МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2022 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ОБРОБКИ ІНФОРМАЦІЇ ПО ЯКОСТІ ФІЛЬТРАЦІЇ ВОДИ

Спеціальність 122 «Комп'ютерні науки»

122 – МКР – 601.21610318

Студент _____ І.А.Пушняк

«15» лютого 2022 р.

Консультант _____ І.О.Калініна

Керівник: к.т.н., доцент

«15» лютого 2022 р.

Миколаїв – 2022

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

IoT	–	Інтернет речей
MVC	–	Архітектурний шаблон
SASS	–	Препроцессор CSS
HTML	–	Язык гипертекстовой разметки
XML	–	Расширяемый язык разметки
СУБД		Система управления базами данных
БД	–	База даних
БЗ	–	База знань
HTTP	–	Протокол прикладного уровня передачи данных
ІС	–	Інтелектуальна система
ІІС	–	Інтелектуально – інформаційна система

ВСТУП

Актуальність роботи:

Розробка інтелектуальної системи обробки інформації по якості фільтрації води, пов’язана с актуальною проблемою погіршення якості питної води в усьому світі.

Питна вода – це харчовий продукт, вироблений системою водопостачання для щоденного споживання. У зв'язку з цим до неї ставлять достатньо високі вимоги відносно її безпеки і якості для здоров'я людини. Виконувати свою гігієнічну роль вода може лише тоді, коли вона якісна щодо органолептичних, хімічних та бактеріологічних властивостей. В іншому разі - неякісна або забруднена вода може спричинити низку інфекційних хвороб.

Тобто завдання полягає у тому, щоб створити інтелектуальну систему, яка зможе зберігати інформацію людей, котрі користуються фільтрами для води. На основі зібраних даних проводити аналіз по якості фільтрації води для покращення подальшої фільтрації води.

Метою роботи є створення інтелектуальної системи для обробки інформації по якості фільтрації води.

Об’єктом дослідження – інтелектуальна система, як спосіб збереження, відображення, аналіз даних по якості фільтрації води.

Предметом дослідження є набір даних по якості фільтрації води, отриманих від клієнтів, що використовують фільтри для води.

Для досягнення зазначеної мети необхідно виконати наступні **завдання**:

1. Розглянути основні проблеми, через які падає якість фільтрації води;

2. розглянути інтелектуальні системи та методи обробки інформації та обрати найбільш оптимальний для вирішення поставленої задачі;

3. Розглянути найбільш популярні IoT платформи, виявити їх переваги та недоліки;

4. Створити власну систему, відштовхуючись від потреб та аналізу досліджених IoT платформи ;

Практичне значення отриманих результатів: Результатом роботи є створення інтелектуальної системи, яка допоможе обробляти інформацію по якості фільтрації води для подальшого її покращення .

1 АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРИ ТА ПАТЕНТНОЇ ІНФОРМАЦІЇ

1.1 Типи фільтрів для води та їх фільтрація

Фільтр для води - пристрій для очищення води від механічних, нерозчинних частинок, домішок, хлору та його похідних, а також від вірусів, бактерій, важких металів тощо. Фільтри для води видаляють з води небажані домішки, такі як осад, смак і запах, жорсткість і бактерії, щоб вода була кращої якості. Побутові фільтри, що використовуються для отримання питної води, умовно можна розділити на три категорії - найпростіші побутові фільтри, середнього ступеня очищення та побутові фільтри вищого ступеня очищення: від виробництва смачнішої питної води до більш спеціалізованих застосувань.

Виділяють основні типи фільтрів для води:

1. Фільтри-глечики;
2. Фільтри грубої очистки;
3. Фільтри попереднього очищення;
4. Магістральні фільтри;
5. Фільтри зворотного осмосу;
6. Фільтри під мийку;

Фільтри-глечики

Рис.1.1 Фільтр-глечик

Глечик – це прозора пластикова місткість 1,5-4 літрів з промальованою шкалою об'єму. Деякі моделі оснащені кришками з клапанами, що запобігає попаданню пилу та мікроорганізмів при наповненні чаші.

Половину глека займає вирва, на дні якої розташований картридж для очищення. Об'єм вирви також варіюється, для очищення повного об'єму глечика воду у вирву набирають двічі.

Якість очищеної води за допомогою глечика варіюється і залежить від жорсткості рідини, що надходить, і тривалості використання картриджа. Перевага таких фільтрів – невисока вартість, це бюджетний варіант. Вони розраховані на використання в будинку, де мешкають 1-3 особи. Якщо кількість членів сім'ї більша, то часта заміна картриджів збільшить витрати на очищення. Глечики прості у використанні, легко транспортуються, тому їх використовують і вдома, і на дачних ділянках. Невисокий ступінь очищення

в порівнянні з іншими системами фільтрації та часта зміна картриджів – це головні недоліки такого методу.

Фільтри грубої очистки

Рис.1.2 Фільтр грубої очистки води

Фільтри грубої очистки встановлюють на трубах видалення з рідини нерозчинних великих частинок. Актуально використання таких приладів для заміських будинків, де вода надходить безпосередньо зі свердловини, вони затримують бруд, пісок, глину, органічні сполуки. У квартирах експлуатують з метою очищення від іржі, вапняного нальоту, окалини.

Усі фільтри грубої очистки з однаковим принципом роботи. Процес очищення простий: вода проходить через сітку, яка перешкоджає проходженню частинок більших за розміром, ніж комірки. Залежно від моделі варіюється матеріал корпусу (метал, поліпропілен), монтаж та внутрішній вміст.

Фільтри попереднього очищення

Рис.1.3 Фільтр попереднього очищення

Фільтри попереднього очищення призначені для видалення великих частинок, піску та іржі. Такі прилади призначені для механічного очищення та не видаляють з рідини важкі метали, хлор та мікроорганізми. Попереднє очищення води перед подачею в пральну та посудомийну машини, нагрівальні та сантехнічні прилади – це основне завдання фільтрів такого типу. Установка приладів відбувається в холодний і гарячий трубопроводи після крана, що перекриває, але до лічильників[20].

Металеві сітки кожні 1-2 місяці знімають, промивають накопичені частинки іржі, піску та встановлюють на місце. Додатково деякі моделі обладнані системою промивання сітки, що не вимагає проведення демонтажу пристрою з перекриттям води, зустрічається і автоматичне промивання сітки від забруднень з виведенням у каналізацію. Певні моделі обладнані клапанами, що регулюють тиск рідини. Манометри – пристрої вимірювання тиску таких моделях встановлюють до фільтра і після, у своїй занижений

показник після пристрою свідчить про необхідність промивання сітки. Розміри пристрою відповідають діаметру труб.

Видалення лише великих частинок сміття з води – головний недолік пристроїв попереднього очищення.

Магістральні фільтри

Рис.1.4 Магістральні фільтри

Фільтр, який безпосередньо підключають до водопровідної труби, називають магістральним. Він складається із сталевої або пластикової розбірної колби, всередині якої встановлено очисний елемент[21].

Пропускна спроможність становить 20-50 літрів за хвилину. Залежно від встановленого змінного картриджа, магістральні пристрої очищають рідину від великих частинок, домішок, хлору, покращують смак і пом'якшують її.

Магістральний фільтр із зернистим завантаженням використовують для комплексного усунення хімічних та біологічних забруднень, а також він виступає пом'якшувачем води. Фільтруючий шар наповнений зернами

кварцового піску, керамзиту, гірських порід, полімерів, антрациту, мармурової крихти різними за розміром.

Фільтри зворотного осмосу

Рис.1.5 Фільтр зворотного осмосу

Системи зворотного осмосу – це устрою, вода з яких очищена від домішок на 99%. За рахунок наявності багатоступінчастої системи очищення рідини – на виході виходить вода, аналогічна за своїми параметрами бутильованої.

Процес роботи фільтра складається з наступних етапів:

1. Попереднє очищення;
2. Проходження через мембрану;
3. Заповнення бака;
4. Надходження очищеної води у кран.

Встановлюють систему зворотного осмосу під миттям. Але можлива й інша установка з проведенням додаткових шлангів для подачі та виведення чистої та брудної води. Продуктивність системи залежить від тиску, осмос працює за 2-6 барів. Якщо показники відрізняються від заявлених, тиск підвищують за рахунок установки насоса або знижують.

До недоліків такої системи відносять низьку швидкість очищення і велику витрату рідини (2/3 обсягу йде з домішками в каналізацію)[22].

Фільтри під мийку

Рис.1.6 Фільтр під мийку

Фільтри під миття представляють єдину систему, що складається з 3-5 фільтруючих секцій. Вода, що надходить із водопроводу, проходить кілька етапів очищення. Кожна секція – це пластикова, сталева чи склонаповнена колба з картриджем. Підключається така конструкція до холодного водопроводу під миттям із виведенням очищеної води через окремий кран[24].

До фільтрів під миття відносять:

1. Проточні;
2. Зворотного осмосу.

Проточний фільтр складається з таких модулів:

1. Механічного очищення від домішок піску, іржі, мулу;
2. Вугільного сорбенту, що видаляє хлор, солі, органічні сполуки, важкі метали;
3. Іонообмінних смол, які пом'якшують воду, видаляють залізо, магній та калій[23].

Проточні пристрої мають високу пропускну здатність і економічність у споживанні води на відміну від осмотичних моделей. Системи зворотного осмосу додатково оснащені мембраною, резервуаром для збирання та зберігання очищеної води, а також мінералізатором та біокерамічним картриджем.

Недоліки – це висока ціна обладнання та стаціонарна установка.

1.2 Інтелектуальна система

Інтелектуальні системи (від латів. intellectus - розум, розум) - це технічні або програмні системи, які реалізують деякі риси людського інтелекту, що дають змогу долати важкі завдання, вирішення яких людиною в даний час неможливе. ІС є основним продуктом досліджень штучного інтелекту. Структура інтелектуальної системи включає три основні блоки – базу знань, вирішувач та інтелектуальний інтерфейс.

Комп'ютерні системи оточують нас усюди і є найважливішим компонентом у функціонуванні бізнесу, урядових та військових організацій, закладів охорони здоров'я, програм навчання тощо. Ефективність комп'ютерних систем залежить від можливостей доступу, обробки та аналізу інформації. Для повної співпраці з користувачем комп'ютерні системи повинні мати зародки інтелекту, щоб кваліфіковано зберігати та обробляти

великі обсяги інформації, використовуючи аналоги природних засобів комунікації.

Штучний інтелект (інтелектуальна система) – це концепція, що дозволяє комп'ютерам робити такі речі, які люди виглядають розумно. Область застосування: докази теорем, ігри, розпізнавання образів, прийняття рішень, адаптивне програмування, твір машинної музики, обробка даних природною мовою, мережі (нейросети), що навчаються, вербальні концептуальні системи навчання і т.д.

Комп'ютерні технології для інтелектуальних обчислень переживають розквіт. Зараз відбувається стрімке зростання кількості програмних продуктів, що використовують нові технології, а також типів завдань, де їх застосування дає значний економічний ефект. Елементи автоматичної обробки та аналізу даних, які називають Data Mining (видобуток знань), стають невід'ємною частиною концепції електронних сховищ даних та організації інтелектуальних обчислень.

Хоча інструментарій інтелектуального аналізу та звільняє користувача від можливих складнощів у застосуванні статистичних методів, він все-таки вимагає від нього розуміння роботи та алгоритмів, на яких він базується. Крім цього, технологія знаходження нового знання бази даних не може дати відповіді на не задані питання. Вона не замінює аналітиків чи менеджерів, а дає їм сучасний, потужний інструмент для покращення виконуваної роботи.

Сучасні технології інтелектуального аналізу переробляють інформацію з метою автоматичного пошуку шаблонів, притаманних якихось фрагментів неоднорідних багатовимірних даних. Тяжкість формулювання гіпотез та виявлення незвичайних шаблонів переведена з людини на комп'ютер.

Ключем до успішного застосування методів інтелектуальних обчислень служить не просто вибір алгоритму, а майстерність людини, що створює модель та можливості програми, що моделює процес. Існують дві сторони

успіху. По-перше - чітке і ясне формулювання завдання, що підлягає вирішенню. По-друге - використання правильних даних та методів. Після вибору даних із усіх доступних джерел (або отримання даних із зовнішніх джерел) необхідно їх перетворити чи згрупувати у певному порядку. Чим більше аналітик може «грати» з даними, будувати моделі, оцінювати результати, тим кращим може бути результат. Робота з даними стає ефективнішою, при інтеграції наступних компонентів: візуалізації, графічного інструментарію, засобів формування запитів, оперативної аналітичної обробки, що дозволяє зрозуміти дані та інтерпретувати результати.

Виділяють такі алгоритми інтелектуальних обчислень:

1. Нейронні сіті;
2. Дерева рішень;
3. Системи роздумів з урахуванням аналогічних випадків;
4. Алгоритми визначення асоціацій та послідовностей;
5. Нечітка логіка;
6. Генетичні алгоритми;
7. Еволюційне програмування;
8. Візуалізація даних;
9. Комбінація.

На думку фахівців, у недалекій перспективі інтелектуальні системи відіграватимуть провідну роль у всіх фазах проектування, розробки, виробництва, розподілу, продажу, підтримки та надання послуг. Їхня технологія, отримавши комерційне поширення, забезпечить революційний прорив в інтеграції додатків із готових інтелектуально – взаємодіючих модулів.

1.3 Види інтелектуальних систем

Виділяють 5 видів інтелектуальних систем:

1. Інтелектуальна інформаційна система;
2. Експертна система;
3. Розрахунково-логічні системи;
4. Гібридна інтелектуальна система;
5. Рефлекторна інтелектуальна система.

1. **Інтелектуальна інформаційна система** (ІІС, англ. intelligent system) - різновид інтелектуальної системи, один із видів інформаційних систем, іноді ІІС називають системою, заснованою на знаннях. ІІС є комплексом програмних, лінгвістичних і логіко-математичних засобів для реалізації основного завдання: здійснення підтримки діяльності людини, наприклад можливість пошуку інформації в режимі просунутого діалогу природною мовою.

2. **Експертна система** (ЕС, expert system) - комп'ютерна програма, здатна частково замінити спеціаліста-експерта у вирішенні проблемної ситуації. Сучасні ЕС почали розроблятися дослідниками штучного інтелекту у 1970-х роках, а у 1980-х отримали комерційне підкріплення. Предтечі експертних систем були запропоновані в 1832 С. Н. Корсаковим, який створив механічні пристрої, так звані «інтелектуальні машини», що дозволяли знаходити рішення за заданими умовами, наприклад визначати найбільш підходящі ліки за симптомами захворювання, що спостерігаються у пацієнта.

В інформатиці експертні системи розглядаються спільно з базами знань як моделі поведінки експертів у певній галузі знань з використанням процедур логічного висновку та прийняття рішень, а бази знань – як сукупність фактів та правил логічного висновку у обраній предметній галузі діяльності.

Подібні дії виконує такий програмний інструмент як Майстер (Wizard). Майстри застосовуються як у системних програмах, так і в прикладних для спрощення інтерактивного спілкування з користувачем (наприклад, при встановленні ПЗ). Головна відмінність майстрів від ЕС – відсутність бази знань – всі дії жорстко запрограмовані. Це просто набір форм для заповнення користувачем.

Інші подібні програми – пошукові або довідкові (енциклопедичні) системи. За запитом користувача вони надають найбільш відповідні (релевантні) розділи бази статей (уявлення про об'єкти знань, їх віртуальну модель).

3. До **розрахунково-логічним системам** відносять системи, здатні вирішувати управлінські та проектні завдання за декларативними описами умов. При цьому користувач має можливість контролювати у режимі діалогу всі стадії обчислювального процесу. Дані системи здатні автоматично будувати математичну модель задачі та автоматично синтезувати обчислювальні алгоритми формулювання завдання. Ці властивості реалізуються завдяки наявності бази знань у вигляді функціональної семантичної мережі та компонентів дедуктивного виведення та планування.

4. Під **гібридною інтелектуальною системою** прийнято розуміти систему, в якій для вирішення задачі використовується більше одного методу імітації інтелектуальної діяльності. Таким чином, ГІС — це сукупність:

1. аналітичних моделей
2. експертних систем
3. штучних нейронних мереж
4. нечітких систем
5. генетичних алгоритмі
6. імітаційних статистичних моделей

Міждисциплінарний напрямок «гібридні інтелектуальні системи» об'єднує вчених та фахівців, що досліджують застосовність не одного, а кількох методів, як правило, з різних класів, до вирішення завдань управління та проектування.

5. **Рефлекторна система** - це система, яка формує відповідні реакції на різні комбінації вхідних впливів, що виробляються спеціальними алгоритмами. Алгоритм забезпечує вибір найбільш ймовірної реакції інтелектуальної системи на безліч вхідних впливів, при відомих ймовірностях вибору реакції на кожну вхідну дію, а також деякі комбінації вхідних впливів. Це завдання подібне до тієї, яку реалізують перцептрони[28].

По комбінації впливів на рецептори формуються числові характеристики рефлекторів через проміжний шар. Зв'язки між шарами забезпечують передачу деякої величини (імпульсу), від елементів одного шару, елементів іншого. Якщо сумарна величина (сумарний імпульс) на вході деякого елемента перевищує його граничне значення, він передає своє значення (свій імпульс) на елементи наступного шару. По суті, кожен із елементів є моделлю нейрона.

На відміну від перцептронів, рефлекторний алгоритм безпосередньо розраховує адекватну вхідним впливам реакцію інтелектуальної системи. Адекватність реакції базується на припущенні, що закони несилової взаємодії однакові на будь-яких рівнях уявлення взаємодіючих систем: чи то живі, чи неживі об'єкти.

Рефлекторні програмні системи застосовуються до наступних завдань: природно-мовний доступ до баз даних; оцінки інвестиційних пропозицій; оцінки та прогнозування впливу шкідливих речовин на здоров'я населення; прогнозування результатів спортивних ігор.

1.4 Класифікація задач, розв'язуваних ІС

1. **Інтерпретація даних.** Це одне із традиційних завдань для експертних систем. Під інтерпретацією розуміється процес визначення змісту даних, результати якого мають бути узгодженими та коректними. Зазвичай передбачається багатоваріантний аналіз даних [31].

2. **Діагностика.** Під діагностикою розуміється процес співвідношення об'єкта з деяким класом об'єктів та/або виявлення несправності в деякій системі. Несправність – це відхилення від норми. Таке трактування дозволяє з єдиних теоретичних позицій розглядати і несправність обладнання в технічних системах, захворювання живих організмів, і всілякі природні аномалії. Важливою специфікою тут є необхідність розуміння функціональної структури («анатомії») діагностуючої системи.

3. **Моніторинг.** Основне завдання моніторингу - безперервна інтерпретація даних у реальному масштабі часу та сигналізація про вихід тих чи інших параметрів за допустимі межі. Головні проблеми — «перепустка» тривожної ситуації та інверсне завдання «хибного» спрацьовування. Складність цих проблем у розмитості симптомів тривожних ситуацій та необхідність урахування тимчасового контексту.

4. **Проектування.** Проектування полягає у підготовці специфікацій створення «об'єктів» із заздалегідь певними властивостями. Під специфікацією розуміється весь набір необхідних документів-креслення, пояснювальна записка і т.д. Основні проблеми тут – отримання чіткого структурного опису знань про об'єкт та проблема «сліду». Для організації ефективного проектування та ще більшою мірою перепроектування необхідно формувати як самі проектні рішення, а й мотиви їх прийняття. Таким чином, у завданнях проектування тісно пов'язуються два основні процеси, що виконуються в рамках відповідної ЕС: процес виведення рішення та процес пояснення[30].

5. **Прогнозування.** Прогнозування дозволяє передбачати наслідки деяких подій чи явищ на підставі аналізу наявних даних. Прогнозуючі системи логічно виводять можливі наслідки із заданих ситуацій. У системі прогнозування зазвичай використовується параметрична динамічна модель, в якій значення параметрів «підганяються» під задану ситуацію. Наслідки, що виводяться з цієї моделі, становлять основу для прогнозів з імовірнісними оцінками.

6. **Планування**. Під плануванням розуміється знаходження планів дій, які стосуються об'єктів, здатних виконувати деякі функції. У таких ЕС використовуються моделі поведінки реальних об'єктів для того, щоб логічно вивести наслідки планованої діяльності.

7. **Навчання.** Під навчанням розуміється використання комп'ютера на навчання якійсь дисципліни чи предмету. Системи навчання діагностують помилки щодо будь-якої дисципліни з допомогою ЕОМ і підказують правильні рішення. Вони акумулюють знання про гіпотетичного «учня» та його характерні помилки, потім у роботі вони здатні діагностувати слабкості в пізнаннях учнів і знаходити відповідні засоби для їх ліквідації. Крім того, вони планують акт спілкування з учнем залежно від успіхів учня з метою передачі знань.

Нейронні мережі не програмуються у звичному значенні цього слова, вони навчаються. Можливість навчання - одна з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно навчання полягає у знаходженні коефіцієнтів зв'язків між нейронами. У процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними та вихідними, а також виконувати узагальнення. Це означає, що, у разі успішного навчання, мережа зможе повернути правильний результат на підставі даних, які були відсутні в навчальній вибірці.

8. **Управління.** Під управлінням розуміється функція організованої системи, що підтримує певний режим діяльності. Такі ЕС здійснюють управління поведінкою складних систем відповідно до заданих специфікацій.

9. **Підтримка ухвалення рішень.** Підтримка прийняття рішення - це сукупність процедур, що забезпечує особа, яка приймає рішення, необхідною інформацією та рекомендаціями, що полегшують процес прийняття рішення. Ці ЕС допомагають фахівцям вибрати та/або сформувати потрібну альтернативу серед безлічі виборів при прийнятті відповідальних рішень[29].

У випадку всі системи, засновані на знаннях, можна поділити на системи, вирішальні завдання аналізу, і системи, вирішальні завдання синтезу. Основна відмінність завдань аналізу від задач синтезу полягає в тому, що якщо в задачах аналізу безліч рішень може бути перераховано і включено в систему, то в задачах синтезу безліч рішень потенційно не обмежена і будується з рішень компонентів або під-проблем. Завданнями аналізу є: інтерпретація даних, діагностика, підтримка ухвалення рішення; до завдань синтезу відносяться проектування, планування, керування. Комбіновані: навчання, моніторинг, прогнозування.

Висновки до розділу 1

Розглянуто типи фільтрів для води. Обрано підходящий тип, а саме фільтр зворотного осмосу. Даний тип фільтру був обрано спеціально для досягнення поставленої мети, тому як дані для ІС будуть поступати через спеціальний контролер, який кріпиться до основи фільтру.

Проаналізувавши основні види інтелектуальних систем, виявивши їх спеціалізацію та направленість, була обрана найбільш підходяща система, а саме інтелектуальна інформаційна система(ІІС). Структура інтелектуальної системи включає три основні блоки – базу знань, вирішувач та інтелектуальний інтерфейс.

Для досягнення поставленої мети роботи буде використовуватися – моніторинг(інтерпретація) даних, як різновид класифікацію задач, що вирішують ІС.

2 МЕТОДИ ВІЗУАЛІЗАЦІЇ ТА ПЕРЕДАЧИ ДАННИХ. IOT ПЛАТФОРМА

2.1 IoT платформа. Різновиди та основні принципи роботи

Для досягнення поставленої мети, а саме реалізації інтелектуальної системи, необхідно розглянути вже готові рішення від різних компаній та на їх основі створити власну систему.

Інтернет речей (IoT) описує мережу фізичних об’єктів — «речей», які вбудовані з датчиками, програмним забезпеченням та іншими технологіями з метою підключення та обміну даними з іншими пристроями та системами через Інтернет. Ці пристрої варіюються від звичайних побутових предметів до складних промислових інструментів.

Платформи IoT можна визначити як багаторівневу технологію, яка використовується для керування та автоматизації підключених пристроїв. Крім того, ці платформи IoT корисні для розміщення фізичних об’єктів на онлайн-платформі. Ця платформа запропонує вам найкращі послуги для підключення пристроїв для зв’язку між машиною та машиною.

Різновиди IoT платформ

Google Cloud Platform

Рис.2.1. IoT Google Platform

Особливості Google Cloud Platform:

1. Потужне машинне навчання;
2. Надає бізнес-аналітику в режимі реального часу для пристроїв;
3. Потужні можливості штучного інтелекту;
4. Величезна підтримка геоданих;
5. Можливість прискорити бізнес-процеси;
6. Можлисвість збільшити швидкість роботи пристрою;
7. Використовує хмарні сервіси для мінімізації витрат;

Particle

Рис. 2.2. IoT Particle

Particle пропонує різні рішення IoT для підключення, обладнання і додатків. Для кращого підключення він пропонує три основні продукти: Wi-Fi, Cellular і Mesh. Будучи програмним забезпеченням IoT, пропонує хмару пристроїв, ОС, механізм правил IoT та інструменти розробника.

Особливості Particle:

1. Надійна інфраструктура;
2. Платформу може використовувати будь-хто, для її використання не потрібен експерт;
3. Має захищену брандмауером хмару Google Cloud або Microsoft Azure;
4. Можливість легкого інтегрування з чим завгодно за допомогою REST API.

IBM Watson IoT

Рис.2.3. IoT IBM Watson IoT

IBM Watson IoT — це платформа розробки на основі Pass, яку пропонує IBM. Платформа допоможе вам у зборі та дослідженні даних про пристрої обладнання та пошук кращого варіанту прийняття рішення. Крім того, платформа IBM Watson IoT дозволить вам оптимізувати ваші операції та ресурси.

Особливості IBM Watson IoT

1. Пропонує архітектуру відкритого контейнера, яка підтримуватиме міграцію робочих навантажень у хмари;
2. Допоможе створити та підключити пристрої з додатками в хмарі без будь-яких зусиль;
3. Наявна інформаційна панель для покращення візуалізації;
4. Наявність службу аналітики як додаткова послуга;

Amazon AWS IoT Core

Рис.2.4. IoT Amazon AWS IoT Core

AWS IoT — це керований хмарний сервіс. Даний сервіс надаж можливість пристроям підключатися до хмари та спілкуватися з іншими пристроями та хмарними програмами. Наявність підтримки MQTT, HTTP протоколу[24].

Особливості Amazon AWS IoT Core:

1. Обробка великої кількості даних;
2. Надійна та безпечна платформа для маршрутизації повідомлень до кінцевих точок AWS та інших пристроїв;
3. Безпечний доступ до пристроїв;
4. Використовуючи цю платформу, є можливість користуватися іншими службами AWS, такими як Amazon Kinesis, Lambda і Amazon QuickSight тощо.

Microsoft Azure IoT Suite

Рис.2.5. IoT Microsoft Azure

Microsoft Azure IoT — це хмарна платформа з відкритим вихідним кодо. Ця платформа розроблена для різноманітних галузевих потреб. Є можливість обслуговувати різні галузі промисловості від виробництва до транспорту та роздрібної торгівлі. Microsoft Azure надає різні рішення для віддаленого моніторингу[25].

Особливості пакету Microsoft Azure IoT Suite:

1. Є підтримка миттєвого реєстру пристроїв, для створення унікальний ідентифікатор для кожного пристрою;
2. Наявність хмарної інформаційної панелі, яка забезпечує миттєвий доступ до даних з різних пристроїв і програм;
3. Швидка передача даних для аналітики власних пристроїв;
4. Наявна можливість віддаленого моніторингу для відстеження всіх пристроїв і програм;

Роблячи висновки, головними недоліками переглянутих IoT платформ є: обмежений доступ для додавання пристроїв; необхідність в ручному додаванні пристроїв; ліміт на збереження даних; ціна послуг.

2.2 Способи візуалізації та представлення даних

Обсяги даних, із якими потрібно працювати, постійно збільшуються. І чим більше інформації, то складніше її обробляти. Саме тому потрібно визначитися с можливістю візуалізації даних[26].

amCharts

Рис.2.6. amCharts

Основні можливості:

1. Багато різноманітних типів графіків, включаючи інтерактивні карти та діаграми Ганта.
2. Можливість аналізу даних «вглиб» (дрілдаун) та інші добре опрацьовані інтерактивні опції.
3. Документація, що описує всі необхідні методи, добре написана, щоправда, на мій смак, не надто зручна у використанні.
4. Крута анімація графіків.
5. Інтеграція з React, Angular, Vue, Ember та ін.
6. Готовий плагін для WordPress.
7. Експорт графіків у зображення або PDF-файл.
8. "Живі" графіки, повноцінна кастомізація, спеціальні можливості відповідно до стандартів W3C.

9. Повна техпідтримка для всіх користувачів та пріоритетна для клієнтів з ліцензією.

10. Клієнти: Microsoft, Amazon, Ebay, NASA, Samsung, Yandex, AT&T та ін.

AnyChart

Рис.2.7. AnyChart

AnyChart — добре опрацьована, легковажна, багатофункціональна бібліотека JS для візуалізації даних з рендерингом в SVG/VML. Дозволяє веб-розробникам створювати різноманітні графіки та діаграми для подальшого прийняття рішень на їх основі[27].

Основні можливості:

1. Більше 80 типів графіків, включаючи звичайні графіки, біржові графіки, геовізуалізації (карти), а також діаграми Ганта та мережеві графіки PERT.

2. Багато варіантів, як можна завантажувати дані: XML, JSON, CSV, JS API, Google Sheets, HTML-таблиці.

3. Можливість «заглиблюватися» у дані на графіку (дрілдаун).

4. Технічні індикатори для технічного аналізу та анотації (інструменти для малювання) "з коробки".

5. Вичерпні документація та опис API, чуйна підтримка.

6. Інтеграція з Angular, Oracle APEX, React, Elasticsearch, Vue.js, Android, iOS та ін. Для розробників додатків та дашбордів у Qlik є спеціальний готовий екстеншн для Qlik Sense.

7. Багато готових прикладів графіків, діаграм, дашбордів, а також власна спеціальна пісочниця з автозаповненням коду.

8. Підтримка старих версій браузерів.

9. Експорт даних у різні формати, включаючи PDF звіти; JPG, PNG, SVG зображення; Excel (XLSX) або CSV файли із даними графіків.

10. Клієнти: Oracle, Microsoft, Citi, Samsung, Nokia, AT&T, Ford, Volkswagen, Lockheed Martin та ін.

Chart.js

Рис.2.8. Chart.js

Chart.js - проста і в той же час досить гнучка JavaScript бібліотека для візуалізації даних, популярна серед веб-дизайнерів та розробників. Вона є прекрасним базовим рішенням для тих, кому не потрібна велика різноманітність типів графіків та індивідуальних налаштувань, але кому достатньо, щоб графіки виглядали акуратно, наочно та інформативно.

Основні можливості:

1. 8 типів графіків та діаграм: лінійний (Line), лінійний з областями (Area), лінійний (Bar), круговий (Pie), пелюстковий «Радар» (Radar), полярний (Polar), пухирцевий (Bubble) та точкова діаграма розсіювання (Scatter).
2. Всі типи графіків можна кастомізувати та забезпечити анімацією, і всі вони адаптивні при роботі в мережі.
3. Функціональність може бути розширена за допомогою плагінів.
4. Гарна документація.
5. Підтримка через Stack Overflow.
6. Підтримка браузерів IE9+.

FusionCharts

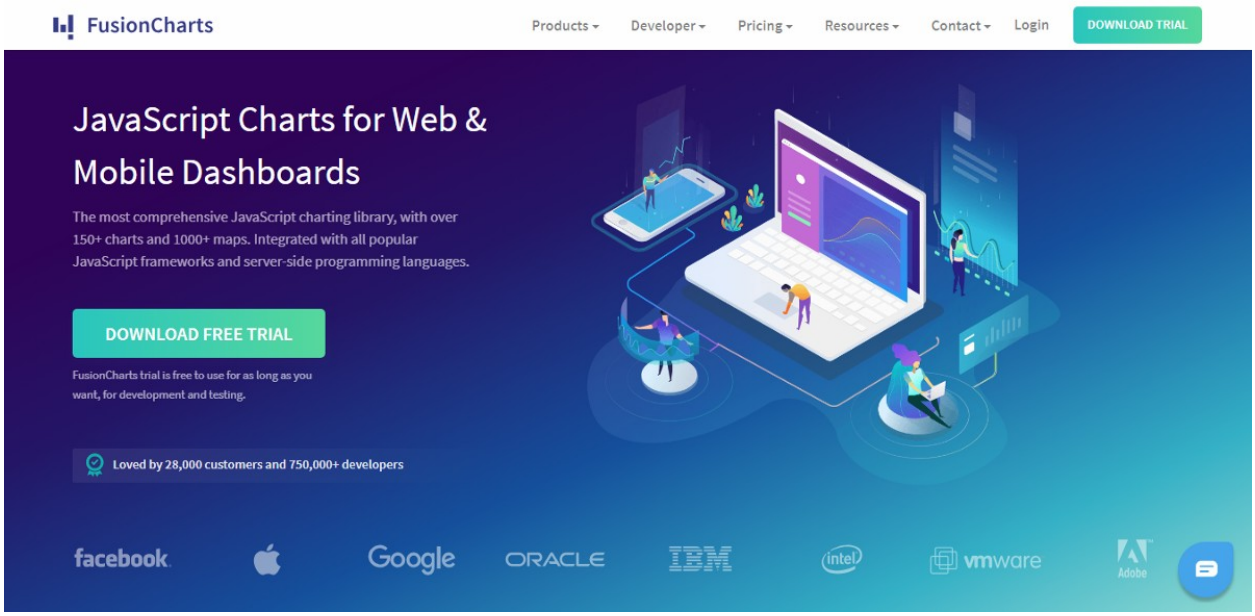

Рис.2.9. FusionCharts

FusionCharts –бібліотека для побудови інтерактивних графіків та діаграм, із сотнями готових прикладів. Графіки FusionCharts підтримують дані як у форматі JSON, так і XML. Рендеринг – через HTML5/SVG та VML.

Основні можливості:

1. Десятки типів графіків, 2D та 3D, і більше 950 карт, що покривають усі континенти.

2. Анімовані та повністю інтерактивні графіки та карти.

3. Серверні API для ASP.NET, PHP, Ruby on Rails.

4. Сумісність з jQuery, Angular, PHP, ASP.NET, React Native, Django, React, Ruby on Rails, Java та ін.

5. Досить детальні посібник користувача та опис API.

6. Безліч прикладів графіків та дашбордів.

7. Підтримка старих версій браузерів.

8. Експорт в PNG, JPG та PDF.

9. Підтримка – через базу знань та ком'юніті-форум.

10. Безлімітна пріоритетна підтримка для користувачів, які придбали ліцензію.

11. Клієнти: Apple, IBM, Google, Intel, Microsoft, PayPal, Oracle, Adobe та ін.

2.3 Способи збереження даних

Інформацією, що зберігається в базі даних (БД), може бути що завгодно: каталог продукції, інформація про клієнтів, контент веб-сайту та ін. Для забезпечення доступу до інформації, що зберігається в базі даних, а також для управління нею, застосовують систему управління базами даних (СУБД). СУБД - це комплекс мовних та програмних засобів, призначений для створення, ведення та спільного використання БД багатьма користувачами. Зазвичай СУБД розрізняють по моделі даних. Так, СУБД, що базуються на використанні реляційної моделі даних, називають реляційними СУБД. Системи управління базами даних допомагають відсортувати інформацію, а також пов'язати бази даних між собою, при цьому надавши звіт про зміни та зареєстровані події.[36]

Незважаючи на те, що всі системи управління базами даних виконують ту саму основну задачу (тобто дають можливість користувачам створювати, редагувати та отримувати доступ до інформації, що зберігається в базах

даних), сам процес виконання цього завдання варіюється в широких межах. Крім того, функції та можливості кожної СУБД можуть суттєво відрізнятися. Різні СУБД документовані по-різному: більш-менш ретельно. По-різному надається технічна підтримка.

Якщо йдеться про вибір СУБД для підприємства, слід взяти до уваги можливість СУБД «зростати» разом з розвитком організації. Малому бізнесу можуть знадобитися лише базові функції та можливості, а також невелика кількість інформації, що розміщується у БД. Але вимоги можуть суттєво зростати з часом, а перехід на іншу СУБД може стати проблемою.

1. Oracle 12c

Не дивно, що корпорація Oracle пропонує однойменний продукт, з якого починається розгляд варіантів популярних СУБД. Першу версію Oracle було створено наприкінці 70-х років. На даний момент цей продукт має блискучу репутацію. Крім того, існує кілька версій цього продукту для задоволення потреб конкретної організації.

Переваги

1. Найсвіжіші інновації та вражаючий функціонал вже запроваджено у цьому продукті, оскільки компанія Oracle прагне тримати планку навіть на тлі інших розробників СУБД;[35]

2. Оракул є вкрай надійною, фактично це зразок надійності серед подібних систем.

Недоліки

1. Вартість Oracle може бути непомірно високою, особливо для невеликих організацій;

2. Система може вимагати значних ресурсів вже відразу після установки, тому можливо, потрібно модернізувати обладнання для впровадження Oracle;

3. Ідеально підходить для великих організацій, які працюють з величезними базами даних та різноманітними функціями.

2. MySQL

MySQL – одна з найпопулярніших СУБД для веб-додатків. Фактично є стандартом de facto для веб-серверів, які працюють під управлінням операційної системи Linux. MySQL – це безкоштовний пакет програм, проте нові версії виходять постійно, розширюючи функціонал та покращуючи безпеку. Існують спеціальні платні версії, що призначені для комерційного використання. У безкоштовній версії максимальний акцент робиться на швидкість і надійність, а не на повноту функціоналу, який може стати і перевагою і дефіцитом - залежно від галузі застосування[34].

Розробку та підтримку MySQL здійснює корпорація Oracle, яка отримала права на торгову марку разом із поглиненою Sun Microsystems, яка раніше придбала шведську компанію MySQL AB. Продукт поширюється як під GNU General Public License, і під власною комерційною ліцензією. Крім цього, розробники створюють функціональність на замовлення ліцензійних користувачів. Саме завдяки такому замовленню майже в ранніх версіях з'явився механізм реплікації.

Ця СУБД дозволяє вибирати різні двигуни для системи зберігання, які дозволяють змінювати функціонал інструменту та виконувати обробку даних, що зберігаються у різних типах таблиць. Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, і таблиці InnoDB, підтримують транзакції лише на рівні окремих записів. Понад те, СУБД MySQL постачається зі спеціальним типом таблиць EXAMPLE, демонструючим принципи створення нових типів таблиць. Завдяки відкритій архітектурі та GPL-ліцензування, в СУБД MySQL постійно з'являються нові типи таблиць. Вона також має простий у використанні

інтерфейс і пакетні команди, які дозволяють зручно обробляти величезні обсяги даних. Система неймовірно надійна і прагне підпорядкувати собі всі доступні апаратні ресурси.

Переваги

1. Поширюється безкоштовно;
2. Чудово документована;
3. Пропонує багато функцій, навіть у безкоштовній версії;
4. Пакет MySQL включений у стандартні репозиторії найпоширеніших дистрибутивів операційної системи Linux, що дозволяє встановлювати її просто;
5. Підтримує набір інтерфейсів користувача;
6. Може працювати з іншими базами даних, включаючи DB2 та Oracle.

Недоліки

1. Потрібно витратити багато часу і зусиль, щоб змусити MySQL виконувати нескладні завдання, хоча інші системи роблять це автоматично, наприклад: створювати інкрементні резервні копії.
2. Відсутня вбудована підтримка XML або OLAP.
3. Для безкоштовної версії доступна лише платна підтримка.
4. Ідеально підходить для: організацій, які потребують надійного інструменту управління базами даних, але безкоштовні.

3. Microsoft SQL сервер

Ще однією з найпопулярніших СУБД є програмний продукт Microsoft SQL-сервер.[33] Це система управління базами даних, двигун якої працює на хмарних серверах, а також локальних серверах, причому можна комбінувати типи серверів одночасно. Незабаром після випуску Microsoft SQL Server 2016 Microsoft адаптувала продукт для операційної системи Linux, а на платформі Windows він працював спочатку.

Однією з унікальних особливостей версії 2016 є temporal data support (тимчасова підтримка даних), яка дозволяє відслідковувати зміни даних протягом часу. Остання версія Microsoft SQL-сервер підтримує dynamic data masking (динамічний маскування даних), яка гарантує, що лише авторизовані користувачі будуть бачити конфіденційні дані.[32]

Переваги

1. Продукт дуже простий у використанні
2. Поточна версія працює швидко та стабільно
3. Двигун надає можливість регулювати та відстежувати рівні продуктивності, які допомагають знизити використання ресурсів.
4. Ви можете отримати доступ до візуалізації на мобільних пристроях.
5. Він добре взаємодіє з іншими продуктами Microsoft.

Недоліки

1. Ціна для юридичних осіб виявляється неприйнятною для більшості організацій
2. Навіть при ретельному налаштуванні продуктивності SQL Server здатний задіяти всі доступні ресурси
3. Повідомляється про проблеми з використанням служби інтеграції для імпорту файлів
4. Є сенс купувати ліцензію на цей продукт, якщо вже впроваджено (читай "куплено") екосистему Microsoft.
5. Ідеально підходить для великих організацій, які вже використовують ряд продуктів Microsoft.

4. PostgreSQL

PostgreSQL є одним з декількох безкоштовних популярних варіантів СУБД, що часто використовується для ведення баз даних веб-сайтів. Це дуже стара система, тому в даний час вона добре розвинена, і дозволяє

користувачам управляти як структурованими, так і неструктурованими даними. Може бути використана більшості основних платформ, включаючи Linux (де особливо добре проявляється продуктивність). Чудово справляється із завданнями імпорту інформації з інших типів баз даних за допомогою власного інструментарію.

Двигун БД може бути розміщений у ряді середовищ, у тому числі віртуальних, фізичних та хмарних. Найсвіжіша версія, PostgreSQL 9.5, пропонує обробку великих обсягів даних та збільшення кількості одночасно працюючих користувачів. Безпека була покращена завдяки підтримці DBMS_SESSION.

Переваги

1. Є масштабованим рішенням і дозволяє обробляти терабайти даних.
2. Підтримує формат JSON.
3. Існує безліч зумовлених функцій.
4. Доступний ряд інтерфейсів.

Недоліки

1. Документація туманна, тому, можливо, відповіді деякі питання доведеться шукати в інтернеті.
2. Конфігурація може збентежити непідготовленого користувача.
3. Швидкість роботи може падати під час пакетних операцій або виконання запитів читання.
4. Ідеально підходить для організацій з обмеженим бюджетом, але вимагає залучення кваліфікованих спеціалістів, коли потрібно вибрати унікальний інтерфейс і використовувати json.

5. MongoDB

Ще одна безкоштовна система, яка має комерційну версію – MongoDB. Вважається одним із класичних прикладів NoSQL-систем, використовує JSON-подібні документи та схему бази даних. Написана мовою C++. Вона призначена для програм, які використовують як структуровані, так і неструктуровані дані. Ядро є дуже гнучким і працює при підключенні бази даних до програм через драйвери MongoDB. Існує широкий вибір доступних драйверів, тому легко знайти драйвер, який працюватиме з необхідною мовою програмування.

Оскільки система MongoDB не була розроблена для обробки моделей реляційних даних (хоча може це виконувати), можуть виникнути проблеми продуктивності, якщо спробувати використовувати її таким чином. Однак, двигун призначений для обробки різних даних, які не можна віднести до реляційних, і може добре справлятися там, де інші двигуни працюють повільно або безсилі.

MongoDB 5.0 - це остання версія (на липень 2021 р.), і вона має нову систему движків зберігання, що підключається. Документи можуть бути перевірені в процесі оновлення або виконання вставок, а функції текстового пошуку покращено. Нова здатність часткового індексування може призвести до більш високої продуктивності зменшуючи розмір індексів.

Переваги

1. Швидкість та простота у використанні
2. Двигун підтримує json та інші традиційні документи NoSQL.
3. Дані будь-якої структури можуть бути збережені/прочитані швидко та легко.

Недоліки

1. SQL не використовується як мови запитів.

2. Інструменти для перекладу SQL-запитів у MongoDB доступні, але їх слід розглядати як доповнення.

3. Програма установки може тривати багато часу.

4. Підходить для організацій, які працюють із різнорідними даними, які важко піддаються класифікації. Для застосування потрібні висококласні фахівці.

6. MariaDB

Ця СУБД є безкоштовною, але, як і багато інших безкоштовних програм, пропонує платні версії. Є безліч доступних плагінів розширень, мабуть, це СУБД, що найшвидше розвивається на даний момент.

MariaDB фактично - це відгалуження від СУБД MySQL, яке розробляється спільнотою під ліцензією GNU GPL. Розробку та підтримку MariaDB здійснює компанія MariaDB Corporation Ab та фонд MariaDB Foundation. Поштовхом до створення стала необхідність забезпечення вільного статусу СУБД, на противагу політиці ліцензування MySQL компанією Oracle. Система ліцензування MariaDB зобов'язує учасників, які бажають додати свій код до основної гілки СУБД, обмінюватися своїми авторськими правами з MariaDB Foundation для охорони ліцензії та можливості створювати критичні виправлення для MySQL.

Ядро бази даних дозволяє вибирати з кількох систем зберігання, і це робить використання ресурсів більш оптимізованим, що підвищує продуктивність запитів та обробки. До складу MariaDB включені підсистеми зберігання даних XtraDB для можливості заміни InnoDB як основної підсистеми зберігання. Також включені підсистеми Aria, PBXT та FederateX. Вона повністю сумісна з MySQL, і цілком підходить як заміна, т.к. повністю клонований як набір команд, і API. Багато розробників MySQL були залучені до процесу розробки, а зараз беруть участь у розвитку.

Переваги

1. Продуктивність
2. Індикатори дадуть вам знати, як обробляється запит.
3. Розширювана архітектура та плагіни дозволяють налаштовувати інструмент відповідно до ваших потреб.
4. Шифрування доступне в мережі, сервері та рівні програми.

Недоліки

1. На сьогодні стабільність нижча, ніж у MySQL, тому навіть на нових проектах можна рекомендувати встановлювати mysql.
2. Двигун досить новий, тому поки що немає жодних гарантій подальших оновлень.
3. Як і в багатьох інших безкоштовних базах даних, вам доведеться платити за підтримку.

Ідеальна як альтернатива MySQL, якщо MySQL не влаштовує з якихось причин.

Висновки до розділу 2

В другому розділі були розглянуто такі основні положення: розглянуто основні приклади IoT платформ, визначено їх переваги та недоліки. Розглянуто основні способи візуалізації та збереження даних.

На основі отриманих результатів було зроблено такі висновки:

Для реалізації інтелектуальної системи необхідно розробити самостійну IoT платформа, яка буде розтошовуватися на власному сервері. Для візуалізації даних обрана бібліотека Chart.js, тому що вона є безкоштовною та має відкритий код. Для збереження даних обрана БД – MongoDB, це аналог MySql, але вона є швидшою при роботі с великими обсягами даних.

3 МОДЕЛЮВАННЯ ТА ДОСЛІДЖЕННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Функціональна частина інтелектуальної системи обробки якості фільтрації води

Структура інтелектуальної системи включає три основні блоки – базу знань, вирішувач та інтелектуальний інтерфейс.

Вирішувач – набір функціоналу, за допомогою якого клієнт може контролювати показники свого фільтру

Розроблений функціонал ІС:

1. Збір даних від клієнтів по якості фільтрації води;
2. Створення звітів для подальшого аналізу;
3. Сповіщення клієнтів про неполадки фільтру.

Інтелектуальний інтерфейс – представлений у вигляді графіків та діаграм для кращого розуміння зібраної інформації по фільтру.База знань – база даних, що містить правила виведення та інформацію про людський досвід та знання в деякій предметній області.

База знань в інтелектуальній системі – здатність системи виводити нові знання із старих, знаходити закономірності в БЗ.

Інформацію, яку збирає ІС від клієнтів складається с 7 найменувань:

1. Температура води;
2. Вхідний TDS;
3. Вихідний TDS;
4. Ресурс картриджу для води у літрах та днях;
5. Тиск;
6. Інформація про протік води з фільтра.

id	api	flowspeed	liters	temperature	pressure	tds_in	tds_out	leak	reading_time	ostatok_l	ostatok_d
1	1	0.00	0.00	14.44	5.83	147	30	0	2022-02-15 00:00:02	5530.99	118
3	24	1	0	1	1	1	1	0	2022-02-15 00:00:02	6000	159
4	3	0.00	0.00	15.48	3.17	238	195	0	2022-02-15 00:00:02	5952.05	163
5	4	0.00	0.00	14.60	5.80	199	148	0	2022-02-15 00:00:02	5951.88	111
24059	2	0.00	0.00	15.81	89.45	231	69	0	2022-02-15 20:43:40	5996.39	180
24058	2	0.00	0.00	15.79	89.39	229	83	0	2022-02-15 20:43:30	5996.39	180
24057	2	0.00	0.00	15.81	89.41	231	77	0	2022-02-15 20:43:20	5996.39	180
24056	2	0.00	0.00	15.78	89.28	229	75	0	2022-02-15 20:43:09	5996.39	180
24055	2	0.00	0.00	15.81	89.01	230	77	0	2022-02-15 20:42:59	5996.39	180
24054	2	0.00	0.00	15.81	89.26	231	75	0	2022-02-15 20:42:49	5996.39	180
24053	2	0.00	0.00	15.81	89.38	231	77	0	2022-02-15 20:42:39	5996.39	180
24052	2	0.00	0.00	15.81	89.43	227	80	0	2022-02-15 20:42:28	5996.39	180
24051	2	0.00	0.00	15.81	89.60	230	81	0	2022-02-15 20:42:18	5996.39	180
24050	2	0.00	0.00	15.81	89.74	231	81	0	2022-02-15 20:42:08	5996.39	180
24049	2	0.00	0.00	15.99	89.68	230	80	0	2022-02-15 20:41:57	5996.39	180
24048	2	0.00	0.00	15.81	89.70	229	63	0	2022-02-15 20:41:47	5996.39	180
24047	2	0.00	0.00	15.81	89.92	229	84	0	2022-02-15 20:41:37	5996.39	180
24046	2	0.00	0.00	15.80	90.04	232	81	0	2022-02-15 20:41:27	5996.39	180
24045	2	0.00	0.00	15.81	90.29	229	78	0	2022-02-15 20:41:16	5996.39	180
24044	2	0.00	0.00	15.81	90.13	230	80	0	2022-02-15 20:41:06	5996.39	180
24043	2	0.00	0.00	15.80	90.06	231	75	0	2022-02-15 20:40:56	5996.39	180
24042	2	0.00	0.00	15.81	90.01	231	78	0	2022-02-15 20:40:46	5996.39	180
24041	2	0.00	0.00	15.81	89.63	230	77	0	2022-02-15 20:40:35	5996.39	180
24040	2	0.00	0.00	15.81	89.51	228	77	0	2022-02-15 20:40:25	5996.39	180
24039	2	0.00	0.00	15.81	89.63	228	74	0	2022-02-15 20:40:15	5996.39	180

Рис.3.1. База даних ІС

БЗ відрізняється від бази даних наявністю механізму висновків. Розроблений функціонал ІС представлений у вигляді діаграм.

Діаграма – це графічне уявлення для візуалізації даних, у якому «дані представлені символами, такими як стовпці на лінійній діаграмі, лінії на лінійній діаграмі або зрізи на круговій діаграмі».

3.1.1 Механізм висновку – сповіщення клієнта про протік фільтру

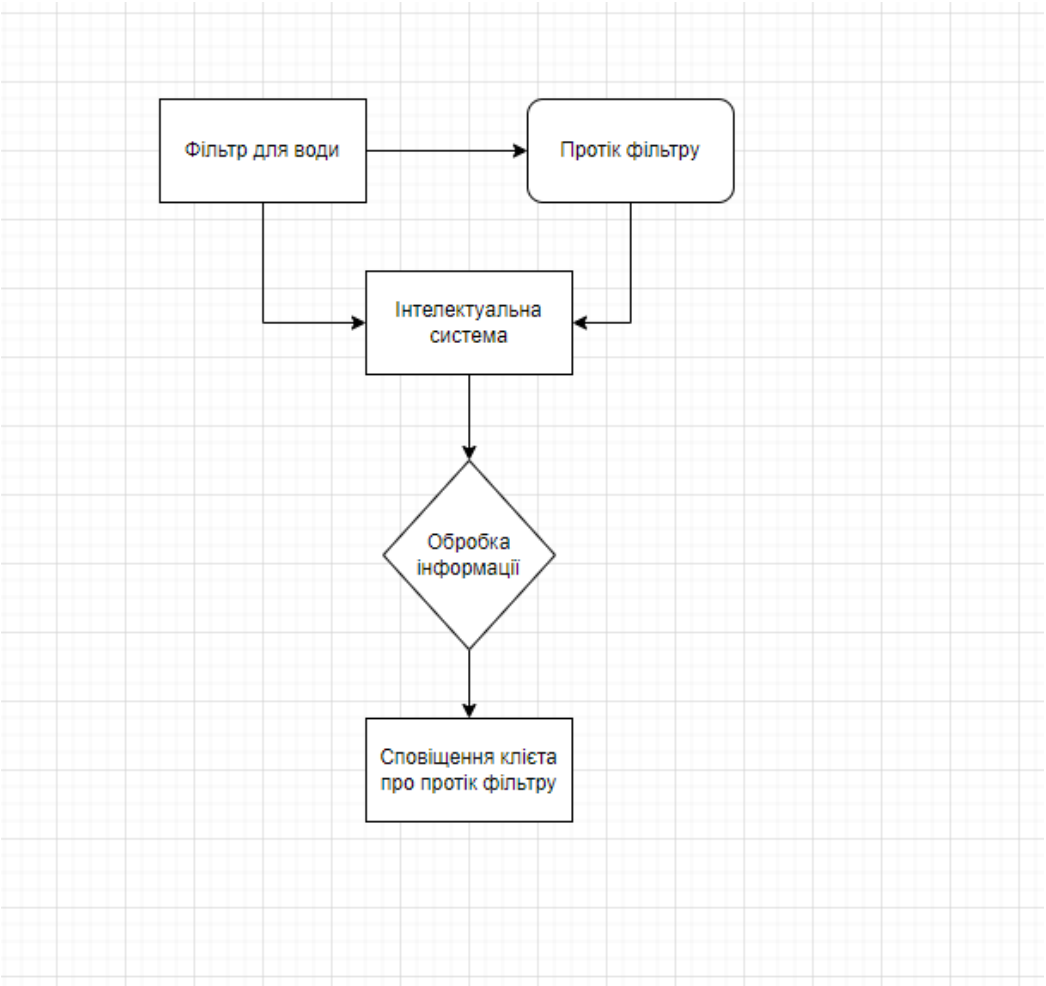

Рис.3.2. Сповіщення клієнта про протік фільтру

Одним із головних механізму висновків є сповіщення клієнта про протік фільтру. Даних механізм спрацювує в момент, коли з фільтром відбуваються несправності. В цей момент кліжнт отримує на свій телефон або, якщо клієнт знаходиться в даний момент на сторінці водобреження результатів роботи фільтру сповіщення про помилку.

Дану помилку неможливо закрити, адже вона з’являється автоматично та зникає, коли клієнт виправить помилку фільтру та зупинить протік.

3.1.2 Механізм висновку – сповіщеня клієнта про необхідність заміни картриджу

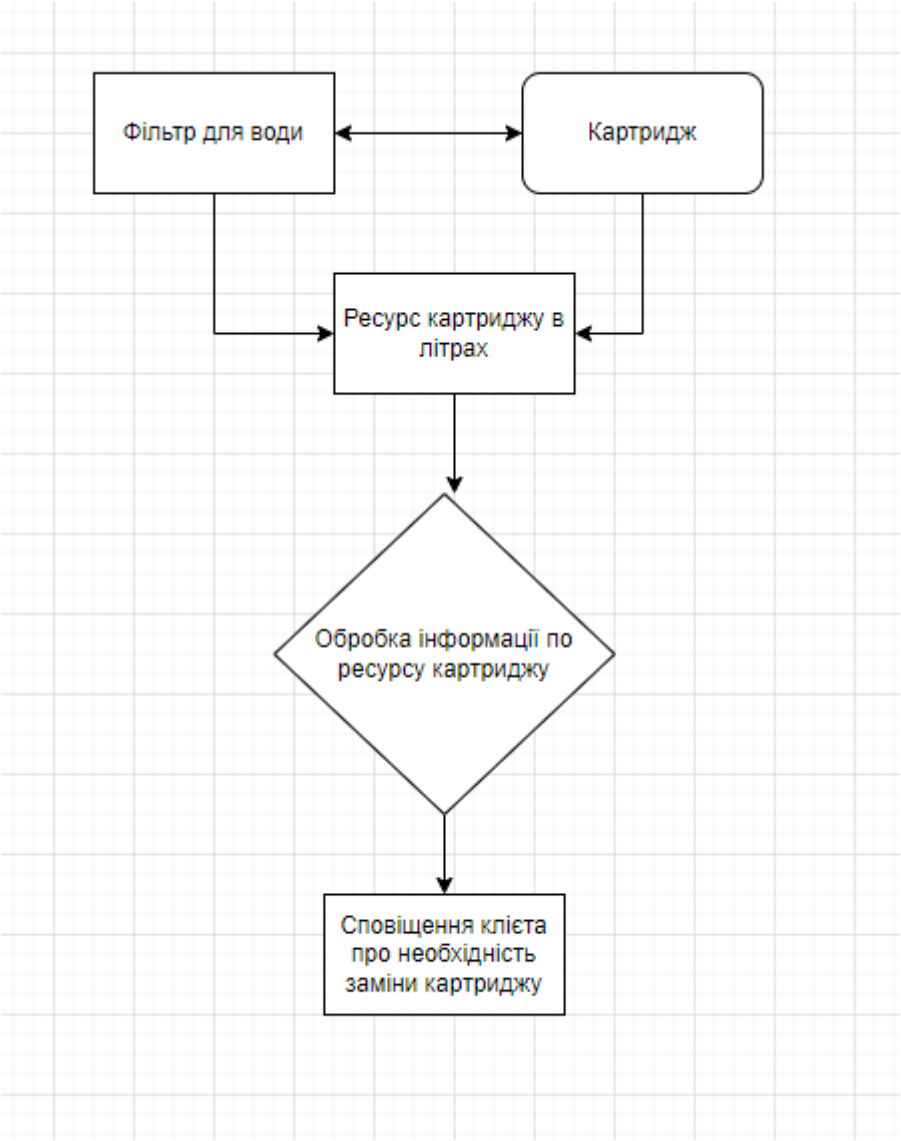

Рис.3.3. Сповіщення клієнта про необхідність заміни картриджу через ліміт в літрах

Другим основним механізмом висновку є сповіщення клієнту про необхідність заміни картриджу.

Так як кожен для ісправної роботи фільтру для води необхідні картриджі. Головною особливістю картриджів є їх ресурс в літрах та в добах. Тому даний функціонал про своєчасне сповіщення клієнтів об необхідності зміни картриджу для фільтру є необхідним, бо допомогає зберігати правельність роботи фільтру, тим самим допомогає пити чисту воду.

Механізм висновку про необхідність заміни картриджу скалдається з 2 блоків:

1. Ресурс картриджу в літрах(рис.3.4);
2. Ресурс картриджу в добах(рис.3.5).

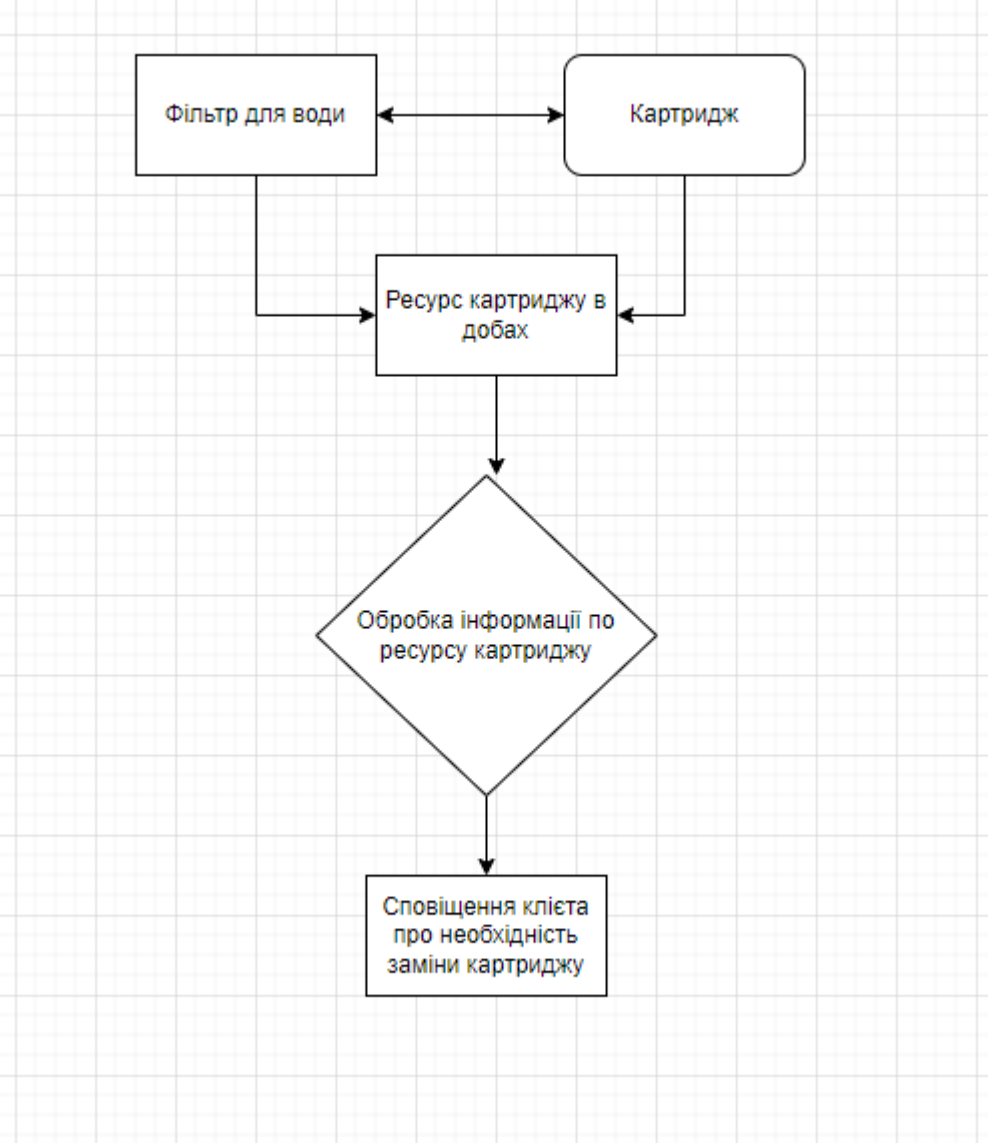

Рис.3.4. Сповіщення клієнта про необхідність заміни картриджу через ліміт в добах

3.1.3 Механізм висновку – сповіщення клієнта про сильний тиск у фільтрі

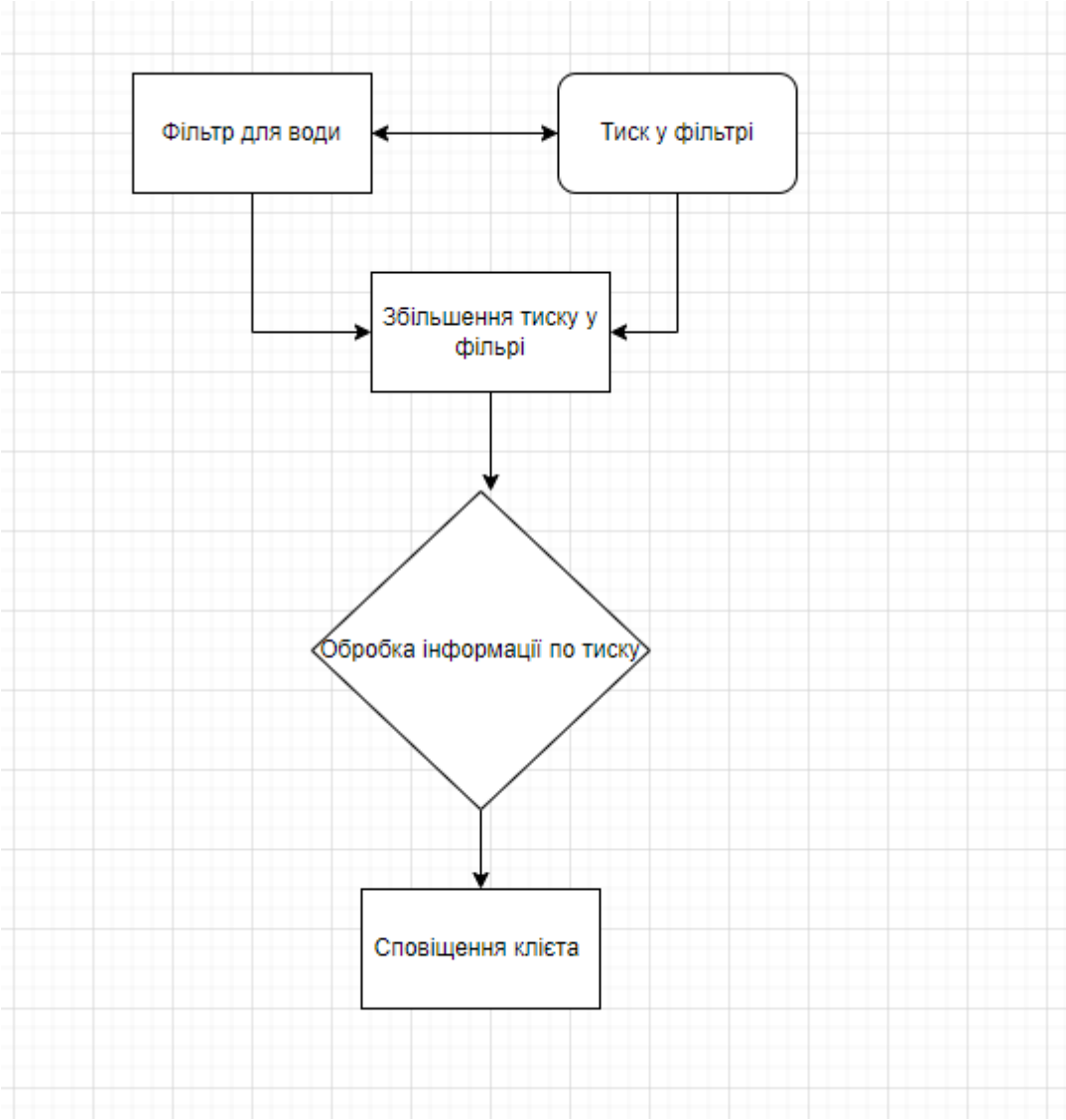

Рис.3.5. Сповіщення клієнта про збільшення тиску у фільтрі

Задля того, щоб попередити клієнта про несправність роботи фільтру був розроблен функціонал про сповіщення збільшення тиску у фільтрі.

Даний механізм висновку є дуже корисним, адже попереджає клієнта заздалегідь про ймовірну несправність фільтру.

3.1.4 Механізм висновку – сповіщення клієнта про збільшення різних домішок у воді

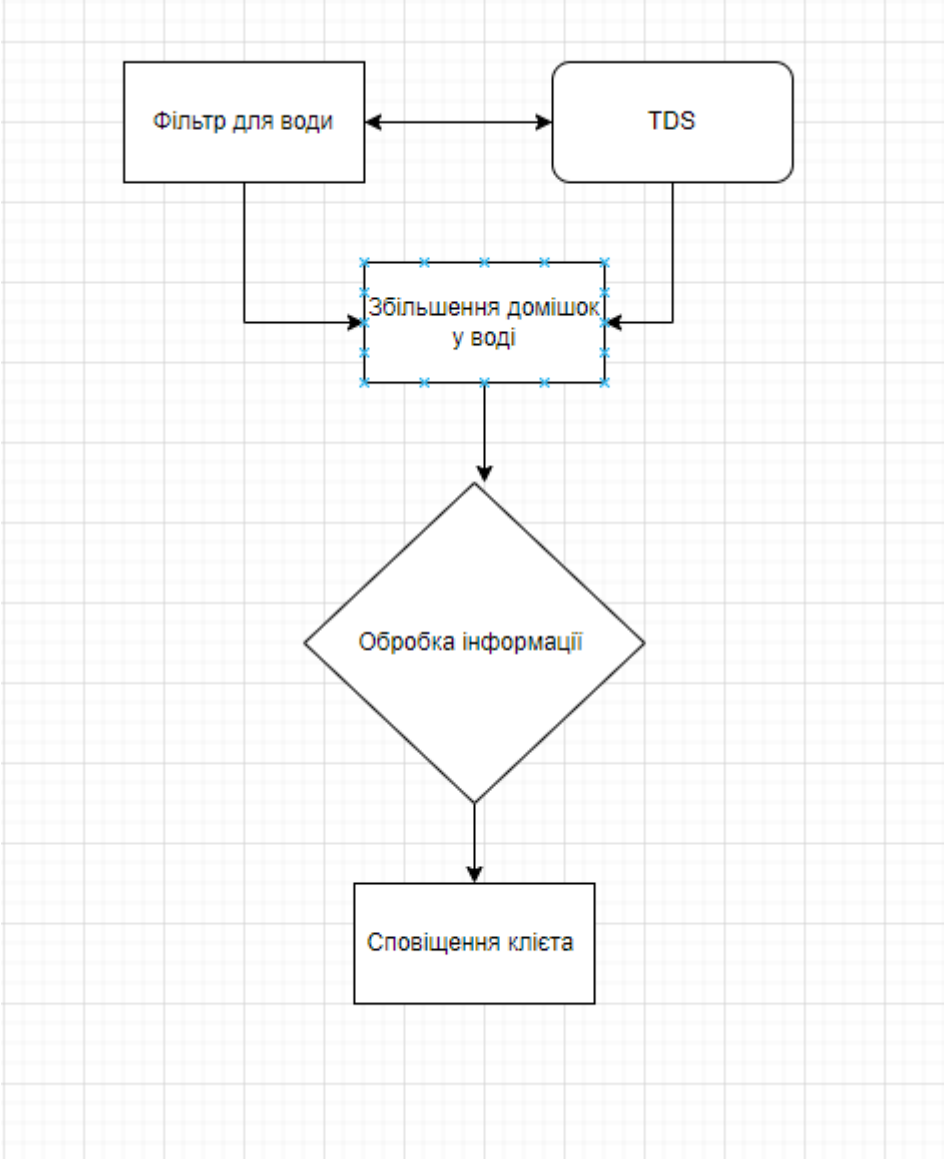

Рис.3.6. Сповіщення клієнта про збільшення домішок у воді

Даний механізм висновку показує інформацію клієнту про якість питної води, а саме кількість домішок у воді. Якщо кілікість домішок у воді перевищує норму, клієнт отримує повідомлення про підвищення домішок. Тим самим допомогає контролювати якість питної води.

3.2 Методи обробки даних

Обробка даних — це збір та обробка даних у використовуваній та бажаній формі. Дії — це не що інше, як обробка, яка може виконуватися

вручну або автоматично у заздалегідь визначеній послідовності дій. Раніше це виконувалося вручну, що займало багато часу та призводило до помилок при обробці, тому зараз більшість обробки виконується автоматично комп'ютерами, які можуть швидко обробляти та давати правильні результати. Наступним кроком є перетворення в потрібну форму, обробка зібраних даних відповідно до вимог програми та перетворення їх у потрібну форму, що означає перетворення даних у корисну інформацію, яка може виконувати конкретні завдання в програмі. зібрані з різних джерел, таких як дані текстових файлів, дані файлів Excel, бази даних і навіть неструктуровані дані, такі як зображення, аудіокліпи, відеокліпи, дані GPRS.

Результатом обробки даних є змістовна інформація, яка може бути в різних формах, таких як таблиці, зображення, діаграми, графіки, векторні файли, аудіо тощо, у всіх форматах залежно від бажаної програми чи програмного забезпечення для отримання.

Тому визначення обробки даних – це перетворення даних в корисну інформацію.

Як обробляються дані?

Обробка даних починається зі збору даних. Дані, зібрані для перетворення в необхідну форму, повинні оброблятися поетапно, наприклад, зібрані дані повинні зберігатися, класифікуватися, оброблятися, аналізуватися та подаватися. Таким чином, обробка даниї розділена на 6 основних етапів, як показано нижче:

1. Збір даних;
2. Зберігання даних;
3. Сортування даних;
4. Обробка даних;
5. Аналіз даних;
6. Подання даних та висновки.

1. Збір даних

Логічно пов’язані дані збираються з різних джерел, різних форматів, різних типів, наприклад, XML, файли CSV, соціальні мережі, зображення структуровані чи неструктуровані дані тощо.

2. Зберігання даних

Тепер зібрані дані мають зберігатися у фізичній формі, наприклад, на папері, блокноті та у будь-якій іншій фізичній формі. Зараз, завдяки обробці даних і великим даним, збір даних є величезним, навіть у структурованому чи неструктурованому вигляді. Дані повинні зберігатися в цифровій формі для змістовного аналізу та представлення відповідно до вимог програми.

3. Сортування даних

Після етапу зберігання безпосередніми кроками є сортування та фільтрація. Для сортування та фільтрація потрібно організувати дані в певному порядку вмісту та відфільтрувати лише необхідну інформацію для легкого розуміння візуалізації та аналізу.

4. Обробка даних

Серія або безперервне використання та обробка даних для перевірки, перетворення, упорядкування, інтеграції та вилучення даних у корисній формі вихідних даних для майбутнього використання.

5. Аналіз даних

Аналіз даних – це процес систематичного застосування або оцінки даних з використанням аналітичних і логічних міркувань, щоб проілюструвати кожен компонент представлених даних і зробити висновки про результати або рішення.

6. Подання даних та висновки

Результат аналізу можна представити в різних формах, таких як діаграми, текстові файли, файли Excel, графіки тощо.

Програмне забезпечення або комбінація програмного забезпечення можна використовувати для зберігання, сортування, фільтрації та обробки даних за потребою та бажанням. Це можна зробити за допомогою певного програмного забезпечення відповідно до вимог програми та попередньо визначеного набору операцій.

Різні типи вихідних даних

Різні типи вихідних файлів, наприклад:

1. Звичайний текстовий файл – експорт у записник або файл WordPad. Це найпростіша форма файлу даних.

2. Електронна таблиця/Електронна таблиця – у цьому форматі файлу дані представлені в рядках і

3. Стовпці, які полегшують розуміння та аналіз даних. Цей формат файлу виконує різні операції, такі як фільтрація та сортування в порядку зростання або спадання, а також статистичні операції.

4. Діаграми та діаграми – діаграми та формати діаграм є стандартними функціями більшості програмного забезпечення. Цей формат дуже простий для аналізу даних, йому не потрібно читати кожну числову інформацію, він займає багато часу, а дані можна зрозуміти й проаналізувати з першого погляду.

5. Файли зображень або карти/вектори - якщо програми, необхідні для зберігання та аналізу просторових даних, корисні для експорту даних у файли зображень, файли карт або векторні файли.

6. Крім того, іншим форматом може бути деякий формат файлу програмного забезпечення, який може використовуватися та оброблятися спеціалізованим програмним забезпеченням.

Способи обробки даних

Існують в основному три способи обробки даних: ручний, механічний та електронний.

1. Вручну: у цьому методі дані обробляються вручну. ціль завдання обробки, Обчислення, сортування та фільтрація, а також логічні операції виконуються вручну без використання будь-яких інструментів, електроніки чи програмного забезпечення для автоматизації.

2. Механічний. У цьому методі дані не обробляються вручну, а генеруються за допомогою дуже простих електронних і механічних пристроїв, таких як калькулятори та друкарські машинки.

3. Електронізація – це найшвидший метод обробки даних і сучасна технологія з сучасними необхідними характеристиками, такими як найвища надійність і точність. Цей метод реалізується набором програм або програмного забезпечення, що працюють на комп’ютері.

В інформаційних системах з розподіленою обробкою даних застосовують різні види інформаційно-технологічної архітектури, що залежать від використовуваних програмних, технічних засобів, структури інформаційних баз даних, типу мереж. Найбільш поширеними видами інформаційно-технологічної архітектури є архітектури "файл-сервер", "клієнт-сервер".

В розробленій ІС була використана архітектура "клієнт-сервер".

Архітектура «клієнт-сервер» дозволяє подолати непродуктивне пересилання великих інформаційних потоків у мережі. Це досягається за рахунок поділу програм на дві частини: клієнтську та серверну. Клієнтська частина встановлюється на комп'ютері робочого місця, серверна - на мережному сервері. Таким чином, на сервері знаходяться не лише загальні бази даних, а й програми пошуку та запису. Це дозволяє «клієнтам» (іншим програмам, розташованим на робочих станціях) надсилати серверу запит не всю інформацію з бази даних, лише на потрібну, причому частково чи

повністю оброблену. При цьому суттєво зменшується трафік мережі, знижується завантаженість каналу передачі даних.

Архітектура «клієнт-сервер» може бути дволанковою та багатоланковою. При дволанковій архітектурі клієнти безпосередньо взаємодіють із сервером. Багатоланкова архітектура відрізняється існуванням ще однієї чи кількох ланок, про серверів додатків чи серверів обслуговування, які є проміжними ланками між клієнтами і сервером. Сервер програм виконує ряд функцій, як системних, так і користувацьких, які у разі використання дволанкової архітектури виконує або клієнт, або сервер.

За способом організації обміну даними між клієнтом та сервером розрізняють моделі «товстого» та «тонкого» клієнта.

У моделі «товстий клієнт» на сервері реалізовані переважно функції доступу до даних, проте прикладні обчислення виконуються на «клієнтських» програмах, тобто. сервер лише відбирає потрібні дані та пересилає їх на робочу станцію, де і виконується їхня обробка. Результати обробки надсилаються назад серверу для збереження їх у загальній базі даних.

У моделі "тонкий клієнт" значна частина прикладної обробки даних виконується безпосередньо на сервері, а на робочу станцію передаються дані для перегляду в екранних формах та результати виконання звітів.

Організація розподіленої обробки даних залежить від способу їхнього розподілу. Існують централізована, децентралізована та змішана організації розподілу даних.

1. Централізована організація даних передбачає наявність на сервері загальної бази даних (рис. 3.7). Усі операції з базою даних забезпечуються цим сервером. Кожен клієнт мережі має доступ до єдиної бази даних за допомогою віддаленого запиту. Перевага цієї організації - простота підтримки бази даних в актуальному стані. До недоліків можна віднести

обмеженість обсягу зовнішньої пам'яті, що впливає розмір бази даних, обмеження на паралельну обробку запитів клієнтів, порушення роботи клієнтів у мережі при відмові апаратних засобів.

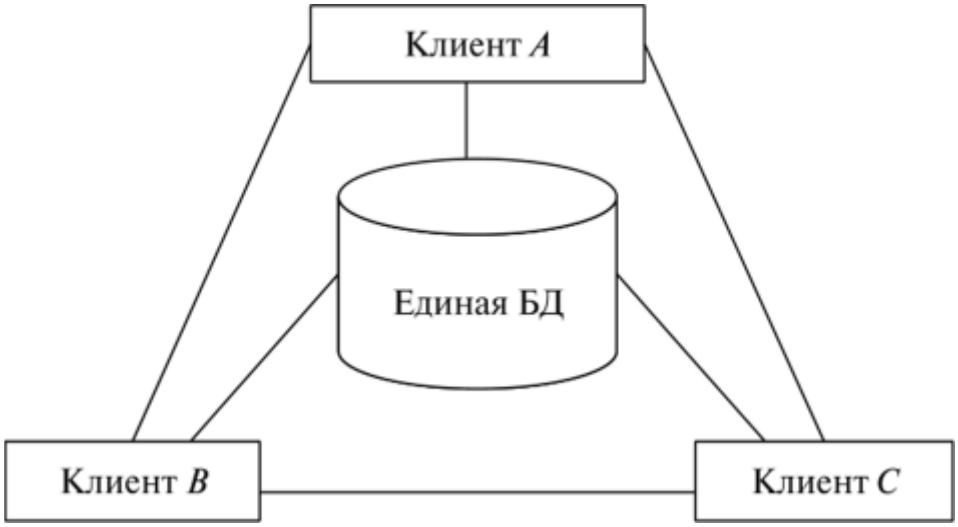

Рис. 3.7 Централізована організація даних

2. Децентралізована організація даних передбачає розбиття загальної бази на кілька фізично розподілених. Кожен клієнт має доступ до своєї БД, яка може бути частиною загальної бази даних (рис. 3.8, а), або її копією (рис. 3.8, б).

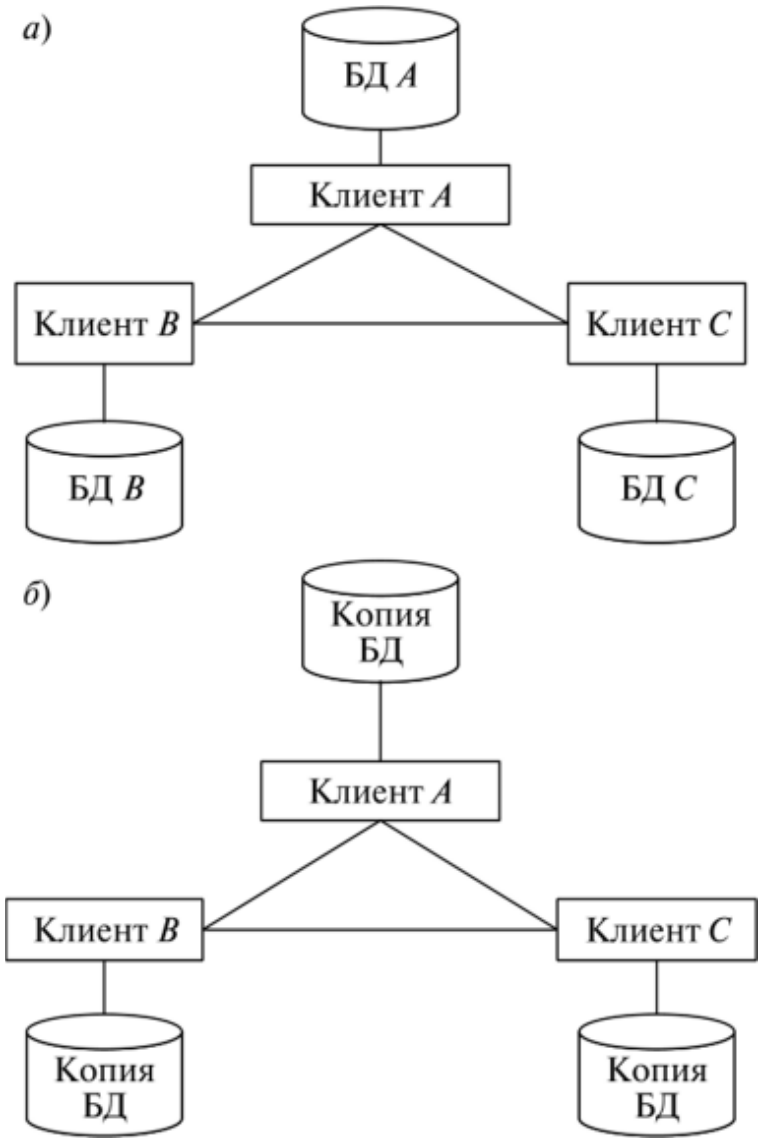

Рис. 3.8 Децентралізована організація даних способом розподілу (а) та способом дублювання (б)

При розбитті БД кожна частина зберігається на окремому сервері. При дублюванні БД кожному сервері мережі розміщується повна копія БД.

3. Змішана організація даних передбачає об'єднання двох способів розподілу - розбиття та дублювання. При цьому зберігаються переваги та недоліки кожного з цих способів.

Таким чином, інтегрована інформаційна система базується на інтегрованому використанні персональних комп'ютерів, спеціалізованих потужних комп'ютерів (серверів), об'єднаних каналами зв'язку та єдиною

технологією у єдину глобальну корпоративну мережу. Ця технологія й у великих підприємств, мають розгалужену мережу філій чи дочірніх компаній. При цьому в корпоративній мережі виділяються два рівні обробки: локальна обчислювальна мережа центрального органу управління підприємством та локальна обчислювальна мережа філій. Центральний офіс та філії об'єднуються в єдине інформаційно-обчислювальне середовище, зв'язок у якому здійснюватись по некомутованих провідних каналах, оптоволоконних каналах, радіо- та супутникових каналах,

3.3 Алгоритми обробки даних

Алгоритм – це система точних і зрозумілих розпоряджень про зміст та послідовність виконання кінцевого числа дій, необхідних для вирішення будь-якого завдання даного типу.

3.3.1 Алгоритм обробки даних – дерево рішень

Дерева рішень — це популярний алгоритм класифікації, який витягує правила прийняття рішень безпосередньо з вихідних даних під час процесу навчання. Вони являють собою ієрархічну послідовність правил виду «якщо...то».

Дерева рішень здатні виявляти нелінійні залежності та нетипові (рідкісні) ситуації.

Щоб вирішити, до якого класу слід віднести певний об’єкт або ситуацію, потрібно відповісти на запитання, розташовані в цьому вузлі дерева, починаючи з кореня. Наприклад, питання виглядає так: «чи більше значення параметра A від B?». Якщо відповідь ствердна, перейдіть до правильного вузла наступного шару і т.д.

Для інтерпретації результатів класифікації за допомогою дерев рішень використовуються інструменти візуалізації, щоб показати структуру дерева та правила, створені в ньому.

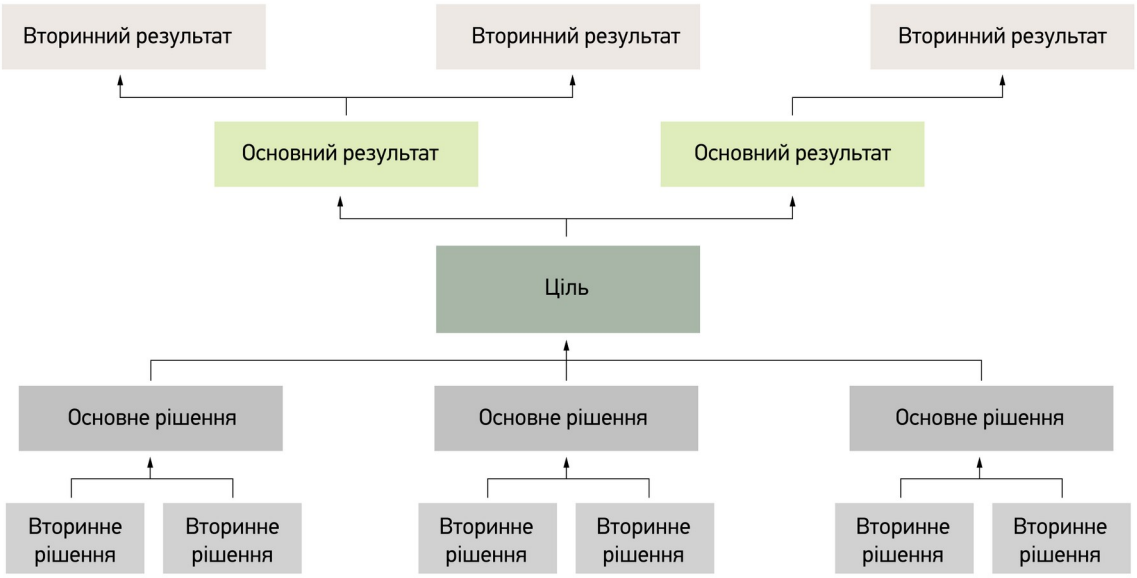

Рис.3.9. Приклад алгоритму обробки даних – дерево рішень

3.3.2 Алгоритм обробки даних – розбір XML

Мова розмітки xml призначена для зручного кодування та читання інформації машинним та ручним способом. Структура файлу та його параметри прописуються за допомогою тегів, атрибутів та препроцесорів. За призначенням та розв'язуваними завданнями він нагадує html, але простіше і зрозуміліше у використанні, а теги встановлюють самі розробники. Його головна перевага – це читання. Наприклад, для позначення жирного тексту в XML достатньо написати <Жирний></Жирний>, тоді як для html синтаксис жорстко прописаний і запис буде виглядати так: .

У деяких ситуаціях, за даними, доводиться звертатися до веб-сервісу. Однак, якщо відповідь від веб-сервісу була кудись збережена (наприклад, в базу даних), то для подальшого аналізу отримати набір даних у звичному табличному вигляді можна тільки за допомогою вузла Розбір XML.

Для коректної роботи даного обробника заздалегідь має бути підключена xml-схема. Відповідно до неї, кожен рядок вихідного набору даних трансформується на таблицю. У разі аналізу кількох рядків, що містять xml, результуючі набори даних об'єднуються. Для того щоб визначити з

якого вихідного xml-рядка було отримано той чи інший рядок в результуючій таблиці, в налаштуваннях вузла можна вказати поле-ідентифікатор.

```
<?xml version="1.0" encoding="utf-8"?>
<UkrainianCadastralExchangeFile>
  <AdditionalPart>
    <ServiceInfo>
      <FileID>
        <FileDate>2016-10-03</FileDate>
        <FileGUID>cb8952f3-c385-4b68-bbf3-74a40366f010</FileGUID>
      </FileID>
      <FormatVersion>0.7</FormatVersion>
      <ReceiverName></ReceiverName>
      <Software>Digitals</Software>
      <SoftwareVersion>2013.11.04</SoftwareVersion>
    </ServiceInfo>
    <InfoLandWork>
        </CadastralQuarterInfo>
      </CadastralQuarters>
    </CadastralZoneInfo>
  </InfoPart>
</UkrainianCadastralExchangeFile>
```

Рис.3.10. Приклад файлу XML

Висновки до розділу 3

У третьому розділі було розглянуто такі основні положення: розглянута функціональна частина інтелектуальної системи обробки якості фільтрації водию Було виявлено основні механізму висновків: сповіщення клієнта про протік фільтру, сповіщення клієнта про необхідність заміни картриджу, сповіщення клієнта про сильний тиск у фільтрі, сповіщення клієнта про підвищення різних домішок у воді.

Було виделено основні методи обробки даних. Виділено алгоритми обробки даних, а саме: дерево рішень та розбір XML.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ОБРОБКИ ЯКОСТІ ФІЛЬТРАЦІЇ ВОДИ

4.1 Обґрунтування та вибір базових програмних засобів

Патерн MVC

MVC (Model-View-Controller) – це архітектурна схема, яка розділяє додаток на три основні логічні компоненти: модель, вигляд та контролер. Традиційно використовується для настільних графічних додатків, модель стала популярною з появою вебдодатків. Сьогодні майже всі популярні мови підтримують цю архітектуру.

Компоненти

Модель є центральною складовою патерну. Це динамічна структура даних програми, яка функціонує незалежна від інтерфейсу користувача. Модель відповідає всій логіці, пов'язаній з даними, з якою працює користувач. Модель безпосередньо керує даними, логікою та правилами програми.

Вигляд (view) може бути будь-яке вихідне представлення інформації, наприклад діаграма або діаграми, таблиці даних, будь-яка складов frontend розробки .

Контроллер (controller), приймає вхід і перетворює його в команди для Model або View. Він діє як інтерфейс між компонентами Model і View для обробки всієї бізнес-логіки та вхідних запитів, маніпулює даними за допомогою компонента Model і взаємодіє з View для надання кінцевого результату.

Взаємодія:

Model відповідає за управління даними програми. Даний компонент отримує введені дані користувача від контролера. View означає представлення моделі у певному форматі. Controller відповідає за введення користувачем будь-яких даних та здійснює взаємодію з об'єктами моделі та її

даних. Контролер отримує інформацію, додатково перевіряє її через логіку програми, а потім передає перевірену інформацію до моделі .

Взаємодію між трьома компонентами добре видно на рисунку 4.1.

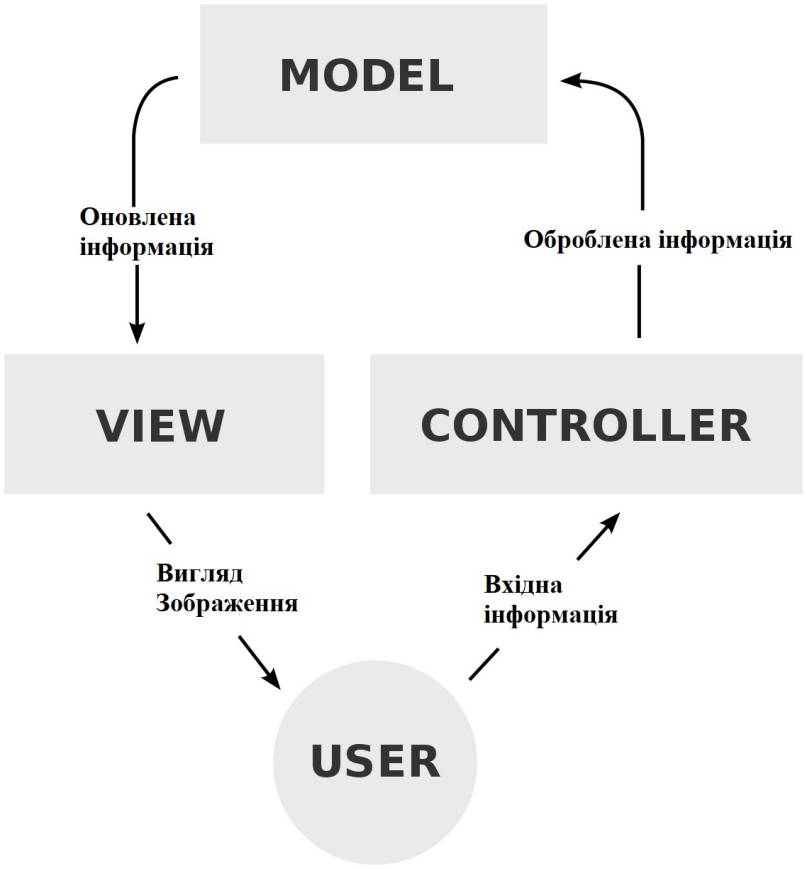

Рис. 4.1. Взаємодія компонентів MVC патерну

Інструменти автоматизации front-end разробки

Для того щоб прискорити та полегшити розробку програм потрібно подбати про технічне оточення.

Gulp – це інструмент автоматизації front-end розробки, він допоможе автоматизувати рутинні завдання і прискорити роботу. [9]

Основні задачі, які вирішує Gulp:

1. Мінімізація та конкатенація Javascript і CSS файлів;

2. Мінімізація HTML файлів;

3. Використання CSS препроцесорів Sass, Less, Stylus та інших;

4. Оптимізація зображень;

5. Автоматична простановка Вендорний префіксів;

6. Використання шаблонизатор (Jade);

7. Видалення невикористаного CSS (UnCSS);

8. Валідація коду і тестування;

Для того щоб встановити Gulp на комп'ютер потрібно перш за все встановити Node.js, тому що для коректної роботи Gulp потрібен пакетний менеджер npm від Node.js.

Розуміння того, що таке Node.js допоможе краще розібратися з npm. Node.js – це інтерпретатор мови JavaScript. Сам по собі Node.js є C-додатком, яке отримує на вході JavaScript-код і виконує його.[8]

Пакети в Node.js:

Пакетом в Node.js називається один або кілька JavaScript-файлів, що представляють собою якусь бібліотеку або інструмент.

Npm (аббр. node package manager) - це стандартний менеджер пакетів, автоматично встановлюється разом з Node.js. Він використовується для скачування пакетів з хмарного сервера npm, або для завантаження пакетів на ці сервера. [7]

Файл package.json

Файл package.json містить в собі інформацію про додаток: назва, версія, автор тощо. Будь-яка директорія, в якій є цей файл, інтерпретується як Node.js-пакет, навіть якщо ви не збираєтеся публікувати його.[6]

4.2 Обґрунтування та вибір технології розробки ПЗ

Для досягнення поставленної задачі та розробки інструментарію необхідо визначитися з наборои технологій.

PHP (рекурсивний акронім словосполучення PHP: Hypertext Preprocessor) – це поширена мова програмування загального призначення з

відкритим вихідним кодом. PHP спеціально сконструйований для веброзробок та його код може впроваджуватися безпосередньо в HTML [1].

Через те , що система управління інтернет-магазином OpenCart написана за допомогою мови програмування PHP буде доречним використовувати саме його в написанні модулей для системи.

HTML (HyperText Markup Language - «мова гіпертекстової розмітки») – визначає зміст і структуру вебконтенту. Є базовим інструментом для веб-технологій. Інші технології, крім HTML, зазвичай використовуються для опису зовнішнього вигляду / уявлення (CSS) або функціональності / поведінки (JavaScript) вебсторінки. В даній роботі HTML потрібний для створення розмітки модулей, програм 2].

CSS (Cascading Style Sheets = Каскадні таблиці стилів) – це мова, яка відповідає за візуальне відображення веб-сторінок. Переважно використовується як засіб опису, оформлення зовнішнього вигляду веб-сторінок, написаних за допомогою мов розмітки HTML і XHTML, але може також застосовуватися до будь-яких XML-документів, наприклад, до SVG або XUL [3].

JavaScript (часто JS) – це об'єктно-орієнтована мова з функціями першого класу. Найбільш широке застосування знаходить як мова сценаріїв вебсторінок щоб додати інтерактивності вебсторінам, але також використовується і в інших програмних продуктах, наприклад, node.js або Apache CouchDB [4].

XML (eXtensible Markup Language) – «розширена мова розмітки».

XML був створенний для більш зручного зберігання і передачі даних, в тому числі через Інтернет.

Важливо розуміти, що XML – це не виконуваний код, а мова опису даних. Після того, як описавши дані за допомогою XML, потрібно написати

код (наприклад, на Java), який зможе ці дані відправити / прийняти / обробити [5].

SASS (Syntactically Awesome Stylesheets) – це розширення, препроцессор CSS. Sass дає можливість використовувати змінні, вкладені правила, міксини і багато іншого і все це з повністю сумісним з CSS синтаксисом. Sass допомагає зберігати величезні таблиці стилів добре організованими. Sass має одну велику перевагу, за допомогою цього метаязику можно зекономити час та пришвидшити процес розробки.

4.3 Опис програмної реалізації

База даних

id	api	flowspeed	liters	temperature	pressure	tds_in	tds_out	leak	reading_time	ostatok_l	ostatok_d
1	1	0.00	0.00	14.44	5.83	147	30	0	2022-02-15 00:00:02	5530.99	118
3	24	1	0	1	1	1	1	0	2022-02-15 00:00:02	6000	159
4	3	0.00	0.00	15.48	3.17	238	195	0	2022-02-15 00:00:02	5952.05	163
5	4	0.00	0.00	14.60	5.80	199	148	0	2022-02-15 00:00:02	5951.88	111
24059	2	0.00	0.00	15.81	89.45	231	69	0	2022-02-15 20:43:40	5996.39	180
24058	2	0.00	0.00	15.79	89.39	229	83	0	2022-02-15 20:43:30	5996.39	180
24057	2	0.00	0.00	15.81	89.41	231	77	0	2022-02-15 20:43:20	5996.39	180
24056	2	0.00	0.00	15.78	89.28	229	75	0	2022-02-15 20:43:09	5996.39	180
24055	2	0.00	0.00	15.81	89.01	230	77	0	2022-02-15 20:42:59	5996.39	180
24054	2	0.00	0.00	15.81	89.26	231	75	0	2022-02-15 20:42:49	5996.39	180
24053	2	0.00	0.00	15.81	89.38	231	77	0	2022-02-15 20:42:39	5996.39	180
24052	2	0.00	0.00	15.81	89.43	227	80	0	2022-02-15 20:42:28	5996.39	180
24051	2	0.00	0.00	15.81	89.60	230	81	0	2022-02-15 20:42:18	5996.39	180
24050	2	0.00	0.00	15.81	89.74	231	81	0	2022-02-15 20:42:08	5996.39	180
24049	2	0.00	0.00	15.99	89.68	230	80	0	2022-02-15 20:41:57	5996.39	180
24048	2	0.00	0.00	15.81	89.70	229	63	0	2022-02-15 20:41:47	5996.39	180
24047	2	0.00	0.00	15.81	89.92	229	84	0	2022-02-15 20:41:37	5996.39	180
24046	2	0.00	0.00	15.80	90.04	232	81	0	2022-02-15 20:41:27	5996.39	180
24045	2	0.00	0.00	15.81	90.29	229	78	0	2022-02-15 20:41:16	5996.39	180
24044	2	0.00	0.00	15.81	90.13	230	80	0	2022-02-15 20:41:06	5996.39	180
24043	2	0.00	0.00	15.80	90.06	231	75	0	2022-02-15 20:40:56	5996.39	180
24042	2	0.00	0.00	15.81	90.01	231	78	0	2022-02-15 20:40:46	5996.39	180
24041	2	0.00	0.00	15.81	89.63	230	77	0	2022-02-15 20:40:35	5996.39	180
24040	2	0.00	0.00	15.81	89.51	228	77	0	2022-02-15 20:40:25	5996.39	180
24039	2	0.00	0.00	15.81	89.63	228	74	0	2022-02-15 20:40:15	5996.39	180

Рис.4.2. База даних ІІС

Для роботи ІІС була створена база даних, яка містить натспні поля:

1. Id – номер фільтру для його ідентифікації;
2. Flowspeed – швидкість потоку води;

3. Liters – кількість витрачених літрів;
4. Temperature – температура води у фільтрі;
5. Preassure – тиск у фільтрі;
6. Tds_in – вхіжний показник tds;
7. Tds_out – вихідний показник tds;
8. Leak – протік фільтру;
9. Reading_time – час запису даних з фільтру у базу даних;
10. Ostatok_l – залишок ресурсу в літрах;
11. Ostatok_d – залишок ресурсу в днях.

```
CREATE TABLE SensorData ( id int(6) UNSIGNED NOT NULL, flowspeed varchar(80) COLLATE utf8_unicode_ci DEFAULT NULL, liters varchar(80) COLLATE utf8_unicode_ci DEFAULT NULL, temperature varchar(80) COLLATE utf8_unicode_ci DEFAULT NULL, pressure varchar(80) COLLATE utf8_unicode_ci DEFAULT NULL, tds1 varchar(80) COLLATE utf8_unicode_ci DEFAULT NULL, tds2 varchar(80) COLLATE utf8_unicode_ci DEFAULT NULL, leak varchar(80) COLLATE utf8_unicode_ci DEFAULT NULL, reading_time timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp() ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

Рис.4.3. Лістинг коду запиту до бази даних

За допомоги технології **AJAX** було відправлено дані с бази даних до серверу.

```
$.ajax({

      url: '/index.php',

      method: 'get',

      dataType: 'html',
```

```
      data: {text: 'Текст'

      success: function(data){

            alert(data);

      }

});
```

Рис.4.4. Лістинг коду Ajax запиту

Значення параметрів:

1. **Url** – рядок, що містить URL-адресу, на яку надсилається запит;
2. **Method** – метод HTTP, який використовується для запиту (наприклад, "POST", "GET", "PUT")[37].
3. **DataType** – визначає тип даних, який очікуєте отримати від сервера (скрипт буде виконано навіть якщо не вказати цей параметр)[38].
4. **Data** – дані, які будуть надіслані на сервер. Якщо вони не є рядком, то перетворюються на рядок запиту. Для GET запитів рядок буде додано до URL-адреси. Для того, щоб запобігти автоматичній обробці, ви можете скористатися параметром processData зі значенням false. Якщо дані передаються у складі об'єкта, він повинен складатися з пар ключ/значение. Якщо значення є масивом, то jQuery серіалізує кілька значень з тим самим ключем (залежно від значення параметра traditional, який дозволяє задіяти традиційний тип серіалізації заснований на методі $.param)[39].
5. **Success** – Функція зворотного дзвінка, яка викликається, якщо AJAX запит виконається успішно. Функції передаються три аргументи: data - дані, повернуті з сервера. Дані форматуються відповідно до параметрів dataType, або dataFilter, якщо вони вказані; textStatus - рядок, що описує статус запиту; jqXHR – об'єкт jqXHR (до версії jQuery 1.4.x об'єкт XMLHttpRequest)[40].

Інтелектуальна система

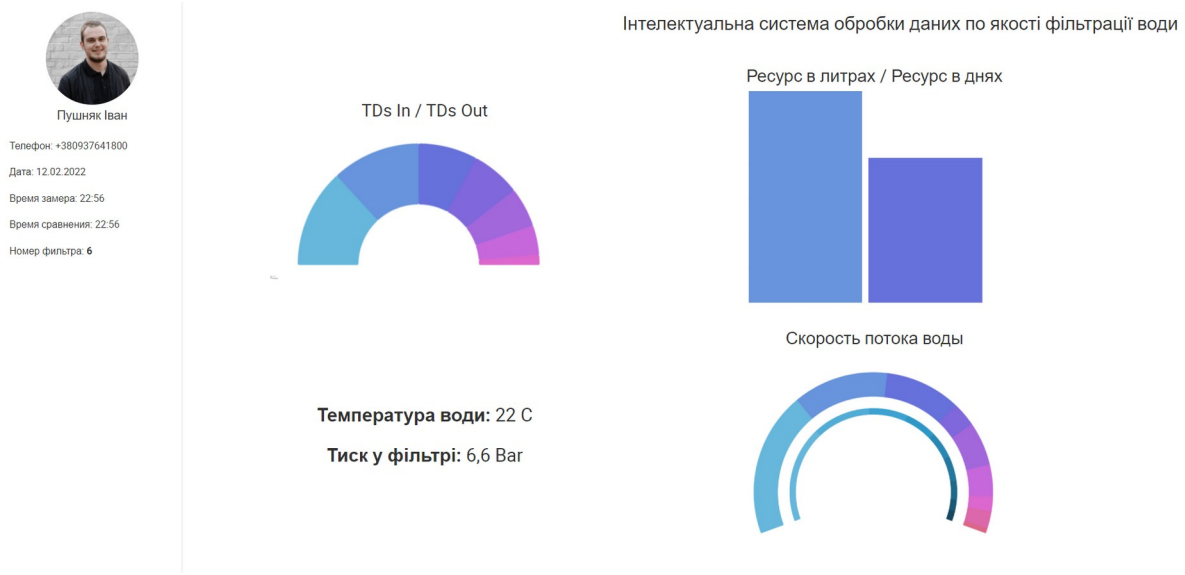

Рис.4.5. Інтелектуальна система

Інтелектуальна система складається з 2 частин:

1. Інформація про клієнта;
2. Графіки та діаграми по фільтру.

Інформація по клієнту складається з:

1. Фото клієнта та ім’я клієнта;
2. Телефон клієнта;
3. Поточна дата;
4. Час заміру та порівняння отриманих даних;
5. Номер фільтру(Артикул).

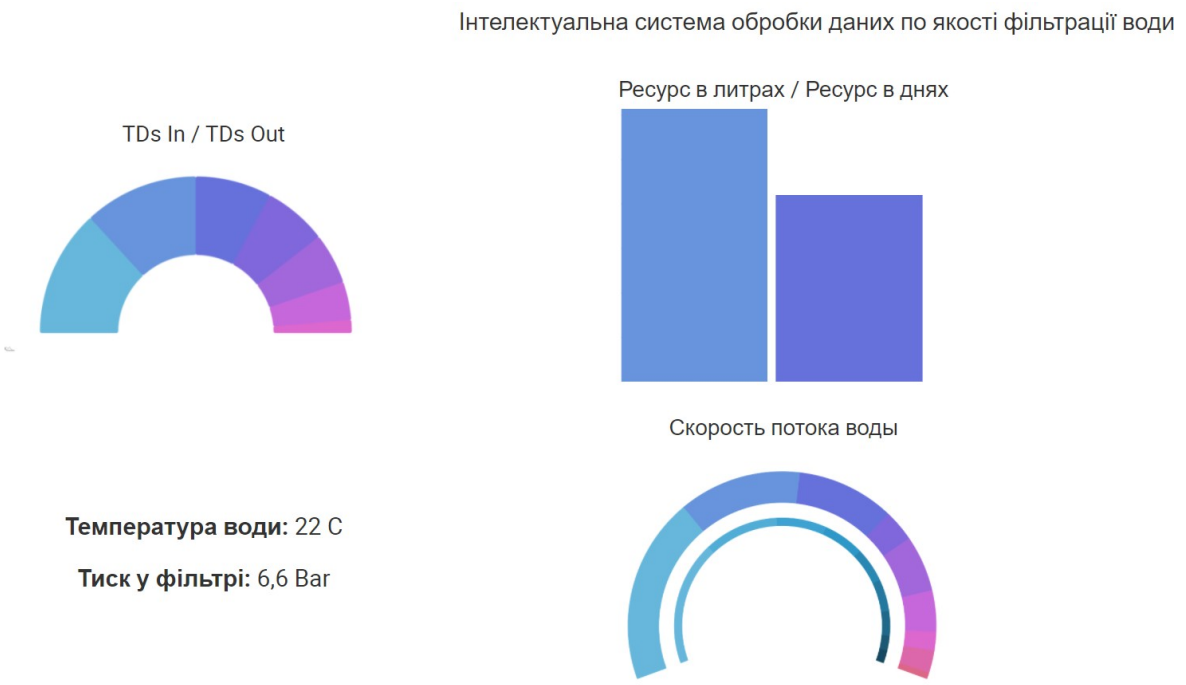

Рис.4.6. Інтелектуальна система – графіки та дафграми по фільтру

Графіки та дафграми по фільтру:

1. Графік вхідного та вихідного TDs;
2. Графік русурсу в літрах та днях;
3. Температура води та тиск у фыльтрі;
4. Швидкість потоку води.

За допомого бібліотеки JS – Chart.js були створені дані графіки.

```
var series0 = chart.series.push(

  am5percent.PieSeries.new(root, {

    valueField: "litres",

    categoryField: "country",

    startAngle: 160,
```

```
    endAngle: 380,

    radius: am5.percent(70),

    innerRadius: am5.percent(65)

  })

);
```

Рис.4.7. Лістинг коду роботи Chart.js

Висновки до розділу 4

У третьому розділи було розглянуто такі основні положення: обґрунтування та вибір базових програмних засобів, обґрунтування та вибір технології розробки програмного забезпечення. Розроблено інтелектуальну систему обробки інформації по якості фільтрації води. ІС складається з двох частин: інформація про клієнта та графіки і діаграми по фільтру.

ВИСНОВКИ

У ході виконання дипломної роботи було виконано такі поставлені задачі:

- Розглянуто основні проблеми, через які падає якість фільтрації води;
- Розглянуто інтелектуальні системи та методи обробки інформації. Обрати найбільш оптимальний для вирішення поставленої задачі;
- Розглянуто найбільш популярні IoT платформи, виявити їх переваги та недоліки;
- Виявлено оптимальні методи та алгоритми обробки даних для досягнення поставленої мети
- Створити власну систему, відштовхуючись від потреб та аналізу досліджених IoT платформи.

Результатом виконання дипломної роботи є створена інтелектуальна система обробки даних по якості фільтрації води.

Розроблена система має наступний функціонал:

1. Збір даних від клієнтів по якості фільтрації води;
2. Створення звітів для подальшого аналізу;
3. Сповіщення клієнтів про неполадки фільтру;
4. Актуальне зображення інформації показників фільтру.

База знань складається з даних по фільтру та механізму висновків, а саме: сповіщення клієнта про протік фільтру, сповіщення клієнта про необхідність заміни картриджу, сповіщення клієнта про сильний тиск у фільтрі, сповіщення клієнта про підвищення різних домішок у воді.

Розроблена система є корисною не лише для користувачів, а й для виробників. Вона допомагає слідкувати за своїми виробами та оперативно реагувати на різні види задач.

Також система є корисною зі сторони маркетингу, так як дозволяє відстежувати ресурси картриджів та своєчасно пропонувати клієнтам їх заміну.

Система є масштабованою, так як дозволяє додавати безліч інших даних клієнтів, дозволяє розширювати функціонал, додавати різні методи обробки інформації, аналізу.

Розроблена методична частина, яка представлена у вигляді лекції та лабораторної роботи по RESP API.

Вивчення й вирішення проблем, пов'язаних із забезпеченням здорових і безпечних умов, у яких відбувається праця людини − одне з найбільш важливих завдань у розробці нових технологій і систем виробництва. Дослідження й виявлення можливих причин виробничих нещасних випадків, професійних захворювань, аварій, вибухів, пожеж, і розробка заходів і вимог, спрямованих на усунення цих причин дозволяють створити безпечні й сприятливі умови для праці людини. Комфортні й безпечні умови праці − один з основних факторів, який впливає на продуктивність і безпеку праці, здоров’я працівників.

Під час роботи над дипломною роботою не було виявлено жодних порушень з питань охорони праці. Робоче місце було оснащене належним чином. Технічний стан обладнання відповідав стандартам безпеки і нормам охорони праці, ніяких дефектів обладнання під час виконання роботи не виявлено.

В результаті написання спеціальної частини з охорони праці було досягнуто поставленої мети, а саме створення безпечних і здорових умов праці на робочих місцях, в робочих зонах, у виробничих приміщеннях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. [Електронний ресурс]. – Режим доступу: URL https://habr.com/ru/post/181772/

2. [Електронний ресурс]. – Режим доступу: URL https://gulpjs.com/

3. [Електронний ресурс]. – Режим доступу: URL https://habr.com/ru/hub/nodejs/

4. [Електронний ресурс]. – Режим доступу: URL https://habr.com/ru/hub/php/

5. [Електронний ресурс]. – Режим доступу: URL https://html5book.ru/html-html5/

6. [Електронний ресурс]. – Режим доступу: URL http://htmlbook.ru/css

7. [Електронний ресурс]. – Режим доступу: URL https://learn.javascript.ru/

8. [Електронний ресурс]. – Режим доступу: URL https://msiter.ru/tutorials/uchebnik-xml-dlya-nachinayushchih/chto-takoe-xml

9. [Електронний ресурс]. – Режим доступу: URL https://sass-scss.ru/

10. Закон України «Про охорону навколишнього природного середовища» –К.: Україна. – 1991. – 59 с. (з усіма редакціями до 2017 року);

11. НПАОП 0.00-1.28-10 Правила охорони праці під час експлуатації електронно-обчислювальних машин/Зареєстровано в Міністерстві юстиції України 19 квітня 2010 р. за №293/17588;

12. Правила улаштування электроустановок. ПУЕ.– Харків.: Форт – 2011 –728 с.;

13. Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості

трудового процесу. Гігієнічні нормативи ГН 3.3.5-8-6.6.1 2002 р. Видання офіційне Київ, 2001 рік – 46 с.

14. ДБН В.2.5-67:2013. Опалення, вентиляція та кондиціювання. – К.: Мінрегіон України, 2013. – 147 с.;

15. ДБН.В.2.5 – 28-2006. Природне і штучне освітлення. – К.: Мінбуд України, - 2008 – 74 с.;

16. НАПБ Б.03.002 – 2007 Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою. Наказ МНС від 03.12.2007 №883;

17. ДСанПін 3.3.2.007– 98 Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. – К.: ГСЕУ України, 1998 – 21 с.;

18. ДБН В.1.1 – 7- 2002. Захист від пожежі. Пожежна безпека об’єктів будівництва. – К.: 2002. – 41 с.;

19. ДСТУ ISO14001 – 97 – 14012-97. Система управления окружающей средой – К.: Держстандарт України – 225 с.;

20. [Електронний ресурс]. – Режим доступу: URL https://www.sciencedirect.com/topics/chemical-engineering/water-filtration

21. [Електронний ресурс]. – Режим доступу: URL https://www.aquacure.co.uk/knowledge-base/how-water-filters-work

22. [Електронний ресурс]. – Режим доступу: URL https://www.thespruceeats.com/best-water-filters-4064605

23. [Електронний ресурс]. – Режим доступу: URL https://www.cdc.gov/healthywater/drinking/home-water-treatment/water-filters/step3.html

24. [Електронний ресурс]. – Режим доступу: URL https://www.aquasana.com/info/water-filter-vs-water-purifier-pd.html

25. [Електронний ресурс]. – Режим доступу: URL https://pidru4niki.com/74257/informatika/intelektualna_informatsiyna_sistema

26. [Електронний ресурс]. – Режим доступу: URL https://www.nam.kiev.ua/files/publications/nester-kovt-fal-2-ostanna.pdf

27. [Електронний ресурс]. – Режим доступу: URL http://nbuv.gov.ua/sites/default/files/all_files/references/201603/vtdo_ro_7.pdf

28. [Електронний ресурс]. – Режим доступу: URL https://ua-referat.com/%D0%86%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%96_%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D1%96_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B8

29. [Електронний ресурс]. – Режим доступу: URL https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

30. [Електронний ресурс]. – Режим доступу: URL https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/

31. [Електронний ресурс]. – Режим доступу: URL https://www.ibm.com/blogs/internet-of-things/what-is-the-iot/

32. [Електронний ресурс]. – Режим доступу: URL https://www.oracle.com/database/what-is-database/

33. [Електронний ресурс]. – Режим доступу: URL https://searchdatamanagement.techtarget.com/definition/database

34. [Електронний ресурс]. – Режим доступу: URL https://www.guru99.com/introduction-to-database-sql.html

35. [Електронний ресурс]. – Режим доступу: URL https://www.britannica.com/technology/database

36. [Електронний ресурс]. – Режим доступу: URL https://www.javatpoint.com/what-is-database

37. [Електронний ресурс]. – Режим доступу: URL https://www.w3schools.com/whatis/whatis_ajax.asp

38. [Електронний ресурс]. – Режим доступу: URL https://www.tutorialspoint.com/ajax/what_is_ajax.htm

39. [Електронний ресурс]. – Режим доступу: URL https://api.jquery.com/jquery.ajax/

40. [Електронний ресурс]. – Режим доступу: URL https://skillcrush.com/blog/what-is-ajax/

ДОДАТОК А

ЛІСТИНГ КОДУ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

```
$data['home'] = $this->url->link('common/home');
$data['wishlist'] = $this->url->link('account/wishlist', '', true);
$data['logged'] = $this->customer->isLogged();
$data['account'] = $this->url->link('account/account', '', true);
$data['register'] = $this->url->link('account/register', '', true)
$data['login'] = $this->url->link('account/login', '', true);
$data['order'] = $this->url->link('account/order', '', true);
$data['transaction'] = $this->url->link('account/transaction', '', true)
$data['download'] = $this->url->link('account/download', '', true);
$data['logout'] = $this->url->link('account/logout', '', true);
$data['shopping_cart'] = $this->url->link('checkout/cart');
$data['checkout'] = $this->url->link('checkout/checkout', '', true);
$data['contact'] = $this->url->link('information/contact');
$data['telephone'] = $this->config->get('config_telephone')
$data['email'] = $this->config->get('config_email');
$data['open'] = nl2br($this->config->get('config_open'))
$this->load->model('design/custommenu');
$this->load->model('catalog/category');
$this->load->model('catalog/product');
$this->load->model('catalog/information');
$data['informations'] = array();
foreach ($this->model_catalog_information->getInformations() as $result) {
   if ($result['bottom']) {
      $data['informations'][] = array(

         'title' => $result['title'],
```

```
            'href'  => $this->url->link('information/information', 'information_id=' .
$result['information_id'])

        );

      }

    }

    $data['categories'] = array();

    if ($this->config->get('configcustommenu_custommenu')) {

    $custommenus = $this->model_design_custommenu->getcustommenus();

    $custommenu_child = $this->model_design_custommenu->getChildcustommenus();

    foreach($custommenus as $id => $custommenu) {

      $children_data = array();

      foreach($custommenu_child as $child_id => $child_custommenu) {

        if (($custommenu['custommenu_id'] != $child_custommenu['custommenu_id']) or !
is_numeric($child_id)) {

          continue;

        }

        $child_name = '';

        if (($custommenu['custommenu_type'] == 'category') and
($child_custommenu['custommenu_type'] == 'category')){

          $filter_data = array(

            'filter_category_id'  => $child_custommenu['link'],

            'filter_sub_category' => true

          );

          $child_name = ($this->config->get('config_product_count') ? ' (' . $this-
>model_catalog_product->getTotalProducts($filter_data) . ')' : '');

        }

        $children_data[] = array(

          'name' => $child_custommenu['name'] . $child_name,


          'href' => $this->getcustommenuLink($custommenu, $child_custommenu)
```

```
            );
        }
        $data['categories'][] = array(
            'name'     => $custommenu['name'] ,
            'children' => $children_data,
            'column'   => $custommenu['columns'] ? $custommenu['columns'] : 1,
            'href'     => $this->getcustommenuLink($custommenu)
        );
    }
    } else {
    $categories = $this->model_catalog_category->getCategories(0);
    foreach ($categories as $category) {
        if ($category['top']) {
            $children_data = array();
            $children = $this->model_catalog_category->getCategories($category['category_id']);
            foreach ($children as $child) {
                $filter_data = array(
                    'filter_category_id'  => $child['category_id'],
                    'filter_sub_category' => true
                );
                $children_data[] = array(
                    'name'  => $child['name'] . ($this->config->get('config_product_count') ? ' (' .
$this->model_catalog_product->getTotalProducts($filter_data) . ')' : ''),
                    'href'  => $this->url->link('product/category', 'path=' . $category['category_id'] . '_'
. $child['category_id'])


                );


            }
            $data['categories'][] = array
```

```
            'name'     => $category['name'],
            'children' => $children_data,
            'column'   => $category['column'] ? $category['column'] : 1,
            'href'     => $this->url->link('product/category', 'path=' . $category['category_id'])
        );
    }
}
$data['language'] = $this->load->controller('common/language');
$data['currency'] = $this->load->controller('common/currency');
if ($this->config->get('configblog_blog_menu')) {
    $data['menu'] = $this->load->controller('blog/menu');
} else {
    $data['menu'] = '';
}
$data['search'] = $this->load->controller('common/search');
$data['cart'] = $this->load->controller('common/cart');
// For page specific css
if (isset($this->request->get['route'])) {
    if (isset($this->request->get['product_id'])) {
        $class = '-' . $this->request->get['product_id'];
    } elseif (isset($this->request->get['path'])) {
        $class = '-' . $this->request->get['path'];
    } elseif (isset($this->request->get['manufacturer_id'])) {
        $class = '-' . $this->request->get['manufacturer_id'];
    } elseif (isset($this->request->get['information_id'])) {
        $class = '-' . $this->request->get['information_id'];
    } else
        $class = '';
    }
```

```
        $data['class'] = str_replace('/', '-', $this->request->get['route']) . $class;
    } else {
        $data['class'] = 'common-home';
    }
    global $js;
    $data['js'] = $js;
    return $this->load->view('common/header', $data);
}
public function getcustommenuLink($parent, $child = null) {
     if ($this->config->get('configcustommenu_custommenu')) {
    $item = empty($child) ? $parent : $child;
    switch ($item['custommenu_type']) {
        case 'category':
            $route = 'product/category';
            if (!empty($child)) {
                $args = 'path=' . $parent['link'] . '_' . $item['link'];
            } else {
                $args = 'path='.$item['link'];
            }
            break;
        case 'product':
            $route = 'product/product';
            $args = 'product_id='.$item['link'];
            break;
        case 'manufacturer':
            $route = 'product/manufacturer/info';
            $args = 'manufacturer_id='.$item['link'];

            break;
```

```
        case 'information':
            $route = 'information/information';
            $args = 'information_id='.$item['link'];
            break;
        default:
            $tmp = explode('&', str_replace('index.php?route=', '', $item['link']));
            if (!empty($tmp)) {
                $route = $tmp[0];
                unset($tmp[0]);
                $args = (!empty($tmp)) ? implode('&', $tmp) : '';
            }
            else {
                $route = $item['link'];
                $args = '';

            break;
    }
    $check = stripos($item['link'], 'http');
   $checkbase = strpos($item['link'], '/');
    if ( $check === 0 || $checkbase === 0 ) {
        $link = $item['link'];
    } else {
        $link = $this->url->link($route, $args);
    }
    return $link;
}
}
```

ДОДАТОК В
ЛІСТИНГ КОДУ ОБРОБКИ ДАНИХ

```
?php
header("Refresh:5");
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
$sql = "SELECT api, flowspeed, liters, temperature, pressure, tds_in, tds_out, leak,
reading_time, ostatok_l, ostatok_d FROM SensorData ORDER BY id DESC";
echo '<table cellspacing="5" cellpadding="5">
     <tr>
       <td>api</td>
       <td>flowspeed(л/мин)</td>
       <td>temperature</td>
       <td>pressure</td>
       <td>tds_in</td>
                  <td>tds_out</td>
                  <td>leak</td>
       <td>Timestamp</td>
                  <td>ostatok_l</td>
                  <td>ostatok_d</td>
     </tr>';
if ($result = $conn->query($sql)) {
   while ($row = $result->fetch_assoc()) {
      $row_api = $row["api"];
      $row_flowspeed = $row["flowspeed"];
      $row_liters = $row["liters"];
```

```
        $row_temperature = $row["temperature"];
        $row_pressure = $row["pressure"];
        $row_tds_in = $row["tds_in"];
                $row_tds_out = $row["tds_out"];
                $row_leak = $row["leak"];
        $row_reading_time = $row["reading_time"];
                $row_ostatok_l = $row["ostatok_l"];
                $row_ostatok_d = $row["ostatok_d"];
        $row_ostatok_lit = number_format($row_ostatok_l, 0, '.', '');
        $row_temperature1 = number_format($row_temperature, 1, '.', '');
        $row_pressure1 = number_format($row_pressure, 1, '.', '');
        echo '<tr>
              <td>' . $row_api . '</td>
              <td>' . $row_flowspeed . '</td>
              <td>' . $row_temperature1 . '</td>
              <td>' . $row_pressure1 . '</td>
              <td>' . $row_tds_in . '</td>
              <td>' . $row_tds_out . '</td>
              <td>' . $row_leak . '</td>
              <td>' . $row_reading_time . '</td>
              <td>' . $row_ostatok_lit . '</td>
              <td>' . $row_ostatok_d . '</td>
            </tr>';
    }
    $result->free();
}
$conn->close();
?>
</table>
```

```
</body>

</html>

if ($_SERVER["REQUEST_METHOD"] == "POST") {

    $api_key = test_input($_POST["api_key"]);

    $flowspeed = test_input($_POST["flowspeed"]);

    $liters = test_input($_POST["liters"]);

    $temperature = test_input($_POST["temperature"]);

    $pressure = test_input($_POST["pressure"]);

    $tds_in = test_input($_POST["tds_in"]);

            $tds_out = test_input($_POST["tds_out"]);

            $leak = test_input($_POST["leak"]);

            $reset = test_input($_POST["reset"]);


    $conn = new mysqli($servername, $username, $password, $dbname);

    if ($conn->connect_error) {

       die("Connection failed: " . $conn->connect_error);

    }

    if ($reset == 0) {

       $sql = "INSERT INTO SensorData (api, flowspeed, liters, temperature, pressure, tds_in,
tds_out, leak, ostatok_l, ostatok_d)

       VALUES (" . $api_key . ", " . $flowspeed . ", " . $liters ., " . $temperature . ", " .
$pressure . ",

            " . $tds_in . ", " . $tds_out . ", " . $leak . ", (SELECT MIN(S.ostatok_l) - " .
$liters . " FROM SensorData AS S WHERE api = " . $api_key . "), (SELECT MIN(P.ostatok_d)
FROM SensorData AS P WHERE api = " . $api_key . "))";

    }

    else {

       $sql = "DELETE FROM SensorData WHERE api = ".$api_key."; INSERT INTO
SensorData (api, flowspeed, liters, temperature, pressure, tds_in, tds_out, leak, ostatok_l,
ostatok_d)

       VALUES (" . $api_key . ", " . $flowspeed . ", 0, " . $temperature . ", " . $pressure . ", " .
$tds_in . ", " . $tds_out . ", " . $leak . ", 6000, 180)";
```

```
        }

            if ($conn->multi_query($sql) == TRUE) {
                echo "New record created successfully";
            }
            else {
                echo "Error pushData: " . $sql . "<br>" . $conn->error;
            }
        $conn->close();

}
else {
    echo "No data posted with HTTP POST.";
}
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
```