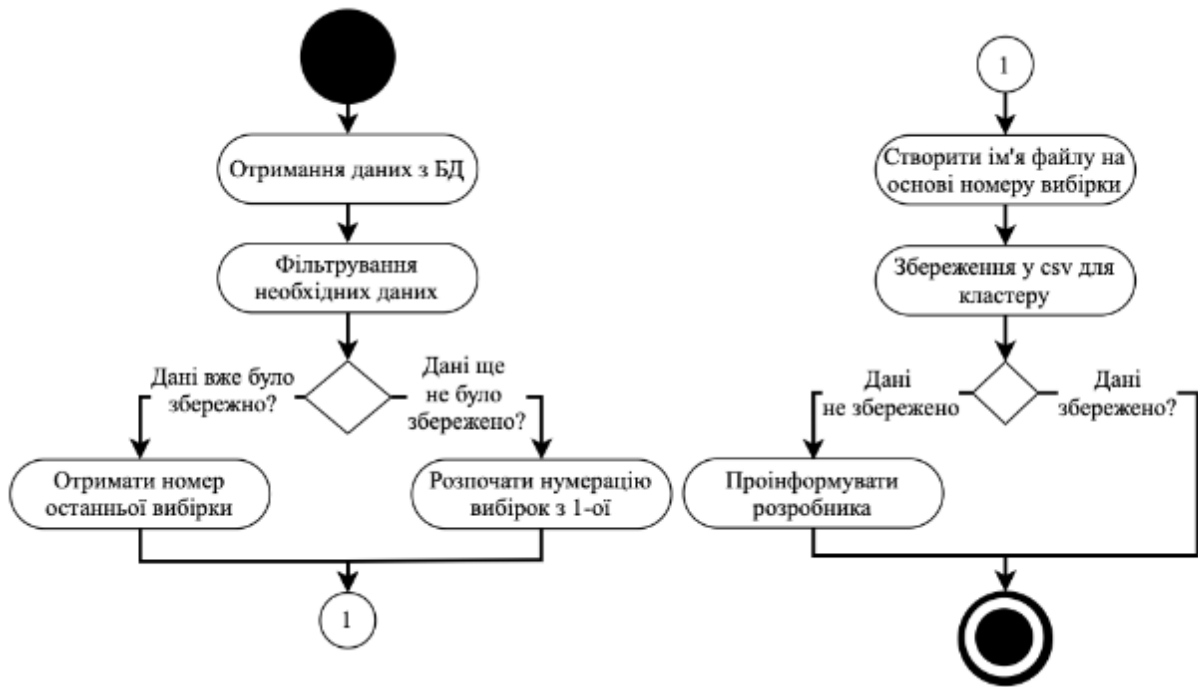


Рисунок 5.9 – Діаграма діяльності функції для генерування вибірок для Azure ML

Дана функція отримує дані з БД, потім згідно із порядковим номером вибірки, який отримується за рахунок кількості вже створених вибірок, зберігає цю ж вибірку даних до спеціально виділеного контейнеру в Azure Blob. Було обрано формат CSV для вибірок даних, оскільки це найзручніший формат для роботи із ним, та для налаштування «target» та «feature» в ресурсі Azure ML.

В результаті було розроблено функціонал для створення вибірок для тренування алгоритмів ресурсів Azure ML. Використання Azure Cron Function дозволило автоматизувати процес створення необхідних вибірок, що дає можливість покращувати результати прогнозування в наслідок постійного навчання алгоритмів. Приклад коду наведено в додатку А.

5.5 Конструювання клієнтської частини

Розроблювана система потребує у клієнтській частині тому що завдяки саме клієнтській частині відбувається взаємодія користувача із системою для аналізу даних енергоспоживання.

Діаграму компонентів клієнтської частини наведено на рисунку 5.10.

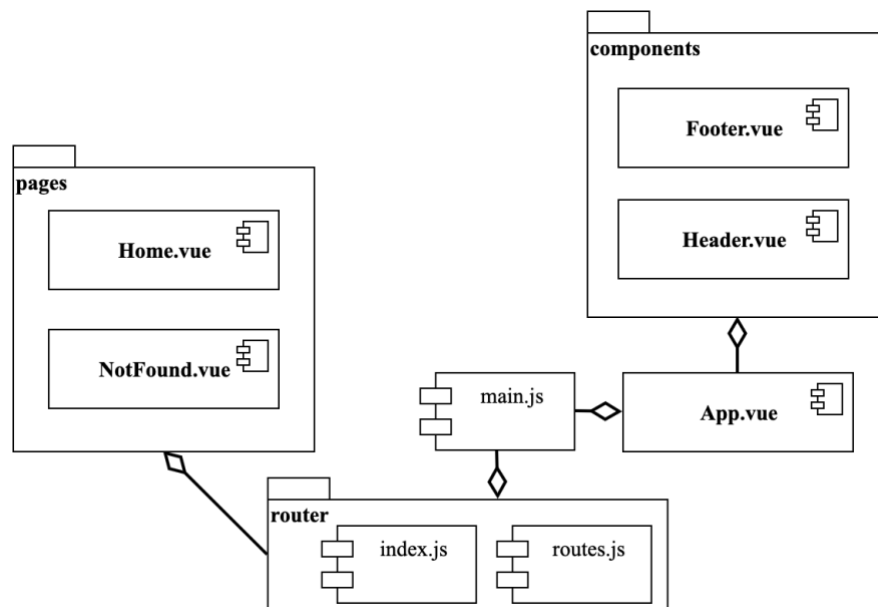


Рисунок 5.10 – Діаграма компонентів клієнтської частини

Модуль «main.js» являється точкою запуску програми. В даному модулі знаходяться конфігурації для структури шаблону системи, підключення компонентів, підключення бібліотек та підключення стилей.

Пакет «router» містить у собі конфігурації для веб сторінок системи із використанням режиму «history». Це дозволило обійти використання «#» в посиланнях та зберегти принцип роботи «Single Page Application» (SPA), тобто система відкриватиме нові сторінки без перезавантаження веб аплікації в цілому.

Було розроблено дві основні сторінки – «Home.vue» та «NotFound.vue». Сторінка «NotFound» відображається користувачу, коли він вказав неправильний шлях до сторінок розроблюваного веб застосунку. А сторінка «Home.vue» – це головна сторінка системи. Вона містить засоби для аналізу даних енергоспоживання.

Також було розроблено спеціальні компоненти, що виступають у ролі шапки та нижньої частини веб-інтерфейсу. Було прийняте таке рішення, щоб уніфікувати їх на всіх сторінках та зробити загальну інформацію для всього застосунку.

Діаграма діяльності взаємодії із програмним інтерфейсом серверної частини відбувається за принципом що наведено на рисунку 5.11.

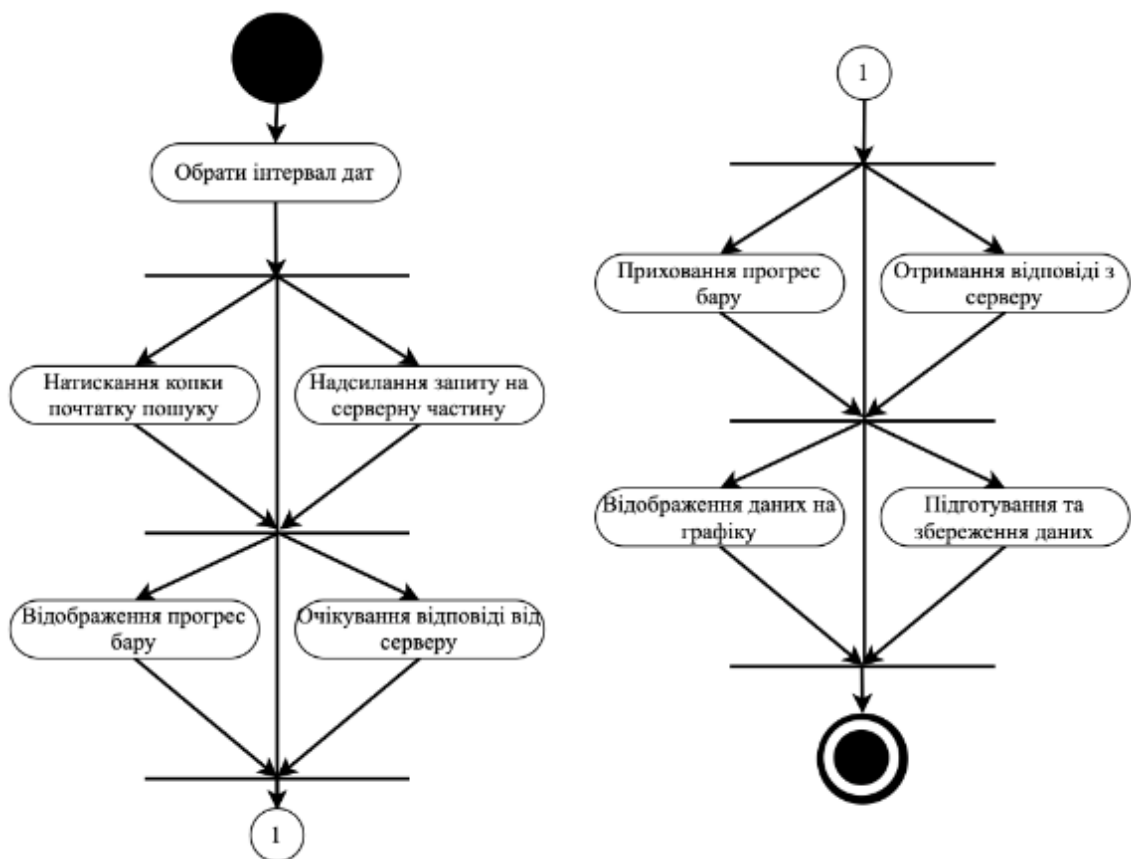


Рисунок 5.11 – Діаграма діяльності взаємодії із програмним інтерфейсом серверної частини

Для отримання даних енергоспоживання треба обрати дати за які є необхідність отримати дані та відобразити їх на графіку.

Після натискання кнопки початку пошуку клієнтська частина надсилає запит до серверної частини, щоб отримати історичні дані та дані прогнозування. Процес пошуку зображен у вигляді прогрес бару, як на рисунку 5.12.



Рисунок 5.12 – Прогрес бар під час очікування даних з серверної частини

Після отримання даних з серверу ці дані підставляються на графіку, і користувач має можливість аналізувати дані енергоспоживання СОА.

В результаті було розроблено зручну систему, та інтуїтивно зрозумілий інтерфейс, що дозволить проводити ІА даних енергоспоживання.

5.6 Конструювання модулю машинного навчання

Система потребує у модулі машинного навчання, щоб була можливість прогнозувати дані енергоспоживання для СОА на основі введених дат.

Оскільки було обрано ресурс Azure ML, тому є сенс використати графічний інтерфейс, щоб максимально швидко та якісно розробити «pipeline» яка буде навчати алгоритм.

Для використання «pipeline» треба спершу перевірити роботу даної системи. Результат її роботи наведено на рисунку 5.13.

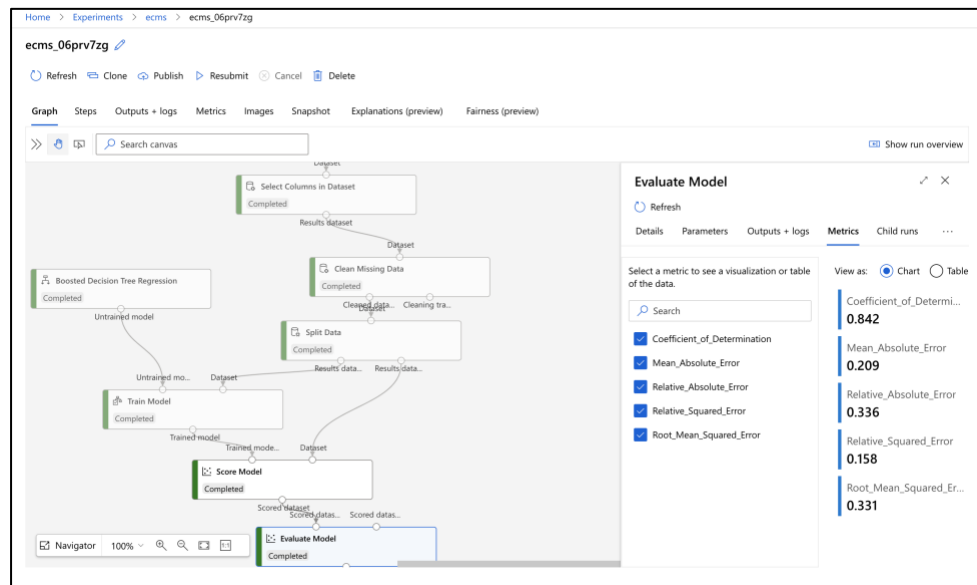


Рисунок 5.13 – Приклад роботи «pipeline»

В результаті оцінки було отримано результати, які схожі з тестами, що було проведено в підрозділі 2.5.

Оскільки є необхідність у взаємодії клієнтської частини із модулем машинного навчання, тому було використано спеціальні налаштування для розробленої «pipeline», а саме – «Web Service Input» та «Web Service Output», як наведено на рисунку 5.14.

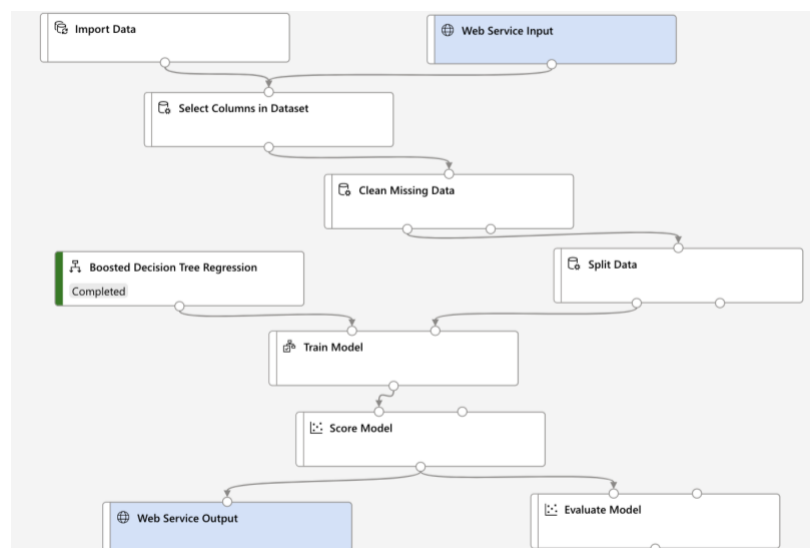


Рисунок 5.14 – Приклад повної схеми роботи Azure ML сервісу

Дане рішення дозволить створити програмний інтерфейс для взаємодії із розробленим модулем машинного навчання, а саме – «real-time endpoint». Оскільки усе налаштування необхідних ресурсів було розроблено в підрозділі 5.1, тому треба лише завантажити дану схему у спеціально розроблену для цього «pipeline».

В результаті було розроблено схему модулю машинного навчання, що дозволить робити прогнозування даних енергоспоживання за конкретними датами і дозволить правильно аналізувати дані із сенсорів COA та прогнозувати потенційні загрози, або прогнозувати потенційний розмір витрат для майбутніх планувань.

5.7 Висновки до розділу 5

В даному розділі описано конструювання системи.

Було розроблено файлову структуру, що включає в собі файл для конфігурацій проекту «.env» та приклад, що зберігається в репозиторії – «.env.default». Також було розроблено файл «.gitignore», щоб не зберігати у репозиторії непотрібну, або приватну інформацію.

Було розроблено та налаштовано серверну частину, що надало можливість для коректної взаємодії із клієнтською частиною, ресурсом Azure Machine Learning. Також був розроблений функціонал для хмарної платформи для перевірки доступності серверної частини для взаємодією із нею.

Було розроблено функціонал, що дозволить приводити до правильного вигляду, зберігати їх у БД та відфільтровувати неправильні записи, що дозволить робити більш чіткий та правильний ІА даних.

Було розроблено функціонал для створення вибірок для тренування алгоритмів ресурсів Azure ML. Використання Azure Cron Function дозволило автоматизувати процес створення необхідних вибірок, що дає можливість

покращувати результати прогнозування в наслідок постійного навчання алгоритмів.

Було розроблено клієнтську частину, що являє собою – зручну систему, та інтуїтивно зрозумілий інтерфейс, що дозволить проводити ІА даних енергоспоживання.

Було розроблено схему модулю машинного навчання, що дозволить робити прогнозування даних енергоспоживання за конкретними датами і дозволить правильно аналізувати дані із сенсорів СОА та прогнозувати потенційні загрози, або прогнозувати потенційний розмір витрат для майбутніх планувань.

6 КЕРІВНИЦТВО З ВИКОРИСТАННЯ СИСТЕМИ

6.1 Призначення та умови використання системи

Розроблена система для ІА даних енергоспоживання, яку було розроблено для аналізу даних енергоспоживання, отриманих з систем СОА та контролю потенційних загроз. Таким чином розроблена система дозволяє аналізувати дані енергоспоживання з СОА, та робити прогнозування на конкретний день, тиждень або місяць.

Система призначена для організації спеціального інтерфейсу для контролю даних з датчиків та сенсорів електроживлення СОА.

6.2 Функції системи

Розроблена система повинна виконувати наступні функції:

- давати доступ тільки для авторизованих користувачів;
- надавати доступ тільки для певної групи користувачів;
- надати засоби перегляду історичних даних напруги у мережі за тиждень та місяць;
- надати засоби для прогнозування напруги на обраний користувачем інтервал;
- отримати повні показники з датчиків та сенсорів енергоспоживання за конкретний день.

6.3 Вхідні та вихідні дані

Початкові або вхідні дані — це дати, що користувач надсилатиме для отримання потенційних даних електроспоживання, та опції, що будуть обрані для

взаємодії із системою. Користувач повинен бути авторизованим ви системі, щоб мати можливість взаємодіяти із системою.

У якості вихідних даних користувач отримуватиме можливість взаємодіяти із розроблюваною системою використовуючи функціонал графічного інтерфейсу.

6.4 Налаштування системи

Для роботи з системою, необхідно виконати наступні інструкції для його запуску та налаштування.

Першим кроком є перевірка на встановленість необхідного програмного забезпечення, а саме:

- Git;
- Azure CLI;
- Terraform CLI;

Наступним кроком є клонування репозиторію у обрану користувачем директорію на локальній машині.

Після цього треба в терміналі увійти в клонований проєкт та в ньому у директорію «infrastructure». Уся подальша конфігурація відбуватиметься в даній директорії.

Далі треба авторизуватися у системі Azure використовуючи команду «az login». Після цієї команди повинна відкритися сторінка авторизації, де треба увійти в свій профіль в системі Azure як наведено на рисунку 6.1.

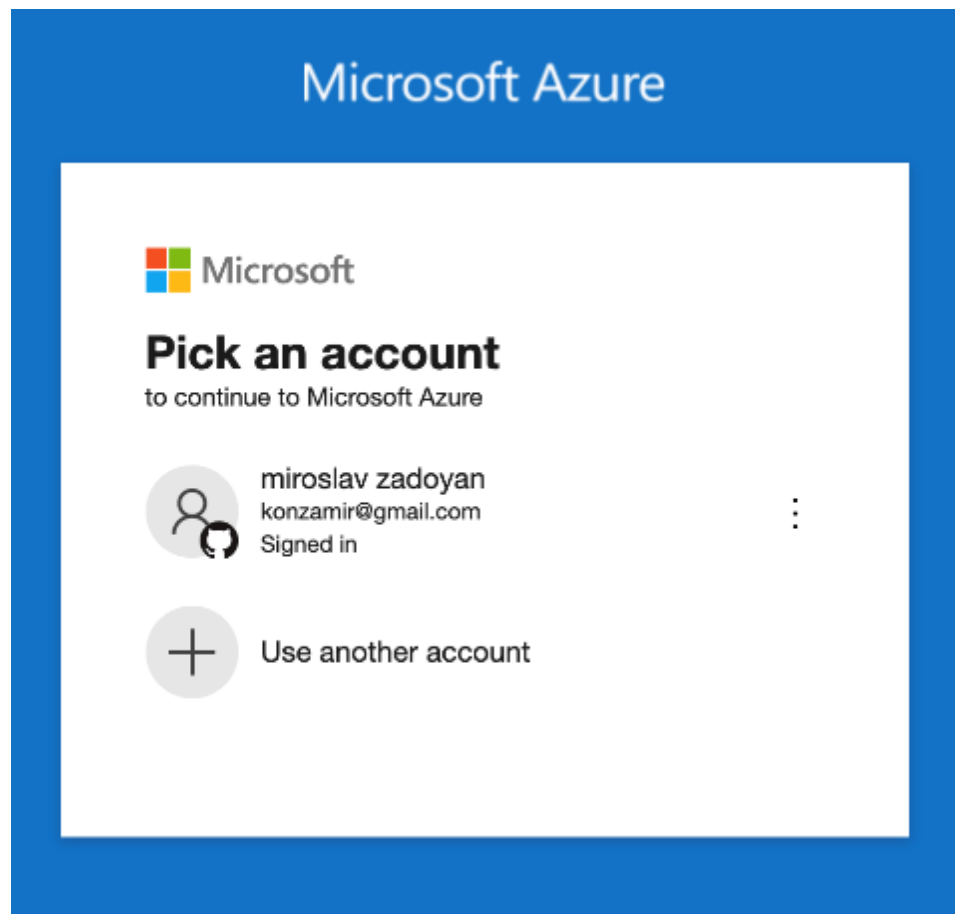


Рисунок 6.1 – Приклад авторизації в системі Azure

Наступним кроком є створення підписки та збереження даних підписки у файлі «variables.tf».

Далі треба проініціалізувати Terraform проєкт, згенерувати «план» запуску та налаштування ресурсів системи та застосувати його до платформи Azure викликавши відповідно команди:

- «terraform init»;
- «terraform plan -out main.tfplan»;
- «terraform apply main.tfplan».

Після успішного виконання всіх кроків в терміналі повинні бути наступні повідомлення, як на рисунку 6.2.

```

terraform v1.3.10
Initializing plugins and modules...
module.resourcegroup.module.names.random_id.default: Creating...
module.resourcegroup.module.names.module.names_location.random_string.min: Creating...
module.resourcegroup.module.names.module.names_environment.random_string.first_letter: Creating...
module.resourcegroup.module.names.module.names_location.random_string.first_letter: Creating...
module.resourcegroup.module.names.module.names_environment.random_string.name: Creating...
module.resourcegroup.module.names.random_id.default: Creation complete after 0s [id=yg0]
module.resourcegroup.module.names.module.names_location.random_string.min: Creation complete after 0s [id=81397axkc4hlw3lhl4ls0btqfxkldszxpilm4krm9ftufjyln8cwy377f]
module.resourcegroup.module.names.module.names_location.random_string.first_letter: Creation complete after 0s [id=u]
module.resourcegroup.module.names.module.names_environment.random_string.first_letter: Creation complete after 0s [id=x]
module.resourcegroup.module.names.module.names_environment.random_string.name: Creation complete after 0s [id=kt4t5kn65h2cx05b5ztl19b6jgq6qau0llaoxghj5pfcnl20n44c0r82]
module.resourcegroup.azurem_resource_group.default: Creating...
module.resourcegroup.azurem_resource_group.default: Creation complete after 1s [id=/subscriptions/22038543-97c5-4f4a-a31e-97fa0c28alb/resourcegroups/rj-ecns]

Apply complete! Resources: 24 added, 0 changed, 0 destroyed.

```

Рисунок 6.2 – Приклад успішного виконання всіх команд

Після отримання повідомлення, інфраструктура налаштована та готова до взаємодії із нею.

Наступний крок це використовуючи Deployment Center завантажити код як наведено на рисунку 6.3.

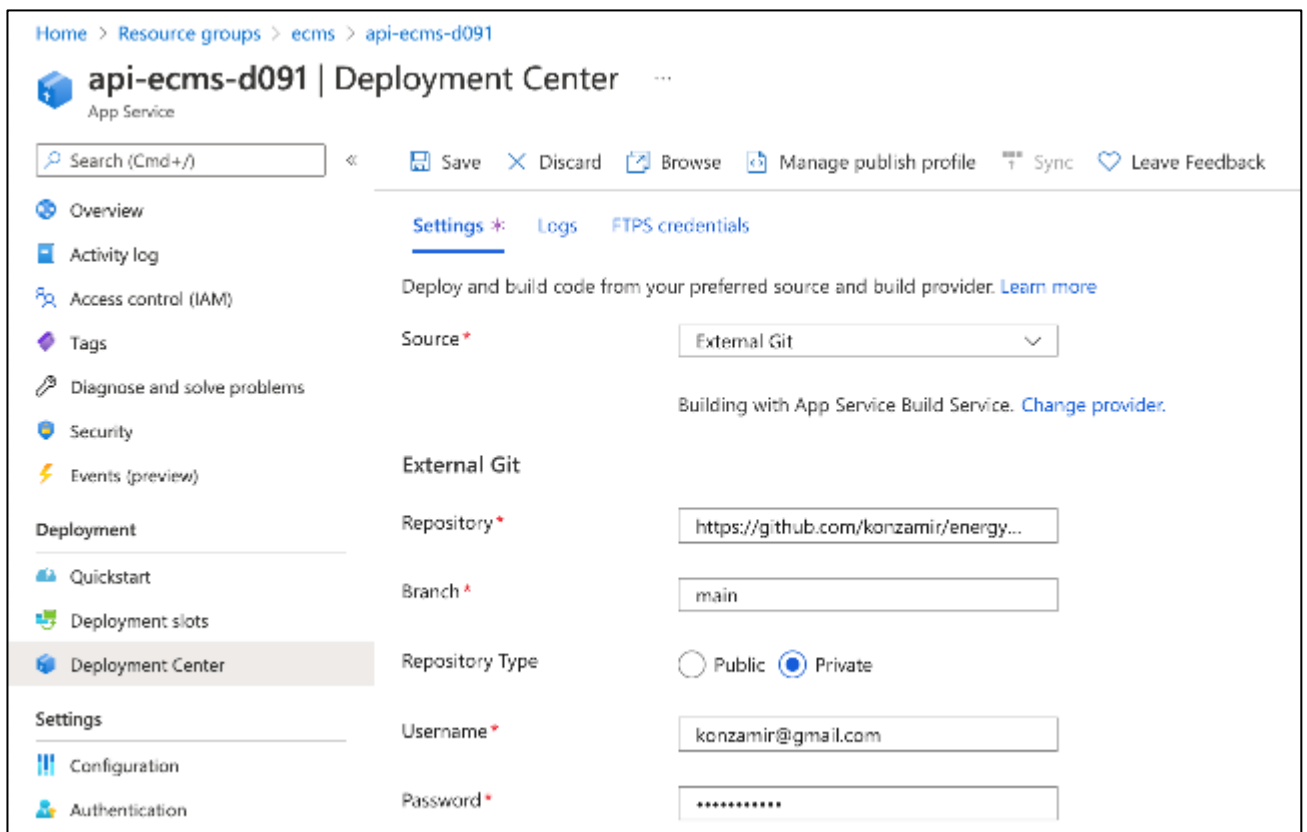


Рисунок 6.3 – Завантаження коду використовуючи Deployment center

Для кращого розуміння, дана інструкція представлений на рисунку 6.4 у вигляді діаграми діяльності.

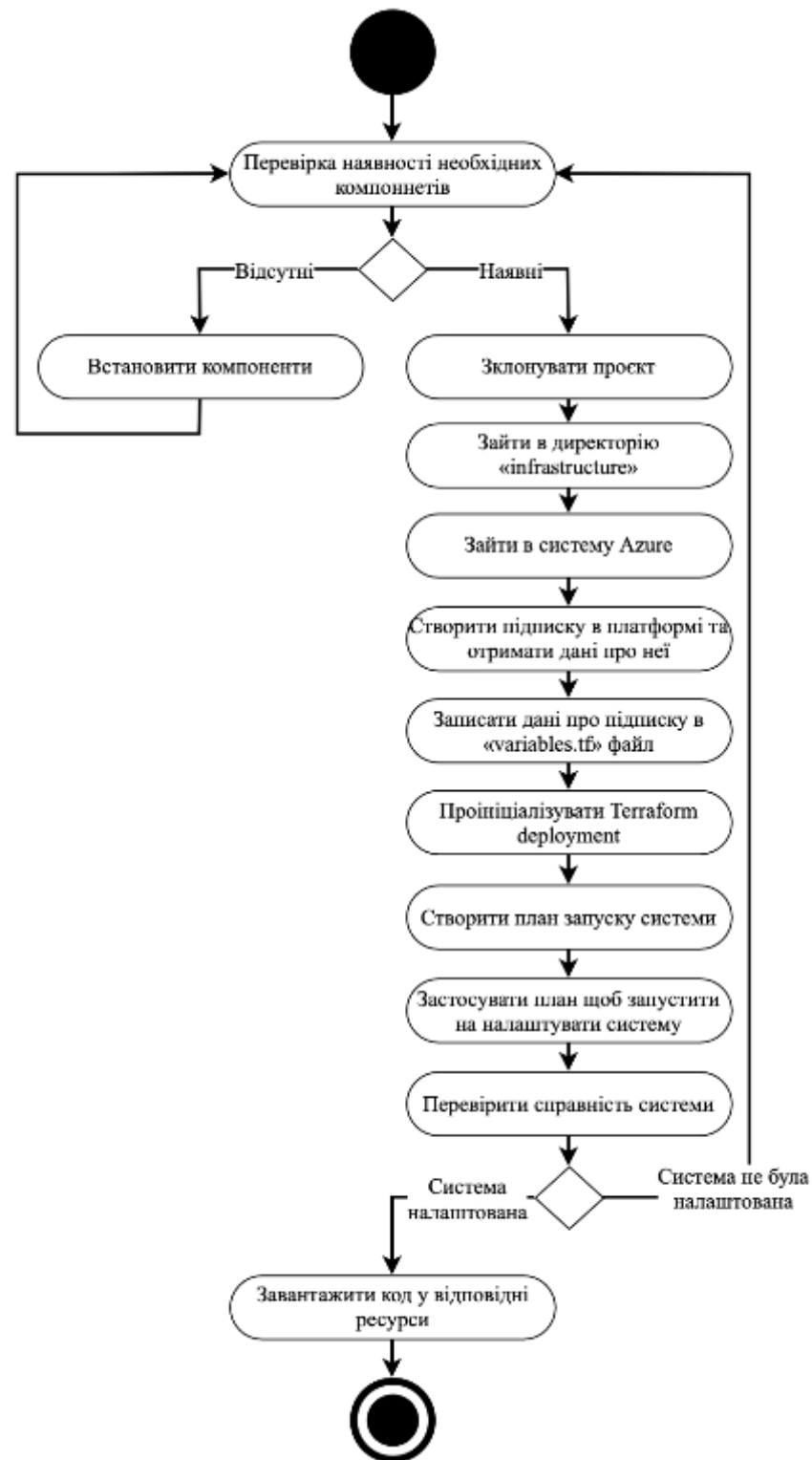


Рисунок 6.4 – Діаграма діяльності для запуску системи

6.5 Початок роботи з системою

Для використання системи необхідно авторизуватися в системі. Авторизація в системі відбувається за рахунок Active Directory. Приклад авторизації наведено на рисунку 6.5.

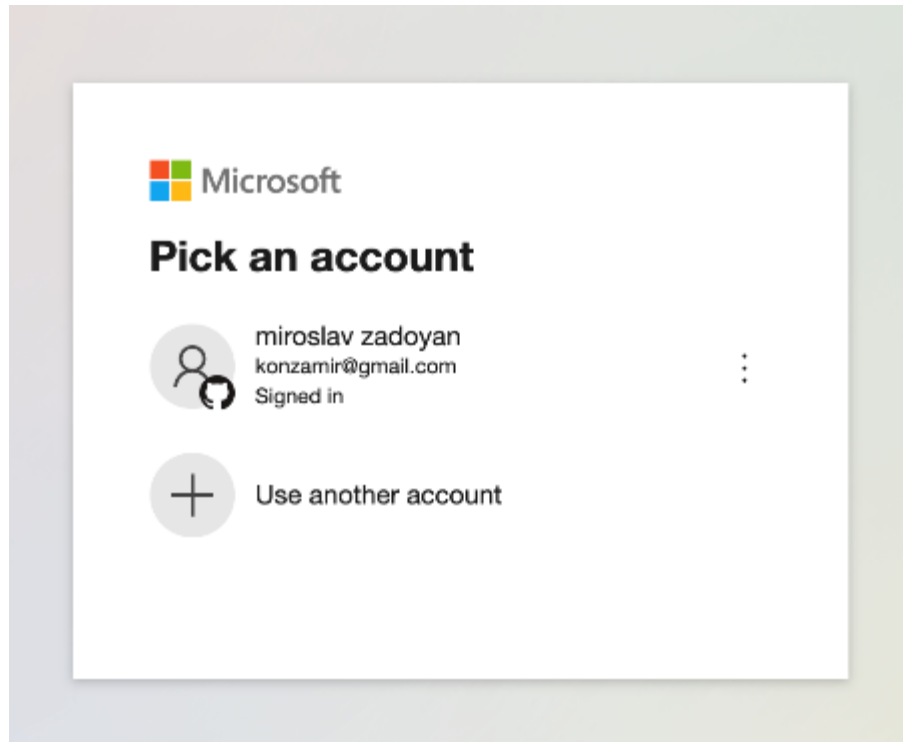


Рисунок 6.5 – Приклад авторизації в системі

Авторизація відбувається коли користувач вперше відкриває систему. Дана авторизація відбувається за рахунок засобів платформи Azure. Далі спеціальний токен для авторизації зберігається в сховищі веб-браузеру, де була відкрита система.

Усі подальші запуски відбуватимуться без попередньої авторизації, за наявності даного токена в сховищі браузеру.

В результаті успішного виконання інструкцій Користувач зможе побачити головну сторінку, приклад наведено на рисунку 6.6.



Рисунок 6.6 – Приклад головної сторінки

6.6 Взаємодія із системою

Керування системою відбувається за допомогою кнопок, що знаходяться вгорі екрану (рис. 6.7).



Рисунок 6.7 – Панель керування системою

Опис даних кнопок наведено в таблиці 6.1.

Таблиця 6.1 – Опис кнопок керування системою

Назва	Опис
Дата початку	Кнопка при натисканні якої з'являється вікно для вибору дати початку пошуку (рис 6.8).
Кінцева дата	Кнопка при натисканні якої з'являється вікно для вибору дати закінчення пошуку (рис 6.8).
Обрати метрику	Кнопка-випадаючий список, що дає можливість обрати метрику, по якій треба відображати графік (рис 6.9). Перед початком пошуку вона деактивована, оскільки ще не біло отримано тих самих метрик.
Почати пошук	Кнопка що починає пошук

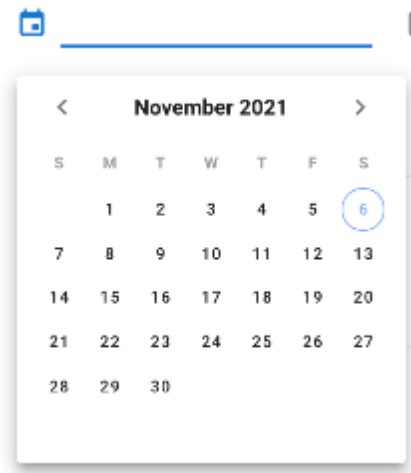


Рисунок 6.8 – Вікно вибору дати

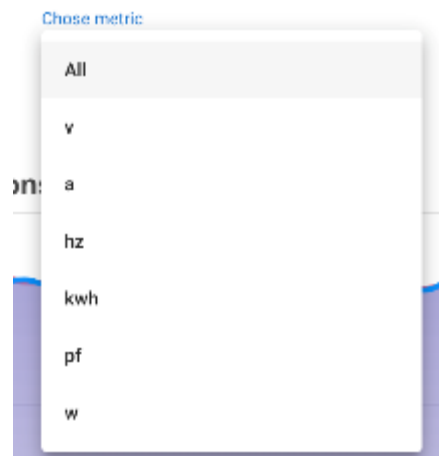


Рисунок 6.9 – Список метрик для відображення на графіку

Після того, як було розпочато пошук, дані, що надійшли з серверної частини будуть відображені на графіку, як наведено на рисунку 6.10.

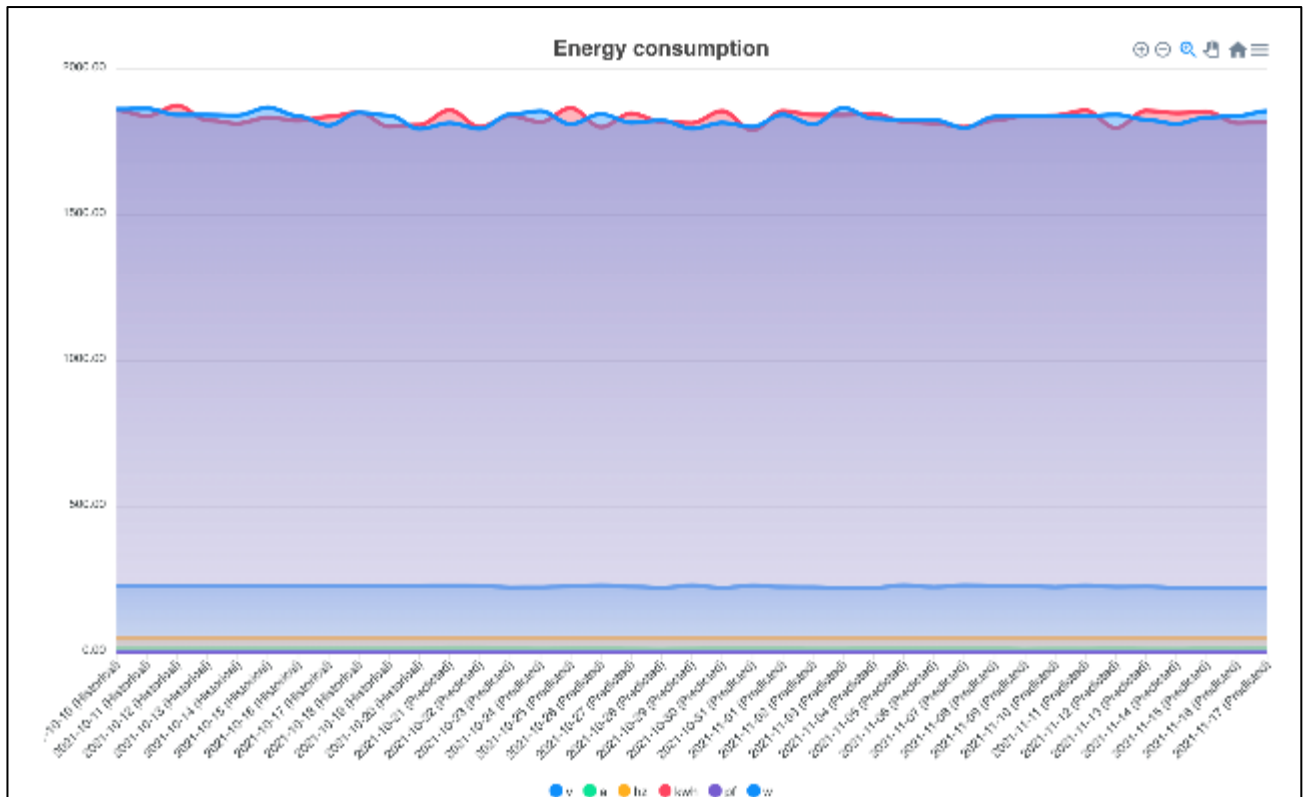


Рисунок 6.10 – Приклад графіку із даними

Даний графік містить показники з усіх датчиків. На нижній осі наведено дата, якій відповідають показники енергоспоживання. Ті значення, що було отримано з СОА помічені у дужках словом «Historical», а ті що було прогнозовано системою мають позначку «Predicted».

В системі наявна можливість отримати інформацію по конкретній точці на графіку, як наведено на рисунку 6.11. Для цього треба просто навести курсор на необхідну точку і графік надасть дані за обрано датою.

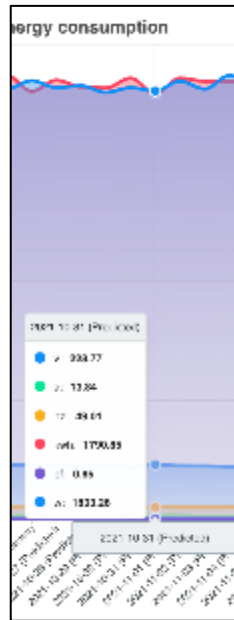


Рисунок 6.11 – Приклад детальної інформації по конкретному дню

В системі наявна можливість отримати дані по більш вузькому інтервалу дат, не запускаючи процес пошуку знову. Для цього треба на графіку обрати інтервал, який цікавить користувача як наведено на рисунку 6.12.

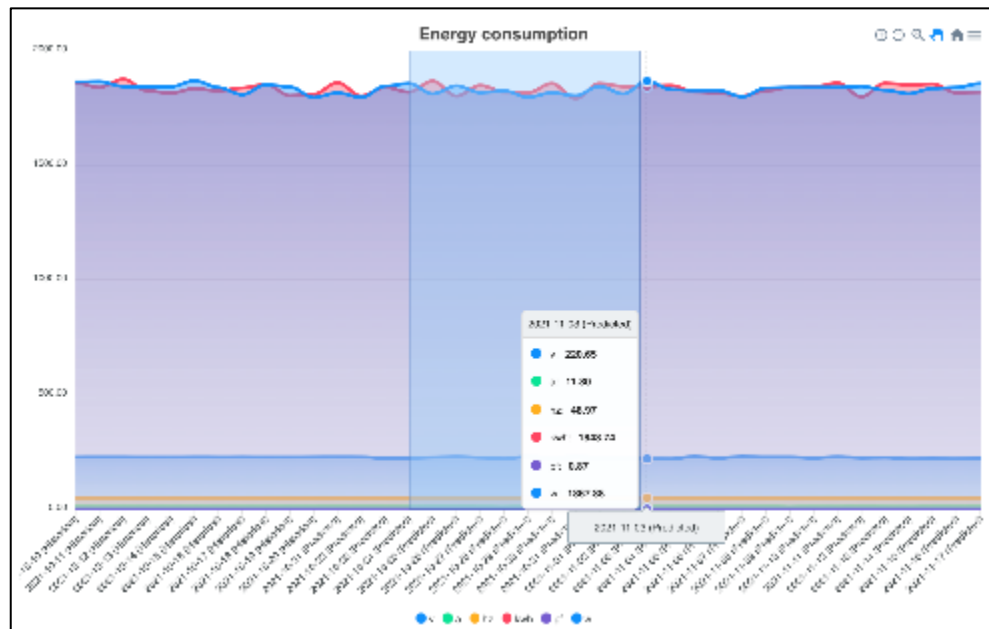


Рисунок 6.12 – Обирання необхідного інтервалу для отримання даних енергоспоживання

В системі також наявна можливість отримати дані за конкретним показником. Для цього треба натиснути на кнопку та у випадаючому списку обрати необхідну метрику, як показано на рисунку 6.13.

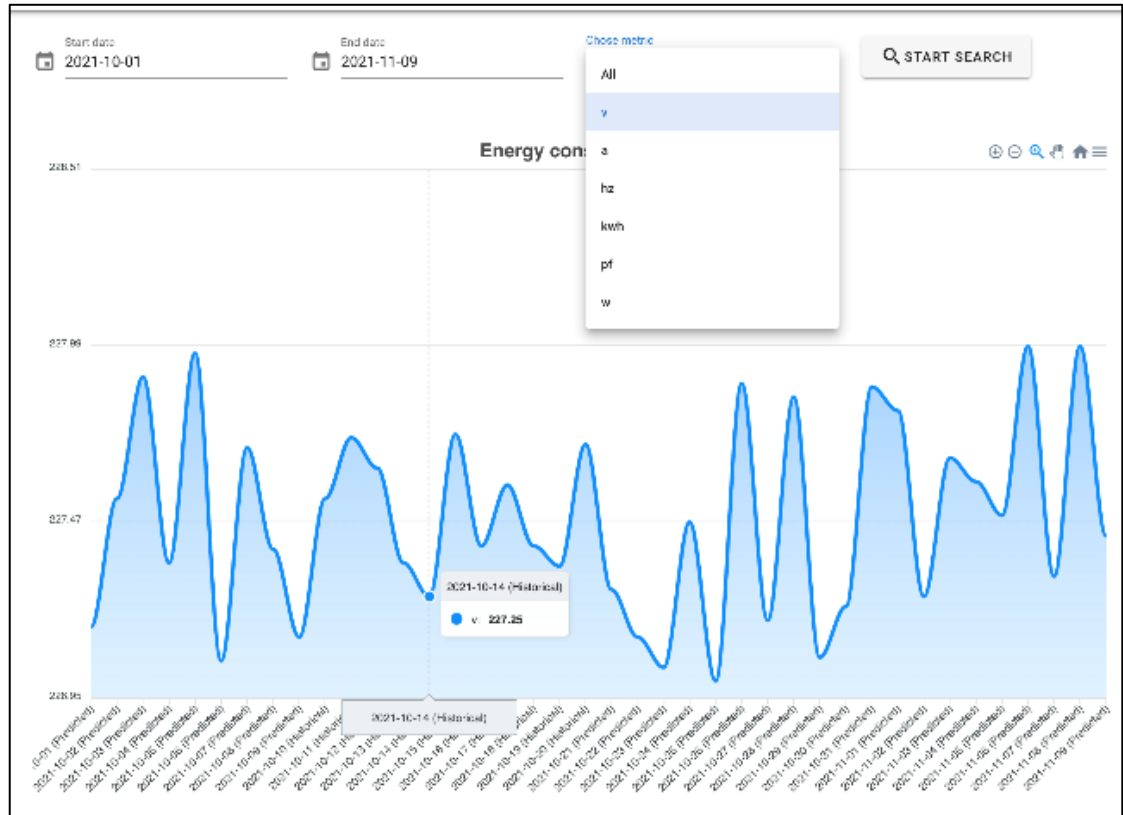


Рисунок 6.13 – Приклад даних по конкретній метриці

6.7 Висновки до розділу 6

Розглянуто умови використання розробленої системи та її призначення, функції, вхідні та вихідні дані системи. Також описано покрокове налаштування та описано інструкцію для користувачів з використання.

В результаті створено спеціальний інтерфейс для контролю даних з датчиків та сенсорів електроживлення СОА. Таким чином розроблена система дозволяє аналізувати дані енергоспоживання з СОА, та робити прогнозування на конкретний день, тиждень або місяць.

ВИСНОВКИ

В результаті проведених досліджень можна зробити висновок, що для сучасних систем СОА необхідні засоби для аналізу даних енергоспоживання, щоб ефективно планувати витрати на електроенергію та контролювати потенційні загрози. Впровадження інтелектуального аналізу даних енергоспоживання є актуальною задачею, оскільки воно забезпечить підвищення точності та зручності обробки великих обсягів даних, отримуваних з СОА. Використання програмного застосунку дозволить зменшити кількість помилок, що спричинені людським фактором.

В результаті порівняння методів ІА для СОА, було прийнято рішення, що доцільно буде використати онлайн аналітику та аналітику прогнозувань. Було зроблено порівняння алгоритмів із використання програмного засобу Orange. Найкращим алгоритмом згідно значення середньої похибки для обраної вибірки є Boosted Decision Tree, який показує найстабільнішу величину похибки та із збільшенням кількості записів в БД він надаватиме кращі результати, оскільки даний алгоритм наділений поєднанням переваг алгоритмів AdaBoost та Tree.

Прийнято рішення запустити розроблюваний програмний продукт на хмарній платформі Microsoft Azure. Проведено аналіз користувацьких інтерфейсів для взаємодії із програмним продуктом. В результаті чого вирішено розробити власну клієнтську частину. Спроектowano архітектуру системи, яка складається з десяти основних компонентів. Для зберігання даних системи було спроектовано структуру СКБД. Спроектowano базовий макет сайту.

В результаті проведеного аналізу в якості інтегрованого середовища розробки було обрано Visual Studio Code. Проаналізовано мови програмування та для розробки серверної частини обрано Python та бібліотеку FastAPI. JavaScript та бібліотеку Vue було обрано як результат аналізу мови програмування для розробки клієнтської частини.

Програмну систему було розроблено та налаштовано згідно вимог та існуючих стандартів.

Розглянуто умови використання розробленої системи та її призначення, функції, вхідні та вихідні дані системи. Також описано покрокове налаштування та описано інструкцію для користувачів з використання.

Наукова новизна роботи полягає в тому, що отримав подальший розвиток алгоритм Boosted Decision Tree за рахунок удосконалення етапів підготовки та виведення даних. Пропонована реалізація представляє собою окремий сервіс для надання даних щодо прогнозування енергоспоживання в СОА та, на відміну від існуючих рішень, забезпечує програмний інтерфейс з використанням TCP/IP засобів для взаємодії з ним.

Практична цінність роботи полягає в тому, що за рахунок використання розробленої програмної системи:

- підвищиться ефективність планування бюджету компанії та заходів щодо безпечного використання техніки та обладнання за рахунок прогнозування даних енергоспоживання;
- зменшиться час обчислень та потрібні ресурси для обробки великих об'ємів даних за рахунок використання оптимізованого алгоритму Boosted Decision Tree.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- 1 Система енергоефективності в Україні [Електрон. ресурс]. – Режим доступу: <https://www.minregion.gov.ua/wp-content/uploads/2018/09/GIZ-brochure.pdf>.
- 2 Енергетична стратегія України на період до 2030 р. [Електрон. ресурс]. – Режим доступу: <https://de.com.ua/uploads/0/1703-EnergyStrategy2030.pdf>.
- 3 How does home automation work? The complete guide [Electronic source]. – Access mode: <https://elementsmartcontrol.com/how-home-automation-works/>.
- 4 Industrial control and automation systems [Electronic source]. – Access mode: <http://www.regelgroup.com/main/industrial-control-automation-systems-icas/>.
- 5 Industry 4.0 based process data analytics platform: A waste-to-energy plant case study [Electronic source]. – Access mode: <https://www.sciencedirect.com/science/article/pii/S0142061518336731>.
- 6 Відкрите ПЗ для керування розумним будинком [Електрон. ресурс]. – Режим доступу: <http://www.rovdo.com/smart-home-platform-diy-software>.
- 7 Home Assistant Vs OpenHAB [Electronic source]. – Access mode: <https://appuals.com/home-assistant-vs-openhab/>.
- 8 Карта-схема магістральних мереж ОЕС України [Електрон. ресурс]. – Режим доступу: http://mpe.kmu.gov.ua/minugol/control/publish/article?art_id=245156044.
- 9 The importance of energy monitoring [Electronic source]. – Access mode: <https://www.em-solutions.co.uk/insights/the-importance-of-energy-monitoring>.
- 10 Розумний дім [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B7%D1%83%D0%BC%D0%BD%D0%B8%D0%B9_%D0%B4%D1%96%D0%BC

- 11 Smart energy monitoring and management in large multi-office building environments [Text] / [A.Orestis, A. Dimitrios, D. Dimitrios, et al.] // In Proceedings of the 17th Panhellenic conference on informatics, 19 September 2013 : proceedings. – 2013. – P. 219–226.
- 12 Intelligent energy system services [Electronic source]. – Access mode: <https://www.mhp.com/de/intelligent-energy-system-services>.
- 13 Баклер – розумна енергетика [Electronic source]. – Access mode: <https://bakler.com.ua/check.html>.
- 14 Баклер онлайн-сервіс [Electronic source]. – Access mode: <https://buckler.pw/>.
- 15 Home energy meter Gen5 [Electronic source]. – Access mode: <https://aeotec.com/z-wave-home-energy-measure/>.
- 16 Умный счетчик электроэнергии с WiFi [Электрон. ресурс]. – Режим доступа: <https://store.smart-mac.com/p684214708-umnyj-schetchik-elektroenergii.html>
- 17 Wi-fi лічильник електроенергії з функцією захисту та управління ем-
129 [Електрон. ресурс]. – Режим доступу: <https://novatek-electro.com/ua/product/wi-fi-lichilnik-elektroenergii-z-funkciie.html>.
- 18 Smart home energy management, using IoT system [Text] / [Hosseinian, Heliasadat, Damghani et al.] // In: 2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI) : proceedings. – IEEE, 2019. – P. 905–910.
- 19 Straub A. User's perspective on home energy management systems [Electronic source] / A. Straub, E. Volmer. – Access mode: <https://www.mdpi.com/2076-3298/5/12/126/htm>.
- 20 KC R. an investigation of the behavioral characteristics of higher- and lower-temperature group families in a condominium equipped with a HEMS system [Electronic source] / R. KC, K. Yoshida, M. Shukuya – Access mode: https://www.mdpi.com/2075-5309/9/1/4/review_report.

21 Web based smart meter for general purpose smart home systems with ESP8266 [Text] / [Yildiz, Sadık, Burunkaya, et al] // In: 2019 3rd International symposium on multidisciplinary studies and innovative technologies (ISMSIT) : proceedings. – IEEE, 2019. – P. 1–6.

22 Tastan, M. Internet of Things based smart energy management for smart home [Text] / M. Tastan // KSII Transactions on internet and information systems : proceedings. – 2019.

23 Open-source free home automation systems [Electronic source]. – Access mode: <https://www.medevel.com/17-home-automation-open-source/#1-home-assistant>.

24 Pelae, A. Open source home automation platforms to use in 2020 [Electronic source] / Pelae, A. – Access mode: <https://ubidots.com/blog/open-source-home-automation/>.

25 Arrows, K. Home Assistant Vs OpenHAB [Electronic source] / Arrows, K. – Access mode: <https://appuals.com/home-assistant-vs-openhab/>.

26 OpenHAB Vs Home Assistant: detailed comparison by an expert in 2021 [Electronic source]. – Access mode: <https://purdylounge.com/openhab-vs-home-assistant/>.

27 Sustainable smart home and home automation: Big Data analytics approach [Electronic source]. – Access mode: https://www.researchgate.net/publication/307855395_Sustainable_Smart_Home_and_Home_Automation_Big_Data_Analytics_Approach.

28 Role of data analytics in Internet of Things (IoT) [Electronic source]. – Access mode: <https://www.fingent.com/blog/role-of-data-analytics-in-internet-of-things-iot/>.

29 Internet of Things with Big Data analytics - a survey [Electronic source]. – Access mode:

https://www.researchgate.net/publication/332223887_Internet_of_Things_with_BIG_DATA_Analytics_-A_Survey.

30 Big IoT data analytics: architecture, opportunities, and open research challenges [Electronic source]. – Access mode: https://www.researchgate.net/publication/316240052_Big_IoT_Data_Analytics_Architecture_Opportunities_and_Open_Research_Challenges.

31 Waas, F.M. Beyond conventional data warehousing—massively parallel data processing with Greenplum database [Text] / F.M. Waas // In International workshop on business intelligence for the real-time enterprise. – Volume 27. – 2008. – P. 89-96.

32 Färber, F. SAP HANA database: data management for modern business applications [Text] / F. Färber // ACM sigmod record. – 2012. – P. 45-51.

33 Castro, M. SCRIBE: A large-scale and decentralized application-level multicast infrastructure [Text] / M. Castro // IEEE Journal on selected areas in communications. – 2002. – P. 1489-1499.

34 Kafka: A Distributed Messaging System for Log Processing [Electronic source]. – Access Mode: <http://notes.stephenholiday.com/Kafka.pdf>.

35 Notsu, H. Time-tunnel: visual analysis tool for time-series numerical data and its extension toward parallel coordinates [Text] / H. Notsu, // In International conference on computer graphics, imaging and visualization (CGIV'05): proceedings. – IEEE, 2005. – P. 167-172.

36 Katz R. Chukwa: A System for Reliable Large-Scale Log Collection [Text] / R. Katz, A. Rabkin // In 24th large installation system administration conference (LISA '10) : proceedings. – USENIX, 2010. – P. 1-15.

37 A dive into Decision Trees [Electronic source]. – 2020. – Access mode: <https://towardsdatascience.com/a-dive-into-decision-trees-a128923c9298>.

38 Machine learning basics with the K-Nearest neighbors algorithm [Electronic source]. – Access mode: <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>

39 Understanding Random Forest. How the algorithm works and why it is so effective [Electronic source]. – Access mode: <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>.

40 Support Vector Machines(SVM) — An Overview [Electronic source]. – 2018. – Access mode: <https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989>.

41 The ultimate guide to AdaBoost algorithm. What is AdaBoost algorithm? [Electronic source]. – Access mode: <https://www.mygreatlearning.com/blog/adaboost-algorithm/>.

42 How does Linear Regression actually work? [Electronic source]. – 2019. – Access mode: <https://towardsdatascience.com/how-does-linear-regression-actually-work-3297021970dd>.

43 Principle of neural network and its main types: review [Electronic source] – Access mode: <https://hal.univ-brest.fr/hal-03114305/document>.

44 Zen of Stochastic Gradient Descent Principle [Electronic source]. – Access mode: <https://hackernoon.com/zen-of-stochastic-gradient-descent-principle-pj1p83vsb>.

45 The top 10 data mining tools of 2018 [Electronic source] – Access mode: <https://www.analyticsinsight.net/the-top-10-data-mining-toolsof-2018/>.

46 10 Best data analysis tools for perfect data management (2021 LIST) [Electronic source]. – Access mode: <https://www.softwaretestinghelp.com/data-analysis-tools/>.

47 Data mining fruitful and fun [Electronic source]. – Access mode: <https://orange.biolab.si/>.

48 Big Data. What it is and why it matters [Electronic source]. – Access mode: https://www.sas.com/en_us/insights/big-data/what-is-big-data.html.

49 Boosted Decision Tree Regression component [Electronic source]. – Access mode: <https://docs.microsoft.com/en-us/azure/machine-learning/component-reference/boosted-decision-tree-regression>.

50 Cloud Computing for Home Automation [Electronic source]. – Access mode: https://www.auto.tuwien.ac.at/bib/pdf_TR/TR0167.pdf.

51 Web and cloud based Home Automation Systems: an overview [Electronic source]. – Access mode: <https://www.irjet.net/archives/V3/i11/IRJET-V3I11217.pdf>.

52 Microsoft technical documentation [Electronic source]. – Access mode: <https://docs.microsoft.com/en-us/>.

53 Google Cloud documentation [Electronic source]. – Access mode: <https://cloud.google.com/docs>.

54 AWS Documentation [Electronic source]. – Access mode: <https://docs.aws.amazon.com/>.

55 Azure Event Hub [Electronic source]. – Access mode: https://www.home-assistant.io/integrations/azure_event_hub/.

56 App Service Pricing | Microsoft Azure [Electronic source]. – Access mode: <https://azure.microsoft.com/en-us/pricing/details/app-service/windows/#pricing>.

57 Power BI embedded pricing [Electronic source]. – Access mode: <https://azure.microsoft.com/en-us/pricing/details/power-bi-embedded/#pricing>.

58 What is Power BI embedded analytics? [Electronic source]. – Access mode: <https://docs.microsoft.com/en-us/power-bi/developer/embedded/embedded-analytics-power-bi>.

59 McConnell S. Code complete [Text] / S. McConnell. – United States: Microsoft Press, 1993. – 914 c.

60 Прототип сайта: зачем он нужен и как создать его самостоятельно [Электрон. ресурс]. – Режим доступа: <https://serpstat.com/ru/blog/prototip-sajta-zachem-on-nuzhen-i-kak-sozdat/>.

61 Как сделать прототип сайта [Электрон. ресурс]. – Режим доступа: <https://blog.ringostat.com/ru/kak-sdelat-prototip-sayta/>.

62 Schults C. What is infrastructure as code? How it works, best practices, tutorials [Electronic resource] / C. Schults. – Access mode: <https://stackify.com/what-is-infrastructure-as-code-how-it-works-best-practices-tutorials/>.

63 Comparing database management systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, elasticsearch and others [Электронный ресурс]. – Режим доступа до ресурсу: <https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/>.

64 SQL против NoSQL на примере MySQL и MongoDB [Электронный ресурс] – Режим доступа до ресурсу: <https://tproger.ru/translations/sql-vs-nosql/>.

65 Method of calculating the scores of the DB-Engines Ranking [Electronic source]. – Access mode: https://db-engines.com/en/ranking_definition.

66 Топ-10 систем управления базами данных в 2019 году [Электронный ресурс] – Режим доступа до ресурсу: <https://proglab.io/p/databases-2019>.

67 Top 10 databases to use in 2021 [Электронный ресурс]. – Режим доступа до ресурсу: <https://towardsdatascience.com/top-10-databases-to-use-in-2021-d7e6a85402ba>.

68 PYPL popularity of programming language [Electronic source]. – Access mode: <https://pypl.github.io/PYPL.html>.

69 Python [Electronic source]. – Access mode: <https://www.python.org/>.

70 Java [Electronic source]. – Access mode: <https://www.java.com/en/>.

71 JavaScript [Electronic source]. – Access mode: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.

72 Typescript vs Javascript [Electronic source]. – Access mode: <https://hackr.io/blog/typescript-vs-javascript>.

73 Typescript vs JavaScript: what's the difference? [Electronic source]. – 2021. – Access mode: <https://www.guru99.com/typescript-vs-javascript.html>.

74 PyCharm Features [Electronic source]. – Access mode: <https://www.jetbrains.com/pycharm/features/>.

75 Documentation Visual Studio Code [Electronic source]. – Access mode: <https://code.visualstudio.com/docs>.

76 Documentation VIM [Electronic source]. – Access mode: <https://www.vim.org/docs.php>.

77 Top 13 Python web frameworks to learn in 2020 [Electronic source] – Access mode: <https://steelkiwi.com/blog/top-10-python-web-frameworks-to-learn/>.

78 What is WSGI? [Electronic source]. – Access mode: <https://wsgi.readthedocs.io/en/latest/what.html>.

79 ASGI Documentation [Electronic source]. – Access mode: <https://asgi.readthedocs.io/en/latest/>.

80 Django documentation [Electronic source]. – Access mode: <https://docs.djangoproject.com/en/3.0/>.

81 FastAPI [Electronic source]. – Access mode: <https://fastapi.tiangolo.com/>.

82 Flask's documentation [Electronic source]. – Access mode: <https://flask.palletsprojects.com/en/1.1.x/>.

83 Vue vs. Angular vs. React comparison [Electronic source] Access mode: <https://theappsolutions.com/blog/development/vue-angular-react-comparison/>.

84 Angular vs React vs Vue: which is the best choice for 2020? [Electronic source]. – Access mode: <https://levelup.gitconnected.com/angular-vs-react-vs-vue-which-is-the-best-choice-for-2020-81f577697c7e>.

85 Angular vs Vue vs React: choosing the best framework in 2020 [Electronic source]. – Access mode: <https://www.educative.io/blog/react-angular-vue>.

comparison?aid=5082902844932096&utm_source = google&utm_medium =
 cpc&utm_campaign = blog-dynamic&gclid=
 Cj0KCQjw6ar4BRDnARIsAITGzlBWg58w1pLFfJKt59Q3CoBma7GOjl0EusFkY-
 XC85F45lLDPLUabGYaArXbEALw_wcB.

86 Infrastructure (IT infrastructure) [Electronic source]. – Access mode:
<https://searchdatacenter.techtarget.com/definition/infrastructure>.

ДОДАТОК А
Текст програми

A.1 backend/src/main.py

```

import logging

import uvicorn
from fastapi import FastAPI, Request
from fastapi.responses import JSONResponse
from starlette.middleware.cors import CORSMiddleware

from src.app.urls import api_router
from src.config import settings
from src.events import setup_events

app = FastAPI(
    title=settings.PROJECT_NAME,
    openapi_url=f"{settings.API_V1_STR}/openapi.json",
    docs_url=f"{settings.API_V1_STR}/docs/",
)

setup_events(app)
# TODO: as separate task implement using gcloud.logging
# 'gunicorn.error' is using as a default logger for gunicorn
logger = logging.getLogger('gunicorn.error')

@app.exception_handler(Exception)
async def unicorn_exception_handler(request: Request, exc:
Exception):
    logger.exception(f"Internal server error: {exc}.",
exc_info=True)
    return JSONResponse(
        status_code=500,
        content={"message": f"Internal server error: {exc}.",
    )

# Set all CORS enabled origins
app.add_middleware(
    CORSMiddleware,
    allow_origins=['*'],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

app.include_router(api_router, prefix=settings.API_V1_STR)

if __name__ == "__main__":
    uvicorn.run(app, host="0.0.0.0", port=8000)

```

A.2 backend/src/config.py

```

import secrets
import os
from pydantic import BaseSettings

basedir =
os.path.dirname(os.path.dirname(os.path.realpath(__file__)))

class Settings(BaseSettings):
    PROJECT_NAME: str = "ECMS"

    # Local debug purposes
    DEBUG: bool = False
    DEBUG_EMAIL: str = None
    DEBUG_USERNAME: str = None

    SECRET_KEY: str = secrets.token_urlsafe(32)
    BASE_DIR: str = basedir
    API_V1_STR: str = "/api"

    DB_HOST: str
    DB_USER: str
    DB_PASSWORD: str
    DB_PORT: int = 1433
    DB_NAME: str

    class Config:
        case_sensitive = True
        env_file = ".env"

settings = Settings()

```

A.3 backend/src/connections/db_connection.py

```

from asyncio import get_running_loop, get_event_loop

from aiopg.sa import create_engine
from src.config import settings

def init_pg_connection(app):
    loop = get_running_loop()

```

```

tsk = loop.create_task(create_engine(
    database=settings.DB_NAME,
    user=settings.DB_USER,
    password=settings.DB_PASSWORD,
    host=settings.DB_HOST,
    port=settings.DB_PORT
))

def set_db(t):
    app.db = t.result()

tsk.add_done_callback(
    set_db
)

def close_pg_connection(app):
    loop = get_running_loop()
    if getattr(app, "db"):
        app.db.close()
        loop.create_task(app.db.wait_closed())

```

A.4 backend/src/app/urls.py

```

from fastapi import APIRouter

from src.app.views import health_view, consumption

api_router = APIRouter()

api_router.include_router(health_view.router, prefix="/health",
tags=["health"])
api_router.include_router(consumption.router,
prefix="/consumption", tags=["consumption"])

```

A.5 backend/src/app/sql_service/query_formatter.py

```

import enum
import os

from sqlalchemy import text

from src.config import settings

class QueryName(enum.Enum):
    """Enumeration with query filenames."""

```

```
HISTORICAL_DATA = "get_historical_states.sql"
```

```
def format_query(query):
    """Reads query text suitable for sqlalchemy from sql file and
    makes formatting with jinja"""

    with open(
        os.path.join(settings.BASE_DIR, "src", "app",
            "sql_service", "sql", query.value), "r"
    ) as f:
        try:
            data = f.read()
        except FileNotFoundError:
            raise ValueError("Query {} not found".format(query))

    return text(data)
```

A.6 backend/src/sql_service/sql/get_historical_states.sql

```
WITH d as (
    select
        "name",
        "timestamp",
        avg("value") as "avg_value"
    from states
    group by "name", "timestamp"
),
tms as (
    select
        distinct "timestamp"
    from states
)
select
    tms."timestamp" as "timestamp",
    (
        select
            d.name,
            d.avg_value
        from d
        where tms."timestamp" = d."timestamp"
        FOR JSON PATH, INCLUDE_NULL_VALUES
    ) as metrics
from tms
FOR JSON PATH
```

A.7 backend/src/app/serializers/response.py

```

import logging
from typing import Any
from datetime import date as date_cls

from pydantic import BaseModel, validator

logger = logging.getLogger('gunicorn.error')

def normalize_date(value):
    return date_cls(value // 10_000, (value % 10_000) // 100, value
% 100)

class HistoricalData(BaseModel):
    date: Any
    a: float
    hz: float
    kwh: float
    pf: float
    v: float
    w: float
    predicted: bool

    _normalize_date = validator('date',
allow_reuse=True)(normalize_date)

class ForecastingData(BaseModel):
    date: Any
    v: float
    predicted: bool

    _normalize_date = validator('date',
allow_reuse=True)(normalize_date)

```

A.8 backend/src/app/views/consumption.py

```

from datetime import datetime, date, timedelta
from typing import List, Union
from collections import ChainMap

from fastapi.requests import Request
from fastapi import APIRouter

from src.app.schemas.response import HistoricalData,
ForecastingData

```

```

from src.app.sql_service.query_formatter import format_query,
QueryName
from src.app.helpers.predictions import get_data_from_prediction

router = APIRouter()

@router.get("/", response_model=List[Union[HistoricalData,
ForecastingData]])
async def consumption(request: Request, start_date: datetime,
end_date: datetime):
    async with request.app.db.acquire() as conn:
        result = [
            {
                'date': x['timestamp'],
                'predicted': False,
                **dict(ChainMap(*x['metrics']))
            } for x in await (await conn.execute(
                format_query(QueryName.HISTORICAL_DATA)
            )).fetchall()]

        exists_dates = [x['date'] for x in result]

        def numeric_date(v):
            d = start_date + timedelta(days=v)
            return d.year * 10_000 + d.month * 100 + d.day

        missing_dates = [
            numeric_date(x)
            for x in range((end_date - start_date).days + 1)
            if numeric_date(x) not in exists_dates
        ]

        return sorted(
            result + get_data_from_prediction(missing_dates),
            key=lambda x: x['date']
        )

```

A.9 frontend/src/main.js

```

import Vue from "vue";

import App from "@App";
import router from "@router";

import Vuetify from "vuetify";
import VueApexCharts from "vue-apexcharts";

```

```

import "vuetify/dist/vuetify.min.css";
import "../styles/main.scss";

Vue.use(VueApexCharts);

Vue.use(Vuetify);

const vuetify = new Vuetify({});

Vue.component('apexchart', VueApexCharts)

new Vue({
  el: "#app",
  render: home => home(App),
  vuetify,
  router,
});

```

A.10 frontend/src/styles/main.scss

```

@import "../header_footer";
@import "../search.scss";
@import "../forms.scss";

.big-process-layout{
  height: 290px !important;
  width: 90%;
  margin-left: 5%;
}
.big-process-dialog, .big-process-dialog:active,
.big-process-dialog:hover{
  box-shadow: none;
}

```

A.11 frontend/src/pages/Home.vue

```

<template>
<div>
  <v-container>
    <ul>
      <li class="red--text subheading" v-for="err in errors">
        {{err}}
      </li>
    </ul>
    <v-container>

      <v-row
        class="fill-height"

```

```

        align-content="center"
        justify="center"
        v-show="true"
    >

<!-- start date -->
<v-col
    >

    <v-menu
        ref="menu_start"
        v-model="menu_start"

        transition="scale-transition"
        offset-y
        max-width="290px"
        min-width="auto"
    >

        <template v-slot:activator="{ on, attrs }">
            <v-text-field
                v-model="date_start"
                label="Start date"
                prepend-icon="mdi-calendar"
                readonly
                v-bind="attrs"
                v-on="on"
            ></v-text-field>
        </template>
        <v-date-picker
            v-model="date_start"
            no-title
            scrollable
            @click="$refs.menu_start.save(date_start)"
        >

            </v-date-picker>
    </v-menu>
</v-col>
<!-- end date -->
<v-col
    >

    <v-menu
        ref="menu_end"
        v-model="menu_end"
        transition="scale-transition"
        offset-y
        max-width="290px"
        min-width="auto"
    >

```

```

<template v-slot:activator="{ on, attrs }">
<v-text-field
  v-model="date_end"
  label="End date"
  prepend-icon="mdi-calendar"
  readonly
  v-bind="attrs"
  v-on="on"
></v-text-field>
</template>
<v-date-picker
  v-model="date_end"
  no-title
  scrollable
  @click="$refs.menu_end.save(date_end)"
>

  </v-date-picker>
</v-menu>
<!-- search button -->
</v-col>
<v-col
>
  <v-select
    :items="['All'].concat(items)"
    label="Chose metric"
    @input="display"
    class="select-pos"
    dense
    v-model="select"
    :disabled="series.length == 0"
  ></v-select>
</v-col>
<v-col
>
  <v-btn @click="search" large class="mt-1">
    <v-icon
      dark
    > search
    </v-icon>
    Start Search
  </v-btn>
</v-col>

</v-row>
<!-- content -->

<v-row

```

```

        class="filll-height pt-5"
        justify="center"
        align-content="center"
    >
    <v-col cols="6" v-show="loading">
        <v-progress-linear
            color="primary accent-4"
            indeterminate
            rounded
            height="6"
        ></v-progress-linear>
    </v-col>

    <v-col
        cols="12"
    >
        <apexchart ref="chart" :options="chartOptions"
:series="series"></apexchart>
    </v-col>
</v-row>

    </v-container>
</v-container>
</div>
</template>

<script>
import axios from "axios";
const { backendHost } = require("../../config");

export default {
  data: () => {
    return {
      menu_start: false,
      menu_end: false,

      loading: false,
      select: null,

      response: [],
      items: ['v', 'a', 'hz', 'kwh', 'pf', 'w'],

      date_start: "",
      date_end: "",
      errors: [],
      series: [],
      chartOptions: {
        chart: {

```

```

    type: 'area',
    zoom: {
      enabled: true,
      type: 'x',
      resetIcon: {
        offsetX: -10,
        offsetY: 0,
        fillColor: '#fff',
        strokeColor: '#37474F'
      },
      selection: {
        background: '#90CAF9',
        border: '#0D47A1'
      }
    },
  },
  plotOptions: {
    bar: {
      borderRadius: 0,
      dataLabels: {
        position: 'center',
      },
    },
  },
  dataLabels: {
    enabled: false,
    offsetX: 35,
    style: {
      fontSize: '12px',
      colors: ["#304758"]
    }
  },
},
xaxis: {
  position: 'bottom',
  axisBorder: {
    show: true
  },
  axisTicks: {
    show: true
  },
  crosshairs: {
    fill: {
      type: 'gradient',
      gradient: {
        colorFrom: '#D8E3F0',
        colorTo: '#BED1E6',
        stops: [0, 100],
        opacityFrom: 0.4,

```

```

        opacityTo: 0.5,
      }
    },
    tooltip: {
      enabled: true,

      intersect: true,
      shared: true
    }
  },
  yaxis: {
    axisBorder: {
      show: false
    },
    axisTicks: {
      show: false,
    },
    labels: {
      show: true,
    }
  },
  title: {
    text: 'Energy consumption',
    floating: true,
    offsetY: -5,
    align: 'center',
    style: {
      color: '#444',
      fontSize: '20',
    }
  }
},
}

},
methods: {
  display(metric=null) {
    if (this.items.includes(metric)) {
      this.series = [
        {
          name: metric,
          data: []
        }
      ]
    } else {
      this.series = this.items.map(item => {
        return {
          name: item,

```

```

        data: []
      }
    });
  }

  for (let i in this.response) {
    for (let j in this.series) {
      let value =
        (this.response[i][this.series[j].name]) ?
        this.response[i][this.series[j].name].toFixed(2) : null
      this.series[j].data.push(
        {
          x: this.response[i].predicted ?
          `${this.response[i].date} (Predicted)` : `${this.response[i].date}
          (Historical)`,
          y: value
        }
      )
    }
  }

  this.$refs.chart.updateSeries(this.series);
},
search(){
  this.loading = true;
  this.select = null;
  this.series = [];

  axios({
    method: "get",
    url:
    `${backendHost}/consumption/?start_date=${this.date_start}T00:00:00
    &end_date=${this.date_end}T00:00:00`,
    headers: {}
  }).then(res => {
    this.loading = false;
    this.response = res.data;
    this.errors = [];
    this.display();
  }).catch(err => {
    this.loading = false;
    console.log(err)
    if (err.data){
      this.errors = err.data
    } else {
      this.errors = ["Something went wrong!"]
    }
  })
}
}

```

```

    }
  }
</script>

```

A.12 frontend/src/router/index.js

```

import Vue from "vue";
import Router from "vue-router";

import routes from "@router/routes";

Vue.use(Router);

export default new Router({
  mode: "history",
  routes
});

```

A.13 infrastructure/main.tf

```

# #####
# TERRAFORM LIFECYCLE MANAGED
# All changes will be overwritten
#####
# Providers
# https://www.terraform.io/docs/configuration/terraform.html
# https://www.terraform.io/docs/configuration/providers.html
# https://www.terraform.io/docs/configuration/expressions.html
# https://www.terraform.io/docs/configuration/functions.html

terraform {
  required_version = "~> 1.0"

  backend "remote" {
    hostname      = "tfe.ecms.dev"
    organization = "Organisation"
    workspaces { name = "terraform-azure-ecms" }
  }
}

provider "random" {}
provider "azuread" {
  tenant_id      = var.project.arm.tenant.id
  client_id      = var.project.arm.client.id
  client_secret  = var.secret.arm.client.secret
}
provider "azurerm" {

```

```

    tenant_id      = var.project.arm.tenant.id
    client_id      = var.project.arm.client.id
    client_secret   = var.secret.arm.client.secret
    subscription_id = var.project.arm.subscription.id
    features {}
  }
  provider "azurerm" {
    alias      = "hub"
    tenant_id  = var.project.arm.tenant.id
    client_id  = var.project.arm.client.id
    client_secret = var.secret.arm.client.secret
    subscription_id = var.project.hub.subscription.id
    features {}
  }

# Variables
# https://www.terraform.io/docs/configuration/variables.html
# https://www.terraform.io/docs/configuration/types.html

variable "secret" {
  type = object({
    arm = object({
      client = object({ secret = string })
    })
    azdo = object({
      client = object({ secret = string })
      token  = string
    })
  })
}

variable "project" {
  type = object({
    name          = string
    environment    = object({ name = string })
    cidrs         = map(list(string))
    hub = object({
      subscription      = object({ id = string })
      resourcegroup     = object({ name = string })
      ddosprotectionplan = object({ name = string })
      virtualhub        = object({ name = string })
    })
    arm = object({
      tenant      = object({ id = string })
      client      = object({ id = string })
      subscription = object({ id = string })
    })
    azdo = object({

```

```

        client = object({ id = string })
    })
})
}

# Data Sources
# https://www.terraform.io/docs/configuration/data-sources.html

data "azuread_client_config" "default" {}
data "azuredevops_client_config" "default" {}
data "azurerm_client_config" "default" {}

data "azuredevops_project" "default" { name = local.azdo.project }
data "azuread_service_principal" "azuredevops" { application_id =
var.project.azdo.client.id }
data "azuread_application" "azuredevops" { application_id =
var.project.azdo.client.id }
data "azurerm_subscription" "default" {}

# Modules and Resources
# https://www.terraform.io/docs/configuration/modules.html
# https://www.terraform.io/docs/configuration/resources.html

```

A.14 infrastructure/variables.tf

```

# Locals
# https://www.terraform.io/docs/configuration/locals.html

locals {
  vnet = {
    enable = true
    subnets = {
      pe = {
        cidrs    = [cidrsubnet(var.project.cidrs["centralus-
default"][0], 3, 0)] # /27 0-32
        private = { endpoint = true, service = false }
        service = { endpoints = ["Microsoft.ContainerRegistry",
"Microsoft.KeyVault", "Microsoft.Storage", "Microsoft.Web"],
delegations = {}
      }
    }
  }
  webapp = {
    cidrs    = [cidrsubnet(var.project.cidrs["centralus-
default"][0], 3, 1)] # /27 32-64
    private = { endpoint = false, service = false }
    service = { endpoints = ["Microsoft.ContainerRegistry",
"Microsoft.KeyVault", "Microsoft.Storage", "Microsoft.Web"],
delegations = {
      "Microsoft.Web/serverFarms" = {

```

```

        actions =
["Microsoft.Network/virtualNetworks/subnets/action"]
        name      = "Microsoft.Web/serverFarms"
    }
}
}
app_gateway = {
    cidrs      = [cidrsubnet(var.project.cidrs["centralus-
default"][0], 3, 2)] # /27 64-96
    private = { endpoint = false, service = false }
    service = { endpoints = ["Microsoft.Web"], delegations = {}
}
}
}

asgs = {}
nsgs = {
    ag_security_nsg = {
        subnets = ["app_gateway"]
        rules = {
            AppGatewayV2 = {
                priority  = 210
                source = {
                    ports  = ["0-65535"]
                    asgs    = []
                    prefix = "Internet"
                }
                access      = "Allow"
                direction    = "Inbound"
                protocol     = "*"
                destination = {
                    ports  = ["65200-65535"]
                    asgs    = []
                    prefix = "*"
                }
            }
            HTTPSWSS = {
                priority  = 200
                access      = "Allow"
                source = {
                    ports  = ["0-65535"]
                    prefix = "Internet"
                    asgs    = []
                }
                direction = "Inbound"
                protocol   = "TCP"
                destination = {
                    ports  = ["443"]

```

```

        asgs    = []
        prefix = "*"
    }
}
}
}
}
}
}

locals {
    azdo = {
        serviceendpoint = false
        organization     = "ecms"
        project          = "ecms"
    }
    custom_tags = {}
}

# Variables
# https://www.terraform.io/docs/configuration/variables.html
# https://www.terraform.io/docs/configuration/types.html

variable "location" {
    type      = string
    default   = "Central US"
}

variable "external_ip" {
    type = object({
        tfe = string
        azdo_services = object({
            region = object({
                canada = string
            })
        })
    })
}

variable "dns_ssl" {
    type = object({
        hub = object({
            keyvault = object({
                name                = string
                subscription        = object({ id = string })
                resourcegroup       = object({ name = string })
                certificate = object({
                    ui = object({ name = string })
                })
            })
        })
    })
}

```

```

    })
    dnszone = object({
      ui          = object({ name = string })
      subscription = object({ id = string })
      resourcegroup = object({ name = string })
    })
  })
})
}

```

A.15 frontend/package.json

```

{
  "name": "app",
  "version": "1.0.0",
  "description": "A YouPlayerFrontend part",
  "author": "ZadoyanMK <konzamir@gmail.com>",
  "private": true,
  "scripts": {
    "dev": "webpack-dev-server --inline --progress --config
build/webpack.dev.conf.js",
    "start": "npm run dev",
    "lint": "eslint --ext .js,.vue src",
    "build": "node build/build.js"
  },
  "dependencies": {
    "apexcharts": "^3.29.0",
    "axios": "^0.19.0",
    "dotenv": "^8.2.0",
    "vue": "^2.5.2",
    "vue-apexcharts": "^1.6.2",
    "vue-router": "^3.0.1",
    "vue-youtube-embed": "^2.2.1",
    "vue3-apexcharts": "^1.4.1",
    "vuetify": "^2.3.14",
    "vuex": "^3.1.1"
  },
  "devDependencies": {
    "autoprefixer": "^7.1.2",
    "babel-core": "^6.22.1",
    "babel-eslint": "^8.2.1",
    "babel-helper-vue-jsx-merge-props": "^2.0.3",
    "babel-loader": "^7.1.1",
    "babel-plugin-syntax-jsx": "^6.18.0",
    "babel-plugin-transform-runtime": "^6.22.0",
    "babel-plugin-transform-vue-jsx": "^3.5.0",
    "babel-preset-env": "^1.3.2",
    "babel-preset-stage-2": "^6.22.0",
    "chalk": "^2.0.1",

```

```

"copy-webpack-plugin": "^4.0.1",
"css-loader": "^0.28.0",
"eslint": "^4.15.0",
"eslint-config-standard": "^10.2.1",
"eslint-friendly-formatter": "^3.0.0",
"eslint-loader": "^1.7.1",
"eslint-plugin-import": "^2.7.0",
"eslint-plugin-node": "^5.2.0",
"eslint-plugin-promise": "^3.4.0",
"eslint-plugin-standard": "^3.0.1",
"eslint-plugin-vue": "^4.0.0",
"extract-text-webpack-plugin": "^3.0.0",
"file-loader": "^1.1.4",
"friendly-errors-webpack-plugin": "^1.6.1",
"html-webpack-plugin": "^2.30.1",
"less": "^3.9.0",
"less-loader": "^5.0.0",
"node-notifier": "^5.1.2",
"node-sass": "^4.12.0",
"optimize-css-assets-webpack-plugin": "^3.2.0",
"ora": "^1.2.0",
"portfinder": "^1.0.13",
"postcss-import": "^11.0.0",
"postcss-loader": "^2.0.8",
"postcss-url": "^7.2.1",
"rimraf": "^2.6.0",
"sass": "^1.21.0",
"sass-loader": "^7.1.0",
"semver": "^5.3.0",
"shelljs": "^0.7.6",
"stylus": "^0.54.5",
"stylus-loader": "^3.0.2",
"uglifyjs-webpack-plugin": "^1.1.1",
"url-loader": "^0.5.8",
"vue-loader": "^13.3.0",
"vue-style-loader": "^3.0.1",
"vue-template-compiler": "^2.5.2",
"webpack": "^3.6.0",
"webpack-bundle-analyzer": "^2.9.0",
"webpack-dev-server": "^2.9.1",
"webpack-merge": "^4.1.0"
},
"engines": {
  "node": ">= 6.0.0",
  "npm": ">= 3.0.0"
},
"browserslist": [
  "> 1%",
  "last 2 versions",

```

```

    "not ie <= 8"
  ]
}

```

A.16 infrastructure/modules/automl/variables.tf

```

variable "training_subnet_address_space" {
  type          = list(string)
  description    = "Address space of the training subnet"
  default       = ["10.0.1.0/24"]
}

variable "aks_subnet_address_space" {
  type          = list(string)
  description    = "Address space of the aks subnet"
  default       = ["10.0.2.0/23"]
}

variable "ml_subnet_address_space" {
  type          = list(string)
  description    = "Address space of the ML workspace subnet"
  default       = ["10.0.0.0/24"]
}

```

A.17 functions/store_states_to_db.py

```

import os
import logging
import json

import azure.functions as func
import pyodbc

REQUIRED_COLUMNS = ['timestamp', 'name', 'value']

def main(msg: func.ServiceBusMessage):
    try:
        body = json.loads(msg.get_body().decode('utf-8'))
        d = body['timestamp']
        body['timestamp'] = d.year * 10_000 + d.month * 100 + d.day
        all_response_keys = list(body.keys())

        def clear_data(x):
            if x not in REQUIRED_COLUMNS:
                del body[x]
        map(clear_data, all_response_keys)

```

```

db_conn = pyodbc.connect(os.getenv("DB_CONN"))

db_conn.execute(
    "INSERT INTO [states](timestamp, name, value) VALUES
(%s, %s, %s)",
    *list(body.values())
)
except:
    logging.warning(f"Something wrong with data
{msg.get_body()}", exc_info=True)

```

A.18 functions/prepare_csv.py

```

import io
import os
import json

import pyodbc
import pandas as pd
from azure.storage.blob import BlobServiceClient

def main():
    db_conn = pyodbc.connect(os.getenv("DB_CONN"))

    rows_per_day = 24 * 60

    data = json.loads(''.join([x for x in db_conn.execute(
        f"""
        select
        name,
        (
            SELECT timestamp, value FROM [states]
            limit {rows_per_day}
            order by timestamp desc
            FOR JSON PATH, INCLUDE_NULL_VALUES
        ) as metrics
        from [states]
        group by name
        FOR JSON PATH, INCLUDE_NULL_VALUES
        """
    ).fetchall()])))

    for name, metrics in data:
        blob_service_client =
        BlobServiceClient.from_connection_string(
            os.getenv('STORAGE_CONNECTION_STRING')
        )
        files_count = len(blob_service_client.get_container_client(

```

```
        container=os.getenv('STORAGE_CONTAINER_NAME')
    ).list_blobs(starts_with=name))

df = pd.DataFrame(metrics, columns=['timestamp', 'value'])

f = io.BytesIO()
df.to_csv(f)
f.seek(0)

blob_client = blob_service_client.get_blob_client(
    container=os.getenv('STORAGE_CONTAINER_NAME'),
blob=f"{name}-{files_count}.csv"
)
blob_client.upload_blob(f.getvalue())
```

ДОДАТОК Б
Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Дипломна кваліфікаційна робота
на тему:
**Дослідження та програмна реалізація
методів інтелектуального аналізу даних
енергоспоживання**
Research and software implementation of energy
consumption intelligent data analysis methods

Виконав
ст. гр. КНТ-120м
Керівник
к.т.н. доц.

М. К. Задоян
А. В. Пархоменко

Рисунок Б.1 – Титульний слайд



Рисунок Б.2 – Огляд існуючої COA

Актуальність роботи

3



Засоби інтелектуального аналізу даних COA



Приклад нотифікацій для Home Assistant

Рисунок Б.3 – Огляд засобів ІА даних енергоспоживання

Мета та завдання роботи

4

- ❑ Мета роботи є дослідження та практична реалізація методів і засобів інтелектуального аналізу даних для підвищення енергоефективності системи офісної автоматизації.
- ❑ Завдання:
 - ✓ дослідити методи та засоби ІА даних енергоспоживання;
 - ✓ проаналізувати вимоги до розроблюваної системи;
 - ✓ розробити архітектуру системи;
 - ✓ порівняти та обрати СКБД;
 - ✓ обрати інструментарій розробки;
 - ✓ розробити програмний продукт;
 - ✓ описати керівництво з використання програмного продукту.

Рисунок Б.4 – Мета роботи

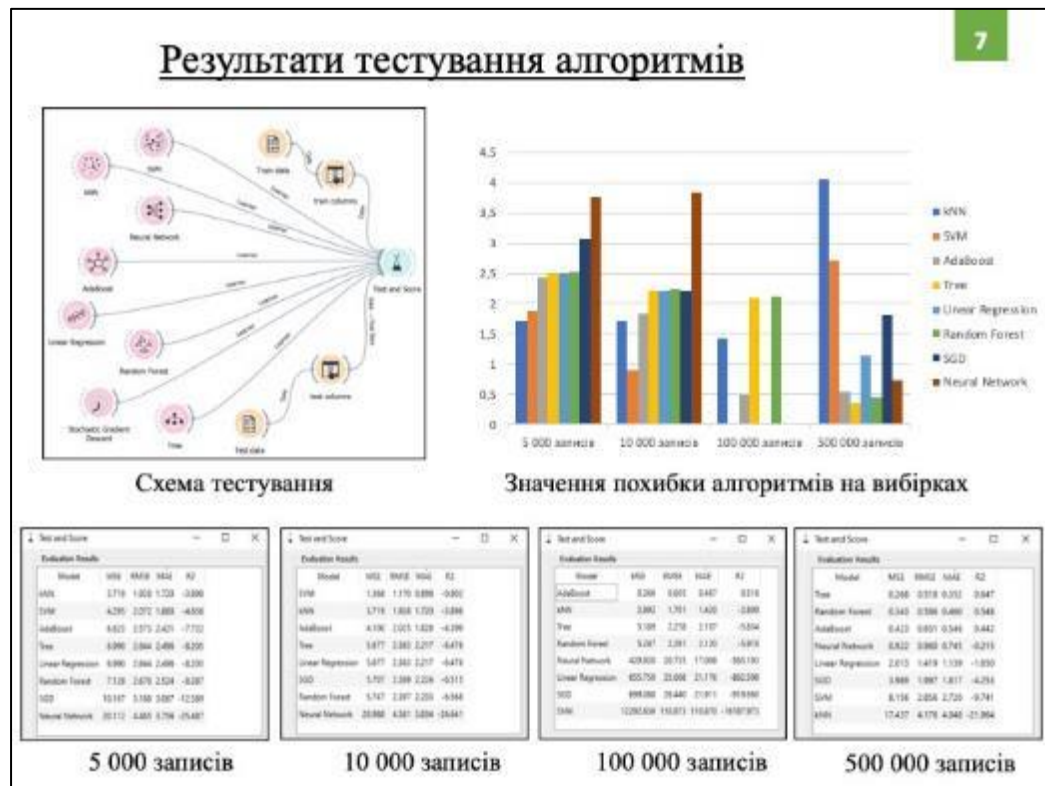


Рисунок Б.7 – Результати тестування алгоритмів



Рисунок Б.8 – Аналіз функціональних вимог

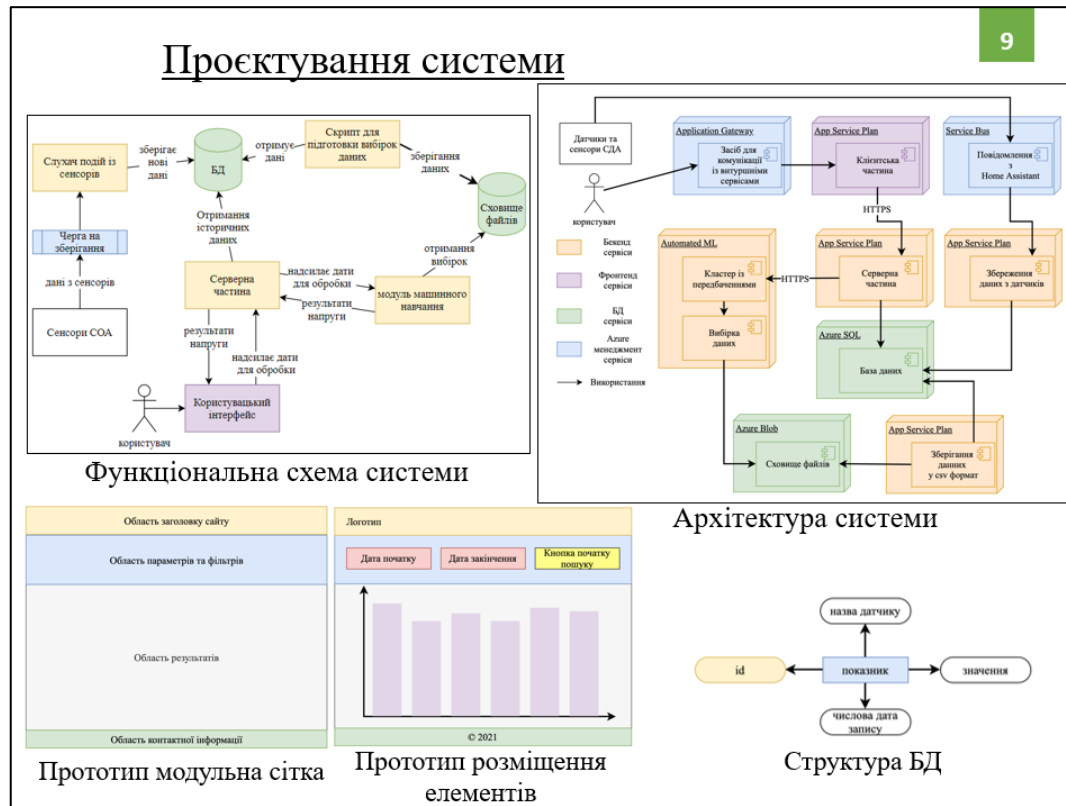


Рисунок Б.9 – Проектування системи

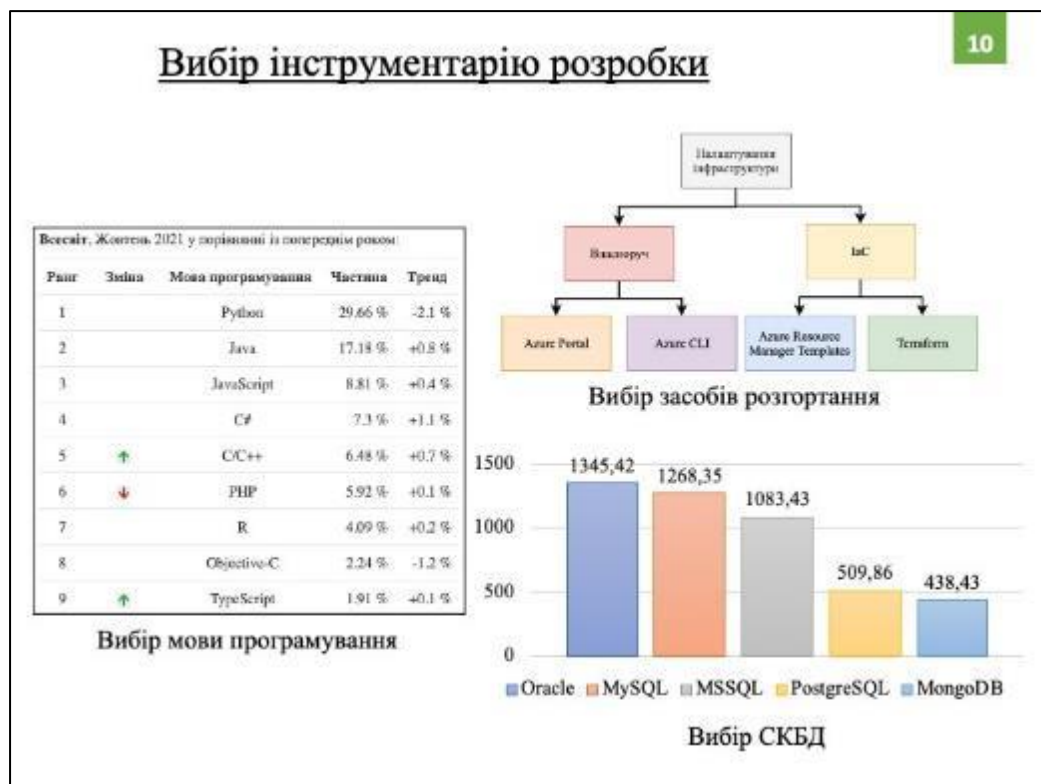


Рисунок Б.10 – Вибір інструментарію розробки



Рисунок Б.11 – Конструювання серверної та клієнтської частин

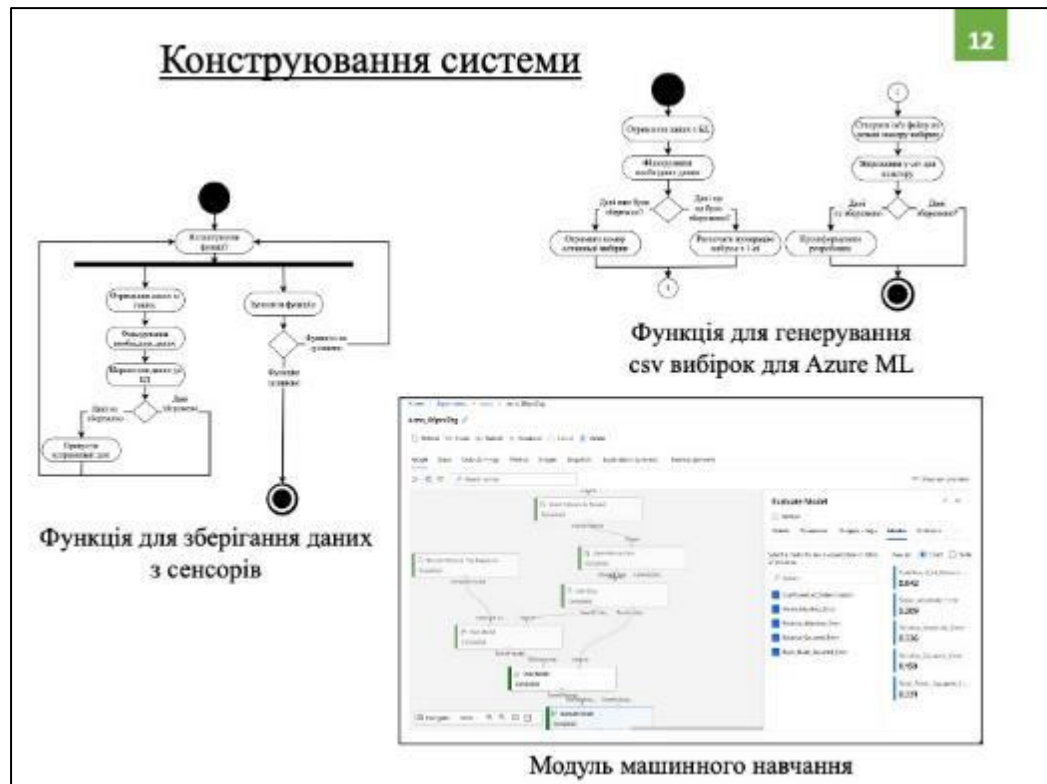


Рисунок Б.12 – Конструювання функцій та модулю машинного навчання

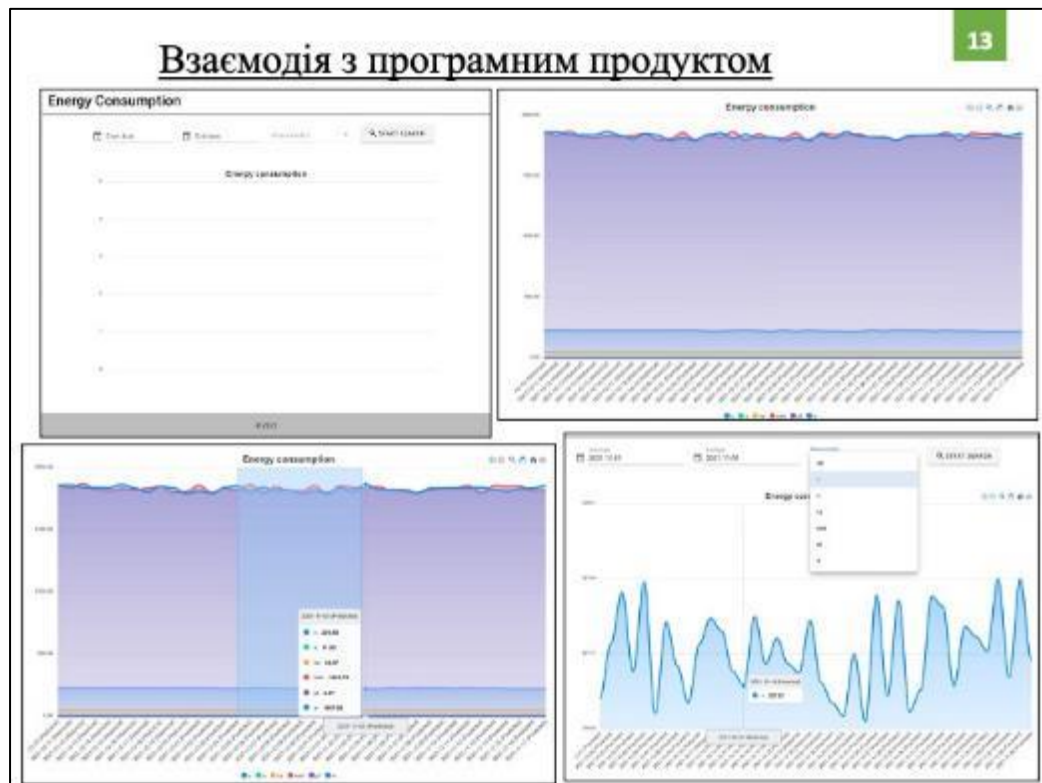


Рисунок Б.13 – Взаємодія з системою

Висновки

- В роботі створено програмний продукт, що дозволяє аналізувати дані енергоспоживання з СОА та здійснювати прогнозування роботи СОА на конкретний день, тиждень або місяць.
- Наукова новизна роботи полягає в тому, що отримав подальший розвиток алгоритм Boosted Decision Tree за рахунок удосконалення етапів підготовки та виведення даних. Пропонована реалізація представляє собою окремий сервіс для надання даних щодо прогнозування енергоспоживання в СОА та, на відміну від існуючих рішень, забезпечує програмний інтерфейс з використанням TCP/IP засобів для взаємодії з ним.
- практична цінність роботи полягає в тому, що за рахунок використання розробленої програмної системи:
 - підвищиться ефективність планування бюджету компанії та заходів щодо безпечного використання техніки та обладнання за рахунок прогнозування даних енергоспоживання;
 - зменшиться час обчислень та потрібні ресурси для обробки великих об'ємів даних за рахунок використання оптимізованого алгоритму Boosted Decision Tree.
- В майбутньому даний програмний засіб можна покращити за рахунок додаткового функціоналу для візуалізації результатів роботи.
- Результати роботи було представлено в тезах доповіді 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS2021), Cracow, Poland, 22-25 September, 2021.

Рисунок Б.14 – Висновки



Рисунок Б.15 – Фінальний слайд